ENGINEER - Vol. L, No. 04, pp. [13-22], 2017

http://doi.org/10.4038/engineer.v50i4.7271

Unconstrained Segue Navigation for an Immersive Virtual Reality Experience

S. Herath, V. Dissanayake, S. Rasnayaka, S. Seneviratne, R. Vidanaarachchi and C. D. Gamage

Abstract: Virtual reality (VR) is a rapidly growing field that has got innovative applications. Two main forms of VR content can be identified in the current applications --computer generated 3D models and adaptations of real world scenarios using digital imagery and video. However, real world adaptations are quite restrictive when it comes to interaction. These restrictions are due either to large storage needs, or to high computational needs, required to generate dynamic intermediate views. Thus, solutions through which real world simulation could be achieved in immersive and interactive VR environments are not available.

The research presented in this paper is intended to provide a high degree of interaction to users who engage in immersive VR experiences related to real world adaptations. The solution offered is a navigable grid map of spherical panoramas, which uses a three stage approach. Firstly, intermediate views among spheres are approximated using novel mechanisms. Secondly, an optimization strategy is introduced based on visual locality in which the areas with a higher probability of immediate interaction are given more prominence for quality during rendering. Thirdly, smooth segue transition is achieved through a machine learning backed gesture input system. The three approaches when combined together allow for an intuitive virtual experience while ensuring optimal resource utilization.

Keywords: Virtual Reality, Gesture Input, Real World Modelling, Interactive Walkthroughs

1. Introduction

With the recent advent of low cost virtual reality (VR) devices such as Google Cardboard, VR frameworks for games and many other applications are becoming increasingly common. However, none of these frameworks offer the ability to interactively explore a real world environment or for an average user/ business to create VR content. One major requirement of the framework is to provide free navigability in a virtual world modelled after a real world environment. In the existing adaptations of the real world into VR, navigation is highly restrictive and a transition between two points appears as a jump which hinders the life-like quality of the experience. Some adaptations use a blurring effect during such transitions which serves to reduce the unnaturalness of the transition. However, even such adaptations do not allow users to stop at an intermediate position but rather forces them to view only at discrete points captured by the camera.

Free navigability would ideally require the capturing of each possible point on the environment. Since this is impractical, the solution would be to adapt a discrete capturing of a space and then to generate intermediate points as shown in Figure 1 to allow for free navigability.

For the playback application, the game engine, Unity[1], which allows exporting into multiple platforms for VR playback was used. In order

***Ms. S. Herath**, B.Sc. Eng. (Hons) (Moratuwa)*
Department of Computer Science and Engineering,
University of Moratuwa.
Email: sachini.11@cse.mrt.ac.lk

***Mr. V. Dissanayake**, B.Sc. Eng. (Hons) (Moratuwa)*
Department of Computer Science and Engineering,
University of Moratuwa.
Email: vipula.11@cse.mrt.ac.lk

***Mr. S. Rasnayaka**, B.Sc. Eng. (Hons) (Moratuwa)*
Department of Computer Science and Engineering,
University of Moratuwa.
Email: sanka.11@cse.mrt.ac.lk

***Mr. S. Seneviratne**, B.Sc. Eng. (Hons) (Moratuwa)*
Department of Computer Science and Engineering,
University of Moratuwa.
Email: sachith.11@cse.mrt.ac.lk

***Mr. R. Vidanaarachchi**, B.Sc. Eng. (Hons) (Moratuwa)*
Department of Computer Science and Engineering,
University of Moratuwa.
Email: raji.11@cse.mrt.ac.lk

***Eng. (Dr.) C. D. Gamage**, C.Eng, MIE(Sri Lanka), Ph.D. (Monash), MEng (AIT), BSc Eng (Hons) (Moratuwa),*
University of Moratuwa, Moratuwa.
Email: chandag@cse.mrt.ac.lk

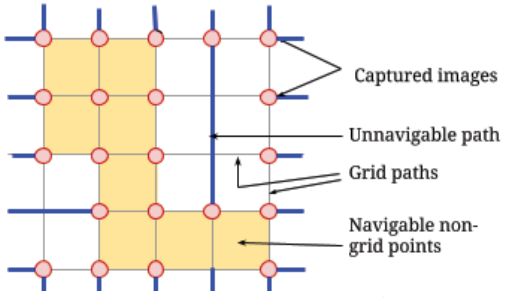

Figure 1 - Discrete capturing of a space

to ensure immersive experience, different types of input mechanisms were explored and a gesture based input system which allows for intuitive motions to get feedback from the user was finally selected[2].

The framework described here can be used to generate and visualize VR content at a very low cost of investment even by users with no technical expertise. Uses for such a framework are numerous including the exploration of architectural designs of houses by potential buyers in the real-estate market, news telecasting augmented with interactive scene exploration and virtual tours of sites of archaeological and tourist interest.

2. Related Work

The core areas of research relevant to the work presented in this paper are in navigability, user interaction, intermediate scene generation, and rendering of virtual reality environments.

2.1 Navigable Real World Environments

Capturing real world environments for navigable 3D environments was a major focus in the paper presented by Aliaga et al. [3] that proposed capturing panoramas and generating a grid for navigable walkthroughs. The proposed 4D plenoptic function interpolates feature positions within an image loop to reconstruct an intermediate view. However, this method requires a specialized omnidirectional camera for image acquisition and the output is not targeted at a VR environment.

A similar method was proposed by Uyttendaele et al. [4]. The method uses input from a six sensor omnidirectional camera, tracks feature points across images and resamples the images to new viewing planes to provide as the output. However, this method does not give full freedom of view to the user and requires the use of a game controller for navigation which does not give an intuitively immersive feel to the user.

2.2 Input to VR Systems

The two most popular methods of providing input to VR environments are by having external sensors like Kinect or by giving a device to the user's hands. However, both these methods require additional hardware which makes VR not much accessible to regular smart phone users. Therefore in this work, a gesture recognition method using sensors in built in mobile phones has been used. The method proposed by Ravi et al. [5]uses tri-axial accelerometer data to classify several categories of activities such as standing, walking etc. Kwapisz et al. [6]proposed a similar machine learning technique through which the data obtained from mobile sensors are aggregated over a sliding window for feature extraction. These feature extraction methods have been evaluated and adapted in the design of gesture based input system presented in this paper.

2.3 Intermediate Scene Generation

The generation of images at intermediate viewpoints which are not on the immediate path of the camera, is an area that has been sparsely researched upon and the available literature presents methods for generating intermediate content using image processing techniques when the panorama is visualized as cylindrical[3][7], cubic[8] or rectangular [9]. The methods used vary from resampling the neighbouring images in a grid obtained using an omnidirectional camera along with location calibration [3] and view interpolation using optical flow fields of the adjacent images [8]. View interpolation methods that require heavy computation have to be supported with dedicated graphic processors, GPU optimizations, buffering, and multithreading if high quality content in real time is to be achieved.

The generated panoramas are projected into spheres for rendering. Different methods of projecting panoramas into spheres are suggested in[4][10]and [11].

2.4 Scene Description Language

Scene description language (SDL) specifies how different components come together to create the VR environment along with the coordinate system used. An extensible markup language (XML) based SDL was developed by Lu et al. [12] for a virtual museum tour for which XML was chosen because of its interoperability, self-description, scalability, simplicity and flexibility. Tutenel et al. [13]used a semantic

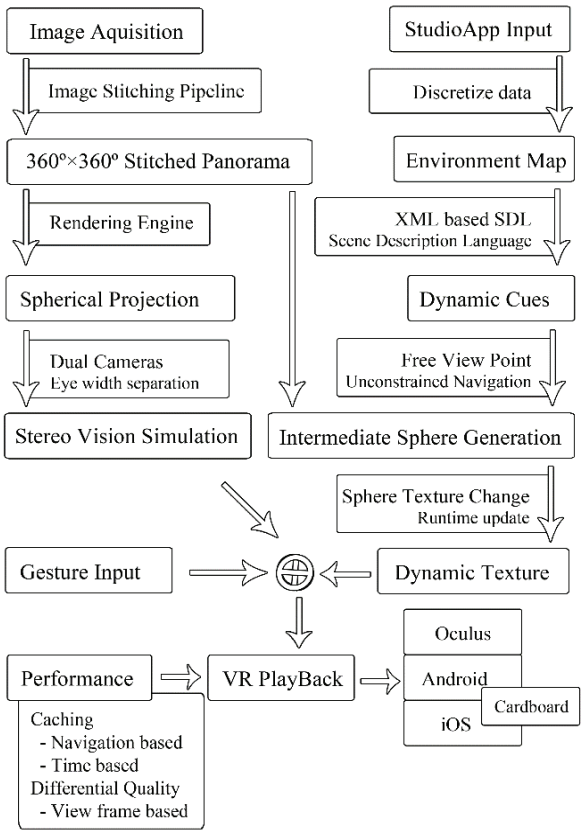

Figure 2 - System workflow

scene description language which specified objects, their inter-relationships and time dependent variations.

3. Methodology

The research work presented in this paper attempted at simplifying the generation and use of virtual reality content for the end user, to make it accessible to a wider user base. The major processing in this framework occurs in two independent workflows as depicted in Figure 2 -- (1) image processing tasks on captured equirectangular images and (2) generation of grids and navigable pathways using captured location data. Once the processing of the scenario is complete, it will be ready for visualization using VR playback platforms with required performance optimizations to ensure a smooth and immersive experience for the users.

The users can capture photospheres throughout the location and upload them to the studio application in a grid according to the sequence of the images. They will also be able to mark navigability information by giving restrictive paths such as obstacles and waterways. This information will be pre-processed by the studio application to generate a scene description language (SDL) encoding all the information.

At run time, this data will be used to dynamically generate navigation cues and allow the user to navigate freely in the recorded environment using an intuitive gesture based navigation system. A spherical surface isused for rendering the scenes as it provides a more realistic visualization when compared to cylindrical and cubic representations which give distortions at edges and flat surfaces thus failing to give a realistic feel to the user.

3.1 Input

To generate the required input, a simple photosphere app running on a mobile phone or a multi-camera rig capturing the entire $360^0 \times 180^0$ scene around a given point can be used. With the latter, the individual images have to be run through the image stitching pipeline to generate an equirectangular image. The capture device chosen in place of the system specific camera rig enables a wide range of users to use the system without having to make a significant initial investment in capture devices.

3.2 Image Stitching Pipeline

The captured image set for a specific location should contain a $360^0 \times 180^0$ field of view with considerable overlaps within the images. The images are stitched together using Matthew Brown stitching method[14]. This particular stitching method was selected based on a performance evaluation of multiple stitching algorithms conducted by Dissanayake et al. [15]. These image stitching algorithms include a stitching pipeline based on invariant features [14], a fast and memory efficient method for image stitching targeted at mobile phones[16], a stitching algorithm based on the use of Harris corner detection [17], and a stitching algorithm based on nonlinear blending [18]. The evaluation of both geometric and photo-metric criteria mentioned above include metrics such as universal image quality index[19],SAM[20]and SSIM [21]. Finally, the stitched image is corrected for lens distortion and skewing and fitted into a standard rectangular shape as a $360^0 \times 180^0$ panorama.

The final stitched image is given in equirectangular format which is a standard projection method for mapping a 3D globe into a rectangle.

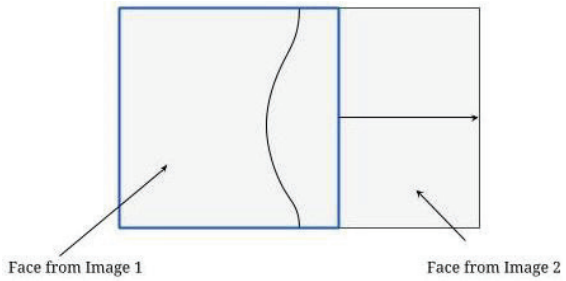

Figure 3 - Moving Window

3.3 Grid of Spheres as a Navigable Map

The environment was modelled as a 3-dimensional grid within the VR framework in which each grid point represents a corresponding location in the real world. At each location, a panorama is generated using captured images and rendered inside the spheres at the time of viewing.

The information relating to inter-sphere connectivity is stored separately for each sphere as an XML file using a SDL, to be embedded into the panorama.

3.4 Intermediate Views

To allow continuous navigability, the system needs to have scenes corresponding to each and every location. Although it is impractical to expect the user to capture these details, the generation of the scenes is quite challenging. Thus, an intermediate viewpoint generation system that uses adjacent equirectangular images to output a scene of acceptable quality was used.

In the generation of an intermediate view from the existing views, the transitions for faces that are perpendicular to the path of motion would be a zoom in or a zoom out, as no new objects would be introduced to the frame which is closer. For the faces that are in parallel to the direction of motion, new objects are added at the edge which will translate forward while objects near the opposite edge will be lost.

With this as inspiration, a methodology to process the adjacent equirectangular images to generate intermediate views was designed. As feature distortion present in equirectangular images makes feature based methods ineffective, they had to be transformed into the faces of the corresponding cube map.

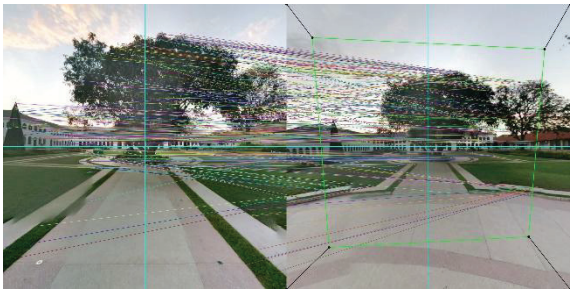

Figure 4 - Homography method on perpendicular faces

3.4.1 Faces Parallel to the Axis of Movement

The sides parallel to the axis of movement including the top and bottom views were stitched together and the movement was modelled using a moving window as shown in Figure 3 to extract the view for each intermediate location. This was calculated according to the distance travelled between captured viewpoints.

3.4.2 Faces Perpendicular to the Axis of Movement

The sides perpendicular to the axis of movement were mapped using a homography. The closer face (Image A) was warped onto the farther one (Image B) and using that, the positions of the four corners of the former image were located in the latter image. A distance based linear interpolation was carried out to find the position of the four corners in intermediate steps. Thereafter, these four points were used to calculate a homography for warping the quadrilateral into a square to create a side of the intermediate cube.

At the pre-processing stage, the faces were split into four quadrants and SURF[22] features were extracted from these quadrants to ensure that there was an even distribution of features among all quadrants. The feature points were then matched to the four corresponding quadrant pairs. This was possible because the movement was towards the centre of this face keeping the features in each quadrant within itself.

All the matches were finally aggregated for the voting in RANSAC[23] to calculate the homography which was used to warp Image A onto Image B. Using this warp, the four corner coordinates of Image A were calculated after the warp. These coordinates were saved in the pre-processing stage to be used in runtime to calculate the intermediate faces. This is shown in Figure 4.

3.4.3 Extending for arbitrary points

For a given point *(x,y)*, we had four sets of images A, B, C and D that are on the grid paths as shown in Figure 5. Therefore to generate any of the six faces of the cube, two methods could be used. Steps required to generate face *pos_x* would be as follows:

1 Get the face *pos_x* of A and C from the stitched images, and zoom the two images using the homography based method.
2 Get the face *pos_x* of B and D from zoomed images, stitch the two images and find the corresponding window.

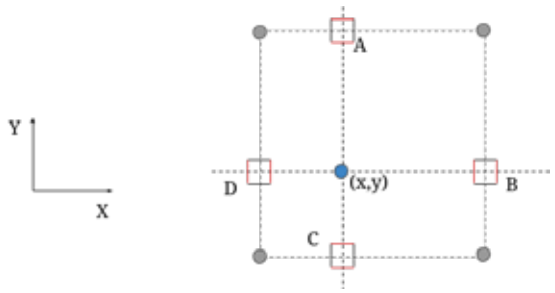

Figure 5 - Generating arbitrary points

The method to be used will depend on the accuracy of the first operation as the propagating error has to be minimized. The drawback, however, is that the operations which are shifted to the pre-processing stage because of their inherent computationally intensive and time consuming nature, would have to be done in real time. Although one way to overcome this problem would be to use Method 1, the distance based linear interpolation method was used instead of the homography based zoom method.

The distance based linear interpolation maintains that the features of the image closer to the intermediate view point are more prominent in the resultant image. For two faces perpendicular to the axis of movement, L and R, captured at a distance d from each other, the intermediate panorama I at a distance d_i from L will be given by:

$$I(x,y) = \alpha L(x,y) + (1-\alpha)R(x,y) \quad \ldots(1)$$

where $\alpha = d_i/d$. While this yields inferior results compared to Method 1 due to ghosting, the views can be generated in real time.

3.5 Scene Description Language

A special XML based scene description language (SDL)was proposed to store all required data and interconnections between the spheres. The SDL has to provide all the information regarding the interconnectivities of the spheres and virtual to real world mapping. Each sphere contains information on markers which upon activation by the user would take them to the adjacent sphere. This will form a graph of inter-navigable spheres laid out in a grid pattern.

The basic structure of the SDL is given in the tree structure shown in Figure 6.

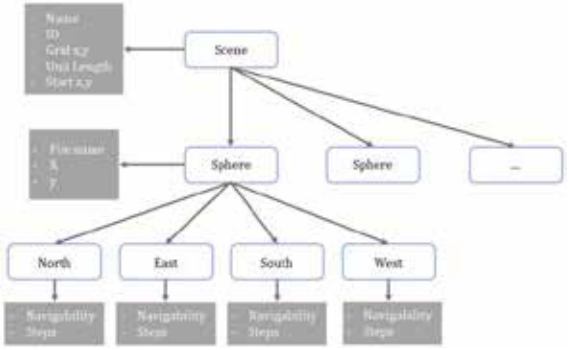

Figure 6 - Structure of SDL

Each scene will have a name and a unique identification code. Grid size will be recorded using *Grid x* and *Grid y* attributes. Unit length will give the distance between two grid points in the physical world. This value should be kept at a roughly equal value throughout the capture process. Each navigable environment will have a starting position which will be kept as Start x, Start y.

Within a scene, there will be multiple spheres. Each sphere will have a file name and a grid position which are given by the x and y values of each sphere. Each sphere will hold navigability information for the four directions north, east, south and west. For each path, Boolean flag navigability will define whether that path is navigable or not. If navigable, the number of grid jumps required to get to the next captured sphere will be given by the steps of each of these.

3.6 Gesture Input

In order to provide an intuitive experience in exploring the scene, markers were placed in the environment to indicate freely navigable areas. These are generally placed along the existing paths in order to provide the user with realistic exploration. The user will be able to switch to a slightly different viewpoint by leaning forward while looking at the marker. The placement of these markers was done by obtaining the relevant direction from SDL data and then placing a new marker object in the appropriate plane.

The gesture to move along the map is inspired by the motion control method of the segway in

which it is sufficient to simply lean forward to move forward and lean back to stop motion.

As there are significant differences among the gestures of different users depending on their height, flexibility etc., a model based on machine learning was used to differentiate between gestures on moving about and those that relate to movements that change the viewing angle of the VR system. The gesture shown in Figure is for the VR motion which is

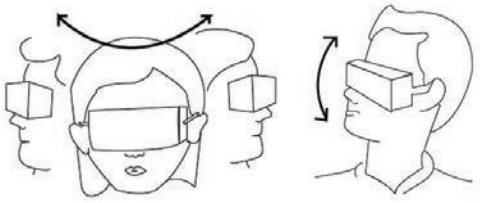

Figure 7 - VR Gesture

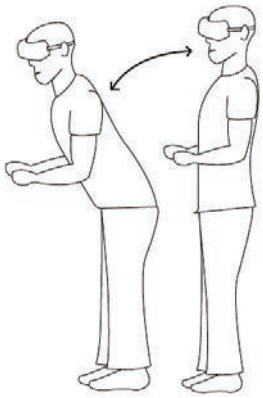

Figure 8 - Navigation gesture

exploring the $360^0 \times 180^0$ view as seen from a single viewpoint. The gesture shown in Figure is for navigation which is moving to the next available viewpoint in the direction faced by the user.

3.6.1 Data Collection and Pre-processing

Accelerometer readings which give the linear acceleration on x, y and z axes, and gyro sensor readings which give the angular velocity around x, y, and z axes were captured on change of value and aggregated over an interval of 0.2 seconds.

The nature of data collected (accelerometer and gyro-sensor data) was such that the data pre-processing needs were minimal. The data collected during the first and last few seconds were removed as a cleaning mechanism, as they usually relate to the time the user was putting on and removing the virtual reality headset. It was found that manually removing the effect of gravitational force is imprecise as there is a propagating error. Hence, raw accelerometer data were dropped and in their place the readings for linear acceleration were used.

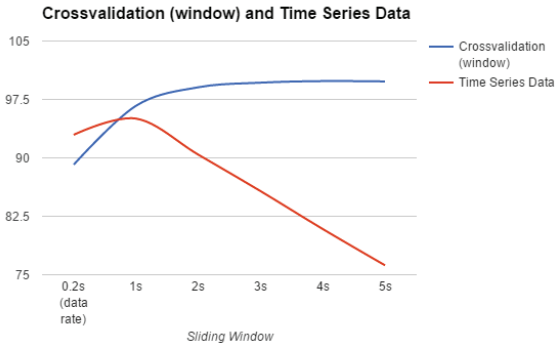

Figure 9 - Accuracy results

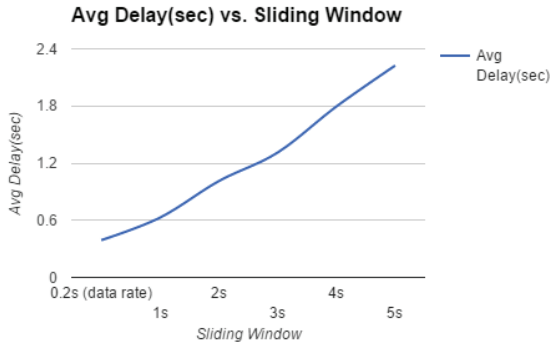

Figure 10 - Latency results

3.6.2 Feature Engineering

Feature extraction was done over a sliding window. Within the window, various metrics such as the mean, standard deviation and difference (final - initial) of the window's distribution were calculated and used as features.

In order to identify the optimal window size, performance measures for the current context were identified as given below.

- Accuracy: Value taken by doing cross validation on the random forest.
- Latency (delay): Time taken to identify the change in the prediction after class labels switch in time series data.

These values were measured while varying the window size. Figure 9 and Figure 10 illustrate the results. Based on the results, a window size of 2 seconds was selected as the size most appropriate to get satisfactory accuracy with low latency.

3.6.3 Results

The data were collected from 20 individuals who used the VR application during a span of three minutes each. The extracted features as explained above were used with three models: random forest [24], support vector machines

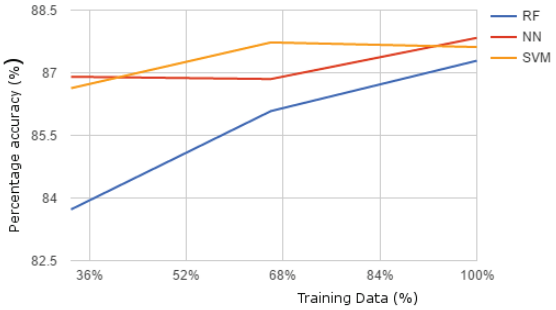

Figure 11 - Final Accuracy Results

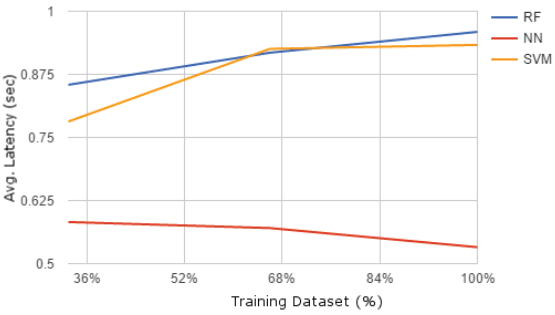

Figure 12 – Finallatency results

(SVM)[25] and artificial neural networks (ANN)[26].

The three algorithms were run on training data sets using 1/3rd, 2/3rd as well as the entire training set. A holdout testing set was used to get the accuracy of the training model. Accuracy results are shown in Figure 11. The latency was also calculated by varying the training set size and the method used. Figure 12 shows how the latency changed with these parameters.

From the final results obtained through the tests, it can be concluded that the model that best suits this application is the neural network. By using a neural network, it is possible to obtain a result with high accuracy and very low latency.

For the navigable VR application, a false classification as a navigation gesture will give a poor user experience. Therefore, the fallout value was calculated for the selected model. The fall-out rate for the ANN was 5.808% with the hold out data set. This was an acceptable rate for the application, as navigation gestures will only get triggered when the user is focusing the view angle onto a navigational cue.

4. Implementation

In the actual implementation of the said system, three major components could be identified -- firstly, the studio application which took user input as panoramic image spheres to be pre-processed, then the rendering and playback component which was at the output end of the system, and finally the various optimizations used at different stages of the system. These three components are discussed in this section.

4.1 Studio Application

Studio application as shown in Figure 13 obtains input images and data from the user through a web based or standalone application. Image stitching is carried out if required and the SDL is generated. All possible operations from the intermediate scene generation pipeline are pushed back to the pre-processing stage, thus reducing the workload on the mobile platform. Therefore, the following steps would be done as pre-processing at the back end of the studio application for any given data set.

1 All spherical images converted into box maps
2 Box maps split into square faces and stored according to a naming convention which encodes location data
3 Required faces stitched together using navigability information and the stitched rectangles stored according to the same naming convention
4 Navigability information encoded into the custom SDL and stored

Finally, a data bundle for the scenario is generated which can be directly exported to the playback application.

4.2 Rendering and Playback

The two most critical activities involved in rendering were projecting on a sphere and obtaining a stereo view.

4.2.1 Projection on a Sphere

The application for rendering the captured panorama images for viewing with navigation capabilities was developed using Unity 3D engine [1].

A custom shader for the unity engine gave the capability to overcome the transparency issue in back face culling and to wrap the panorama inside a sphere.

Figure 13 – Studio application interface

4.2.2 Stereo Playback System

Once the projection was done onto the sphere, a method for viewing it using VR rendering devices such as Google Cardboard or Oculus Rift was required.

In most of these VR systems, stereo visual content is required. This was solved by using multiple cameras on the unity game engine. Two cameras with a small displacement between each other in the horizontal axis were used to feed the left and right content streams to the two eyes.

4.3 Optimizations

Since the main target of this research was to enable this process to run on mobile devices, optimization of the application to reduce runtime workload and latency was essential, while maintaining the minimum data bundle size. The optimizations presented below are for generating intermediate views on grid paths, and non-grid locations using linear interpolation method as discussed in Section 3.

4.3.1 Pre-processing

The processing-heavy and time consuming portions of the view generation algorithm were completed in the pre-processing stage and only the six processed images were transferred to the front end along with the SDL. This included image stitching and locating zoomed coordinates using the homography based method.

4.3.2 Caching

It is possible to exploit locality and symmetry in this design, in order to reduce the computational cost of generating intermediate spheres. The spheres generated when moving from A to B towards east are the same as those that are generated when moving from B to A towards west. In this situation, it is possible to save the spheres and reuse them when making either transition. This reduces the overall computation considerably, as the total computation required for intermediate scene

Table 1 - Runtimes of the algorithm

	Without cache	Cached
Mean (s)	4.464035924	0.166760368
Std. Dev.	0.1125237	0.02903028

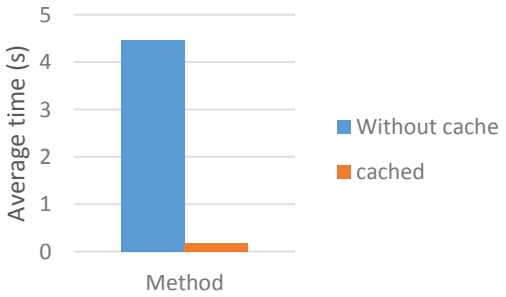

Figure 14 - Performance improvement by caching

generation in the whole scenario is bounded by $(n/2) \times$ *(computation cost of single sphere)*, where n is the total number of transitions between spheres, taking into account the direction as well.

4.3.3 Optimization of Texture Memory Allocation

The Unity primarily uses the Texture 2D class to perform operations on images. This includes image processing operations such as those used in the warp operations required during the intermediate sphere generation.

Since most of the transforms are primarily linear in nature, it is possible to allocate arrays of pixels instead of individually iterating them over all pixels in a 2D sub window and assigning them individually.

Performance improvements of these optimizations were tested on a mobile platform using two versions of the application, i.e., with and without optimizations. The time taken to navigate the same path within each of the two environments was measured and compared. The results that are shown in Table 1 and Figure 14 indicate that the cached version improves the average time by a factor of 4.

5. Conclusion

Through this work, we introduced a novel approach for creating interactive scenarios to explore 3D virtual environments in an unconstrained manner. We also introduced a novel gesture system which facilitates the intuitive exploration of the said environments based on machine learning. Additional optimizations including differential quality

panorama loading based on spatial locality that can increase the responsiveness have also been proposed.

A working product was created for different smartphone based VR playback devices such as Google Cardboard and tested using several different android devices. Evaluators found the interaction with the overall system to be largely smooth and intuitive with issues mostly arising due to the interaction with the viewing device and not from the platform itself.

Only subjective evaluations could be presented as there are no other similar systems that allow users to generate their own virtual reality environments and view them using a mobile device.

6. Future Work

In order to improve response time further, the locality within the virtual environment can be taken into account to pre-load as textures into the rendering engine, the required resources for all neighbouring directions, before the user makes a decision on his direction of movement. As this would be done in parallel with the viewing of the scene by the user, it would further streamline his viewing experience.

References

1. Unity, Unity - Game Engine, https://unity3d.com/, Visited, 17 Aug 2017.

2. Dissanayake V., Herath S., Rasnayaka S., Seneviratne S., Vidanaarachchi R. and Gamage C., "Real-Time Gesture Prediction Using Mobile Sensor Data for VR Applications," *International Journal of Machine Learning and Computing,* vol. 6, p. 215, 2016.

3. Aliaga D. G. and Carlbom I., "Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walk Throughs," in *Proceedings of the 28th Annual Conference on Computer graphics and interactive techniques*, New York, New York, USA, 2001.

4. Uyttendaele M., Criminisi A., Kang S. B., Winder S., Szeliski R. and Hartley R., "Image-based interactive exploration of real-world environments," *IEEE Computer Graphics and Applications,* vol. 24, pp. 52-63, May 2004.

5. Ravi N., Dandekar N., Mysore P. and Littman M. M. L., "Activity Recognition from Accelerometer Data," *In Proceedings of the 17th Conference on Innovative applications of artificial intelligence - Volume 3 (IAAI'05), Bruce Porter (Ed.),* Vol. 3. AAAI Press 1541-1546.

6. Kwapisz J. R., Weiss G. M. and Moore S. A., "Activity Recognition Using Cell Phone Accelerometers," *SIGKDD Explor. Newsl.,* vol. 12, pp. 74-82, Mar 2011.

7. Aliaga D. G., Funkhouser T., Yanovsky D. and Carlbom I., "Sea of Images," in *IEEE Computer Graphics and Applications*, 2001.

8. Kolhatkar S. and Laganière R., "Real-Time Virtual Viewpoint Generation on the GPU for Scene Navigation," in *2010 Canadian Conference on Computer and Robot Vision*, 2010.

9. Lee H., Tateyama Y. and Ogi T., "Realistic visual environment for immersive projection display system," in *2010 16th International Conference on Virtual Systems and Multimedia*, 2010.

10. Nielsen F., "On Representing Spherical Videos,"*IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Technical sketch, 2001.

11. *Photo Sphere XMP Metadata - Photo Sphere Google Developers,* https://developers.google.com/streetview/spherical-metadata Visited, 17 Aug 2017.

12. Lu W.,Zeng D. and Pan J., "An XML-Based Scene Description Language for 3D Virtual Museum," in *Proceedings of the International Conference on Information Technology Interfaces, ITI*, 2008.

13. Tutenel T., Smelik R., Bidarra R. and de Kraker K. and Jan, "A Semantic Scene Description Language for Procedural Layout Solving Problems," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

14. Brown M. and LoweD. G., "Automatic Panoramic Image Stitching using Invariant Features," in *International Journal of Computer Vision*, 2007.

15. Dissanayake V., Herath S., Rasnayaka S., Seneviratne S., Vidanaarachchi R. and Gamage C., "Quantitative and Qualitative Evaluation of Performance and Robustness of Image Stitching Algorithms," in *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2015.

16. Xiong Y. and Pulli K., "Fast Image Stitching and Editing for Panorama Painting on Mobile Phones," in *IEEE Transactions on Consumer Electronics*, 2010.

17. Zhang Y., Jia K. and Liu P., "Video Stitch Algorithm Based on Dynamic Foreground Extraction," in *International Congress on Image and Signal Processing*, 2009.

18. Chen L., Wang X. and Liang X., "An Effective Video Stitching Method," in *2010 International Conference On Computer Design and Applications*, 2010.

19. Wang Z. and BovikA. C., "A universal image quality index," *IEEE Signal Processing Letters,* vol. 9, pp. 81-84, March 2002.

20. Wald L., Data Fusion: Definitions and Architectures : Fusion of Images of Different Spatial Resolutions, *Presses des MINES,* 2002, p. 198.

21. Wang Z., Bovik A. C. A. C., Sheikh H. R. H. R. and SimoncelliE. P. E. P., "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing,* vol. 13, pp. 600-612, Apr 2004.

22. Bay H., Ess A., Tuytelaars T. and Van Gool L., "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding,* vol. 110, pp. 346-359, Jun 2008.

23. Lacey A. J., Pinitkarn N. and Thacker N. A., "An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2002.

24. Liaw A. and Wiener, M., "Classification and Regression by random Forest," *R News,* vol. 2, pp. 18-22, 2002.

25. Meye D. and Wein F. T., *Support Vector Machines: the Interface to libsvm in Package e1071,* vol. 1, 2014, pp. 1-8.

26. Günther F. and Fritsch S., *Neuralnet: Training of Neural Networks,* vol. 2, 2010, pp. 30-38.