Protected Interactive 3D Graphics Via Remote Rendering

David Koller* Michael Turitzin* Marc Levoy* Marco Tarini† Giuseppe Croccia† Paolo Cignoni† Roberto Scopigno†

*Stanford University †ISTI-CNR, Italy

Abstract

Valuable 3D graphical models, such as high-resolution digital scans of cultural heritage objects, may require protection to prevent piracy or misuse, while still allowing for interactive display and manipulation by a widespread audience. We have investigated techniques for protecting 3D graphics content, and we have developed a remote rendering system suitable for sharing archives of 3D models while protecting the 3D geometry from unauthorized extraction. The system consists of a 3D viewer client that includes low-resolution versions of the 3D models, and a rendering server that renders and returns images of high-resolution models according to client requests. The server implements a number of defenses to guard against 3D reconstruction attacks, such as monitoring and limiting request streams, and slightly perturbing and distorting the rendered images. We consider several possible types of reconstruction attacks on such a rendering server, and we examine how these attacks can be defended against without excessively compromising the interactive experience for non-malicious users.

CR Categories: I.3.2 [Computer Graphics]: Graphics Systems—Remote systems

Keywords: security, 3D models, remote rendering, digital rights management

1 Introduction

Protecting digital information from theft and misuse, a subset of the digital rights management problem, has been the subject of much research and many attempted practical solutions. Efforts to protect software, databases, digital images, digital music files, and other content are ubiquitous, and data security is a primary concern in the design of modern computing systems and processes. However, there have been few technological solutions to specifically protect interactive 3D graphics content.

The demand for protecting 3D graphical models is significant. Contemporary 3D digitization technologies allow for the reliable and efficient creation of accurate 3D models of many physical objects, and a number of sizable archives of such objects have been created. The Stanford Digital Michelangelo Project [Levoy et al. 2000], for example, has created a high-resolution digital archive of 10 large statues of Michelangelo, including the David. These statues represent the artistic patrimony of Italy's cultural institutions, and the contract with the Italian authorities permits the distribution of the 3D models only to established scholars for non-commercial use. Though all parties involved would like the models to be widely available for constructive purposes, were the digital 3D model of the David to be distributed in an unprotected fashion, it would soon be pirated, and simulated marble replicas would be manufactured outside the provisions of the parties authorizing the creation of the model.

Digital 3D archives of archaeological artifacts are another example of 3D models often requiring piracy protection. Curators of such artifact collections are increasingly turning to 3D digitization as a way to preserve and widen scholarly usage of their holdings, by allowing virtual display and object examination over the Internet, for example. However, the owners and maintainers of the artifacts often desire to maintain strict control over the use of the 3D data and to guard against theft. An example of such a collection is [Stanford Digital Forma Urbis Project 2004], in which over one thousand fragments of an ancient Roman map were digitized and are being made available through a web-based database, providing that the 3D models can be adequately protected.

Other application areas such as entertainment and online commerce may also require protection for 3D graphics content. 3D character models developed for use in motion pictures are often repurposed for widespread use in video games and promotional materials. Such models represent valuable intellectual property, and solutions for preventing their piracy from these interactive applications would be very useful. In some cases, such as 3D body scans of high profile actors, content developers may be reluctant to distribute the 3D models without sufficient control over reuse. In the area of online commerce, a number of Internet content developers have reported an unwillingness of clients to pursue 3D graphics projects specifically due to the lack of ability to prevent theft of the 3D content [Ressler 2001].

Prior technical research in the area of intellectual property protections for 3D data has primarily concentrated on 3D digital watermarking techniques. Over 30 papers in the last 7 years describe steganographic approaches to embedding hidden information into 3D graphical models, with varying degrees of robustness to attacks that seek to disable watermarks through alterations to the 3D shape or data representation. Many of the most successful 3D watermarking schemes are based on spread-spectrum frequency domain transformations, which embed watermarks at multiple scales by introducing controlled perturbations into the coordinates of the 3D model vertices [Praun et al. 1999; Ohbuchi et al. 2002]. Complementary technologies search collections of 3D models and examine them for the presence of digital watermarks, in an effort to detect piracy.

We believe that for the digital representations of highly valuable 3D objects such as cultural heritage artifacts, it is not sufficient to detect piracy after the fact; we must instead prevent it. The computer industry has experimented with a number of techniques for preventing unauthorized use and copying of computer software and digital data. These techniques have included physical dongles, software access keys, node-locked licensing schemes, copy prevention software, program and data obfuscation, and encryption with embedded keys. Most such schemes are either broken or bypassed by determined attackers, and cause undue inconvenience and expense for non-malicious users. High-profile data and software is particularly susceptible to being quickly targeted by attackers.

Fortunately, 3D graphics data differs from most other forms of digital media in that the presentation format, 2D images, is fundamentally different from the underlying representation (3D geometry). Usually, 3D graphics data is displayed as a projection onto a 2D display device, resulting in tremendous information loss for single views. This property supports an optimistic view that 3D graphics systems can be designed that maintain usability and utility, while not being as vulnerable to piracy as other types of digital content.

In this paper, we address the problem of preventing the piracy of 3D models, while still allowing for their interactive display and manipulation. Specifically, we attempt to provide a solution for maintainers of large collections of high-resolution static 3D models, such as the digitized cultural heritage artifacts described above. The methods we develop aim to protect both the geometric shape of the 3D models, as well as their particular geometric representation, such as the 3D mesh vertex coordinates, surface normals, and connectivity information. We accept that the coarse shape of visible objects can be easily reproduced regardless of our protection efforts, so we concentrate on defending the high-resolution geometric details of 3D models, which may have been most expensive to model or measure (perhaps requiring special access and advanced 3D digitizing technology), and which are most valuable in exhibiting fidelity to the original object.

In the following paper sections, we first examine the graphics pipeline to identify its possible points of attack, and then propose several possible techniques for protecting 3D graphics data from such attacks. Our experimentation with these techniques led us to conclude that remote rendering provides the best solution for protecting 3D graphical models, and we describe the design and implementation of a prototype system in Section 4. Section 5 describes some types of reconstruction attacks against such a remote rendering system and the initial results of our efforts to guard against them.

2 Possible Attacks in the Graphics Pipeline

Figure 1 shows a simple abstraction of the graphics pipeline for purposes of identifying possible attacks to recover 3D geometry. We note several places in the pipeline where attacks may occur:

3D model file reverse-engineering. Fig. 1(a). 3D graphics models are typically distributed to users in data streams such as files in common file formats. One approach to protecting the data is to obfuscate or encrypt the data file. If the user has full access to the data file, such encryptions can be reverse-engineered and broken, and the 3D geometry data is then completely unprotected.

Tampering with the viewing application. Fig. 1(b). A 3D viewer application is typically used to display the 3D model and allow for its manipulation. Techniques such program tracing, memory dumping, and code replacement are practiced by attackers to obtain access to data in use by application programs.

Graphics driver tampering. Fig. 1(c). Because the 3D geometry usually passes through the graphics driver software on its way to the GPU, the driver is vulnerable to tampering. Attackers can replace graphics drivers with malicious or instrumented versions to capture streams of 3D vertex data, for example. Such replacement drivers are widely distributed for purposes of tracing and debugging graphics programs.

Reconstruction from the framebuffer. Fig. 1(d). Because the framebuffer holds the result of the rendered scene, its contents can be used by sophisticated attackers to reconstruct the model geometry, using computer vision 3D reconstruction techniques. The

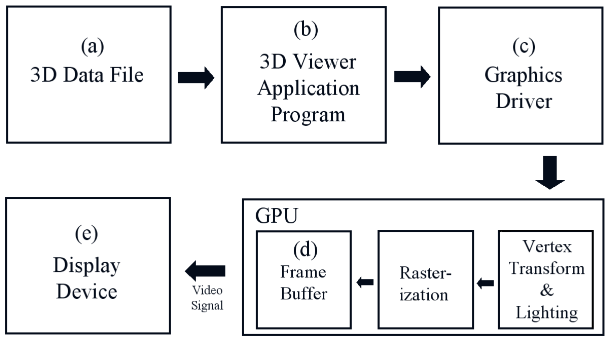


Figure 1: Abstracted graphics pipeline showing possible attack locations (a-e). These attacks are described in the text.

framebuffer contents may even include depth values for each pixel, and attackers may have precise control over the rendering parameters used to create the scene (viewing and projection transformations, lighting, etc.). This potentially creates a perfect opportunity for computer vision reconstruction, as the synthetic model data and controlled parameters do not suffer from the noise, calibration, and imprecision problems that make robust real world vision with real sensors very difficult.

Reconstruction from the final image display. Fig. 1(e). Regardless of whatever protections a graphics system can guarantee throughout the pipeline, the rendered images finally displayed to the user are accessible to attackers. Just as audio signals may be recorded by external devices when sound is played through speakers, the video signals or images displayed on a computer monitor may be recorded with a variety of video devices. The images so gathered may be used as input to computer vision reconstruction attacks such as those possible when the attacker has access to the framebuffer itself, though the images may be of degraded quality, unless a perfect digital video signal (such as DVI) is available.

3 Techniques for Protecting 3D Graphics

In light of the possible attacks in the graphics pipeline as described in the previous section, we have considered a number of approaches for sharing and rendering protected 3D graphics.

Software-only rendering. A 3D graphics viewing system that does not make use of hardware acceleration may be easier to protect from the application programmer's point of view. Displaying graphics with a GPU can require transferring the graphics data in precisely known and open formats, through a graphics driver and hardware path that is often out of the programmer's control. A custom 3D viewing application with software rendering allows the 3D content distributor to encrypt or obfuscate the data in a specific manner, all the way through the graphics pipeline until display.

Hybrid hardware/software rendering. Hybrid hardware and software rendering schemes can be used to take at least some advantage of hardware accelerated rendering, while benefiting from software rendering's protections as described above. In one such scheme, a small but critically important portion of a protected model's geometry (such as the nose of a face) is rendered in software, while the rest of the model is rendered normally with the accelerated GPU hardware. This technique serves as a deterrent to attackers tampering with the graphics drivers or hardware path, but the two-phase drawing with readback of the color and depth buffers can incur a performance hit, and may require special treatment to avoid artifacts on the border of the composition of the two images.

In another hybrid rendering scheme, the 3D geometry is transformed and per-vertex lighting computations are performed in software. The depth values computed for each vertex are distorted in a manner that still preserves the correct relative depth ordering, while concealing the actual model geometry as much as possible. The GPU is then used to complete rendering, performing rasterization, texturing, etc. Such a technique potentially keeps the 3D vertex stream hidden from attackers, but the distortions of the depth buffer values may impair certain graphics operations (fog computation, some shadow techniques), and the geometry may need to be coarsely depth sorted so that Z-interpolation can still be performed in a linear space.

Deformations of the geometry. Small deformations in large 2D images displayed on the Internet are sometimes used as a defense against image theft; zoomed higher resolution sub-images with varying deformations cannot be captured and easily reassembled into a whole. A similar idea can be used with 3D data: subtle 3D deformations are applied to geometry before the vertices are passed to the graphics driver. The deformations are chosen so as to vary smoothly as the view of the model changes, and to prohibit recovery of the original coordinates by averaging the deformations over time. Even if an attacker is able to access the stream of 3D data after it is deformed, they will encounter great difficulty reconstructing a high-resolution version of the whole model due to the distortions that have been introduced.

Hardware decryption in the GPU. One sound approach to providing for protected 3D graphics is to encrypt the 3D model data with public-key encryption at creation time, and then implement custom GPUs that accept encrypted data, and perform on-chip decryption and rendering. Additional system-level protections would need to be implemented to prevent readback of framebuffer and other video memory, and to place potential restrictions on the command stream sent to the GPU, in order to prevent recovery of the 3D data.

Image-based rendering. Since our goal is to protect the 3D geometry of graphic models, one technique is to distribute the models using image-based representations, which do not explicitly include the complete geometry data. Examples of such representations include light fields and Lumigraphs [Levoy and Hanrahan 1996; Gortler et al. 1996], both of which are highly amenable to interactive display.

Remote rendering. A final approach to secure 3D graphics is to retain the 3D model data on a secure server, under the control of the content owner, and pass only 2D rendered images of the models back to client requests. Very low-resolution versions of the models, for which piracy is not a concern, can be distributed with special client programs to allow for interactive performance during manipulation of the 3D model. This method relies on good network bandwidth between the client and server, and may require significant server resources to do the rendering for all client requests, but it is vulnerable primarily only to reconstruction attacks.

Discussion. We have experimented with several of the 3D model protection approaches described above. For example, our first protected 3D model viewer was an encrypted version of the "QSplat" [Rusinkiewicz and Levoy 2000] point-based rendering system, which omits geometric connectivity information. The 3D model files were encrypted using a strong symmetric block cipher scheme, and the decryption key was hidden in a heavily obfuscated 3D model viewer program, using modern program obfuscation techniques [Collberg and Thomborson 2000]. Vertex data was decrypted on demand during rendering, so that only a very small portion of the decrypted model was ever in memory, and only software rendering modes were used.

Unfortunately, systems such as this ultimately rely on "security through obfuscation," which is theoretically unsound from a computer security point of view. Given enough time and resources, an attacker will be able to discover the embedded encryption key or otherwise reverse-engineer the protections on the 3D data. For this reason, any of the 3D graphics protection techniques that make the actual 3D data available to potential attackers in software can be broken [Schneier 2000]. It is possible that future "trusted computing" platforms for general purpose computers will be available that make software tampering difficult or impossible, but such systems are not widely deployed today. Similarly, the idea of a GPU with decryption capability has theoretical merit, but it will be some years before such hardware is widely available for standard PC computing environments, if ever.

Thus, for providing practical, robust, anti-piracy protections for 3D data, we gave strongest consideration to purely image-based representations and to remote rendering. Distributing light fields at the high resolutions necessary would involve huge, unwieldy file sizes, would not allow for any geometric operations on the data (such as surface measurements performed by archaeologists), and would still give attackers unlimited access to the light field for purposes of performing 3D reconstruction attacks using computer vision algorithms. For these reasons, we finally concluded that the last technique, remote rendering, offers the best solution for protecting interactive 3D graphics content.

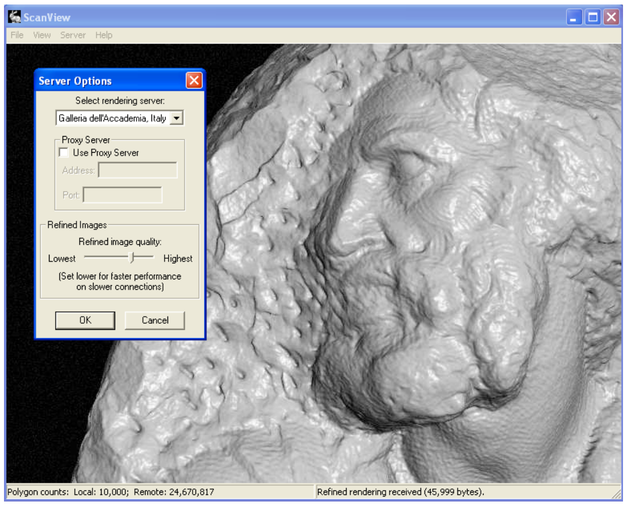
Remote rendering has been used before in networked environments for 3D visualization, although we are not aware of a system specifically designed to use remote rendering for purposes of security and 3D content protection. Remote rendering systems have been previously implemented to take advantage of off-site specialized rendering capabilities not available in client systems, such as intensive volume rendering [Engel et al. 2000], and researchers have developed special algorithmic approaches to support efficient distribution of rendering loads and data transmission between rendering servers and clients [Levoy 1995; Yoon and Neumann 2000]. Remote rendering of 2D graphical content is common for Internet services such as online map sites; only small portions of the whole database are viewed by users at one time, and protection of the entire 2D data corpus from theft via image harvesting may be a factor in the design of these systems.

4 Remote Rendering System

To test our ideas for providing controlled, protected interactive access to collections of 3D graphics models, we have implemented a remote rendering system with a client-server architecture, as described below.

4.1 Client Description

Users of our protected graphics system employ a specially-designed 3D viewing program to interactively view protected 3D content. This client program is implemented as an OpenGL and wxWindows-based 3D viewer, with menus and GUI dialogs to control various viewing and networking parameters (Figure 2). The client program includes very low-resolution, decimated versions of the 3D models, which can be interactively rotated, zoomed, and relit by the user in real-time. When the user stops manipulating the low-resolution model, detected via a "mouse up" event, the client program queries the remote rendering server via the network for a



Select rendering server: Galleria dell'Accademia, Italy

Proxy Server
Use Proxy Server
Address:
Port:

Refined Images
Refined image quality:
Lowest Highest
(Set lower for faster performance on slower connections)

Polygon counts: Local: 10,000; Remote: 24,670,817
Refined rendering received (45,999 bytes).

Figure 2: Screenshot of the client program.

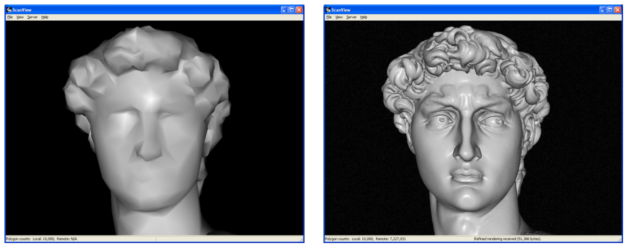
high-resolution rendered image corresponding to the selected rendering parameters. These parameters include the 3D model name, viewpoint position and orientation, and lighting conditions. When the server passes the rendered image back to the client program, it replaces the low-resolution rendering seen by the user (Figure 3).

On computer networks with reasonably low latencies, the user thus has the impression of manipulating a high-resolution version of the model. In typical usage for cultural heritage artifacts, we use models with approximately 10,000 polygons for the low resolution version, whereas the server-side models often contain tens of millions polygons. Such low-resolution model complexities are of little value to potential thieves, yet still provide enough clues for the user to navigate. The client viewer could be further extended to cache the most recent images returned from the server and projectively texture map them onto the low-resolution model as long as they remain valid during subsequent rotation and zooming actions.

4.2 Server Description

The remote rendering server receives rendering requests from users' client programs, renders corresponding images, and passes them back to the clients. The rendering server is implemented as a module running under the Apache 2.0 HTTP Server; as such, the module communicates with client programs using the standard HTTP protocol, and takes advantage of the wide variety of access protection and monitoring tools built into Apache. The rendering server module is based upon the FastCGI Apache module, and allows for multiple rendering processes to be spread across any number of server hardware nodes.

As render requests are received from clients, the rendering server checks their validity and dispatches the valid requests to a GPU for OpenGL hardware-accelerated rendering. The rendered images are read back from the framebuffer, compressed using JPEG compression, and returned to the client. If multiple requests from the same client are pending (such as if the user rapidly changes views while on a slow network), earlier requests are discarded, and only the most recent is rendered. The server uses level-of-detail techniques to speed the rendering of highly complex models, and lower level-of-detail renderings can be used during times of high server load to maintain high throughput rates. In practice, an individual server node with a Pentium 4 CPU and an NVIDIA GeForce4 video card can handle a maximum of 8 typical client requests per second; the



Polygon counts: Local: 10,000; Remote: N/A

Polygon counts: Local: 10,000; Remote: 7,227,031
Refined rendering received (51,386 bytes).

Figure 3: Client-side low resolution (left) and server-side high resolution (right) model renderings.

bottlenecks are in the rendering and readback (about 100 milliseconds), and in the JPEG compression (approximately 25 milliseconds). Incoming request sizes are about 700 bytes each, and the images returned from our deployed servers average 30 kB per request.

4.3 Server Defenses

In Section 2, we enumerated several possible places in the graphics pipeline that an attacker could steal 3D graphics data. The benefit of using remote rendering is that it leaves only 3D reconstruction from 2D images in the framebuffer or display device as possible attacks. General 3D reconstruction from images constitutes a very difficult computer vision problem, as evidenced by the great amount of research effort being expended to design and build robust computer vision systems. However, synthetic 3D graphics renderings can be particularly susceptible to reconstruction because the attacker may be able to exactly specify the parameters used to create the images, there is a low human cost to harvest a large number of images, and synthetic images are potentially perfect, with no sensor noise or miscalibration errors. Thus, it is still necessary to defend the remote rendering system from reconstruction attacks; below, we describe a number of such defenses that we have implemented in combination for our server.

Session-based defenses. Client programs that access the remote rendering system are uniquely identified during the course of a usage session. This allows the server to monitor and track the specific sequence of rendering requests made by each client. Automatic analysis of the server logs allows suspicious request streams to be classified, such as an unusually high number of requests per unit time, or a particular pattern of requests that is indicative of an image harvesting program. High quality computer vision reconstructions often require a large number of images that densely sample the space of possible views, so we are able to effectively identify such access patterns and terminate service to those clients. We can optionally require recurrent user authentication in order to further deter some image harvesting attacks, although a coalition of users mounting a low-rate distributed attack from multiple IP addresses could still defeat such session-based defenses.

Obfuscation. Although we do not rely on obfuscation to protect the 3D model data, we do use obfuscation techniques on the client side of the system to discourage and slow down certain attacks. The low-resolution models that are distributed with the client viewer program are encrypted using an RC4-variant stream cipher, and the keys are embedded in the viewer and heavily obfuscated. The rendering request messages sent from the client to the server are also encrypted with heavily obfuscated keys. These encryptions simply serve as another line of defense; even if they were broken, attackers would still not be able to gain access to the high resolution 3D data except through reconstruction from 2D images.

Limitations on valid rendering requests. As a further defense, we provide the capability in our client and remote server to constrain the viewing conditions. Some models may have particular "stayout" regions defined that disallow certain viewing and lighting angles, thus keeping attackers from being able to reconstruct a complete model. For the particular purpose of defending against the enumeration attacks described in Section 5.1, we put restrictions on the class of projection transformations allowed to be requested by users (requiring a perspective projection with particular fixed field of view and near and far planes), and we prevent viewpoints within a small offset of the model surface.

Perturbations and distortions. Passive 3D computer vision reconstructions of real-world objects from real-world images are usually of relatively poor quality compared to the original object. This failure inspires the belief that we can protect our synthetically rendered models from reconstruction by introducing into the images the same types of obstacles known to plague vision algorithms. The particular perturbations and distortions that we use are described below; we apply these defenses to the images only to the degree that they do not distract the user viewing the models. Additionally, these defenses are applied in a pseudorandomly generated manner for each different rendering request, so that attackers cannot systematically determine and reverse their effects, even if the specific form of the defenses applied is known (such as if the source code for the rendering server is available). Rendering requests with identical parameters are mapped to the same set of perturbations, in order to deter attacks which attempt to defeat these defenses by averaging multiple images obtained under the same viewing conditions.

- **Perturbed viewing parameters** We pseudorandomly introduce subtle perturbations into the view transformation matrix for the images rendered by the server; these perturbations have the effect of slightly rotating, translating, scaling, and shearing the model. The range of these distortions is bounded such that no point in the rendered image is further than either m object space units or n pixels from its corresponding point in an unperturbed view. In practice, we generally set m proportional to the size of the model's geometry being protected, and use values of $n = 15$ pixels, as experience has shown that users can be distracted by larger shifts between consecutively displayed images.

- **Perturbed lighting parameters** We pseudorandomly introduce subtle perturbations into the lighting parameters used to render the images; these perturbations include modifying the lighting direction specified in the client request, as well as addition of randomly changing secondary lighting to illuminate the model. Users are somewhat sensitive to shifts in the overall scene intensity and shading, so the primary light direction perturbations used are generally fairly small (maximum of 10° for typical models, which are rendered using the OpenGL local lighting model).

- **High-frequency noise added to the images** We introduce two types of high-frequency noise artifacts into the rendered images. The first, JPEG artifacts, are a convenient result of the compression scheme applied to the images returned from the server. At high compression levels (we use a maximum libjpeg quality factor of 50), the quantization of DCT coefficients used in JPEG compression creates "blocking" discontinuities in the images, and adds noise in areas of sharp contrast. These artifacts create problems for low-level computer vision image processing algorithms, while the design of JPEG compression specifically seeks to minimize the overall perceptual loss of image quality for human users.

  Additionally, we add pseudorandomly generated monochromatic Gaussian noise to the images, implemented efficiently by blending noise textures during hardware rendering on the server. The added noise defends against computer vision attacks by making background segmentation more difficult, and by breaking up the highly regular shading patterns of the synthetic renderings. Interestingly, users are not generally distracted by the added noise, but have even commented that the rendered models often appear "more realistic" with the high-frequency variations caused by the noise. One drawback of the added noise is that the increased entropy of the images can result in significantly larger compressed file sizes; we address this in part by primarily limiting the application of noise to the non-background regions of the image via stenciled rendering.

- **Low-frequency image distortions** Just as real computer vision lens and sensor systems sometimes suffer from image distortions due to miscalibration, we can effectively simulate and extend these distortions in the rendering server. Subtle non-linear radial distortions, pinching, and low-frequency waves can be efficiently implemented with vertex shaders, or with two-pass rendering of the image as a texture onto a non-uniform mesh, accelerated with the "render to texture" capabilities of modern graphics hardware.

Due to the variety of random perturbations and distortions that are applied to the images returned from the rendering server, there is a risk of distracting the user, as the rendered 3D model exhibits changes from frame to frame, even when the user makes very minor adjustments to the view. However, we have found that the brief switch to the lower resolution model in between display of the high resolution perturbed images, inherent to our remote rendering scheme, very effectively masks these changes. This masking of changes is attributed to the visual perception phenomenon known as *change blindness* [Simons and Levin 1997], in which significant changes occurring in full view are not noticed due to a brief disruption in visual continuity, such as a "flicker" introduced between successive images.

5 Reconstruction Attacks

In this section we consider several classes of attacks, in which sets of images may be gathered from our remote rendering server to make 3D reconstructions of the model, and we analyze their efficacy against the countermeasures we have implemented.

5.1 Enumeration Attacks

The rendering server responds to rendering requests from users specifying the viewing conditions for the rendered images. This ability for precise specification can be exploited by attackers, as they can potentially explore the entire 3D model space, using the returned images to discover the location of the 3D model to any arbitrary precision. In practice, these attacks involve enumerating many small cells in a voxel grid, and testing each such voxel to determine intersection with the remote high-resolution model's surface; thus we term them *enumeration attacks*. Once this enumeration process is complete, occupied cells of the voxel grid are exported as a point cloud and then input to a surface reconstruction algorithm.

In the *plane sweep enumeration attack*, the view frustum is specified as a rectangular, one-voxel-thick "plane," and is swept over the model (Figure 4(a)). Each requested image represents one slice of the model's surface, and each pixel of each image corresponds to a single voxel. A simple comparison of each image pixel against the expected background color is performed to determine whether that

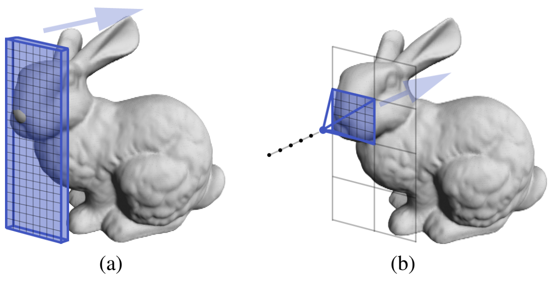


Figure 4: Enumeration Attacks: (a) the plane sweep enumeration attack sweeps a one-voxel thick orthographic view frustum over the model, (b) the near plane sweep enumeration attack sweeps the viewpoint over the model, marking voxels where the model surface is clipped by the near plane.

pixel is a model surface or background pixel. Sweeps from multiple view angles (such as the six faces of the voxels) are done to catch backfacing polygons that may not be visible from a particular angle. These redundant multiple sweeps also allow the attacker to be liberal about ignoring questionable background pixels that may occur, such as if low-amplitude background noise or JPEG compression is being used as a defense on the server.

Our experiments demonstrate that the remote model can be efficiently reconstructed against a defenseless server using this attack (Figure 5(b)). Perturbing viewing parameters can be an effective defense against this attack; the maximum reconstruction resolution will be limited by the maximum relative displacement that an individual model surface point undergoes. Figure 5(c) shows the results of a reconstruction attempt against a server pseudorandomly perturbing the viewing direction by up to 0.3° in the returned images. Since plane sweep enumeration relies on the correspondence between image pixels and voxels, image warps can also be effective as a defense. The large number of remote image requests required for plane sweep enumeration ($O(n)$ requests for an $n \times n \times n$ voxel grid) and the unusual request parameters may look suspicious and trigger the rendering server log analysis monitors. Plane sweep enumeration attacks can be completely nullified by limiting user control of the view frustum parameters, which we implement in our system and use for valuable models.

Another enumeration attack, *near plane sweep enumeration*, involves sweeping the viewpoint (and thus the near plane) over the model, checking when the model surface is clipped by the near plane and marking voxels when this happens (Figure 4(b)). The attacker knows that the near plane has clipped the model when a pixel previously containing the model surface begins to be classified as the background. In order to determine which voxel each image pixel corresponds to, the attacker must know two related parameters: the distance between the viewpoint position and the near plane, and the field of view.

These parameters can be easily discovered. The near plane distance can be determined by first obtaining the exact location of one feature point on the model surface through triangulation of multiple rendering requests and then moving the viewpoint slowly toward that point on the model. When the near plane clips the feature point, the distance between that point and the view position equals the near plane distance. The horizontal and vertical field of view angles can be obtained by moving the viewpoint slowly toward the model surface, stopping when any surface point becomes clipped by the near plane. The viewpoint is then moved a small amount perpendicular to its original direction of motion such that the clipped point moves slightly relative to the view but stays on the new image (near plane). Since the near plane distance has already been obtained, the field of view angle (horizontal or vertical depending on direction of motion) can be obtained from the relative motion of the clipped point across the image.

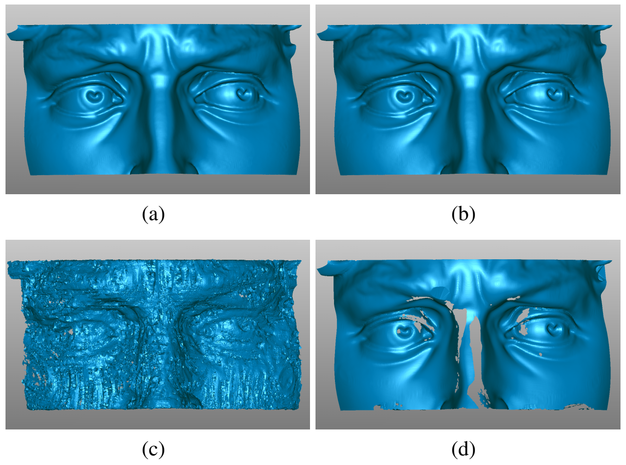


Figure 5: 3D reconstruction results from enumeration attacks: (a) original 3D model, (b) plane sweep attack against defenseless server (6 passes, 3,168 total rendered images), (c) plane sweep attack against 0.3° viewing direction perturbation defense (6 passes, 3,168 total rendered images), (d) near plane sweep attack against defenseless server (6 passes, 7,952 total rendered images).

Because the near plane is usually small compared to the dimensions of the model, many sweeps must be tiled in order to attain full coverage. Sweeps must also be made in several directions to ensure that all model faces are seen. Because this attack relies on seeing the background to determine when the near plane has clipped a surface, concave model geometries will present a problem for surface detection. Although sweeps from multiple directions will help, this problem is not completely avoidable. Figure 5(d) illustrates this problem, showing a case in which six sweeps have not fully captured all the surface geometry.

Viewing parameter perturbations and image warps will nearly destroy the effectiveness of near plane sweep enumeration attacks, as they can make it very difficult to determine where the surface lies and where it does not near silhouette edges (pixels near these edges will change erratically between surface and background). The most solid defense against this attack is to prevent views within a certain small offset of the model surface. This defense, which we use in our system to protect valuable models, prevents the near plane from ever clipping the model surface and thereby completely nullifies this attack.

5.2 Shape-from-silhouette Attacks

Shape-from-silhouette [Slabaugh et al. 2001] is one well studied, robust technique for extracting a 3D model from a set of images. The method consists of segmenting the object pixels from the background in each image, then intersecting in space their resulting extended truncated silhouettes, and finally computing the surface of the resulting shape. The main limitation of this technique is that only a visual hull [Laurentini 1994] of the 3D shape can be recovered; the *line-concave* parts of the model are beyond the capabilities of the reconstruction. Thus, the effectiveness of this attack depends on the specific geometric characteristics of the object; the high-resolution 3D models that we target often have many concavities that are difficult or impossible to fully recover using shape-from-silhouette. However, this attack may also be of use to attackers

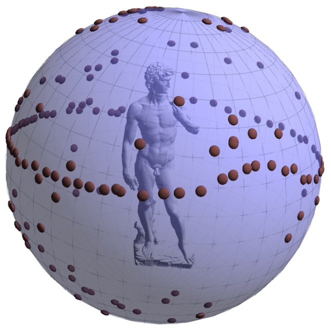


Figure 6: The 160 viewpoints used to reconstruct the model with a shape-from-silhouette attack; results are shown in Figure 7.

to obtain a coarse, low-resolution version of the model, if they are unable to break through the obfuscation protections we use for the low-resolution models distributed with the client.

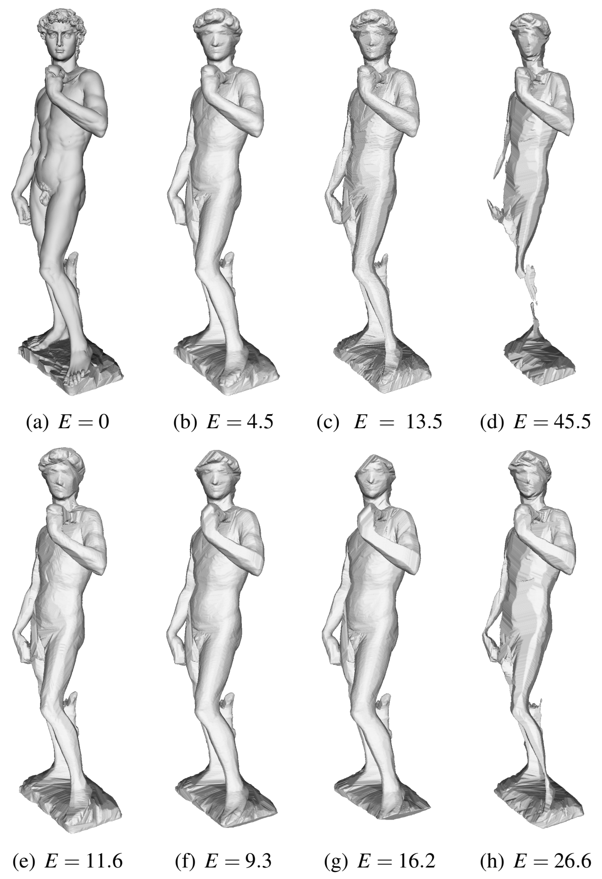
To measure the potential of a shape-from-silhouette attack against our protected graphics system, we have conducted reconstruction experiments on a 3D model of the David as served via the rendering server, using a shape-from-silhouette implementation described in [Tarini et al. 2002]. With all server defenses disabled, 160 images were harvested from a variety of viewpoints around the model (Figure 6); these viewpoints were selected incrementally, with later viewpoints chosen to refine the reconstruction accuracy as measured during the process. The resulting 3D reconstruction is shown in Figure 7(b).

Several of the perturbation and distortion defenses implemented in our server are effective against the shape-from-silhouette attack. Results from experiments showing the reconstructed model quality with server defenses independently enabled are shown in Figures 7(c-g). Small perturbations in the viewing parameters were particularly effective at decreasing the quality of the reconstructed model, as would be expected; Niem [1997] performed an error analysis of silhouette-based modeling techniques and showed the linear relationship between error in the estimation of the view position and error in the resulting reconstruction. Perturbations in the images returned from the server, such as radial distortion and small random shifts, were also effective. Combining the different perturbation defenses, as they are implemented in our remote rendering system, makes for further deterioration of the reconstructed model quality (Figure 7(h)).

High frequency noise and JPEG defenses in the server images can increase the difficulty of segmenting the object from the background. However, shape-from-silhouette software implementations with specially tuned image processing operations can take the noise characteristics into account to help classify pixels accurately. The intersection stage of shape-from-silhouette reconstruction algorithms makes them innately robust with respect to background pixels misclassified as foreground.

5.3 Stereo Correspondence-based Attacks

Stereo reconstruction is another well known 3D computer vision technique. Stereo pairs of similarly neighborhooded pixels are detected, and the position of the corresponding point on the 3D surface is found via the intersection of epipolar lines. Of particular relevance to our remote rendering system, Debevec et al. [1996] showed that the reconstruction task can be made easier and more accurate if an approximate low resolution model is available, by warping the images over it before performing the stereo matching.



(a) $E = 0$ (b) $E = 4.5$ (c) $E = 13.5$ (d) $E = 45.5$

(e) $E = 11.6$ (f) $E = 9.3$ (g) $E = 16.2$ (h) $E = 26.6$

Figure 7: Performance of shape-from-silhouette reconstructions against various server defenses. Error values (E) measure the mean surface distance (mm) from the 5m tall original model. Top row: (a) original model, (b) reconstruction from defenseless server, reconstruction with (c) 0.5° and (d) 2.0° perturbations of the view direction. Bottom row: (e) reconstruction with a random image offset of 4 pixels, with (f) 1.2% and (g) 2.5% radial image distortion, and (h) reconstruction against combined defenses (1.0° view perturbation, 2 pixel random offset, and 1.2% radial image distortion).

Ultimately, however, stereo correspondence techniques usually rely on matching detailed, high-frequency features in order to yield high-resolution reconstruction results. The smoothly shaded 3D computer models generated by laser scanning that we share via our remote rendering system thus present significant problems to basic two-frame stereo matching algorithms. When we add in the server defenses such as image-space high frequency noise, and slight perturbations in the viewing and lighting parameters, the stereo matching task becomes even more ill-posed. Other stereo research such as [Scharstein and Szeliski 2002] also reports great difficulty in stereo reconstruction of noise-contaminated, low-texture synthetic scenes. Were we to distribute 3D models with high resolution textures applied to their surfaces, stereo correspondence methods may be a more effective attack.

5.4 Shape-from-shading Attacks

Shape-from-shading attacks represent another family of computer vision techniques for reconstructing the shape of a 3D object (see [Zhang et al. 1999] for a survey). The primary attack on our remote rendering system that we consider in this class involves first

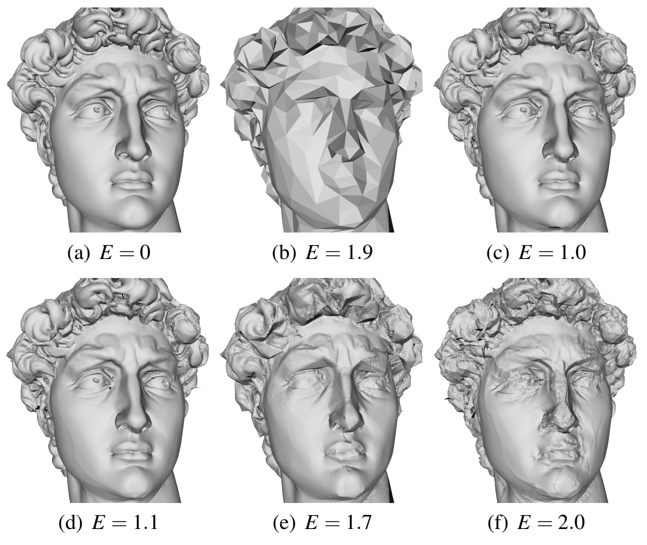


Figure 8: Performance of shape-from-shading reconstruction attacks. Error values (E) measure the mean surface distance (mm) from the original model. Top row: (a) original model, (b) low-resolution base mesh, (c) reconstruction from defenseless server. Bottom row: reconstruction results against (d) high-frequency image noise, (e) complicated lighting model (3 lights), and (f) viewing angle perturbation (up to 1.0°) defenses.

obtaining several images from the same viewpoint under varying, known lighting conditions. Then, using photometric stereo methods, a normal is computed for each pixel by solving a system of rendering equations. The resulting normal map can be registered and applied to an available approximate 3D geometry, such as the low-resolution model used by the client, or one obtained from another reconstruction technique such as shape-from-silhouette.

This coarse normal-mapped model itself may be of value to some attackers: when rendered it will show convincing 3D high frequency details that can be shaded under new lighting conditions, though with artifacts at silhouettes. However, the primary purpose of our system is to protect the high-resolution 3D geometry, which if stolen could be used maliciously for shape analysis or to create replicas. Thus, a greater risk is posed if the normal map is integrated by the attacker to compute a displacement map, and the results are used to displace a refined version of the low-resolution model mesh.

Following this procedure with images harvested from a defenseless remote rendering server and using a low-resolution client model, we were able to successfully reconstruct a high-resolution 3D model. The results shown in Figure 8(c) depict a reconstruction of the David's head produced from 200 1600x1114 pixel images taken from 10 viewpoints, with 20 lighting positions used at each viewpoint, assuming a known, single-illuminant OpenGL lighting model and using a 10,000 polygon low-resolution model (Fig. 8(b)) of the whole statue.

Some of the rendering server defenses, such as adding high-frequency noise to the images, can be compensated for by attackers by simply adding enough input images to increase the robustness of the photometric stereo solution step (although harvesting too many images will eventually trigger the rendering server log analysis monitors). Figure 8(d) shows the high quality reconstruction result possible when only random Gaussian noise is used as a defense. More effective defenses against shape-from-shading attacks include viewing and lighting perturbations and low-frequency image distortions, which can make it difficult to precisely register images onto the low-resolution model, and can disrupt the photometric stereo solution step without a large number of aligned input images. Figure 8(e) shows a diminished quality reconstruction when the rendering server complicates the lighting model by using 3 perturbed light sources with a Phong component unknown to the attacker, and Figure 8(f) shows the significant loss of geometric detail in the reconstruction when the server randomly perturbs the viewing direction by up to 1.0° (note that the reconstruction error exceeds that of the starting base mesh).

The quality of the base mesh is an important determinant in the success of this particular attack. For example, repeating the experiment of Figure 8 with a more accurate base mesh of 30,000 polygons yields results of $E = 0.8$, $E = 0.6$, and $E = 0.7$ for the conditions of Figures 8(b), 8(c), and 8(e), respectively. This reliance on an accurate low-resolution base mesh for the 3D model reconstruction is a potential weak point of the attack; attackers may be deterred by the effort required to reverse-engineer the protections guarding the low-resolution model or to reconstruct an acceptable base mesh from harvested images using another technique.

5.5 Discussion

Because we know of no single mechanism for guaranteeing the security of 3D content delivered through a rendering server, we have instead taken a systems-based approach, implementing multiple defenses and using them in combination. Moreover, we know of no formalism for rigorously analyzing the security provided by our defenses; the reconstruction attacks that we have empirically considered here are merely representative of the possible threats.

Of the reconstruction attacks we have experimented with so far, the shape-from-shading approach has yielded the best results against our defended rendering server. Enumeration attacks are easily foiled when the user's control over the viewpoint and view frustum is constrained, pure shape-from-silhouette methods are limited to reconstructing a visual hull, and two-frame stereo algorithms rely on determining accurate correspondences which is difficult with the synthetic, untextured models we are attempting to protect. Attackers could improve the results of the shape-from-shading algorithm against our perturbation defenses by explicitly modeling the distortions and trying to take them into account in the optimization step, or alternatively by attempting to align the images by interactively establishing point to point correspondences or using an automatic technique such as [Lensch et al. 2001].

Such procedures for explicitly modeling the server defenses, or correcting for them via manual specification of correspondences, are applicable to any style of reconstruction attempt. To combat these attacks, we must rely on the combined discouraging effect of multiple defenses running simultaneously, which increases the number of degrees of freedom of perturbation to a level that would be difficult and time-consuming to overcome. Some of our rendering server defenses, such as the lighting model and non-linear image distortions, can be increased arbitrarily in their complexity. Likewise, the magnitude of server defense perturbations can be increased with a corresponding decrease in the fidelity of the rendered images.

Ultimately, no fixed set of defenses is bulletproof against a sophisticated, malicious attacker with enough resources at their disposal, and one is inevitably led to an "arms race" between attacks and countermeasures such as we have implemented. As the expense required to overcome our remote rendering server defenses becomes greater, determined attackers may instead turn to reaching their piracy goals via non-reconstruction-based methods beyond the scope of this paper, such as computer network intrusion or exploitation of non-technical human factors.

6 Results and Future Work

A prototype of our remote rendering system (*ScanView*, available at *http://graphics.stanford.edu/software/scanview/*) has been deployed to share 3D models from a major cultural heritage archive, the Digital Michelangelo Project [Levoy et al. 2000], as well as other collections of archaeological artifacts that require protected usage. In the several months since becoming publically available, more than 4,000 users have installed the client program on their personal computers and accessed the remote servers to view the protected 3D models. The users have included art students, art scholars, art enthusiasts, and sculptors examining high-resolution artworks, as well as archaeologists examining particular artifacts. Few of these individuals would have qualified under the strict guidelines required to obtain completely unrestricted access to the models, so the protected remote rendering system has enabled large, entirely new groups of users access to 3D graphical models for professional study and enjoyment.

Reports from users of the system have been uniformly positive and enthusiastic. Fetching high-resolution renderings over intercontinental broadband Internet connections takes less than 2 seconds of latency, while fast continental connections generally experience latencies dominated by the rendering server's processing time (around 150 ms). The rendering server architecture can scale up to support an arbitrary number of requests per second by adding additional CPU and GPU nodes, and rendering servers can be installed at distributed locations around the world to reduce intercontinental latencies if desired.

Our log analysis defenses have detected multiple episodes of system users attempting to harvest large sets of images from the server for purposes of later 3D reconstruction attempts, though these incidents were determined to be non-malicious. In general, the monitoring capabilities of a remote rendering server are useful for reasons beyond just security, as the server logs provide complete accounts of all usage of the 3D models in the archive, which can be valuable information for archive managers to gauge popularity of individual models and understand user interaction patterns.

Our plans for future work include further investigation of computer vision techniques that address 3D reconstruction of synthetic data under antagonistic conditions, and analysis of their efficacy against the various rendering server defenses. More sophisticated extensions to the basic vision approaches described above, such as multi-view stereo algorithms, and robust hybrid vision algorithms which combine the strengths of different reconstruction techniques, can present difficult challenges to protecting the models. Another direction of research is to consider how to allow users a greater degree of geometric analysis of the protected 3D models without further exposing the data to theft; scholarly and professional users have expressed interest in measuring distances and plotting profiles of 3D objects for analytical purposes beyond the simple 3D viewing supported in the current system. Finally, we are continuing to investigate alternative approaches to protecting 3D graphics, designing specialized systems which make data security a priority while potentially sacrificing some general purpose computing platform capabilities. The GPU decryption scheme described herein, for example, is one such idea that may be appropriate for console devices and other custom graphics systems.

Acknowledgements We thank Kurt Akeley, Sean Anderson, Jonathan Berger, Dan Boneh, Ian Buck, James Davis, Pat Hanrahan, Hughes Hoppe, David Kirk, Matthew Papakipos, Nick Triantos, and the anonymous reviewers for their useful feedback, and Szymon Rusinkiewicz for sharing code. This work has been supported in part by NSF contract IIS–0113427, the Max Planck Center for Visual Computing and Communication, and the EU IST-2001-32641 ViHAP3D Project.

References

COLLBERG, C., AND THOMBORSON, C. 2000. Watermarking, tamper-proofing, and obfuscation: Tools for software protection. Tech. Rep. 170, Dept. of Computer Science, The University of Auckland.

DEBEVEC, P., TAYLOR, C., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proc. of ACM SIGGRAPH 96*, 11–20.

ENGEL, K., HASTREITER, P., TOMANDL, B., EBERHARDT, K., AND ERTL, T. 2000. Combining local and remote visualization techniques for interactive volume rendering in medical applications. In *Proc. of IEEE Visualization 2000*, 449–452.

GORTLER, S., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proc. of ACM SIGGRAPH 96*, 43–54.

LAURENTINI, A. 1994. The visual hull concept for silhouette-based image understanding. *IEEE Trans. on Pattern Analysis and Machine Intelligence 16*, 2, 150–162.

LENSCH, H. P., HEIDRICH, W., AND SEIDEL, H.-P. 2001. A silhouette-based algorithm for texture registration and stitching. *Graphical Models 63*, 245–262.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *Proc. of ACM SIGGRAPH 96*, 31–42.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital michelangelo project. In *Proc. of ACM SIGGRAPH 2000*, 131–144.

LEVOY, M. 1995. Polygon-assisted jpeg and mpeg compression of synthetic images. In *Proc. of ACM SIGGRAPH 95*, 21–28.

NIEM, W. 1997. Error analysis for silhouette-based 3d shape estimation from multiple views. In *International Workshop on Synthetic-Natural Hybrid Coding and 3D Imaging*.

OHBUCHI, R., MUKAIYAMA, A., AND TAKAHASHI, S. 2002. A frequency-domain approach to watermarking 3d shapes. *Computer Graphics Forum 21*, 3.

PRAUN, E., HOPPE, H., AND FINKELSTEIN, A. 1999. Robust mesh watermarking. In *Proc. of ACM SIGGRAPH 99*, 49–56.

RESSLER, S., 2001. Web3d security discussion. Online article: *http://web3d.about.com/library/weekly/aa013101a.htm*.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. QSplat: A multiresolution point rendering system for large meshes. In *Proc. of ACM SIGGRAPH 2000*, 343–352.

SCHARSTEIN, D., AND SZELISKI, R. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision 47*, 1–3, 7–42.

SCHNEIER, B. 2000. The fallacy of trusted client software. *Information Security* (August).

SIMONS, D., AND LEVIN, D. 1997. Change blindness. *Trends in Cognitive Sciences 1*, 7, 261–267.

SLABAUGH, G., CULBERTSON, B., MALZBENDER, T., AND SCHAFER, R. 2001. A survey of methods for volumetric scene reconstruction from photographs. In *Proc. of the Joint IEEE TCVG and Eurographics Workshop (VolumeGraphics-01)*, Springer-Verlag, 81–100.

STANFORD DIGITAL FORMA URBIS PROJECT, 2004. http://formaurbis.stanford.edu.

TARINI, M., CALLIERI, M., MONTANI, C., ROCCHINI, C., OLSSON, K., AND PERSSON, T. 2002. Marching intersections: An efficient approach to shape-from-silhouette. In *Proceedings of the Conference on Vision, Modeling, and Visualization (VMV 2002)*, 255–262.

YOON, I., AND NEUMANN, U. 2000. Web-based remote rendering with IBRAC. *Computer Graphics Forum 19*, 3.

ZHANG, R., TSAI, P.-S., CRYER, J. E., AND SHAH, M. 1999. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 21*, 8, 690–706.