EXTENSION OF VALIDITY CALCULATION TO MOVING OBJECTS WITHIN A VIRTUAL REALITY SYSTEM USING FRAME TO FRAME COHERENCE

Franz Duckstein
Computer Science Department – Julius Maximilian University
97070 Würzburg, Germany
duckstei@informatik.uni-wuerzburg.de

ABSTRACT

Generating pictures of a complex scene in real-time with constant frame rates still overwhelms modern rendering systems. The exploitation of frame to frame coherence proposes a solution to this problem by reusing the images of rendered objects multiple times. Combining this with a cost-metric and a LOD-system delivers a uniform frame rate system with less additional effort. The fundament of reusing images is a validity calculation, deciding in how far the original and the image might differ. The application is restricted to fixed objects respectively fixed spaces. This paper presents extensions for existing validity calculation to handle moving objects, which e.g. are now standard in the virtual reality description language VRML 2.0. The additional deviation resulting from object movement is estimated together with viewpoint movement.

CR Categories and Subject Descriptors: I.3[Computer Graphics]: I.3.3[Picture/Image Generation] *image based rendering*, I.3.5[Computational Geometry and Object Modelling] *frame coherence, animated objects*, I.3.7[Three-Dimensional Graphics and Realism] *virtual reality, LOD management*

1 INTRODUCTION

The realism delivered by a virtual reality system is tightly coupled to the scene complexity used. A fine tessellation and appropriate texture resolution of close objects can produce astonishing results. But a high level of detail for all objects within a scene reduces the possible refresh rate of the employed algorithm and hardware. This is contradictory to virtual reality. Also sophisticated graphic systems[1] can't display a 20M polygon scene in real-time by brute force algorithms. Several conceptual different optimization possibilities exist. They all profit from the temporal and spatial coherence in VR systems. Visibility culling [Green93] or explicit LOD-management [Funkh93] are interesting methods, but not subject to this paper.

The use of frame to frame coherence for objects or spatial parts of a scene proposes a way to reduce the calculation effort. Based on the fact, that the appearance of a distant object isn't much affected by a slight change of the viewpoint, they introduce the use of images, realized by a textured quadrilateral, instead of redrawing the geometric primitives every frame. The texture is generated by rendering the object once. The quadrilateral fixes the image position.

In general two basic methods exist: **1)** Pictures for distinct objects [Schau96a]. This method bases on a scene definition by objects. A cost-benefit calculation decides, if or not to generate and use an image for an object. It is easy to implement and guarantees a high benefit for distant complex objects. But it is not applicable to unstructured scenes and does not support spatial hierarchies. **2)** Pictures for spatial areas [Shade96],[Schau96b]. The application is also controlled by a cost-benefit calculation. There is no clustering of polygons by objects needed. Through the support of hierarchical spatial structures like a BSP-tree, its a good method to handle large static scenes, e.g. landscapes with wood areas.

[1]e.g. SGI Reality Monster, 80M Polygons/sec

Common to both methods is the use of a *validity calculation*, which is necessary to determine, if the existing image can be reused after changing the current viewpoint. They calculate the possible difference in terms of an *error angle* between actual rendered geometry and the image. The viewpoint translation and rotation are the parameters of the validity calculation. Our paper presents an extension of this calculation for moving objects: the translation and rotation of objects and viewpoint is included. This extension to dynamic scenes has been tested by concrete examples.

2 PREVIOUS WORK

With *Priority Rendering* [Regan94] the first concept of reusing calculated geometry projection several times, instead of calculating the projection every frame, emerged. This method is based on several frame buffers, containing the geometric projections. Each frame buffer has a different *update rate*. The assignment of objects to the appropriate frame buffer is calculated by a predictive validity calculation.

In image based rendering systems the problem is similar. The used error calculation decides, either to use an existing image or to generate a new image by rendering geometric data new. Fig. 1 shows the problem situation.

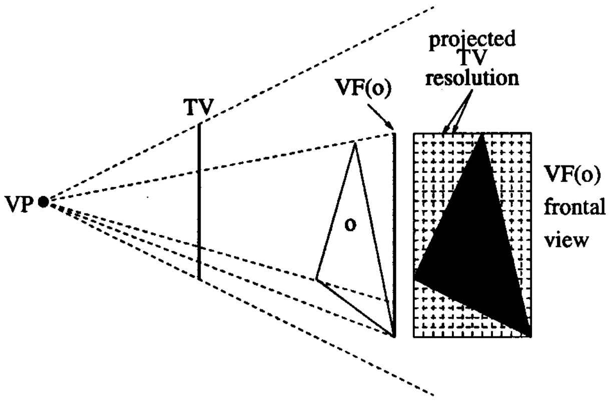

Figure 1: $VF(o, VP, TV_{res})$

VP marks the current viewpoint, TV is the screen area with the resolution TV_{res}, o is the object to display and $VF(o)$ is the image of object o. *Virtual Frame* VF is a synonym to image or imposter [Schau96a]. The calculation of $VF(o)$ must be done with a resolution equal or higher then TV_{res}. The quadrilateral position of $VF(o)$ is perpendicular to VP, the distance to VP can vary between minimal and maximal object distance. The are two different kinds of validity calculation to estimate deviations:

The *Common Validity Calculation* estimates the maximal angle between the existing image and the rendered geometry, caused by the movement of VP to VP'. If the scene consists of distinct objects, each object is surrounded by a bounding box. In spatial structured scenes the subspace limits serve this purpose. Note that the quadrilateral used in Fig. 2 keeps its position. Schaufler [Schau96a] proposes a rotation of $VF(o)$, to keep it perpendicular to VP'. Similar to the previous calculation are calculations, which split the viewpoints move into a translation relative to an object and moving towards an object.

For each *bbox*-edge an error-angle is computed. The resulting maximal error angle is compared to a user defined *error limit*, which can be obtained from the screen resolution TV_{res} and the current field of view FOV. This validity calculations are common usable, because beside VP, VP', $bbox(o)$ and $VF(o)$ no further parameters are necessary.

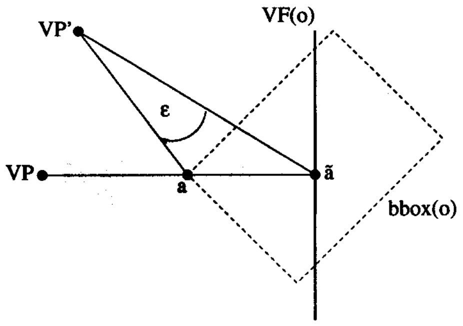

Figure 2: $bbox(o)$ edge a and its error angle ϵ

The maximal ϵ over all *bbox*-edges is compared to the *error limit*. The calculation of ϵ is done as follows. Let $\vec{A} = \overline{VP'a}$, $\vec{B} = \overline{VP'\tilde{a}}$.

$$\epsilon(VP', a, \tilde{a}) = \cos^{-1}\left(\frac{\vec{A} * \vec{B}}{\|\vec{A}\| \cdot \|\vec{B}\|}\right) \qquad (1)$$

The *Predictive Validity Calculation* estimates the maximal number of frames within an image can be used without the error angle exceeding a given threshold. The calculation defines a *safety zone* around VP, which satisfies the error constraints. In order to convert the safety zone into a frame number, the viewpoint translation per frame must be limited by $l_{\triangle VP}$. But this is usually granted in virtual reality systems.

The minimal diameter d of all *bbox*-edges is taken, to calculate the number of valid frames. An advantage of this calculation is, that the prediction supports an easy way to estimate the validity period of super-volumes by the validity period of its contained sub-volumes. So this calculation is well suited for spatial hierarchical structures.

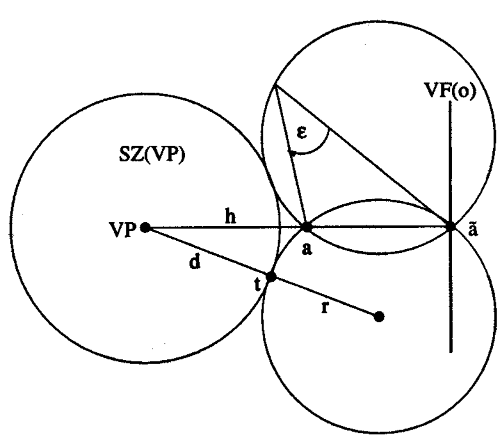

Figure 3: safety zone SZ around VP, defined by a bbox-edge a, its projection $\tilde{a}$ and the error-limit ϵ

$$h = ||VP - a|| \quad \text{and} \quad r = \frac{||a - \tilde{a}||}{2 \sin \epsilon} \tag{2}$$

$$d = \sqrt{h^2 + r^2 + 2hr \sin \epsilon} - r \tag{3}$$

$$t_{valid} = \lfloor \frac{d}{l_{\Delta VP}} \rfloor \tag{4}$$

3 AN ERROR CALCULATION FOR MOVING OBJECTS

First, let us consider, how movement in dynamic scenes is defined. Usually there is a transformation matrix $TM \in \mathcal{R}^{4\times4}$, which is applied to the current object coordinates. TM can execute a rotation and translation by a matrix-vector multiplication. An object o is build up by several primitives (polygons). Each polygon edge is transformed by TM. The application of a transformation TM to an object o can also be expressed by applying a vector addition (translation) for all polygon edges.

Polygon $p = \{p_1, \ldots, p_n\}, \;\; p_i \in \mathcal{R}^4$

$p' = TM \cdot p$ with $p'_i = TM \cdot p_i$

$p'_i = p_i + \Delta p_i$

Note that the Δp_i usually are different to each other. The same transformation can cause different translations to the p_i. Therefore it is useful, to estimate the effects of the transformation pointwise also. At this level we extent existing calculations:

We will show that calculations can handle dynamic objects. It is not subject of this paper, to investigate how a transformation can be integrated into existing scene descriptions. In scenes defined by objects, it is straightforward to apply a transformation to the complete object. Motion within an object, e.g. turning wheels of a car, can only be handled by introducing the wheels as additional objects. Transformation within spatial hierarchical structured scenes is more complicated. Transformations are applied to objects. The subspace of a spatial hierarchical structure might contain several objects with different transformations. This denies the exploitation of hierarchy. The recreation of the spatial structure is also expensive, and cannot be done every frame for greater n. Therefore an hybrid spatial structure, handling static and dynamic parts, will be needed.

Extension of the common validity calculation As a first step we calculate the error angle between the transformed bbox-edges and their former projections. The image keeps its position.

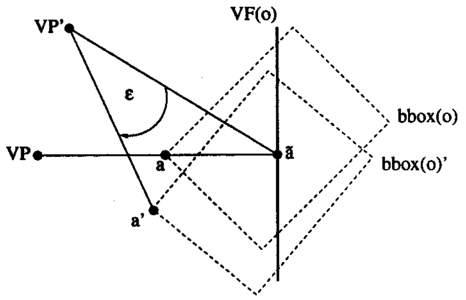

Figure 4: $bbox(o)'$ edge a' and its error angle ϵ

This version suffers from several disadvantages. *a)* if VP and p are transformed to similar direction and distance, the existing image $VF(o)$ could be reused very often. *b)* Shade *et al.* keep the image position fixed, to compensate slight changes in the perspective projection.

We propose to apply the objects transformation also to the quadrilateral containing the image. By this way we can exploit two effects: Motion parallax caused by object transformation and object rotation around the axis between VP and the object.

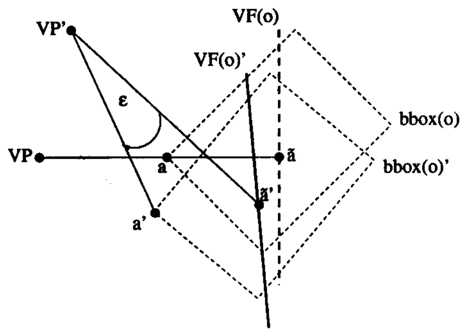

Figure 5: $bbox(o)'$ edge a', the transformed projection point $\tilde{a}'$ and its error angle ϵ

The calculation of ϵ is done as in Eq. 1, substituting $\vec{A} = \overline{VP'a'}$, $\vec{B} = \overline{VP'\tilde{a}'}$.

Extension of the predictive validity calculation Prediction is a more complex problem. Especially if you want to exploit uniform translation of viewpoint and object. The predictive validity calculation needs a limited viewpoint translation per frame ($l_{\Delta VP}$). Now we need in addition a limitation of the object movement. We assume, that the transformation $TM(o)$ applied to object o doesn't change during the next frames. The safety zone $SZ(VP)$ is calculated without change to the static situation by Eq. 4.

In order to integrate objects movement, we keep a and $\tilde{a}$ fixed. Instead we apply the translation ΔVP to VP first, then the inverse translation $TM^{-1}(o)$ of object o. The resulting translation from VP to $relVP'$ is extrapolated. So it indicates how many frames the image is valid. An uniform movement of viewpoint and object o increases the validity. A rotation of object o around the axis from VP to o is now also recognized. Fig. 6 shows the situation.

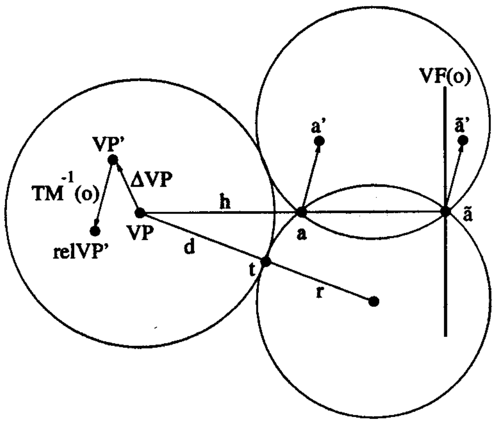

Figure 6: estimation of valid frames by transformation of $(VP + \Delta VP)$ by $TM^{-1}(o)$

$$VP' = VP + \Delta VP \ , \quad \Delta VP \ \text{assumed fix}$$

$$a' = TM(o) \cdot a \ , \quad TM(o) \ \text{assumed fix}$$

$$relVP' = TM(o)^{-1} \cdot (VP + \Delta VP)$$

$$t_{valid} = \left\lfloor \frac{d}{\|TM(o)^{-1} \cdot (VP + \Delta VP) - VP\|} \right\rfloor \quad (5)$$

As in the case without object movement, the minimal resulting t_{valid} is taken.

4 RESULTS

Test Environment In order to prove our calculations, we implemented a C/C++ program [Hearn97] to display animated virtual 3D scenes. The time measurement and object transformation is done frame-wise. As 3D engine we used the OpenGL 1.1 [Woo97] compatible Mesa 2.2 library. Our implementation and test platform was an HP 900 712/80. Within our program no visibility culling, LOD management, constant frame rate management or *cost/benefit* calculation is integrated. These deficiencies can be easily removed, if an effective spatial hierarchical structure for static and dynamic objects exists.

Optimization Restrictions Each object of the scene must provide a surrounding bounding box. As a prerequisite the projection of the *bbox* must lie complete within the screen area. As in [Shade96] the projection delivers the quadrilateral size and position. In the case of the *common validity calculation* a heuristic decides, whether or not to apply a virtual frame. In the case of the *predictive validity calculation* a virtual frame is used, if the prediction value is equal or greater than 2. Objects build up by a few polygons, but covering large screen areas, are excluded from replacement by an image.

Test Scenes The virtual scene is build up as a list of hierarchical objects. Each object has its own transformation, which is updated every frame. We applied the calculations to several small test scenarios. The test sequence length during all tests was 256 frames. The urban-scene consists of two small pieces of paved road with moving trucks. The speed on the first street is limited to 0.5 units per frame, on the second street to 1.0 units per frame. There are two trees and two buildings to complete the scene. Every object of the scene, without the ground and the street, can be replaced by images. The viewpoint moves within these frames constantly forward and slightly up. The view direction is constantly changed downwards. The average viewpoint move distance per frame is 0.9 units. In this test, the *common validity calculation* was used.

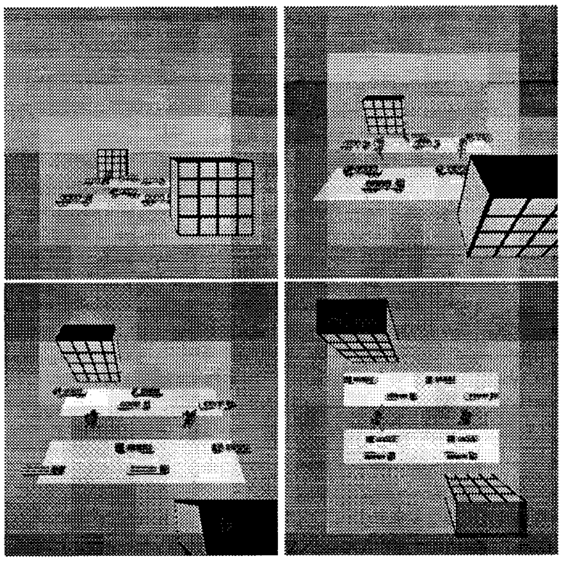

Figure 7: flight over urban-scene, frame 1, 86, 171 and 256

Fig. 8 and Fig. 9 show the *average image validity* and the *average frame calculation time* for the previously described sequence. As an example, it turns out the benefit potential of images applied to dynamic objects.

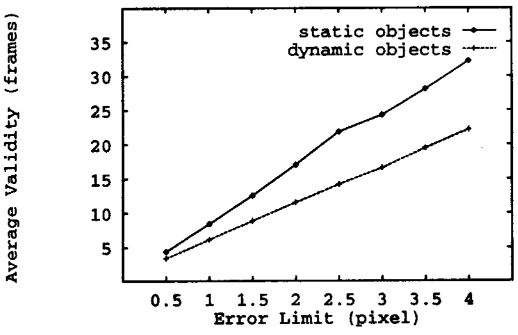

Figure 8: average image validity during flight over urban-scene in 256 frames

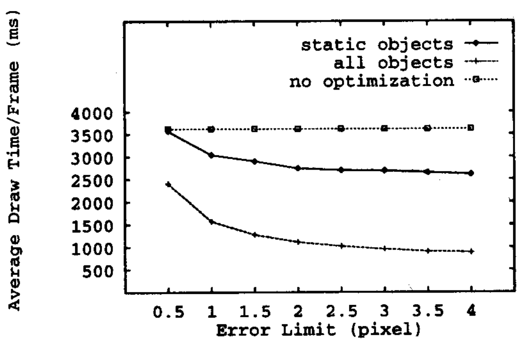

Figure 9: average frame calculation time during flight over urban-scene in 256 frames

The Fig. 10 shows the same urban-scene from a fixed left upper viewpoint, looking to the trees. The used virtual frames are marked by a black border. In the difference picture Fig. 11 we visualize the artefacts of the moving trucks. Despite an *error limit* of 4 pixel, the artefacts are moderate. The building and the trees produce no artefacts, due to the fixed viewpoint.

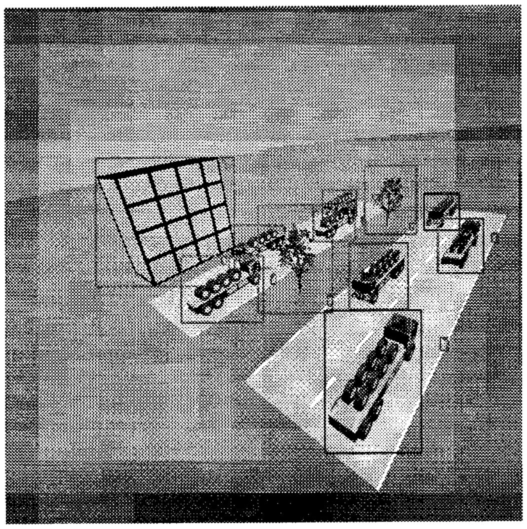

Figure 10: urban-scene from a fixed viewpoint, *virtual frames* visible

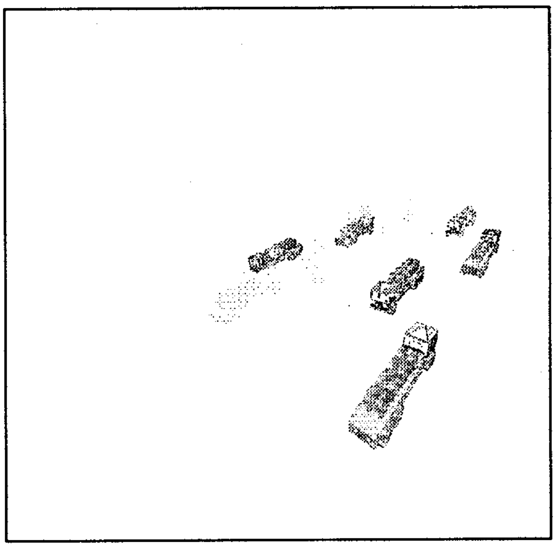

Figure 11: urban-scene from a fixed viewpoint, difference picture

It is interesting to investigate the effect of object translation and rotation towards the validity of their calculated virtual frames. In Fig. 12 we can see the test situations. Both scenes contain no additional objects. The viewpoint was kept in a fixed position, so the object validity depends only on the object movement. The *error limit* for all subsequent tests was set to 4 pixels.

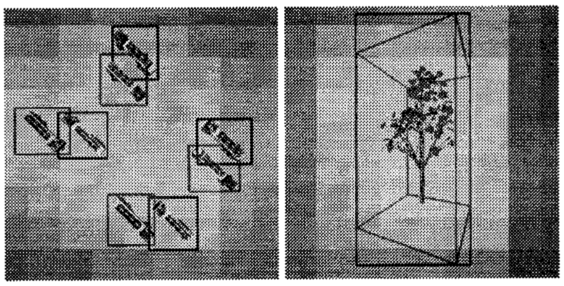

Figure 12: test scenes to evaluate translation speed (left) and rotation speed (right) towards validity

In the translation test all objects (trucks) have the same translation speed. The results are shown in Fig. 13. The displayed curves are similar to the hyperbole of $time = \frac{distance}{speed}$ with an appropiate distance value. In the rotation test, see Fig. 14, the resulting curves show the same correlation of speed and validity. This is based in the piecewise linear interpretation of movement in our calculations. For both tests the *bbox* size and orientation towards the applied transformation influence the result. It was no surprise, that the object validity is more sensitive to rotation than translation. The combination of great object rotation, small distance to the viewer and small error limits reduces the image validity dramatically. By this a *cost/benefit* ratio greater than 1.0 is the result.

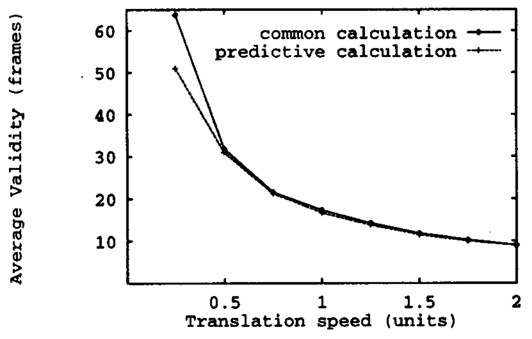

Figure 13: object translation speed / average validity with a fixed viewpoint 135 units away

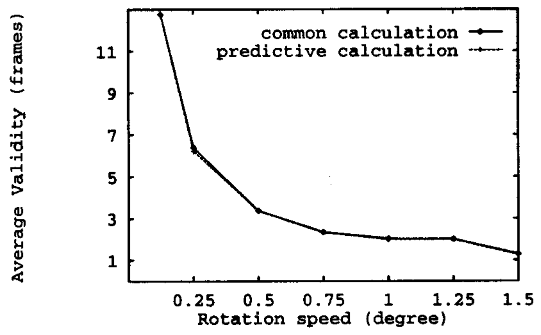

Figure 14: object rotation speed / average validity with a fixed viewpoint 60 units away

The distance between viewer and object is also important to the image validity. In Fig. 15,16 you can see a linear correlation between viewer distance and validity. The test scenes are the same as in the translation and rotation test, but with fixed transformations. Despite the different validity values both curves show comparable gradients.

Within all the previous transformation tests the similarity between the *common validity calculation* and the *predicive validity calculation* is a result of the constant object movement. Objects with a changing dynamic would lead to different validity values.

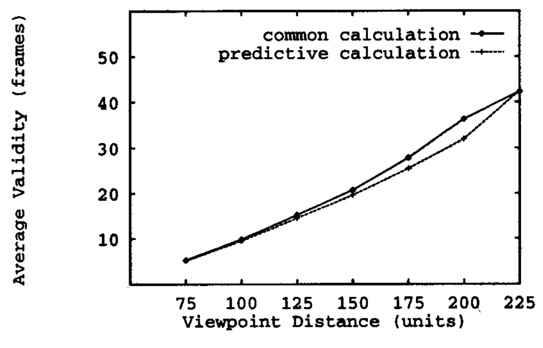

Figure 15: viewpoint distance / average validity for a 1 unit translation per frame

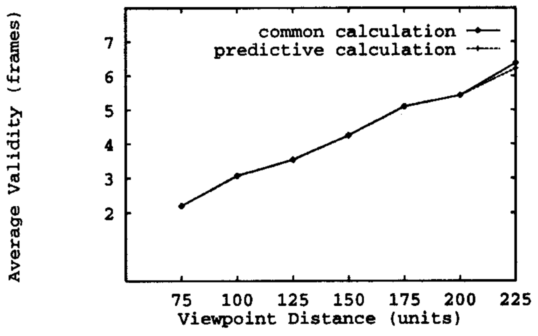

Figure 16: viewpoint distance / average validity for a 1 degree rotation per frame

Texture Filtering In our implementation we used no texture filtering. This leads to distinct artefacts in our test scenes. There is sometimes a one pixel wide shift of the complete image, up or down and left or right. This effect is caused by small rounding errors arising from matrix inversion within our calculations. They are amplified by rounding errors during texture calculation within the graphic library. A *pumping effect* visible within the image itself, is caused by the transformation of the image rectangle. A bi- or trilinear filtering to the image texture would lead to better results. A higher image texture resolution would also reduce the artefacts, but this demands 4 times more texture memory. Both methods are (in software) time expensive. Most of the newer 3D graphic hardware support texture filtering.

Constant Frame Rate The employ of image based rendering reduces the calculation effort per frame. But the effort per frame is staggering. If within one frame calculation, the number of images which must be renewed is small, the frame calculation time is short. The renewal of too much images within one frame can increase the calculation time dramatically, in extreme cases above the time needed without image rendering. In virtual reality a constant frame rate is unalterable for high realism. Image based rendering delivers beside/with LOD a good parameter for time management. The selection of low object detail levels preserves time to calculate new images, the reuse of images preserves time to render detailed objects. The concrete constant frame rate management integration of image based rendering with LOD is well described in [Schau96a]. The algorithm presented there is also appropiate to moving objects. But image based rendering can also be used without LOD management to achieve constant frame rates. This can be done by solving the tradeoff between the number of needed object renderings and the accepted *error-limit* for existing images.

5 CONCLUSIONS AND FUTURE WORK

This paper presented an extension for validity calculations to apply virtual frames to dynamic objects. The frame to frame coherence of translated objects seems to be very profitable. The benefit is smaller with rotating objects. They demand a high *error limit* and a sufficient distance between object and viewpoint. The common use of replacing dynamic objects by images is restricted by several problems:

- *Texturing Artefacts* The quadrilateral transformation and the coordinate calculation by matrix inversion suffer to inherent rounding errors. Combined with simple texturing they lead to notifiable effects.
- *Intersecting Objects* The use of images instead of rendering concrete objects can produce incorrect depth resolution. Critical constellations are intersecting objects and closely neighboured objects. This must be kept under surveillance. The overlapping borders in Fig. 12 (top) displays such situations. A solution to this problem was proposed by [Schau97c].
- *Cost/Benefit Calculation* In our test scenes we decided which objects to replace by images. Sometimes it is less time consuming to render objects with few polygons every frame, then to apply an image. For a general solution this should be decided automatically.
- *Spatial Hierarchical Structure* To achieve speedups of an order of magnitude within complex scenes, its necessary to integrate static and dynamic objects in a spatial hierarchical structure. By the use of superobjects the display algorithm must not handle every single object every frame.
- *Comparison* Our test scenes have been implemented in C/C++. By the implementation of a VRML 2.0 translator we hope to get access to a variety of interesting virtual 3D scenes. A lot of VRML scene are common available within the internet. They can alleviate the problem of getting a useful test scene and a base of comparison.

ACKNOWLEDGMENTS

I would like to thank Reiner Kolla and Winfried Nöth for many useful discussions and contributions to this project.

REFERENCES

[Carey97] Carey,R, Bell,G: *The Annotated VRML 2.0 Reference Manual, Addison Wesley Developers Press*, 1997

[Funkh93] Funkhouser,T, Séquin,C: Adaptive Display Algorithm for Interactive Frame Rates During Visualisation of Complex Virtual Environments, in Computer Graphics Proceedings, *Annual Conference Series*, pp.247-254, 1993

[Green93] Greene,N, Kass,M, Miller,G: Hierarchical Z-Buffer Visibility, in Computer Graphics Proceedings, *Annual Conference Series*, pp.231-238, 1993

[Hearn97] Hearn,D, Baker,M: *Computer Graphics - C Version, Prentice Hall*, 2nd ed., 1997

[Regan94] Regan,M, Pose,R: Priority Rendering with a Virtual Reality Address Recalculation Pipeline, in Computer Graphics Proceedings, *Annual Conference Series*, pp.155-162, 1994

[Schau96a] Schaufler,G: Exploiting Frame to Frame Coherence in a Virtual Reality System, in *Proceedings of VFRAIS '96*, pp.95-102, April 1996

[Schau96b] Schaufler,G, Stürzlinger,W: A Three Dimensional Image Cache for Virtual Reality, in *Proceeding of Eurographics '96*, 1996

[Schau97c] Schaufler,G: Nailboards: A Rendering Primitive for Image Caching in Dynamic Scenes, in *Rendering Techniques '97*, pp.151-162, June 1997

[Shade96] Shade,J, Lischinski,D, Salesin,D, DeRose,T, Snyder,J: Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments, in Computer Graphics Proceedings, *Annual Conference Series*, pp.75-82, 1996

[Woo97] Woo,M, Neider,J, Davis,T: *OpenGL programming Guide, the official Guide to learning OpenGL, Addison Wesley Developers Press*, version 1.1, 2nd ed., 1997