A NEW APPROACH FOR 3D MODELS TRANSMISSION

A. Guarnieri [a], F. Pirotti [a], M. Pontin [a], A. Vettore [a]

[a] CIRGEO, Interdept. Research Center of
Cartography, Photogrammetry, Remote Sensing and GIS
University of Padova, Agripolis
35020 Legnaro (PD) - Italy
e-mail: antonio.vettore@unipd.it ; cirgeo@unipd.it

KEYWORDS: 3D modeling, Internet, data compression, documentation, data retrievial, VR

ABSTRACT

The visualization and interactive navigation of 3D models, like virtual visits of museums, churches or even whole cities (cybercity models), are gaining a more and more increasing use and importance with time. This has been accomplished in last years thank to the augmented power of modern CPUs, graphic hardware devices and thank to a wider access to the web. Even in the survey field the interest towards 3D models has grown, given the advantages offered by ground-based laser scanners in terms of survey time and the pretty good quality of obtained 3D models. Indeed, several tests and applications have been performed so far all around world with such instruments: 3D data have been collected from objects of different shape and size, ranging from cultural heritage to land infrastructures, accuracy has been investigated and corresponding 3D models have been created using appropriated software. Beside issues related to the accuracy and the quality of resulting models, another interesting topic should be considered: the remote use of these 3D models, i. e. how they can be optimally transmitted over the web.
To this aim we have developed a VRML split-browser, i.e. a 3D model visualization shared system based on two different server–side and client-side applications. The former executes the rendering of the scene and send it to the client, which is located on the remote PC of the user. In turn, the client provides the user with a graphical interface to explore the 3D model interactively.
The system is based on image compression and projective transformations, which allow to share the 3D models over the web with limited throughput to the client. In this paper, we will present the frame prediction algorithm, based on a space displacement of the viewpoint and the way adopted to apply it in the remote visualization system.

1. INTRODUCTION

Over last times, the 3D modeling of real objects, of whatever size and complexity, is gaining a more and more interest and importance by the scientific community. For instance, the availability of close-range and long-range ground-based laser scanners, which are able to collect several thousands of points in short time, have increased the interest towards 3D models as a useful and alternative mean for analysis and study in the field of medicine, industry, cultural heritage, engineering, architecture, etc. Tests and applications have been extensively performed all around the world in order to assess the accuracy and the quality of those 3D models. Beside such topics, another interesting issue should be considered: the remote use of these 3D models, i. e. how they can be optimally transmitted over the web.
To this aim we have developed a VRML split-browser, i.e. a 3D model visualization shared system based on two different server–side and client-side applications. The main advantage of proposed solution relies on the fact that the server sends to the client a periodic image sequence, composed by a reference image and a set of error images, instead of the whole VRML model. Basically, the server executes the rendering of the scene and sends it to the client, which is located on the remote user's PC. In turn, the client provides the user with a graphical interface to explore the 3D model interactively. Thus, using suited image compression techniques and projective transformations, 3D models can be shared over the web with limited throughput to the client.
Though the developed technique looks like the algorithms implemented in the MPEG compression, in this case the key-advantage relies on the a-priori knowledge of the image transformation parameters, as they are determined directly by the request of viewpoint change made by the user through the client GUI.
In this paper, we will present the frame prediction algorithm, based on a space displacement of the viewpoint and the way adopted to apply it in the remote visualization system.
The paper is structured as follows. In the section 2 a short overview of the VRML language is provided, highlighting the disadvantages that show up when large 3D data sets are employed. In section 3 we describe the framework of our VRML split-browser, we developed in order to improve the remote transmission of 3D data. Then, section 4 focuses on the mathematics which the implemented frame compression algrotihm is based on, while in section 5 the results of the tests performed using simple geometric shapes are presented. Finally, section 6 reports the conclusions.

2. THE VRML LANGUAGE

Technically speaking, VRML is neither virtual reality nor a modeling language; at its core, VRML is simply a 3D interchange format. It defines most of the commonly used semantics found in today's 3D applications such as hierarchical transformations, light sources, viewpoints, geometry, animation, fog, material properties, and texture mapping. One of the primary goals in designing VRML was to ensure that it at least succeeded as an effective 3D file interchange format. Moreover, VRML can be considered as a 3D analog to HTML. This means that VRML serves as a simple, multiplatform language for publishing 3D Web pages. This is motivated by the fact that some information is best experienced three dimensionally, such as games, engineering and scientific visualizations, educational experiences, and architecture. Typically these types of projects require intensive interaction, animation, and user participation and exploration beyond what is capable with a page-, text-, or image-based format (i.e., HTML). VRML provides the technology that integrates three dimensions, two dimensions, text, and multimedia into a coherent model. When these media types are combined with scripting languages and Internet capabilities, an entirely new genre of interactive applications are possible.

In 1997 a community of programmers, engineers, graphers founded the VAG (VRML Architecture Group) in order to extend the capabilities of the previous VRML version (1.0), which feautured several properties of the Inventor File Format, a graphic standard developed by Silicon Graphics. The overall goal for the new VRML vs. 2.0 was fairly modest: to allow objects in the world to move and to allow the user to interact with the objects in the world, allowing the creation of more interesting user experiences than those created with VRML 1.0. The vs. 2.0 featured following properties:

1) *Composability*, it should be relatively easy to take files created by various people or tools and compose them together to create a new document. This is another property that VRML shares with HTML: It is easy to cut and paste text from several HTML documents using either a generic text editor or a specialized editing tool, just as it is easy to cut and paste objects between VRML worlds.
2) *Scalability*, is a constraint on the VRML 2.0 design. VRML is designed to scale in three ways. First, it should be theoretically possible for a VRML browser to handle a world distributed across the Internet that contains millions or billions of objects. Second, VRML should work well when used with both very powerful and very inexpensive machines, allowing the VRML browser to trade off image or simulation quality for improved performance and to scale well with increased hardware performance. And third, VRML worlds should scale with network performance, from the 14.4K modems that are common today to multigigabit connections that might become common in the future.
3) *Extensibility*, the language should allow the developer to add functionalities, like new geometric primitives, for instance.

The VRML format allows, therefore, to create 3D environments, ensuring basic properties like full web access to the model and platform independence: only a suitable viewer is needed, capable to interpret the VRML files.

Such interactive technology can be profitably employed in different application fields, like for example the virtual visit of a museum or even of a whole city (cybercity). E-Commerce is another interesting topic: a consumer could virtually explore, in all its parts, the 3D model of the object to be purchased, before to move to the shop. More recently, the availability of 3D models of cultural heritage elements and land infrastructures (like buidlings, churches, bridges, road assets, etc.) contributed to increase the use of VRML as a mean to share such models through the web. However, the interest of the scientific community (medicine, engineering, architecture, etc) towards 3D models has revealed in the same time some limits of the VRML as interchange file format when large data sets, acquired typically by close-range or long-range ground-based laser scanners, are used. Major disadvantages are the following:

1) *File size*. It is not uncommon to manage VRML files of hundreds of MBytes;
2) *Input peripherals*. Interactive user commands are activated through mouse and/or keyboard, which often are not the optimal peripherals to navigate in the 3D environment;
3) *Computing resources*. A VRML viewer needs often high computational performance in order to manage and render a complex scene, composed by thousands of points or triangles.

Obviously, a more profitable way to share complex and large 3D models requires the development of solutions aimed to overcome above mentioned issues.

3. THE VRML SPLIT-BROWSER

Beside the VRML language, the structure of the browser/viewer has to be considered, as well. Basically, it is a software application which is able to display a 3D scene, whose content is described in a VRML file. Moreover, a specific GUI (Graphical User Interface) is provided to the user along with a suited set of commands, allowing to navigate the 3D environment. A VRML viewer works according to a stratified structure, whose main components can be summarized as follows (fig. 1):

1) *Parser*. It checks the grammatical and sintactical content of the file. If any error is encountered, the file is discarded;
2) *World loader*, which deals with the loading of all elements, like textures, sounds and scripts, required to display correctly the virtual world content;
3) *World manager*, whose task entails with the loading in the system memory of the representation of the virtual scene. Furthermore some basic components of the scene have to be tracked, like object positions, position and orientation of the eye-point, position and properties of light sources, object materials and the viewing volume. The world manager can be considered as the analog of the graphical engine used for the 3D models processing;
4) *World renderer*, which works together with the world manager to correctly display on the user's monitor the content of the VRML file. The renderer checks the scene continuously, if any change is detected (displacement of the user's eye-point) the new image to be displayed is computed as fast as possible. Of course this task requires high computational effort from the user's hardware, which is not always the case.

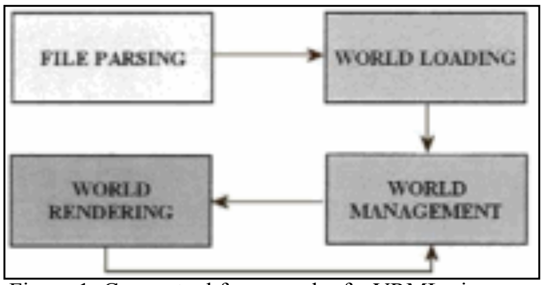

Figure 1: Conceptual framework of a VRML viewer

In order to solve for the issues described in previous section, a VRML split-browser has been developed. Its working is based on a computer network model, where a central processor (the server) provides different kind of services to a number of hosts (clients). In practice, this kind of viewer is obtained by application of a client-server model to a generic VRML viewer. As shown in figure 2, the proposed model clearly distinguishes between server-side and client-side tasks. The server manages the whole virtual environment computing new scene views, processing user's commands and executing scripts or animations. From the client-side, an interactive GUI is provided to the user by which the virtual views sent form the server are displayed. Moreover, the user can interact with the server by means of a number of navigation commands available on the client's GUI. After an initial setup phase, needed to negotiate the parameters of the communication protocol (e.g. maximum throughput accepted by the client), a bidirectional communication loop begins between server and client. In practice, the user activates a command in the client's GUI, through mouse and/or keyboard. Such command is transmitted to the server as event, i.e. as a request of scene change. Then the server intreprets the event and, if possible, sends back to the client the new view, which will displayed on the user's GUI. The described loop is repeated according to the user interaction.

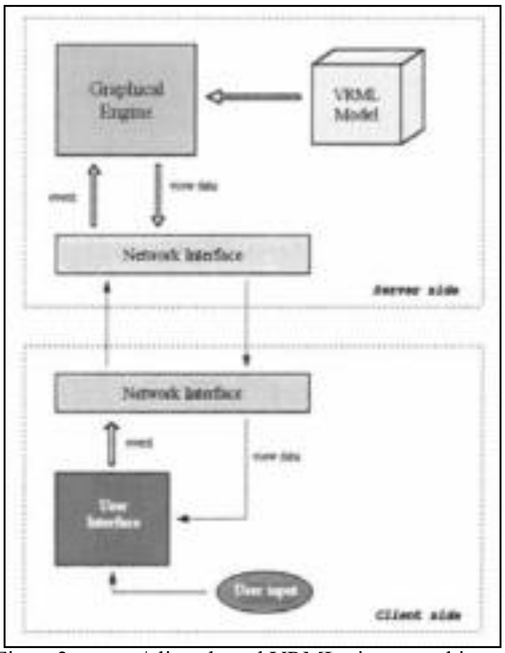

Figure2: server/client-based VRML viewer architecture

This client-server approach shows following advantages:

1) The amount of data sent to the client is greatly reduced respect with the transmission of the whole 3D model in VRML format. Indeed the server sends to the client an image sequence only: adopting an image compression technique, the throughput could be further reduced;
2) The client doesn't need high computational capabilities, as the world management step is fully performed on the server;
3) Since server and client are independent each other, the LOD (Level of Detail) of each scene can be interactively changed by the client ;
4) The system features a certain degree of portability, as the source codes can be modified accordingly to the Operative System on which the server-side and the client-side applications should run.
5) The copyright on the 3D model is keeped, as the user will never receive on his PC the whole VRML model, but only a set of images corresponding to the actual scene displayed on the monitor screen;
6) The user's GUI is managed by the client only, independently from the server. In this way a higher effectiveness of the whole system is ensured, since the server-side of the viewer doesn't need to dedicate computing resources for the user's GUI management.

In order to allow the client to navigate the 3D model as fluent as possible, regardless the size and complexity of the model and client's computing capabilities, above mentioned features of our proposed split-browser are still not sufficient. An image compression technique has to be implemented to get an optimized client/server communication protocol. To this aim two different compression algorithms have been investigated, the LZ77 and the JPEG, comparing the results of their applications to our split-browser transmitting simple geometric 3D shapes. However obtained results revealed a limited compression capability (higher for the LZ77 respect with the JPEG) for our goals. This can be explained considering that those algorithms are "general purpose": they can be applied to whatever kind of file, regardless its content. Therefore we developed an "ad hoc" compression technique which would exploit the "knowledge" about the context where it should be applied. The description of this technique will be the topic of the following section.

4. THE FRAME COMPRESSION ALGORITHM

The alternative procedure we developed in order to further reduce the throughput between server and client is based on the application of projective transformations. The underlying idea is that subsequent views, processed by the server and displayed by the client to the user, are joined each other through such kind of transformations. Since the knowledge about the geometry of the virtual environment (e.g. eye-point position, viewing direction, etc) is needed, projective transformations don't allow to preserve the same level of flexibility and versatility as LZ77 and JPEG image compression algorithms. Therefore our method can be applied to specific cases only, where geometry related images are handled. As shown in figure 3, the server/client communi-cation procedure works as follows:

Server-side operations

1) Given the user input (scene change) at time *n*, the server processes the request generating the corresponding new view of the 3D model (view B).
2) As the geometrical parameters describing the 3D scene at time *n-1* and *n* are known, the server can determine the projective transformation, mapping view A to view B. Computed parameters are then sent to the client.
3) The view A, describing the 3D scene at time n-1, is applied the previous computed projective transforma-tion. Consequently, a *predicted view* is generated, which should represent a good approximation ov view B. the prediction is generated by transforming the reference frame (view A) on the basis of the displacement of the user point of view (known from the selected command of the GUI) and the distance from the image pixel projection plane (known from the content of the Z buffer).

4) Then, a pixel by pixel difference between view B and the predicted view is calculated by the server. A so called *error-image* is obtained, which is compressed with LZ77 algorithm and sent to the client.

Client-side operations

1) The client receives the projective transformation parameters and the error-image, which is decompressed, retrieving the original image.
2) The projective transformation is applied to the view A, generating a so called *predicted view*.
3) Then, the error-image is added pixel by pixel to the predicted view, in order to create the requested view B, which is then displayed by the client in the user's GUI.

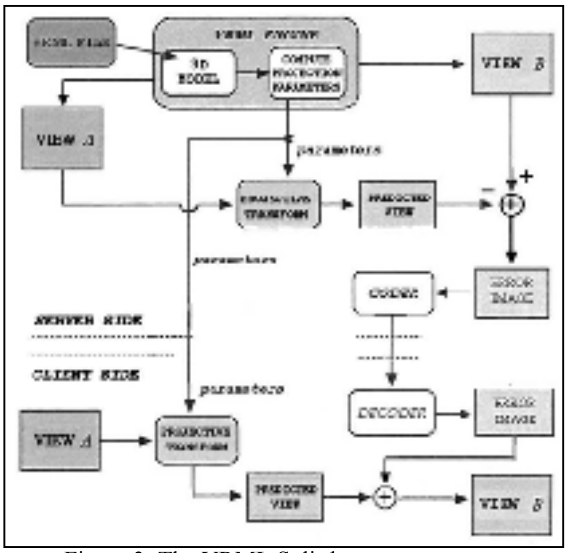

Figure 3: The VRML Split-browser structure

The decription presented above shows that the computational load on the client is greatly reduced, respect with the classical approach (transmission of the whole VRML model). Indeed, the client is required only to apply the projective transformation to the current view and to decompress the error-image as fast as possible. Nowadays, requested computational capabilities to perform such tasks are commonly available on most home desktop PCs.
In the following subsection, some mathematics will be provided about the developed frame compression algorithm.

4.1 The projective transformation

A projective transformation can be considered as a subset of the more general group of coordinate transformations, which maps a given input 2D image point $\mathbf{x} = [x_1, x_2]$ into a new image point $\mathbf{y} = [y_1, y_2]$. Adopting a matrix notation, this mapping fucntion can be defined as follows:

$$\mathbf{y} = \frac{\mathbf{Ax}+\mathbf{b}}{\mathbf{c}^t\mathbf{x}+d} \qquad (1)$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \; ; \; \mathbf{b} = [b_1\ b_2]^t \; ; \; c^t = [c_1\ \ c_2] \; ; \; d \in R$$

Typically, each projective transformation is associated a matrix $\mathbf{P} \in \Re^{3x3}$, called *projective matrix*, allowing a more compact notation for eq. (1):

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \vdots & \mathbf{b} \\ \cdots & \cdots & \cdots \\ \mathbf{c}^t & \vdots & d \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ c_1 & c_2 & d \end{bmatrix} \qquad (2)$$

In the following, the procedure adopted to compute the projective transformation parameters will be described using a simple polygon as target object in the 3D space. This assumption doesn't limit the effectiveness of our procedure, since it is well known that each 3D object can be described in terms of a number of interconnected polygons (typically triangles). Therefore, in principle, it would be sufficient to apply the algorithm to each composing polygon.
According to figure 4, the aim of our procedure deals with the computation of the position of point T_2, by application of a projective transformation to point T_1 (identified by $\mathbf{P}_3$).

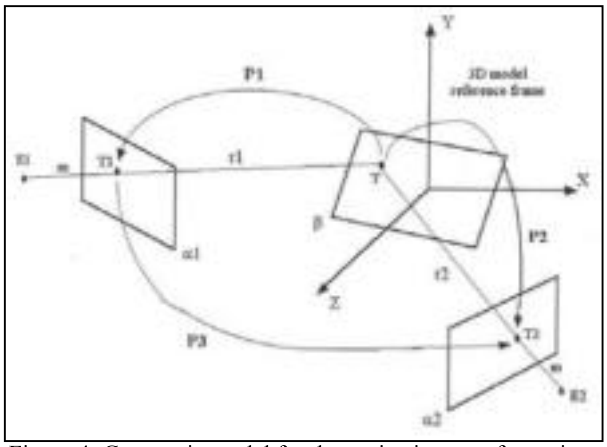

Figure 4: Geometric model for the projective transformation

If projective transformations P_1 and P_2, mapping T on T_1 and T_2 respectively, are known, following relationship can be established:

$$\mathbf{T}_2 = \mathbf{P}_2(\mathbf{T}) = \mathbf{P_2}(\mathbf{P}_{1,inv}(\mathbf{T}_1)) \qquad (3)$$

In practice, the projective transformation which maps the view-plane α_1 (part of polygon β displayed on the user's GUI at time *n-1*) on α_2 (view of polygon β at time *n*) can be defined in terms of a matrix product:

$$\mathbf{P}_3 = \mathbf{P}_2 \cdot \mathbf{P}_1^{-1} \qquad (4)$$

where $\mathbf{P}_i$ denotes the projective matrix associated to the i-th projective transformation. As shown by eq (4), in order to determine $\mathbf{P}_3$, we need to compute $\mathbf{P}_1$ and $\mathbf{P}_2$ before. To this aim, we consider firstly the parametric equations of α and β planes (see figure 5):

$$\alpha \rightarrow \mathbf{x''} = \mathbf{A} \cdot \mathbf{t''} + \mathbf{d} \qquad \beta \rightarrow \mathbf{x'} = \mathbf{A} \cdot \mathbf{t'} + \mathbf{c} \qquad (5)$$

where $\mathbf{A}, \mathbf{B} \in \Re^{3x2}$; $\mathbf{c}, \mathbf{d} \in \Re^3$; t, t'' $\in \Re^2$ and $\mathbf{x}', \mathbf{x}'' \in \Re^3$. Then we add two constrains: a) vectors generating the planes are orthogonal (eq. 6); b) vectors **c** and **d**, defining the distance of α and β planes from the origin of the absolute reference frame, are orthogonal to their corresponding planes (eq. 7).

$$\mathbf{B}^t\mathbf{B} = \mathbf{A}^t\mathbf{A} = \mathbf{I} \qquad (6)$$

$$\mathbf{c}^t\mathbf{B} = \mathbf{d}^t\mathbf{A} = \mathbf{0} \qquad (7)$$

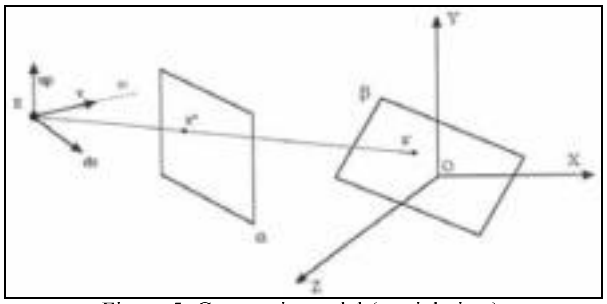

Figure 5: Geometric model (partial view)

Now let us suppose to know **B** and **c** (i.e. the β plane), the position in the 3D space of the eye-point **E**, the user's viewing direction **v**, the unit vector **up** of the eye-point and m, the distance from the eye-point to α plane. Given such parameters, we can compute the α plane, i.e. parameters **A** and **d** of eq. 5, according to the constraints reported in eq 6,7. These constraints mean that the two vectors generating the α plane should be orthonormal and coincident with the columns of matrix **A**, and vector **d** is orthogonal to them. Assuming the view-plane (α) is orthogonal to unit vector **v**, it can be easily demonstrated that matrix **A** becomes

$$\mathbf{A} = [\mathbf{up} \quad | \quad \mathbf{dx}] = \begin{bmatrix} up_x & dx_x \\ up_y & dx_y \\ up_z & dx_z \end{bmatrix} \qquad (8)$$

where **dx** = **up** ∧ **v**. Similarly, vector **d** can be easily obtained according to following relationship:

$$\mathbf{d} = (<\mathbf{E},\mathbf{v}> - m)\cdot\mathbf{v} \qquad (9)$$

where the symbol <,> denotes the scalar product.
At this step we have all elements needed to determine the projective transformation P, which maps a generic point **x**' of β plane on point **x**", lying on the α plane. Let be *s* the bundle of straight lines passing through the eye-point E,

$$s \rightarrow \mathbf{s} = \mathbf{k}t + \mathbf{E} \;;\; \forall \mathbf{k} \in R^3,\, t \in R \qquad (10)$$

With some straightforward algebra, we get the equation of the straight line of *s*, which passes through point **x**':

$$\mathbf{r} = \mathbf{k}t + \mathbf{E} \;;\quad with\;\; \mathbf{k} = \mathbf{x}' - \mathbf{E} \qquad (11)$$

Thus, in order to compute the value t* assumed by parameter t, when the straight line r intersects the α plane on point **x**", we consider following equality:

$$\mathbf{k}t^* + \mathbf{E} = \mathbf{x}'' = \mathbf{A}t'' + \mathbf{d} \qquad (12)$$

After some substitutions and taking into account eq. 7, we get

$$t^* = \frac{\mathbf{d}^t(\mathbf{d}-\mathbf{E})}{\mathbf{d}^t\mathbf{B}\mathbf{t}' + (\mathbf{d}^t\mathbf{c} - \mathbf{d}^t\mathbf{E})} \qquad (13)$$

Using eq. 10 and 13, the following fundamental relationship is obtained, which represents the projective transformation we searched for:

$$\mathbf{t}'' = \frac{\mathbf{F}\cdot\mathbf{t}' + \mathbf{g}}{\mathbf{h}^t\cdot\mathbf{t}' + \mathbf{q}} \qquad (14)$$

Indeed, the 2D vector **t**' contains the coordinates of a generic point **x**' in a β plane fixed reference frame, while the vector **t**", obtained from **t**' by eq. 14 , contains the 2D coordinates of point **x**" on a α plane fixed reference frame.

5. TEST AND RESULTS

In order to evaluate the performance of the compression algorithm, a server/client data transmission with the developed split-browser has been carried out. To this end we employed simple geometric shapes (polygons) of different colors, as shown in figure 6. Here view A represents the scene observed by the user at time n-1, B is the exact view as computed by the server, to be displayed at time n, according to user input, while the predicted view is the one obtained by application of the projective transformation to view A on the client.

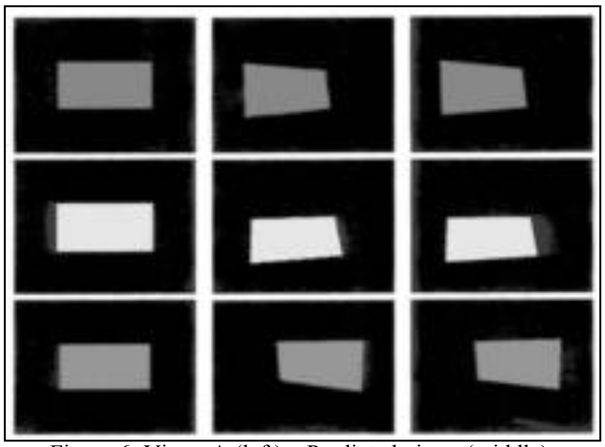
Figure 6: Views A (left) – Predicted views (middle) Views B (right)

Following criterions were defined to compare the results each other:

1) Average bit/pixel number of compressed view B. This parameter defines the number used on average to code each pixel ov compressed view B. As we dealt with 24 bit RGB images, the resulting value should be lower than 24 to pass the test.
2) Average bit/pixel number of compressed error-image. It is similar to the previous parameter, but in this case the threshold value for the pass/fail test should be lower than the previous one.
3) Compression ratio. It defines the ratio between the size (in bytes) of compressed view B and the sum of the size

of compressed error-image and the bytes needed to code the projective transformation parameters.

Table 1: Results of the frame compression algorithm

Polygon	Compr.sed Error-image (bytes)	Compr.sed View B (bytes)	Bit/ pixel error	Bit/ pixel view B	Ratio
Green	625	780	0.07	0.08	1.25
Red	603	768	0.06	0.08	1.27
Yellow	586	753	0.058	0.079	1.28

It should be noted that the compression ratio is always greater than 1, what confirms the effectiveness of implemented compression procedure. Indeed, this means that sending to the client both the compressed error-image and the parameters of the projective transformation, generates a throughput lower than the transmission of the compressed view only. As shown in table 1, the gain is ranging between 25% and 28% in terms of bytes employed.

6. CONCLUSIONS

In this paper an innovative concept of VRML browser has been proposed. It is aimed mainly to allow the display of complex and large 3D models, as the ones obtained by ground-based laser scanners, reducing the computational load on the user-side. To this end, we developed a VRML split-browser, which is conceptually based on a server/client paradigm, but at the same time it keeps all the advantages of the VRML interchange file format as suitable mean to share 3D objects along the web. The main idea on the ground of developed architecture is the splitting of the tasks between server and client. The former performs all the operations requiring the most computational effort, reducing the working load of the latter. This has been accomplished using perspective transformations and image compression algorithms, like LZ77. In our system the server sends to the client only a set o parameters, defining the perspective transformation between to 3D scenes, and a compressed error-image, instead of the whole VRML 3D model. Then the client uses such elements to reconstruct the view of the scene according to the input command activated by the user on the client's GUI.
The proposed work is still in progress, as improvements can be still applied, in order to further optimize the performance of our split-browser. We are investigating the use of the Bounding Box, view-images caching and the adoption of a linear prediction scheme in the LZ77 algorithm. The former would allow to compress only the portion of the whole 3D model that is actually enclosed by the bounding box, saving in this way the number of bytes used for the image coding. The second solution deals with the creation of a memory buffer on the client side, where all the views sent by the srever are sequentially stored. In this way, if the eye-point (user's viewing direction) returns back on a previous visited position, the client needs to restore the corresponding view from the buffer: no data have to be sent from server to the client. A further server/client throughput reduction could be achieved by means of the linear prediction: the pixel color of an image can be predicted on the baiss of the color of the neighbours. Accordingly, the server would not compress the original image but rather the so called residual-image, whose pixels are obtained as difference of the color values of adjacent pixels ont he source image.

ACKNOWLEDGEMENTS

This work was developed with the project "Application in the survey, store and management of environmental and cultural resources of GNSS/INS positioning and satellite, aerial, terrestrial photographic and laser scanning data, transmitted by DARC, GSM/GLOBAL STAR, INTERNET methods", partly financed by MURST (Italian Ministry of University and Research) in 2002 as project of relevant National interest. National coordinator: Giorgio Manzoni, head of the Research unit Antonio Vettore.

REFERENCES

Carey R., Bell G, 1997. The Annotated VRML 97 Reference

Foley J., Dam A. V., Feiner S., Hughes J., 1996. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2nd edition.

Gleicher M., Projective Registration with Difference Decomposition. URL: http//www.cs.wisc.edu/Graphics/Papers/Gleicher/California/track-Final.pdf

Hartley R., Zissermann A., 1999. Multiple View Geometry. *CVPR*, June.

Mason W, 1997. *Open-GL Programming Guide*. Addison-Wesley, 2nd edition.

Mian G.A., Rinaldo R., 2000. *Elaborazione e Trasmissione delle Immagini*. Ed. Libreria Progetto, Padova.

Nelson M., 1992. *The Data Compression Book*. M&T Books.

Ziv J., Lempel A., 1977. A Universal Algorithm for Sequential Data Compression. *IEEE Transactions on Information Theory*, n. 23, pp. 337-343.