PanView: An Extensible Panoramic Video Viewer for the Web

Cassio E. dos Santos Jr, Jessica I. C. Souza, Virginia F. Mota,
Guilherme S. Nascimento, Guilherme S. Gorgulho, Arnaldo de A. Araujo
Department of Computer Science
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil
{cass, jessicaione, virginiaferm, gsnasc, gsad, arnaldo}@dcc.ufmg.br

***Abstract*—In this paper we present a panoramic video viewer for the web. We propose PanView, an open-source video-based panoramic viewer that provides a greater immersion of filmed environments. The main advantage of consider panoramic video is that users can pan or tilt virtual cameras in the recorded scene, allowing him/her to focus on information presented in different angles or moments, thus, allowing him/her to access more information than in ordinary videos or in static images. PanView is fully extensible with easy customization using user coded modules and is implemented using modern web-browser standards, which reduces the computational requirements. To motivate the use of PanView, we present a performance comparison considering a panoramic viewer based on Adobe Flash Player. The applicability of PanView is shown in two projects: virtual tour on historical cities and analysis of a railroad network.**

I. Introduction

Panoramas consist in representing the surroundings of a scene in a single image, achieving, this way, several advantages over ordinary single view methods to acquire images. Panoramas allow users to generate images from specific views of the scene at the same time that other users are able to choose other views. The development of advanced devices and algorithms to build panoramas leverage the growth of services based on panoramic videos and images. The Google Street View [1], for instance, allow users to see panoramic images from different places around the world.

Among the different kinds of panoramas, panoramic videos are preferable since most applications, such as street view tours, remote meetings, or e-meeting, surveillance; require temporal information along with the image of the scene. The main advantage of considering panoramic video rather than regular videos is that users can pan or tilt virtual cameras in the recorded scene, allowing them to focus on a specific event of the video that would rather be lost in a regular camera with fixed view. In an e-meeting scenario, for instance, an attendee can focus on the presentation screen while other attendees can focus on the speaker or on the audience.

There are several challenges that hamper the use of panoramic videos, such as the large amount of video footage often required to analyze, store or index; and the lack of a common platform to visualize, analyze or distribute panoramic videos. Considering the aforementioned challenges, we propose *PanView*, a extensible video-based panoramic viewer.

Fig. 1: PanView interface. The map in the top-right corner and the control buttons on the bottom are drawn using user modules, which demonstrate the extension capabilities of PanView.

Figure 1 illustrate the extensibility of PanView regarding the add-in of a map to display geo-location data and control buttons. PanView is an open-source[1] and extensible framework, designed to be customized for a large set of applications.

The framework is organized following a microkernel architecture, composed by a *core* and several *user* modules. The *core* module includes common functionalities that are useful for different applications, such as draw the panoramic video and handle basic mouse inputs. *User* modules enable, for instance, components to be drawn in the panoramic video interface, such as a map to display geo-location data or on-screen control buttons. Since user modules are independent of each other, including or removing the file of a user module in a web-page code is enough to enable or disable that user module. Moreover, PanView is coded using modern web standards, such as WebGL and HTML5, ensuring, this way, higher performance when compared to panoramic viewers based on Adobe Flash Player[2].

The remainder of this paper is organized as follows. We discuss related works in Section II. Section III presents PanView. As motivation, we compare PanView with a public and

[1]PanView will be available at http://www.npdi.dcc.ufmg.br/PanView
[2]http://www.adobe.com/products/flashplayer.html

978-1-4799-6953-1/14 $31.00 © 2014 IEEE
DOI 10.1109/LAWeb.2014.19

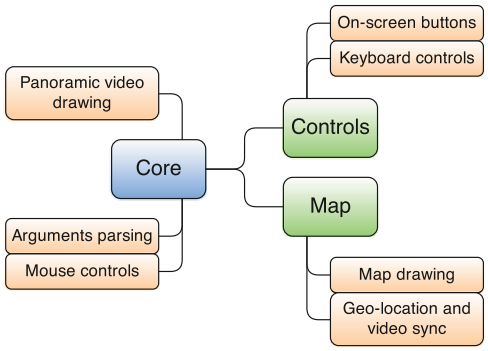

Fig. 2: PanView components. The *core* is the main module and it is responsible for draw the panoramic video, parse the input arguments and handle mouse inputs. The *controls* module handles keyboard inputs and on-screen buttons. The *map* module draws the map and synchronizes the geo-location with the panoramic video.

free panoramic viewer based on Adobe Flash Player. Examples of applications that use PanView are depicted in Section V. Finally, we conclude with final remarks and future directions in Section VI.

II. Related Works

There are several works in the literature describing approaches to assemble images acquired from different angles into a single continuous panoramic image. Examples of these works include match correspondent pixels in overlapping parts of the images to estimate warping parameters [2], [3], and join accelerometer and gyroscope data in mobile devices while capturing the surroundings of the scene [4]. Regarding panoramic videos, the common approach to assemble images consists in fasten cameras in a rigid structure to estimate warping parameters offline [5], [6]. Then the same transformation is applied to every frame in the captured video.

Panoramic videos draw attention in applications such as e-meetings [5], [6], [7], street view [1], [8], [9] and surveillance [10], [11]. Google Street View [1] is the most popular panoramic viewer available, but it lacks support to videos and present few customization options. Salado Player[3] provides the source code, thus, it allows more customization options, but still lacks video support. A video-based panoramic viewer built using Adobe Flash Player is krpano[4], however its source code is not public available. Kolor Eyes[5] is an advanced player for panoramic videos with versions for smartphones and desktops. It is a free application, although the source code is not available. The web version of Kolor Eyes is built on HTML5 and WebGL, but its use is limited to their own hosting service. Other websites can embed the hosted videos, which will be executed in Kolor Eyes web player with few customization options.

[3]http://openpano.org/
[4]http://krpano.com/
[5]http://www.kolor.com/360-video/kolor-eyes-player

Code 1: Required code to insert the PanView in a web-page.

```
<script src="panview_core.js"/>
<script src="plugins/panview_controls.js"/>
<script>
var options = { /* some player options */
        video: "video.mp4",
        mouseControls: true,
        keyControls: true,
        buttonControls: true,
        autoplay: true,
        autoresize: true,
        fov: 80,
        pole: {x: 0, y: 0, z: 0}
};
var pano = new PanView(div, options);
</script>
```

More examples of panoramic viewers can be found at PanoTools website[6], although most of them focus on high resolution panoramic images and do not support videos. For instance, Pano2VR, Pannellum and Leanorama, which are listed in the PanoTools website, are implemented in HTML5. None of them have support for videos. Moreover Pano2VR is not open-source while Leanorama has been discontinued.

Our work differs from the aforementioned works considering that our proposed panoramic viewer support panoramic videos, is open-source, presents easy customization using user modules and is implemented using modern web-browser standards, such as HTML5 and WebGL, which reduce the computational requirements.

III. PanView Architecture

The architecture of PanView follows a microkernel design pattern [12] and consists in a single *core* and several *modules* that can be written by users to extend basic functionalities. The core provides common functionalities to applications that require a panoramic viewer, such as parse user arguments, draw the panoramic video in the web-page and control basic input. The modules implement other functionalities such as map preview, button control and keyboard shortcuts. Each user module is independent from the others and it is added to the core when its file is included in the web-page source code. Then, a small code, illustrated in Code 1, is required to configure and draw PanView in the web-page. An overview of PanView architecture is presented in Figure 2.

In the following sections we describe the core, in Section III-A, and two user modules: *controls*, described in Section III-B, and *map*, described in Section III-C.

A. Core

The core consists in a single JavaScript file that holds the main class definition. The following functionalities are included in the core: draw each frame of the panoramic video in the web-page, control which part of the video sphere is

[6]http://wiki.panotools.org/Panorama_Viewers

Fig. 3: Sample frame from a panoramic video in spherical projection. Horizontal coordinates of pixels are mapped in longitude in the video sphere while vertical coordinates of pixels are mapped in latitude.

being draw, and handle basic mouse input. The mouse input control is included in the core since most applications require mouse navigation and the variables that control the view angles of the video sphere are closely related to the mouse control. To facilitate the correction of the navigation axis in the mouse control, the north pole of the video sphere can be adjusted by indicating Euler rotation angles of the sphere. This way, videos acquired using cameras in different rotations can be corrected in the parameters of the PanView.

The panoramic video is drawn considering spherical projection, therefore, the input video must be presented in P pixels height, mapped to latitude coordinates of the video sphere, and $2 \times P$ pixels width, mapped to longitude coordinates. An illustration of a panoramic image is present in Figure 3. We choose spherical mapping to be implemented in the core since it is more common to find applications to generate panoramic videos in this type of projection than in other (cubic or hexagonal, for instance). However, other projections can be implemented in user modules.

B. Controls Module

The controls module allows the user to interact with the panoramic video using keyboard inputs and buttons on the screen. The implemented actions are play/pause and the ones already possible to be executed with the mouse input, which are zoom in/out and camera rotation. The on-screen buttons can be selected using the tab key and activated using the *enter* key. Functions are available to disable or enable the keyboard controls and hide or show the buttons. The controls work with the most common web-browsers[7].

PanView default on-screen buttons are highly customizable and present effects on mouse hover, tab focus and when they are activated. A css style sheet is included allowing easy customization. It is possible to completely personalize the on-screen buttons with new images and css attributes. Each action is implemented as a separated function thus the keyboard controls and the on-screen buttons share the same action

[7]Google Chrome (http://www.google.com/chrome) and Mozilla Firefox (http://www.mozilla.org/firefox)

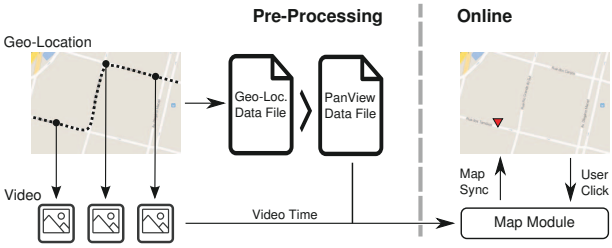

Fig. 4: Overview of the two processing steps of the map module. The panoramic video is composed by the video file and the geo-location file. On the pre-processing step, the geo-location file is converted to a structured file compatible with PanView. During the online step, the map module synchronizes the marker on the map according to the current video time. If the user performs a click on the map, the map module synchronizes the video and the marker according to the new position.

functions. It is possible to create new actions and buttons or change the default ones easily with simple modifications in the module source code.

C. Map Module

The map module draws a small map in the upper-right corner of the panoramic video (illustrated in Figure 1) which is used to display geo-location data associated to the panoramic video. A marker in the map indicates the geographic location of the frame being displayed in the panoramic video in the moment when that frame was captured, thus, it is required to update the marker for every frame of the video.

To avoid successive updates of the marker in the map, specially when the frame rate of the video is high, we employ a grid-based approach as follow. The geographical coordinates associated to each frame of the video are divided in cells of a grid and the marker is updated only once for each cell.

There are two processing steps in the map module: a *pre-processing* step, which converts the geo-location data to a structured file containing the grid information; and an *online* step, which draws the map and updates the marker and the time of the video when the user clicks in some location in the map. Figure 4 presents an overview of those two processing steps.

1) Pre-Processing: in this step we convert the geo-location data file (such as KML or GeoRSS) in a file that is compatible with PanView. The conversion script of the geographic meta-data file has certain procedures, adopted to associate the time in the panoramic video with the position of the marker in the map. The meta-data has a list of geographic coordinates and, associated with it, the time of each transition from a geographical location to another. The script read the input geographic data and converts to a structured data file which can be read by PanView.

2) Online: the online step consists in the user module, which receives the pre-processed structured file containing

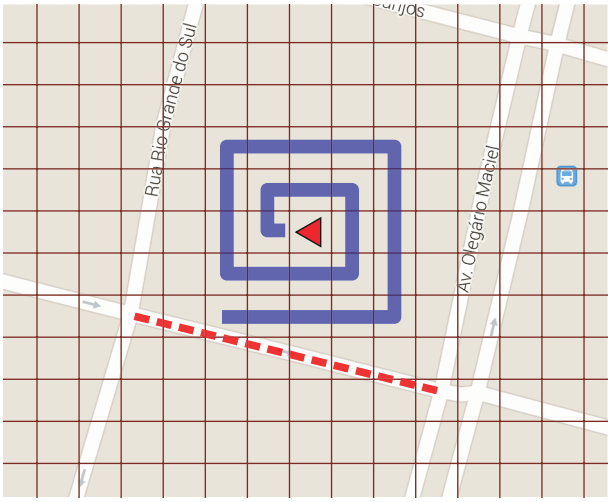

Fig. 5: Process to correct clicks of the user (triangle) out of the path described by the coordinates associated to the panoramic video (dotted line over the street). The cells around the clicked area are visited in a spiral pattern until it reaches a cell which contains coordinates of the panoramic video or the maximum number of cells are visited.

the grid data and draws the marker in the map. Then, the marker is updated in the map each time that the geo-position of the current frame being displayed in the panoramic video corresponds to a new cell in the grid. To change the time of the video whenever the user click in a geo-location in the map, we search in the neighborhood of the clicked cell for one that contains a least one time interval. The search is performed following a spiral pattern as illustrated in Figure 5. This way, the number of cells visited and the cell size determine the maximum error allowed between the click of the user and the coordinates of the video.

IV. Performance Evaluation

In this section we compare PanView with krpano, an Adobe Flash Player panoramic viewer available in the web. Our goal in this experiment is to motivate the use panoramic viewers built using HTML5/WebGL instead of Adobe Flash Player, which correspond to the majority of available tools[8]. Although krpano provides a HTML5/WebGL version, at the time of submission of this work the available version did not support panoramic videos.

To compare krpano with PanView, we consider the mean and 95% confidence interval of the number of Frames Per Second (FPS) that each panoramic viewer presents when displaying panoramic videos in different resolutions. To calculate the FPS, we use Fraps tool[9]. The krpano is available in demo and premium versions. The only difference between them is that a warning text is displayed in the demo version

[8]See PanoTools in Section II for a list of panoramic viewers.
[9]http://www.fraps.com/

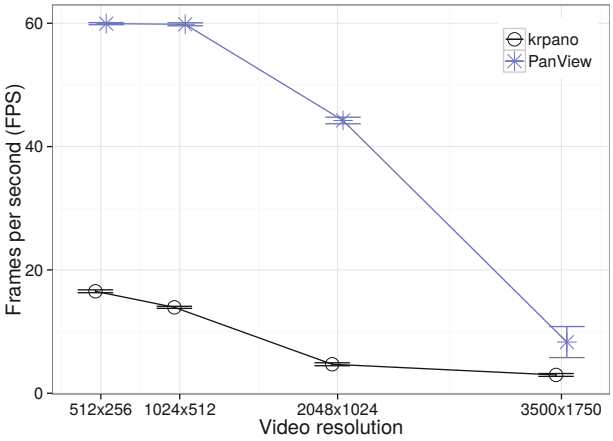

Fig. 6: Frames per seconds (FPS) when playing videos of different sizes considering krpano (panoramic viewer based on Adobe Flash Player) and PanView (based on WebGL and HTML5). Vertical bars indicate a 95% confidence interval of the FPS when playing the video. PanView presents higher FPS than krpano for any video size.

when playing the video. We use a panoramic video with 180 frames scaled to the sizes 512x256, 1024x512, 2048x1024 and 3500x1750. All tests were performed in a computer with Google Chrome web-browser Version 35, Windows 8.1 operational system, processor Intel(R) Core(TM) i7-3537U and 8 gigabytes of RAM.

Results presented in Figure 6 stand that, for any video resolution, PanView presents higher FPS than krpano. We further perform a paired t-test to certify that the differences between the averaged FPS for each video size is significant, which returns a p-value equal to 0.03, asserting that our proposed panoramic viewer presents higher FPS. The results are expected since the WebGL, used to draw videos in the PanView, take advantage of optimized codes for graphical cards in modern web-browsers.

V. Applications

PanView was originally developed to provide a web-based panoramic viewer for two projects under development, which are Historical Town Navigation System, described in Section V-A, and Railroad Analysis, described in Section V-B. Those projects are examples of applications for PanView, since our panoramic viewer extended to an independent project and it can be applied in other environments.

A. Historical Town Navigation System

PanView was originally part of the project Historical Town Navigation System [13], in which the streets of four historical cities of Minas Gerais, Brazil (Ouro Preto, São João Del Rei, Congonhas do Campo and Tiradentes) were registered in panoramic videos. The idea of the project is to provide a virtual city tour using a desktop or a web interface. PanView is the tool developed for the web interface.

B. Railroad Analysis

The idea of this project [14] is to capture panoramic videos from a railroad network for latter analysis. With the panoramic videos, the technicians in transportation engineering can analyze the railroads for events of interest, which are then used to elaborate a revitalization plan. The PanView allows easy access to the panoramic videos, since users interested in developing the revitalization plan can check the events of interest online and update them when convenient.

VI. Conclusions and Future Works

We presented PanView, a panoramic video viewer based on modern web standards, such as HTML5 and WebGL. Our player was built to be easily customizable and to support user modules which enrich the functionalities already presented in the core. We implemented two modules, one to handle a map synchronized with the video and another that provides on-screen buttons and keyboard controls. PanView achieved a better performance compared with an available panoramic viewer based on Adobe Flash Player.

The main advantages presented for PanView were open source code, low computational resource requirements, easy customization, extension modules and no dependence on external plugins. To illustrate the applicability of PanView, we presented two projects: a virtual tour on historical cities, within the Historical Town Navigation System project, and the analysis of a railroad network.

In future works, we will study the inclusion of other panoramic projections in PanView core. The user interaction can be enhanced through the implementation of clickable areas inside the videos, known as hotspots. In order to improve the performance, PanView can be modified to decode only the part of the video frame which is being shown.

Acknowledgment

The authors are grateful to FAPEMIG, CAPES and CNPq funding agencies.

References

[1] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, "Google street view: Capturing the world at street level," *Computer*, vol. 43, 2010.

[2] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, vol. 74, no. 1, pp. 59–73, 2007.

[3] Y. Xiong and K. Pulli, "Fast panorama stitching for high-quality panoramic images on mobile phones," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 2, pp. 298–306, 2010.

[4] A. Au and J. Liang, "Ztitch: A mobile phone application for immersive panorama creation, navigation, and social sharing," in *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop*. IEEE, 2012, pp. 13–18.

[5] J. Foote and D. Kimber, "Flycam: Practical panoramic video and automatic camera control," in *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference*, vol. 3. IEEE, 2000, pp. 1419–1422.

[6] R. Pea, M. Mills, J. Rosen, K. Dauber, W. Effelsberg, and E. Hoffert, "The diver project: Interactive digital video repurposing," *MultiMedia, IEEE*, vol. 11, no. 1, pp. 54–61, 2004.

[7] B. Erol and Y. Li, "An overview of technologies for e-meeting and e-lecture," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 2005, pp. 6–pp.

[8] M. Meilland, A. I. Comport, and P. Rives, "Dense visual mapping of large scale environments for real-time localisation," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference*. IEEE, 2011, pp. 4242–4248.

[9] J. Kopf, B. Chen, R. Szeliski, and M. Cohen, "Street slide: browsing street level imagery," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 96, 2010.

[10] R. Patil, P. E. Rybski, T. Kanade, and M. M. Veloso, "People detection and tracking in high resolution panoramic video mosaic," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2004, pp. 1323–1328.

[11] R. Kaiser, M. Thaler, A. Kriechbaum, H. Fassold, W. Bailer, and J. Rosner, "Real-time person tracking in high-resolution panoramic video for automated broadcast production," in *Visual Media Production (CVMP), 2011 Conference for*. IEEE, 2011, pp. 21–29.

[12] B. P. Douglass, *Real-time design patterns: robust scalable architecture for real-time systems*. Addison-Wesley Professional, 2003, vol. 1.

[13] M. de Miranda Coelho, "Recuperação de informação visual em bases de imagens de cidades históricas: contribuições para o reconhecimento e classificação de imagens," Doctoral thesis in Computer Science, Universidade Federal de Minas Gerais (UFMG), 2013.

[14] M. F. Porto, L. C. d. J. Miranda, N. T. R. Nunes, and C. E. Santos Jr., "The Feasibility Study for Establishment of Passenger Rail in The Metropolitan Region of Belo Horizonte / Brazil," *5th International Multi-Conference on Complexity, Informatics and Cybernetics (IMCIC)*, pp. 201–206, 2014.