Real-time Rendering of Dynamic Objects in Dynamic, Low-frequency Lighting Environments

Ruifeng Xu
University of Central Florida
rxu@cs.ucf.edu

Sumanta N. Pattanaik
University of Central Florida
sumant@cs.ucf.edu

Charles E. Hughes
University of Central Florida
ceh@cs.ucf.edu

1. Abstract

This paper presents a pre-computation based method for real time global illumination of dynamic objects. Each frame of animation is rendered using spherical harmonics lighting basis functions. The pre-computed radiance transfer (PRT) associated with each object's surface is unfolded to a rectangular light map. A sequence of light maps is compressed using a high dynamic range video compression technique, and uncompressed for real-time rendering. During rendering, we fetch the light map corresponding to each frame, and compose a light map corresponding to any arbitrary, low-frequency lighting condition. The computed surface light map can be applied to the object using the texture mapping facility of a graphics pipeline.

The primary contribution of this paper lies in its pre-computation based real time global illumination rendering of dynamic objects. Spherical harmonics light maps (SHLM) are used to represent the pre-computation results, and the animation can be viewed from arbitrary viewpoints and in arbitrary low-frequency environment lighting in real time. The consequence is an algorithm that is capable of high quality rendering of animated characters in real-time.

Keywords: Real-Time Techniques, Light Map, Global Illumination, Mesh Parameterization, High Dynamic Range Video

1 Introduction

Global illumination (GI) is a key to realism. Despite over 20 years work [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], GI of static scenes is still expensive. Adding dynamics (character animation) makes GI even more challenging, for instance, complicating maintenance of the rendering acceleration data structure.

The work presented in this paper is inspired by the pre-computed radiance transfer (PRT) work of [13, 14, 15, 16]. With the incident radiance to each vertex pre-computed, real-time rendering is achieved by performing a few spherical harmonics coefficient multiplication and addition operations for each vertex [17, 18]. The use of PRT makes the solution of this previously daunting real-time GI problem possible. A straightforward extension to dynamic objects is to pre-compute each animation frame for use in the later rendering stage, but this will produce a huge volume of data (Figure 5). This quantity of data will prevent real-time rendering performance by requiring substantial time to load data from disk to memory. The work presented in this paper addresses this problem. We unfold the object surface to a 2D parameter plane, where each point is associated with the PRT of its corresponding 3D point on the object surface. The "image" of the PRT is then compressed. The compression drastically reduces the data volume and thus allows us to carry out our rendering task in real-time. Moreover, the choice of sampling rates and compression algorithms can lead to a variety of level-of-detail strategies, supporting the applicability of this approach to complex scenes.

2. Previous Work

Because of the difficulties of rendering dynamic objects in real time, not much prior work has been done in this area. The most relevant to ours is [19], where an impulse response function (IRF) is used to record the pre-computed radiance transfer (PRT) for later rendering of impulsive dynamics. Their work does well in, but is limited to, impulsive dynamics. Our work is to pre-compute the PRT of primitive animation sequences, like a walk or a turn, which are most likely used in the character's rendering. Just like an animation along any route can be assembled from many basic actions [20, 21, 22, 23], the rendered animation sequence can also be assembled from many basic animation PRTs.

All work related to PRT [13, 14, 15, 16] until now, are about static scene pre-computation. Our work builds upon them, extending this approach to scenes with animation.

2.1. Organization of the Paper

The rest of the paper first discusses the pre-computation process and the real-time rendering process. We then present experimental results. Our final sections present our conclusion and indicate future directions that this work might take.

```
Pre-computation phase:
Step A)
 For each animation frame $k$
   For each SH basis lighting $n$
     For each triangle $abc$
        Find $a'b'c'$ in parametric space
        For each pixel $d'$ inside $a'b'c'$
          Find $d$ in object space
          Store PRT of $d$ in $SHLM_n^k(d')$
Step B)
 Compress $SHLM^k$
```

(a) (see Figure 3)

```
Rendering phase:
Step C)
 During the rendering of frame $k$
   …
   Find lighting L
   Load $SHLM^k$
   Compute SHLM for L
   Feed SHLM to graphics pipeline
   …
```

(b)

Figure 1: Outline of the pre-computation and rendering processes. (a) is the pre-computation process; (b) is the rendering process.

3. GI Pre-computation

Our work is built upon the fact that an animation is made up of a sequence of animation frames. Each animation frame constitutes a particular pose of the animation. The PRT for each animation frame is computed. This process generates a huge volume of PRT data for an animation. Fortunately, the PRT is in a form ready for compression if recorded in the parameter space of the object surface. The general process is outlined in Figure 1(a).

We make use of a known locality property in our work: neighboring vertices in space and time tend to have similar PRT data, i.e., spatial and temporal coherence applies. This coherence also exists in image/video, and has been successfully exploited for compression. In a similar manner, the use of coherence-based compression techniques can greatly reduce PRT data.

Unlike previous work, our method computes the PRT for each sample in the 2D parameter space of the object surface. In this way, the PRT can be recorded as a 2D "super image"-each pixel consists of an incident radiance spherical harmonics coefficient vector. Thus, the inherent spatial and temporal coherence can be retained and exploited using image/video compression algorithms applied to PRT data compression.

One key component of our approach is the parameterization of the object surface, i.e., build a one-to-one correspondence between the object surface and 2D parameter space, say [0..1]×[0..1]. For generality and simplicity, we assume that the object surface is made up of triangle meshes. The object surface is unfolded using a mesh parameterization scheme [24, 25, 26, 27, 28], so that a one-to-one correspondence is built up between each object surface 3D point and each parameter space 2D point, as shown in Figure 2. The 2D parameter space [0..1]×[0..1] is sampled and the PRT of each sample is pre-computed and recorded there.

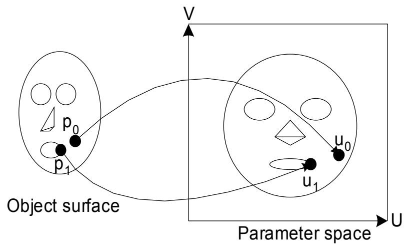


Figure 2: Unfolding object surface to 2D parameter space. The left object surface is unfolded to the right parameter space. p_0 is correspondent to u_0; p_1 to u_1.

The non-vertex correspondence can be obtained by using barycentric coordinates, as shown in Figure 3.

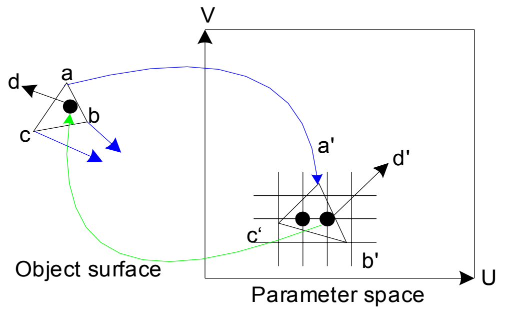


Figure 3: Mapping of non-vertex points using barycentric coordinates. a, b, c are triangle vertices, and a',b',c' are mapped triangle vertices in parameter space. Barycentric coordinates of d' are used to recover its original surface point d.

The incident PRT for each pixel in the parameter space is then calculated by applying the global illumination algorithm with each harmonics basis function as environment lighting [13, 14, 15, 16]. The result is an "image" with each pixel associated with its incident PRT spherical harmonics coefficients vector. We call this "image" the spherical harmonics light map (SHLM). In the rest of the paper we use

$SHLM_n^k$ to denote the SHLM for animation frame k and spherical harmonics basis lighting Y_n[1].

The sampling points located on the triangle edge can be shared by several neighboring triangles. PRTs for such sampling points are often computed more than once, and the mean of all the results are stored at that point.

3.1 Storage Compression of PRT

The amount of raw data from the pre-computation for an animation is huge (see Figure 5) and to make matters worse the data has a high dynamic range. Fortunately, the image structure of the SHLM, and the spatial and temporal coherence in the data stored in the SHLM lends itself to very high compression. A sequence of SHLMs constitutes a high dynamic range video, and can be compressed by making use of our HDR video compression approach. Details of our HDR video codec are given in the next subsection. It is convenient to tile all $SHLM_n^k$ for the same animation frame together as one $SHLM^k$. If up to N-th order spherical harmonics lighting are used in the PRT computation, $SHLM_n^k$ are placed at location $(\lfloor n/N \rfloor,\ n - N \cdot \lfloor n/N \rfloor)$, as shown in Figure 4. This results in a total of $N \times N$ tiles.

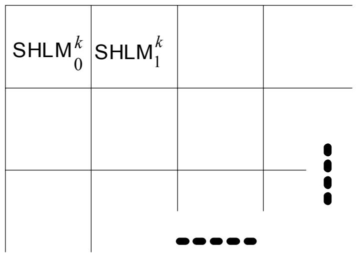


Figure 4: Arrangement of $SHLM_n^k$. The SHLM at row r, column c, will be the $SHLM_{5r+c}^k$.

The final $SHLM^k$ is sent to the HDR video coder for compression. In our test case, each map is 128×128. The order of the SH used is 2 and this makes a total of 3×3 tiles. Thus the resolution of each $SHLM^k$ is 384×384. Two SHLMs are used to cover the whole body, so the total SHLM size for one animation frame is 384×768. The original data 384×768×4×100 = 118M is compressed to 3.5M in 3.6 min on a Xeon 2.4GHz PC with 1G memory. The decompression cost is 78ms for each frame on the same machine.

[1] Y_n is actually spherical harmonics basis function Y_l^m where $l = \lfloor \sqrt{n} \rfloor$ and $m = n - l^2 - l$.

3.2 HDR Video Compression

We use the compression scheme shown in Figure 5. We store the floating point values of the spherical harmonic coefficients for the 3 color channels using the base and exponent used in the RGBE encoding schema [29]. This encoding converts the RGB triplet in floating point to a RGBE quadruplet with 8 bits per component. After this encoding, we separate the RGB components to compress them using an existing normal video compression approach, like MPEG [30]. We compress the E component in lossless mode [31]. The decision to use a lossless scheme for the E component is because of the fact that the quality of the decompressed HDR data is very sensitive to the errors introduced in the E component when compressed using a lossy compression scheme.

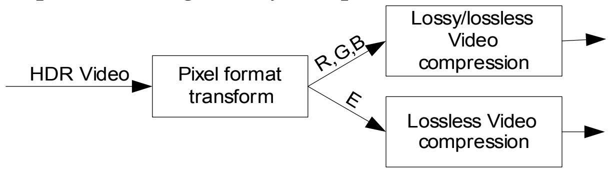


Figure 5: General HDR video compression scheme. R,G,B, i.e., color base, and E, i.e., exponential, are separated and sent to different compression schemes.

The HDR video compression mode used in this paper, which is implied by Figure 5 is:

- ☐ HDR video lossy compression
  R,G,B channels: lossy (MPEG-4, H.264, etc.)
  E channel: lossless (FFV1)

We control the quality of lossy compression for the RGB components appropriate to our memory budget. An alternative approach we are investigating uses JPEG 2000 [32].

JPEG 2000 is a wavelet-based image coding system for various types of still images (like grayscale, multicomponent, etc.). Each of its components supports dynamic range up to 16 bits. It provides a natural way to encode HDR image in lossy/lossless mode through linear quantization. The contrast sensitivity function (CSF) of the human visual system used in JPEG 2000 is easy to apply to HDR image encoding. The encoding/decoding of HDR images using the JPEG 2000 technique [33] is observed about 2 times slower than that using JPEG technique [34], which uses discrete cosine transform (DCT), although the former has much better compression quality in very high compression. We are exploring the possibility of developing an efficient GPU implementation of the wavelet decoding scheme for the application of JPEG 2000 technique to HDR video.

4. Rendering of Dynamic Objects

For rendering the *k*-th animation frame during the animation, we first compute the SH coefficients L_i, $i = 0,1 \ldots$, corresponding to the environmental light at the place [13] where our dynamic object is placed.

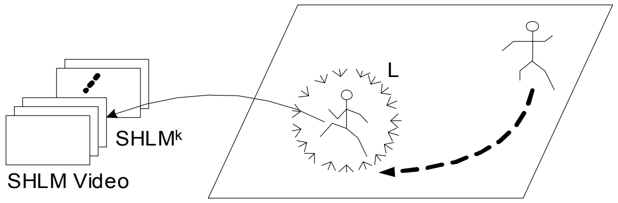


Figure 6: Rendering of a moving character. Current pose is used as index to fetch SHLM^k. Current position of character is used to estimate the environment lighting.

The *SHLM^k* is retrieved from the compressed HDR video. (See section 3.2 for HDR video codec details). The current lighting map is constructed by simply summing up the product of lighting coefficients and *SHLM^k*, as shown below.

$$SHLM = \sum_i SHLM_i^k \cdot L_i$$

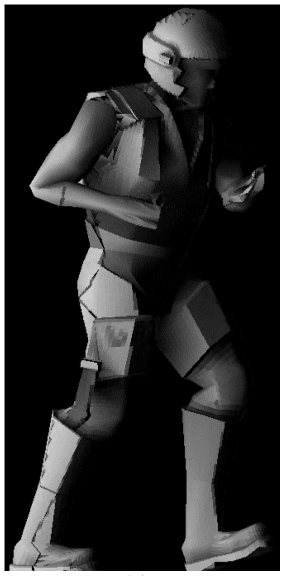
The obtained SHLM can be sent to the graphics pipeline as a texture. The dynamic object is then rendered by texture mapping. See Figure 1(b) for the outline of the program. Figure 6 gives the scenario of rendering.

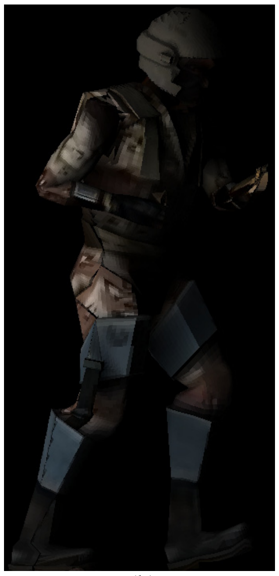
5. Experimental Results

Our experimental subject is a future soldier, an Offensive Force Warrior (OFW), modeled using 3DS MAX5.0. For a walk action lasting 2 seconds, we extract 100 frames by sampling its pose every 0.02 seconds. Each animation frame consists of 2811 vertices and 2197 triangles. A single frame is shown in Figure 7, displayed with lighting (7a) and lighting and texture (7b). The associated SHLM is shown in (7c).

We use "Radiance" (see http://radsite.lbl.gov/) to pre-compute the PRT. The spherical harmonics basis lighting is defined as a "glow" type in the "Radiance" scene definition file. For the parameterization, we make use of the *uv* coordinates generated by 3DS MAX.

The compressed HDR video size varies with the compression quality. Using the highest compression quality, compressed data is about 4.2M. Using middle quality, the size dropped to about 3.5M.


(a)


(b)

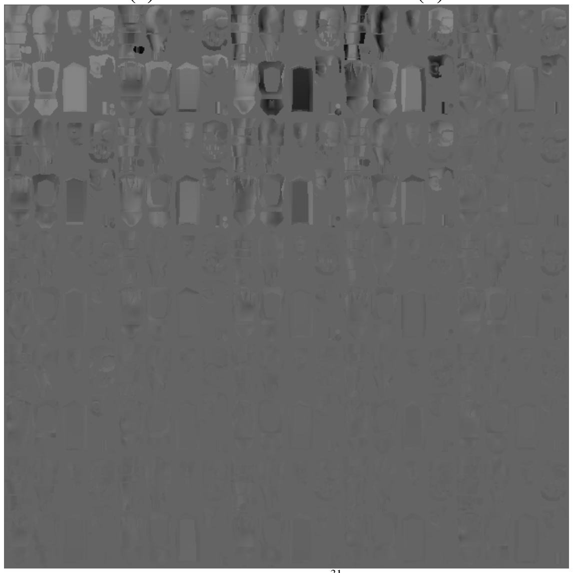

(c) SHLM^{31}

Figure 7: Some results. (a) with only lighting; (b) with lighting and texture;(c) SHLM for animation frame 31 with SH order up to 4.

The object surface is diffuse with a reflectance of 0.5. Since illumination is executed as a texture mapping process, our algorithm is ready for implementation on a GPU. SHLM is assembled in the CPU and sent to the GPU for rendering.

Some statistical data are recorded in Figure 8.

There are some ways to improve the performance of rendering. First, we can run the decompression step in a separate process from the main rendering, and prefetch the SHLM for the next animation frame. Second, it is possible to feed each compressed SHLM to graphics hardware and decompress it using the texture codec capability of graphics hardware to reduce the traffic between the graphics hardware and the CPU. Third, we can trade some trivial rendering quality by selecting fewer SH basis lighting functions. As shown in Figure 7(c), most PRT information converges to the first 9 SH basis lighting functions. We can also send key frame SHLMs to graphics hardware, and compute all the in-between frames by doing interpolation in graphics hardware.

Object	OFW (2811 vertices; 2197 triangles)
Action	2 sec. of walk (100 frames)
Max. order of SH	2 (9 coefficients)
Sampling Rate	128 by 128
Pre-computation time of GI computation	1.5 hours for 100 frames
Raw video data	>100 MB (RGBE format)
Compressed HDR video size	3.5M
Rendering speed	>10 frames/sec

Figure 8: Some statistics data of our experiment. The experiment is performed on an Xeon 2.4G with 1G memory running Windows XP.

6. Conclusions

We present a pre-computation based approach for real-time rendering of dynamic objects. The PRT of object surface points are computed and recorded in 2D parameterized space to form a sequence of SHLMs. To save storage this SHLM sequence is compressed using an HDR video compression technique. Dynamic objects are rendered by adding up the products of SHLMs and their lighting coefficients, and then applying the result to the object surface as a texture.

Our approach can perform GI of dynamic objects in real-time, and the objects can be viewed from arbitrary viewpoints and illuminated by arbitrarily low-frequency environment lighting. It is a new way for rendering dynamic objects, but it is restricted to the rendering of predefined actions. Fortunately, this limit can be overcome by combining this approach with motion synthesis techniques, like motion graphs [20].

Compared to [19], our work is suitable for fixed long actions. It is possible to combine our approach with that of James and Fatahalian [19] to take advantage of the fine dynamics rendering of their approach and of the capability for long actions of our approach.

Recording each PRT as a 2D rectangular image provides several other benefits. Since we record the PRT in 2D parametric space, the size of a PRT is independent of the number of vertices, but depends on only the object surface area and sampling rate. Thus, level-of-detail management is possible. Since our approach keeps the neighborhood of surface 3D points, the coherence between these neighboring points is exploited for data compression. Greater compression rates are achieved by using lossy compression schemes that throw away some high frequency information invisible to the human eye. Lossy compression of HDR image/video is a feasible way to achieve high compression ratio. Finally, this representation of PRT is easily implemented on current graphics hardware as a simple texture mapping.

Our work supports GI features, like self-shadowing and self-reflection to make objects look more realistic, but it cannot create neighboring shadows.

7. Future Work

The pre-computed data can be first compressed by applying PCA/CPCA [15, 16]. PCA/CPCA is independent of our approach and can be used to reduce the number of dimensions before HDR video compression.

We can use any mesh parameterization scheme. For example, [25] gives a signal specialized parameterization, which is a non-uniform parameterization approach that uses more samples wherever there are more details.

Our approach to store PRT has possible applications to other problems. The data from the surface light field [35] is of huge volume, and can be reduced by mapping light field data to the parametric space of its surface for compression. It is also possible to enhance the rendering effects by using BTF [14].

Another way of improving performance is to pre-compute only key animation frames, and interpolate for in-between frames in later rendering. We can also use non-uniform sampling; where the motion is smooth, we use a lower frame rate; where the motion is abrupt, we use a higher frame rate

In some cases where animation blending or inverse kinematics is applied, SHLMs can be blended or modified accordingly.

Acknowledgements

This work is partially supported by Office of Naval Research, ATI Research and the I-4 Corridor Fund.

References

[1] J. F. Blinn and M. E. Newell. Texture and Reflection in Computer Generated Images. In Communications of the ACM, 19, 542-546, 1976

[2] R. L. Cook, L. Carpenter, and E. Catmull. The Reyes image Rendering Architecture. In Computer Graphics (ACM SIGGRAPH 87 Proceedings)(July 1987), M. C. Stone, Ed., 21, 273-281, 1987

[3] P. Debevec. Rendering Synthetic Objects into Real Scenes: Bridging Traditional And Image-based Graphics with Global Illumination and High Dynamic Range Photography. In Proceedings of ACM SIGGRAPH 98 (July 1998), M. Cohen, Ed., 189-198, 1998

[4] H. W. Jensen and P. H. Christensen. Efficient Simulation of Light Transport in Scenes with Participating Media using Photo Maps. In ACM SIGGRAPH 98 Proceedings, 311-320, 1998

[5] J. T. KaJiya. The Rendering Equation. In Computer Graphics (ACM SIGGRAPH 86 Proceedings)(Aug. 1986), D. C. Evans and R. J. Athay, Eds., 20, 143-150, 1986

[6] E. Veach and L. J. Guibas. Metropolis Light Transport. In ACM SIGGRAPH 97 Proceedings (Aug. 1997), T. Whitted, Ed., 65-76, 1997

[7] G. J. Ward. The RADIANCE Lighting Simulation and Rendering System. In Proceedings of ACM SIGGRAPH 94, A. Glaussner, Ed.,459-472, 1994

[8] M. Levoy and P. Hanrahan. Light Field Rendering. In ACM SIGGRAPH 96 Proceedings, H.Rushmeier, Ed., 31-42, 1996

[9] D. A. Forsyth, C. K. Yang, and K. B. Teo. Efficient Radiosity in Dynamic Dnvironments. In Proc 5'th Eurographics Workshop on Rendering, 313-323, 1994

[10] P. Tole, F. Pellacini, B. Walter, and D. P. Greenberg. Interactive Global Illumination in Dynamic Scenes. In ACM Transactions on Graphics(Proceedings of ACM SIGGRAPH 2002), 21, 3, 537-546, 2002

[11] K. Bala, B. J. Walter and D. P. Greenberg. Combining Edges and Points for Interactive High-Quality Rendering. In ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003), 22, 3, 631-640, 2003

[12] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. Interactive Global Illumination using Fast Ray Tracing. In Rendering Techniques 2002(Proceedings of the 13th Eurographics Workshop on Rendering), 15-24, 2002

[13] P. -P. Sloan, J. Kautz, and J. Snyder. Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments. In ACM Transactions on Graphics, 21, 3, 527-536, 2002

[14] P. -P. Sloan, X. Liu, H. -Y. Shum, and J. Synder. Bi-Scale Radiance Transfer. In ACM Transactions on Graphics(Proceedings of ACM SIGGRAPH 2003), 22, 3, 370-375, 2003

[15] P. -P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered Principal Components for Precomputed Radiance Transfer. In ACM Transactions on Graphics(Proceedings of ACM SIGGRAPH 2003), 22, 3, 382-391, 2003

[16] J. Lehtinen and J. Kautz. Matrix Radiance Transfer. In Proceedings of ACM SIGGRAPH 2003 Symposium on Interactive 3D Graphics, 59-64, 2003

[17] R. Ramamoorthi and P. Hanrahan. An Efficient Representation for Irradiance Environment Maps. In Proceedings of ACM SIGGRAPH 2001, 497-500, 2001

[18] R. Ramamoorthi and P. Hanrahan. Signal-Processing Framework for Inverse Rendering. In Proceedings of ACM SIGGRAPH 2001, 117-128, 2001

[19] D. L. James and K. Fatahalian. Precomputing Interactive Dynamic Deformable Scenes. In ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003), 22, 3, 879-887, 2003

[20] L. Kovar, M. Gleicher and F. Pighin. Motion Graphs. In Proceeding of ACM SIGGRAPH 2002, 473-482, 2002

[21] C. –K. Liu, and Z. Popovic. Synthesis of Complex Dynamic Character Motion from Simple Animations. In Proceedings of ACM SIGGRAPH 2002, 408-416, 2002

[22] Y. Li, T. Wang, and H. –Y. Shum. Motion Texture: A Two-Level Statistical Model for Character Motion Synthesis. In Proceedings of ACM SIGGRAPH 2002, 465-472, 2002

[23] O. Arikan and D. A. Forsyth. Interactive Motion Generation from Examples. In Proceedings of ACM SIGGRAPH 2002, 483-490, 2002

[24] X. Gu, S. J. Gortler, and H. Hoppe. Geometry Images. In Proceedings of ACM SIGGRAPH 2002, 355-361, 2002

[25] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-Specialized Parameterization. In Thirteenth Eurographics Workshop on Rendering (2002), 87-98, 2002

[26] P. Alliez, M. Meyer, and M. Desbrun. Interactive Geometry Remeshing. In Proceedings of ACM SIGGRAPH 2002, 347-354, 2002

[27] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least Square Conformal Maps for Automatic

Texture Atlas Generation. In Proceedings of ACM SIGGRAPH 2002, 362-371, 2002

[28] A. Khodakovsky, N. Litke and P. Schroder. Globally Smooth Parameterizations with Low Distortion. In Proceedings of ACM SIGGRAPH 2003, 350-357, 2003

[29] G. W. Larson. Real pixels. Graphics Gems II. Academic Press. pp. 80-83, 1991

[30] D. Salomen. Data Compression: The complete Reference (2nd edtion), Springer, 2000

[31] FFV1. http://www.mplayerhq.hu/~michael/ffv1.html. 2004

[32] D. S. Taubman and M. W. Marcellin. JPEG 2000: Image Compression Fundamentals, Standards, and Practice. Kluwer Academic Publishers, Boston, 2002

[33] JasPer. http://www.ece.uvic.ca/~mdadams/jasper/index.html. 2004

[34] FFMPEG. http://ffmpeg.sourceforge.net/index.org2.html. 2004

[35] W. –C. Chen, J. –Y. Bouguet, M. Chu, and R. Grzeszczuk. Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Field. In Proceedings of ACM SIGGRAPH 2002, 447-456, 2002