A case study of 3D technologies in higher education: scanning the Metalwork Collection of Museums Sheffield and its implications to teaching and learning

RODRIGUES, Marcos <http://orcid.org/0000-0002-6083-1303>, KORMANN, Mariza and DAVISON, Lucy

Citation:

RODRIGUES, Marcos, KORMANN, Mariza and DAVISON, Lucy (2011). A case study of 3D technologies in higher education: scanning the Metalwork Collection of Museums Sheffield and its implications to teaching and learning. In: 2011 International Conference on Information Technology Based Higher. IEEE, 1-6. [Book Section]

A Case Study of 3D Technologies in Higher Education: Scanning the Metalwork Collection of Museums Sheffield and its Implications to Teaching and Learning*

Marcos A Rodrigues[1], Mariza Kormann[1] and Lucy Davison[2]

[1]Geometric Modeling and Pattern Recognition Research Group, Communication and Computing Research Centre, Sheffield Hallam University, Sheffiedl, S1 2NU

[2]Museums Sheffield, Leader House, Surrey Street, Sheffield, S1 2LH (email lucy.davison@museums-sheffield.org.uk)

Abstract—This paper describes results from the fast 3D scanning project conducted at Sheffield Hallam University in collaboration with Museums Sheffield. We focus on the technological aspects that are required for fast scanning and discuss the steps in the process from scanning and noise removal to 3D post-processing and how the resulting 3D models can be made available on a standard web browser. We also discuss some implications to the teaching and learning of 3D technologies to undergraduate and postgraduate courses.

Index Terms—3d scanning, 3d post-processing, WebGL, browser

I. Introduction

3D technologies have a number of useful applications in several diverse areas such as games, animation, TV, medical applications, security, industrial monitoring and control to mention a few. Traditionally, higher education (HE) institutions have been the main enablers of 3D technologies bringing them to mainstream applications through research and development. Increasingly, high technology companies such as Google and Mozilla are driving the technology agenda and this is most apparent in recent years through their efforts to bring interactive 3D to the web browser. Typically, once the required background technology is available, companies are able to establish themselves. In turn, this creates markets for professionals that need to be educated and trained in 3D technologies. This evolutionary model creates opportunities for both industry and academia and it is important that HE institutions fast adapt their curricula and teaching methods to industry's training needs.

This paper describes results from the fast 3D scanning project conducted at Sheffield Hallam University in collaboration with Museums Sheffield and some implications to the teaching and learning of 3D technologies to both undergraduate and postgraduate courses. The major challenge for the project is the scanning of metal surfaces, which are notoriously difficult to scan. In the context of 3D scanning to preserve cultural heritage it is often easier to scan large non-reflective surfaces such as the structures of an ancient temple than stainless steel small objects. Optimal scanning conditions have to be found and a number of post-processing operations are required, some are discussed below. The focus of this paper is on the technological aspects that are required for fast scanning and discuss the steps in the process from scanning and noise removal to 3D post-processing and how the resulting 3D models can be made available on a standard web browser.

Most museums worldwide have space constraints and normally more items are kept in their stores than we see on display. However, merely displaying an object does not mean that it can be fully appreciated. In real physical displays items cannot be seen from all angles, as some surfaces of the objects remain hidden. Also, many items can only be appreciated from non-optimal distances for a variety of reasons such as security or the objects being too fragile to be handled. Worldwide, there has been a recent trend towards digitization of cultural heritage and making this accessible to the wider public. In this context, 3D technologies offer a unique way to make available large collections online where the viewer can interact with and close inspect each item.

The metalwork collection at Museums Sheffield has

* RODRIGUES, Marcos, KORMANN, Mariza and DAVISON, Lucy (2011). A case study of 3D technologies in higher education: scanning the Metalwork Collection of Museums Sheffield and its implications to teaching and learning.In: *2011 International Conference on Information Technology Based Higher.* IEEE, 1-6.

Designated status, which means it is recognised as internationally important. Staff have worked on a series of projects to increase access and understanding of this renowned collection. All these projects strive to ensure that researchers, academics, teachers and school children can explore significant items from the collection that are unavailable to them because of distance. The ability to offer lifelike 3D representations of objects will greatly enhance the experience of distance users, replicating as close as possible the experience of seeing the items in person. Museums Sheffield has the gallery space to display around 10% of its collections at one time. This means the majority of objects are beyond the reach of visitors. 3D imaging means that these stored collections can be used and enjoyed in a way simply not possible with physical displays.

In Section II we briefly discuss the options for generating 3D models. In Section III we describe the steps in scanning and reconstruction. In Section IV the technologies required to place 3D models on the web are discussed and Section V highlights some implications to teaching and learning. Finally a conclusion is presented in Section VI.

II. Generating 3D Models

3D models can be generated in a number of ways. Some options include:

- Using a 3D modelling package,
- Using transform graphs,
- Using a 3D scanner,
- Using a combined approach with any of the above.

A brief discussion on modelling packages and transform graphs is presented below while 3D reconstruction using a 3D scanner is discussed in the next section.

A. *Generating 3D Models Using a 3D Modelling Package*

A 3D scene can be generated using a number of different tools such as 3D Studio Max, Maya, and AutoCAD among others. The simplest way of creating a 3D model involves the drawing of primitive geometric shapes (e.g. triangle, rectangle, circle) and then extruding these in a desired direction. Extrusion is an essential operation in 3D modelling and all modelling packages offer this functionality. A simple 3D model can be created in a couple of minutes using any of those applications. Figure 1 depicts 3D models created from primitive shapes and extrusion from an earlier project in archaeological reconstruction from the Helike Project Collection, Aegion, Greece [1]. Texture mapping was added resembling the original colour of the objects; for instance, the first object had a rusty colour while it is known the second

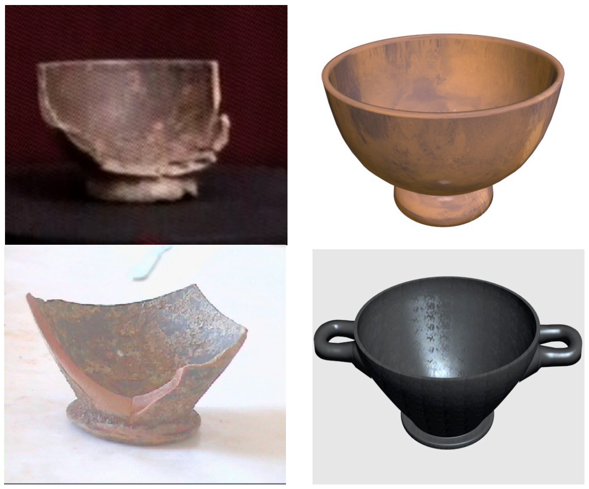
Fig. 1. Examples of 3D reconstruction in archaeology from sherds where the symmetry of the objects simplifies modelling. From the Helike Project Collection [1].

object to have been painted black. Note that shape symmetry greatly simplifies modelling but it is very difficult to capture the exact dimensions and imperfections existing in the real objects. Moreover, more complex scenes or intricate objects

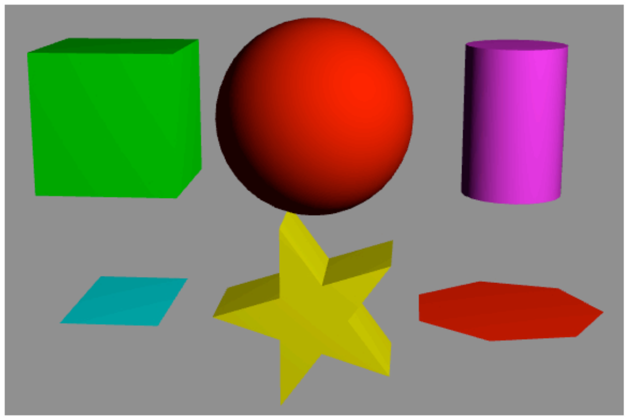
Fig. 2. Examples of synthetic 3D models derived from primitive shapes and transform graphs.

may demand an unspecified amount of time and effort. For the purposes of this project this option is not a realistic alternative as for most pieces in the collection it would be too laborious to build 3D models using this technique.

B. *Generating 3D Models Using Transform Graphs*

Transform graphs specify the colour, normals, effects, and the position of objects in 3D space. These normally are defined through low level programming language constructs (such as C) and thus, require a good knowledge of programming. OpenGL [2] is a low level programming language providing a number of constructs allowing programmers to create 3D models by defining and manipulating transform graphs. Some examples of 3D models are depicted in Figure 2; such objects are defined

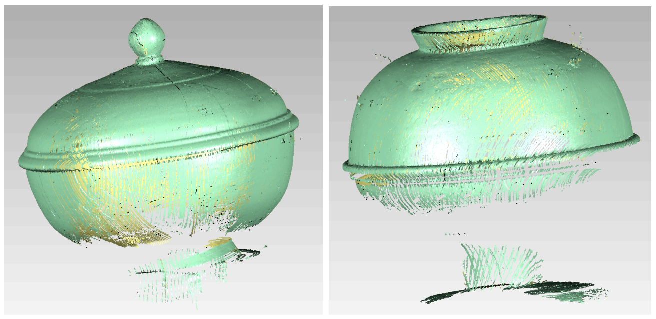

Fig. 3. On the left the top of an object is scanned. Right shows a scan from the bottom of the object. The angle of the incident light has been tightly controlled to reduce noise and unwanted reflections.

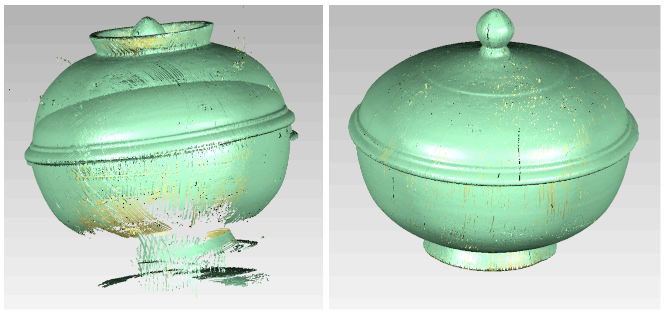

Fig. 4. Left, the two views of the same object do not naturally fit together after scanning. On the right, after cleaning the multiple views, 3D registration is performed aligning the two parts of the model together.

from primitives and transformations are performed on them such as extrusion, scale, rotation and translations.

For instance, a 3D model of a cube can be defined in OpenGL by specifying the position of 8 vertices in space. An example in shown below where one of its vertices coincides with the origin and the length of each side of the cube is 1:

```
glBegin(GL QUADS);
    glColor3f(0.0,1.0,0.0); // A Green Cube
    glVertex3f( 0.0, 0.0, 0.0);
    glVertex3f( 0.0, 1.0, 0.0);
    glVertex3f( 1.0, 1.0, 0.0);
    glVertex3f( 1.0, 0.0, 0.0);
    glVertex3f( 0.0, 0.0, 1.0);
    glVertex3f( 0.0, 1.0, 1.0);
    glVertex3f( 1.0, 1.0, 1.0);
    glVertex3f( 1.0, 0.0, 1.0);
glEnd();
```

Using an API (Application Programming Interface) such as Google's O3D [3] to define and display 3D models on a web browser allows one to work at a higher level than OpenGL. The reason is that O3D provides a wrapping around some basic OpenGL functions so many low level functions can be encapsulated into a higher-level call. For instance, the code for creating the 3D cube using O3D would look like:

```
function createShapes()
{
    var cube = o3djs.primitives.createCube(
        g pack,
        createPhongMaterial([0,1,0,1]), // green shaded
        Math.sqrt(2)); // The length of each side of the cube.
}
```

Although this is much simpler to write and maintain than the OpenGL version, this is still not an option considering the requirements of the project. It works well for simple geometric shapes, but for complex shapes such as the ones in the Metalwork Collection, the only alternative left is using a 3D scanner.

III. 3D Scanning and Reconstruction

The project aims to scan representative items from the Designated Metalwork Collection of Museums Sheffield. Since objects in the museum are physical expressions that do not have an equivalent CAD (Computer-Aided Design) model and most have very complex shapes, the use of a scanner is required for 3D reconstruction. We have tested structured light scanners using both white and near-infrared light using techniques described in [4],[5],[6],[7],[8]. We observed that due to the highly reflective surfaces of metallic objects, the resulting models are too noisy and deemed not appropriate. The major problem with structure light scanning is the angle of incident light that needs to be controlled to a small value. The multiple stripe patterns of structured light mean that some stripe patterns will always be at unfavourable angles. The solution is to use a single stripe scanner whose angle can then be closely controlled. For the task we used a Faro Arm with laser scanner [9] to scan 225 items from the collection.

There are several steps in the scanning process and some of the main intermediate results are illustrated in Figures 3—5:

1. Scan object surfaces producing a point cloud. Typically a point cloud will contain between 1 and 4 million points (*x*,*y*,*z*). Normally multiple views of the same object are necessary as depicted in Fig 3 where the top and bottom of the object were scanned.
2. Clear the noise from the surface of the scans by directly editing the point cloud and deleting the unwanted data points. It is necessary to have knowledge of the object's shape to guide this process.
3. Register (align) the various surfaces into a single 3D model as depicted in Fig 4. Normally this is accomplished in two steps from a coarse multiple point registration to a fine global registration.
4. Fuse the multiple point clouds into a single cloud. Some operations performed here include detection and removal of outliers, and deletion of redundant points within a set threshold. We chose the minimum distance between points to be 0.25mm.
5. Convert the point cloud into a triangulated mesh as depicted on the left of Fig 5. This is necessary for obtaining correct illumination effects and texture

mapping.

6. Add texture to the model. Textures are defined from image files, which can be synthetically generated or from a photograph of the object.

Once a model is complete with texture mapping it is then ready to be deployed and visualized.

IV. Making 3D models interactive

One difficulty with 3D scenes is their integration into HTML files such that other e-learning materials can be incorporated in the usual way. Web browsers are not designed to deal with large 3D data files that require vast amount of computational resources and thus, palliative solutions are often used. For instance, a common option is to develop a Flash animation from a 3D scene but the price is that this is not fully interactive and not as immersive as provided by a standalone playback application. Other options exist such as producing “flat” 3D environments by taking several pictures covering 360 degrees (from the centre of a room for instance) and stitching them together and then playing back in the web browser. This is very limited as the end result is simply a series of pictures projected on the surface of a cylinder rotating around its centre axis.

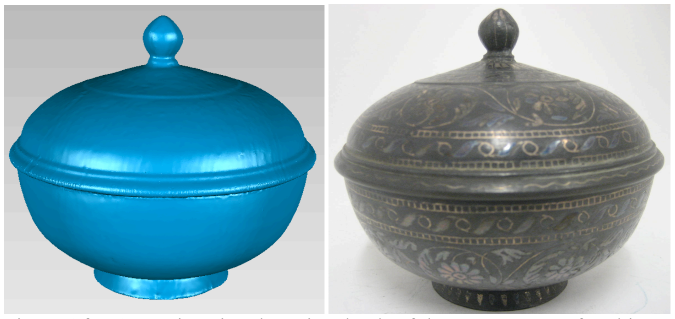
Fig. 5. After 3D registration the point clouds of the two scans are fused into a single object and transformed into a triangulated surface shown on the left blue model. On the right, texture mapping can then be applied to the triangulated surface.

It is clear that what is required is the ability to develop full 3D scenes that can be manipulated (rotated, translated, scaled) interactively using an input device such as a mouse and using a standard web browser. Standards and technologies to achieve this are evolving as discussed by Rodrigues and Robinson [10] [11]. We have had announcements from Mozilla in March 2009 that they were working on a specification to be released in early 2010 [12]. Immediately after that, Google announced in April 2009 the release of their O3D API (Application Programming Interface) in a browser plug in [13]. These followed earlier announcement from Opera releasing their 3D Canvas in November 2007 [14]. As it happened, the Khronos Group released the first WebGL [15] specification in February 2011. The latest developments is that Google has incorporated its O3D API into the WebGL standards and Apple and Opera are working towards implementing WebGL specifications into their web browsers. In order to handle 3D graphics, web browsers need access to a number of core technologies and some are highlighted as follows.

A. OpenGL

OpenGL stands for Open Graphics Library. The main source of information is the OpenGL web site [2], which contains news, specifications, tutorials, and downloads among other materials. OpenGL was developed by Silicon Graphics Inc. in 1992 and has become the industry standard for graphics applications throughout the world. Its API interface is well developed and is being nurtured by the ARB (OpenGL Architecture Review Board), which is an industry consortium responsible for steering the evolution of the software.

The most appealing aspects of OpenGL are its high performance and portability or “device independence”. OpenGL comes pre-installed on all major operating systems (Windows, Linux, Unix, Mac) and applications can be developed to run on all platforms without the need for changing the code. The OpenGL API provides a set of rich and highly usable graphics functions allowing learners to produce stunning 3D simulations in a short time.

OpenGL code has been written in structured programming style providing easy integration to C/C++ applications. Other languages such as Java require a wrapping such as JOGL (Java OpenGL) that allows OpenGL functions, which were written in a non-object oriented way, to be used in the Java language [16]. The sheer power and ease of integration of OpenGL led it to be the graphics engine of choice providing functionality to a large number of current 3D modelling packages and graphics and visualization applications.

While OpenGL provides the engine driving graphics applications, it requires a front-end program to handle the visualization. Unfortunately, web browsers do not understand OpenGL and thus do not interface with. Thus, a wrapper is required to interface with and translate OpenGL graphics outputs into statements that the web browser can understand and display; such a wrapper is JavaScript.

B. JavaScript

JavaScript is the most popular scripting language on the Internet. Its main purpose is to add interactivity to a web page by embedding JavaScript code into standard HTML pages. It is an interpreted lightweight programming language with very simple syntax and it is freely available. With JavaScript, web designers can perform a number of useful operations such as using dynamic text into an HTML page, write event-driven code such as reacting to mouse events, modify HTML elements, validate data before submitting to a server to alleviate server load, detect the user browser and load appropriate pages designed for that browser, create cookies by storing and retrieving information on the user's computer and more [17], [18].

One of the most useful aspects of JavaScript is the ability to handle events. The JavaScript Event Reference lists 21 events that include the various mouse events (click, double click, mouse button down and up, mouse motion), keyboard events, window events (moved, get into focus, loses focus, resized), loading of a file is interrupted, the occurrence of an error, button is pressed, user exits the page and so on. Error handling is a strong feature of JavaScript and the designer can make use of try, throw and catch statements. Object oriented programming is also supported; in addition to the built-in JavaScript objects, it is also possible to access and manipulate all of the HTML DOM objects with JavaScript such as Document, Frame, Table, Image, Button, and so on.

To help designers writing compatible code for all browsers, JavaScript allows browser detection, such that particular code can be written for specific browsers. Creating and updating cookies and data validation such as checking whether or not an email is valid can also be performed in JavaScript before forwarding to a server. It is also possible to create animated images by setting events to load different images as the user hover the mouse over the page. Similarly, one can create an image map with several clickable areas on the image and define event handlers to react to user input.

If one wishes to access resources residing on the client's machine (such as OpenGL) and display the outputs of the computation using a web browser, a client-side programming language would be a preferred choice such as JavaScript. If a server-side language were to be used (such as ASP or PHP) this would place undue load on the server and on the network, resulting in performance degradation especially for demanding applications such as 3D graphics. However, the standard JavaScript language is not designed to interface with OpenGL so that an extension is required; such an extension is provided by WebGL.

C. The WebGL API

The Khronos Group defines WebGL as a cross-platform, royalty-free web standard for low level graphics API based on OpenGL [19]. A WebGL JavaScript application is defined entirely within an HTML document that is loaded into a web browser. In principle, all that a web designer needs is a text editor to write WebGL statements. The WebGL interface takes care of the communication with the client's graphics hardware through OpenGL libraries. The full power of the underlying graphics hardware is thus, harnessed by WebGL for a quality user experience.

Whereas WebGL does not specify file formats, current trends to define 3D scenes use the COLLADA format [20], which has been developed by the Khronos Group. COLLADA is an open standard to facilitate the use and interchange of 3D assets and its format is supported by all major content creation applications including 3ds Max, Maya, SketchUp and others. The problem of loading COLLADA files directly using WebGL is that the programming soon becomes very complex, as the developer needs to adapt to the file format and to what COLLADA supports. A simpler, more useful and faster solution is to load data that have been defined in JSON (JavaScript Object Notation) format [21]. JSON is a text file containing pairs of values in a specified order. These values make up vertices, textures, indices, and so on.

If the 3D scene is very large with texture, animation and other effects such as skinning, the JSON files may turn out to be too large to be loaded in acceptable time frames. The solution is still to convert to JSON and load those types of assets from binary data that has been compressed. Such solution is provided by Google through their O3D API [3], which is a Java Script implementation on top of WebGL, thus fully compatible with WebGL specifications.

D. The O3D API

The O3D can be seen as an extension to JavaScript providing an API for 3D graphics using standard event processing and callback methods. Since it is not part of the standard JavaScript specification it requires the installation of a browser plug-in that is available for Windows, Macintosh, and Linux platforms. O3D uses its own file format for describing 3D scenes. The file has the extension *tgz* (tar-gnu-zipped), which is a set of files that have been tarred (grouped) together such that they can be handled as a single file and then compressed using the GNU zip application. The specific file format of O3D (with extension *o3dtgz*) requires the original 3D scene to be defined in the COLLADA format.

V. Implications to Teaching and Learning

Current technologies able to support interactive 3D graphics suggest that WebGL will become the standard way of delivering 3D contents to be visualized on a web browser without the need for plug-ins. WebGL is a client-side programming language based on JavaScript. The major functional difference between standard JavaScript and WebGL is that WebGL provides the necessary extensions to communicate with the underlying OpenGL libraries. Thus, OpenGL is the engine driving the 3D graphics, accessed by the browser through WebGL constructs – an alternative option is to access OpenGL through Google's O3D API. Initially, Google pushed its own technology but now it has aligned itself with WebGL, so that O3D is simply a JavaScript extension that operates on top of WebGL.

Any higher education institution planning to deliver learning content that requires interactive 3D graphics needs to consider current trends. It is clear that there will be a demand for programmers with OpenGL and WebGL skills. Therefore, a programme of studies should include the teaching of OpenGL followed by WebGL. It is also clear that in order to follow this path, students will be required to have a good mathematics background together with C programming skills. Here there is an opportunity to develop specialized courses perhaps at MSc level to address those requirements.

From the point of view of Museums Sheffield, the 3D models generated by this project have significant value to the Museums' learning offer. They have allowed the creation of a bank of pre and post visit resources for schools, college and HE groups. The models can be used to further investigate the objects back in the classroom, and act as an aide memoire for students. The resources provide a sustainable model for access as learners are effectively given access to objects without requiring facilitation by museum staff. This is hugely important at a time when museums are critically underfunded and the future provision of learning activities remains in doubt. Virtual objects enable an infinitely repeatable leaning experience by allowing detailed remote examination of objects, with no detrimental effect to the condition of the artefact. The models provide rich source material to support talks and workshops for informal learners, offer study sources for art, design and technology students, and offer inspiration for practitioners in the decorative arts. Crucially the techniques used here can provide a minutely accurate version of real objects in line with current state of the art in 3D object models.

VI. Conclusion

A major concern of our approach has been public accessibility and the project has been driven by the need for universal access. We have considered a number of candidate technologies including OpenGL and WebGL. We have focused on full interactive 3D models and discussed how the WebGL solution contrasts with previous solutions such as flash animations. The discussion in this paper is very timely as the first specification of webGL has been released in March 2011 by the Khronos Group. The specification is supported by all major players in industry including Mozilla, Google, and Apple and it is likely to become the standard way of delivering 3D content over the Internet with no need for browser plug-ins. Concerning online access of resources represented by the Metalwork Collection of Museums Sheffield its value has been highlighted to the provision of the Museums' offer. Furthermore, it is clear that HE institutions have to address the needs for teaching and training professionals in these technologies.

Acknowledgements

This project is funded by the UK-JISC Rapid Digitization Programme 2011.

References

[1] M. Kormann, The Demise of Ancient Helike in 373BC: Analysis of Archaeological and Environmental Evidence and Historical Records, Certificate in Archaeology, The University of Hull, May 2003.

[2] OpenGL. The Industry's Foundation for High Performance Graphics, http://www.opengl.org/

[3] O3D: WebGL Implementation of O3D, http://code.google.com/p/o3d/

[4] M.A. Rodrigues and A. Robinson, Novel Methods for Real Time 3D facial Recognition, in Strategic Advantage of Computing Information Systems in Enterprise Management, Majid Sarrafzadeh and Panagiotis Petratos (Eds) ISBN: 978-960-6672-93-4, p169—180, 2010

[5] W. Brink, A. Robinson and M.A. Rodrigues, Indexing Uncoded Stripe Patterns in Structured Light Systems by Maximum Spanning Trees, British Machine Vision Conference BMVC 2008, Leeds, UK, 1-4 Sep. 2008.

[6] M.A. Rodrigues, A. Robinson and W. Brink, Fast 3D Reconstruction and Recognition, in New Aspects of Signal Processing, Computational Geometry and Artificial Vision, 8th WSEAS ISCGAV, Rhodes, p15-21, 2008

[7] M.A. Rodrigues, A. Robinson and W. Brink, Issues in Fast 3D Reconstruction from Video Sequences, Lecture Notes in Signal Science, Internet and Education, Proceedings of 7th WSEAS International Conference on MULTIMEDIA, INTERNET and VIDEO TECHNOLOGIES (MIV '07), Beijing, China, September 15-17, pp 213-218, 2007

[8] M.A. Rodrigues, A. Robinson, L. Alboul, W. Brink, 3D Modelling and Recognition, WSEAS Transactions on Information Science and Applications, Issue 11, Vol 3, pp 2118- 2122, 2006.

[9] Faro Arm http://www.faro.com/uk.aspx

[10] M.A. Rodrigues and A. Robinson: Developing Interactive 3D Models for E-Learning Applications, *In* M.T. de Mello, C.Z. Carvalho Neto, and F.J. Spanhol (Eds) "Hipermidias Interfaces Digitais em EAD", Editora Laborciencia Ltd, Sao Paulo, 2009, pp 155-175

[11] M.A. Rodrigues and A. Robinson: 3D Post Processing Methods for Web Based Integration, Invited Keynote Speaker at ICBL 5th International Conference on Computer Blended Learning, 5-7 November 2009, Florianopolis, Brazil

[12] CNET News. Mozilla, graphics group seek to build 3D Web. http://news.cnet.com/8301- 17939 109-10203458-2.html, 24 March 2009

[13] AJAXIAN News. 3D Canvas in Opera. http://ajaxian.com/archives/3d-canvas-in-opera, November 2007

[14] AJAXIAN News. 2009. O3D: Google releases 3D API in a Browser plug-in. http://ajaxian.com/ archives/o3d-google-releases-3d-api-in-a-browser-plugin 21 April 2009

[15] The Khronos Group, Media Authoring and Acceleration http://www.khronos.org/

[16] JOGL, 2009. Java Bindings for OpenGL API. The JOGL API Project https://jogl.dev.java.net/

[17] JavaScript Tutorial, 2011. W3Schools Online Web Tutorials. http://www.w3schools.com/js/default.asp

[18] JavaScript Source, 2011. The JavaScript Source, http://javascript.internet.com/

[19] WebGL: OpenGL ES 2.0 for the Web. http://www.khronos.org/webgl/

[20] COLLADA. Digital Asset and FX Exchange Schema, https://collada.org [JavaScript Tutorial, 2009] W3Schools Online Web Tutorials, http://www.w3schools.com [JavaScript Source, 2009] The JavaScript Source, http://javascript.internet.com/ [JOGL, 2009] Java Bindings for OpenGL API. The JOGL API Project https://jogl.dev.java.net/

[21] JSON Java Script Object Notation http://json.org/