SUPER-REALISTIC RENDERING USING REAL-TIME TWEENING

Holger Winnemöller and Shaun Bangay
Rhodes University
PO Box 94, Grahamstown 6140, South Africa
g97w6248@campus.ru.ac.za / s.bangay@ru.ac.za

ABSTRACT1: - The realism of contemporary computer graphics (and especially Virtual Reality (VR)) is limited by the great computational cost of rendering objects of appropriate complexity with convincing lighting and surface effects. We introduce a framework that allows rendering of objects in true photographic quality using tweening. The simple but effective design of our system allows us not only to perform the necessary operations in real-time on standard hardware, but also achieve other effects like morphing. Furthermore, it is shown how our system can be gainfully employed in non-VR contexts like extreme low-bandwidth video-conferencing and others.

KEYWORDS: - Morphing; Tweening; Real-time; Super-Realism; Virtual Reality; Control-Vertex selection and identification; Facial Expression Reconstruction.


1. Introduction
The field of Virtual Reality (VR) tries to create ever more immersive worlds with impressive graphics, three-dimensional positional sound and even tactile feedback. Due to the computational overhead and complexity of scenes the graphical output in real-time of most VR systems is far from realistic, but with faster and more advanced 3D graphics accelerators appearing on the market all the time, we expect ever increasing realism to parallel this hardware development.

We foresee a future in which photorealistic stereoscopic images will truly convince a VR user that they are part of a virtual world. To study the effect this degree of visual immersion would have on a user, we want to create super-realistic interactive objects in our VR environment and investigate the user’s reactions to these. This paper describes the approach we take to implement our super-realistic render.

For the purpose of our discussion, we define super-realistic as beyond the scope of normal real-time renderers, in the sense that we require:

- Very detailed objects in both geometry as well as surface properties (e.g. hair, fur, textured and smooth surfaces, etc.)
- TV-like Image Quality
- Real-Time interaction with these objects in a VR environment

We acknowledge the fact that these requirements cannot be fulfilled in a straightforward manner on today’s hardware and graphics APIs, but argue that given certain compromises we can achieve a very close approximation. In our case this compromise is the memory requirement. With current memory sizes available and appropriate compression techniques, we can overcome memory limitations easier than processing speed limitations.

1.1. Definitions
In the context of this paper we will use the following definitions:

- **Tweening** – process of deriving intermediate frames in order to convert an object or scene in one key-frame into a similar object or scene in an adjacent key-frame.
- **Morphing** – process of deriving intermediate frames in order to convert an object or scene in one key-frame into another object or scene in an adjacent key-frame.

2. Previous Work
The limitations mentioned above have also been acknowledged by many other researchers and spawned a whole new field of graphical research: Image-based rendering. In a broad sense this means that instead of re-creating scenes from scratch (purely based on geometrical data and material properties), images (either photographs or pre-rendered) are used in order to augment or completely represent the rendered scene. Some depth information is usually required in order to distort input pixels in perspective-correct manner. The degree in which geometrical structures are used and of what complexity these are depends largely on the application.

---
1 This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom, Lucent Technologies, Dimension Data, and THRIP.
One very popular example of such a technique is Quicktime VR by Apple [5], [6]. A 360-degree panoramic image of a natural scene (various examples such as a view inside a spacecraft or a museum exist) is either taken with a special type of camera or computed by distorting a series of conventional photographs. To view such an image, the viewer is placed inside some convenient geometry (usually spherical or cuboid), onto which the images are mapped. By rotating the geometry and moving inside it (to a limited extent) the viewer achieves a certain degree of spatial awareness of the scene rendered. This technique lends itself well to render objects such as rooms or surroundings, where the viewer can be assumed to be situated near the centre of geometry and is guaranteed not to come too close to the boundaries of the object (this will result in undesirable distortions). Recently, a feature similar to the technique described in this paper was added to the Quicktime VR capabilities: Using a (limited) set of images of a given object taken from different angles, the viewer can move the object around in real-time (a demo can be found at [7]). Our approach continues, where Quicktime’s abilities stop: the smooth interpolation of these basic images.

Schödl et al [2] developed another approach that is quite similar to ours and which can be applied to render objects other than rooms and surroundings. They use constructs, which they call Video Textures, to produce a continuous, infinitely varying stream of images. Their main contribution is the concatenation of video sequences of various moving objects in such a manner that is undetectable to the human eye. They achieve this by analysing video footage and detecting key-frames where transitions to other frames are possible (this considers a spatial context, i.e. the content of a given frame, as well as a temporal context, i.e. the movement of objects, which has to remain coherent). While their approach can well be used to simulate objects which behave interactively, their rendering is limited to the video-sequences captured and does not envisage a general 3D usage.

Kim et al [3] use Multi-view video data to composite with geometrically rendered scene elements to increase the immersive quality of their virtual environments. A video avatar representation of a user is one of their application examples, but whole backdrops and other scene elements are also incorporated into their virtual environments. The environment issues are their main focus.

Debevec et al [8] use a hybrid-approach (image- and geometry based) to render architectural scenes. One of their contributions is a process called photogrammetric modelling, which allows the recovery of basic object shape from a very limited set of input pictures. Assumptions about typical architectural constraints are a key element in their successful implementation.

Some brute force approaches also exist, which do not resort to image-based rendering. As the standard photorealistic rendering techniques such as ray tracing or radiosity are extremely computationally expensive, these approaches usually focus on the technical issues of speeding up the rendering process to interactive rates. One such approach is described by Stuttard et al [1], where various techniques such as discontinuity meshing, adaptive subdivision of meshes and ray casting for form-factor calculations are used to optimise the rendering process. Nonetheless the system has to be implemented on a parallelised transputer network in order to return with a solution to the radiosity equation in reasonable time.

Very interesting work on morphing has been published by Seitz and Dyer [9], who focus on the view-dependency issues of morphing (i.e. perspective correction). They classify their contribution as an extension to classic morphing and use a set of pre-render distortion techniques to accommodate for different views onto objects/scenes. Their main focus is on correct camera simulation as they differentiate between image morphing and view-morphing and present a set of tools for the latter.

3. Spherical Sampling with rotational distortion information

3.1. A different way of storing 3D objects

Almost all contemporary graphics accelerators are based on a 3D geometry engine that can transform 3D vertices into screen-space and in the process apply certain texture and lighting effects to surfaces of objects. In most practical situations, the computational cost of rendering an object and its visual realism are proportional to the number of vertices that describe an object and the quality of textures used. A compromise is made in having a fairly low geometric detail on which fairly high quality textures are superimposed. The result, while providing a certain degree of realism in games and VR, is far from photo-realistic. Achieving a realistically rendered object with traditional methods would necessitate a geometric level of detail (LOD) that would thwart any real-time constraints. For our super-realistic renderer we therefore suggest a different way of storing object data, which still facilitates the use of contemporary graphics accelerators.
Assuming we had a photograph of an object from every conceivable angle, we could scan these pictures and by choosing the appropriate one for a given viewpoint could rotate, scale and translate it so that we have a super-realistic representation of an object (this implies lowest LOD geometrically and highest with respect to textures). The necessary transformations can easily be performed with 3D graphics cards. The pictures then merely need to be texture-mapped onto a quadrilateral, also easily implemented in hardware. The most obvious problem with this is the theoretical amount of texture-memory required.

3.2. Importance of Rotation
We will focus our discussion of super-realistic rendering on the realistic rotation of objects. This is because scaling and translation of an object can be achieved by scaling and translating a given image of that object if it’s orientation relative to the viewer does not change (in most practical situations though a translation of an object will bring about a rotation relative to the viewer). A rotation of the object on the other hand will hide or reveal geometric detail from the viewer and it is this case, which has to be attended to in order to convincingly render image-based objects.

3.3. Discrete Spherical Sampling
To overcome the texture-memory limitation we decide to sample an object spherically with evenly distributed sampling (view-) points (this is illustrated in Figure 1). This means that we now do not have samples for every conceivable view-point available and two situations can arise: The viewer looks onto the object from a view-point for which there is a sample, in which case this will be used. Otherwise, (and this is more likely) the viewer will be slightly off a given sample point and we have to deal with this situation specifically. A naïve solution would be to use the nearest (in terms of distance on the sphere) sample point, but this will result in images jumping and not be very convincing. For many samples this may not be very noticeable, but our intention is to lower the spherical sampling frequency as much as possible, while retaining the highest possible visual quality. We therefore have to interpolate the different viewpoints.

3.4. View-point Tweening
A first attempt to rotate the quadrilateral to compensate for rotational motion of the object proved fruitless due to the fact that the quadrilateral is flat and its perspective distortions do not correlate well to those of most 3D objects.
pixels in the images. Firstly, this would be far too costly, and secondly, various pixels are present in one image and not in the other. The solution therefore is to specify pixels at key positions (roughly following object geometry) and interpolate the other pixels. One readily available method to do this is texture-mapping. By specifying texture co-ordinates for every geometric vertex, we can pinpoint pixel locations and let the texture mapping routine interpolate all pixels in between. To obtain geometry vertices, we simply connect the control-points in such a way, that our whole image is triangulated. This is shown in Figure 2d. It should be noted that this geometry is strictly two-dimensional and does not correspond to the three-dimensional Teapot geometry (even though it is related to the perspective projection of it).

By linearly interpolating the control-points along the paths shown in Figure 2c we can deform one image to look like another. If we do this from two directions (transform the start-image towards the end image and vice versa) and blend the two outputs relative to the position on the interpolation path (100% of the start image and 0% of the end image to begin with, then slowly decreasing the start image percentage and increasing the end image percentage until the relation is reversed) we obtain a visually excellent result.

Our samples demonstrate only one plane of the geospherical sampling, thus justifying the bi-linear interpolation. In the three-dimensional case, we have three close neighbours to consider and thus use trilinear interpolation.

While the result of this approach is both convincing and extremely efficient, there are situations in which certain artefacts may appear. These artefacts are:

- Triangles which move into each other, producing overexposed areas (due to the blending – see Figure 3a&b)
- The linear interpolation produced discontinuities across interpolation couples (when one set of interpolation data is exchanged for the next. (A typical such set is shown in Figure 2c)

![Figure 3 - Triangulation issues: a) Initial triangulation; b) overlapping of triangles after marked vertex has moved into neighbouring triangle; c) possible triangulations of same vertices](image)

To deal with the first problem, we employ the z-buffer to ensure that each pixel is only ever written to once during each render-pass. An alternative solution would be to re-triangulate the vertex-data on each frame, which also avoids overlapping triangles, but unfortunately can result in triangle-flipping, another unwanted artefact (see Figure 3c). As the vertices comprising the triangles move over the image plane, any given triangulation algorithm will connect neighbouring vertices according to their relative distances. This means that vertex connectivity can change spontaneously from one frame to the next, which in turn affects texturing of the triangles. While our z-buffer method requires several (relatively costly) buffer clear operations, these are commonly implemented in hardware and we are confident that the improved visual quality justifies this expense.

The second problem was overcome by using cubic splines, along which the control-vertices can move in a more natural and non-linear fashion.

### 3.5. Acquisition of Control-Vertices

In order to be able to move control-vertices from a starting-point along a spline to an end-position, first suitable control-points have to be established. We perform this in several ways.

#### 3.5.1. Manual Selection

For testing purposes and adjustment of our main algorithms, we select control-vertices manually. While this proves to be extremely reliable, we quickly came to an understanding that this task had to be automated. Nonetheless we could identify from our own experience the steps necessary to gather control-vertices. They are hereby listed:

1. Identify suitable control-Vertices (these include edges, contours and other recognisable landmarks).
2. Find these control-Vertices in adjacent images.
3.5.2. Automatic Selection

To implement the steps identified above, we define the following set of rules that we follow in order to identify viable control-points. For this discussion we define a cluster of Image-pixels as a landmark (in our case a rectangular area of pixels, the location of which is defined by its geometric mid-point). The rules are listed in descending precedence so that a short-circuit evaluation can be performed:

1. Landmarks may not lie too close to image boundaries (discontinuities)
2. Landmarks may not lie too close to one another (redundancy)
3. Landmarks must lie on region of high frequency (it is implied that most of the landmarks are enclosed by edges)
4. Normalised Cross Correlation (NCC in short, see[4], Section 3.2 for details) value of surrounding Landmarks varies significantly from Landmark under inspection (this makes a correct match in the adjacent Image more likely)
5. Landmarks can be found in the original and neighbouring image (self-verification)

In our tests we found that while the NCC works extremely well to find corresponding landmarks in neighbouring images, the identification of landmarks still leaves room for improvement. For example it is often difficult to identify landmarks along silhouette-edges, which are important to retain the shape of the object. Another problem area is that of appearing and disappearing features of an object. These need to be carved out of adjoining parts of the object, but by definition have no corresponding parts in neighbouring Images. These issues are part of an ongoing investigation into the matter.

3.5.3. Inherent Geometric Selection

Another possibility of selecting control-points, though not always applicable, can be applied if the geometric shape of an object is known. The teapot in Figure 2 for example has been rendered by a ray-tracer using a 3D-object file. If this kind of information is known, we can easily deduce the necessary control-points to perform the tweening. It is obvious that this is almost never the case, when taking pictures of live elements like faces, people etc. Nonetheless crude 3D replicas could be constructed from which the control-points can then be derived.

4. Performance Issues

Since real-time considerations are a main design-aspect, we pause briefly and investigate the performance issues involving our approach.

Firstly, image acquisition, image evaluation as well as definition of control-vertices are non-trivial tasks and computationally expensive, but can all be performed off-line, before the actual rendering. At run-time the following tasks have to be performed:

- Translation of image position and rotation to a corresponding set of neighbouring Images.
- Depending on the quality of output needed, linear, bilinear or trilinear interpolation of neighbours; the speed of which mainly depends on the hardware-capabilities of the system.

Depending on the way the images are stored, the translation step is largely trivial. In most of our examples, we are dealing with about 30-80 triangles (a fairly low number considering the processing power of most modern 3D accelerators), which have to be rendered twice for bilinear interpolation. Even for three render passes, this can easily be performed in real-time.

One important consideration remains, which is the one of texture memory. Even though we can drastically reduce the amount needed for our renderer by using an interpolation scheme, we may still fill up texture memory rather quickly. This in turn may lead to thrashing if the amount of memory is too small. Possible solutions are:

- Using hardware texture compression offered by many modern graphics cards
- Taking advantage of object symmetry
- Decreasing degrees of freedom of object or observer (e.g. if a VR user is limited to wander the surface of a flat world, the possible elevation range towards the object is limited as well)
- Increasing interpolation span (which will in most cases result in loss of animation quality)

It should be considered that the rendering of objects of similar quality in a standard fashion (e.g. ray tracing), would also require a large amount of data (geometry, textures, materials, bump maps, noise maps, etc.) and could not be performed in real-time.

5. Demos and Applications

Though we designed our super-realistic renderer to be integrated alongside several other renderers in a VR context, we envision many other potential application areas.
5.1. Extreme Low Bandwidth Video Conferencing

Panagou et al [4] amongst many other researchers acknowledge the need for bandwidth conservation especially for such high-bandwidth applications such as video-conferencing. They argue processing power is a more readily available and expandable resource than networking bandwidth. With this in mind they develop a virtual reality video-conferencing system that can track facial expressions (such as frowning, laughing, etc.) and sends these expressions over the network instead of the video data as such. A geometric model is then deformed using free-form deformation and a texture-mapped to reconstruct images on the remote side. Several issues reside with this approach: The geometric model in many cases is quite complex and will have to be transmitted over the network. The free-form deformation is also expensive.

We developed a similar system, which bypasses the 3D difficulties of the above approach. As most video-conferencing is two-dimensional in nature (the above pertains specifically to video-conferencing in a virtual environment), we allow ourselves to remain totally in the 2D domain. Using an expression evaluation system like the one mentioned above, we can identify and track key-positions on input images. As above, we simply send emotion data to the remote side, but there is no need for a 3D model of an input object or free-form deformation. Since our approach is totally image based, we can reconstruct expressions with subtle morphs of the input image. To allow for a wider range of expressions and grimaces, additional input images fragments can be sent over the network, when the network load allows this (similar to progressive GIF and JPEG loading).

Bregler et al [10] demonstrated a system, which uses soundtrack analysis to identify phonemes. These in turn are used in a lookup procedure to select from a variety of mouth positions, which are stitched into video material. With their technique they can reproduce video to match a given voice sequence, thus literally putting words into peoples mouths. A similar identification and encoding technique can be gainfully employed in our case in order to not only reproduce emotions and facial expressions, but also mouth movements.

Appendix B shows a schematic of this system: A camera captures the portrait of a person. Using Image analysis, the positions of eyebrows, lower eyelids, nose, mouth and chin are determined and used for the selection of a corresponding facial expression (which include grimaces like laughing and frowning, but also mouth-positions when speaking certain phonemes). Panagou notes that only very few facial features have to be tracked to correctly identify a certain expression. A code for the facial expression is then transmitted to the remote side for reconstruction.

The remote side is initialised with an image of a neutral facial expression, which is morphed according to expression templates. Several points should be noted with respect to this application:

- Reconstructed Images cannot correspond 100% with original Images (reconstruction of sad face is much better than that of happy face, because laugh-lines and wrinkles cannot be formed from neutral image)
- Reconstruction is very easy and fast with our method
- Expression evaluation may be relative expensive depending on the complexity of the Image Analysis

5.2. Motion Picture Tweening & Morphing

Many modern Motion Pictures use Computer Special Effects to draw audiences and make the seemingly impossible a reality (at least on Celluloid). Movies where morphing and/or tweening have been extensively used include Terminator II (morphing) and more recently The Matrix (for information on an effect called Bullet-time, which uses tweening, see [11]). We have shown how both Morphing (Appendix A – “Two Faces) and Tweening (Appendix A – “Furry Teddybear”) can be achieved with one and the same method – easily and in real-time. If the texture-size and image quality of most 3D accelerator cards cannot compete with Motion Picture standards, we certainly believe that our system can aide in previewing and scene visualisation.

5.3. Facial Character Animation

Another application comes to mind when considering the difficulties one encounters when animating virtual characters. In many cases some form of motion-capture (MC) is used to give characters a more lively appearance. MC can be performed amongst other methods using radio-trackers or video-evaluation but is in most cases expensive and time-consuming. This is especially true for facial expressions since, if the expression is formed on a purely geometrical level, many vertices have to undergo transformation and therefore have to be motion-captured. A good example of one facial animation system that uses emotion databases and very complex geometry is [12].

We on the other hand believe that if the facial expression is performed at texture level (as with the Video Conferencing above) we can map an entire expression on a much less detailed and much less animated geometry. By segmenting a facial image into regions of importance (Moore [12] states that 80% of emotions are expressed solely by the eyebrow and mouth regions), we can control them individually and cut down on the total number of entries in our database.
6. Conclusion (Future Work)

We have shown in this paper how we can get a glimpse of what may lie in store for future generations of graphics users with respect to visual reality. We developed a system that can produce interactive super-realistic images and tackled most of the problems associated with our method. The feature that sets our method apart from similar ones like Quicktime VR, is the ability to perform smooth interpolation of sample images in real-time. By using existing and readily available graphics hardware instead of software image manipulation routines, we devise a simple but effective method to implement morphing and tweening with minimal processing requirements. Due to the inexpensive nature of our system, a large field of application areas opens itself up, some of which we explored successfully.

Future work will improve on the automatic identification of suitable control-point.

7. Acknowledgements

The authors would like to thank the Distributed Multimedia Center of Excellence and Rhodes University for its ongoing support and sponsorship of this project.

8. References


Appendix A – Interpolation Samples

Tweening - “Furry Teddybear” [0° - 40°]

This sequence illustrates the use of our technique in animating a rotating object. Key-frames are marked with dots (0, 20 and 40 degrees); other frames are tweened. The resolution of the tweening is arbitrary and the number of interpolated views can be as high or low as desired.

Morphing - “Two Faces” [full]

Interpolation between different content key-frames (i.e. morphing) was achieved by moving 70 control-points and thus deforming the key-frames towards each other. The transformed extremes (input frame transformed to output frame and vice versa) are shown below.
Appendix B – Extreme Low Bandwidth Video Conferencing Demo
SUPER-REALISTIC RENDERING USING REAL-TIME TWEENING

Holger Winnemöller and Shaun Bangay
Rhodes University
PO Box 94, Grahamstown 6140. South Africa
e-mail: g97w6248@campus.ru.ac.za or s.bangay@ru.ac.za

NAME: WINNEMÖLLER, Holger
PLACE AND DATE OF BIRTH: Hamburg, Germany, 24 December 1973
NATIONALITY: German

EDUCATION:
• 1981-1985 Grundschule Rahlstedt, Hamburg, Germany
• 1985-1990 Gymnasium Rahlstedt, Hamburg, Germany
• 1990,1991 Alta High School, Salt Lake City, Utah, U.S.A.
• 1991-1993 Gymnasium Rahlsted, Hamburg, Germany (completed with the Abitur)
• 1993-1994 Civil Service in Germany for the Deutsche Muskelschwund Hilfe e.V.
• 1994, 1995 Linguistics and Language, History & Culture of America at Hamburg University
• 1995-1996 Physics and Computer Science at Hamburg University
• 1997-1998 Physics and Computer Science at Rhodes University (completing the B.Sc. with Distinction in both subjects)
• 1998-2000 B.Sc. (Hons.) in Computer Science at Rhodes University (completed with Distinction).
• Present: Masters of Science in Computer Science

OTHER EXPERIENCES:
• 1990 Practical at the “Deutsche Schiffbau-Versuchsanstalt” (German Experimental Ship-building Institute)
• 1993-1994 Civil Service in Germany for the Deutsche Muskelschwund Hilfe e.V.
• 1995 Practical at a Hamburg newspaper.
• 1995 Installation and maintenance of Computer equipment as well as programming for my father’s project in Mbabane, Swaziland (3 months).

OTHER QUALIFICATIONS:
• First Aid (1994, 1996)
• Fitness Instructor (1995)
• Naui Scuba Diver (2000)

NATURAL LANGUAGES: German (mother tongue), English, French.


OPERATING SYSTEMS: Windows (3.1, 95, 98, NT), MSDOS, Unix, Linux, Irix.

INTERESTS: Computers, Sports (Fitnesstraining, Squash, Diving, Karate), Drawing, Painting, Reading, Music.