A RATE DISTORTION FRAMEWORK FOR 3D BROWSING

Pietro Zanuttigh, Nicola Brusco, Guido M. Cortelazzo — University of Padova, Italy

David S. Taubman — UNSW, Australia

ABSTRACT

This paper contains an overview of a novel remote 3D visualization scheme. The scene is represented by a set of scalably compressed views and depth maps stored at server side. The JPIP transmission protocol is exploited to transmit the data in a progressive way while the interactive browsing goes on. The received images are then warped onto the required view and combined together. We develop a model to estimate the distortion in the rendered views and use it both to combine information from multiple views at client side and to select the data that need to be transmitted from the server.

Index Terms— Rendering, Image communication, Image coding

1. INTRODUCTION

Remote browsing of 3D scenes is a new problem posing many challenging conceptual and practical questions. An efficient solution requires a good understanding of how transmission resources should be distributed between the different elements of the scene representation. This includes the optimal subdivision between texture and geometry but also the definition of which elements of the scene description are more useful in the rendering of a particular view considering previously transmitted information.

In recent years highly scalable compression and transmission tools, such as JPEG2000 and its interactive protocol JPIP [1], have offered an efficient solution to the remote browsing problem for standard images. The work presented in this paper exploits some of these features and ideas for the interactive browsing of 3D models.

2. GENERAL FRAMEWORK

The proposed approach is based on a client-server scheme. The server holds the scene description as a set of views and depth maps of the scene together with the corresponding acquisition points and camera parameters. To achieve an efficient browsing over band-limited channels, all the information available at server side is compressed in a scalable way using JPEG2000. This compression standard has many scalability features that allow to access particular spatial regions and resolution levels in every image. In the proposed framework the server does not generate new views or compress differential imagery, instead, it is able to select and transmit only the part of the scalably compressed bit-streams that best fits the user's required viewpoint and the available bandwidth, taking into account also the previously transmitted data.

The JPIP interactive protocol [1] is very useful to transmit efficiently and progressively the scene description over the Internet. This protocol also allows the server to make its own decision on the information that needs to be transmitted to the client on the basis of the received requests. The client application exploits the data received from the server to render the views required by the user during the interactive browsing. This is achieved by reprojecting all the available images onto the required viewpoint using depth information and then combining all the warped views into the requested rendering. In the proposed approach the rendering process at the client and the server communication process are decoupled. After the user requests a particular view, the client communicates the new coordinates to the server, but at the same time it starts the rendering using the available information without waiting for the arrival of new data from the server. As soon as more data is received, it is added to the local cache and the rendered image is progressively improved.

An efficient remote visualization system based on this framework should solve two fundamental issues: the first is how the client should combine the information from the different views and depth maps into the required rendering; the second is how the available transmission resources should be distributed between the different elements of the various images and depth maps on the basis of the requested view and of the data already transmitted and stored into the client's cache.

3. DISTORTION-SENSITIVE VIEW SYNTHESIS

After the user requires a particular view V^*, the client reprojects each available view V^i onto it, obtaining a set of warped images $V^{i\rightarrow *} \triangleq \mathcal{W}^i(V^i)$, where $\mathcal{W}^i$ is the warping operator that maps the samples of V^i to the corresponding positions in V^*. The warpings cover overlapping regions in the required rendering and it is thus necessary to combine them together. The simplest solutions to combine the information from multiple images such as averaging and stitching tend to cause blurring or discontinuities. To avoid these issues it is possible

to perform stitching within a multi-resolution framework. We decided to perform the stitching inside the subbands of the Discrete Wavelet Transform. The stitching procedure is made of 3 steps. Firstly each warped image is decomposed using a D level DWT, then the contributions from the different images are stitched separately in each resolution component and finally wavelet synthesis is applied to reconstruct the required image.

The missing fundamental point is how to decide which source view is going to be used for every sample in the various subbands. Our approach is based on the estimation of distortion in the rendered view and the subsequent choice of the source views that minimize the distortion. In [2] we developed a model to estimate how the distortion from the various source images is mapped to the final rendering. Three main sources of distortion are taken into account: the first is the quantization error in the DWT samples introduced by image compression: the error in every sample in subband b of the image V^i finds its way into resolution component $R_d^{i\rightarrow *}$ of $V^{i\rightarrow *}$ through DWT synthesis, warping and further DWT analysis. The quantization error contribution to $R_d^{i\rightarrow *}[\mathbf{p}]$ can be approximated by multiplying the source code-block distortions by a collection of pre-computed distortion weights ($W_{b\rightarrow d}^i$), depending on the properties of the locally affine warping operator.

$$D_{quant,d}^{i\rightarrow *}[\mathbf{p}] = \sum_b W_{b\rightarrow d}[\mathbf{p}] \cdot D_b^i \left[\left(\mathcal{W}_{b\rightarrow d}^i \right)^{-1} (\mathbf{p}) \right] \quad (1)$$

where $D_b^i[\mathbf{k}]$ denotes the mean squared distortion at location $\mathbf{k}$ in subband b of view V^i and the weights $W_{b\rightarrow d}[\mathbf{p}]$ represent how the distortion is mapped from the source subband b in V^i to the target resolution component d on V^* on the basis of the surface warping operator $\mathcal{W}^i$. We are using $\left(\mathcal{W}_{b\rightarrow d}^i\right)^{-1}$ for the operator which maps locations $\mathbf{p}$ in the warped resolution component R_d, back to the corresponding location in subband b of V^i. The second source of distortion we consider is the uncertainty in the geometry description due both to the precision of the acquisition procedure and to the lossy compression of the depth maps. The error in the geometry description causes a translation uncertainty in the position of samples in the warped images. We exploited the results obtained in [3] to estimate the distortion due to geometry uncertainty. The computation procedure for this term is presented in [2]. Finally we take into account also the distortion due to effects such as shading and reflection, that roughly grows with the angle between views V^* and V^i. Combining all these contributions we obtain a distortion model of the form

$$D_d^{i\rightarrow *}[\mathbf{p}] = \Theta_d^{i\rightarrow *}[\mathbf{p}] + \sum_b W_{b\rightarrow d}[\mathbf{p}] \cdot D_b^i \left[\left(\mathcal{W}_{b\rightarrow d}^i \right)^{-1} (\mathbf{p}) \right] \quad (2)$$

where $\Theta_d^{i\rightarrow *}[\mathbf{p}]$ represents the contribution of geometry uncertainty and lighting effects.

The geometry is represented as a set of depth maps taken from some fixed points, usually coincident with the viewpoints of the available images V^i, while the warping and distortion mapping requires the availability of depth information for the required rendering V^*. The client must synthesize this depth information Z^* from the data on the available depth maps received from the server. The same procedure we used to synthesize the views can be used also to reconstruct Z^*. Each depth map Z^i is transformed into a corresponding estimate $Z^{i\rightarrow *}$ for Z^*. Then the distortion in the compressed source depth maps Z^i is mapped to Z^* through a set of weights that depend on the locally affine warping operators and finally a minimum distortion criterion is used to select the samples for the stitching procedure. The only difference is that now the stitching is performed directly at full resolution without the multi-resolution framework because the smoothing introduced by the wavelet transform can damage discontinuities in the depth values.

4. SERVER TRANSMISSION POLICY

The second critical issue is how to select which parts of the compressed bit-streams corresponding to the various images and depth maps need to be transmitted. The images and depth maps are scalably compressed using JPEG2000 and the compressed data stream is divided into many contributions corresponding to the different resolution levels and spatial regions (called *precincts* in JPEG2000). These themselves are divided into different quality layers. In every time interval (*epoch* in JPIP) the server must decide how to allocate the available bandwidth between all these contributions in order to obtain the best image quality at client side on the basis of the required view and of the already transmitted information.

To solve the server optimization issue we can exploit the distortion framework used at client side. The total distortion associated with the reconstructed view V^* can be obtained by summing up the contributions coming from the samples taken from the different views:

$$D^* \approx \sum_d \sum_{\mathbf{p}} \sum_i \left(\rho_d^i[\mathbf{p}] \right)^2 D_d^{i\rightarrow *}[\mathbf{p}] \quad (3)$$

where $\rho_d^i[\mathbf{p}]$ are the blending weights (usually $\rho_d^i[\mathbf{p}] = 1$ if the sample is taken from V^i and 0 otherwise). For simplicity at present we assume that geometry has already been transmitted and we solve the optimization problem for the transmission of the image data. The target is to deliver the precinct data-bin packets that minimize the objective D^* subject to the constraint on the data that can be transmitted in each epoch $L \leq L^{\text{epoch}}$. This problem can be reformulated in Lagrangian fashion as a family of unconstrained optimization objectives. The optimization is complicated by the fact that the blending weights $\rho_d^i[\mathbf{p}]$ themselves depend upon the local distortion in the received images. An optimal solution would require to recompute the weights at each iteration of the Lagrangian procedure, but it is too slow for a real-time system. We instead decided to consider two sets of blending weights: $\mathring{\rho}_d^i[\mathbf{p}]$

denotes the weights which the client is currently using, and $\vec{\rho}_d^i[\mathbf{p}]$ represent the ones which would be used if all source views were available in uncompressed form.

The server considers two types of enhancement to the client's existing cache contents that we called *reinforcing enhancements* and *disruptive enhancements*. Reinforcing enhancements are based on the assumption that the blending choices will not change between this epoch and the next, so that $\rho_d^i[\mathbf{p}] = \mathring{\rho}_d^i[\mathbf{p}]$. In this case $\Theta_d^{i\rightarrow *}[\mathbf{p}]$, that does not depend on the image distortion, is multiplied by the constant blending weights in the computation of D^* and represents a constant offset that can be excluded from the optimization formulation. By expressing the objective in term of precinct data-bin decision we obtain a set of independent optimization objectives for each precinct that the server can easily solve to determine reinforcing enhancements. Using these enhancements the server tends to send more information for those code-blocks which already contribute most strongly to the client's current view synthesis process.

Disruptive enhancements are instead based on the assumption that the blending weights used by the client will change from $\mathring{\rho}_d^i[\mathbf{p}]$ to $\vec{\rho}_d^i[\mathbf{p}]$ once all data in this epoch has been transmitted, and usually force the transmission of data from better aligned views that have not yet been transmitted. The switch from an image for which a good amount of data has already been transmitted to a closer one for which no (or only a little) data is available, leads at the beginning to an increase of distortion that we call "policy switching penalty". This penalty is compensated by the fact that the distortion associated with better aligned views decreases faster as more data is received and the best image quality at client side can ultimately be reached only by using the better aligned views. Disruptive enhancements become useful when the policy switching penalty is compensated by the delivery of sufficient bytes from the data-bins in the new view. A critical question, then, is how to determine the right point at which we should start sending them. A final remark is that these enhancements are expected to cause the client synthesis policy to change its blending weights. Since a transmission epoch has limited duration and disruptive enhancements have a minimum length, we might only be able to introduce them in a few local regions at a time.

The final step is to combine together the two procedures. Reinforcing enhancements are based on the assumption that $\rho_d^i[\mathbf{p}]$ will not change, which means that there should be no disruptive enhancements. The optimal solution would be to first determine the disruptive enhancements and then recompute $\rho_d^i[\mathbf{p}]$ before finding reinforcing enhancements for each iteration of the optimization algorithm. Again this approach is infeasible in a real-time system. Instead, it is possible to simply take the maximum of the amount of data yielded by the two methods for each precinct and adjust the Lagrange multiplier in an outer loop until the bandwidth constraint is satisfied. Even if this solution is not completely optimal, the sub-optimality is reduced by the fact that the blending weights will be recomputed in the next epoch, so that any sub-optimality represented by the "max of solutions" approach is limited by the size of the epoch.

5. EXPERIMENTAL RESULTS

To evaluate the performance of the system let us start from a simple example. The user has been looking at a 3D model of the cartoon character *Goku* from the viewpoint of view V^1 (shown in Fig. 1 and associated to the light green colour). Then he moves to a new viewpoint, corresponding to view V^2 (dark green). At the beginning not enough data is available for V^2 and most samples are warped from V^1. Figure 2a shows the corresponding rendering, most of the image is warped from V^1 and has a good quality, only the leftmost samples that are not visible in V^1 are taken from the other view and show a poor quality. As soon as more data is received for V^2 its distortion is reduced and more and more samples are taken from it. Figure 1 shows how the blending choices progressively move to V^2. The rendering corresponding to V^2 at 0.2bpp has already a good quality on the whole image (see Fig. 2b). Finally, when the complete description of V^2 is available, as expected most samples are taken from it. This example shows clearly how the proposed system can handle a real-time browsing of the scene by firstly rendering the required view by warping the previous ones and then progressively moving to the new one as more data becomes available for it.

Fig. 1. *V^1 , V^2 and blending choices with V^1 at 0.8bpp and V^2 at different bitrates*

Fig. 2. *Rendering: a) V^1 at 0.8bpp and V^2 at 0.0125bpp, b) V^1 and V^2 at 0.8bpp.*

Moving to server side, an interesting problem is how the server should behave when some data has already been transmitted to the client and the user requires a new view of interest V_f. In the following example three images of the *Goku* model are available at sever side: one is the required view V_f and is taken in the front of the object, while the other two, V_l and V_r are taken at the left and right side of it. The client has already received around 2KB of data on the two images V_l and V_r. No data for view V_f is available at client side.

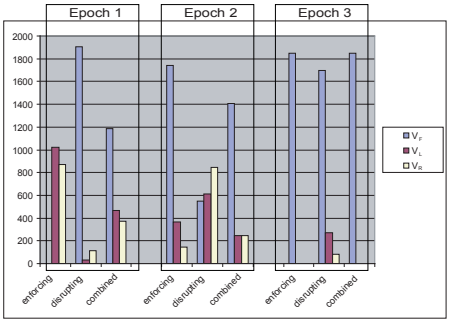

Fig. 3. *Bytes allocation between different views.*

The plot of Fig. 3 shows the results of the reinforcing step alone, of the disruptive one and the combined solution for the first three epochs. As it is possible to see from the first group of columns, in the first epoch reinforcing enhancements force the transmission of data for the two side views because no data is available for the front one and no samples are taken from it. V_f is is perfectly aligned with the required rendering and disruptive ehancements force the transmission of more and more data from it in the next epochs. At the beginning disruptive enhancements turn larger than reinforcing ones, but the system will continue to send data also for the two side views because in some precincts of V_f the additional data transmitted in this epoch will not decrease the distortion enough to compensate the switching penalty. But as soon as more data is transmitted more samples are taken from the front view and from epoch 3 most of the data is transmitted on V_f. Even if it seems that at the end the system will take a bit longer to receive the complete description, the bytes "wasted" at the beginning on the side views allow to obtain an acceptable rendering in epoch 1 and 2 even if not many bytes are available for V_f yet. Figure 4 shows a detail of the rendered images in the different epochs (we provided the visual results instead of PSNR curves because illumination issues and translational shifts due to imperfections in the geometry introduce a large contribution to the MSE that does not correspond to the perceived visual quality of the image). It compares the results obtained by just transmitting V_f (upper row) with the combined use of the three views (in the lower row). It shows clearly that the proposed approach allows to obtain a much better quality in early epochs by exploiting the data transmitted during the previous browsing.

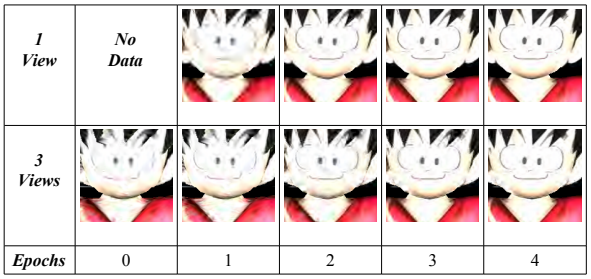

Fig. 4. *Detail of Goku in different epochs.*

6. CONCLUSIONS AND FUTURE WORK

In this paper an efficient remote visualization scheme for 3D scenes is described. The rate distortion framework we developed to estimate the distortion in the rendered views has been used not only to combine together the information coming from different views at client side as in [2] but also to decide which part of the scalably compressed data need to be transmitted at server side. Experimental results show how this approach allows to transmit only the information really necessary for the current view and to exploit the already transmitted data in the rendering of new views. The proposed approach allows also to effectively exploit some ideas and techniques behind the JPEG2000 and JPIP standards for image compression and transmission in the field of remote 3D browsing.

The current server implementation deals with the transmission of the contributions to the texture information. Geometry transmission can be optimized independently using the same procedure we used for the images and the geometry dependent component in the distortion formulation can be used to understand how geometric distortion affects the rendered image and to optimally subdivide the bandwidth between texture and geometry. Further research will deal with the problem of the combined transmission of texture and geometry.

7. REFERENCES

[1] D. Taubman and R. Prandolini, "Architecture, philosophy and performance of jpip: internet protocol standard for JPEG 2000," *Int. Symp. Visual Comm. and Image Proc.*, vol. 5150, pp. 649–663, July 2003.

[2] P. Zanuttigh, N. Brusco, D. Taubman, and G.M. Cortelazzo, "A novel framework for the interactive transmission of 3d scenes," *Signal Processing: Image Communication*, vol. 21, no. 9, pp. 787–811, October 2006.

[3] A. Secker and D. Taubman, "Highly scalable video compression with scalable motion coding," *IEEE Trans. Image Proc.*, vol. 13, no. 8, pp. 1029–1041, Aug 2004.