Hybrid Visualisation of Digital Production Big Data

Alun Evans* Javi Agenjo Josep Blat
Department of Information and Communications Technologies
Universitat Pompeu Fabra, Barcelona, Spain

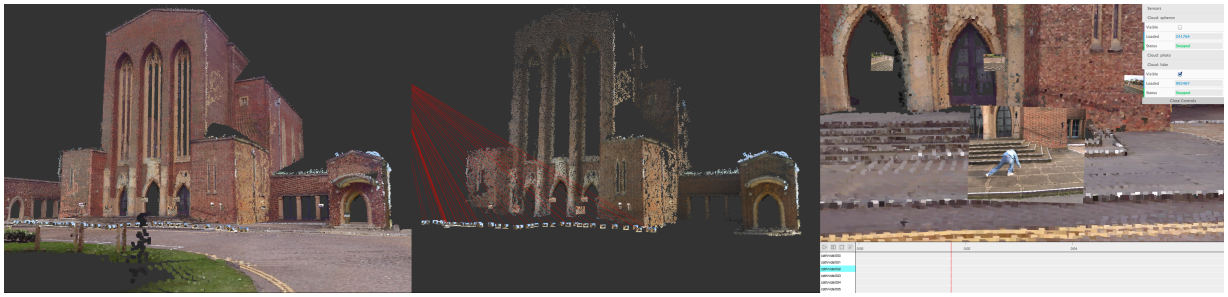

Figure 1: *Web-based visualisation of multimodal data recorded on set (LIDAR, static image reconstruction, and witness video)*

Abstract

In this paper, we present a web application for the hybrid visualisation of digital production Big Data. In a typical film or television production, several terabytes of data can be recorded per day, such as film footage from multiple cameras or background information regarding the set. Interactive visualisation of this multimodal data, integrating 2D (image and video) and 3D graphics modes, would result in enhanced use. A browser-based context is capable of this integration in a seamless and powerful manner, but faces significant challenges related to data transfer and compression which must be overcome. This paper presents an application designed to harness the power of a hybrid web context while attempting to overcome or compensate for the difficulties of data transfer limitations and rendering power. Results are presented from three, publicly available test datasets, which represent a realistic sample of data recorded on a typical high-budget production set.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; D.2.13 [Software Engineering]: Reusable Software—Reusable libraries;

Keywords: WebGL, Pointclouds, Big Data, Web, Hybrid, Visualisation, Video

1 Introduction

Cinema and television production are becoming increasingly digital, and data sourced from various modalities are now an integral part of the production stages. Beyond the footage coming from a single main camera of the recent past, a modern production set will make use of multiple witness cameras, 3D laser scanners, spherons or RGBD cameras, aiming at gathering as much data as possible regarding the surrounding environment, and facilitated by larger and cheaper storage devices. The traditional barriers between production, pre- and post- production are being broken. The large amount of data from different sources has to be understood, especially in terms of its quality and suitability, and has to be integrated in the workflow.

A standard approach to use this data is to view and analyse each modality separately, as part of the post-production process. However, this approach suffers from several problems: the process takes place largely away from the production environment; users must be present in the same location as where the data is physically stored; and viewing each modality individually is a process requiring the use of separate software for each modality, which makes it difficult for users to understand the integrated picture, and slows the whole process down. Addressing these issues is the primary motivation for the work in this paper.

We present a web-based visualisation application which allows remote users to access the production data recorded on-set, opening new possibilities for remote collaboration in the production and post-production process. The hybrid interactive visualization, tightly integrating video, static image, hypertext and real-time 3D graphics, which is enabled by the 3D web, is more difficult to replicate with traditional offline methods [Jankowski and Decker 2013] [Gonzales et al. 2009], and is better for the user. We address the web challenges to deal with big data in this context and our application demonstrates new functionalities in a challenging industrial environment.

2 Background

Jankowski and Decker's [Jankowski and Decker 2012; Jankowski and Decker 2013] research suggests that a dual-mode user interface, integrating traditional hypertext and 3D graphics, leads to greater user engagement, and better and more efficient understanding, than interfaces where the modalities are shown in a more traditional way of hypertext, figures and captions, where the individual modes are more separate. Inspired by this paradigm, our work tightly integrates 2D image data (video and still images) and metadata within an interactive 3D context. When carried out in an offline applica-

*e-mail:alun.evans@upf.edu

tion, such a hybrid approach requires specific software engineering to allow individual video frames to be passed to the 3D rendering engine as texture data [Gonzales et al. 2009]. However, in a browser-based context, the HTML5 video element [Pfeiffer and Pearce 2010] provides a convenient wrapper for reading and accessing the pixel data provided by videos (and stored in a variety of formats), as well as providing HTTP 1.1 byte-range access to server-based files, which facilitates buffering and permits the implementation of video scrubbing interfaces (where the user can move forward and backwards through the duration of the video to view desired moments). Without byte-range access, the entire video is downloaded linearly, and users could only view frames within the data downloaded. A rich interactive video interface is required for an application suitable to review film material, assess its quality and use it further in production and post-production.

The main technical difficulty in web-based visualisation of 3D data from industrial digital productions is usually related to the size of media, and transferring the data from the server to the client. While 3D data compression techniques for web-based transmission are very well studied in the literature [Evans et al. 2014b], [Limper et al. 2013] demonstrate that, as bandwidth speeds increase, the computing power and time required to decompress heavily compressed data becomes the limiting factor, as it makes interactivity unfeasible. As a result, techniques that emphasise progressive download and refinement of 3D data (over pure compression) seem to provide a better user experience [Lavoué et al. 2013][Evans et al. 2014a].

3 Server-Side Processing

Besides the principal camera footage (the so called Dailies), some key types of data which are typically recorded during a standard day of film or high-end television production can be grouped as:

1. Multiple witness cameras video, stored in high resolution, uncompressed formats (either as image sequences or loosely wrapped into a format such as avi or mov). This footage relates to the core action and is used to support high quality post-production or visual effects.

2. Pointcloud data, recorded from laser scanning technologies such as LIDAR, with or without color information. Pointclouds are used in production and post-production, for example, to enable production staff to visualise the set in 3D at a later date, or as a basis for the generation of digital effects.

3. Other image data, such as multiple static photographs of the set, spheron images, RGBD images etc. These are primarily used to assist in the post-production process, although they are gradually being used more on-set.

Additionally, the data from various sources can be combined. Multiple LIDAR scans can be registered and stored as a single pointcloud dataset, representing the entire structure of the set environment. And feature detection, matching and registration algorithms can be run on spheron and image data to generate 3D point clouds of the areas covered by those images. As production tends to store original format data in the highest possible resolution and bit depth, and employs lossless (if any) compression, this leads to very large file sizes. Pointcloud data depends on the source and number of combined scans, but typically a full-set representation will result in a file of between 200MB and 1GB. Even static image data from DSLR cameras means dozens of MB per image, for potentially hundreds of stored images.

Publically available examples of such data are reasonably difficult to find, largely because the majority of production data is proprietary and has very restricted access. However, some large European projects, such as i3DPost [1] and IMPART [2] have made their generated data available. For this application, we used data from the IMPART project, the most recent, representative and challeging available, with permission.

Managing this data (which might be called 'Big Data') requires pre-processing and image analysis in order to extract features or higher level metadata, etc. In this paper we focus on the processing related to making it suitable for interactive visualisation in a web-based client, due to the fact that it must be transferred over the internet, while the visualisation must be interactive enough to be user friendly. The remainder of this section details these processing steps used in our application.

3.1 Compression of 2D Image Data

Our system ingests 2D data, and uses simple scripts and command line tools such as `ffmpeg` and `imagemagick` to convert and save a lower-resolution copy of each file, more suitable for web viewing. Video data is converted to OGG-Theora format at medium-high quality, and downsized to 480p. A single image frame of the video is also stored at thumbnail resolution, which is later used as a static placeholder in the visualisation (see below). Static image data is first saved as JPEG (if this is not already available) and then a copy is also saved at thumbnail resolution for fast preview in the visualisation. Similar thumbnail/preview versions are created for 2D image representations of the recorded spheron and lidar scans.

3.2 Progressive Pointcloud Visualisation

Raw LIDAR data from several sensor is registered and stored as a single file in OFF format (position and color information). We then use a similar approach to [Evans et al. 2014a] and [Schutz] to process the pointcloud data into a format suitable for progressive download to a web-based client. In an offline application, the pointcloud data is added to a memory efficient octree, which is then saved breadth first as a series of binary files, containing information about the structure and color of each node of the tree. The data is then ready to be transferred file-by-file to the web-client, thus gradually increasing the resolution of the downloaded pointcloud, which is reflected by progressive refinement in the client-side 3D visualisation (see section 4.1 below).

3.3 3D Reconstruction and Registration

The dataset used for this application includes data obtained via feature matching and registration algorithms [Kim and Hilton 2014] [Kim and Hilton 2013a] [Kim and Hilton 2013b], which are used in order to back-calculate the real position in the scene of the sensor used to record the data. These positions are then registered to the LIDAR data which is taken as the ground truth reference for the scene. The result is a set of matrices representing the position (and, for video camera footage, the orientation) of each sensor in the scene. These matrices are stored as text files and are associated with the relevant data files and thumbnails. The results of feature extraction from multiple 2D data sources such as spheron data, static image data, and RGBD data are also used to create 3D pointclouds. While the resolution of this data is much lower than that obtained by LIDAR scanning, it is still processed using the same technique presented in Section 3.2, in order to allow progressive transfer and visualisation, as it is important to have integrated visualisation of multi-source 3D data, to which interactivity features

[1] http://cvssp.info/i3dpost_action/

[2] http://impart.upf.edu

are added as described later.

3.4 Scene Metadata and gzip Compression

A JSON scene-description file stores the relative path to each of the elements of data available to be visualised by the client. All the data is stored in directories which are served to the web via an APACHE web server; this is configured in order to enable HTTP gzip compression of all the file formats to be served to the client. While this compression has minimal effect on files which are already compressed (e.g. JPEG or OGG-Theora), it has a significant effect on the transfer time of the uncompressed binary files used to store the data for progressive point-cloud visualisation, which are typically reduced to 60% of their initial size.

4 Hybrid rendering results

The browser application is constructed using a WebGL context as a basis for all 3D elements in the scene, and using Document Object Model (DOM) elements to manage video and image data as required. The JSON scene description is first downloaded to initialise the scene.

4.1 Progressive Pointcloud Rendering

For each point-cloud in the scene (i.e. from the LIDAR data and from the reconstructions from the Spheron /Photos /RGBD) the browser starts downloading, sequentially, the list of files which contains the breadth-first description of each pointcloud. After each file is downloaded, it draws the octree to the scene, with each node in the tree represented by a GL_POINT, whose size is that of the width of the octree node. Point widths are kept constant with respect to the distance to the camera by multiplying the desired size by the height of the near projection plane, in homogenous coordinates.

As more data is downloaded, the pointcloud is updated. However, this is not simply a case of drawing higher-resolution data over the (previously drawn) lower resolution data, as the lower resolution data will occlude any higher resolution points. To deal with this issue, the 'level' of the octree is tracked, and lower-resolution data is periodically culled from the draw-buffer (see figure 2). Table 1 shows the time taken to download the 10 test pointclouds, spread over three scenes, at different resolution levels; the percentages are of the final draw-buffer size of each cloud. The *first view* corresponds to the lowest resolution of the octree and it appears within a second for each cloud. The clouds in each of the 3 scenes are downloaded simultaneously and this explains the inconsistent results between cases with clouds of similar sizes. These values are similar to those obtained with the technique presented in [Evans et al. 2014a], despite multiple pointclouds (at least three) being downloaded simultaneously, and compare well with those state-of-the-art on the different but related problem of progressive mesh transmission [Lavoué et al. 2013]. Figure 2 shows a visual representation of the progressive rendering effect.

4.2 Rendering of 2D video in a 3D context

Once the scene description is parsed, a hidden HTML5 video element is created for each video in the scene, and added to the DOM. The fact that the element is hidden ensures that it does not affect the 3D visualisation; however, the WebGL API can read the HTMLVideoElement in the DOM, and extract the pixels of the current frame into texture data, which can be used within the 3D scene.

Using the scene metadata as described in 3.3, the application creates a simple plane mesh, using the position and orientation of each

Figure 2: *Progressive and simultaneous rendering of four point-clouds, with resolution increasing from top-left to bottom-right*

video camera in the scene as the model matrix. Then, every draw frame, the DOM video elements pipe their texture information as WebGL textures, which are displayed on plane meshes. The result is the video data being rendered in real time on meshes within the 3D scene with the actual position and orientation of the camera, providing a tight 2D/3D integration.

To control the playback of the video and to ensure that at most one video is playing at a time (avoiding needless bandwidth and rendering power use), the application features a timeline interface created as a 2D Canvas and added to the DOM separately. This element allows the user to select a camera, which then moves the 3D camera to a position immediately behind the plane mesh displaying the selected camera. Play/pause/stop controls exist for video playback, and scrubbing allows the user to skip forward and backwards (by setting the time of the HTML5 video element via javascript). Figure 3 shows a screenshot of the timeline interface and the mesh planes featuring the video frames, illustrating the enhanced naturalness of the visualisation. On the other hand, the timeline component is less tightly integrated in the interface, an issue to be improved in the future.

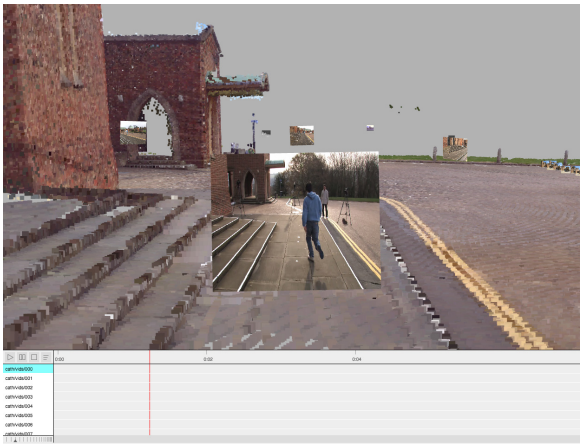

Figure 3: *Mesh plane with video camera images, and timeline*

4.3 Sensor Positions

The positions of the original sensors used for construction of the pointclouds (LIDAR, photo cameras, spherons) are visualised by

Table 1: *Time taken (in milliseconds) to download and render different point clouds at three resolution levels. The clouds in each of the three scenes were downloaded simultaneously. Bandwidth is clamped to 8mbps.*

Dataset	Cathedral			Patio			Studio			
Data source	Spheron	Photos	LIDAR	Spheron	Photos	LIDAR	Spheron	Photos	RGBD	LIDAR
Num. points	75838	276113	1123222	300405	365155	3000000	314567	315282	442137	670324
First view	464	387	295	620	625	992	356	483	444	294
50%	931	1524	3151	9892	9699	53057	2758	2783	4011	5559
100%	1171	2675	7099	14454	15365	75210	4877	5009	6340	7771

positioning a billboard in the 3D scene at the position provided by the dataset, which displays a thumbnail of the final image. The rotation data is not available, so they are rendered as billboards which always face the camera position. Clicking or tapping on the billboard displays a lightbox (created with the DOM) showing the original full-resolution image of the sensor. Figure 4 shows a screenshot of the sensors positioned relative to the scene pointclouds. Their positions within the 3D scene can be better visualised using a feature which the user can enable so that coloured vertical lines are overlaid above each billboard. Different sensor types can be assigned different colours (see Figure 1).

Figure 4: *Sensor positions are visualised by billboards*

5 Discussion

Usually, visualisation of production big-data is split between individual tools in 2D and 3D, with video and images processed in a 2D domain then visualised using a thumbnail browsing interface, while 3D data in dedicated 3D production and rendering software, which makes difficult to get an integrated picture. In this paper we present an application which takes advantage of the web context to create a hybrid 2D and 3D visualisation which enables the interactive viewing of such data with one application. The application succeeds in blending these modalities, as well as enabling further advantages of the web context, machine and platform independence. With WebGL and HTML5 elements now supported on all major browsers, the application can be viewed on a variety of desktop and portable hardware without specialized software or licenses.

The results presented show that the steps taken (progressive visualisation of pointclouds, pre-compression of videos and images) to reduce the problems related to remote viewing of big data have been partially successful, though it is clear that the performance will not be identical to an application where the data is stored locally. Our future work is now geared towards taking advantage of browser access to specialized hardware such as accelerometers and cameras in portable devices, and explore the possibilities of mixed and augmented reality applications in this context.

Acknowledgements

This work was supported by the European Commission, FP7 IMPART project (grant agreement No 316564), and by the Spanish EEE (TIN2011-28308-C03-03) project.

References

EVANS, A., AGENJO, J., AND BLAT, J. 2014. Web-based visualisation of on-set point cloud data. In *Proceedings of the 11th European Conference on Visual Media Production*, ACM, 10.

EVANS, A., ROMEO, M., BAHREHMAND, A., AGENJO, J., AND BLAT, J. 2014. 3d graphics on the web: A survey. *Computers & Graphics 41*, 43–61.

GONZALES, E., EVANS, A., GONZALES, S., ABADIA, J., AND BLAT, J. 2009. Real-time visualisation and browsing of a distributed video database. In *Advances in Computer Entertainment Technology*, ACM, 423–424.

JANKOWSKI, J., AND DECKER, S. 2012. A dual-mode user interface for accessing 3d content on the world wide web. In *Proceedings of the 21st international conference on World Wide Web*, ACM, 1047–1056.

JANKOWSKI, J., AND DECKER, S. 2013. On the design of a dual-mode user interface for accessing 3d content on the world wide web. *International Journal of Human-Computer Studies 71*, 7, 838–857.

KIM, H., AND HILTON, A. 2013. 3d scene reconstruction from multiple spherical stereo pairs. *International Journal of Computer Vision 104*, 1, 94–116.

KIM, H., AND HILTON, A. 2013. Planar urban scene reconstruction from spherical images using facade alignment. In *Proc. IVMSP*.

KIM, H., AND HILTON, A. 2014. Hybrid 3d feature description and matching for multi-modal data registration. In *Proc. ICIP*, 3493–3497.

LAVOUÉ, G., CHEVALIER, L., AND DUPONT, F. 2013. Streaming compressed 3d data on the web using javascript and webgl. In *Proceedings of the 18th International Conference on 3D Web Technology*, ACM, 19–27.

LIMPER, M., WAGNER, S., STEIN, C., JUNG, Y., AND STORK, A. 2013. Fast delivery of 3d web content: a case study. In *Proceedings of the 18th International Conference on 3D Web Technology*, ACM, 11–17.

PFEIFFER, S., AND PEARCE, C. 2010. *The definitive guide to HTML5 video*. Springer.

SCHUTZ, M. Potree. http://potree.org. Accessed: 2015-03-16.