Mixed-reality geometric algebra animation methods for gamified intangible heritage

Papagiannakis George¹, Greasidou Elissavet¹, Trahanias Panos¹ and Tsioumas Michalis²

¹University of Crete, Computer Science Department, Voutes Campus, 70013, Heraklion and Foundation for Research and Technology Hellas, 100 N. Plastira Str., 70013, Heraklion, Greece
²IST Ephorate of Prehistoric and Classical Antiquities, M. Alexandrou, 54646, Thessaloniki, Greece
Mixed-reality geometric algebra animation methods for gamified intangible heritage

Papagiannakis George, Greasidou Elissavet, Trahanias Panos and Tsioumas Michalis

Abstract:
In this work we propose a novel, mobile, cross-platform open framework specifically for animated, skinned and life-size augmentations of real scenes with virtual characters. Already ten years ago we had demonstrated such technologies on the site of ancient Pompeii however, with a VR/AR closed-system running on a powerful mobile workstation hidden on a backpack. Nowadays we exhibit similar functionality running on modern smartphones running on an open-source, cross-platform gamification framework. Due to the openness of this framework we integrate a modern AR markerless camera tracking toolkit coupled with our novel algorithm for animation interpolation blending based on geometric algebra, which would not be otherwise easily achievable only in commercial, off-the-self toolkits. Thus our approach allows for open-standards research in virtual character simulation as well as novel AR augmentations for the creative and cultural heritage industries. Based on recent Mixed Reality software 3D platforms and 3D content creation pipelines, intangible digital heritage environments can benefit significantly from such versatile, efficient and robust 3D gamification methodologies.

1. INTRODUCTION
Rapid, recent advances in computer graphics, user interfaces and mobile computing, empower the creative industries with the inception of new mixed reality experiences that enhance the way we acquire, interact, play and display 3D information within the world that
surrounds us [5]. In [3] we introduced augmented reality (AR) virtual characters reenacting dramaturgical intangible heritage in an archaeological heritage site (ancient Pompeii) in real-time. Latest innovations in AR [4],[5] focus on innovative methodologies to blend information from our senses and mobile devices in myriad ways that were not possible before. Recently in [1] and previously in [3] we introduced open frameworks for graphics research & teaching for both AR and VR application development while in [2] we employed the former framework for geometric algebra (GA) simulations in VR desktop 3D character animation blending.

In this paper we describe the porting of a novel GA animation method specifically for mobile AR, as extensions of our previous results [2] that allow for a more efficient animation interpolation performance based on a new general mathematical framework suitable for outdoors AR and heritage sites, as shown in the figure below. In addition, we employ life-size virtual characters as 3D augmentations of the real heritage environments, in real-time, as shown in [3], [5], [14], [16]. In the following sections we commence with a summary of the previous work in the field (Section 2), then proceed with a brief overview of GA (Section 3). In Section 4 we describe the mobile AR architecture on which we integrate our GA algorithm illustrated in Section 5. The main novelties are then summarized in Section 6 and the implementation details are illustrated in Section 7 with Conclusions and future work described in the final Section 8.

Figure 1. Real-time AR characters in indoors as well as outdoors environments

2. PREVIOUS WORK

Ten years ago in [3] we had successfully demonstrated a complete methodology for real-time mobile mixed reality systems that featured realistic, animated virtual human actors (with clothes, body, skin, face 3D simulation) who augmented real heritage environments (the archaeological site of ancient Pompeii) and re-enacted staged storytelling dramas. Although initially targeted at cultural heritage sites, the paradigm was by no means limited to such subjects. However, portability, usability and AR form factor of enabling technologies (a laptop on a backpack was then required) were major impediments for wider adoption of that suite of AR technologies and algorithms.
Modern s/w AR systems have been constantly progressing since then coupled with the ‘ peace dividend’ of the result of “smartphone wars” [5]. Furthermore, already component-based frameworks are now being investigated for such mobile outdoor AR applications [4].

[11] has already illustrated the connection between quaternions and GA bivectors as well as the different Clifford algebras with degenerate scalar products that can be used to describe quaternions and dual quaternions. [8] have shown the use of all three GA models with applications from computer vision, animation as well as a basic recursive ray-tracer. [12] employed the GA conformal model (refer to next section for an explanation of the different GA models) to encode position and pose interpolation for skinning. In this work we compare previous quaternion algebraic models with the Euclidean GA model as a fast and robust alternative forward-looking representation (more on this in future work). The use of dual quaternions for enhanced character skinning was primarily been employed by [9] in computer graphics and later by [7] for character animation blending. In [10] the matrix operator and linear algebra was employed in order to enhance skinning and vertex collapse during animation blending. In this paper we aim to verify that as quaternions and dual quaternions can be expressed as GA rotors (although the later is not explored in this work) they can be equally efficient and computationally viable as alternatives for character animation blending in modern mobile devices.

2.1 Ancient Agora of Thessaloniki – Roman Forum

The Ancient Agora – Roman Forum [6], the administrative centre of ancient Thessaloniki, occupied an area about two hectares in the heart of the city. Its construction began at the end of the 2nd century A.D. on the site of an older forum dating from early Imperial times (parts are illustrated in Figure 1, right).

The complex was arranged around a rectangular paved square. There were stoas on three sides, each of which consisted of a double row of columns and provided direct access to a surrounding zone of public buildings. The southern stoa stood on a vaulted substructure (cryptoporticus) - a double arcade which lay partly underground, making use of the natural slope of the land. To the south, along the whole of the cryptoporticus, lay a row of shops fronting the ancient shopping street which ran along the north side of present-day Philippou St. Off this street lay minor entrances to the square, while the latter opened north, to the present-day Olympou St. In the middle of the east wing, on the site of an earlier council - chamber, a building for public performances was erected, which, on the basis of the inscription and the statues of Muses found there, must have functioned...
as an Odeon. Although there is a modern museum currently on-site, the Ephorate is currently exploring with this work modern AR technologies as novel curation enhancements for the site visitors and personnel.

3. THE EUCLIDEAN GEOMETRIC ALGEBRA MODEL

Geometric algebra (GA) is a powerful and practical computational framework for the representation and solution of geometrical problems. Its roots can be traced to the 19th century and mainly the work of W. Clifford that unified previous algebras from Grassmann and Hamilton [11], [8].

The basic computational elements in GA are subspaces and various products involving them. The most basic real, m-dimensional linear space $V_m$, contains in 3D ($V_3$) the basis vectors $\{e_1, e_2, e_3\}$ and is called the Euclidean, Vector space 3D GA model. There is also the 4D vector Homogeneous GA model and the 5D vector Conformal GA model for which the reader might refer to [8] for complete coverage.

Although coordinates are necessary for input/output, computations in GA are performed directly on subspaces, in a coordinate-free manner. In this work we focus on the Euclidean, Vector GA model whose vectors are characterized by the same attitude and magnitude algebraic properties as in linear algebra (LA) and can be multiplied to produce a scalar using the inner product: $e_1 \cdot e_2$.

In GA, a common way to construct a higher dimensional oriented subspace from vectors is to use a product that constructs the span of vectors called the outer (or wedge) product and denoted by $\wedge$. Such a subspace is called a blade and the term $k$-blade is used for a $k$-dimensional subspace. E.g. a vector is a 1-blade, an area is a 2-blade, a volume a 3-blade etc. where 1,2,3 are referred as grade. The highest grade element is called pseudoscalar and often denoted as $I$.

To compute $a \wedge b$ for two vectors $a$ and $b$ (forming a plane) with scalar coefficients $\alpha, \beta$:

$$a \wedge b = (\alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3) \wedge (\beta_1 e_1 + \beta_2 e_2 + \beta_3 e_3) =$$

$$(\alpha_1 \beta_2 - \alpha_2 \beta_1) e_1 \wedge e_2 + (\alpha_2 \beta_3 - \alpha_3 \beta_2) e_2 \wedge e_3 + (\alpha_3 \beta_1 - \alpha_1 \beta_3) e_3 \wedge e_1 \quad (1)$$

Hence any 2-blade in 3D space is decomposed onto a basis of 3 elements (also called bivectors): $\{e_1 \wedge e_2, e_2 \wedge e_3, e_3 \wedge e_1\}$. In a similar manner the outer product of 3 vectors results in a trivector subspace $\{e_1 \wedge e_2 \wedge e_3\}$. This space of scalars, vectors, bivectors and trivectors with $+$ and $\wedge$ as operations is called the Grassmann algebra of 3D space.
Clifford introduced a new product, the geometric product that unified inner and outer products and formed the basis of GA, encompassing also the quaternions of Hamilton as basic algebraic elements (rotors) and not a separate, special case.

The geometric product between the previous vectors $a, b$ in a common plane with unit 2-blade $I$ is defined as: $ab = a \cdot b + a^\wedge b$ which is an element of mixed grade (a 0-grade scalar accompanied by a 2-blade where the whole is called a multivector). The geometric product is the cornerstone in reflection and rotation calculations, especially in its exponential representation (with $\phi$ the angle from $a$ to $b$):

$$ab = a \cdot b + a \wedge b = |a||b|(\cos \phi + I \sin \phi) = |a||b|e^{I\phi}$$  \hspace{1cm} (2)

The pseudoscalar $I$ is also often used to indicate duality in GA, so that for a given element $A$, its dual is denoted as $A^* = IA$, e.g. the $e^*1 = e123e1 = e23$ (where the geometric product of $e1, e2, e3$: $e1e2e3 = e123$ and $e2i = 1$).

GA has a special way to represent orthogonal transformations, more powerful than using orthogonal matrices: sandwiching a multivector between $R = ab$ and its inverse, this $R$ is called a rotor, and will be mostly used in section 5.

4. MOBILE, CROSS-PLATFORM MR ARCHITECTURE

The central part of our open-source system is the OpenGL Geometric Application (glGA) framework [1]. The glGA framework is a lightweight, shader-based, C++ Computer Graphics (CG) framework developed for educational as well as research purposes. It is a cross platform application development framework and currently supports most major desktop and mobile platforms. It provides our main integration system with the functionalities of a) loading shaders, sounds, animations and textures, b) loading both 3D static and animated, skinned character models and c) animating and rendering these models.

We integrate in our framework the Metaio SDK [1] for markerless, SLAM-based 3D camera tracking. In order for the real-scene to be tracked, offline 3D feature-point maps were created with the mobile Metaio Toolbox SLAM authoring application and subsequently processed by the Metaio Creator application. The tracking functionality provided by the Metaio SDK is utilized as an external component plugin in glGA. The main AR glGA application consists of two iOS specific classes: a) the AppDelegate class whose main role is to handle state transitions within the application (i.e. launch time initialization, handling transitions to and from the background) and b)
the ViewController class which is responsible for handling the View, the object that provides the visual representation of the application’s content. It also responds to events that occur within it (e.g. gestures), collaborating with three singleton classes: a) the MetaioManager mentioned above, b) the SystemManager which is responsible for initializing and updating the system and c) the GestureManager which, as the name suggests, handles the gestures recognized in the ViewController.

4.1. AR authoring Pipeline

In order to be able to utilize the 3D tracking functionality provided by the Metaio SDK, we need to have a 3D feature-based map of the environment in which we want the animated character to be rendered as real-time, life-size augmentation. As a first step, we use the Metaio Toolbox [1] application to create the 3D tracking map. However, the Metaio SDK requires that the tracking data are read from an XML file format, in order to be able to configure the tracking system and the coordinate systems. For that reason, we subsequently use the Metaio Creator [1] application, which for a given 3D map that was created with the Metaio Toolbox application, it produces an XML file (the tracking configuration) and an .f3b file (encoded tracking features).

Finally, we need to add these two files exported from the Metaio Creator to our project, compile it and run it on the iOS device. The whole process takes no more than a few minutes to be completed in any indoors or outdoors real environment. However, if this process ends here, then the augmentation gets arbitrarily attached at a random origin in the 3D world with a default transformation matrix (not life-size). Hence we devised a simple real-time gesture-based authoring so that the virtual character is once transformed by the AR scene author (scale, rotation, translation) to the desired position and size and this result is saved once, so that subsequent startups of the app will automatically employ that predefined, authored 3D scene transformation. This last but important step is described in the following section.

4.2 AR runtime simulation and gesture-based life-size augmentation

After the view controller has loaded its view hierarchy into memory the viewDidLoad method is called and all the system’s components are initialized. The SystemManager, MetaioManager (Metaio SDK) and GestureManager instances are created. The main application loop consists of updating the system’s variables and then rendering the scene. The rendering starts after the onSDKready callback function is
triggered which notifies us that the Metaio SDK is ready (e.g. splash screen is finished). We use the camera image as the background. This is done by rendering a quad geometry with the camera image that we acquired from Metaio SDK as its texture. Since, we need to render the camera image we register the onNewCameraFrame callback method which delivers the next camera image. We trigger that callback method exactly once in the application loop by calling the requestCameraImage method, and then we use the camera image returned to update the background texture.

The Metaio SDK is also responsible for the 3D tracking. We call the render() method in the application loop to update the tracking state. Calling this method will not render anything as we initialize the Metaio SDK with the ERENDER_SYSTEM_NULL configuration since we only want its tracking and camera image capturing functionalities.

The application supports user interaction through gestures to transform the models that are rendered. When the user has transformed the scene to his liking, he can then save the transformations which will be available to load at any point in the current or future executions of the application. The following Table summarizes the supported gestures and their functionalities:

<table>
<thead>
<tr>
<th>Gesture</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan (one finger)</td>
<td>Translation along the X and Y axes</td>
</tr>
<tr>
<td>Pan (two fingers)</td>
<td>Rotation along the X and Y axes</td>
</tr>
<tr>
<td>Rotation</td>
<td>Z-axis Rotation</td>
</tr>
<tr>
<td>Pinch</td>
<td>Uniform Scale</td>
</tr>
<tr>
<td>Tap (double)</td>
<td>Load transformations from file</td>
</tr>
<tr>
<td>Long Press</td>
<td>Save transformations to file</td>
</tr>
</tbody>
</table>

Table 2 Transformations with Gestures

The ViewController together with the GestureManager are responsible for handling the gestures with the later also ensuring that only one gesture can be active at a given time.

Finally the 3D scene including the animated, skinned virtual character, is rendered using the glGA framework methods provided for that purpose. The Metaio SDK getProjectionMatrix method allows us to obtain the OpenGL projection matrix retrieved from the camera calibration. The getTrackingValues method provides a matrix containing the camera transformation matrix so that that the augmentation scene will be placed on the markerless tracking target. We only need to multiply this matrix with the transformations specified by the user to get the final view matrix.
5. MAIN GEOMETRIC ALGEBRA ALGORITHM FOR MOBILE IOS DEVICES

Our main algorithmic novelty lies in the employment of Euclidean GA rotors as fast, drop-in replacements for quaternion algebra, during animation orientation interpolation for mobile iOS devices. We provide a different approach to express quaternions as GA rotors and subsequently utilize the exponential rotor formula of Euclidean 3D GA as alternative to spherical quaternion interpolation as shown in the table below. For a complete derivation of previous desktop-only implementation of this algorithm the reader should refer to [2].

<table>
<thead>
<tr>
<th>Method to replace quaternions with GA rotors</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Method to express a quaternion as GA</td>
</tr>
<tr>
<td>%Rotor (MATLAB code):</td>
</tr>
<tr>
<td>Rs  =  srcQ.e(1) + srcQ.e(2)*(e2^e3) -</td>
</tr>
<tr>
<td>srcQ.e(3)<em>(e1^e3) + srcQ.e(4)</em>(e1^e2)</td>
</tr>
</tbody>
</table>

After expressing the existing input and output quaternions as GA rotors, we were able to fast interpolate between them using the rotor exponential formula that allows to interpolate between two orientations RA and RB, via their geometric product, using n interpolation steps, on an axis of rotation $\theta$ and the Euclidean 3D GA pseudoscalar $e_3$ over and angle $\phi$:

$$ R_A R^n = R_B \Rightarrow R = e^{-I_3 \frac{\theta}{(2n)}} $$

(3)

5.1 Algorithm Pipeline

Step 1:

- Input: quaternion orientation of existing animation & skinning frameworks
- Output: express with different ways input and output quaternions as GA rotors and members of the algebra:

$$ i = e113 = e2^e3 $$
$$ j = e213 = -e1^e3 $$
$$ k = e313 = e1^e2 $$
$$ ijk = -1 $$
$$ q = 1 + i + j + k $$

(4)
\[
R = ba = b \cdot a + b \wedge a = \cos \phi - I \sin \phi \\
R = e^{-i\phi} = e^{-i\frac{\phi}{2}}
\]

Step 2:
- Input: source and destination Rotors in GA
- Output: interpolate between them using the closed-form formula in Euclidean 3D space and code in Table 1 above.

Step 3:
- Input: interpolated GA rotors
- Output: translate interpolated rotor to matrix or quaternion for real-time rendering of rigged characters (results shown in Figure 1: left image shows standard (dual) quaternion interpolation, right image our drop-in GA rotor replacement) based on code as shown in the Table below.

```
% GA Rotor interpolation (MATLAB code):

Rtot = Rdest/Rsrc

for i=0 : 0.125 : factor
    \[ R_i = \exp((i/2) * \text{sLog}(R_{tot})) \]
    \[ R_{int} = (R_i * \text{srcRotor} / R_i) \]
    draw(R_{int}, 'k')
end
```

### Table 2 GA Rotor interpolation

6. MAIN NOVELTIES

In this work we address the following research questions: a) can GA rotors be more efficient and faster than quaternions also for mobile devices? b) can we transfer our innovative character-based AR research methodologies from ten years ago [3] into modern mobile devices? Our work so far has illustrated the following comparative novelties and advantages of GA rotors to other methods (e.g. dual quaternions) as already explained in [2] this time adapted and ported within the constraints of mobile iOS devices. To the best knowledge of the authors, this is the first time that 3D life-size virtual characters (animated and deformed) are augmenting an archaeological site based on modern smartphones and tablets coupled with a GA animation method.
7. IMPLEMENTATION

Our AR application implementation is based on the glGA iOS port [1] and using for the camera tracking part the Metaio™ SDK (http://www.metaio.com/sdk) [13]. We have employed fully open-source APIs, 3D standard file formats and s/w libraries for all parts of the experiments and case studies (except the Metaio tracker): a) the OpenGLES 2.0 API was used for real-time, shader-based GPU rendering with the GLM mathematics library, b) the Collada 3D file format for skinned, animated virtual characters, c) the ASSIMP library for asset loading, and e) the Gaigen & extracts from the libGASandbox libraries for GA expressions in modern C++, using the LLVM compiler. For h/w we utilised iOS devices: iPad mini and iPhone 5s tablets and smartphone devices respectively.

We employed different articulated figures with variable skeleton joints and polygonal complexity (from 3 joints and 4 triangles up to 42 joints and 30000 triangles), employing vertex-shader skinning in all cases. Finally we adapted a Poser Pro collada character in our tests and also other characters from online tools can also be used (Mixamo, Autodesk Character Generator, MakeHuman) in both Collada as well as FBX format. In the figure below we illustrate our ongoing efforts for the next version of historical characters to populate the Ancient Agora site (currently under strict evaluation from the site archaeologists).

The ViewController class manages the view that is the application’s user interface. In the viewDidLoad method the gesture recognizers are created by allocating and initializing an instance of the appropriate concrete UIGestureRecognizer subclass for each one. Then they are attached to the view using the addGestureRecognizer method and each one is connected to an action method. These action methods are used to respond to each recognized gesture (i.e. compute relative transformations). The GestureManager class holds the transformations and ensures that no more than one gesture is recognized at each time. In the viewDidLoad method the SystemManager instance is also created an initialized.

In the SystemManager singleton class the scene (i.e. 3D models) is created with using the relative glGA Framework methods. The MetaioManager instance is also created. In the figure below the overall concept architectural diagram is provided for the glGAMR (glGA Mixed Reality) iOS app based on glGA and the Metaio SDK (Figure 2).

The MetaioManager singleton class creates the Metaio SDK instance and assigns the tracking configuration. It utilizes the functionality of the Metaio SDK and handles its callback methods. In the update method the current camera captured image is received by calling the requestCameraImage method which triggers the onNewCameraFrame callback method and it is then copied to a
texture for rendering as the background image of the application. The tracking state is also updated with the call to the SDK's render method. In the display method the OpenGL projection matrix retrieved from camera calibration and a matrix containing the pose transformation such that the rendered model will be placed on the markerless tracking target are obtained from the Metaio SDK getProjectionMatrix and getTrackingValues methods respectively. We only need to multiply the matrix obtained from the call to getProjectionMatrix with the transformations in the GestureManager (i.e. the ones specified by the user) to get the final view matrix. Then we render the 3D models using the functionality the glGA frameworks provides us with.

From the sample results illustrated in Figure 1 and Figure 5, it is evident that our method is indeed faster than current separate linear algebra (matrix based) and quaternion algebras as we avoid the step of converting back and forth between these two separate algebras and we can successfully augment indoors and outdoors scenes, with markerless, life-size based virtual characters as augmentations. Although further commercial game engines could be employed regarding the virtual characters (e.g. Unity Pro) such a solution would incur additional costs as well as not easily allowing for open research in virtual characters regarding (e.g. fast and efficient adoption of GA).
7.1. Animated virtual character authoring pipeline for intangible heritage simulations

Most 3D content creation pipelines employ a combination of in-house or proprietary software chain and 3D file formats in order to create and rig virtual characters. So it is really challenging for an independent developer to have a fully detailed and rigged character without these special tools. After some research and using available—and in parts easy to use-existing open and closed software tools we produced a three-stage pipeline to accomplish similar rigged, animated virtual character results but employing fully open-source, 3D file formats for asset management, namely the Collada™ format.

7.1.1 Stage 1: Designing the basic 3D virtual character mesh

The very first step involved a modern character-based content-creation tool to design quickly and efficiently a clothed virtual character. We employed the commercial Poser Pro™ (http://my.smithmicro.com/poser-pro-2014.html) but also tested Autodesk Character Generator™ (https://charactergenerator.autodesk.com), Mixamo™ (https://www.mixamo.com) and the open-source MakeHuman (http://www.makehuman.org) applications, for designing the basic, clothed 3D mesh for a virtual character (without a skeleton or animation). All performed similarly into producing a basic clothed, textured 3D mesh for a virtual character (example shown in the figure below).

Figure 3. Modelling the 3D virtual character of an ancient Roman lady

When the result met the archeologists expectations, we chose to export the 3D textured character mesh as a Collada XML file.
7.1.2 Stage 2: Correcting and preparing the 3D mesh for rigging and animation

The second stage involved the Autodesk Maya s/w suite, via the open-source OpenCOLLADA Importer, as necessary step to correct manually various exporting errors from the previous stage 1. Due to various Collada exporting issues, all previous s/w tools employed produced results that had to be corrected in Maya, namely regarding correcting texture coordinates, transparent meshes or even misplaced transformations on certain meshes.

7.1.3 Stage 3: Rigging, animating and exporting the virtual character

This was our pipeline final and easiest stage. All we needed was a browser to visit Mixamo’s site and their specific head to Rig Page. There we uploaded our zip file and we had to assign specific marks on our character from a two-dimensional point of view, corresponding to Mixamo’s autorigging semi-automatic algorithm (shown in the figure below). After this stage, we proceeded to the animation page. There we could select an animation from the provided database, preview it and finally export our complete, animated, rigged virtual character as a Collada file.

This concluded our Open-Pipeline containing all the steps in terms of creating, correcting, rigging and animating an open-format virtual character. In this point the character was ready for use in both our VR games, in Unity as well as glGA.

8. CONCLUSIONS AND FUTURE WORK

In this work we have shown first early results of our ongoing efforts to provide complete AR simulations (rendering and animation) of life-size virtual characters in outdoors scenes, without markers. Although a
commercial camera tracker was employed, our open computer graphics simulation framework empowered us to perform novel research at the virtual character level (not possible with commercial systems) and extend its applicability into novel AR heritage scenarios (also other camera trackers could be easily employed, although the Metaio one yielded best mobile performance). In this work we have employed GA rotors for life-size virtual character simulation in the AR heritage site of Ancient Agora – Roman Forum, Thessaloniki, for the first time, as shown in the figure below.

Our future work involves extending the above framework by expressing dual-quaternions in either the Euclidean 3D or the Conformal GA models and hence fully subsuming existing different algebras (linear algebra, (dual) quaternion) under a single representation in a GA algebraic framework, including translation and scaling transformations. Furthermore the Ephorate is exploring the possibility of releasing the above application (with the final historical 3D characters) as a complete mobile application available for the thousands visitors of the site.

In this manner we believe that GA is a forward-looking integrated algebraic framework not only limited to transformations for character animation and could be also be applied in real-time character rendering suitable to be integrated with low level APIs (e.g. Apple Metal architecture) or other 3D engines (e.g. Unity3D). Finally a complete VR reconstruction is also under construction for Serious Games type simulations, such as those already shown in [13].
ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 274669, the HIFI-PRINTER Marie Curie IEF project and the GSRT RealEstate project. We would also like to express our gratitude to Ms. Vasiliki Misailidou-Despotidou, Director of the IST Ephorate of Prehistoric and Classical Antiquities, Thessaloniki, Greece, the archaeologist Ms. Christina Moustantami and all the staff (engineers, guards etc.) from the IST Ephorate.

REFERENCES


