## Revision Record

<table>
<thead>
<tr>
<th>Version number</th>
<th>Revised date</th>
<th>Revised chapter</th>
<th>Revised content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.04-1</td>
<td>2015.3.31</td>
<td>-</td>
<td>Release</td>
</tr>
</tbody>
</table>
## Contents

1 Introduction ......................................................................................................................... 4  
  1.1. Overview ...................................................................................................................... 4  
  1.2. System Environment ..................................................................................................... 5  

2 Installation ............................................................................................................................ 6  
  2.1. Filter ................................................................................................................................ 6  
    2.1.1. Installation of Load Module ..................................................................................... 6  
    2.1.2. Installation from Source Code .................................................................................. 7  
  2.2. Server ............................................................................................................................... 7  
    2.2.1. Installation of Load Module ....................................................................................... 7  
    2.2.2. Installation from Source Code ................................................................................... 8  
    2.2.3. Windows ..................................................................................................................... 9  
  2.3. Client ................................................................................................................................ 9  
    2.3.1. Installation of Load Module ....................................................................................... 10  
    2.3.2. Installation from Source Code ................................................................................... 10  
    2.3.3. Linux, Mac ................................................................................................................... 10  
    2.3.4. Windows .................................................................................................................... 10  

3 Filter ....................................................................................................................................... 12  
  3.1. Overview ......................................................................................................................... 12  
  3.2. Data decomposition model ............................................................................................. 12  
  3.3. Launch Filter ................................................................................................................... 15  
  3.4. File Format ...................................................................................................................... 15  
    3.4.1. Input Data Format ....................................................................................................... 16  
    3.4.2. Endian ........................................................................................................................ 16  
    3.4.3. Filter Output Information File (.pfi) ......................................................................... 17  
    3.4.4. SPLIT File Format ...................................................................................................... 19  
    3.4.5. Sub-volume Aggregate Format ................................................................................. 22  
    3.4.6. Step Aggregate Format ............................................................................................. 25  
  3.5. Parameter File .................................................................................................................. 29  
  3.6. MPI Parallel Processing .................................................................................................... 30  
  3.7. Execution Method in Staging Environment ..................................................................... 31  
    3.7.1. Execution Shell Script and Parameter File ................................................................. 32  
    3.7.2. Input-output Files and Directories ............................................................................ 33  
  3.8. Unstructured Grid data with Mixed Elements ................................................................ 34  

4 Server ..................................................................................................................................... 36
4.1. Overview ......................................................................................................................... 36
4.2. Launch Server .................................................................................................................. 36
  4.2.1. Launch in Batch Mode ............................................................................................... 37
  4.2.2. Launch in Client-Server Mode .................................................................................. 38
  4.2.3. Connecting Client and Server by Socket Communication ........................................ 38

5 Client ........................................................................................................................................... 44
  5.1. Overview ......................................................................................................................... 44
  5.2. Launching Client .............................................................................................................. 44
  5.3. Terminating PBVR .......................................................................................................... 46
    5.3.1. Standard Termination ................................................................................................. 46
    5.3.2. Forced Termination ..................................................................................................... 46
  5.4. Operating Instructions ...................................................................................................... 47
    5.4.1. Viewer ............................................................................................................................. 47
    5.4.2. Main Panel ..................................................................................................................... 48
    5.4.3. CROP Panel ................................................................................................................... 50
    5.4.4. Transfer Function Editor ............................................................................................... 52
    5.4.5. Time panel ..................................................................................................................... 62
    5.4.6. Animation panel ........................................................................................................... 63

6 Examples Using Sample Data ................................................................................................. 64
  6.1. Filter Processing ............................................................................................................... 64
  6.2. Starting PBVR .................................................................................................................. 65
  6.3. Designing Transfer Function .......................................................................................... 67
    6.3.1. Volume Rendering of Single Variable .......................................................................... 67
    6.3.2. Multivariate Volume Rendering .................................................................................. 68
    6.3.3. Slice Plane .................................................................................................................... 68
    6.3.4. Synthesis of Transfer Functions ................................................................................... 70
  6.4. Preserving Result .............................................................................................................. 71
  6.5. Example of Batch Mode .................................................................................................. 72
1 Introduction

1.1. Overview

This manual describes a remote visualization system PBVR, developed at the Center for Computational Science & e-Systems, Japan Atomic Energy Agency. The system realizes high-speed remote visualization of large-scale volume data using the KVS library and the Particle Based Volume Rendering (PBVR) technique developed at the Koyamada Visualization Laboratory, Kyoto University. The system consists of the following three components.

(1) Filter

Filter reads the original volume data and divides it into sub-volumes which are processing units for parallel visualization.

(2) PBVR Server

PBVR Server reads sub-volumes for the parallel visualization, and generates particle data based on the PBVR technique.

(3) PBVR Client

PBVR Client receives the particle data, and renders it to display images using Open GL.

Figure 1 The whole configuration of the visualization application system
1.2. System Environment

The system has been verified on the following environments.

- **Filter**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Compiler</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit</td>
<td>gcc version 4.4.6</td>
<td></td>
</tr>
<tr>
<td>Mac 64bit※2</td>
<td>gcc version 4.8.2</td>
<td></td>
</tr>
<tr>
<td>Windows7 64bit</td>
<td>Visual Studio 2013</td>
<td></td>
</tr>
<tr>
<td>K Computer</td>
<td>Fujitsu Compiler</td>
<td></td>
</tr>
<tr>
<td>FX10</td>
<td>Fujitsu Compiler</td>
<td></td>
</tr>
<tr>
<td>BX900</td>
<td>Fujitsu Compiler</td>
<td></td>
</tr>
</tbody>
</table>

- **Server**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Compiler</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit</td>
<td>g++ version 4.4.6</td>
<td>kmath, ntl※1</td>
</tr>
<tr>
<td>Mac 64bit※2</td>
<td>g++ version 4.8.2</td>
<td></td>
</tr>
<tr>
<td>Windows7 64bit</td>
<td>Visual Studio 2013</td>
<td></td>
</tr>
<tr>
<td>K Computer</td>
<td>Fujitsu Compiler</td>
<td>kmath, ntl※1</td>
</tr>
<tr>
<td>FX10</td>
<td>Fujitsu Compiler</td>
<td>kmath, ntl※1</td>
</tr>
<tr>
<td>BX900</td>
<td>Fujitsu Compiler</td>
<td>kmath, ntl※1</td>
</tr>
</tbody>
</table>

- **Client**

<table>
<thead>
<tr>
<th>Platform</th>
<th>Compiler</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit</td>
<td>g++ version 4.4.6</td>
<td>OpenGL, GLUT</td>
</tr>
<tr>
<td>Mac 64bit※2</td>
<td>g++ version 4.8.2</td>
<td>OpenGL, GLUT</td>
</tr>
<tr>
<td>Windows7 64bit</td>
<td>Visual Studio 2013</td>
<td>OpenGL, GLUT</td>
</tr>
</tbody>
</table>

※1. kmath is a massively parallel random number library developed at the RIKEN, Advanced Institute for Computational Science. kmath uses the ntl library.

※2. On Mac, OpenMP is not available on the default version of gcc installed with Xcode. To use load modules or to compile source codes, newer versions of gcc compilers with OpenMP are needed. This binary was compiled with gcc48 installed through Macports. gcc can be installed through Macports as follows.

- Sport install gcc48
- Sport select –set gcc mp-gcc48
2 Installation

PBVR is provided as a load module package, load_module_v1.04.tgz, and a source code package, pbvr_v1.04.tgz. The following sections show how to install Filter, Server, and Client.

2.1 Filter

The filter program is implemented in C, and a MPI+OpenMP version for massively parallel computing and an OpenMP version for thread parallel computing are available.

2.1.1 Installation of Load Module

The following table shows a list of load modules stored in the filter directory of the load module package. To complete the installation, copy the load modules in each environment to an appropriate directory where PATH is set up.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Parallelization</th>
<th>Name of load module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit</td>
<td>OpenMP</td>
<td>filter_Linux_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>filter_Linux_mpi_omp</td>
</tr>
<tr>
<td>Mac 64bit</td>
<td>OpenMP</td>
<td>filter_mac</td>
</tr>
<tr>
<td>Windows7 64bit</td>
<td>OpenMP</td>
<td>filter_Win_omp.exe</td>
</tr>
<tr>
<td>K Computer※1</td>
<td>OpenMP</td>
<td>filter_K_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>filter_K_mpi_omp</td>
</tr>
<tr>
<td>FX10※1</td>
<td>OpenMP</td>
<td>filter_FX10_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>filter_FX10_mpi_omp</td>
</tr>
<tr>
<td>BX900※1</td>
<td>OpenMP</td>
<td>filter_BX_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>filter_BX_mpi_omp</td>
</tr>
</tbody>
</table>

※1. The load modules for supercomputers are used only on computing nodes. The load modules for Linux should be used on login nodes and post-processing nodes consisting of Linux servers.
2.1.2. Installation from Source Code

Extract a source code package, filter.tgz, to an arbitrary directory and compile it using Makefile listed below. To complete the installation, copy generated load modules to an appropriate directory where PATH is set up.

Table 2  List of Makefiles for the filter program

<table>
<thead>
<tr>
<th>Platform</th>
<th>Parallelization</th>
<th>Name of Makefile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit</td>
<td>OpenMP</td>
<td>Makefile.linux_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>Makefile.linux_mpi_omp</td>
</tr>
<tr>
<td>Mac 64bit</td>
<td>OpenMP</td>
<td>Makefile.mac</td>
</tr>
<tr>
<td>Windows7 64bit※1</td>
<td>OpenMP</td>
<td>Makefile.win</td>
</tr>
<tr>
<td>K Computer※2</td>
<td>OpenMP</td>
<td>Makefile.omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>Makefile.mpi_omp</td>
</tr>
<tr>
<td>FX10※2</td>
<td>OpenMP</td>
<td>Makefile.fx10_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>Makefile.fx10_mpi_omp</td>
</tr>
<tr>
<td>BX900※2</td>
<td>OpenMP</td>
<td>Makefile.bx_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>Makefile.bx_mpi_omp</td>
</tr>
</tbody>
</table>

※1. On the Windows environment, launch the VS2013 x64 Native Tools command prompt to use nmake.
※2. Makefiles for supercomputers are designed for cross-compilers on login nodes.

2.2. Server

The server program is implemented in C++, and a MPI+OpenMP version for massively parallel computing and an OpenMP version for thread parallel computing are available.

2.2.1. Installation of Load Module

The following table shows a list of load modules stored in the sever directory of the load module package. To complete the installation, copy the load modules in each environment to an appropriate directory where PATH is set up.

Table 3  List of load modules for the server program

<table>
<thead>
<tr>
<th>Platform</th>
<th>Parallelization</th>
<th>Name of load module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit</td>
<td>OpenMP</td>
<td>CPUServer_linux_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>CPUServer_linux_mpi_omp</td>
</tr>
</tbody>
</table>
### 2.2.2. Installation from Source Code

Extract a source code package, server.tgz, to an arbitrary directory and compile it using Makefile listed below. To complete the installation, copy generated load modules to an appropriate directory where PATH is set up.

#### Table 4  List of Makefiles for the server program

<table>
<thead>
<tr>
<th>Platform</th>
<th>Parallelization</th>
<th>Name of Makefile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit※1</td>
<td>OpenMP</td>
<td>makefile.linux_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>makefile.linux_mpi_omp</td>
</tr>
<tr>
<td>Mac 64bit※3</td>
<td>OpenMP</td>
<td>makefile.mac</td>
</tr>
<tr>
<td>K Computer※2</td>
<td>OpenMP</td>
<td>makefile.k_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>makefile.k_mpi_omp</td>
</tr>
<tr>
<td>FX10※2</td>
<td>OpenMP</td>
<td>makefile.fx10_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>makefile.fx10_mpi_omp</td>
</tr>
<tr>
<td>BX900※2</td>
<td>OpenMP</td>
<td>makefile.bx_omp</td>
</tr>
<tr>
<td></td>
<td>MPI+OpenMP</td>
<td>makefile.bx_mpi_omp</td>
</tr>
</tbody>
</table>

※1. kmath provided in the Linux environment is compiled using OpenMPI. To use other MPI libraries, recompile kmath as follows.

[Step 1]
Modify the following file depending on the system environment.

```
./server/kmath_linux/random/Makefile.machine
```

[Step 2]

```
$cd ./server/kmath_linux/random/c++
$make
```
When the compilation of numerical libraries, ntl and kmath, fails, modify the following lines in Makefile in order to compile the server program without using the numerical libraries.

Delete ‘-DKMATH’ from ‘CXXFLAGS=’
Delete ‘-lkm_random -lntl’ from ‘LDFLAGS=’

Note that, in this case, the independence of pseudo random numbers is not guaranteed in the MPI+OpenMP version. Therefore, the use of kmath is recommended on the MPI+OpenMP version.

※2. Makefiles for supercomputers are designed for cross-compilers on login nodes.
※3. The Mac version does not use numerical libraries, ntl and kmath.

2.2.3. Windows

The Windows version does not use the numerical libraries, ntl and kmath, and it is compiled as follows.
(1) Extract the server environment on a machine with the Visual Studio 2013 environment.
(2) On the server directory, run ‘\project_vc\PBVR_Server_win.sln’ and launch Visual Studio 2013.
(3) Choose Release x64 as a configuration (see Figure 2).

![Figure 2](sample.png)

Figure 2  Sample configuration of Visual Studio 2013
(4) Go to Visual Studio 2013 Menu [Build] ⇒ Run [Build solution].
(5) A load module, PBVR_Server_win.exe, is created under ‘\project_vc\x64\Release’.

2.3. Client

The client program is implemented in C++ and OpenGL.
2.3.1. Installation of Load Module

The following table shows a list of load modules stored in the client directory of the load module package. To complete the installation, copy the load modules in each environment to an appropriate directory where PATH is set up.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Parallelization</th>
<th>Name of load module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux 64bit</td>
<td>OpenMP</td>
<td>PBVRViewer_linux</td>
</tr>
<tr>
<td>Mac 64bit</td>
<td>OpenMP</td>
<td>PBVRViewer_mac</td>
</tr>
<tr>
<td>Windows7 64bit±1</td>
<td>OpenMP</td>
<td>PBVRViewer_win.exe</td>
</tr>
</tbody>
</table>

±1. For the Windows version, copy ‘glut32.dll’ to the same directory.

2.3.2. Installation from Source Code

Extract client.tgz to an arbitrary directory and compile it using a compilation script. To complete the installation, copy generated load modules to an appropriate directory where PATH is set up.

2.3.3. Linux, Mac

Execute a compilation script as follows.

$ cd ./client/PBVRViewer
$ sh build_cpu.sh

±1. To compile all the sources without using past object files, use ‘rebuild_cpu.sh’.

2.3.4. Windows

(1) Installation of glut (64bit)

1. Download ‘glut-3.7.6-bin_x64.zip (64bit)’ from the link below.
   http://ktm11.eng.shizuoka.ac.jp/lesson/modeling.html
2. Extract 1.
   glut.h
   glut32.lib
   glut32.dll

(2) Compilation of glui-2.36

1. Go to ‘glui-2.36\src\msvc’.
Choose Release x64 for a configuration as Figure 2.

Go to Solution Explore and right click on ‘glui’, open property [configuration property] →[C/C++], and add a path of ‘glut.h’ to the additional include directory.

Go to Solution Explore and right click on ‘glui’, and execute build.

(3) Compilation of ‘KVS-src-1.1.1’

Go to ‘KVS-src-1.1.1’ and copy ‘build_win.bat’ with a different name.

Change ‘kvs.conf’ as follows.

KVS_SUPPORT_GLEW = 1 ⇒ KVS_SUPPORT_GLEW = 0

Specify ‘[Directory storing PBVRViewer.sln]¥kvs’ for ‘KVS_DIR’ in the bat file in ①.

Specify ‘[Directory storing glut.lib]’ for ‘KVS_GLUT_DIR’ in the bat file in ①.

Go to [start menu]→[Visual Studio 2013]→[Visual Studio Tool] and open “VS2013 x64 Native Tools command prompt.”

Go to ‘KVS-src-1.1.1’ and execute the bat file created in ①③④.

(4) Compilation of PBVR Client

Choose Release x64 for a configuration.

Go to Solution Explore and right click on PBVRViewer, open property [configuration property] →[C/C++], and add an include path of ‘glut’ to “additional include directory.”

Go to Solution Explore and right click on ‘PBVRViewer’, open property [configuration property] →[linker], and add a ‘lib’ path of ‘glut’ to “additional library directory.”

Go to [menu] → [solution build] to execute the compilation

A load module, PBVRViewer.exe, is created under ‘¥¥x64¥Release¥’.

When execute ⑤, place ‘glut32.dll’ in the same directory.
3 Filter

3.1. Overview

The filter program is an independent preprocessing program from the PBVR system. The program divides time-series volume data for parallel processing of a PBVR server, and generates sub-volume data for visualization. The data decomposition is based on the octree model, and the program divides structured grids data and unstructured grids data into arbitrary octree regions to generate input files for parallel processing of the PBVR Server.

3.2. Data decomposition model

As shown in Figure 3, the octree structure is generated by dividing each side of a cuboid in half and generating eight sub-cuboids. By repeating the division procedure recursively, the volume is segmented. Each cuboid has eight child cuboids while child cuboids have a single parent cuboid.
As shown in Figure 4, the boundaries of the child-cuboids can be computed by dividing the sum of the minimum and maximum values in each coordinate by two. The inclusion relation of each vertex can be determined by comparing the coordinates of the vertex and the boundaries.
Figure 4  Coordinate values of the boundaries in the octree structure
3.3. Launch Filter

By specifying a parameter file name in the command line option, the filter program is executed with parameters described in the parameter file. When no parameter file name or a non-existent file name is specified, the execution fails.

(launch in MPI+OpenMP processing with N processes)
$ mpiexec -n N ./filter param.txt

(launch only in OpenMP processing)
$ ./filter param.txt

※1. In both processing modes, the number of OpenMP threads is set in the environmental variable ‘OMP_NUM_THREADS’.
※2. In Windows, the command is launched from VS2013 x64 Native Tools command prompt.

3.4. File Format

This section describes file formats used in the filter program. All binary format data in the output files is given in single precision, no header/footer, and little endian. Three file formats are available including SPLIT format (KVSML format), sub-volume aggregate format, and step aggregate format (see Figure 5). The SPLIT format generates independent files for each step and sub-volume. However, in this format, the number of files grows explosively as the layer of octree. To resolve this problem, the sub-volume aggregate format, which aggregates files at different time step (and at the same sub-volume) in a single file, and the step aggregate format, which aggregates files at different sub-volume (and at the same time step) in a single file, are proposed. The following sections describe their file formats in details.
3.4.1. Input Data Format
The filter program can process the following input data.

(1) AVSFLD binary data
(2) AVSUCD binary data※1, 2
  ※1. AVSUCD binary data with ‘data’ format can be used. ‘geom format’ and ‘data_geom format’ are not supported.
  ※2. 2D and 3D elements in Table 8 and their mixed elements are supported.

For more details of AVS data formats, refer to the following link (Japanese) or the AVS manual.
http://www.cybernet.co.jp/avs/products/avsexpress/dataformat.html

3.4.2. Endian
Binary files used in the filter program are unified as little endian. On a machine employing big endian, visualization data needs to be produced in little endian or to be converted to little endian in advance before executing the filter program.
### 3.4.3. Filter Output Information File (.pfi)

This file contains binary data summarizing the information about the volume data.

<table>
<thead>
<tr>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of node</td>
<td>int</td>
</tr>
<tr>
<td>Total number of element</td>
<td>int</td>
</tr>
<tr>
<td>Element type</td>
<td>int ※1</td>
</tr>
<tr>
<td>File type</td>
<td>int ※2</td>
</tr>
<tr>
<td>Number of file</td>
<td>int ※3</td>
</tr>
<tr>
<td>Number of component</td>
<td>int</td>
</tr>
<tr>
<td>Start step</td>
<td>int</td>
</tr>
<tr>
<td>End step</td>
<td>int</td>
</tr>
<tr>
<td>Number of sub-volume</td>
<td>int ※4</td>
</tr>
<tr>
<td>Whole 3 dimensional space X-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Whole 3 dimensional space Y-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Whole 3 dimensional space Z-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Whole 3 dimensional space X-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Whole 3 dimensional space Y-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Whole 3 dimensional space Z-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 1 node number</td>
<td>int</td>
</tr>
<tr>
<td>Sub-volume 2 node number</td>
<td>int</td>
</tr>
<tr>
<td>Sub-volume 3 node number</td>
<td>int</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Sub-volume n node number</td>
<td>int</td>
</tr>
<tr>
<td>Sub-volume 1 element number</td>
<td>int</td>
</tr>
<tr>
<td>Sub-volume 2 element number</td>
<td>int</td>
</tr>
<tr>
<td>Sub-volume 3 element number</td>
<td>int</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Sub-volume n element number</td>
<td>int</td>
</tr>
<tr>
<td>Sub-volume 1 X-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 1 Y-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 1 Z-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 1 X-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 1 Y-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 1 Z-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 2 X-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 2 Y-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 2 Z-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 2 X-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume 2 Y-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Sub-volume n X-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume n Y-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume n Z-axis minimum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume n X-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume n Y-axis maximum value</td>
<td>float</td>
</tr>
<tr>
<td>Sub-volume n Z-axis maximum value</td>
<td>float</td>
</tr>
</tbody>
</table>
Step 1 variable 1 minimum value
Step 1 variable 1 maximum value
Step 1 variable 2 minimum value
Step 1 variable 2 maximum value
: 
Step 1 variable N minimum value
Step 1 variable N maximum value
: 
Step m variable 1 minimum value
Step m variable 1 maximum value
Step m variable 2 minimum value
Step m variable 2 maximum value
: 
Step m variable N minimum value
Step m variable N maximum value

※1. Refer to Table 8 for the definition of element types.
※2. The following three formats are available.
   • 0 : SPLIT format
   • 1 : Sub-volume aggregate format
   • 2 : Step aggregate format
※3. When a file type is the sub-volume aggregate format, specify the total number of files.
※4. The number of sub-volumes is expressed as ‘$8^{\text{n\_layer}}$’.
   • n\_layer=0 : 1
   • n\_layer=1 : 8
   • n\_layer=2 : 64
   • n\_layer=3 : 512
   • n\_layer=4 : 4,096
   • n\_layer=5 : 32,768
   • n\_layer=6 : 262,144
   • n\_layer=7 : 2,097,152
3.4.4. SPLIT File Format

An element configuration file and a node coordinate file are produced for each sub-volume. A ‘kvsml’ file and a variable file are generated for each sub-volume and step. Accordingly, the total number of files is calculated from the following formula.

The number of sub-volume × 2 + the number of sub-volume × the number of step × 2

(Example: n_layer=7 & step=100 : 423,624,704)

3.4.4.1 File Name

prefix_XXXXX_YYYYYYY_ZZZZZZZ.kvsml …………… kvsml file (ASCII format)
prefix_YYYYYYY_ZZZZZZZ_connect.dat ……… element configuration file (binary format)
prefix_YYYYYYY_ZZZZZZZ_coord.dat …………… node coordinate file (binary format)
prefix_XXXXX_YYYYYYY_ZZZZZZZ_value.dat ……… variable file (binary format)

XXXXX : Number of steps (5 digit number)
YYYYYYY : Sub-volume number (7 digit number)
ZZZZZZZ : Total number of Sub-volumes (7 digit number)

3.4.4.2 Kvsml File Format

```xml
<?xml version="1.0" ?>
<KVSML>
  <Object type="UnstructuredVolumeObject">
    <UnstructuredVolumeObject cell_type="type of elements">
      <Node nnodes="number of nodes in the sub-volume">
        <Value veclen="number of variables">
          <DataArray type="float" file="prefix_XXXXX_YYYYYYY_ZZZZZZZ_value.dat" format="binary" />
        </Value>
        <Coord>
          <DataArray type="float" file="prefix_YYYYYYY_ZZZZZZZ_coord.dat" format="binary" />
        </Coord>
      </Node>
      <Cell ncells="number of elements in the sub-volume">
        <Connection>
          <DataArray type="uint" file="prefix_YYYYYYY_ZZZZZZZ_connect.dat" format="binary" />
        </Connection>
      </Cell>
    </UnstructuredVolumeObject>
  </Object>
</KVSML>
```
### 3.4.4.3 Element Configuration File Format

<table>
<thead>
<tr>
<th>Element 1 configuration node 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element 1 configuration node 2</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Element 1 configuration node n</td>
</tr>
<tr>
<td>Element 2 configuration node 1</td>
</tr>
<tr>
<td>Element 2 configuration node 2</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Element 2 configuration node n</td>
</tr>
<tr>
<td>Element 3 configuration node 1</td>
</tr>
<tr>
<td>Element 3 configuration node 2</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Element 3 configuration node n</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Element N configuration node 1</td>
</tr>
<tr>
<td>Element N configuration node 2</td>
</tr>
<tr>
<td>:</td>
</tr>
<tr>
<td>Element N configuration node n</td>
</tr>
</tbody>
</table>

### 3.4.4.4 Node Coordinate File Format

<table>
<thead>
<tr>
<th>Node 1</th>
<th>X coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>Y coordinate</td>
</tr>
<tr>
<td>Node 1</td>
<td>Z coordinate</td>
</tr>
<tr>
<td>Node 2</td>
<td>X coordinate</td>
</tr>
<tr>
<td>Node 2</td>
<td>Y coordinate</td>
</tr>
<tr>
<td>Node 2</td>
<td>Z coordinate</td>
</tr>
<tr>
<td>Node 3</td>
<td>X coordinate</td>
</tr>
<tr>
<td>Node 3</td>
<td>Y coordinate</td>
</tr>
<tr>
<td>Node 3</td>
<td>Z coordinate</td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td></td>
</tr>
<tr>
<td>Node m</td>
<td>X coordinate</td>
</tr>
<tr>
<td>Node m</td>
<td>Y coordinate</td>
</tr>
<tr>
<td>Node m</td>
<td>Z coordinate</td>
</tr>
</tbody>
</table>
### 3.4.4.5 Variable File

<table>
<thead>
<tr>
<th>Node 1</th>
<th>Variable 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 2</td>
<td>Variable 1</td>
</tr>
<tr>
<td>Node 3</td>
<td>Variable 1</td>
</tr>
<tr>
<td>⋮</td>
<td></td>
</tr>
<tr>
<td>Node n</td>
<td>Variable 1</td>
</tr>
<tr>
<td>Node 1</td>
<td>Variable 2</td>
</tr>
<tr>
<td>Node 2</td>
<td>Variable 2</td>
</tr>
<tr>
<td>Node 3</td>
<td>Variable 2</td>
</tr>
<tr>
<td>⋮</td>
<td></td>
</tr>
<tr>
<td>Node n</td>
<td>Variable 2</td>
</tr>
<tr>
<td>Node 1</td>
<td>Variable m</td>
</tr>
<tr>
<td>Node 2</td>
<td>Variable m</td>
</tr>
<tr>
<td>Node 3</td>
<td>Variable m</td>
</tr>
<tr>
<td>⋮</td>
<td></td>
</tr>
<tr>
<td>Node n</td>
<td>Variable m</td>
</tr>
</tbody>
</table>
3.4.5. Sub-volume Aggregate Format

Element configuration, node coordinates, and variables of all time steps are divided into sub-volumes. By specifying ‘Number of file’, one can aggregate the information of several sub-volumes into arbitrary number of files from one to the number of sub-volumes. (n_layer=7:2,097,152)

3.4.5.1 File Name

prefix_YYYYYY_ZZZZZZZ.dat ……… (Binary format)

<table>
<thead>
<tr>
<th>prefix</th>
<th>: Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYYY</td>
<td>: File number (7 digit number)</td>
</tr>
<tr>
<td>ZZZZZZ</td>
<td>: Total number of files (7 digit number )</td>
</tr>
</tbody>
</table>
### 3.4.5.2 File Format

<table>
<thead>
<tr>
<th>Start sub-volume number</th>
<th>Sub-volume information</th>
</tr>
</thead>
<tbody>
<tr>
<td>End sub-volume number</td>
<td>Element configuration</td>
</tr>
<tr>
<td>Element 1 configuration node 1</td>
<td>Sub-volume 1</td>
</tr>
<tr>
<td>Element 1 configuration node 2</td>
<td></td>
</tr>
<tr>
<td>Element 1 configuration node n</td>
<td></td>
</tr>
<tr>
<td>Element N configuration node 1</td>
<td>Node coordinate</td>
</tr>
<tr>
<td>Element N configuration node 2</td>
<td></td>
</tr>
<tr>
<td>Element N configuration node n</td>
<td></td>
</tr>
<tr>
<td>Node 1 X coordinate</td>
<td></td>
</tr>
<tr>
<td>Node 1 Y coordinate</td>
<td></td>
</tr>
<tr>
<td>Node 1 Z coordinate</td>
<td></td>
</tr>
<tr>
<td>Node m X coordinate</td>
<td></td>
</tr>
<tr>
<td>Node m Y coordinate</td>
<td></td>
</tr>
<tr>
<td>Node m Z coordinate</td>
<td></td>
</tr>
<tr>
<td>Element 1 configuration node 1</td>
<td></td>
</tr>
<tr>
<td>Element 1 configuration node 2</td>
<td></td>
</tr>
<tr>
<td>Element 1 configuration node n</td>
<td></td>
</tr>
<tr>
<td>Element N configuration node 1</td>
<td></td>
</tr>
<tr>
<td>Element N configuration node 2</td>
<td></td>
</tr>
<tr>
<td>Element N configuration node n</td>
<td></td>
</tr>
<tr>
<td>Node 1 X coordinate</td>
<td></td>
</tr>
<tr>
<td>Node 1 Y coordinate</td>
<td></td>
</tr>
<tr>
<td>Node 1 Z coordinate</td>
<td></td>
</tr>
<tr>
<td>Node m X coordinate</td>
<td></td>
</tr>
<tr>
<td>Node m Y coordinate</td>
<td></td>
</tr>
<tr>
<td>Node m Z coordinate</td>
<td></td>
</tr>
</tbody>
</table>
3.4.6. Step Aggregate Format

This format has an element configuration file and a node coordinate file, containing all the sub-volume information. A variable file is produced for each step. The total number of files becomes the number of steps + 2.

3.4.6.1 File Name

prefix_connect.dat  ………Element configuration file (binary format)
prefix_coord.dat  ………Node coordinate file (binary format)
prefix_XXXXX_value.dat…… Variable file (binary format)

prefix  : Prefix
XXXXX  : Number of steps (5 digit number)
### 3.4.6.2 Element Configuration File Format

<table>
<thead>
<tr>
<th>Sub-volume 1 element 1 configuration node 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-volume 1 element 1 configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume 1 element 1 configuration node n</td>
</tr>
<tr>
<td>Sub-volume 1 element 2 configuration node 1</td>
</tr>
<tr>
<td>Sub-volume 1 element 2 configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume 1 element 2 configuration node n</td>
</tr>
<tr>
<td>Sub-volume 1 element 3 configuration node 1</td>
</tr>
<tr>
<td>Sub-volume 1 element 3 configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume 1 element 3 configuration node n</td>
</tr>
<tr>
<td>Sub-volume 1 element N configuration node 1</td>
</tr>
<tr>
<td>Sub-volume 1 element N configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume 1 element N configuration node n</td>
</tr>
<tr>
<td>Sub-volume M element 1 configuration node 1</td>
</tr>
<tr>
<td>Sub-volume M element 1 configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M element 1 configuration node n</td>
</tr>
<tr>
<td>Sub-volume M element 2 configuration node 1</td>
</tr>
<tr>
<td>Sub-volume M element 2 configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M element 2 configuration node n</td>
</tr>
<tr>
<td>Sub-volume M element 3 configuration node 1</td>
</tr>
<tr>
<td>Sub-volume M element 3 configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M element 3 configuration node n</td>
</tr>
<tr>
<td>Sub-volume M element N configuration node 1</td>
</tr>
<tr>
<td>Sub-volume M element N configuration node 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M element N configuration node n</td>
</tr>
</tbody>
</table>
### 3.4.6.3 Node Coordinate File Format

<table>
<thead>
<tr>
<th>Sub-volume 1 node 1 X coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-volume 1 node 1 Y coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node 1 Z coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node 2 X coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node 2 Y coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node 2 Z coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node 3 X coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node 3 Y coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node 3 Z coordinate</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Sub-volume 1 node m X coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node m Y coordinate</td>
</tr>
<tr>
<td>Sub-volume 1 node m Z coordinate</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Sub-volume M node 1 X coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 1 Y coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 1 Z coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 2 X coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 2 Y coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 2 Z coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 3 X coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 3 Y coordinate</td>
</tr>
<tr>
<td>Sub-volume M node 3 Z coordinate</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Sub-volume M node m X coordinate</td>
</tr>
<tr>
<td>Sub-volume M node m Y coordinate</td>
</tr>
<tr>
<td>Sub-volume M node m Z coordinate</td>
</tr>
</tbody>
</table>
3.4.6.4 Variable File Format

<table>
<thead>
<tr>
<th>Sub-volume 1 node 1 variable 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-volume 1 node 2 variable 1</td>
</tr>
<tr>
<td>Sub-volume 1 node 3 variable 1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume 1 node n variable 1</td>
</tr>
<tr>
<td>Sub-volume 1 node 1 variable 2</td>
</tr>
<tr>
<td>Sub-volume 1 node 2 variable 2</td>
</tr>
<tr>
<td>Sub-volume 1 node 3 variable 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume 1 node n variable 2</td>
</tr>
<tr>
<td>Sub-volume 1 node 1 variable m</td>
</tr>
<tr>
<td>Sub-volume 1 node 2 variable m</td>
</tr>
<tr>
<td>Sub-volume 1 node 3 variable m</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume 1 node n variable m</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M node 1 variable 1</td>
</tr>
<tr>
<td>Sub-volume M node 2 variable 1</td>
</tr>
<tr>
<td>Sub-volume M node 3 variable 1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M node n variable 1</td>
</tr>
<tr>
<td>Sub-volume M node 1 variable 2</td>
</tr>
<tr>
<td>Sub-volume M node 2 variable 2</td>
</tr>
<tr>
<td>Sub-volume M node 3 variable 2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M node n variable 2</td>
</tr>
<tr>
<td>Sub-volume M node 1 variable m</td>
</tr>
<tr>
<td>Sub-volume M node 2 variable m</td>
</tr>
<tr>
<td>Sub-volume M node 3 variable m</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sub-volume M node n variable m</td>
</tr>
</tbody>
</table>
3.5. Parameter File

The Parameter file is in an ASCII format. By specifying the file name in a command line option of the filter program, the parameters are interpreted and set as input parameters.

Table 6  List of filter input parameters

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter detail</th>
<th>Default value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>in_dir</td>
<td>Input file directory</td>
<td>“.”</td>
<td>Directory path of input files※1</td>
</tr>
<tr>
<td>field_file</td>
<td>fld file name</td>
<td>-</td>
<td>Only for structured grid data※2, 3</td>
</tr>
<tr>
<td>in_prefix</td>
<td>Input file prefix</td>
<td>“input.”</td>
<td>Only for unstructured grid data</td>
</tr>
<tr>
<td>in_sufix</td>
<td>Input file suffix</td>
<td>“.dat”</td>
<td>Only for unstructured grid data</td>
</tr>
<tr>
<td>out_dir</td>
<td>Output file directory</td>
<td>“.”</td>
<td>Directory path of output files※1</td>
</tr>
<tr>
<td>out_prefix</td>
<td>Output file prefix</td>
<td>“output.”</td>
<td></td>
</tr>
<tr>
<td>Format</td>
<td>Step number format</td>
<td>“%05d”</td>
<td>Only for unstructured grid data</td>
</tr>
<tr>
<td>start_step</td>
<td>Start step number</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>end_step</td>
<td>End step number</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>n_layer</td>
<td>Number of octree layer</td>
<td>0</td>
<td>An integer from 0 to 7</td>
</tr>
<tr>
<td>output_type</td>
<td>File format</td>
<td>0</td>
<td>0: SPLIT format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: sub-volume aggregate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: step aggregate</td>
</tr>
<tr>
<td>file_number</td>
<td>Number of output files</td>
<td>0</td>
<td>An integer greater than 0. When 0 is set, the number of sub-volume is used. Valid only in Sub-volume aggregate file format.</td>
</tr>
<tr>
<td>mpi_volume_div</td>
<td>Number of MPI parallelism in sub-volume</td>
<td>1</td>
<td>The total number of MPI processes is given by mpi_volume_div × mpi_step_div.※4</td>
</tr>
<tr>
<td>mpi_step_div</td>
<td>Number of MPI parallelism in time step</td>
<td>1</td>
<td>The total number of MPI processes is given by mpi_volume_div × mpi_step_div.※4</td>
</tr>
</tbody>
</table>

※1 Directory path of input files
※2 Only for structured grid data
※3
※4 The total number of MPI processes is given by mpi_volume_div × mpi_step_div.
### MPI Parallel Processing

This section describes how to define the number of divisions in MPI parallel processing. In the following, we consider data processing for 50 steps × 8 sub-volumes.

#### (1) Step preferred parallelization
- When the number of processes is equal to or less than the number of steps, divide all the steps by the number of processes.
  
  Example) When 8 processes are used, each process treats 6 steps × 8 sub-volumes, or 7 steps × 8 sub-volumes.
- When the number of processes is larger than the number of steps, the number of...
processes used in the filter program is given by multiples of the number of steps, and sub-volume parallelization is also used.

Example) When 128 processes are used, the filter program works with $50 \times 2 = 100$ processes (with the residue of 28 processes), and each process treats 1 step $\times$ 4 sub-volumes.

(2) Sub-volume preferred parallelization
- When the number of processes is equal to or less than the number of sub-volumes, divide all the sub-volumes by the number of processes.
  
  Example) When 8 processes are used, each process treats 50 steps $\times$ 1 sub-volume.

- When the number of processes is larger than the number of sub-volumes, the number of processes used in the filter program is given by multiples of the number of sub-volumes, and step parallelization is also used.
  
  Example) When 128 processes are used, the filter program works with $8 \times 16 = 128$ processes (with the residue of 0 process), and each process treats 6 steps $\times$ 1 sub-volume or 7 steps $\times$ 1 sub-volume.

(3) User-specified parallelization
- When the parallel processing number, ‘mpi_volume_div’ and ‘mpi_step_div’ are specified, an error occurs if ‘mpi_volume_div $\times$ mpi_step_div’ does not agree with the number of processes.

3.7. Execution Method in Staging Environment

This section describes how to execute the filter program in the staging environment on the K computer. The filter program should be launched with keeping a consistency between the parameter file and staging parameters. Depending on the output data format of the filter program, multiple processes may produce a single file. In such a case, the data output should be made on a shared domain on local file system which is accessible from multiple processes.
3.7.1. Execution Shell Script and Parameter File

```bash
#!/bin/bash -x
#
#PJM --rsc-list "elapse=01:00:00"
#PJM --rsc-list "node=64"
#PJM --rsc-list "rsgrp=small"
#PJM --stg-transfiles all
#PJM --mpi "proc=64"
#PJM --mpi "use-rankdir"
#PJM --stg-transfiles all

#PJM --stgin "rank=*.filter %r:./"  #Stage in for load module……①
#PJM --stgin "rank=*.param.txt %r:./"  #Stage in for file……②
#PJM --stgin "rank=0 /data/ucd/ucd*.dat 0:../"  #Stage in for shared file……③
#PJM --stgout "rank=* %r:../output*.dat ./"  #Stage out for resulting file……④
#PJM --stgout "rank=* %r:./pbvr_filter.* ./LOG/"  #Stage out for file……⑤

#PJM -S

./work/system/Env_base

export PARALLEL=8
export OMP_NUM_THREADS=8

mpiexec -n 64 lpgparm -p 4MB -s 4MB -d 4MB -h 4MB -t 4MB ./filter param.txt  ………⑥
```

1. Transfer a load module to a rank directory of each process.
2. Transfer a parameter file to a rank directory of each process.
3. Transfer input data to a shared domain in local file system.
4. Transfer output data from a shared domain to a directory in global file system.
5. Transfer log and error files from a rank directory to a directory in the global file system.
6. Launch the load module in a rank directory of each process using the parameter file in a rank directory of each process as an argument.
3.7.2. Input-output Files and Directories

This section describes the relation between input-output files treated in the filter program and directories in the staging environment. Output data in the SPLIT format can be written on a rank directory, while output data in the other formats requires a shared directory for data aggregation.

Table 7   Table of input-output files and directories on K computer

<table>
<thead>
<tr>
<th>IO</th>
<th>File type</th>
<th>Rank directory</th>
<th>Shared domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Parameter file</td>
<td>○※1</td>
<td>○</td>
</tr>
<tr>
<td>Input</td>
<td>Input data</td>
<td>○※2</td>
<td>○</td>
</tr>
<tr>
<td>Output</td>
<td>Output data</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPLIT format</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>Step aggregate format</td>
<td>×</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td>Sub-volume aggregate format</td>
<td>×</td>
<td>○</td>
</tr>
<tr>
<td>Log &amp; error file</td>
<td>○※3</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

※1. The parameter file is read only from rank 0.
※2. The size and number of the input files should not exceed the resource limit of the staging environment (800 files/node, 14GB/node).
※3. The output directory is fixed to a rank directory.
3.8. Unstructured Grid data with Mixed Elements

When unstructured grid data contains several element types, the filter program firstly generates UCD binary data for each element type, and then divides the UCD binary data with a single element type into sub-volumes, which are read from the PBVR server.

By setting 1 to the parameter ‘multi_element_type’ in the parameter file, the filter program produces sub-volume files for each element type.

```
#
in_dir=.
in_prefix=MULTI
in_suffix=.dat
out_dir=.
out_prefix=div
out_prefix=.dat
format=%03
start_step=1
end_step=20
multi_element_type=1
```

Output files for each element type are generated with file names, which have a 2 digit code representing an element type at the beginning. The following list shows element names and codes.
Table 8  List of element types

<table>
<thead>
<tr>
<th>Element name</th>
<th>Element type code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle Linear</td>
<td>2</td>
</tr>
<tr>
<td>Quadrilateral Linear</td>
<td>3</td>
</tr>
<tr>
<td>Tetrahedron Linear</td>
<td>4</td>
</tr>
<tr>
<td>Pyramid</td>
<td>5</td>
</tr>
<tr>
<td>Prism</td>
<td>6</td>
</tr>
<tr>
<td>Hexahedron Linear</td>
<td>7</td>
</tr>
<tr>
<td>Triangle Quadratic</td>
<td>9</td>
</tr>
<tr>
<td>Quadrilateral Quadratic</td>
<td>10</td>
</tr>
<tr>
<td>Tetrahedral Quadratic</td>
<td>11</td>
</tr>
<tr>
<td>Hexahedral Quadratic</td>
<td>14</td>
</tr>
</tbody>
</table>

When the input data defined by the above parameter file consists of linear tetrahedral elements and quadratic tetrahedral elements, the following output files are generated.

Table 9  File names for mixed elements

<table>
<thead>
<tr>
<th>Original mixed elements data</th>
<th>Linear tetrahedral data</th>
<th>Quadratic tetrahedral data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULTI001.dat</td>
<td>04-div001.~</td>
<td>11-div001.~</td>
</tr>
<tr>
<td>MULTI002.dat</td>
<td>04-div002.~</td>
<td>11-div002.~</td>
</tr>
<tr>
<td>MULTI003.dat</td>
<td>04-div003.~</td>
<td>11-div003.~</td>
</tr>
<tr>
<td>MULTI004.dat</td>
<td>04-div004.~</td>
<td>11-div004.~</td>
</tr>
<tr>
<td>MULTI005.dat</td>
<td>04-div005.~</td>
<td>11-div005.~</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>MULTI020.dat</td>
<td>04-div020.~</td>
<td>11-div020.~</td>
</tr>
</tbody>
</table>

⇒ Decomposition

35
4 Server

4.1. Overview

The server program reads sub-volume files, which are produced by the filter program, and performs parallel visualization with the PBVR technique to generate particle data as the result of visualization.

4.2. Launch Server

PBVR can operate on supercomputers both in a batch mode, which generates only particle data in batch processing, and in a client-server mode, which generates particle data in interactive processing by connecting the client program and the server program via socket communications. Stand-alone processing on PCs or workstations is also possible by launching the client program and the server program in the client-server mode on the same machine. The following shows how to launch the server.

(launch in MPI+OpenMP processing with N processes)
$ mpiexec -n N ./CPUServer

(launch only in OpenMP)
$./CPUServer

※1. As the MPI+OpenMP server operates with master-slave MPI processing, the number of process N should be specified by the number of slave process + 1.
※2. In both processing modes, the number of OpenMP threads is set in the environmental variable 'OMP_NUM_THREADS'.
※3. In Windows, the command is launched from VS2013 x64 Native Tools command prompt.

Table 10 List of command line options for the server program

<table>
<thead>
<tr>
<th>Option</th>
<th>Launch mode</th>
<th>Possible parameters</th>
<th>Default parameters</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>SB</td>
<td>-</td>
<td>-</td>
<td>shows a list of options and possible parameters</td>
</tr>
<tr>
<td>-B</td>
<td>B</td>
<td>-</td>
<td>-</td>
<td>launch in the batch mode</td>
</tr>
</tbody>
</table>
-pa  B  File name  -  name of the visualization parameter file
-sl  B  1~99  1  sub-pixel level※2
-rl  B  1~99999  1  repeat level※2
-plimit  B  1~99999  10  limit of the number of particles (10⁶)※2
-vin  B  File name  -  name of the pfi file of input volume data※2
-pout  B  File name  ./  name of the output particle data file
-p  S  Port number  60000  port number of socket communication
-viewer  B  100 ~ 9999 × 100 ~ 9999  620×620  viewer resolution

※1. In Launch mode, S, B denote S: client-server mode and B: batch mode.
※2. In the current version, a transfer function in the batch mode is given by the visualization parameter file, which is specified by the command line option ‘-pa’. However, the other parameters in the visualization parameter file do not work, and they should be given by command line options.

4.2.1. Launch in Batch Mode

By specifying the command line option ‘-B’, the server program is launched in the batch mode. Refer to the following example for launching the server program in the batch mode (MPI+OpenMP.)

$ mpiexec -n 5 ./CPUServer -B -vin ./data/case.pfi -pout ./output/case -sl 2 -rl 2 -plimit 10 -pa ./param.in

In the above example, the input data, ‘./data/case.pfi’, is processed with the visualization parameter file, ‘./param.in’, and with command line options, ‘-sl 2 -rl 2 -plimit 10’, to output the following particle data.

./output/case_XXXXX_YYYYYYY_ZZZZZZZ.kvsml

XXXXX  : Number of steps (5 digit number)
YYYYYYY  : Sub-volume number (7 digit number)
ZZZZZZZZ  : Total number of Sub-volumes (7 digit number)
The visualization parameter file, which is specified by the command line option ‘-pa’, is generated by interactive processing in the client-server mode. Large-scale data processing in the batch mode is executed by using this file as it is, or by modifying a part of parameters in it.

4.2.2. Launch in Client-Server Mode

When the command line option ‘-B’ is not specified, the server program is launched in the client-server mode as shown in the following example.

$ mpiexec -n 5 ./CPUServer
  first reading time[ms]:0
  Server initialize done
  Server bind done
  Server listen done
  Waiting for connection ...

When “Waiting for connection” appears as the above example and the server program wait for socket communications from the client program, launch the client program on another terminal. In the client-server mode, an input volume data name is given on the client program.

The default value of a port number used in the socket communication is 60000. To change the port number, specify the command line option ‘-p’ as follows.

$ mpiexec -n 5 ./CPUServer -p 55555

4.2.3. Connecting Client and Server by Socket Communication

4.2.3.1 Local Connection

The following example shows how to launch both the client program and the server program on a single machine (machine A). In this example, they cooperate using the default port number 60000 of machine A.

Step 1 [Launch server]
  machineA$ mpiexec -n 5 ./CPUServer
Step 2 [Launch client]
  machineA$ ./PBVRViewer -vin filename
### 4.2.3.2 Remote Connection between Two Points

The following example shows how to launch the client program on machine A and the server program on machine B where the two machines are located at distant places. This example uses ssh port forwarding to connect the 60000 port of machine A to the 60000 port of machine B. The server and client programs on the two machines cooperate through the default port number 60000. Once the ssh port forwarding is established, the launching method is basically the same as the stand-alone mode or the local connection. In Windows, the ssh port forwarding is set up by using ssh software such as Tera Term and Putty.

**Step 1** [ssh port forwarding A→B]
```
machineA> ssh -L 60000:localhost:60000 username@machineB
(Forwarding the 60000 port of machine A to the 60000 port of machine B)
```
**Step 2** [Launch server]
```
machineB> mpiexec –n 5 ./CPUServer
```
**Step 3** [Launch client]
```
machineA> ./PBVRViewer –vin filename
```

### 4.2.3.3 Remote Connection via Several Machines

This section provides an example of connection between the client program and the server program on two remote machines A and B via machine C for security or other reasons. Once the ssh port forwarding is established, the launching method is basically the same as the stand-alone mode and the two point remote connection.

**Step 1** [ssh port forwarding A→C]
```
machineA> ssh -L 60000:localhost:60000 username@machineC
(Forwarding the 60000 port of machine A to the 60000 port of machine C)
```
**Step 2** [ssh port forwarding C→B]
```
machineC> ssh -L 60000:localhost:60000 username@machineB
(Forwarding the 60000 port of machine C to the 60000 port of machine B)
```
**Step 3** [Launch server]
```
machineB> mpiexec –n 5 ./CPUServer
```
**Step 4** [Launch client]
```
machineA> ./PBVRViewer –vin filename
```
4.2.3.4 Test Method for ssh Port Forwarding Connection

To check if ssh port forwarding is available, use the following sample program, which simply transfer input characters on a server program to a client program. This program is available from the link below.


How to launch the server program
./server port_number

How to launch the client program
./client server_hostname port_number

4.2.3.5 Connection with the Pre-post Server of the K-computer

This section shows an example of connecting a PC (machineA) in a laboratory to the data processing server of the K-computer (Pre-Post server pps3) via the login node of the K-computer (klogin).

Step1 [ssh port forward A→klogin]
machineA> ssh -L 60000:localhost:60000 username@k.aics.riken.jp
(Forwarding the 60000 port of machineA to the 60000 port of klogin)
Step2 [ssh port forward K login node→pre-post server]
klogin> ssh -L 60000:localhost:60000 username@pps3
(Forwarding the 60000 port of klogin to the 60000 port of pps3)
Step3 [Launch server]
pps3> mpiexec –n 5 ./CPUServer
Step4 [Launch client]
machineA> ./PBVRViewer -vin filename
(Forwarding the 60000 port of klogin to the 60000 port of pps3)

4.2.3.6 Local Connection on Windows

This section shows how to launch both the client and server programs on a sigle Windows machine. The VS2013 x64 Cross Tools command prompt in VisualStudio2013 is used as a terminal for launching the program.

Step1 [Launch server]
Windows> CPUServer.exe
Step 2 [Set a client parameter for Windows]

Windows> set TIMER_EVENT_INTERVAL=1000

Step 3 [Launch client]

Windows> PBVRViewer.exe –vin filename

Another launching method is to execute the following batch file.

set TIMER_EVENT_INTERVAL=1000
start PBVR_Server_win.exe
PBVRViewer.exe –vin filename

4.2.3.7 Remote Connection of Windows Client

To connect a Windows machine and a remote server, execute port forwarding by using SSH client softwares such as TeraTerm or Putty. The following example shows how to make port forwarding with TeraTerm.

1. Launch TeraTerm and cancel the “new connection” panel.

2. Select “Setup → SSH Forwarding” from the menu bar. Select “add” in the “SSH transfer” panel.
(3) In the “Select direction for forwarded port” dialogue, select “Forward local port” and enter the port number to be used on the client program. In the “to remote machine” box, enter the domain name or the IP address of the server. In the “port” box, enter the port number to be used on the server. Click on “OK” to complete the setup of Port forwarding.

(4) Connect to the server. Select “File → New connection” from the menu bar. In the “new connection” panel, enter the host name of the serve and click on “OK”. In the “SSH Authentication” panel, enter the user name and passphrase, or specify the location of the private key file, and click on “OK”.
The following procedures show how to launch the client and server programs after establishing port forwarding. This example uses the VS2013 x64 Cross Tools command prompt in VisualStudio2013 as a terminal for launching the client program.

Step1 [Launch server]

Server> mpiexec –n 4 ./CPUServer –p port_number

Step2 [Set a client parameter for Windows]

Windows> set TIMER_EVENT_INTERVAL=1000

Step3 [Launching the client]

Windows> PBVRViewer.exe –vin filename –p port_number

Note that, the client program on a Windows machine can be launched also by using the following batch file.

set TIMER_EVENT_INTERVAL=1000
PBVRViewer.exe –vin filename –p port_number
5 Client

5.1. Overview

Client operates in the client-server mode or in a stand-alone mode.

In the client-server mode, the client program receives particle data, which is generated as the results of visualization on the server, and renders it using OpenGL. The client program also sets visualization parameters (Sub-pixel level, Transfer function etc.), which are used in processing volume rendering on the server program, and sends them to the server. Data transfer between the client program and the server program is performed using socket communications with an arbitrary port number.

In the stand-alone mode, the client program reads and displays particle data which is generated by the server program in the batch mode.

5.2. Launching Client

The following examples show how to launch the client program in the client-server mode and in the stand-alone mode. The client program opens five panels, Viewer, Main panel, Transfer function editor, Time panel, and Animation panel, when it starts successfully.

(Launch in the client-server mode) ※1
$ ./PBVRViewer -vin [sub-volume file name※2] [command line option]

(Launch in the stand-alone mode)
$ ./PBVRViewer [particle data file name※3] [command line option※4]

※1. The client-server mode should be launched after starting the server program.
※2. The sub-volume file name should be specified by the absolute or relative path of the pfi file.
※3. The particle data should be stored in a directory, which does not contain other kvsml files.
※4. Note that in the stand-alone mode, the particle size is not rendered correctly if it is not specified by the command line options, the subpixel level (-sl), and the repeat level (-rl).

<table>
<thead>
<tr>
<th>Option</th>
<th>Launch mode</th>
<th>Possible parameters</th>
<th>Default parameters</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>C, S</td>
<td>-</td>
<td>-</td>
<td>display a list of options and possible parameter</td>
</tr>
<tr>
<td>-sl</td>
<td>C</td>
<td>1~99</td>
<td>1</td>
<td>sub-pixel level</td>
</tr>
</tbody>
</table>
-rl C 1~99999 1 repeat level

-S C u, m u particle sampling method (u: uniform sampling, m: metropolis sampling)

-tdata C all, div all particle data transfer method (all: step batch transmission, div: sub-volume divide forwarding)

-vin C file name - name of the pfi file of input volume data

-tf※2※3 C file name - name of the transfer function file

-p C port number 60000 port number of socket communication

-viewer C, S 100 ~ 9999 × 620×620 viewer resolution

-shading C, S {L/P/B},ka,kd,ks,n shading method※3

※2. Transfer function files are generated by the Export File button in the “Transfer function editor” panel.
※3. The transfer function, which is specified in this command line option, is reflected by pushing “Apply” button in “Transfer function editor”. The transfer function file can be loaded also by “Import File” button.
※4. This argument specifies shading parameters. The shading process gives lighting effect by emphasizing color tone of the viewing object. The lighting is expressed by the sum of three components: ambient, diffusion, specular reflection. The parameters specified by the argument modifies intensity of these three components.

L: Lambert Shading
effect: this method calculates the lighting without the specular reflection.
parameters: ka (coefficient for ambient 0~1), kd (coefficient for diffusion 0~1)

P: Phong Shading
effect: this method calculates the lighting with the specular reflection which is seen in smooth metal or mirror (also called highlight).
parameters: ka, kd, ks (coefficient for specular reflection 0~1), n (strength of highlight 0~100)

B: Blinn-Phong Shading
effect: the reduction model of the Phong Shading
parameters: ka, kd, ks, n
5.3. Terminating PBVR

5.3.1. Standard Termination

The client program renders time-series data from the initial time step to the final time step and repeats loop operation by returning to the initial time step after the final time step. To terminate the operation, enter ‘Ctrl+c’ in a console in which the client program is launched.

In the client-server mode, entering ‘Ctrl+c’ in the client console terminates both the client and server programs, when the time step is updated and they are synchronized. However, the client program ignores the ‘Ctrl+c’ key while client-server communications are interrupted with “Stop” button in “Time panel”.

5.3.2. Forced Termination

If the server program is terminated with the ‘Ctrl+c’ key, the client program sticks and cannot be terminated with the ‘Ctrl+c’ key. Also, even if the ‘Ctrl+c’ key is pressed to terminate the client program, both the client and server programs might stuck, when the time step is not updated because of heavy particle generation processes and some other reasons. In such a case, obtain process IDs of the client and server processes using ‘ps’ command, and then, force-quit them with ‘kill’ command as follows.

【Forced termination of the client process】
$ ps -C PBVRViewer
  PID TTY           TIME CMD
  19582 pts/6     00:00:00 PBVRViewer
$ kill -9 19582

【Forced termination of the server process】
$ ps -C CPUServer
  PID TTY           TIME CMD
  19539 pts/5     00:00:00 CPUServer
$ kill -9 19539
5.4. Operating Instructions

5.4.1. Viewer

Figure 6 shows “Viewer” displaying a rendering result of particle data.

【Operation】
Rotation: move a mouse while left-clicking
Translation: move a mouse while right-clicking
Zoom: scroll a mouse wheel up and down, or move a mouse up and down while holding down the ‘Ctrl’ key

【Display】
time step: time step of displayed data
fps: frame rates (frame/sec)
Figure 7 shows a main panel of the client program. Each item in the panel is described below.

- **SUB PIXEL LEVEL**
  Specify the subpixel level value $L_s$ which controls the image resolution.

- **REPEAT LEVEL**
  Specify the repeat level value $L_r$ which controls the image resolution.

Under the condition of $L_r=L_s^2$, almost the same image resolution is obtained, and therefore, the parameters $(L_s,L_r)=(2,1)$ and $(L_s,L_r)=(1,4)$ give almost the same picture. When $(L_s,L_r)=(2,1)$, the rendering process is executed at a time. However, when $(L_s,L_r)=(1,4)$, the rendering process is divided into four steps and ensemble average of image buffers is performed before displaying the image. As image
buffer sizes are proportional to $L_s^2$, it is recommend to use $L_r$ when the memory size is limited in the rendering process.

- **PARTICLE LIMIT**
  Specify the maximum number of particles, which are generated in the server program, in the unit of $M (10^6)$ particles to avoid explosive increase of the particle number due e.g. to false setting of a transfer function or any other reason.

- **EYE POINT**
  Specify a camera’s eye point.

- **CENTER POINT**
  Specify a camera’s center point.

- **UP VECTOR**
  Specify a camera’s up vector.

- **RESOLUTION**
  Specify viewer’s resolution.

- **SENDING**
  Show the progress of data send to the server program.

- **RECEIVING**
  Show the progress of data receive from the server program.

- **CPU MEMORY**
  Display the system memory usage (unit: MB).

- **GPU MEMORY**
  Display the GPU memory usage (unit: MB). Active when GPU rendering is used.

- **PBVR Filter information**
  Display information about the volume data on the server program (contents in the pfi file).

- **CROP button (the current version support this function only on Linux and Windows)**
  Display “CROP panel”. Details of “CROP panel” is described in the next section.

- **Set parameter button**
  Send parameters specified in “Main panel” to the server program.
5.4.3. CROP Panel

“CROP panel” appears by pressing “CROP button” in “Main panel”, and is related to operations of extracting and rendering elements involved within a region of interest (ROI), which is specified with a given shape. A shape of a ROI can be selected from cuboid, sphere, and cylinder. Note that “CROP panel” is not available on Mac in the current version.

![CROP panel](image)

Figure 8  CROP panel

Select Object : Specify a shape of a region of interest (ROI)
- CROP : Cuboid region
- SPHERE : Sphere
- PILLAR-XY base : Cylinder with a XY base
- PILLAR-YZ base : Cylinder with a YZ base
- PILLAR-XZ base : Cylinder with a XZ base
CROP : Specify a range of the cuboid
SPHERE : Specify the center and radius of the sphere
PILLER : Specify the radius, height, and center coordinate value of the cylinder
RESET : Reset CROP panel
APPLY : Extract a ROI
CLOSE : Close the panel

By displaying this panel, the shape of ROI is overwritten in “Viewer” as follows.

Feature 9   Viewer panel when Crop feature is applied
5.4.4. Transfer Function Editor

“Transfer Function Editor” creates a transfer function defining the correspondence between scalar data and color/opacity in volume rendering. In standard volume rendering, a transfer function is defined by a single physical quantity. However, in PBVR, a new multi-dimensional transfer function is implemented with the following three features:

1. Assign independent variable quantities to color and opacity.
2. Define each variable quantity by an arbitrary function of the coordinates X, Y, Z, and variables q1, q2, q3….
3. Multiple one-dimensional transfer functions t1-t5 are synthesized with an arbitrary function to define a multidimensional transfer function.

The newly developed transfer function design realized extremely flexible visualization. Figure 10 shows “Transfer Function Editor”. Each item in the panel is described below.
Specify resolution of a transfer function

- Transfer Function SYNTHESIZER Box
  Specify a function to synthesize one dimensional transfer functions t1～t5 ※1

- Transfer Function Name Pull-down Menu
  Select a transfer function (t1～t5) to edit.

- Reset Button
  Reset the panel.

- Apply Button
  Send a transfer function created in this panel to the server.

- File Path Box
  Specify a file path for saving and loading a transfer function file.

- Export File Button
  Save a transfer function created in this panel to a file in the same format as a parameter file specified with the command line option ‘-pa’.

- Import File Button
  Load a transfer function stored in a file to this panel.

5.4.4.1 Color Map Editor

[Transfer Fuction Color Map Category]
Define the correspondence between a variable quantity and color for a transfer function selected in “Transfer Fuction Name”.

- Color Bar
  Display a color bar of a transfer function created in this editor.

- Variable Box
  Specify the definitional equation of a (synthesized) variable quantity used for color of the selected transfer function. The following variables can be used in the equation.
  Physical quantities: q1, q2, q3, ... qn
  Coordinates: X, Y, Z

- Range Min Box
  Specify the minimum value of the variable quantity.

- Range Max Box
  Specify the maximum value of the variable quantity.

- Server side range min Box
  Display the minimum value of the (synthesized) variable quantity obtained in the server program.

- Server side range max Box
  Display the maximum value of the (synthesized) variable quantity obtained in the server program.
• **Color Map Editor (freeform curve) Button**
  Display a sub-panel which can create a transfer function with freeform curves drawn by a mouse.

![Color Map Editor (freeform curve) panel](image)

**Figure 11** Color Map Editor (freeform curve) panel

• **Color palette**
  Specify saturation, brightness, and hue of a color by placing a mouse. In the left space, the horizontal axis shows saturation while vertical axis shows brightness. The neighboring bar shows color hue.

• **RGB bar**
  Specify a hue of a color by placing a mouse. The upper-right box displays a color created by the Color palette and the RGB bar.

• **Color**
  Overpaint the bar with a color created by the Color palette and the RGB bar by tracing the color bar with a mouse while left-clicking. The blending ratio of the background color and overpainting color is determined by the mouse’s vertical position. For example, when the upper side of the color bar is traced side to side, the traced area is painted only by the specified color; while the middle part between upper and lower sides of the color bar is traced side to side, 50% of the specified color and 50% of the existing color are mixed in the traced area.

• **Reset Button**
  Reset the panel.
• Undo Button
  Undo the last action made by a mouse.
• Redo Button
  Redo the last action undone by a mouse.
• Save Button
  Save a transfer function created in this panel.
• Cancel Button
  Close the panel.

• Color Map Editor (expression) Button
  Display a panel which can create a transfer function in a functional expression.

![Color Map Editor (expression) panel](image)

Figure 12  Color Map Editor (expression) panel

• Color
  Display a color bar of a transfer function created in this panel.
• R Box
  Describe a transfer function of the R component of the color.
• G Box
  Describe a transfer function of the G component of the color.
• B Box
  Describe a transfer function of the B component of the color.
• Save Button
  Save a transfer function created in this panel.
• Cancel Button
  Close the panel.

• Color Map Editor (control points) Button
  Display a panel which can create a transfer function by specifying control points.
Figure 13  Color Map Editor (control points) panel

- **Color**
  Display a color bar of a transfer function which is created in this panel.

- **Control Point**
  Specify values of control points (up to 10 points) in the CP1)-10) boxes.

- **Red**
  Describe the R component of the color at the control points.

- **Green**
  Describe the G component of the color at the control points.

- **Blue**
  Describe the B component of the color at the control points.

- **Save Button**
  Save a transfer function created in this panel.

- **Cancel Button**
  Close the panel.

- **Color Map Editor (select colormap) Button**
  Display a panel which can create a transfer function by selecting color bar templates.
Figure 14  Color Map Editor (select colormap) panel

- Color
  Display a color bar of a transfer function which is created in this panel.
- Default Color pull-down Menu
  Select a color bar to be set as a transfer function from the following options.
  - RainBow
  - Blue-white-red
  - Black-red-yellow-white
  - Black-blue-violet--yellow-white
  - Black- yellow-white
  - Blue-green-red
  - Green-red-violet
  - Green- blue--white
  - HSV model
  - Gray-scale
- Save Button
  Save a transfer function created in this panel.
- Cancel Button
  Close the panel.

5.4.4.2 Opacity Editor

【Transfer Function Opacity Map Category】
Define the correspondence between a variable quantity and opacity for a transfer
function selected in “Transfer Fuction Name”.
- Opacity Bar
  Display a transfer function curve created in this editor.
- Variable Bar
Specify the definitional equation of a (synthesized) variable quantity used for opacity of the selected transfer function. The following variables can be used in the equation.

Physical quantities: q1, q2, q3, • • • qn
Coordinates: X, Y, Z

• Range Min Box
Specify the minimum value of the variable quantity.

• Range Max Box
Specify the maximum value of the variable quantity.

• Server side range min Box
Display the minimum value of the (synthesized) variable quantity obtained in the server program.

• Server side range max Box
Display the maximum value of the (synthesized) variable quantity obtained in the server program.

• Color Map Editor (freeform curve) Button
Display a panel which can create a transfer function with freedom curves drawn by a mouse.

![Opacity Map Editor (freeform curve) panel](image)

Figure 15  Opacity Map Editor (freeform curve) panel

• Opacity
Create a transfer function of opacity. A freeform curve is drawn by clicking the left mouse button and dragging the mouse. A linear interpolation line between two points is drawn by specifying control points with right clicks.

• Reset Button
Reset the panel.

• Undo Button
Undo the last action made by a mouse.

• Redo Button
Redo the last action undone by a mouse.

- **Save Button**
  Save a transfer function created in this panel.

- **Cancel Button**
  Close the panel.

- **Color Map Editor (expression) Button**
  Display a panel which can create a transfer function in a functional expression.

![Opacity Map Editor (expression) panel](image)

**Figure 16** Opacity Map Editor (expression) panel

- **Opacity**
  Display a transfer function of opacity following a functional expression given in the \( O \) box.

- **O Box**
  Describe a functional expression of a curve which specifies a transfer function of opacity.

- **Save Button**
  Save a transfer function created in this panel.

- **Cancel Button**
  Close the panel.

- **Color Map Editor (control point) Button**
  Display a panel which can create a transfer function by specifying control points.
Figure 17 Opacity Map Editor (control points) panel

- **Opacity (upper bar)**
  
  Display a transfer function of opacity created in this panel.

- **Control Point**
  
  Specify values of control points (up to 10 points) in the CP1)-10) boxes.

- **Opacity (right column)**
  
  Specify opacity at the control points.

- **Save Button**
  
  Save a transfer function created in this panel.

- **Cancel Button**
  
  Close the panel.
5.4.4.3 Function Editor

Built-in functions available in the function editor which is used to synthesize transfer functions and variable quantities, and to define color map curves, and opacity curves are listed below.

Table 12 Operation available in function editor

<table>
<thead>
<tr>
<th>Operation</th>
<th>Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>×</td>
<td>*</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Sin</td>
<td>sin(x)</td>
</tr>
<tr>
<td>Cos</td>
<td>cos(x)</td>
</tr>
<tr>
<td>Tan</td>
<td>tan(x)</td>
</tr>
<tr>
<td>Log</td>
<td>log(x)</td>
</tr>
<tr>
<td>Exp</td>
<td>exp(x)</td>
</tr>
<tr>
<td>Square root</td>
<td>sqrt(x)</td>
</tr>
<tr>
<td>Power</td>
<td>x^y</td>
</tr>
</tbody>
</table>
5.4.5. Time panel

Visualization time step is controlled by “Time panel” in Figure 18. Each widget works as follows.

![Time panel](image)

- **Progress**
  - Show the progress of time step by percentage.
- **Time step**
  - Specify the time step for rendering.
- **Min Time**
  - Specify the minimum time step for ROI.
- **Max Time**
  - Specify the maximum time step for ROI.
- **Start/Stop**
  - Start or Stop the communication between the client and server programs.
5.4.6. **Animation panel**

Image files displayed in “Viewer” are saved as sequential number image files for each time step. The file format is BMP. The sequential number image files can be converted and compressed into movie files like mpeg using freewares such as ImageMagic and ffmpeg.

![Animation panel](image)

**Figure 19  Animation panel**

**mode**: Only image files are available.

**capture**: Sequential number image files for each time step are produced when turned on.

**image file**: Specify the prefix of the sequential number image files.

The sequential number image files are stored in the client directory with the following names.

PBVR_image.00001.bmp
PBVR_image.00002.bmp
6 Examples Using Sample Data

The following sections show the functions of PBVR system by using the sample data ‘gt5d.tgz’.

6.1. Filter Processing

By extracting ‘gt5d.tgz’, the following files are generated under ‘./gt5d’.

gt5d.fld       AVS field file
co3d.dat       coordinate data
pd3d.dat       variable1
psid.dat       variable2
param.txt      filter program input parameter
demo1.tf       transfer function file for demonstration1
demo2.tf       transfer function file for demonstration2

Execute the filter program with the following command (OpenMP ver.)

$ ./filter ./gt5d/param.txt

The contents of param.txt are shown in the following.

#
in_dir=./
field_file=gt5d.fld
out_dir=./
out_prefix=case
start_step=0
end_step=4

The above example specifies the SPLIT file format by default, a single sub-volume (no
sub-volume decomposition), and the same directory both for input and output. This filter
process generates the following files in the specified directory.

case.pfi       pfi file
case_YYYYYYY_ZZZZZZZ_connect.dat  element configuration file
case_YYYYYYY_ZZZZZZZ_coord.dat    node coordinate file
case_XXXXX_YYYYYYY_ZZZZZZZ.kvsmli  kvsmli file
6.2. Starting PBVR

[step 1] Launch server (OpenMP ver.)

$./CPUServer
first reading time[ms]: 0
Server initialize done
Server bind done
Server listen done
Waiting for connection ...

[step 2] Launch client. This example uses the metropolis sampling, the sub-pixel level and the repeat level are set as \((L_s, L_r) = (4, 2)\), and the particle limit is given as 100M particles.

$ ./PBVRViewer -S m -vin ./gt5d/case.pfi -plimit 100 -rl 2 -sl 4
Figure 20  Initial state of GUI of PBVR system
6.3. Designing Transfer Function

This section shows visualization examples of ‘gt5d.fld’ using the multi-dimensional transfer function, that is produced by the advanced transfer function design capability of PBVR. ‘gt5d.fld’ is structured grid volume data consisting of two variables.
※1. Transfer functions used in section 6.3.1〜6.3.3 are stored in ‘demo1.tf’.
※2. A synthesized transfer function used in section 6.3.4 is stored in ‘demo2.tf’.

6.3.1. Volume Rendering of Single Variable

First of all, the volume rendering result for the variable q1 is confirmed by setting the transfer function t1 as shown in Figure 21. The transfer function is designed using “Transfer Function Editor”. The left side of this panel is color editor and the right side is opacity editor. This example shows the normal volume rendering using the single variable q1 for both color and opacity.

![Volume rendering result for the variable q1.](image)

Figure 21   Volume rendering result for the variable q1.
6.3.2. Multivariate Volume Rendering

The next example shows the result of multivariate volume rendering, in which the variables q1 and q2 are synthesized as shown in Figure 22. In this example, the color is designed by the variable q1, while the opacity is given by the variable q2. The opacity map extracts two torus surfaces, which are given by iso-surfaces of the variable q2, and the color means distributions of the variable q1 on these iso-surfaces.

Figure 22 Multivariate volume rendering result, which maps the color of the variable q1 on the iso-surfaces of the variable q2.

6.3.3. Slice Plane

The slice plane can be extracted by applying the multivariate volume rendering as shown in Figure 23. In PBVR, an arbitrary function can be used to design the transfer function. In this example, the cylindrical surface \(X^2+Z^2=\text{const.}\) is extracted and the color of the variable q1 is mapped on it.
Figure 23  The slice plane by using the multivariate volume rendering
6.3.4. Synthesis of Transfer Functions

This section explains the transfer function synthesizer of PBVR. Figure 24 shows the transfer function \( t_4 \) whose opacity function makes the region for \( Y>0 \) transparent. By synthesizing the previously described transfer functions \( t_1, t_2, t_3 \) and the new transfer function \( t_4 \) as \((t_1+t_2)*t_4+t_3\), sub-regions are extracted. In this example, the colors of \( t_2 \) and \( t_3 \) are set as \((R,G,B) = (0,0,0)\), while the color of \( t_4 \) is set to be \((R,G,B) = (1,1,1)\). By using the above synthesis equation, the final color becomes the rainbow defined for \( t_1 \). On the other hand, the opacity of \( t_4 \) is multiplied to the sum of \( t_1 \) and \( t_2 \) to extract the lower half \((Y<0)\) region of \( t_1 \) and \( t_2 \), and then, the resulting region is synthesized with the cylindrical surface given by \( t_3 \). Although PBVR is equipped with “CROP panel”, the transfer function synthesizer is more flexible because it can extract different ROIs for each variable quantity, simultaneously.

![Figure 24 Transfer function synthesizer](image)
6.4. Preserving Result

After the design of the transfer function, PBVR system can preserve the resulting image and parameters in following three ways.

(1) Output image
   In order to output the resulting images, select ‘on’ from the drop down menu ‘capture’ in “Animation panel”. The bitmap image files (PBVR_image.xxxxx.bmp) are generated in working directory.

(2) Transfer function file
   In order to preserve the transfer function file, write file name of the transfer function in ‘File Path’ box in “Transfer Function Editor” and press ‘Export File’. This file is loaded by ‘Import File’. Note that the input transfer function is not reflected until ‘Apply’ is pressed.

(3) Visualization parameter file
   For the server program in the batch mode, all the visualization parameters including the transfer function can be exported. Open “File Sub-panel” from “Main panel”, specify the parameter filename in it, and press ‘Export File’.

Figure 25 Export visualization results
6.5. Example of Batch Mode

This section explains the batch mode server processing using the visualization parameter file exported in the previous section. This mode is developed for massively parallel processing on supercomputer. It is useful also for high speed rendering of time series data in the stand-alone mode client processing, in which latency due to particle generation and particle data transfer can be eliminated.

[Step 1] Launch server in the batch mode (OpenMP ver.) ※1

```
$./CPUServer -B -vin ./gt5d/case.pfi -pout ./particle/ -S m -plimit 100 -sl 4 -pa ./param.in
```

[Step 2] Launch client in the stand-alone mode.※2

```
$./PBVRViewer ./particle/ -sl 4
```

※1. In the current version, the server program in the batch mode requires command line options for visualization parameters except for the transfer function.
※2. The sub-pixel level and the repeat level should be specified also in launching the client in the stand-alone mode because they determine the size of particles in the rendering process.