Illustrative Volume Visualization with Style Transfer Functions

for Hewlett-Packard Scalable Visualization Array

October, 2008

Budapest University of Technology and Economics

Department of Control Engineering and Information Technology

Contents

1 Introduction **3**

2 Parallel Implementation **4**
2.1 HP Parallel Compositing Library 4

3 Installation and Usage **6**
3.1 Installation 6
3.1.1 Library dependencies 6
3.1.2 RPM Package 7
3.1.3 Building from Sources 7
3.2 Usage 7
3.2.1 SVA Startup Script 7
3.2.2 User Interface 8
3.2.3 Test Volumes 8
3.2.4 Configuration File 9
3.2.5 Volume Descriptor File 9

4 Results **11**

Chapter 1

Introduction

Illustrative volume visualization frequently employs non-photorealistic rendering techniques to enhance important features or to suppress unwanted details. Classical approaches use transfer functions to just assign colors and opacities values to density values, while style transfer functions allow us to combine a multitude of different shading styles in a single rendering. In this method an image-based lighting model uses lit sphere maps to represent non-photorealistic rendering styles. An arbitrari redering style can be stored using an image of a sphere shaded in the desired style. The basic idea is to capture color variations of an object as a function of normal direction. As a sphere provides coverage of the complete set of unit normals, an image of a sphere under orthographic projection will capture all such variations on one hemisphere. This image is then used as a sphere map indexed by the eye-space normals to shade an object. The style transfer function lookup table contains references to these styles instead of colors.

From a user's point of view, the transfer function now not only specifies the color over the range of data values, but also the shading as a function of eye-space normal direction. The complexity of specifying a transfer function, however, is not increased. Instead of assigning a single color to a certain value range, a pre-defined shading style represented by a lit sphere map is chosen. In this context, one advantage of sphere maps as opposed to other mappings is that they can be directly presented to the user as an intuitive preview image for the style. Style transfer functions allow for a flexible combination of different shading styles in a single transfer function.

Reference: Stefan Bruckner, Meister Eduard Grller : "Style Transfer Functions for Illustrative Volume Rendering". Computer Graphics Forum, 26(3):715-724, September 2007.

Chapter 2

Parallel Implementation

2.1 HP Parallel Compositing Library

The *HP Parallel Compositing Library (ParaComp)* is a *sort-last parallel compositing* API suitable for *hybrid object-space screen-space decomposition*. The API was originally developed by Computational Engineering International (CEI) to make its products run efficiently in a distributed environments. The latest version is based on the abstract Parallel Image Compositing API (PICA) designed by Lawrence Livermore National Lab, HP, and Chromium team.

ParaComp is a *message passing* library for graphics clusters enabling users to take advantage of the performance scalability of clusters with network-based pixel compositing without understanding its inner structure and operation. The library makes it possible for multiple graphics nodes in a cluster to collectively produce images, thus significantly larger data sets can be processed and larger images can be created than on any individual graphics hardware by distributing the load over multiple nodes.

However, there is no explicit data distribution so no load balancing is done by the API. The philosophy of the designers is keeping the API as thin as possible. Therefore, only a *global frame* is defined and one or more nodes can contribute pixels to this frame and one or more nodes can receive a specified subset of the frame. ParaComp controls the operation of the nodes based on their request; it takes the results of their renderings and generates the needed composited images (see Figure 2.1). According to the nomenclature of the API a sub-image contribution is called *framelet* and the received image area is called the *output*. These framelets and the outputs can overlap each other without any restriction to their origin or destination nodes. The attributes of a framelet are the following:

- **horizontal and vertical position** in the global frame;
- **width and height** of the framelet in pixels;
- the **data source** which can be both the system memory and the frame buffer; and
- the **depth order** of the framelets which is needed by non-commutative compositing operators like alpha blending.

The size of the output does not necessarily equal the size of the global frame. For example, each tile can be connected to a separate node in a multi-tile display. The attributes of an output are:

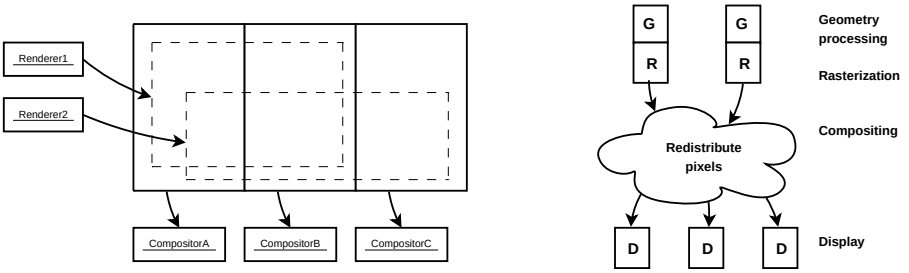

Figure 2.1: The operation of the *HP Parallel Compositing Library*

- **horizontal and vertical position** in the global frame;
- **width and height** of the output in pixels; and
- the **pixel data** to be returned (RGB, RGBA, RGBA+depth).

For details see the official documentation of the HP Parallel Compositing Library [1].

Chapter 3

Installation and Usage of the Illustrative Volume Visualization application

This program is the SVA implementation of the style transfer function algorithm. It uses NVidia's Cg toolkit for rendering and HP's Paracomp for compositing.

3.1 Installation

Both source and prebuilt versions of the application and the library can be found on the web site of the project[1].

3.1.1 Library dependencies

The following libraries are required by the volume rendering application:

- **paracomp**: Hewlett Packard implementation of the Parallel Compositing API (version 1.0-beta1 or later)
- **devil**: Developer's Image Library (version 1.6.7)
- **glew**: OpenGL Extension Wrangler library (version 1.3.4 or later)
- **Cg** and **CgGL** : NVIDIA Cg library
- **gl**: library implementing OpenGL API
- **glu**: OpenGL Utility Library
- **glut**: OpenGL Utility Toolkit

There are prebuilt packages for HP XC V3.2 RC1 platform for AMD64 architecture on the web site of the project for Developer's Image Library, OpenGL Extension Wrangler, Cg and CgGL libraries. If one of them is missing from the target system, it can be installed in the usual way using the `rpm` package manager program:

[1] `http://amon.ik.bme.hu/styletransfer/`

```
# rpm -i devil-1.6.7-1.x86_64.rpm
# rpm -i devil-devel-1.6.7-1.x86_64.rpm
# rpm -i glew-1.3.4-1.x86_64.rpm
# rpm -i glew-devel-1.3.4-1.x86_64.rpm
# rpm -i Cg-1.5.x86_64.rpm
```

The `XXX-devel-YYY.rpm` packages are only needed when the visualization application is built from sources. Otherwise, only the shared libraries are to be installed.

The other libraries like the Parallel Compositing library, the standard C/C++ libraries, and the OpenGL libraries are platform specific and have to be installed based on the actual software stack.

3.1.2 RPM Package

The 3D texture volume renderer (`texturevr`) can be also installed from a prebuilt RPM[2] package in the same way:

```
# rpm -i styletransfer-0.1-1.x86_64.rpm
```

3.1.3 Building from Sources

The build system of the volume rendering application application is based on CMake. So, it can be built with the usual procedure:

```
$ cmake .
$ make
$ sudo make install
```

However, please note that the volume renderer uses OpenGL extensions, therefore OpenGL Extension Wrangler support should not be disabled. Please also note that the application has a graphical user interface that requires font rendering, so Developer's Image Library is also needed. Nevertheless, FreeType support can be disabled if necessary, since the fonts are read from precalculated image files.

3.2 Usage

3.2.1 SVA Startup Script

A SLURM[3] startup script is provided to use `styletransfer` for parallel rendering. It can be invoked with the following command:

```
$ style.sh -r <renderers> -cf <descriptor-file>
```

[2] Red Hat Package Manager

[3] SLURM is an abbreviation for Simple Linux Utility for Resource Management. It is an open-source resource manager designed for Linux clusters of all sizes. This software solution is used for HP-XC clusters.

The startup script has two parameters that should be set. The first one (`-r`) tells SLURM the number of *additional render nodes* to be allocated. The later one (`-cf`) sets the volume descriptor file. The config file describes the path of the volume file to load, and the texture file names used as styles during rendering. If ”-r n” is not given 2 rendering nodes will be set up. If ”-cf file” is not given the file ”default.config” will be choosen, which loads a head volume set (”head128.volume”) and three style textures (”style1.bmp”, ”style2.bmp” and ”style4.bmp”). (The given image files should have the same resolution and pixel format.)

3.2.2 User Interface

The virtual camera can be rotated around the examined volume by holding the left mouse button and moving the mouse. The camera can be moved closer or further with the ”W” and ”S” keys. Pressing SPACE will show or hide the transfer curves.

The transfer curves can be adjusted with the mouse. Clicking on them will select the closest control point or create a new control point if no point exists nearby. Selected control points can me moved by holding the left mouse button and dragging the mouse. Selected control points can be deleted with the right mouse button.

The actual transfer curves can be saved into a file by pressing ”P”, and the last saved curves can be loaded with the ”L” key.

Meaning of the transfer curves:

- The upper curve is the style transfer curve: The x axis represents the density values and the y axis represents the style indices (0 is for the first style given in the config file and 1 is for the last style image). (This curve should be set once for the given data set and styles.)
- The second curve is a classical opacity curve: x axis stands for densitiy and y for opacity values. (This curve is typically changed during examination.)

3.2.3 Test Volumes

We used the following data sets for presenting the application. *The Visible Human Data Set* can be downloaded from the web site of the U.S. National Library of Medicine[4]. The *astrophysical data set* was created at the McMaster University[5]. The *hydrodynamical data set* was provided by the Hewlett-Packard. The other data sets are classical volume rendering testing data sets. The *present*, the *Christmas tree*, and the *stag beetle* data sets were created at the Vienna University of Technology[6]. The preprocessed version of these data sets that can be visualized with this volume rendering application can be downloaded from our data server[7].

[4] `http://www.nlm.nih.gov/research/visible/visible_human.html`
[5] `http://www.mcmaster.ca/`
[6] `http://www.cg.tuwien.ac.at/`
[7] `http://visdata.ik.bme.hu/`

```
kopf.volume
style1.bmp
style2.bmp
style3.bmp
style4.bmp
```

Listing 3.1: Sample Config File

3.2.4 Configuration File

The config file describes the path of the volume file to load, and the texture file names used as styles during rendering.

3.2.5 Volume Descriptor File

The volume descriptor file has two main sections. In the first one, there are name-value pairs for setting different parameters like resolution, physical size, and voxel type. In the second part the data files are listed in a sequence. The list of parameters is the following:

- `width`, `height`, and `depth` describe the dimensions of the volumetric data, i.e. the number of voxels in each dimension,
- `voxeltype` specifies the data type of the volumetric data. Currently the following values are accepted:
  - `unsigned-char` sets byte/voxel data type,
  - `unsigned-short` sets word/voxel data type,
  - `float-msb` sets IEEE 754 float/voxel data type;
- `sizex`, `sizey`, and `sizez` sets the sizes of the bounding box.

See Listing 3.2 for a sample descriptor file. The volume descriptor files for the Visible Human and the McMaster University's data sets can be also downloaded from our data server.

```
# Resolution
width=256
height=256
depth=159

# Physical size
sizex=1
sizey=1
sizez=0.621

# Voxel type
voxeltype=unsigned-char

# Data files
kopf
```

Listing 3.2: Sample Volume Descriptor File)

Chapter 4

Results

For our experiments we used a *Hewlett-Packard's Scalable Visualization Array* consisting of five computing nodes. Each node has a dual-core AMD Opteron 246 processor, an nVidia 8800GTX graphics controller, and an InfiniBand network adapter. One node is only responsible for compositing and managing the framelet generations and does not take part in the rendering processes, so we could divide our data set into maximum four parts.

Table 4.1 shows our results. It can be seen that using 2 nodes gives better performance than the single computer version. Using more than 2 nodes becomes useful only in case of larger data sets, for small data sets the communication between the nodes becomes a bottleneck. The N/A sign means that a $256 \times 256 \times 256$ data set cannot be simulated on a single computer since it has too high memory needs.

It is worth examining the resolutions of $64 \times 64 \times 64$ and $80 \times 80 \times 80$. The 80 resolution needs about twice as many voxels as the 64 one. It can be clearly seen that the performance of the two node implementation is the double of the performance obtained on a single node. Similarly the two node implementation nearly doubles performance in almost all cases.

Table 4.1: Performance results.

Bibliography

[1] HEWLETT PACKARD. *HP Scalable Visualization Array Parallel Compositing Library Reference Guide*, 2007.