VISUALIZING AND ANALYZING LARGE AND DETAILED 3D DATASETS

Louis Borgeat, Guy Godin, François Blais, J-Angelo Beraldin, Philippe Massicotte and Guillaume Poirier

Visual Information Technology Group
Institute for Information Technology
National Research Council of Canada
Ottawa, Ontario, Canada
first.last@nrc-cnrc.gc.ca

KEY WORDS: Multi-resolution modeling, interactive visualization, GPU programming, large datasets, texture mapping, foveal displays, tele-collaboration.

ABSTRACT:

This paper presents a set of tools developed to model, visualize and analyze large 3D datasets built from 2D and 3D sensor data. These tools, grouped under the Atelier3D framework, first include a technique for processing and interactively visualizing datasets made of hundreds of millions 3D samples and tens of gigabytes of texture from digital photography. It also includes various analysis tools to visualize and transform the data, including non-photorealistic rendering techniques implemented on the GPU that provide the capabilities to extract information from the datasets not always visible using classical shading techniques. Atelier3D also contains advanced display and interaction functionalities to support multi-projector and multi-resolution tele-collaborative hardware configurations.

1 INTRODUCTION

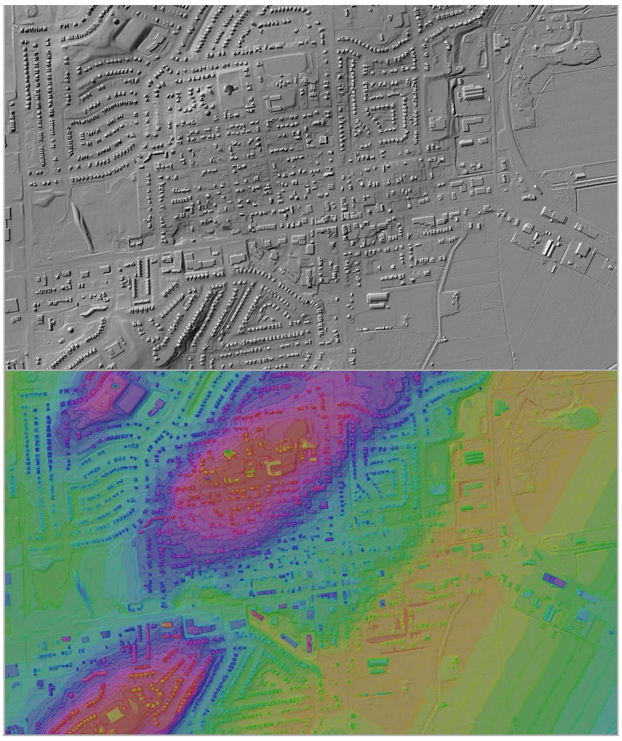


Figure 1: Two representations of a dataset obtained from LiDAR range data. Top: a regular OpenGL shading. Bottom: a combination of contour-line-based shading and of regular shading. This last representation was chosen to provide a global impression of terrain elevation relative to the height of the buildings.

Cultural heritage and conservation applications have provided a stimulating testing ground for many developments in 3D acquisition, modeling, and display over the last decades (Godin et al., 2002, Levoy et al., 2000). This is in part because this field provides a wide variety of very detailed objects of all size and shape and is therefore a source of limitless challenge for new technology development. As of today, sensors can acquire up to billions of sample points and tens of gigabytes of texture in a single day of data acquisition. But as the resulting models increase in size and improve in detail and quality, there is a growing need for the development of more advanced tools to visualize and analyze those datasets to be able to access all the valuable knowledge they contain: data must not only be displayable at various scales under various representations without creating any misleading artefacts, but it also increasingly needs to be transformed to enhance understanding by the viewer. In this paper we present the Atelier3D framework, a prototype application composed of a set of tools developed over the years to fulfill the visualization and analysis needs of different application projects, in good part in the field of cultural heritage. We first describe a technique for processing and interactively visualizing datasets made of hundreds of millions 3D samples and tens of gigabytes of texture from digital photography. We then present tools to analyze the data, including non-photorealistic rendering and filtering techniques implemented in real-time on the GPU(Graphics Processing Unit) to be able to extract information from the datasets not always visible using typical shading techniques. We also describe advanced display and interaction functionalities to support multi-projector and multi-resolution tele-collaborative hardware configurations. We will finally present renderings of different models produced using the Atelier3D framework, illustrating how the capacity to interactively switch between different representations, many already well known, allows for the extraction of much more information from the datasets, and in many cases, information simply not visible using traditional rendering techniques. Figure 1 gives a simple example of a representation adapted from contour lines for aerial 3D data.

2 RELATED WORK

Solutions aimed at interactively displaying models that are too large for brute force rendering on graphics systems are well established in computer graphics (Clark, 1976, Funkhouser and Séquin, 1993). But these solutions must be constantly renewed as the graphics hardware evolves and the datasets grow in size and complexity. In the case of sensor-based datasets, models usually have a relatively simple structure, so visibility culling of the data is rarely the main issue. However, such models can easily be composed of hundreds of millions polygons and contain tens of gigabytes of texture data, far more than even a modern graphics adaptor can handle. The main challenge is therefore to be able to adapt in real-time the resolution of the rendered data to the resolution of the display, making it small enough for interactive display, yet high resolution enough to produce a quality image from the current viewpoint. The first step is to choose some criteria to perform the actual simplification of the model. The most popular approaches are based on quadric error metrics(Garland and Heckbert, 1998, Hoppe, 1999). Techniques that allow for a very fine grained real-time adaptation of the surface based on the viewpoint were popular a few years ago(Hoppe, 1997). But as the power of graphics adaptors grew rapidly, coarser methods that operate on larger geometry blocks were proposed. They required less CPU and made a closer to optimal use of graphic resources (Borgeat et al., 2005, Cignoni et al., 2004, Niski et al., 2007). Methods that render hierarchies of points instead of meshes have also been proposed(Rusinkiewicz and Levoy, 2000), but the performance benefits are less significant when a high quality rendering is targeted(Zwicker et al., 2004). Techniques to properly order vertices in meshes have also made these solutions less advantageous(Yoon et al., 2005). For the now frequent cases where the model is too large to fit even in system memory, many solutions have also been proposed. We refer you to (Borgeat et al., 2005) for an example and a more complete review of the field. Less work has been published that presents actual analysis results obtained from non-photorealistic representations of 3D datasets. A good example would be (Anderson and Levoy, 2002), where scanned cuneiform tablets are processed using curvature and accessibility coloring to make them easily readable.

3 LARGE DATASETS MANAGEMENT

We have developed a general technique to interactively display large scanned surface datasets(Borgeat et al., 2005) that can easily handle models composed from hundreds of millions of polygons and tens of gigabytes of associated texture data. The method is an extension of view-dependent hierarchical levels of detail (LOD), where we use geomorphing to ensure temporal and spatial continuity between the different levels of detail and across the entire model.

This rendering technique combines several important advantages:

- By using static pre-optimized geometry units as the primitives for the view-dependent computations, we strike a better balance between GPU and CPU usage, and can benefit from on-GPU geometry and texture caching and from better use of the GPU vertex cache. On a model such as the Mona Lisa of Figure 10, we can render well over 100 millions geomorphed polygons per second when rendering with the regular OpenGL lighting, thus allowing to render the multi-resolution models with sub-pixel accuracy at interactive frame rates.

- We also minimize the visual artefacts associated with the view-dependent transformation of the displayed data through the use of geomorphing. One of the known inconvenients of LOD-based approaches are the popping transitional artefacts associated with changes in the resolution level. When rendering models at high resolution, no artifacts are visible even if we set a target resolution slightly over one pixel, but we can also get very good results on various models when rendering at a lower geometrical resolution if resources are limited.

- This technique requires very little CPU usage. All the geomorphing being performed on the GPU, CPU is only required to cull the coarse hierarchical level-of-detail structure as with any scene graph, and to pre-fetch data during navigation. Resources are therefore available for other tasks such as A/V transcoding for tele-collaboration (see Sect. 4.3).

- The method is well adapted to models that contain massive amounts of texture. The pre-processing can in these cases be optimized to adapt the created hierarchical structure to minimize the amount and size of texture units.

- Finally, one of the key aspects of this technique is that we display the original model when the user is sufficiently close to the model to require its full resolution to produce an image. This is very important since we want to be able to differentiate between real features present in the data and artifacts caused by the processing and rendering steps associated with the large dataset management issues. This is absolutely crucial for many analytical application both in heritage and other application fields.

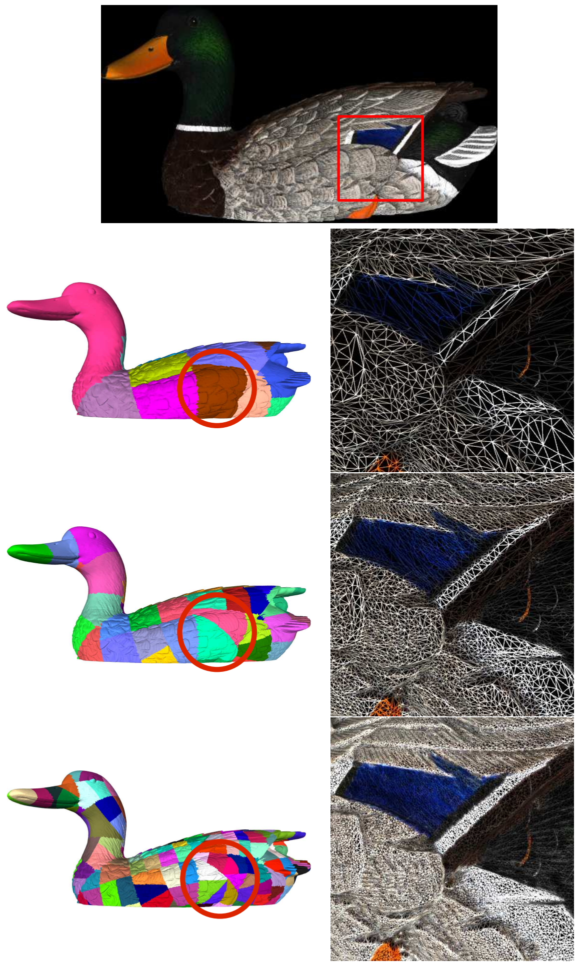


Figure 2: Illustration of the data structure for the interactive rendering. Left: Recursive subdivision process in the level of detail hierarchy. Right: corresponding geometry for each level of detail for a selected area of a wooden duck model.

3.1 method overview

Before we can interactively display a model with Atelier3D, we must first precompute the appropriate multi-resolution data structure. This process will take a few hours for a model of the size of the Mona Lisa (333 millions polygons) on a recent workstation. During this pre-processing, the triangular mesh model is basically converted into a compressed level-of-detail hierarchy of optimized geometry patches and associated texture or attribute data. The first step of this preprocessing is to simplify the entire model into a sequence of discrete LODs using an algorithm based on vertex aggregation. This choice of simplification technique is important since the aggregation paths become the geomorphing paths at rendering time. Figure 2 shows different levels of simplification for an example 3D model. The lowest resolution LOD is then decomposed into a set of triangle patches. The next higher resolution level is then partitioned along the same boundaries, and each group in this level is sub-partitioned until the desired granularity for the view-dependent computations is reached. This process is applied in sequence to all levels of the model, resulting in a hierarchy of group subdivisions spanning the whole sequence of LODs. Groups can be shaped according to criteria such as compactness, common orientation, texture/viewpoint association, culling requirements, model structure, existing partitions, and number of primitives per group. These criteria change depending on the type of model to be displayed: for example, for a model with much more texture than geometry, we will want to create patches that segment images in texture space in order to minimize the size and number of texture units. While for a color-per-vertex model we will simply make patches that are more compact to minimize errors in the view dependent computations. Groups are finally individually converted into triangle strips optimized for the GPU vertex-cache in order to maximize rendering speed. These groups or patches constitute the basic units for all the view-dependent computations.

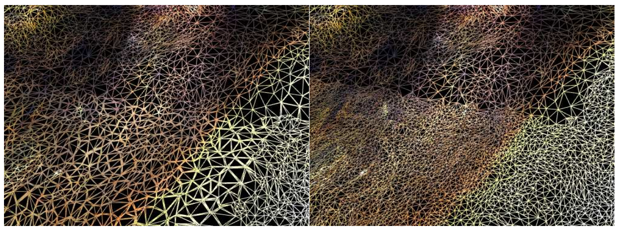


Figure 3: View-dependent mesh for a given frame with(left) and without(right) geomorphing.

At run time, a front in the LOD patch hierarchy is selected during the culling process just as with traditional hierarchical level-of-detail structures. Groups are selected based on a determined maximum screen pixel error and on a unique worst-case precomputed error measurement associated with each group. Geomorphing ratios between selected groups and their lower resolution parent are computed for each frame. Seamless continuity between neighboring LOD groups is maintained at all time by geomorphing boundary points between groups according to a specific set of rules while the other points are morphed using a uniform ratio per individual patch. The geomorphing is actually performed by a vertex program on the graphics processing unit (GPU), so it requires almost no CPU resources. Geomorphing is applied to space and texture coordinates, normals and colors. Out-of-core pre-fetching is done asynchronously on a different thread to insure smooth navigation and interaction. Groups are assigned for pre-fetching during culling by selecting another front in the LOD hierarchy, this time using a lower pixel-size error. Using geomorphing maintains a more uniform polygon size during navigation, as seen in Figure 3.

In the previously described method(Borgeat et al., 2005), per-vertex geomorphing ratios were computed on the CPU and uploaded to the GPU for interpolation. The method has been improved significantly and now only a few ratios have to be computed for each patch and uploaded to the GPU for interpolation. Basically, each unique interpolation ratio shared by a group of border points of the patch is computed only once. This is possible since points who share the same set of neighboring patches by definition also share the same unique geomorphing ratio. The amount of CPU processing for this technique is therefore almost identical to the load of culling the equivalent static LOD scene graph structure. This allows for the creation of a more fine-grained tree structure, making frustum culling more efficient and minimizing the selection of uselessly high resolution data associated with more coarse-grained multi-resolution techniques. The geometric data is also now uploaded to the GPU in a simple compressed format. This new optimization actually reduces the amount of data to upload to the GPU by a factor of 2 for color-per-vertex models.

3.2 texture management

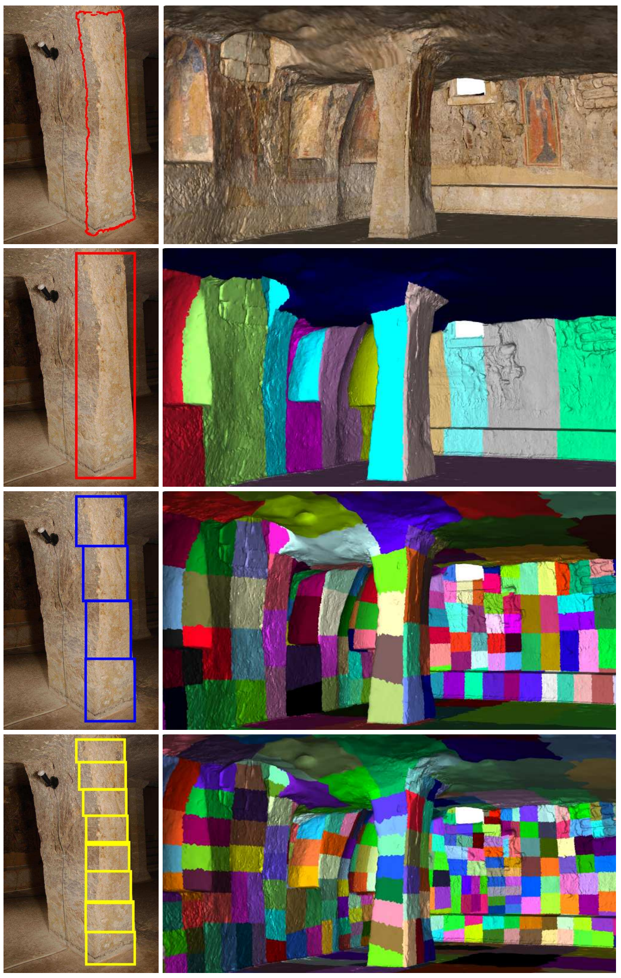


Figure 4: Construction of the level of detail structure in the context of high resolution textured models. Recursive segmentation is performed in texture space instead of in geometric space to minimize the size of the final texture units set for the paging process.

In the cases where dense 3D information is derived from photographic data (El-Hakim, 2006), the resolution of texture and geometry are by definition the same. But in more typical cases where digital photographs are mapped onto data acquired with a 3D sensor(Beraldin et al., 2002), we are usually faced with much more texture than 3D data. In those contexts, the performance of the interactive rendering process will be mostly driven by the amount of texture that will need to be paged between the disk, the system memory and the GPU, and by the amount of GPU memory needed to store all the texture associated with the displayed geometry. We must therefore optimize the segmentation process used to create the level of detail hierarchy to minimize the size and amount of texture units generated. In those cases, instead of subdividing the geometry in Euclidian 3D space, we subdivide it in the parameterization texture space associated with each texture image, as shown in Figure 4. We assume as a starting point that we already have an initial texture-geometry association, and that we have a projection space obtained through photogrammetry. We then apply a heuristic that recursively subdivides textures to maximize the following set of criteria: minimal texture area, squareness of units, and size uniformity. Minimal constraints on the resulting shape of the corresponding 3D geometry units are also applied. Such an approach will end up producing more patches for a given target 3D size for the patches, but will minimize the amount of texture to page and store in real-time, therefore maximizing the real-time performance and minimizing the global file size.

The initial association of geometry and texture must be done sequentially, but all the subsequent processing of the image data can be done in parallel on a per-image basis, making this technique highly scalable. Depending on the application, displayed texture resolution can be set to a different target than resolution of the geometry. We might want for example to display geometry at lower resolution to save resources but to have high resolution textures at all time. This association has to be specified during the pre-processing phase.

4 INTERACTIVE ANALYSIS

One key feature of Atelier3D is the ability to analyze the model using a wide range of data representations. In that regard, the intent of Atelier3D is to provide conservators, art historians and archaeologists with a set of intuitive tools and filters to be able to extract all available information from a large 3D dataset. Ultimately, the objective is to have something analogous to what is available in 2D image processing environments such as Photoshop or the Gimp but adapted to the 3D context. In this section we will describe a few of those tools and representations, illustrating how they are implemented on the GPU and what are the actual performance costs associated with them in the context of large dataset visualization.

Modern GPUs provide significantly more control on the image generation pipeline than their early counterparts. They are also significantly more powerful and can perform much more computations in real time. Thus it becomes interesting to harness this additional power to further process the rendered multi-resolution model by implementing transformations that are useful as analytical tools. With such hardware, we can easily implement simple tricks such as rendering highlighted depth instead of the color to visualize other aspects of the datasets; but we can also perform in real time complex multi-step filtering and image composition techniques that would take a significant amount of effort and manipulation to reproduce in a conventional 2D image processing tool such as Photoshop for a single viewpoint, notwithstanding the need to generate and carry cumbersome intermediate representations. These newer GPUs also provide the possibility to operate on 32-bit float buffers, allowing for high precision computations and measurements to be executed directly on the GPU.

The first representation we describe is the one used to produce the wood grain detail and face close-up images of the Mona Lisa in Figure 10. One of the problems encountered when analyzing the painting came from the multi-resolution nature of the 3D data of the wood panel. Indeed, important shape information from the paint layer is hidden in larger shape variations associated with the wood grain structure of the wood panel, which in turn gets lost in the significant curvature of the wood piece itself. We therefore implemented this filter in order to be able to separate small detail variations in the local shape of an object at a given resolution from its global more general structure, acting as a high-pass filter for the data at the current level of detail. Figure 5 illustrates what we try to achieve here. The objective is to have the interesting details being represented using the entire color or gray scale by removing the global shape of the 3D data.

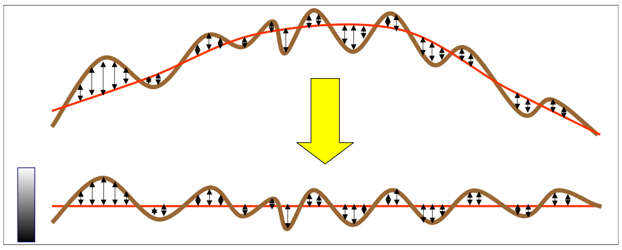


Figure 5: Illustration of the principle behind the relative depth filter. Higher frequency variations in the datasets are highlighted by removing global shape variations. They can then be mapped more efficiently to a color or grayscale spectrum.

The first step to implement this representation is to render the multi-resolution model from the chosen viewpoint. This is done mostly in the same way as for the photorealistic mode, using the same interpolation process between levels of detail as described earlier when processing the vertices. The first difference is that the target buffer is a single channel 32-bit float buffer. At the end of the vertex processing stage, we pass the position of the vertex in the observer's reference frame as a full floating point value to the rasterizer, so that we get a precise depth value for each candidate fragment in the next stage, without the non-linear resolution distribution of the depth buffer under perspective transformation. The fragment program simply renders the floating point value it receives into the 32-bit float buffer instead of assigning the color information. At the end of our first pass, we obtain a 32-bit depth image of the 3D data from the chosen observer viewpoint. The problem with such an image is that the human eye can only distinguish a limited number of different shades of gray. So even if small shaped details are actually present in this 32-bit high precision image, they are not visible when mapped into the color space because they are hidden in the global shape of the data.

The second step is to create a representation of the overall shape in the form of a second float image obtained by convolving the depth image with a large Gaussian blur kernel. Convolution is simply implemented in a new rendering pass by fetching all surrounding values overlapping the kernel as texture fetches in the unfiltered image. In fact, such a filter is implemented even more efficiently on a GPU (Fernando, 2004) by filtering sequentially with a unidimensional kernel along both image axes, resulting in much less texture fetches. We in effect filter out all the fine details we seek to recover while keeping the global warping of the panel. Very large kernel filters can be implemented by simply repeating this sequence. By adapting the filter size, different elements can obviously be highlighted under user control.

In the following step, we subtract the original image from the filtered one in a new rendering pass. The outcome of this process is a new image that contains only the local variations that were contained only in the original buffer. We now only need to transform those small variations into color values for display. Since the GPU needs to re-scale values between zero and one to produce color values, we need to find the minimum and maximum depth values in our depth image. The fastest way to achieve this on a GPU is to iteratively combine neighboring pixel values by rendering recursively into smaller buffers until we are left with a one-by-one image containing our result, a technique called parallel reduction (Fernando, 2004). We use two color channels to find the minimum and the maximum at the same time. The final result can be mapped to either a color or a gray-scale spectrum depending on user preference. In practice, for a typical rendering at 2560x1600 of a large model with sub-pixel geometry resolution, we observe a reduction by a factor of between 2 and 3 in rendering speed, even if we now actually perform 6 full passes (plus the smaller ones for the reduction) instead of a single one.

The second representation we describe relates to the classical contour lines representation that is at the basis of 2D cartographic representations of terrain data. Again, implementing it on the GPU in a multi-resolution context comes with many advantages:

- Computations are only performed at the visible resolution for every frame, therefore at very low cost.
- Instead of being fixed, the reference frame for the virtual elevation can be based either on the observer viewpoint, on a real elevation axis when applicable, or on any interactively chosen frame.
- The density, color and phase of contour lines can be adjusted in real time to easily choose appropriate contrasts and balance between precision of the information and clutter of the underlying scene.

From an implementation perspective, this is actually simpler than the multi-pass filtering we just described. Here, we modify the fragment program so that it will blend or replace the traditional shading computed for the vertices and interpolated for the fragments. This is done based on simple mathematical rules. The interface provide control on the frequency of contour lines relative to depth variation, phase of the apparition of the lines, and speed of the color cycling in the contour line set. We must also ensure that the contour line is only one pixel wide, even in flat areas that fall within the boundary values characterizing a specific line. This is achieved by taking the derivative of the depth image at that fragment, a functionality provided by the driver, and only coloring the pixels that border fragments of different depths. By adding another pass, it would be possible to smooth and fine-tune the width of contour lines and remove very small contours that tend to clutter the image.

Figures 8 and 10 both contain illustrations of this visualization tool. A variation of this technique is the color coding frequently used in scientific visualization where a full color spectrum is mapped onto the value range of a variable. In the case of the left image of Figure 10 we actually blend contour lines, regular color shading and depth coloring to produce the imagery.

4.1 the Atelier3D Interface

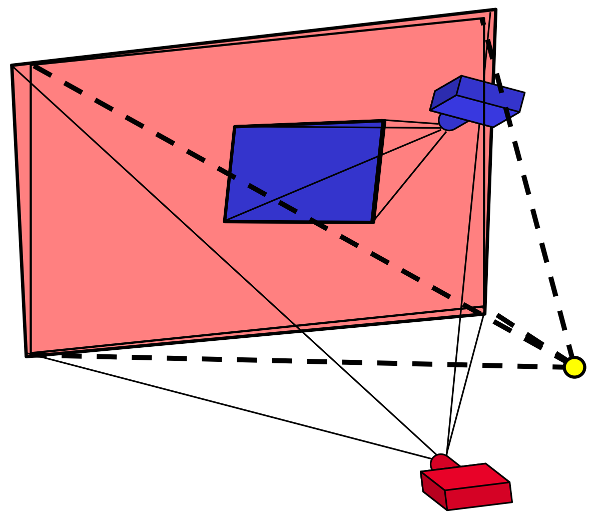
The Atelier3D GUI provides interface tools to first control the multi-resolution process, such as choosing the resolution or the amount of data to pre-fetch to balance speed, memory usage and pre-fetch misses. It provides tools to control classical lighting and texturing, allowing to add new sources and control their color and parameters such as their specularity and ambient vs directional component, change their orientation, remove texture, and other simple features expected from a versatile 3D viewer. But as we have shown the interface also provides a set of tools to interactively transform the rendered geometry and pixel data to produce other representations. The interface is completed with other tools to render large images, apply histogram adjustments, fit planes in the data for better orientation selection, make virtual cuts and perform distance measurements.

The need for Atelier3D came from the fact that many of the required tools to analyze 3D data exist but are dispersed and sometimes ill-adapted to the specific context of large sensor-based datasets. For example, analysis software for industrial applications provides tools to directly perform measurements and filter 3D data but cannot handle models that size. Another possibility is to render the datasets into images using large model visualization tools and then try to extract information using 2D image processing tools. But this is limitative and sometimes cumbersome. Applying the relative depth filter in a 2D tool from a initial full size depth rendering would involve selecting areas, cropping, applying a filter to data that is at too high a resolution and therefore waiting for the result, figuring out numerical values for the filters, etc., not to mention that many of the common image editing/processing software tools do not operate on 32-bit floating point images. And in that case one also needs to choose the right viewpoint before seeing the result. Navigating the result space is obviously more desirable.

Movie animations are a very powerful way of communicating results from 3D modeling projects and have been used extensively in recent years. Atelier3D provides adapted functionality to produce movies in contexts where datasets are too large for existing production software or when rendering resources are limited. Simple navigation paths in the scene for the observer and for light sources can be imported from other authoring softwares as X3D or VRML in Atelier3D to produce pre-animated sequences, these can then be played interactively or rendered into a sequence of images to be integrated in a movie production. The application can divide each frame into tiles to render scene at arbitrarily large resolutions. We have produced movie sequences of the Mona Lisa at a resolution of up to 4096x2160 on a single inexpensive PC with individual tiles being rendered in near real-time. With 3D datasets of the size of what can easily be produced with current sensor technology, the ability to render at very high resolution becomes essential. Complete movies can then be easily assembled from those sequences using existing editing and post-processing software.

4.2 display resolution

High resolution models yield very high resolution imagery which should be appropriately displayed in order to be fully appreciated. When working in a workstation configuration, we can leverage our datasets by using one or two relatively affordable digital 30 inch screens with a resolution of 2560x1600 pixels. However, when working collaboratively on larger screens, we are quickly limited by the much lower resolution of commodity projectors. Indeed, a 1024x768 meeting room projector is more expensive than our high resolution digital display although it can display five time less pixels. Typical solutions to this problem involve using multiple projectors and image tiling to obtain more resolution. But such setups can become very expensive and require very large installations, especially in the context of stereoscopic visualization which requires either doubling the number of projectors for passive techniques, or the use of time-multiplexed stereo projectors. We have developed an alternative technique that combines stereoscopic visualization with a focus+context or foveated approach(Figure 6). On a large display wall, we project a high



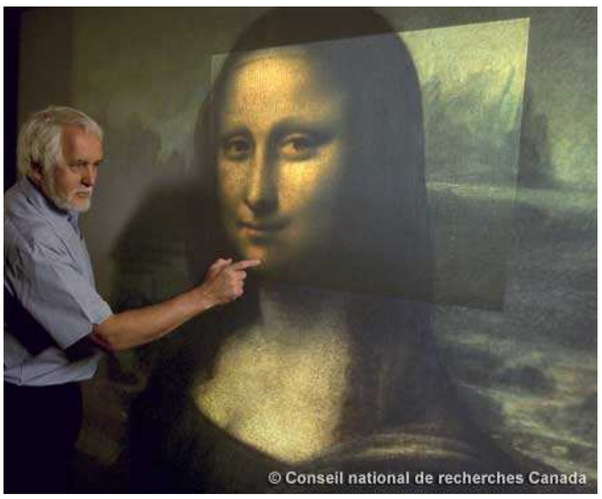


Figure 6: Top: Structure of a foveated display wall: a high resolution inset is added to a normal display wall setup using a second projector, or pair of projectors for a stereo configuration. Bottom: resulting image in a monoscopic configuration.

resolution, high brightness stereo image into a larger, lower resolution one. Each image is produced by a pair of commodity PCs and projectors. Such a setup combines the advantages of accessing a high resolution area of interest while keeping a larger context within sight. Techniques have been developed which insure that the presence of the inset does not affect the stereoscopic perception along its boundary, and that no special manual efforts are required to align the projectors.(Godin et al., 2004a)(Godin et al., 2004b)

4.3 Collaborative environment

We have also included modalities to support collaborative work on our display wall(Borgeat et al., 2004). Figure 7 illustrates such a collaborative session where participants can share a virtual world based on a multi-resolution model, and interact by inserting X3D avatars with video insets and audio links in the environment. Participants can annotate the model using 3D drawing and guide the viewpoint of other participants to collaborate. The application has been entirely designed with large datasets in mind, and numerous optimizations have been implemented to maximize resource usage. For example, video is partially encoded and decoded on the GPU as part of the 3D rendering, and all the 3D annotations are performed using information from the depth buffer

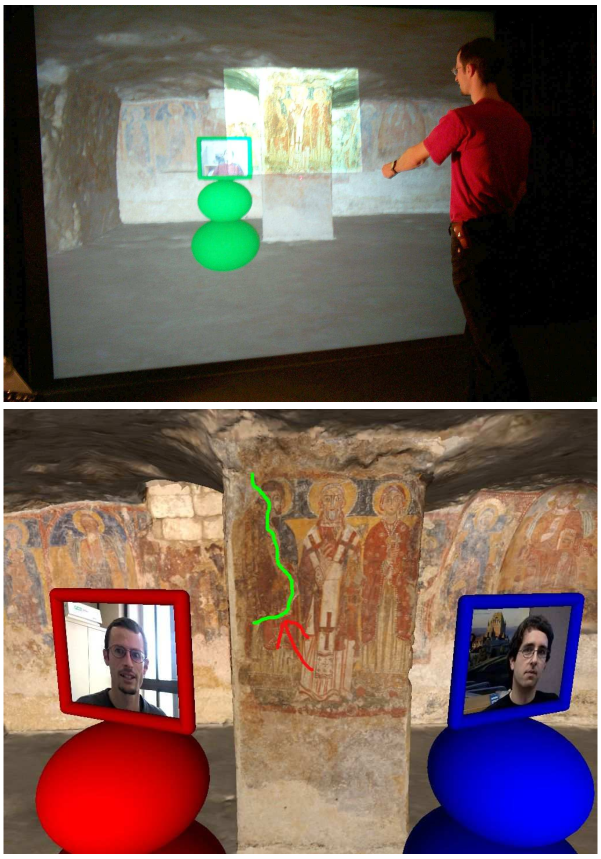


Figure 7: Top: Participant interacting within a collaborative 3D environment using a laser pointer. Bottom: Video insets and avatars are used to locate participants, and 3D annotations to the model provides interaction with the dataset.

at screen resolution so that no useless intersection computations have to be done with the datasets. The scene graph is also synchronized between the different collaborators using efficient differential encoding.

5 RESULTS AND DISCUSSION

We now present a few results produced from snapshots of the interactive Atelier3D interface rendering window. These images come from models of various size and types and are produced using different acquisition and modeling techniques.

Figure 10 shows snapshots of a 3D model of Leonardo's Mona Lisa illustrating results that were of interest to conservators and art historians. The 3D scan of the Mona Lisa was part of the largest scientific examination ever conducted on the Mona Lisa (Mohen et al., 2006). It was led by the Centre de recherche et de restauration des musées de France (C2RMF) at the request of the Paintings Department of the Louvre; the study coincided with the move of the painting to the newly renovated Salle des États. The obverse(front), reverse and sides of the wood panel were scanned using a prototype high-resolution polychromatic 3D laser scanner. The laser scans of the Mona Lisa were assembled into a 333 million polygon color-per-vertex model, with higher resolution (average sampling of 60 μm) on the obverse (front) surface. Some of the preliminary results from the study have already been published (Mohen et al., 2006). Many of the features in Atelier3D were developed in the context of that project.

The 3D modeling of the Mona Lisa aimed at documenting the state of the painting and at providing complementary information for the analysis of the pictorial layer, concerning both conservation issues and indications relative to the painting technique of Leonardo. These objectives could not be reached using only photorealistic rendering of the model. Specific transformation techniques were required to enhance features of interest, such as wood grain, or measured local variations of the pictorial layer. The 3D model also allowed detailed examination of various features, such as insect galleries or other traces of events such as restorations that occurred throughout the painting's five centuries of existence.

The landscape model in Figure 1 and 8 was produced from data gathered by TerraPoint (Terrapoint, 2005) using their proprietary ALMIS350 LiDAR system and a digital camera. It is composed of 103 million data points and texture from 6800 4Mpixels photographs. The hydrographic network is clearly visible using the relative depth technique described in the previous section. The contour lines are also an obvious representation for such a dataset. The hybrid contour/color coding/shading image in Figure 1 is also interesting in that it provides a good perception of the importance of elevation relative to the height of the buildings, giving a good impression of possible lines of sights between the buildings before having to confirm it using an appropriate grazing viewpoint.

The model from Figure 9 was produced from digital photographs using the technique described in (El-Hakim, 2006). It is composed of 15M polygons. The figure illustrates again how transforming the data can help efficiently extracting useful information, in this case by easily being able to read the text. This high resolution modeling technique from photographs has obviously the potential to lead to the creation of very large 3D models. Such data could be either converted to a textured multi-resolution representation if keeping all the geometric resolution is unnecessary, or to a simpler color-per-vertex one since there is naturally a one-to-one mapping between vertices and pixels in the images.

6 CONCLUSIONS AND FUTURE WORK

This paper introduced the Atelier3D framework, which provides a useful set of tools to support the visualization and analysis of datasets made from range and color sensor data. The representations illustrated here are only examples of the tools required to analyze 3D datasets. It is clear that many more could and should be implemented to reach that goal. Nevertheless, they give a good overview of the kind of processing that can be easily implemented on a GPU and of the importance of not limiting the visualization of our 3D models strictly to realistic representations. Finally, these results also emphasize the fact that there is no substitute to actual 3D visualization for exploring and understanding 3D data.

ACKNOWLEDGMENTS

The authors would like to thank Christian Lahanier and Bruno Mottin at the C2RMF, Paul Mrstik and Kresimir Kusevic from Terrapoint Inc., Prof. Virginia Valzano of the University of Lecce (Italy), and our NRC colleagues Luc Cournoyer, Michel Picard, John Taylor, Louis-Guy Dicaire, Marc Rioux and Daniel Gamache.

REFERENCES

Anderson, S. and Levoy, M., 2002. Unwrapping and visualizing cuneiform tablets. IEEE Comput. Graph. Appl. 22(6), pp. 82–88.

Beraldin, J.-A., Picard, M., El-Hakim, S., Godin, G., Valzano, V., Bandiera, A. and Latouche, C., 2002. Virtualizing a Byzantine

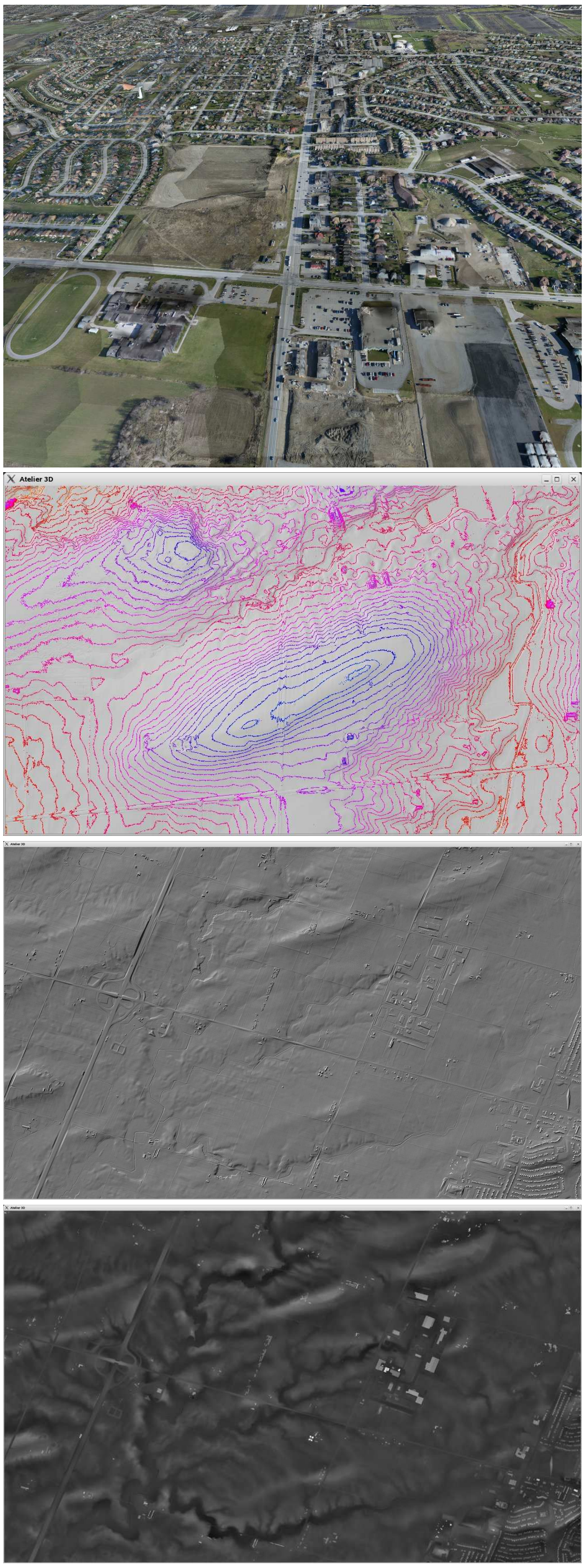


Figure 8: From the top: Textured rendering of a dataset composed of 103 million LiDAR range samples and 6800 digital photographs. Another area of the model rendered using interactive contour lines. Two bottom images: a third area of the model first rendered using classical OpenGL shading with a raking light source, then rendered using monochrome relative depth rendering.

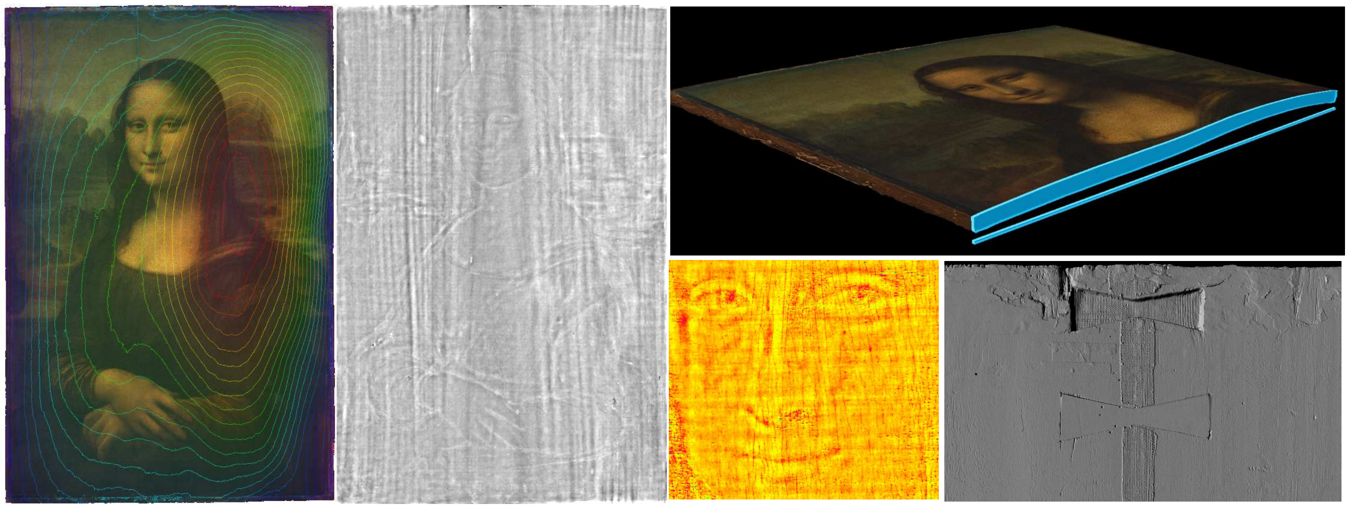


Figure 10: Different snapshots taken from the Atelier3D interactive visualization software illustrating selected aspects of the Mona Lisa: the global curvature of the wood panel, the wood grain structure, pictorial layer depth, and restored areas on the back of the panel.

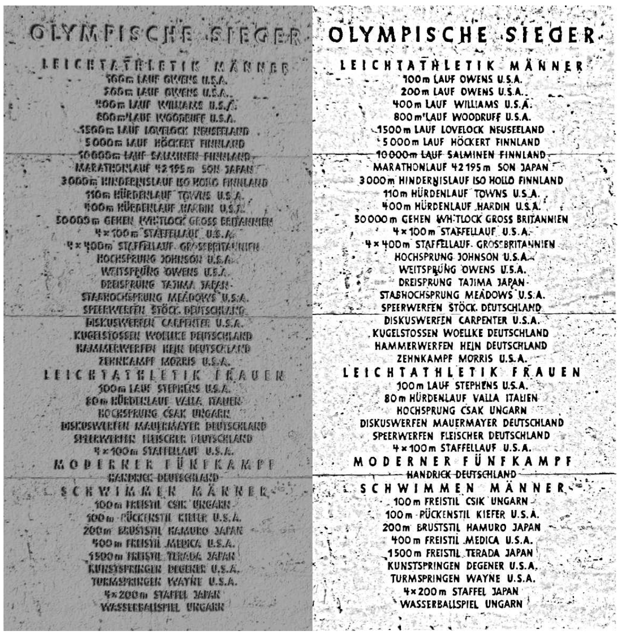

OLYMPISCHE SIEGER
LEICHTATHLETIK MÄNNER
100m LAUF OWENS U.S.A.
200m LAUF OWENS U.S.A.
400m LAUF WILLIAMS U.S.A.
800m LAUF WOODRUFF U.S.A.
1500m LAUF LOVELOCK NEUSEELAND
5000m LAUF HÖCKERT FINNLAND
10000m LAUF SALMINEN FINNLAND
MARATHONLAUF 42195m SON JAPAN
3000m HINDERNISLAUF ISO HOLLO FINNLAND
110m HÜRDENLAUF TOWNS U.S.A.
400m HÜRDENLAUF HARDIN U.S.A.
50000m GEHEN WHITLOCK GROSS BRITANNIEN
4×100m STAFFELLAUF U.S.A.
4×400m STAFFELLAUF GROSSBRITANNIEN
HOCHSPRUNG JOHNSON U.S.A.
WEITSPRUNG OWENS U.S.A.
DREISPRUNG TAJIMA JAPAN
STABHOCHSPRUNG MEADOWS U.S.A.
SPEERWERFEN STÖCK DEUTSCHLAND
DISKUSWERFEN CARPENTER U.S.A.
KUGELSTOSSEN WOELLKE DEUTSCHLAND
HAMMERWERFEN HEIN DEUTSCHLAND
ZEHNKAMPF MORRIS U.S.A.
LEICHTATHLETIK FRAUEN
100m LAUF STEPHENS U.S.A.
80m HÜRDENLAUF VALLA ITALIEN
HOCHSPRUNG CSAK UNGARN
DISKUSWERFEN MAUERMAYER DEUTSCHLAND
SPEERWERFEN FLEISCHER DEUTSCHLAND
4×100m STAFFELLAUF U.S.A.
MODERNER FÜNFKAMPF
HANDRICK DEUTSCHLAND
SCHWIMMEN MÄNNER
100m FREISTIL CSIK UNGARN
100m RÜCKENSTIL KIEFER U.S.A.
200m BRUSTSTIL HAMURO JAPAN
400m FREISTIL MEDICA U.S.A.
1500m FREISTIL TERADA JAPAN
KUNSTSPRINGEN DEGENER U.S.A.
TURMSPRINGEN WAYNE U.S.A.
4×200m STAFFEL JAPAN
WASSERBALLSPIEL UNGARN

Figure 9: Two representations of a 3D model build from photogrammetry and shape from shading. Left: regular shading, Right: relative depth analysis and histogram adjustments.

crypt by combining high-resolution textures with laser scanner 3D data. In: Proceedings of the 8th International Conference on Virtual Systems and Multimedia, pp. 3–14.

Borgeat, L., Godin, G., Blais, F. and Lahanier, C., 2005. GoLD: Interactive display of huge colored and textured models. In: Proc. of SIGGRAPH 2005, Los Angeles, California.

Borgeat, L., Godin, G., Lapointe, J.-F. and Massicotte, P., 2004. Collaborative visualization and interaction for detailed environment models. In: Proc. of the 10th International Conference on Virtual Systems and Multimedia, Softopia Japan, Ogaki, Japan, pp. 1204–1213.

Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F. and Scopigno, R., 2004. Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. ACM Transactions on Graphics 23(3), pp. 796–803.

Clark, J. H., 1976. Hierarchical geometric models for visible surface algorithms. Commun. of the ACM 19(10), pp. 547–554.

El-Hakim, S., 2006. A sequential approach to capture fine geometric details from images. In: ISPRS Commission V Symposium, Image Engineering and Vision Metrology, IAPRS, Dresden, Germany, pp. 97–102.

Fernando, R., 2004. GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics. Pearson Higher Education.

Funkhouser, T. A. and Séquin, C. H., 1993. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In: Proceedings of ACM SIGGRAPH 93, Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press, New York, pp. 247–254.

Garland, M. and Heckbert, P. S., 1998. Simplifying surfaces with color and texture using quadric error metrics. In: Proceedings IEEE Visualization '98, IEEE Computer Society Press, pp. 263–269.

Godin, G., Beraldin, J.-A., Taylor, J., Rioux, M., El-Hakim, S., Baribeau, R., Blais, F., amd J. Domey, P. B. and Picard, M., 2002. Active optical 3D imaging for heritage applications. Computer Graphics and Applications 22(5), pp. 24–35.

Godin, G., Lalonde, J.-F. and Borgeat, L., 2004a. Projector-based dual-resolution stereoscopic display. In: IEEE Conference on Virtual Reality 2004, pp. 223–224.

Godin, G., Massicotte, P. and Borgeat, L., 2004b. Foveated stereoscopic display for the visualization of detailed virtual environments. In: Eurographics Symposium on Virtual Environments, Grenoble, France, pp. 7–16.

Hoppe, H., 1997. View-dependent refinement of progressive meshes. In: Proceedings of ACM SIGGRAPH 97, Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press, New York, pp. 189–198.

Hoppe, H., 1999. New quadric metric for simplifiying meshes with appearance attributes. In: Proceedings IEEE Visualization '99, IEEE Computer Society Press, pp. 59–66.

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:

3D scanning of large statues. In: Proceedings of ACM SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, ACM, pp. 131–144.

Mohen, J.-P., Menu, M. and Mottin, B., 2006. Mona Lisa: Inside the Painting. Harry N. Abrams.

Niski, K., Purnomo, B. and Cohen, J., 2007. Multi-grained level of detail using a hierarchical seamless texture atlas. In: SI3D ’07: Proceedings of the 2007 symposium on Interactive 3D graphics and games, ACM Press, New York, NY, USA, pp. 153–160.

Rusinkiewicz, S. and Levoy, M., 2000. QSplat: a multiresolution point rendering system for large meshes. In: Proceedings of ACM SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, ACM, ACM Press/Addison-Wesley Publishing Co., pp. 343–352.

Terrapoint, 2005. Terrapoint Inc. http://www.terrapoint.com/.

Yoon, S.-E., Lindstrom, P., Pascucci, V. and Manocha, D., 2005. Cache-oblivious mesh layouts. ACM Trans. Graph. 24(3), pp. 886–893.

Zwicker, M., Räsänen, J., Botsch, M., Dachsbacher, C. and Pauly, M., 2004. Perspective accurate splatting. In: GI ’04: Proceedings of the Graphics Interface Conference, Canadian Human-Computer Communications Society, pp. 247–254.