HuMoRS: Huge models Mobile Rendering System

Marcos Balsa Rodríguez CRS4   Marco Agus CRS4   Fabio Marton CRS4   Enrico Gobbetti CRS4*

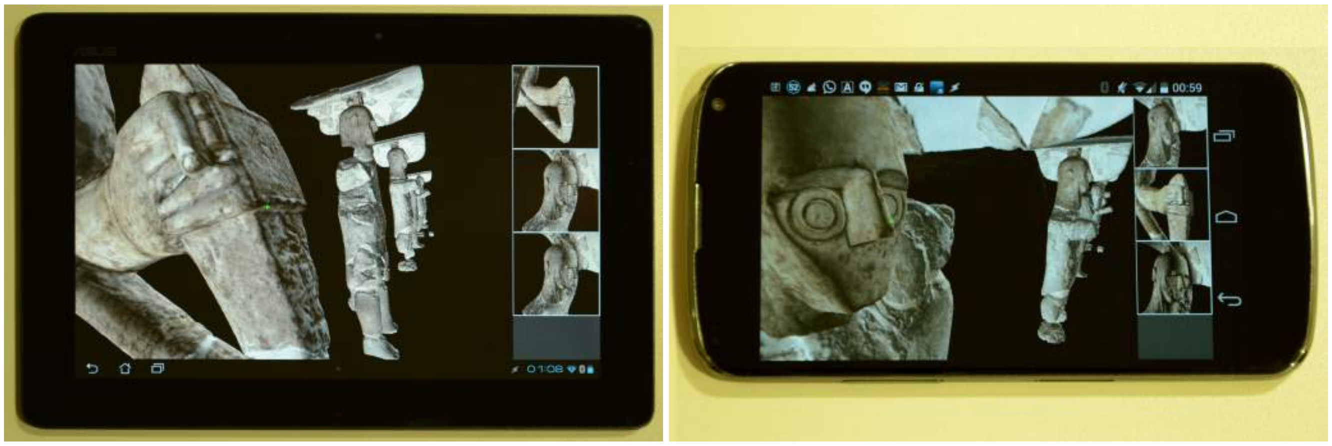

Figure 1: *Remote virtual exploration of a scene composed by several high resolution statues on a Asus TF201 tablet (left image), and on a LG Nexus 4 smartphone (right image). During interaction, models are adaptively downloaded from the network, and knowledge of the currently rendered scene is exploited to automatically center a rotation pivot for the camera controller and to propose context-dependent precomputed viewpoints.*

Abstract

We present *HuMoRS*, a networked 3D graphics system for interactively streaming and exploring massive 3D mesh models on mobile devices. The system integrates a networked architecture for adaptive on-device rendering of multiresolution surfaces with a simple and effective interactive camera controller customized for touch-enabled mobile devices. During interaction, knowledge of the currently rendered scene is exploited to automatically center a rotation pivot and to propose context-dependent precomputed viewpoints. Both the object of interest and the viewpoint database are resident on a web server and adaptive transmission is demonstrated over wireless and phone connections in a Cultural Heritage application for the exploration of sub-millimetric colored reconstructions of stone statues. We report also on a preliminary user-study comparing the performances of our camera navigation method with respect to the most popular Virtual TrackBall implementations, with and without pivoting.

CR Categories: I.3.2 [Computer Graphics]: Graphics Systems—Distributed/Network Graphics I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques;

Keywords: mobile graphics, massive models, camera control, cultural heritage

*CRS4 Visual Computing, POLARIS Ed. 1, 09010 Pula, Italy www: `http://www.crs4.it/vic/` e-mail: {`mbalsa|magus|marton|gobbetti`}`@crs4.it`

1 Introduction

The increased availability and performance of mobile graphics terminals, including smartphones and tablets with high resolution screens and powerful GPUs, combined with the increased availability of high-speed mobile data connections, is opening the door to a variety of networked graphics applications. In this world, native apps or mobile sites coexist to reach the goal of providing us access to a wealth of multimedia information while we are on the move.

There are in the world today more than 4.6 billion mobile phone subscriptions, of which about 2 billions with internet access, and the ability to exchange information is continuing to grow exponentially [Ellison 2010], with multimedia data taking a large share.

Cultural heritage valorization and cultural tourism are one of the sectors that are benefiting from this evolution, as the new ICT(Information and Communication Technologies) technologies allow to fit into a context of strong global competition, as they provide effective means to cover the pre-visit (documentation), visit (immersion) and post-visit (emotional possession) phases [Economou and Meintani 2011].

Nowadays, most applications of mobile devices in this sector are, however, focused to provide aids for navigation inside museal spaces [Filippini-Fantoni et al. 2011; Rubino et al. 2013] or to substitute the audio guides with enriched 2D multimedia content, and they do not provide support for the inspection of 3D artworks, such as sculptures or bas-reliefs. In this context, for museum promotion, it would be of great interest to provide tools for planning the visit of statue collections, allowing potential visitors to explore with the now ubiquitous tablets or smartphones the detailed 3D digital representation of artworks. To serve this goal, networked, user-friendly, flexible, scalable systems are required, and many challenges need to be addressed in parallel [Kuflik et al. 2011]. Of particular importance is the need to present information at the highest possible visual quality, in order to convey as much as possible the aura of the original artifact.

In this paper, we present a networked framework, dubbed *HuMoRS*, for streaming and interactive exploring huge highly detailed surface models on mobile platforms. The system aims at efficiently integrating web-based communication components, scalable multiresolution structures, adaptive rendering techniques, and simple and effective touch-based user interfaces. In our approach, 3D models and associated information are stored in a web server and streamed in real-time on mobile devices (tablets and smartphones). The mobile rendering application, implemented as an Android app, is controlled by a simple user interface, which combines an interactive camera controller, to incrementally explore the 3D model, with an interactive point-of-interest selector (see Fig. 1). During interaction, the camera controller exploits knowledge of the currently rendered scene to automatically center the trackball pivot and to propose context-dependent precomputed viewpoints in the neighborhood of the current view.

Our system combines and extends a number of state-of-the-art results. Although not all the techniques presented here are novel in themselves, their elaboration and combination in a single networked system for high resolution exploration of 3D artworks on mobile devices is non trivial and represents a substantial enhancement to the state-of-the-art. Two of the main novel contributions are the context-dependent camera controller and point-of-interest selectors, which are able to take context-based decisions based on an adaptive structure maintained on the mobile device and streamed from the web server. Our simple 3D camera controller extends the Two Axis Valuator Trackball [Chen et al. 1988; Zhao et al. 2011] with automatic pivoting, i.e, an automatic way to determine a good center of rotation based on the current view. In this way, we obtain a user interface for inspecting complex objects which is general, predictable, robust, scalable, smooth, and intuitive.

Our scalable implementation supports giga-triangle-sized scenes composed by different models and hundreds of points-of-interest on different Android platforms, including middle-level smartphones and tablets. Moreover we demonstrate the effectiveness of the proposed interface with a preliminary user-study comparing the time performances with respect to the most typical Virtual Trackball implementations, with and without pivot positioning.

2 Related work

Creating an interactive system for exploration of massive 3D models on mobile platforms requires combining and extending state-of-the-art results in a number of technological areas. In this section we briefly discuss the methods that most closely relate to ours. For a wider coverage, we refer the reader to well-established surveys on 3D interaction techniques [Jankowski and Hachet 2013], massive model rendering [Dietrich et al. 2007; Gobbetti et al. 2008; Yoon et al. 2008], and mobile graphics [Capin et al. 2008] for further details.

Massive model rendering on mobile platforms Exploring massive models on mobile devices is still a hot research topic: much of the work in model distribution has focused so far on compression of mesh structures [Jovanova et al. 2008; Blume et al. 2011; Niebling et al. 2010], but most methods are CPU-bound and spend a great deal of rendering time computing a view-dependent triangulation prior to rendering, making their implementation in a mobile setting particularly challenging. With the increasing raw power of GPUs, the currently higher-performance methods typically reduce the per-primitive workload by pre-assembling optimized surface patches [Cignoni et al. 2004; Borgeat et al. 2005; Goswami et al. 2013], and this kind of approaches has been demonstrated to work on mobile devices in the context of point-based rendering [Balsa Rodriguez et al. 2012], image-based mesh representations of scenes that can be parameterized as isometric quad patches [Gobbetti et al. 2012] and triangle meshes [Balsa Rodriguez et al. 2013]. In our system we employ a general multiresolution structure based upon tetrahedral space partitioning specifically tailored for mobile devices [Balsa Rodriguez et al. 2013].

Interactive exploration on mobile platforms In the context of visualization of complicated scenes, users require interactive control to effectively explore the data. Most of the work in this area is connected to camera/object motion control [Jankowski and Hachet 2013]. Variations of the virtual trackball [Chen et al. 1988; Shoemake 1992; Henriksen et al. 2004], which decompose motion into pan, zoom, and orbit, are the most commonly employed approaches. In order to solve the *lost-in-space problem* and avoid collisions with the environment, Fitzmaurice et al. [2008] have proposed the Safe Navigation Technique, which, however, requires explicit positioning of rotation pivot, and thus needs precise pointing. Moreover, motion decomposition and pivot positioning can be difficult for novice users. For this reason, a number of authors have proposed techniques for effective object inspection that constrain viewpoint by using precomputed/authored camera properties [Burtnyk et al. 2006; McCrae et al. 2009; Marton et al. 2012]. However, most of these systems require that the user has direct control over the visualization space, but this solution can be ineffective when the visualization area is very small, like it happens in smartphones or model-details. To solve this problem, Decle and Hachet [Decle and Hachet 2009] proposed an indirect method based on strokes for moving 3D objects in a touch screen mobile phone, while Kratz et al. [Kratz and Rohs 2010] introduced an extension of the virtual trackball metaphor, which is typically restricted to a half sphere and single-sided interaction, to actually use a full sphere, by employing the "iPhone Sandwich" hardware extension which allows for simultaneous front-and-back touch input. However, latter solutions are suited for small scale models and they employ fixed center of rotation in the barycenter. An automatic pivoting method has been recently presented by Trindade and Raboso [Trindade and Raposo 2011], but the method requires access to depth buffer, and it is not easily customizable to all mobile systems. Furthermore, their method computes the rotation center as intersection of the viewing vector with the object surface, and it suffers from discontinuities when complex models with sharp features are considered. Our automatic centering method overcomes these problems by considering a screen-space stochastic sampling of the visible surface, and it computes the pivot as a weighted filter of a random set of 3D visible points around the view target.

Image-assisted exploration Thumbnail-bars are tools which enable the navigation of a dataset by employing a set of precomputed images. At any given time, one image of the dataset can be selected by the user as current focus. The focus image can be regarded as the current "position" where the user sits within the dataset. The exact meaning of selecting a focus depends on the application: in an image viewer, the focus can be the image being shown at full resolution in the main area of the screen; in a 3D navigation application [Pintore et al. 2012], where images are linked to viewpoints (the physical camera positions of the shot), the focus image can drive the setup of the virtual camera used to render the 3D scene inside the main screen area. Often, these images are also linked to additional information, which is displayed when the user reaches them, as an alternative to the usage of hot-spots [Andujar et al. 2012]. The organization of the images in this kind of tools can be considered a challenging problem in a networked scenario, since simple grid layout approaches do not scale up well enough with the number of images. A trend consists in clustering images hierarchically, according to some kind of image semantic, like combining time and space [Ryu et al. 2010], spatial image-distances [Jang et al. 2009], or a mixture of them [Mota et al. 2008] to automatically or interactively compute image-clusters. Most of these works strive to identify good clustering for images, rather than good way to dynamically present and explore the clustered dataset. Our approach is navigation-oriented and tailored for mobile devices: it is organized in a way that, in any moment, users can decide to change the point of view by browsing a limited number of thumbnails organized in a scroll list and automatically chosen according to the similarity to the current view point.

3 System overview

While our methods are of general use, the **HuMoRS** system is motivated by a *Digital Cultural Heritage* project, which aims to present collections of digital sculptures, and covers aspects ranging from 3D digitization to exploration. We designed the HuMoRS system architecture by integrating the following components: a preprocessing component which builds multiresolution databases starting from high resolution triangle meshes, a web server for storing and streaming 3D models and associated information, and a mobile client integrating an adaptive multi-resolution renderer and a simple and effective user-interface (see section 4).

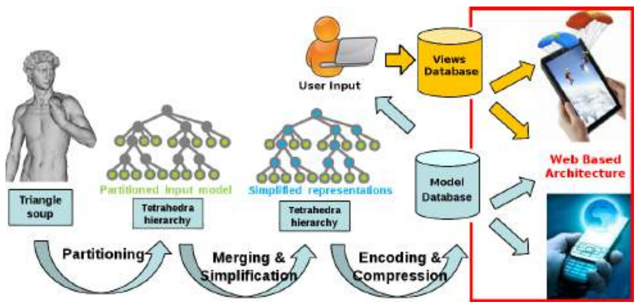

Figure 2: ***Multi-resolution processing pipeline.***

Model preparation In order to ensure real-time performance on large datasets presented at high resolution on mobile devices, we employ specialized multiresolution structures and adaptive algorithms.

Specifically, the renderer and the user-interface subsystem share structure and work on a view-adapted representation of the dataset based on a variation of Adaptive TetraPuzzles [Balsa Rodriguez et al. 2013]. The construction process of the multiresolution database starts from a *triangle soup*, i.e., a flat list of triangles with direct vertex information, together with a list of boundary vertices. The process is composed of three main phases schematized in figure 2: a first one where the dataset is partitioned into a tetrahedra hierarchy organized as diamonds [Weiss and De Floriani 2010], a second phase where data is simplified in a bottom-up fashion to build inner node representations, and a final phase where geometric data is compressed for streaming. The quantized vertex coordinates are encoded together with normals and colors into a compact 64bit representation suitable for direct rendering, where 3 bytes are used for position, 2 bytes for normal and 3 bytes for color. Position is parameterized with 4 barycentric coordinates, and normals are encoded using the *octahedron normal vector* approach [Meyer et al. 2010], which maps unit vectors to two parametric coordinates. In order to maximize data correlation for the entropy coding, color is transformed to the YCoCg reversible format [Malvar et al. 2008], and all of the vertex attributes (i.e., position, normal and color) are deinterleaved and separated into their components and stored as a sequence of streams. Building the multiresolution database takes about half an hour on an Intel Core i7 for a $70Mtri$ model, see Fig. 9.

Additional information is stored in the server in order to enable image-based navigation. Through a manual authoring process the user defines a set of interesting views on the model, and a thumbnail of the viewpoint is generated and stored together with the associated view matrix.

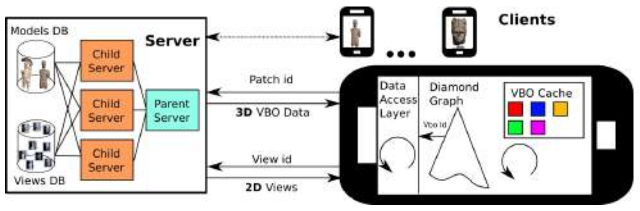

Figure 3: ***Client-server architecture.***

Client-server architecture The HuMoRS networked communication system is composed by the following components: a server for storing the models databases and related information, and a client for interactive rendering (see Fig. 3).

An abstraction layer handles the communication process through HTTP 1.1, relying on the CURL library for efficient communication. A persistent connection is maintained, due to the streaming nature of the communication, to avoid reconnection overhead.

A simple module for Apache2 is in charge of handling HTTP requests, which is built upon a local database to efficiently locate the requested data. Berkeley DB is used for storage, accessing and caching data in the server side due to its open source license and its matureness as embeddable database. On the server side, a custom Apache module implements a connectionless protocol based on HTTP which receives queries composed of database name and node identifier. This module extracts the query parameters, retrieves the corresponding data from the DB, and sends back either node's data or an empty message if it is not present. On the server side is also present a database of precomputed view snapshots. On a client request the server sends to the user the view thumbnails which will be used for image based navigation. Rendering is performed as what has been proposed by Balsa et al [Balsa Rodriguez et al. 2013]. Taking into account the mobile architecture constraints, the rendering engine has been designed to minimize fragment processing while feeding the GPU with large geometry batches using cache optimized indexed triangle strips. At each frame, depending on the viewing parameters and a given fixed screen space tolerance, the client performs an adaptive rendering of the multiresolution model. For this purpose, the client relies on a hierarchical multiresolution representation of the model that is incrementally refined depending on the navigation. In RAM memory, we maintain the cache of tetrahedra compact geometries, which are indexed through the diamond graph. The cache implements a LRU policy that maximizes the reuse of nodes while enforcing a resource usage below a given limit. The retrieval of data is performed through an asynchronous data access layer which encapsulates the data fetching mechanism and avoids blocking the application when the requested data is not yet available. This thread is also responsible for performing the decompression from the entropy coded version to the compact GPU friendly representation.

For each vertex buffer object the tetrahedra corners are bound to a vertex shader which transforms data expressed in quantized local

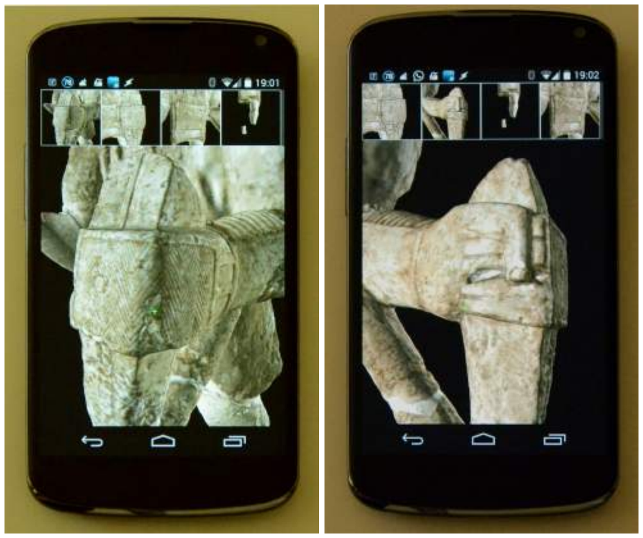

Figure 4: ***Detail of model interactively rendered on a Nexus 4 smartphone.*** *This 70Mtriangles model is colored using post-restoration color data. Note how our compression preserves extremely high quality details in shape, normal, and colors.*

barycentric coordinates into world coordinates. The transformation is given by this equation $\mathbf{v} = ||\mathbf{M}|| \cdot ||\mathbf{c_0 c_1 c_2 c_3}|| \cdot |\mathbf{v_b}|$, where $\mathbf{M}$ is the model matrix, $\mathbf{c_i}$ represent the corner ith while $\mathbf{v_b}$ is the vector of the 4 barycentric coordinates. Attribute decoding cost is negligible with respect to the other work performed by the shader (in particular, transformation, projection, and shading). Fig. 4 illustrates the quality of rendering that can be achieved using compressed data.

4 User interaction

The design of our method has taken into account requirements gathered from domain experts, as well as our analysis of related work (see Sec. 2).

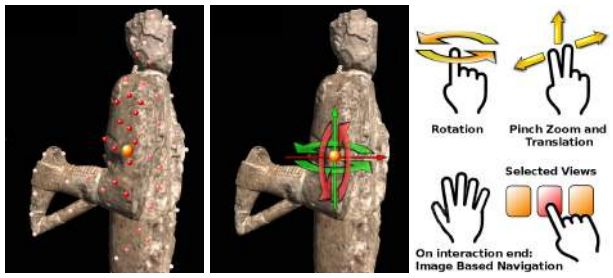
Rotation
Pinch Zoom and Translation
Selected Views
On interaction end: Image Based Navigation

Figure 5: ***Auto-centering and Interaction.*** *Left: automatic pivot computation performed by a weighted averaging of a uniform point sampling of the model in that moment observed by the camera (lighter samples have lower weights). Center: rotations are performed by dragging and mapping displacements according to the classical Virtual Trackball.Right: interaction gestures: one finger for rotation, two fingers for pinch zoom and translation. When interaction stops a set of precomputed views is presented to the user.*

Since we deal with decorated and highly detailed Cultural Heritage objects, like statues and bas-reliefs, we had to take into account the fact that they present information at multiple scales (global shape and carvings), which could require sub-millimetric model precision. This carving information should be clearly perceivable at all scales, and should be magnified for close inspection. Furthermore, camera navigation should provide real-time feedback, in order to engage users providing them the sense of control, and support smooth and seamless object inspection, going back and forth from shape inspection to detail inspection. The user interface should thus also be perceived as simple and immediately usable. Given the limited size of display, the visualized object should not be obstructed by fingers and other interaction widgets, and finger movements should be limited in order to reduce user's effort during the exploration. Finally, since the application has to work in mobile settings, all context-dependent operations have to be designed so as to work on locally maintained structures, as querying the server for information would introduce too much latency on mobile networks.

To fulfill these requirements, we designed a user interface composed by a simple and effective metaphor (**ACeViT**: Auto-Centering Virtual Trackball) for interactive camera motion, and a context-based point-of-interest selector for moving the camera to precomputed viewpoints, see Fig. 5 and Fig.6.

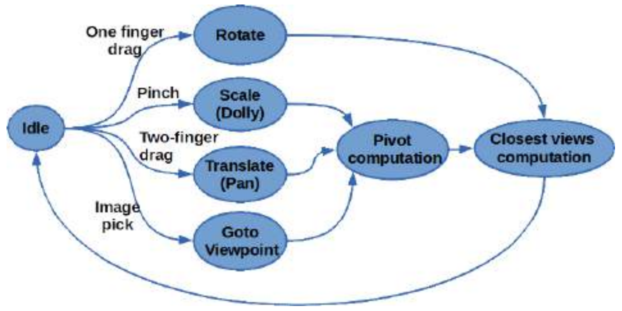

Figure 6: ***Interaction states.*** *This diagram shows the various state transitions that compose the user interface.*

4.1 Auto-Centering Virtual Trackball

During the last two decades many devices, interfaces and algorithms have been designed and proposed to map user input to virtual camera motions [Christie and Olivier 2009], aiming to maximize user performance and comfort during interactive explorations. The majority of navigation methods are devoted to typical desktop systems, in which user focus is directed to a monitor screen surface, and the input is given by employing 2D pointing devices, like mice, pads or joysticks. In these scenarios in which the work area size is comparable to the view area size, direct interaction methods are normally considered, and many exploration metaphors can be designed to fulfill the system requirements given by the data to explore and the user needs.

The widespread diffusion of touch devices, such as tablets or smartphones, has made people used to touch-based user interfaces based on direct manipulation of on-screen objects. 3D variations of the well-known 2D multi-touch RST technique, that allows the simultaneous control of Rotations, Scaling, and Translations from two finger inputs are becoming increasingly used as viewpoint manipulation technique in this context [Kurtenbach et al. 1997; Jankowski and Hachet 2013]. While no real standard for 3D touch-based interaction exists [Keefe and Isenberg 2013], touch surfaces are now so common that people are encouraged to try to interact with them using typical 2D gestures, which is an important aspect for reducing training time. Even if the mapping between 2D and 3D motions is non-trivial and varies from a user interface to the next, users are encouraged to learn by trial and error while interacting.

It should be noted, however, that in the case of small mobile touch screens standard co-located interaction techniques are not always effective due to occlusion problems since fingers can occlude the scene during motion, and simpler incremental techniques which reduce the user input need to be considered. It is therefore important to design the technique so as to avoid the need for precise colocation.

According to previous user analysis [Bade et al. 2005], the Two-Axis Valuator [Chen et al. 1988] appears to be the best incremental 3D rotation technique in terms of speed and satisfaction in standard settings, and it was considered as the base for our method. This interface maps 2D device motions to two axes of the view coordinate system, both orthogonal to the view-vector of the camera through a center of rotation [Shuralyov and Stuerzlinger 2011]. Specifically, horizontal mouse movement Δu is mapped to rotations about the up-vector of the camera and vertical mouse movement Δv is mapped to a rotation about the vector perpendicular to the up-vector and the view-vector (see figure 5). Then diagonal device movements are mapped to a combined rotation about both axes. The center of rotation (pivot) can be defined in different ways: traditional trackballs fix the pivot as the center of the object bounding sphere, but many systems give the user the possibility to manually define the rotation center as a 3D point in the object surface. Both solutions can be tedious for users, since in the first case they are forced to many motion corrections, while in the latter they have to change operation mode when they want to select a new pivot position. In our method, users do not have to take care of controlling the rotation center, since it is automatically placed. Whenever a panning or rotation interaction ends (see Fig. 6), the pivot position is computed according to the current projection and viewing matrices (P,V) and a stochastic sampling $\Sigma = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}$ of current visible points on the surface. The following weighted sum is employed:

$$\mathbf{c} = \sum_i \gamma(u_i, v_i, w_i) \cdot \mathbf{p}_i \qquad (1)$$

where $(u_i, v_i, w_i) = PV \cdot \mathbf{p}_i$ are the NDC coordinates of the projected point, and $\gamma(u_i, v_i, w_i) = \gamma(u_i) \cdot \gamma(v_i) \cdot \gamma(w_i)$ is a 3D separable Gaussian filter, giving the maximum weight to the current view matrix target position, and lower values when the samples are in the peripheral areas of the viewport. With respect to depth, the Gaussian is centered at the near plane, as to give to closest points the maximum contribution. The random sampling and the distance-based weighting ensure the avoidance of abrupt pivot changes when the user performs multiple small rotations. In order to suit different hardware settings, a variety of implementations can be considered. An efficient GPU-based screen-space implementation can be obtained by performing a stochastic sampling of the depth buffer and weighted accumulation of samples, by employing typical multi-pass pyramid methods derived from image processing [Strengert et al. 2006]. A model-based implementation instead can be applied to platforms where fragment processing and GPU feedback are not well supported or have performance issues (e.g., many mobile phones). In our implementation, we perform all computations in CPU and, since the models are rendered using patches (e.g., [Cignoni et al. 2004]), the sampling algorithm is applied to individual patches during hierarchy traversal and results are combined at run-time according to the renderer selection.

4.2 Image-based navigation

We provide the user also with a context-based guided navigation relying on nearest point-of-interest selection. This alternative control method allows the user to explore the object by traveling through a set of previously defined interesting views. On startup, the application loads a list of precomputed view points of the scene, which is organized into a KD-tree in order to provide fast searches. For each

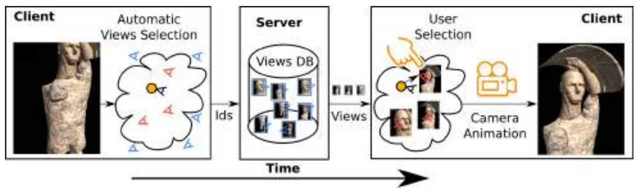

Figure 7: ***Views selection process.*** *When the camera stops, the application identifies the best closest views and posts a request to the server which sends back the selected view thumbnails. Finally the user selects the desired view and an animation moves the camera to the requested position.*

view point, a view matrix is stored together with the url of the corresponding view point thumbnail. Whenever an interaction ends, a list of best view candidates is computed and a separate thread is in charge of gathering the associated thumbnails to each view from the server. A small two level LRU cache in the client alleviates the latency of thumbnail retrieval for already visited viewpoints. The first cache is in charge of handling image requests to the server, containing various dozens of compressed images. The second level caches 2D textures, and must handle at least the number of simultaneous views that will be visible at the same time. Computation of best view set if performed in two steps. First, the closest views with respect to current view position are retrieved performing a Knn search. The resulting set of views is further filtered to select the views subset that best approximate the current view point. For that purpose, a uniform point sampling of the visible model is performed. The resulting point set is projected both from current view and from the candidate view in order to compute pointwise 2D distances. The views with lowest distances are then selected and presented to the user as a grid selection interface shown on the screen side, see Fig. 1.

The user can then navigate the model by clicking on the various thumbnails, thus starting an animation that will take the observer from its current position to the target view point, see Fig. 7.

The approach can be furthermore extended with static overlays, for authoring annotations and presenting heterogeneous enriched multimedia information, like required in typical cultural heritage applications [Marton et al. 2014].

5 Results

The **HuMoRS** system has been successfully tested in a variety of mobile platforms, in particular for the exploration of a set of 3D highly detailed models derived from the 3D scan acquisition of the statues of Mont'e Prama, ancient stone sculptures created by the Nuragic civilization of Sardinia, Italy, see Fig. 4, 8. The 3D models of these statues are highly detailed and often made of a few disconnected parts, posing important problems to navigation techniques. See Bettio et al. [2013; 2014] for details on the Mont'e Prama dataset.

5.1 System performance

The mobile client was implemented on Android 4.4 using C++, OpenGL ES 2.0 and Qt. The Qt library is in charge of handling UI events and GL context creation providing good portability for Android, Windows, Linux or iOS. We evaluated the rendering performance of the system on a number of inspection sequences on a variety of devices that represent both mid-class and top-class SoC(System on Chip) currently available in the market, see Table 1 for device characteristics.

Model	CPU	GPU	RAM	Screen
Nexus 7	NVIDIA Tegra 3 4x1.3Ghz	GForce ULP	1GB	7” 800x1280 0.9MPix
ASUS TF201	NVIDIA Tegra 3 4x1.3Ghz	GForce ULP	1GB	10.1” 1232x800 0.9Mpix
Nexus 4	Qualcomm Snapdragon S4 Pro 4x1.5Ghz	Adreno 320	2GB	4.7” 768x1280 0.9Mpix
Nexus 5	Qualcomm Snapdragon 800 4x1.5Ghz	Adreno 330	2GB	4.95” 1080x1920 2Mpix
Asus TF701	NVIDIA Tegra 4 4x1.9Ghz	GeForce ULP 72 core	2GB	10.1” 2560x1600 4Mpix

Table 1: ***Device characteristics.*** *Hardware characteristics of the devices used for testing.*

Device	Rendering speed	Download speed	Decode speed	Whole view refine(data size,time)	Close view refine(data size,time)
ASUS TF201 / Nexus 7	10 Mtri/s	11.2Mbps	1MB/s	4.6MB, 7s	37MB, 57s
Nexus 4	20 Mtri/s	18Mbps	1.2MB/s	4.6MB, 5s	37MB, 41s
Nexus 5	20 Mtri/s	20Mbps	1.4MB/s	7.4MB, 8s	45MB, 40s
Asus TF701	25 Mtri/s	24.8Mbps	1.5MB/s	8.2MB, 8s	65.8MB, 59s

Table 2: ***Hardware performance.*** *Performance results for the various hardware configurations.*

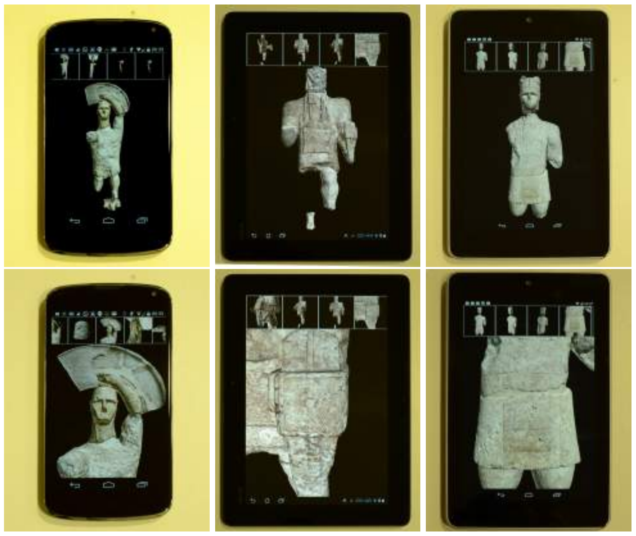

Figure 8: ***Various levels of detail of a statue.*** *Images correspond to Nexus 4, Asus TF201 and Nexus 7, respectively. Top: View of the whole statue. Bottom: A close-up of the statue showing small-scale details.*

Rendering performance Models were rendered using a target resolution of 0.3tri/pixel on screens with less than 2Mpix and 0.2tri/pixel when there are more than 2Mpix, leading to graph cuts of 150 nodes on average, with approximately $8Ktri/node$. Navigation was interactive with negligible interaction delays for all datasets, with frame rates constantly above 20 fps, for the Tegra 4 and SnapDragon processors, while Tegra 3 handled above 10 fps. The multiresolution structure used for rendering is also exploited for object queries during interaction, proving successful, as camera transformation computation, and pivot calculation, in our camera controller always took below 10% of the frame time. The sessions were designed to be representative of typical mesh inspection tasks and to heavily stress the system, including rotations and rapid changes from overall views to extreme close-ups. The qualitative performance of our adaptive renderer is also illustrated in an accompanying video, that shows live recordings of the analyzed sequences. During the tests the average triangle count was 1-$3Mtris$ per frame, depending on screen coverage and model refinement. The mesured performance shown average frame rates of 10-30 fps on most devices, while Tegra 3 devices performance was about 5-15fps, see Table 2. When the user is interacting, the maximum triangle budget is adjusted to ensure a minimum of 10 fps. As demonstrated in the video, performance is perfectly adequate for interactive inspection tasks, while providing extremely high representation quality. An example is presented in Fig. 4.

We have also performed tests on a scene composed of 10 statues, ranging from $40Mtri$ to $70Mtri$ each, resulting in about $600Mtri$, see Fig. 1. Performance was measured between 4-10 fps on Tegra 3 devices, and 10-20 fps on the other tested devices. Average triangle count per frame depends mostly on the statue in foreground, while other statues covering less than $1/4$ of the viewport add no more than $60Ktri$ each, representing about 10-30% overhead.

All the measures correspond to a rendering viewport of 5/6th of the screen, excluding the area used for image-based navigation, ranging from $0.7Mpixels$ to $3.3Mpixels$.

Streaming performance The latency time needed to download the data at the application start-up and to refine the model during the exploration is the most critical bottleneck affecting mobile devices, and it is independent from the rendering thread only relying on the network bandwidth. Performance was measured on a wireless connection using a Linksys WAP200 802.11b/g AP 54 Mbps, obtaining a peak performance of 28Mbps under a heavily shared environment.

For a *full view*, where the whole statue fits in the rendering viewport (see Fig. 8 top-left), about 4.6-8.2MB ($500Ktri$-$1.3Mtri$) were required for full refinement, depending on the rendering viewport resolution. In the case of a *detail view*, a close view of a small part of the statue (see Fig. 8 center-bottom), 37-65.8MB (1.6-$2.3Mtri$) are needed (see Table 2 for further details).

Data fetching is performed asynchronously in a separate thread, so it doesn’t interfere with interactive rendering performance. Data fetching over Wifi shows download speeds about 11-26Mbps, representing 40-88% of the available bandwidth. The decoding performance varies among the various devices ranging from 0.9MB/s to 1.5MB/s, thus determining the maximum streaming throughput, see Table 2. Achieving a fully refined *full view* of the statue requires about 5-8s depending on the screen resolution and the device performance when decoding. For a *detail view*, between 40-59s are required for full refinement. Nonetheless, thanks to the progressive refinement, within just few seconds the statues can be inspected with a reasonable quality.

Under UMTS/HSPA connection, we measured an average data fetch speed of 2.5Mbps over a measured peak performance of

3.4Mbps. Full refinement of a *full view* took about 20-50s, and 120-140s for a *detail view*.

All this times correspond to a fresh start with no cached data. However, during a typical inspection of the model, when the user gets to a new close-up view most data from the coarse representation is already present, thus requiring less time for full refinement.

5.2 User study

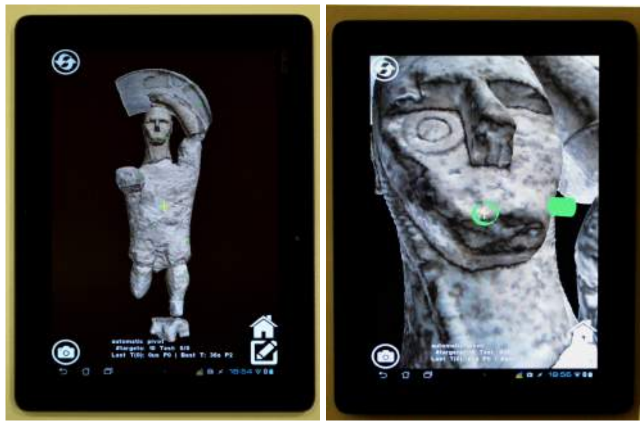

Figure 9: ***User study performed on a Nexus 7 tablet.*** *ACeViT compared to Virtual TrackBall with fixed pivot and with manual pivot positioning. Left: test scene composed of a detailed surface model composed of 70Mtri. Right: tasks consisted of reaching and shooting specific positions and orientations indicated by green cylinders.*

In order to test the effectiveness of the interaction method on mobile devices, we carried out also a preliminary performance evaluation of our auto-centering Virtual Trackball (AceViT) with respect to the standard Two Axis Evaluator Virtual Trackball with fixed pivot, and with manual selection of the pivot obtained by ray-casting against the scene on user request.

Setup The user tests were performed on an ASUS Nexus 7, see Table 1. All the interfaces were implemented by using the typical RST approach: one finger to generate rotations, two fingers to perform zoom (by pinching) or pan (by dragging), continuous pressure with one finger to perform pivot update. As testing scene, we considered a scene composed of a Boxer statue (see Fig. 9) consisting of $70Mtri$.

Tasks The experiments consisted in letting users try the three different manipulation controllers (ACeViT, TrackBall with fixed pivot and TrackBall with manual pivot) in the context of a point-and-shoot interaction task [Bade et al. 2005]. Participants were asked to point-and-shoot a small set of green cylinders, which had to be shoot through in order to get a positive hit. By forcing the user to align the camera view direction with that of the cylinder axis, we obtained a task composed of a global approaching phase and a later local step for aligning camera with the cylinder. The cylinder radius was adjusted in order to avoid trivial alignments so the second step was hard to skip (see Fig. 9 left). When the targets were precisely pointed and aligned with respect to a cross viewfinder, users could shoot them by pressing a touch button (see Fig. 9 right).

Subjects Ten subjects with ages ranging from 27 to 51 (mean 38.4 ± 7.9) were selected between the researchers of our research department. All persons involved had previous experience with 3D software systems and interfaces (particularly with Virtual Trackball interfaces).

Design Subjects were proposed the interfaces in randomized order. After a brief training with the touch interface, the measured tests consisted of shooting 5 targets, randomly selected from a list of 10 potential candidates, in order to avoid any bias due to a-priori knowledge of target positions. For a complete testing session, users needed times ranging from 180 to 280 seconds. The times for hitting all tasks for each interface were measured and recorded.

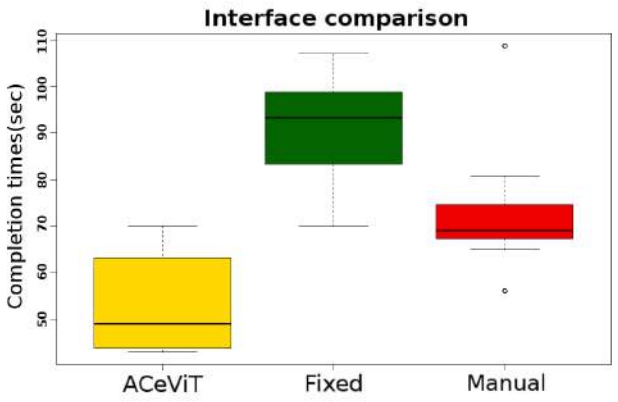

Figure 10: ***Performance comparison: Timings***

Analysis In summary, the complete test design consisted of 10 subjects, each one testing 3 camera controllers for a total of 30 time measurements. We performed a statistical analysis of completion times for the shooting tasks experiment. All computations were done by using the *R* package. Mean completion times were 53.68 ± 11.41 seconds for ACeViT, 73.64 ± 14.84 seconds for manual pivoting, and 90.95 ± 12.26 seconds for fixed pivoting. After performing a Shapiro-Wilk test for normality ($W = 0.948$, $p = 0.2101$), an analysis of variance revealed a significant effect of the interface ($F = 17.652$ and $p < 0.001$). Finally a post-hoc Tukey multiple comparisons of means revealed dramatic differences between ACeViT and fixed pivoting($p < 0.001$), important differences between ACeViT and manual pivoting ($p < 0.01$), and significant differences between manual pivoting and fixed pivoting ($p = 0.029$). Fig. 10 shows the boxplots of the task completion times, as rendered by the *R* package. Statistical analysis and boxplots showed that automatic pivoting improves the time performances of shooting tasks. Moreover, by observing and listening think-aloud comments during the experiments, it appeared evident that users felt very comfortable with ACeViT since they could explore in intuitive manner the complex scene, and they could perform tasks with less motion corrections with respect to standard Virtual Trackball implementations, with fixed and manual pivoting.

6 Conclusions

We have presented *HuMORS*, an interactive system for natural exploration of extremely detailed surface models on mobile devices. The system successfully integrates web components, scalable multiresolution structures, and adaptive rendering techniques. The mobile rendering application is decorated with an effective user interface, which combines an interactive camera controller, to incrementally explore the 3D model, with an interactive image-based point-of-interest selector. The system has been successfully tested in a variety of mobile platforms, in particular for the exploration of a set of 3D highly detailed models obtained with high resolution laser scanning of Cultural Heritage artworks. Furthermore, our auto centering camera controller has been compared with two consolidated Virtual Trackball implementations, collecting quantitative results from a series of tests on a mobile device involving 10 people. The camera controller appears to be intuitive and simple enough even for casual users who quickly understand how to browse complex models immediately. Our current work concentrates on extending the system for supporting bidirectional connection between multiple multimedia types as well as narrative contents.

Acknowledgments

This work is partially supported by the EU FP7 Program under the DIVA project (290277), by POR Sardegna 2007-2013 under the TADES project, and by *Soprintendenza per i Beni Archeologici per le Province di Cagliari ed Oristano (ArcheoCAOR)*. We also acknowledge the contribution of Sardinian Regional Authorities.

References

ANDUJAR, C., CHICA, A., AND BRUNET, P. 2012. Cultural heritage: User-interface design for the Ripoll monastery exhibition at the National Art Museum of Catalonia. *Computers and Graphics 36*, 1, 28–37.

BADE, R., RITTER, F., AND PREIM, B. 2005. Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In *Proc. Smart Graphics*, Springer, 138–150.

BALSA RODRIGUEZ, M., GOBBETTI, E., MARTON, F., PINTUS, R., PINTORE, G., AND TINTI, A. 2012. Interactive exploration of gigantic point clouds on mobile devices. In *Proc. VAST*, 57–64.

BALSA RODRIGUEZ, M., GOBBETTI, E., MARTON, F., AND TINTI, A. 2013. Compression-domain seamless multiresolution visualization of gigantic meshes on mobile devices. In *Proc. ACM Web3D*, ACM Press, 99–107.

BETTIO, F., GOBBETTI, E., MERELLA, E., AND PINTUS, R. 2013. Improving the digitization of shape and color of 3D artworks in a cluttered environment. In *Proc. Digital Heritage*. To appear.

BETTIO, F., JASPE VILLANUEVA, A., MERELLA, E., PINTUS, R., MARTON, F., AND GOBBETTI, E. 2014. Mont'e scan: Effective shape and color digitization of cluttered 3D artworks. *Submittted for publication.*

BLUME, A., CHUN, W., KOGAN, D., KOKKEVIS, V., WEBER, N., PETTERSON, R., AND ZEIGER, R. 2011. Google Body: 3D human anatomy in the browser. In *ACM SIGGRAPH Talks*, ACM, 19.

BORGEAT, L., GODIN, G., BLAIS, F., MASSICOTTE, P., AND LAHANIER, C. 2005. GoLD: interactive display of huge colored and textured models. *ACM TOG 24*, 3, 869–877.

BURTNYK, N., KHAN, A., FITZMAURICE, G., AND KURTENBACH, G. 2006. ShowMotion: camera motion based 3D design review. In *Proc. ACM I3D*, ACM, 167–174.

CAPIN, T., PULLI, K., AND AKENINE-MOLLER, T. 2008. The state of the art in mobile graphics research. *IEEE CG&A 28*, 4, 74–84.

CHEN, M., MOUNTFORD, S. J., AND SELLEN, A. 1988. A study in interactive 3-D rotation using 2-D control devices. In *Proc. ACM SIGGRAPH*, ACM, 121–129.

CHRISTIE, M., AND OLIVIER, P. 2009. Camera control in computer graphics: models, techniques and applications. In *ACM SIGGRAPH ASIA Courses*, ACM, 3:1–3:197.

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 TOG 23*, 3, 796–803.

DECLE, F., AND HACHET, M. 2009. A study of direct versus planned 3d camera manipulation on touch-based mobile phones. In *Proc. MobileHCI*, ACM, 32–35.

DIETRICH, A., GOBBETTI, E., AND YOON, S. 2007. Massive-model rendering techniques: A tutorial. *IEEE CG&A 27*, 6 (nov/dec), 20–34.

ECONOMOU, M., AND MEINTANI, E. 2011. Promising beginnings? evaluating museum mobile phone apps. In *Proc. Rethinking Technology in Museums Conference*, 26–27.

ELLISON, S., 2010. Worldwide and u.s. mobile applications, storefronts, and developer 2010–2014 forecast and year-end 2010 vendor shares: The "appification" of everything. Doc. 225668 Market Analysis. IDC Corporate US.

FILIPPINI-FANTONI, S., MCDAID, S., AND COCK, M. 2011. Mobile devices for orientation and way finding: the case of the british museum multimedia guide. In *Proc. Museums and the Web*.

FITZMAURICE, G., MATEJKA, J., MORDATCH, I., KHAN, A., AND KURTENBACH, G. 2008. Safe 3D navigation. In *Proc. ACM I3D*, ACM, 7–15.

GOBBETTI, E., KASIK, D., AND YOON, S. 2008. Technical strategies for massive model visualization. In *Proc. ACM SPM*, ACM, 405–415.

GOBBETTI, E., MARTON, F., BALSA RODRIGUEZ, M., GANOVELLI, F., AND DI BENEDETTO, M. 2012. Adaptive Quad Patches: an adaptive regular structure for web distribution and adaptive rendering of 3D models. In *Proc. ACM Web3D*, ACM Press, 9–16.

GOSWAMI, P., EROL, F., MUKHI, R., PAJAROLA, R., AND GOBBETTI, E. 2013. An efficient multi-resolution framework for high quality interactive rendering of massive point clouds using multi-way kd-trees. *The Visual Computer 29*, 1, 69–83.

HENRIKSEN, K., SPORRING, J., AND HORNBÆK, K. 2004. Virtual trackballs revisited. *IEEE TVCG 10*, 2 (Mar.), 206–216.

JANG, C., YOON, T., AND CHO, H.-G. 2009. A smart clustering algorithm for photo set obtained from multiple digital cameras. In *Proc. ACM SAC*, ACM, 1784–1791.

JANKOWSKI, J., AND HACHET, M. 2013. A survey of interaction techniques for interactive 3D environments. In *Eurographics STAR*.

JOVANOVA, B., PREDA, M., AND PRETEUX, F. 2008. MPEG-4 Part 25: A generic model for 3D graphics compression. In *Proc. IEEE 3DTV*, IEEE, 101–104.

KEEFE, D., AND ISENBERG, T. 2013. Reimagining the scientific visualization interaction paradigm. *Computer 46*, 5 (May), 51–57.

KRATZ, S., AND ROHS, M. 2010. Extending the virtual trackball metaphor to rear touch input. In *Proc. IEEE 3DUI*, IEEE, 111–114.

KUFLIK, T., STOCK, O., ZANCANARO, M., GORFINKEL, A., JBARA, S., KATS, S., SHEIDIN, J., AND KASHTAN, N. 2011. A visitor's guide in an active museum: Presentations, communications, and reflection. *JOCCH 3*, 3, 11:1–11:25.

KURTENBACH, G., FITZMAURICE, G., BAUDEL, T., AND BUXTON, B. 1997. The design of a gui paradigm based on tablets, two-hands, and transparency. In *Proc. ACM SIGCHI*, ACM, ACM, 35–42.

MALVAR, H. S., SULLIVAN, G. J., AND SRINIVASAN, S. 2008. Lifting-based reversible color transformations for image compression. 707307–707307–10.

MARTON, F., AGUS, M., GOBBETTI, E., PINTORE, G., AND BALSA RODRIGUEZ, M. 2012. Natural exploration of 3D massive models on large-scale light field displays using the fox proximal navigation technique. *Computers & Graphics 36*, 8 (December), 893–903.

MARTON, F., BALSA RODRIGUEZ, M., BETTIO, F., AGUS, M., JASPE VILLANUEVA, A., AND GOBBETTI, E. 2014. Isocam: Interactive visual exploration of massive cultural heritage models on large projection setups. *JOCCH*. To appear.

MCCRAE, J., MORDATCH, I., GLUECK, M., AND KHAN, A. 2009. Multiscale 3D navigation. In *Proc. I3D*, ACM, 7–14.

MEYER, Q., SUESSMUTH, J., SUSSNER, G., STAMMINGER, M., AND GREINER, G. 2010. On floating-point normal vectors. *Computer Graphics Forum 29*, 4, 1405–1409.

MOTA, J. A., FONSECA, M. J., GONÇALVES, D., AND JORGE, J. A. 2008. Agrafo: a visual interface for grouping and browsing digital photos. In *Proc. ACM AVI*, ACM, 494–495.

NIEBLING, F., KOPECKI, A., AND BECKER, M. 2010. Collaborative steering and post-processing of simulations on hpc resources: Everyone, anytime, anywhere. In *Proc. ACM Web3D*, ACM, 101–108.

PINTORE, G., GOBBETTI, E., GANOVELLI, F., AND BRIVIO, P. 2012. 3DNSITE: A networked interactive 3D visualization system to simplify location recognition in crisis management. In *Proc. ACM Web3D*, ACM Press, 59–67.

RUBINO, I., XHEMBULLA, J., MARTINA, A., BOTTINO, A., AND MALNATI, G. 2013. Musa: Using indoor positioning and navigation to enhance cultural experiences in a museum. *Sensors 13*, 12, 17445–17471.

RYU, D.-S., CHUNG, W.-K., AND CHO, H.-G. 2010. PHOTOLAND: a new image layout system using spatio-temporal information in digital photos. In *Proc. ACM SAC*, ACM Press, 1884–1891.

SHOEMAKE, K. 1992. ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. In *Proc. ACM SIGGRAPH*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 151–156.

SHURALYOV, D., AND STUERZLINGER, W. 2011. A 3d desktop puzzle assembly system. In *Proc. 3DUI*, IEEE Computer Society, 139–140.

STRENGERT, M., KRAUS, M., AND ERTL, T. E. 2006. Pyramid methods in gpu-based image processing. In *Proc. VMV*, 169–176.

TRINDADE, D. R., AND RAPOSO, A. B. 2011. Improving 3d navigation in multiscale environments using cubemap-based techniques. In *Proc. SAC*, ACM, 1215–1221.

WEISS, K., AND DE FLORIANI, L. 2010. Simplex and diamond hierarchies: Models and applications. In *Eurographics STAR*, Eurographics Association, H. Hauser and E. Reinhard, Eds., 113–136.

YOON, S., GOBBETTI, E., KASIK, D., AND HA, D. M. 2008. *Real-time Massive Model Rendering*, vol. 2 of *Synthesis Lectures on Computer Graphics and Animation*. Morgan and Claypool, August.

ZHAO, Y. J., SHURALYOV, D., AND STUERZLINGER, W. 2011. Comparison of multiple 3d rotation methods. In *Proc. IEEE VECIMS*, IEEE, 1–5.