1. INTRODUCTION

In the 40 years that N-body simulations have been used in cosmology research, visualization has been the most indispensable tool. Physical processes have often been identified first and studied via images of simulations. A few examples are the formation of filamentary structures in the large-scale distribution of matter (Jenkins et al. 1998; Colberg et al. 2005; Springel et al. 2005; Dolag et al. 2006), growth of feedback bubbles around quasars (Sijacki et al. 2007; Di Matteo et al. 2005), cold flows of gas-forming galaxies (Dekel et al. 2009; Keres et al. 2005), and the evolution of ionization fronts during the re-ionization epoch (Shin et al. 2008; Zahn et al. 2007). The size of current and upcoming peta-scale simulation data sets can make such visual exploration to discover new physics technically challenging. Here we present techniques that can be used to display images at full resolution of data sets of hundreds of billions of particles in size.

Several implementations of visualization software for cosmological simulations already exist. IRFIT (Gnedin 2011) is a general purpose visualization suite that can deal with mesh-based scalar, vector, and tensor data, as well as particle-based data sets as points. YT (Turk et al. 2011) is an analysis toolkit for mesh-based simulations that also supports imaging. SPLASH (Price 2007) is a visualization suite specialized for simulations that use smooth particle hydrodynamic (SPH) techniques. Aside from the CPU-based approaches mentioned above, Szalay et al. (2008) implemented a GPU-based interactive visualization tool for SPH simulations.

The Millennium I and II simulations (Springel et al. 2005; Boylan-Kolchin et al. 2009) have been used to test an interactive scalable rendering system developed by Fraedrich et al. (2009). Both SPLASH and the Millennium visualizer support high-quality visualization of SPH data sets, while IRFIT treats SPH data as discrete points.

Continued improvements in computing technology and algorithms are allowing SPH cosmological simulations to be run with ever increasing numbers of particles. Runs are now possible on scales which allow rare objects, such as quasars, to form in a large simulation volume with uniform high resolution (see Section 2.1; Di Matteo et al. 2011; DeGraf et al. 2011; N. Khandai et al. 2011, in preparation). Being able to scan through a vast volume and seek out the tiny regions of space where most of the activity occurs, while still keeping the large-scale structure in context, necessitates special visualization capabilities. These should be able to show the largest scale information but at the same be interactively zoomable. However, as the size of the data sets quickly exceeds the capability of moderately sized in-house computer clusters, it becomes difficult to perform any interactive visualization. For example, a single snapshot of the MassiveBlack simulation (Section 2.1) consists of 8192 files and is over 3 TB in size.

Even when a required large-scale high-resolution image has been rendered, actually exploring the data requires special tools. The GigaPan Collaboration has essentially solved this problem in the context of viewing large images, with the GigaPan viewer enabling anyone connected to the Internet to zoom into and explore in real-time images that would take hours to transfer in totality. The viewing technology has been primarily used to access large photographic panoramas, but is easily applicable to simulated data sets. A recent enhancement to deal with the time dimension, in the form of gigapixel frame interactive movies
In this work we combine an off-line imaging technique together with GigaPan technology to implement an interactively accessible visual probe of large cosmological simulations. While GigaPan is an independent project (uploading and access to the GigaPan Web site is publicly available), we release our toolkit for the off-line visualization as Gaepsi, a software package aimed specifically at GADGET (Springel et al. 2001; Springel 2005) SPH simulations.

The layout of our paper is as follows. In Section 2, we give a brief overview of the physical processes modeled in GADGET, as well as describing two P-GADGET simulations which we have visualized. In Section 3, we give details of the spatial domain remapping we employ to convert cubical simulation volumes into image slices. In Section 4, we describe the process of rasterizing an SPH density field, and in Section 5 the image rendering and layer compositing. In Section 6, we address the parallelism of our techniques and give measures of performance. In Section 7, we briefly describe the GigaPan and GigaPan Time Machine viewers and present examples of screenshots from two visualizations (which are both accessible on the GigaPan Web sites).

2. SIMULATION

Adaptive Mesh Refinement (e.g., Bryan & Norman 1999) and SPH (Monaghan 1992) are the two most frequently used schemes for carrying out cosmological simulations. In this work we focus on the visualization of the baryonic matter in SPH simulations run with P-GADGET (Springel 2005).

GADGET is an SPH implementation, and P-GADGET is a version which has been developed specifically for petascale computing resources. It simultaneously follows the self-gravitation of a collision-less $N$-body system (dark matter) and gas dynamics (baryonic matter), as well as the formation of stars and supermassive black holes. Dark matter particles, gas particle positions, and initial characteristics are set up in a comoving cube, and black hole and star particles are created according to sub-grid modeling (Hernquist & Springel 2003; Di Matteo et al. 2008). Gas particles carry hydrodynamical properties, such as temperature, star formation rate, and neutral fraction.

Although our attention in this paper is limited to imaging properties of the gas, stars, and black holes in GADGET simulations, similar techniques could be used to visualize the dark matter content by computing and using an SPH-like adaptive smoothing kernel for the dark matter particles. Also, the software we provide should be easily adaptable to the data formats of other SPH codes (e.g., GASOLINE; Wadsley et al. 2004).

2.1. MassiveBlack

The MassiveBlack simulation is the state-of-the-art SPH simulation of a ΛCDM universe (Di Matteo et al. 2011). P-GADGET was used to evolve $2 \times 3200^3$ particles in a volume of side length $533 h^{-1}$ Mpc with a gravitational force resolution of $5 h^{-1}$ Kpc. One snapshot of the simulation occupies $3 \text{TB}$ of disk space, and the simulation has been run so far to redshift $z = 4.75$, creating a data set of order $120 \text{TB}$. The fine resolution and large volume of the simulation permits one to usefully create extremely large images. The simulation was run on the high performance computing facility, Kraken, at the National Institute for Computational Sciences in full capability mode with 98,000 CPUs.

2.2. E5

To make a smooth animation of the evolution of the universe typically requires hundreds of frames directly taken as snapshots of the simulation. The scale of the MassiveBlack run is too large for this purpose, so we ran a much smaller simulation (E5) with $2 \times 336^3$ particles in a $50 h^{-1}$ Mpc comoving box. The model was again a ΛCDM cosmology, and one snapshot was output per 10 million years, resulting in 1367 snapshots. This simulation ran on 256 cores of the Warp cluster in the McWilliams Center for Cosmoslogy at CMU.

3. SPATIAL DOMAIN REMAPPING

Spatial domain remapping can be used to transform the periodic cubic domain of a cosmological simulation to a patch whose shape is similar to the domain of a sky survey, while making sure that the local structures in the simulation are preserved (Carlson & White 2010; Hilbert et al. 2009). Our usage focuses on making a thin slice that includes the entire volume of the simulation.

A GADGET cosmological simulation is usually defined in the periodic domain of a cube. As a result, if we let $f(\mathbf{X} = (x, y, z))$ be any position-dependent property of the simulation, then

$$f(\mathbf{X}) = f((x + \mu L, y + v L, z + \sigma L)),$$

where $\mu, v, \sigma$ are integers. The structure corresponds to a simple cubic lattice with a lattice constant $a = L$, the simulation box side length. A bijective mapping from the cubic unit cell to a remapped domain corresponds to a choice of the primitive cell. Figure 1 illustrates the situation in two dimensions.

While the original remapping algorithm by Carlson & White (2010) results in the correct transformations being applied, it has two drawbacks: (1) the orthogonalization is invoked explicitly and (2) the hit-testing for calculation of the shifting (see below) is against non-aligned cuboids. The second problem especially undermines the performance of the program. In this work we...
Numerical errors. Multiplying by a QR decomposition (which is widely available as a library routine), and hit-testing against an AABB (Axis Aligned Bounding Box).

Boundary effects and smoothed particles. Four images of a particle intersecting the boundary are shown. The top-right image is contained in the transformed domain, but the other three are not. The contribution of the two bottom images is lost. By requiring the size of the transformed domain to be much larger than typical SPH smoothing lengths, most particles do not intersect a boundary of the domain and the error is contained near the edges.

First, the transformation of the primitive cell is given by a unimodular integer matrix,

\[
M = \begin{pmatrix}
M_{11} & M_{12} & M_{13} \\
M_{21} & M_{22} & M_{23} \\
M_{31} & M_{32} & M_{33}
\end{pmatrix},
\]

where \(M_{ij}\) are integers and the determinant of the matrix \(|M| = 1\). It is straightforward to obtain such matrices via enumeration (Carlson & White 2010). The QR decomposition of \(M\) is

\[
M = QR,
\]

where \(Q\) is an orthonormal matrix and \(R\) is an upper-right triangular matrix. It is immediately apparent that (1) application of \(Q\) yields rotation of the basis from the simulation domain to the transformed domain, the column vectors in matrix \(Q^T\) being the lattice vectors in the transformed domain and (2) the diagonal elements of \(R\) are the dimensions of the remapped domain.

For imaging it is desired that the thickness along the line of sight is significantly shorter than the extension in the other dimensions, thus we require \(0 < R_{33} \ll |R_{22}| < |R_{11}|\). Note that if a domain that is much longer in the line-of-sight direction is desired, for example, to calculate long range correlations or to make a sky map of a whole simulation in projection, the choice shall be \(0 < |R_{33}| < |R_{22}| \ll |R_{11}|\).

Next, for each sample position \(X\), we solve the indefinite equation of integer cell number triads \(I = (I_1, I_2, I_3)^T\),

\[
\tilde{X} = Q^T X + a Q^T I,
\]

where \(a\) is the box size and \(\tilde{X}\) is the transformed sample position satisfying \(\tilde{X} \in [0, R_{11}) \times [0, R_{22}) \times [0, R_{33})\). In practice, the domain of \(\tilde{X}\) is enlarged by a small number \(\epsilon\) to address numerical errors. Multiplying by \(Q\) on the left and re-organizing the terms, we find

\[
I = \frac{Q \tilde{X}}{a} - \frac{X}{a}.
\]

Note that \(Q \tilde{X}\) is the transformed sample position expressed in the original coordinate system and is bounded by its AABB box. If we let \((Q \tilde{X})_i/a_i \in [B_i, T_i]\), where \(B_i\) and \(T_i\) are integers, and note \((X_i/a_i) \in [0, 1)\), the resulting bounds of \(I\) are given by

\[
I_i \in [B_i, T_i].
\]

We then enumerate the range to find \(\tilde{X}\).

When the remapping method is applied to the SPH particle positions, the transformations of the particles that are close to the edges give inexact results. The situation is similar to the boundary error in the original domain when the periodic boundary condition is not properly considered. Figure 2 illustrates the situation by showing all images of the particles that contribute to the imaging domain. We note that for the purpose of imaging, by choosing an \(R_{33}\) (the thickness in the thinner dimension) much larger than the typical smoothing length of the SPH particle, the errors are largely constrained to lie near the edge. These issues are part of general complications related to the use of a simulation slice for visualization. For example, in an animation of the distribution of matter in a slice it is possible for objects to appear and disappear in the middle of the slice as they pass through it. These limitations should be borne in mind, and we leave three-dimensional visualization techniques for future work.

The transformations used for the MassiveBlack and E5 simulations are listed in Table 1.

### Table 1

| Simulation | MassiveBlack | E5 |
|------------|--------------|----|
| Matrix \(M\) | \(\begin{pmatrix} 5 & 6 & 2 \\ 3 & 7 & 3 \\ 2 & 7 & 0 \end{pmatrix}\) | \(\begin{pmatrix} 3 & 1 & 0 \\ 2 & 1 & 0 \end{pmatrix}\) |
| \(R_{11}\) (\(h^{-1}\) Mpc) | 3300 | 180 |
| \(R_{11}\) Pixels (kilo) | 810 | 36.57 |
| \(R_{22}\) (\(h^{-1}\) Mpc) | 5800 | 101 |
| \(R_{22}\) Pixels (kilo) | 1440 | 20.48 |
| \(R_{33}\) (\(h^{-1}\) Mpc) | 7.9 | 6.7 |
| Resolution (\(h^{-1}\) Kpc) | 4.2 | 4.9 |

4. **RASTERIZATION**

In a simulation, many field variables are of interest in visualization.

1. Scalar fields: density \(\rho\), temperature \(T\), neutral fraction \(x_{HI}\), star formation rate \(\phi\).
2. Vector fields: velocity, gravitational force.

In an SPH simulation, a field variable as a function of spatial position is given by the interpolation of the particle properties. Rasterization converts the interpolated continuous field into raster pixels on a uniform grid. The kernel function of a particle at position \(y\) with smoothing length \(h\) is defined as

\[
W(y, h) = \frac{8}{\pi h^3} \begin{cases} 
1 - 6 \left(\frac{y}{h}\right)^2 + 6 \left(\frac{y}{h}\right)^3, & 0 \leq \frac{y}{h} \leq \frac{1}{2} \\
2 \left(1 - \frac{y}{h}\right)^3, & \frac{1}{2} < \frac{y}{h} \leq 1, \\
0, & \frac{y}{h} > 1.
\end{cases}
\]
Note that in MassiveBlack and E5, the smoothing length of the gas particles is calculated during the run. Following the usual prescriptions (e.g., Springel 2005; Price 2007), the interpolated density field is taken as

$$\rho(x) = \sum_i m_i W(x - x_i, h_i),$$

where $m_i$, $x_i$, and $h_i$ are the mass, position, and smoothing length of the $i$th particle, respectively. The interpolation of a field variable, denoted by $A$, is given by

$$A(x) = \sum_i A_i m_i W(x - x_i, h_i) \rho(x_i) + O(h^2),$$

where $A_i$ is the corresponding field property carried by the $i$th particle. Note that the density field can be seen as a special case of the general formula.

To obtain a two-dimensional image, we take the line-of-sight projection when calculating the pixel-wise mean. Let $P$ be a pixel. For the volume-weighted mean of the density field,

$$\bar{\rho}_2(P) = \frac{M(P)}{P} = \frac{1}{P} \int_z \int_p d^2x \rho(x)$$

$$= \frac{1}{P} \sum_i m_i \int_p d^2x W_2(x - x_i, h_i)$$

$$= \frac{1}{P} \sum_i M_i(P),$$

where $M_i(P) = m_i \int_p d^2x W_2(x - x_i, h_i)$ is the mass overlapping of the $i$th particle and the pixel, and $W_2(x, h) = \int_z d^2x W(x, h)$ is the two-dimensional projected kernel.

For the mass-weighted mean of a field $(A)^{11}$

$$\bar{A}_2(P) = \frac{\int_z \int_p d^2x A(x) \rho(x)}{\int_z \int_p d^2x \rho(x)}$$

$$= \sum_i A_i M_i(P) \frac{M(P)}{P} + O(h^2).$$

Both formulas require frequent calculation of the overlap between the kernel function and the pixels $\int_p d^2x W_2(x, h)$. An effective way to calculate the overlap is via a lookup table that is pre-calculated and hard coded in the program. The lookup table is indexed by the corner positions, of the two-dimensional overlapping area between a pixel and a particle, normalized to a particle quadrant. The resolution of the table is $(64, 64)\times(64, 64)$.

Three levels of approximation are used in the calculation of the contribution of a particle to a pixel.

1. When a particle is much smaller than a pixel, the particle contributes to the pixel as a whole. Neither interpolation nor lookup occurs.
2. When a particle and pixel are of similar size (up to a few pixels in size), the contribution to each of the pixels is calculated by interpolating between the overlapping areas read from a lookup table.
3. When a particle is much larger than a pixel, the contribution to a pixel is taken to be the center kernel value times the area of a pixel.

11 The second line is an approximation due to the bounded second-order derivative of the smoothing kernel.

Note that Levels 1 and 3 are significantly faster than Level 2 as they do not require interpolations. In both MassiveBlack and E5, at the finest resolution, the majority of gas particles are well resolved and rasterized with Level 3 approximation. However, due to the wide spread of the smoothing length, even at the finest resolution, some particles (which tend to reside in overdense regions) still invoke the Level 1 and Level 2 approximations.

The rasterization of the $z = 4.75$ snapshot of MassiveBlack was run on the SGI UV Blacklight supercomputer at the Pittsburgh Supercomputing Center. Blacklight is a shared memory machine equipped with a large memory for holding the image and a fairly large number of cores enabling parallelism, making it the most favorable machine for the rasterization. The rasterization of the E5 simulation was run on the local CMU machine Warp. The pixel dimensions of the raster images are also listed in Table 1. The pixel scales have been chosen to be around the gravitational softening length of $\sim 5\ h^{-1}\ Kpc$ in these simulations in order to preserve as much information in the image as possible.

5. IMAGE RENDERING AND LAYER COMPOSING

The rasterized SPH images are color mapped into red, green, blue, and opacity (RGBA) layers. Two modes of color mapping are implemented, the simple mode and the intensity mode.

In the simple mode, the color of a pixel is directly obtained by looking up the normalized pixel value in a given color table. To address the large (several orders of magnitude) dynamic range of the fields, the logarithm of the pixel value is used in place of the pixel value itself.

In the intensity mode, the color of a pixel is determined in the same way as done in the simple mode. However, the opacity is reduced by a factor $f_m$ that is determined by the logarithm of the total mass of the SPH fluid contained within the pixel. To be more specific,

$$f_m = \left\{ \begin{array}{ll} 0 & \log M < a, \\ 1 & \log M > b, \\ \left[ \frac{\log M - a}{b - a} \right]^{\gamma} & \text{otherwise}, \end{array} \right.$$ 

where $a$ and $b$ are the underexposure and overexposure parameters: any pixel that has a mass below $10^a$ is completely transparent, and any pixel that has a mass above $10^b$ is completely opaque.

The RGBA layers are stacked one on top of another to composite the final image. The compositing assumes an opaque black background. The formula to composite an opaque bottom layer $B$ with an overlay layer $T$ into the composite layer $C$ is (Porter & Duff 1984)

$$C = a F + (1 - a) T,$$

where $C$, $B$, and $T$ stand for the RGB pixel color triplets of the corresponding layer and $a$ is the opacity value of the pixel in the overlay layer $T$. For example, if the background is red and the overlay color is green, with $a = 50\%$, the composite color is a 50%-dimmed yellow.

Point-like (non SPH) particles are rendered differently. Black hole particles are rendered using circles, with the radius proportional to the logarithm of the mass. Star particles are currently rendered as colored points, due to a lack of smoothing length in the direct simulation data. In our example images, the Massive-Black simulation visualization used a fast rasterizer that does
not support anti-aliasing, while the frames of E5 are rendered using matplotlib (Hunter 2007) that does anti-aliasing.

The choice of the colors in the color map has to be made carefully to avoid confusing different quantities. We choose a color gradient which spans black, red, yellow, and blue for the color map of the projected gas density field. This color map is shown in Figure 3. Composited above the projected gas density field is the projected mass-weighted average of the star formation rate field, shown in dark blue, and with complete transparency where the field vanishes. Additionally, we choose solid white pixels for the star particles. In the future, we plan to calculate the smoothing length of star particles and render them with a smoothed field as done with the gas particles. Black holes are shown as green circles. In the E5 animation frames, the normalization of the gas density color map has been fixed so that the maximum and minimum values correspond to the extreme values of density in the last snapshot.

6. PARALLELISM AND PERFORMANCE

The large simulations we are interested in visualizing have been run on large supercomputer facilities. In order to image them with sufficient resolution to be truly useful, the creation of images from the raw simulation data also needs significant computing resources. In this section we outline our algorithms for doing this and give measures of performance.

6.1. Rasterization in Parallel

We have implemented two types of parallelism, which we shall refer to as “tiny” and “huge,” to make the best use of shared memory architectures and distributed memory architectures, respectively. The tiny parallelism is implemented with OpenMP and takes advantage of the case when the image can be held within the memory of a single computing node. The parallelism is achieved by distributing the particles in batches to the threads within one computing node. The raster pixels are then color mapped in serial, as is the drawing of the point-like particles. The tiny mode is especially useful for interactively probing smaller simulations.

The huge version of parallelism is implemented using the Message Passing Interface libraries and is used when the image is larger than a single computing node or the computing resources within one node are insufficient to finish the rasterization in a timely manner. The imaging domain is divided into horizontal stripes, each of which is hosted by a computing node. When the snapshot is read into memory, only the particles that contribute to the pixels in a domain are scattered to the hosting node of the domain. Due to the growth of cosmic structure as we move to lower redshifts, some of the stripes inevitably have many more particles than others, introducing a load imbalance. We define the load imbalance penalty \( \eta \) as the ratio between the maximum and the average of the number of particles in a stripe. The computing nodes with fewer particles tend to finish sooner than those with more. The color mapping and the drawing of point-like particles are also performed in parallel in the huge version of parallelism.

| Pixels (Kpc pixel\(^{-1}\)) | \( \bar{n} \) | CPUs | Wall-time (hours) | \( \eta \) | Rate (Ks\(^{-1}\)) |
|-----------------------------|---------------|-------|-------------------|---------|------------------|
| 5.6G                        | 58.4          | 80    | 128               | 1.63    | 1.45             | 11.3 |
| 22.5G                       | 29.2          | 330   | 256               | 3.17    | 1.66             | 13.4 |
| 90G                         | 14.6          | 1300  | 512               | 3.35    | 1.57             | 24.0 |
| 90G                         | 14.6          | 1300  | 512               | 3.06    | 1.39             | 23.3 |
| 360G                        | 7.3           | 5300  | 512               | 7.65    | 1.39             | 37.2 |
| 1160G                       | 4.2           | 16000 | 1344              | 10.1    | 1.56             | 37.2 |

Notes. Here \( N \) is the number of SPH particles, Res is the resolution (pixel scale), \( \bar{n} \) is the mean number of pixels that overlap each particle, \( \eta \) is the load imbalance penalty averaged over patches, and the final column, Rate, is the number of kilopixels rasterized per second.

6.2. Performance

The time spent in domain remapping scales linearly with the total number of particles \( N \),

\[ T_{\text{remap}} \sim O(N). \]

The time spent in color mapping scales linearly with the total number of pixels \( P \),

\[ T_{\text{color}} \sim O(P). \]

Both processes consume a very small fraction of the total computing cycles.

The rasterization consumes a much larger part of the computing resources and it is useful to analyze it in more detail. If we let \( \bar{n} \) be the number of pixels overlapping a particle, then \( \bar{n} = K^{-1}N^{-1}P \), where \( K^{-1} \) is a constant related to the simulation. Now we let \( t(n) \) be the time it takes to rasterize one particle as a function of the number of pixels overlapping the particle. From the three levels of detail in the rasterization algorithm (Section 4), we have

\[ t(n) = \begin{cases} 
C_1, & n \ll 1 \\
C_2n, & n \sim 1 \\
C_3n, & n \gg 1
\end{cases} \]

with \( C_2 \gg C_1 \approx C_3 \). The effective pixel filling rate \( R \) is defined as the total number of image pixels rasterized per unit time,

\[ R = P\left[ t(\bar{n})N \right]^{-1} = \bar{n}K[t(\bar{n})]^{-1} = \begin{cases} 
\bar{n}K^{-1}C_1^{-1}, & \bar{n} \ll 1 \\
K^{-1}C_2^{-1}, & \bar{n} \sim 1 \\
K^{-1}C_3^{-1}, & \bar{n} \gg 1
\end{cases} \]

The rasterization time to taken to create images from a single snapshot of the MassiveBlack simulation (at redshift 4.75) at various resolutions is presented in Table 2 and Figure 4.

The rasterization of the images was carried out on Blacklight at the Pittsburgh Supercomputing Center (PSC). It is interesting to note that for the largest images the disk I/O wall-time, limited by the I/O bandwidth of the machine, overwhelms the total computing wall-time. The performance of the I/O subsystem shall be an important factor in the selection of machines for data visualization at this scale.

7. IMAGE AND ANIMATION VIEWING

Once large images or animation frames have been created, viewing them presents a separate problem. We use the GigaPan for doing this and give measures of performance.
of sharing and interactive viewing of large single images by streaming in real time the portions of images actually needed by the viewer, based on the viewers' current area of focus inside the image. To support this real-time streaming, the image is divided up and rendered into small tiles of multiple resolutions. The viewer pulls in only the tiles needed for a given view. Many mapping programs (e.g., Google Maps) use the same technique.

We have uploaded an example terapixel image\textsuperscript{12} of the redshift \(z = 4.75\) snapshot of the MassiveBlack simulation to the GigaPan Web site, which is run as a publicly accessible resource for sharing and viewing large images and movies. The dimension of the image is 1,440,000 \(\times\) 810,000, and the finished image uncompressed occupies 3.58 TB of storage space. The compressed hierarchical data storage in GigaPan format is about 15\% of the size or 0.5 TB. There is no fundamental limit to size, provided the data can be stored on a disk. It is possible to directly create the compressed tiles of a GigaPan, bypassing the uncompressed image as an intermediate step, and thus reducing the requirement of memory and disk storage. We leave this for future work.

On the viewer side, static GigaPan works well at different bandwidths; the interface remains responsive independent of bandwidth, but the imagery resolves more slowly as the bandwidth is reduced. 250 kilobits s\(^{-1}\) is a recommended bandwidth for exploring with a 1024 \(\times\) 768 window, but the system works well even when the bandwidth is lower.

An illustration of the screen output is shown in Figure 5. The reader is encouraged to visit the Web site to explore the image.

7.2. GigaPan Time Machine

In order to make animations, one starts with the rendered images of each individual snapshot in time. These can be

\textsuperscript{12} Image at http://gigapan.org/gigapans/76215/
Figure 6. GigaPan Time Machine animation of the E5 simulation. These are screen grabs from the GigaPan Time Machine viewer. In the left column we show eight frames (out of 1367 in the full animation) which illustrate the evolution of the entire simulation volume (at time intervals of 2 Gyr). The middle panel zooms in to show formation of the largest halo through merger event. The right panel shows some of the history of a smaller halo.

(A color version of this figure is available in the online journal.)

To solve this problem, we created a gigapixel video streaming and viewing system called GigaPan Time Machine (Sargent et al. 2010), which allows the user to fluidly explore gigapixel-scale videos across both space and time. We solve the bandwidth and CPU problems using an approach similar to that used for individual GigaPan images: we divide the gigapixel-scale video spatially into many smaller videos. Different video tiles contain different locations of the overall video stream at different levels.
of detail. Only the area currently being viewed on a client computer needs be streamed to the client and decoded. As the user translates and zooms through different areas in the video, the viewer scales and translates the currently streaming video, and over time the viewer requests from the server different video tiles which more closely match the current viewing area. The viewing system is implemented in Javascript+HTML and takes advantage of current browsers’ ability to display and control videos through the new HTML5 video tag.

The architecture of GigaPan Time Machine allows the content of all video tiles to be precomputed on the server; clients request these precomputed resources without additional CPU load on the server. This allows scaling to many simultaneously viewing clients and allows standard caching protocols in the browser and within the network itself to improve the overall efficiency of the system. The minimum bandwidth requirement to view videos without stalling depends on the size of the viewer, the frame rate, and the compression ratios. The individual videos in “Evolution of the Universe” (the E5 simulation, see below for link) are currently encoded at 25 Frames per Second (FPS) with relatively low compression. The large video tiles require a continuous bandwidth of 1.2 megabits s\(^{-1}\) and a burst bandwidth of 2.5 megabits s\(^{-1}\).

We have uploaded an example animation\(^{13}\) of the E5 simulation, showing its evolution over the interval between redshift \(z = 200\) and \(z = 0\) with 1367 frames equally spaced in time by 10 Myr. Figure 6 is the screen grabs of the Gigapan Time Machine viewer. Again, the reader is encouraged to visit the Web site to explore the image.

8. CONCLUSIONS

We have presented a framework for generating and viewing large images and movies of the formation of structure in cosmological SPH simulations. This framework has been designed specifically to tackle the problems that occur with the largest data sets. In the generation of images, it includes parallel rasterization (for either shared and distributed memory) and adaptive pixel filling, which leads to a well-behaved filling rate at high resolution. For viewing images, the GigaPan viewers use hierarchical caching and cloud-based storage to make even the largest of these data sets fully explorable at high resolution by anyone with an Internet connection. We make our image-making toolkit publicly available, and the GigaPan web resources are likewise publicly accessible.

This work was supported by NSF Awards OCI-0749212, AST-1009781, and the Moore foundation. The following computer resources were used in this research: Kraken (NICS), Blacklight (PSC), and Warp (Carnegie Mellon University). Development of this work has made extensive use of the Bruce and Astrid McWilliams eScience Video Facility at Carnegie Mellon University.

REFERENCES

Boylan-Kolchin, M., Springel, V., White, S. D. M., Jenkins, A., & Lemson, G. 2009, MNRAS, 398, 1150
Bryan, G. L., & Norman, M. L. 1999, in Numerical Astrophysics: Proceedings of the International Conference on Numerical Astrophysics 1998 (NAP98), held at the National Olympic Memorial Youth Center, Tokyo, Japan, March 10–13, ed. S. M. Miyama, K. Tomisaka, & T. Hanawa (Astrophys. Space Sci. Lib. vol. 240; Boston, MA: Kluwer Academic), 19
Carlson, J., & White, M. 2010, ApJS, 190, 311
Colberg, J. M., Krughoff, K. S., & Connolly, A. J. 2005, MNRAS, 359, 272
DeGraf, C., Di Matteo, T., Khandai, N., et al. 2011, MNRAS, submitted (arXiv:1107.1254)
Dekel, A., Birnboim, Y., Engel, G., et al. 2009, Nature, 456, 451
Di Matteo, T., Colberg, J., Springel, V., Hernquist, L., & Sijacki, D. 2008, ApJ, 676, 33
Di Matteo, T., Khandai, N., DeGraf, C., et al. 2011, ApJL, submitted (arXiv:1107.1253)
Di Matteo, T., Springel, V., & Hernquist, L. 2005, Nature, 433, 604
Dolag, K., Meneghetti, M., Moscardini, L., Rasia, E., & Bondi, A. 2006, MNRAS, 370, 656
Fraedrich, R., Schneider, J., & Westermann, R. 2009, IEEE Trans. Vis. Comput. Graphics, 15, 1251
Gnedin, N. 2011, http://sites.google.com/site/frithome/Home
Hernquist, L., & Springel, V. 2003, MNRAS, 341, 1253
Hilbert, S., Hartlap, J., White, S. D. M., & Schneider, P. 2009, A&A, 499, 31
Hunter, J. D. 2007, Comput. Sci. Eng., 9, 90
Jenkins, A., Frenk, C. S., Pearce, F. R., et al. 1998, ApJ, 499, 20
Keres, D., Katz, N., Weinberg, D., & Dave, R. 2005, MNRAS, 363, 2
Monaghan, J. I. 1992, ARA&A, 30, 543
Porter, T., & Duff, D. 1984, SIGGRAPH Comput. Graph., 18, 253
Price, D. J. 2007, PASA, 24, 159
Sargent, R., Bartley, C., Dille, P., et al. 2010, Fine Intl. Conf. on Gigapixel Imaging for Science, http://repository.cmu.edu/gigapixel/22/
Shin, M., Tray, H., & Cen, R. 2008, ApJ, 681, 756
Sijacki, D., Springel, V., Di Matteo, T., & Hernquist, L. 2007, MNRAS, 380, 877
Springel, V. 2005, MNRAS, 364, 1105
Springel, V., White, S. D. M., Jenkins, A., et al. 2005, Nature, 435, 629
Springel, V., Yoshida, N., & White, S. D. M. 2001, New Astron., 6, 79
Szalay, T., Springel, V., & Lemson, G. 2008, Microsoft eScience Conference (arXiv:0811.2055)
Turk, M. J., Smith, B. D., Oishi, J. S., et al. 2011, ApJS, 192, 9
Wadsley, J. W., Stadel, J., & Quinn, T. 2004, New Astron., 9, 137
Zahn, O., Lidz, A., McQuinn, M., et al. 2007, ApJ, 654, 12

\(^{13}\) http://timemachine.gigapan.org/wiki/Evolution_of_the_Universe