The Wavelet Stream: Interactive Multi Resolution Light Field Rendering

Ingmar Peter and Wolfgang Straßer
WSI/GRIS, University of Tübingen, Germany
[peter|strasser]@gris.uni-tuebingen.de

Abstract. One of the most general image based object representations is the Light Field. Unfortunately, a large amount of data is required to reconstruct high quality views from a Light Field. In this paper, we present the *wavelet stream* which employs non-standard four-dimensional wavelet decomposition for Light Field compression. It allows for progressive transmission, storage, and rendering of compressed Light Field data. Our results show that 0.8% of the original coefficients or 0.3 bits per pixel, respectively are sufficient to obtain visually pleasing new views. Additionally, the wavelet stream allows for an adaptive multi-resolution representation of the Light Field data. Furthermore, a silhouette-encoding scheme helps to reduce the number of coefficients required. Our data structure allows to store arbitrary vector-valued data like RGB- or YUV-data. The Light Field data stored in the wavelet stream can be decompressed in real time for interactive rendering. For this, the reconstruction algorithm uses supplementary caching schemes.

1 Introduction

Although the capabilities of computer graphics hardware have dramatically increased in the past years, the handling of scene complexity is still one of the most fundamental problems in computer graphics. One proposed solution for this is *image based rendering (IBR)*. Instead of constructing an exact geometry model, image based rendering uses pictorial information to describe an object. This has several advantages: The rendering time of the object is independent from its actual properties and depends only on the resolution of the data. Additionally, image based objects can be easily obtained from synthetic or real world objects, utilizing only pictures of them.

In the recent years a large number of image based approaches have been proposed. They differ from each other in generality, utilization of geometrical information, and memory requirements. One of the most general image based object representations is the *Light Field* respectively the *Lumigraph*, which were simultaneously introduced by Levoy and Hanrahan [7] and Gortler et al. [2], respectively. Although the Light Field and the Lumigraph differ in detail, in the following we will refer to both data structures uniformly as *Light Field*.

Assuming that a static object is located inside non-participating media, a four-dimensional, RGB-valued sample of the plenoptic function [1] is sufficient to describe the object's visual appearance. According to this a Light Field represents an arbitrary object by a two-dimensional array of images where all images share the same image plane. It is independent from the geometry as well the material properties of the object it represents. Furthermore a Light Field allows for reconstruction of almost arbitrary

views of an object and can be used to represent synthetic as well as real world objects. Unfortunately, a large amount of data has to be stored in a Light Field to obtain new views of good quality. Therefore, various approaches were proposed for efficient compression of Light Field data.

In this paper, we present a new data structure for progressive transmission, storage, and rendering of compressed Light Field data. The *wavelet stream* employs non-standard four-dimensional wavelet decomposition to make use of the coherence in all of the four dimensions of the Light Field data. Only a small fraction of the original coefficients are sufficient to obtain visually pleasing reconstruction results. An additional *silhouette-encoding scheme* helps to reduce the number of coefficients required for encoding the Light Field. Furthermore the wavelet stream enables storing of arbitrary vector-valued data like RGB- or YUV-data. The data can be decompressed in real time and thus allows for use in interactive graphic applications. The reconstruction algorithm used during rendering is completed by supplementary caching schemes.

The remaining part of this paper is structured as follows: In the next section related work is briefly reviewed. The approach used for wavelet decomposition of Light Field data is introduced in Section 3. The wavelet stream data structure is described in detail in Section 4. Section 5 presents the results obtained. We conclude with some ideas for future research.

2 Related Work

Various approaches for Light Field data compression were proposed. Although the size of the data grows with $O(n^4)$ with respect to the resolution, the chance of obtaining high compression ratios without significant loss of quality is very good, since the sample values are highly correlated in every single dimension.

Levoy and Hanrahan [7] use vector quantization (VQ) and in a second step Lempel-Ziv encoding to compress the data. The drawback of this approach is, that the whole Light Field has to be uncompressed before it can be used, because Lempel-Ziv coding does not allow for random access. In [10] the use of Discrete Cosine Transform in combination with block coding techniques is suggested. Spherical harmonics were used by Wong et al. [18] to solve the compression problem.

Ihm et al. [4] proposed an alternative parameterization of the Light Field on the surface of a positional sphere enclosing the object. On its surface smaller directional spheres are placed which encode the directions of the samples. The directional spheres are then adaptively triangulated and the associated values are compressed using wavelet compression. Due to the fact, that only two-dimensional wavelet compression is used, the coherence in two dimensions of the data remains unused. In [5, 9] coding schemes similar to the techniques used in video compression are proposed. Prediction images are selected in two dimensions from the light field data and subdivided into blocks. The remaining data is then reconstructed applying simple operations on these blocks. Schirmacher et al. [13] uses an initial sparse Lumigraph which is interactively refined through the use of a specialized warping algorithm.

In [6, 11, 8] four-dimensional wavelet decomposition is used for compression of Light Field data. Up till now only wavelet-based compression provides a real multi resolution representation of Light Field data. Despite the use of elaborate caching schemes during data decompression, the approach of Lalonde et al. [6] reconstructs Light Fields with a resolution of 32^4 samples only in real time. In [8] high compression ratios using wavelet compression are reported. However, this approach by Magnor et al. too is not able to reconstruct the Light Field data fast enough for use in interactive applications.

These two examples [6, 8] show that two prerequisites have to be taken into account when developing a wavelet-based compression scheme for Light Field data:

- Since the generation of a specific view of an Light Field is equivalent to projecting the four-dimensional Light Field onto a plane, particular values only of the compressed data have to be accessed and decompressed [3]. Thus random access of arbitrary values of the data has to be supported by the compression scheme.
- To enable the utilization of Light Fields in interactive applications, the data has to be uncompressed in real time.

The wavelet stream data structure as described in Section 4 archives high compression ratios, while preserving a good image quality. Supplementary caching schemes allow for interactive rendering of a wavelet-compressed Light Field.

3 Wavelet Decomposition for Light Fields

Wavelets are a well-established tool for computer graphics and used in many applications including image editing and compression, automatic level-of-detail control for editing, and rendering curves and surfaces or global illumination. For a comprehensive introduction into wavelets we refer to [17].

A *wavelet decomposition* transforms a signal into a collection of a *scaling coefficient* c_0^0 and *detail coefficients* d_i^j. Beside the capability of providing different level of details (LOD), while having the same storage requirements as the original signal, the wavelet decomposition of a signal allows for (lossy) compression of the signal by leaving out coefficients with low or zero value.

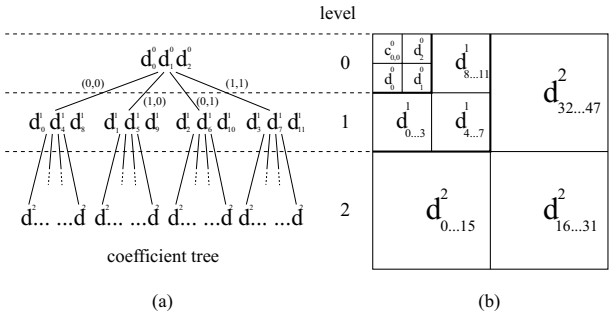


Fig. 1. Connection between structure of decomposed data (b) and corresponding tree of wavelet coefficients (a).

To meet the requirements for a fast and effective wavelet compression of Light Field data, the employed wavelet basis have to be chosen carefully. As stated in Section 2, particular values only of the Light Field data are accessed and reconstructed during rendering. To allow the use of wavelet compressed Light Fields in interactive applications the number of coefficients required for reconstructing a specific value should be small. Furthermore the mandatory arithmetic operations should be as simple as possible.

Both requirements are fulfilled by the Haar wavelet basis. The Haar wavelets are *tree wavelets*. This means that the wavelet functions ψ_i^j of each level j have disjunct support. Because of this, the coefficients can be organized in a *coefficient tree* as shown

in Figure 1(a). In contrast to approaches from the field of image compression e.g. [12, 14] each node in the coefficient tree holds a set of coefficients which describe a contiguous subset of the Light Field data at a particular resolution. Thus for reconstruction of a particular Light Field value the coefficients stored in the nodes of the coefficient tree which relate to the respective Light Field value are sufficient only. Additionally, a value can be reconstructed from Haar wavelet coefficients using subtraction and addition only.

The *nonstandard construction* approach [17] is used to create the four-dimensional Haar wavelets and scaling functions from the one-dimensional ones. In practice, the wavelet decomposition of a data set using nonstandard Haar wavelets means to perform one step of pair wise averaging and differencing sample values in each row of the data. After doing so in each of the at most 4 dimensions of the data, the calculated coefficients are reordered separating the scaling and the detail coefficients. Then the process is repeated recursively on the scaling coefficients. At the end of the recursion, the data consists of detail coefficients d_i^j for each resolution level j and one scaling coefficient c_0^0 representing the mean value of the Light Field as shown in Figure 1 (b).

Given a threshold $\tau > 0$ the decomposed Light Field data is compressed discarding all coefficients $|d_i^j| < \tau$. The L^2 error introduced by this equals

$$\epsilon_{rms} = \sum_{|d_i|<\tau} d_i^2$$

for an orthonormal wavelet basis.

4 The Wavelet Stream Data Structure

Without loss of generality, we will use in the following for simplicity and comprehensibility two-dimensional notation and data which consists of single values in figures. All described algorithms and data structures are capable of handling four-dimensional vector-valued data sets.

4.1 Coefficient Storage

As stated in the preceding section, the Haar wavelets are tree wavelets. Therefore, the wavelet coefficients can be organized in a coefficient tree as shown in Figure 1 (a). In this tree the coefficients can be grouped disjunctly for each level j. To support progressive transmission, storage, and refinement of Light Field data, the entries of the coefficient tree are ordered in decreasing importance in the *coefficient stream*. This means to write the coefficients into the stream in the order as found in the tree when it is traversed in breadth-first order (Figure 2).

Usually the input data for wavelet compression of Light Fields consists of entries of 3×8 bit for RGB or 4×8 bit size for RGBα valued Light Fields [7]. During wavelet decomposition non-integer coefficients d_i^j are created[1]. To keep the coefficients byte-aligned for easy access only 8 bit are used to store a coefficient d_i^j. In this way some additional error is introduced due to the necessary quantization of the values.

A simple and powerful approach to reduce the quantization error significantly is to analyze all coefficients of each level j in the coefficient tree and find their absolute

[1]The decomposition is carried out using floating-point precision.

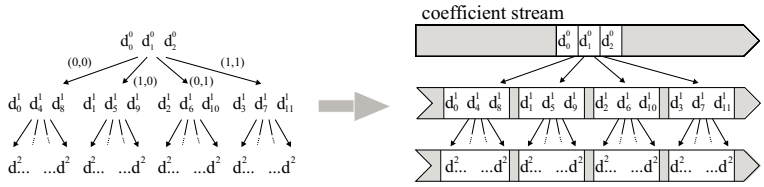


Fig. 2. Order of coefficients in the coefficient stream: The coefficients are ordered in decreasing importance by traversing the coefficient tree in breadth-first order.

minimum and maximum values min_j and max_j. These are stored in a table with floating point precision. The table is later added to the wavelet stream. For each level all coefficients are then scaled so that the maximum value range $[-128, 127]$, which can be represented using a signed byte, is exploited by the coefficients. The minimum and maximum values obtained earlier are used to correct the coefficient values during decompression. We found that this technique reduces the error introduced by quantization by a factor of approximately 3.

4.2 The Node Description Data Structure

The data stored in the wavelet stream can consist of several independent *channels* e.g. red, green, and blue or YUV. These can have different compression properties. Because of this the number of channels encoded in a node can decrease with increase of the level in the wavelet stream. We say a channel to be *active* if a node or one of its children stores information regarding to this channel. During wavelet stream traversal an array of flags is used to keep track of the channels currently active.

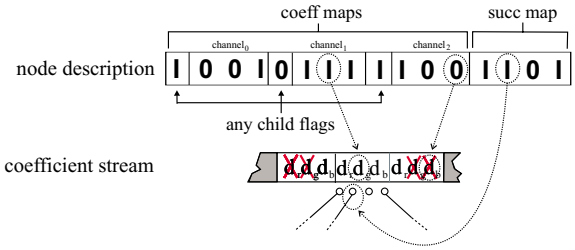


Fig. 3. Node description with the respective part of the coefficient stream. For three entries in the node description their meaning for the entries in the coefficient stream is depicted. Discarded coefficients are marked with a cross.

The coefficient tree as shown in Figure 2 can become incomplete due to the removal of coefficients during compression. Subtrees can be discarded completely if all of their coefficients are discarded. Therefore, additional meta information has to be added to the coefficient stream to encode the position of existing coefficients, active channels, and existing children of each node. As stated in Section 2, during rendering only parts of the Light Field data has to be accessed and decompressed. Unfortunately, approaches like [12, 14] do not support random access of arbitrary wavelet compressed values. Therefore, we had to develop our own approach to store the coefficients and the meta

information necessary for correct value reconstruction.

For each existing node N in the coefficient stream a *node description* is used, as depicted in Figure 3. For each active channel in N a single *coeff map* significance map [14] is used to encode the position of the existing coefficients.

A node N in the coefficient tree can be father of up to $2^{dim(N)}$ children, which themselves are the root nodes of up to $2^{dim(N)}$ subtrees, where $dim(N)$ is the dimension of N. If N has any children, an additional significance map, the *succ map*, is attached to the coeff maps of N's node description to encode the position of the existing children (Figure 3).

The *any child flag* (Figure 3) encodes which of N's channels are active in any of N's subtrees. If the any child flag of a channel C is set, the succ map significance map exists and in all subtrees of N information relating to C is stored. When the succ map exists, at least one of the any child flags of a node description has to be set. An unset any child flag indicates that no information which relates to the respective channel C is stored in any successors of the node. C is then said to be *deactivated* in all of the node's children.

4.3 Node Description Packing

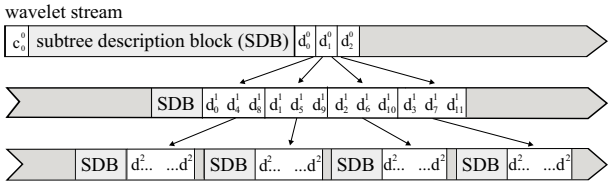


Fig. 4. Wavelet stream with subtree description blocks (SDB) in front of each group of nodes with a mutual father.

One can think of inserting a node description into the coefficient stream directly in front of the coefficients it describes. But this would waste a significant amount of memory since the wavelet coefficients are stored byte aligned for efficient access. The dimensionality of the nodes usually decreases towards the leafs of the coefficient tree. Therefore, the size of the significance maps stored in a node description can shrink to 3 bits or only 1 bit for a two-dimensional or one-dimensional node, respectively. If none of the any child flags in a node description are set which is the case for all leaf nodes, the size of a node description is $2^{dim} \times c$ bits in total, where dim denotes the dimension of the node and c the number of active channels. Since at least one byte has to be used for each node description, 4 respectively 6 bits are left unused ($c = 1$) for two- respectively one-dimensional nodes. For example, in the worst case in a wavelet stream encoding a Light Field with a resolution of $256 \times 256 \times 8 \times 8$ samples, 512 Kbytes or 8 percent of the total size of the wavelet stream would be wasted.

Instead, the node descriptions of all nodes which have the same father, are stored in a mutual *subtree description block (SDB)* in front off all coefficients they describe (Figure 4). They are packed behind each other in a single block of meta information leaving a gap of unused bits at most in the last byte of the SDB. This saves not only memory it also reduces the number of bytes necessary for efficient navigation inside the wavelet stream (Section 4.4). Behind the subtree description block all coefficients belonging to the nodes described in the SDB are stored.

After adding the subtree description blocks, the wavelet stream contains all information necessary to store, transmit, and reconstruct an incomplete coefficient tree. The scaling coefficient c_0^0 is stored at the beginning of the wavelet stream.

4.4 Value Reconstruction and Navigation

The reconstruction of a particular value from the wavelet stream is a recursive process starting at the root of the tree. It is equivalent to traverse the coefficient tree on a path which is determined by the coordinate of the requested value. During traversal the coefficients of all visited tree nodes contribute to the result.

For traversal of the tree at each node of the path a particular child has to be accessed. This means to go forward a number of bytes. Since single coefficients or complete subtrees might have been removed from the coefficient tree during compression, it is not possible to compute an offset from an arbitrary node to any of its children using a simple rule.

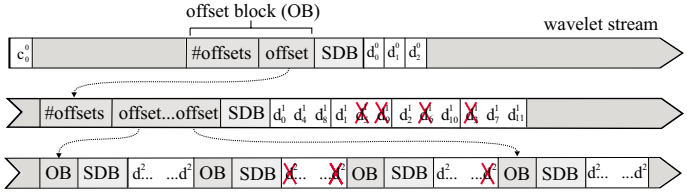


Fig. 5. Wavelet stream with offsets inserted in front of the subtree description blocks (SDB). The offsets point to the first child of each subtree described in a SDB. Discarded coefficients are marked with a cross.

To allow fast traversal along arbitrary paths, additional navigation information is added to the wavelet stream, when it is loaded into memory for rendering. For each node that has at least one child (Figure 5), an offset is inserted into the wavelet stream. This offset points to the first child of the respective node. Since all nodes with a mutual father are packed into a single block as described in Section 4.3, this means that the offset simply marks the beginning of the respective subtrees *offset block* (Figure 5). To save memory, the size of the offset representation itself is reduced to the actual needs. For each level the maximum offset and the number of bytes sufficient to encode its value are determined and stored in an additional table.

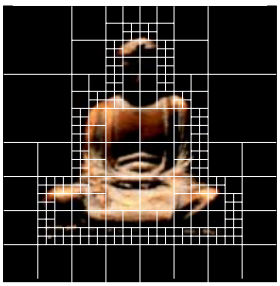
During value reconstruction a particular child of a node has to be found so that its coefficients can be used for reconstruction. Since the size of the offset block and the subtree description block (SDB) can not be known in advance, for each block an additional byte is inserted into the wavelet stream giving the number of offsets and the size of the SDB (Figure 5).

The coefficients of a particular node N can be found by parsing its SDB until the node description of N is reached. During value reconstruction the *coeff map* significance maps of N indicates which of the coefficients exist. The traversal of the wavelet stream continues if at least one *any child flag* and the corresponding bit in the *succ map* significance map is set.

4.5 Silhouette Encoding

Usually a solitary object is encoded in a Light Field. Unfortunately, due to the high frequency of the object's border, it causes the generation of a large number of detail coefficients during wavelet decomposition. If for each Light Field value its affiliation to the object is known, a number of wavelet coefficients, which is not larger than the number of pixels belonging to the object, must be sufficient to encode the object's visual appearance.

To utilize this fact the silhouette of the object is encoded. The silhouette information is stored in a *silhouette tree* as depicted in 6(a). It is fully embedded into the wavelet stream. Since the silhouette tree's subdivision scheme is identical to that used by the wavelet decomposition, we can exploit coherence in the wavelet stream and encode the silhouette tree using binary information only. In Figure 6(b) the extension of the node description data structure for silhouette encoding is depicted. Only for nodes whose Light Field values partly belong to the object and partly not, an additional *silhouette map* significance map is stored.



(a)

(b)

succ map	silhouette map	object?
1	1	mixed
1	0	yes
0	0	no
0	1	yes

(c)

Fig. 6. Extension of the wavelet stream for encoding of silhouette information. (a) subdivision scheme used by silhouette tree (b) extended node description data structure (c) interpretation of silhouette map and succ map entries to determine the affiliation of a particular pixel to the object.

During uncompressing the affiliation of pixels to the object is determined by interpreting the entries of the succ map and the silhouette map according to Table (c) in Figure 6.

When calculating the wavelet coefficients during decomposition using silhouette information a special approach has to be used. Since all four-dimensional Haar wavelets of a particular level have disjunct support, the entire process of reconstructing a value from the wavelet coefficients stored in a node can be described by a matrix M. The Light Field values $\vec{v} := [v_0, ..., v_{n-1}]$ can be reconstructed by multiplying wavelet coefficients $\vec{d} := [c, d_0, ..., d_{n-2}]$ with M

$$\vec{v} = M\vec{d}^T$$

with $n = 2^{dim}$ where dim is the dimension of the respective node. Since we know from the silhouette encoding that $n' \leq n$ coefficients belong to the object, $n' - 1$ wavelet coefficients are sufficient to encode the objects visual appearance. Therefore, wavelet coefficients $c', d'_0, ..., d'_{n'-2}$ have to be calculated accomplishing

$$\vec{v} = M\vec{d'}^T$$

where $\vec{d'} = [c', d'_0, ..., d'_{n'-2}, 0, ..., 0]$. Let M' be the matrix M whose last $2^{dim} - n'$ columns and the rows relating to the non-object values in $\vec{v}$ were removed. Then the new wavelet coefficients $\vec{d''}$ can be obtained with

$$\vec{d''} = M'^{-1}\vec{v'}^{T}$$

where $\vec{v'}$ is the vector formed of the n' Light Field values which belong to the object and $\vec{d''} = [c', d'_0, ..., d'_{n'-2}]$. Using the wavelet coefficients $\vec{d'}$ for reconstruction the Light Field values which do not belong to the object, are assigned arbitrary values. Since the usage of silhouette information allows excluding of those values this is not a problem.

For typical data sets about 10 percent of the entire data have to be spend to encode the silhouette information. On the other hand the utilization of silhouette information reduces the number of coefficients up to $20 - 30$ percent while obtaining the same compression error compared to data sets compressed without using silhouette information.

4.6 Rendering and Caching Scheme

The Light Field is rendered in a fashion similar to [16]. The retrieved Light Field values are plotted into a texture whose resolution is identical to the image plane resolution of the Light Field. This has the advantage that the coordinates for the requested Light Field values can be calculated by simple increments. The texture is then mapped onto a rectangle positioned on the image plane of the Light Field and displayed using the OpenGL.

To improve rendering speed, three different caching levels are used during Light Field rendering. The *texture cache* utilizes texture memory of the graphics hardware to store already calculated Light Field views. According to a last recently used policy a fixed number of Light Field views are stored together with the respective observer's positions. Before generating a new Light Field view the texture cache is searched for the Light Field view whose observer's position is next to that of the current observer. If the distance between current and cached observer does not exceed a user-specified limit, the Light Field view from the texture cache is used. In a background process the correct Light Field view is rendered and cross-faded onto the cached view.

If the texture cache misses, the *image based* cache is queried next. It stores for each pixel of the most recent generated Light Field view all pixel values along with their camera plane coordinates. As the image plane coordinate of each pixel is given implicitly a simple comparison of the camera coordinate is sufficient to decide whether a cached pixel value can be reused or not. This technique is only applicable if no interpolation is applied to the Light Field values as it is done during interactive Light Field rendering.

If a particular Light Field value has to be reconstructed the third-level *tree cache* is employed. Usually the coordinates of consecutive requested Light Field values differ only slightly. Therefore during traversal of the coefficient tree at every node the result calculated so far is stored together with some navigation information. When the next value is requested from the Light Field the current path is compared to the cached path and cached values are reused as long as possible.

5 Results

The wavelet stream data format as described in the previous section was implemented in C++ using a flexible framework of classes. It is possible to exchange, modify, and

compare the various parts of our method. Furthermore, the class framework provides an environment for developing high-performance interactive applications.

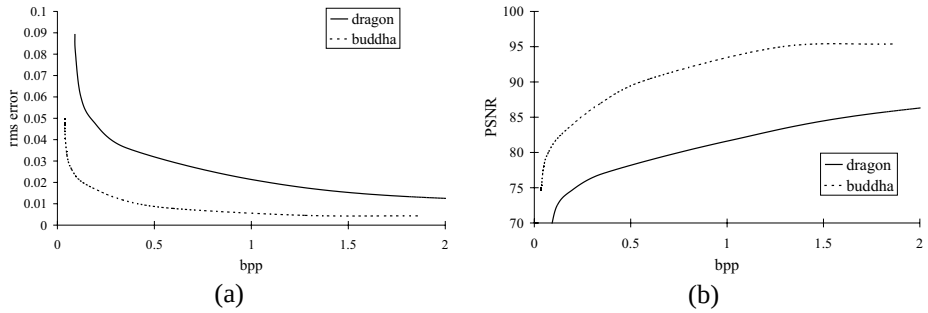


Fig. 7. Root-mean-square (rms) error (a) and peak-signal-to-noise-ratio (PSNR) (b) for Buddha and Dragon Light Field.

To measure the error introduced by compression and subsequent Light Field data reconstruction, we used the root-mean-square (rms) error ϵ_{rms} and the peak-signal-to-noise-ratio (PSNR) for relative measure. To test our approach we used the well known "Dragon" and "Buddha" Light Fields which are freely available from the Stanford Light Fields Archive [15]. Both Light Fields have a resolution of $256 \times 256 \times 32 \times 32$ 24 bit RGB values, which results in a total size of 192 Mbytes. For further improvement all Light Fields were transformed prior to compression from RGB to YUV and vice versa during Light Field rendering.

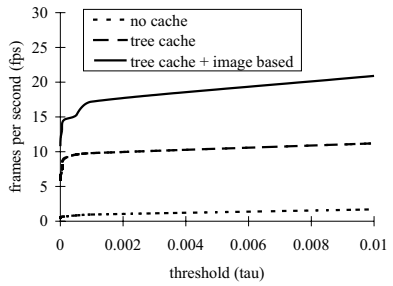


Fig. 8. Rendered frames-per-second (fps) for different caching schemes as function of compression threshold τ.

Figure 7(a) shows ϵ_{rms} for both Light Fields and different compression ratios. The results obtained from the Buddha Light Field are better because of its smooth surface. The curve showing the rms error does not intersect the x-axis, due to the fact that zero coefficients can be removed from the wavelet stream without loss of information and some error is introduced by the quantization of the non-zero coefficients. In this way an initial compression is always given when using wavelet compression. All compression ratios were measured with respect to the file size of the compressed Light Field data. The PSNR depicted in Figure 7(b) shows the good performance of wavelet-based Light Field compression as well. Again the error measurements obtained from the Buddha Light Field are better compared to the Dragon Light Field.

Figure 8 depicts the frames per second (fps) measured while rendering the Dragon light field with a resolution of 400×400 pixels on a Athlon 1200 MHz PC with

GeForce2 GTS graphics board. The frame rate increases when the threshold τ, used to discard the wavelet coefficients during compressing, is raised. This is due to the fact that the average length of the paths, on which the coefficient tree has to be traversed during reconstruction, is reduced as more of the coefficients are discarded. The acceleration caused by the texture cache (Section 4.6) is not depicted in chart 8, since on a cache hit the texture cache works as fast as the employed graphics hardware can load and display a texture. Since the speedup is archived only when a Light Field next to a previously used viewpoint in requested, this might lead to glitches in the archived frame rate.

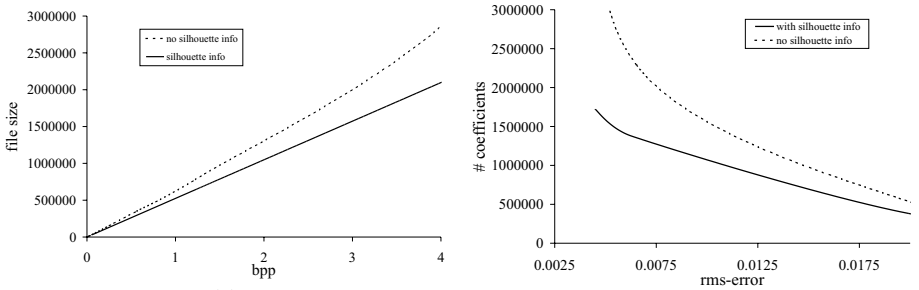


Fig. 9. Comparison of (a) the total file size and the (b) the number of coefficients of a wavelet-compressed Light Field using silhouette information and no silhouette information.

In Figure 9(a) the total file size is plotted as function of the bits per pixel (bpp) used for encoding the Light Field data when using silhouette information and not using silhouette information. Figure 9(b) shows the number of coefficients sufficient to encode a wavelet-compressed Light Field for obtaining a particular rms-error. The silhouette-encoding scheme as described in Section 4.5 works best for Light Fields with a small compression error. This is due to the fact that Light Fields compressed preserving a high image quality has to store a higher number of coefficients and thus the number of redundant coefficients due to object's border is larger.

Finally, Figure 10 (see Appendix) shows Light Fields which were compressed with different ratios. The image quality is only slightly decreased when the compression ratio increases.

6 Discussion and Future Directions

We presented a new approach for storage, transmission, and rendering of compressed Light Field data. The wavelet stream is especially well suited for progressive transmission of Light Fields over networks, because the most important parts of the data are transferred first. To decorrelate the data, 4D wavelet compression with nonstandard orthonormal Haar wavelets was used. For encoding the positions of the coefficients not discarded during compression, the new wavelet stream data format was introduced. Our approach is capable of storing arbitrary vector-valued data. The compression ratios obtained proved that only a small fraction of the original data is sufficient to obtain visually pleasing views from highly compressed Light Fields.

Since the wavelet stream can store arbitrary values, we will use this in future to investigate the visualization of Light Fields decorated with additional values, e.g. material properties or multi-dimensional pixel values. Future work will also include the implementation of the wavelet stream in Java to provide easy and fast access to Light

Field data over the Internet.

Acknowledgements

The authors wish to thank Stanislav Stoev and Michael Wand for valuable discussions and careful proof reading of the manuscript. We also wish to thank the anonymous reviewers who improved this work with their comments and suggestions. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) and is part of the V^3D^2-project "Distributed Processing and Exchange of Digital Documents".

References

1. E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, 1991.
2. Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH 96 Conference Proceedings*, pages 43–54. August 1996.
3. Xianfeng Gu, Steven J. Gortler, and Michael F. Cohen. Polyhedral geometry and the two-plane parameterization. In *Eurographics Rendering Workshop 1997*, pages 1–12, June 1997. Springer Wien.
4. Insung Ihm, Rae Kyoung Lee, and Sanghoon Park. Rendering of Spherical Light Fields. In *5th Pacific Conference on Computer Graphics and Applications*, pages 59–68, 1997.
5. Ming-Hoe Kiu, Xiao-Song Du, Robert J. Moorhead, David C. Banks, and Raghu Machiraju. Two Dimensional Sequence Compression Using MPEG. In *SPIE Vol. 3309, SPIE/IS&T Electronic Imaging '97*, San Jose, CA, January 1998.
6. Paul Lalonde and Alain Fournier. Interactive rendering of wavelet projected light fields. In *Graphics Interface*, pages 170–114, June 1999.
7. Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH 96 Conference Proceedings*, pages 31–42. August 1996.
8. M. Magnor, A. Endmann, and B. Girod. Progressive compression and rendering of light fields. In *VMV 2000*, pages 199–204, Saarbrücken, Germany.
9. Marcus Magnor and Bernd Girod. Data Compression in Image-Based Rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, April 2000.
10. Gavin Miller, Steve Rubin, and Dulce Ponceleon. Lazy decompression of surface light fields for precomputer global illumination. In *Rendering Techniques '98*, pages 281–292. 1998 Springer Wien.
11. Ingmar Peter and Wolfgang Straßer. The wavelet stream: Progressive transmission of compressed light field data. In *IEEE Visualization 1999 Late Breaking Hot Topics*, pages 69–72. IEEE Computer Society, October 1999.
12. Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, June 1996.
13. Hartmut Schirmacher, Wolfgang Heidrich, and Hans-Peter Seidel. High-quality interactive lumigraph rendering through warping. In *Graphics Interface*, pages 87–94, May 2000.
14. J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 41(12):3445–3462, 1993.
15. The Stanford Light Fields Archive. `http://www-graphics.stanford.edu/software/lightpack/lifs.html`.
16. Peter-Pike Sloan and Charles Hansen. Parallel Lumigraph Reconstruction. In *Proc. of PVG 99*, San Francisco, October 1999.
17. Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, 1996.
18. T.-T. Wong, P.-A. Heng, S.-H. Or, and W.-Y. Ng. Illumination of image-based objects. *The Journal of Visualization and Computer Animation*, 9(3), 1998.

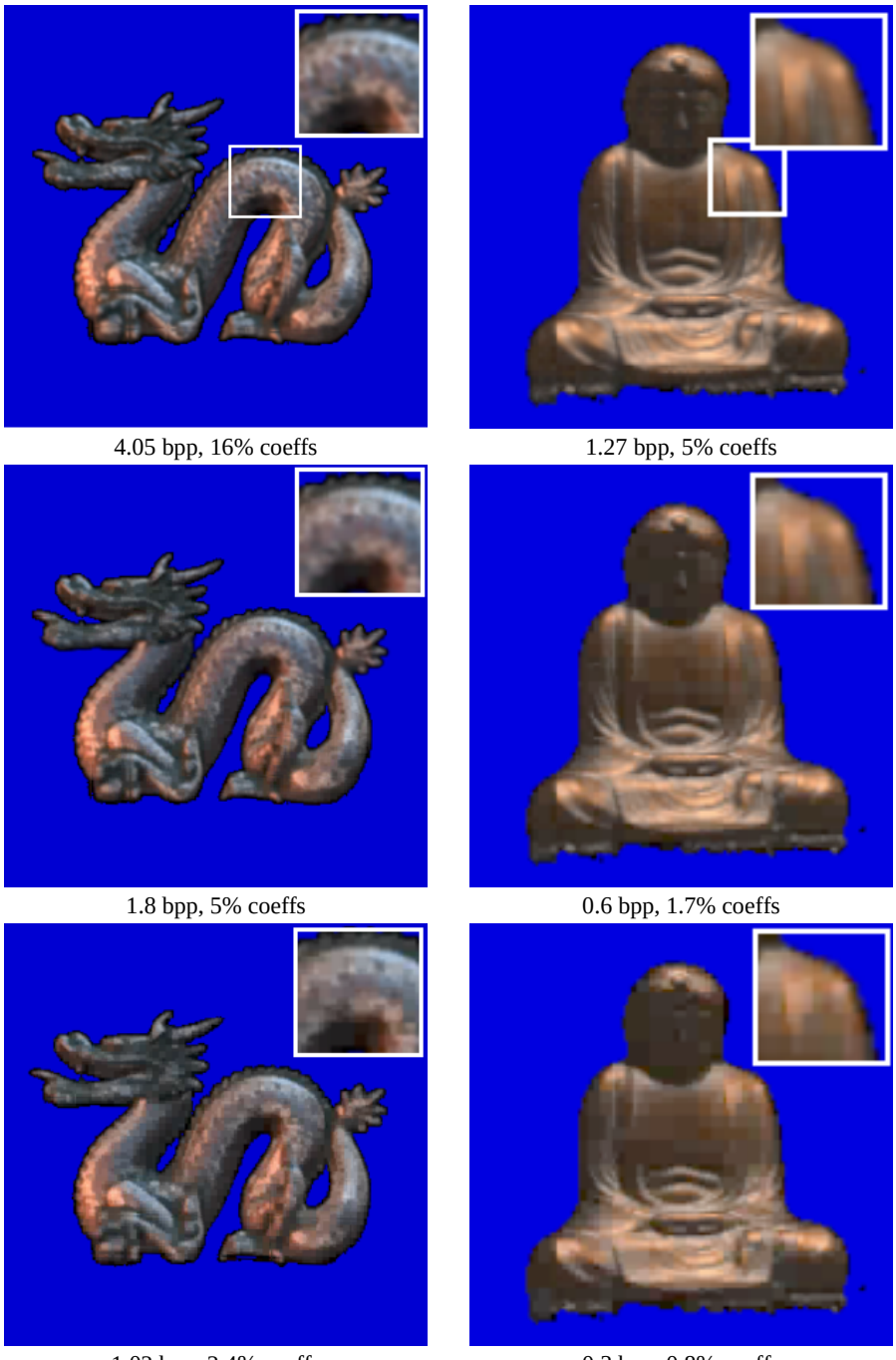


Fig. 10. Dragon and Buddha Light Field with magnified detail for different compression ratios: With increasing compression ratio, the obtained image quality decreases only slightly.