VisIVO

VisIVO Team

Aug 24, 2023

VISIVO

CHAPTER

ONE

INTRODUCTION

VisIVO (Visualization Interface for the Virtual Observatory) performs multi-dimensional data analysis and knowledge discovery of a-priori unknown relationships between multi-variate and complex datasets in Astrophysics. VisIVO development started within the activities of the Virtual Observatory framework. It supplies users with functionality to render meaningfully highly-complex, large-scale datasets and create movies of such views using distributed computing infrastructures.

VisIVO provides an integrated suite of tools and services that can also be used in other scientific fields. The VisIVO suite offers a variety of flavours as follows:

- **VisIVO Server** a platform for high performance visualization.
- **VisIVO Library** for running complex workflows on DCI, clouds and HPC infrastructures to efficiently produce complex views of the dataset and full movies directly with the user-code internal data representation (i.e. without the need to create intermediate files).
- **VisIVO ViaLactea Visual Analytics (VLVA)**, developed within the ViaLactea project, which allows to exploit a combination of all new-generation surveys of the Galactic Plane to analyze star forming regions of the Milky Way.
- **VisIVO ViaLactea Web (VLW)**, a work-in-progress simplified web version of the VLVA, developed in collaboration with University of Portsmouth (UK) providing an efficient visualisation (GPU and CPU rendering) on remote server.

VisIVO technologies have been demonstrated as success stories in numerous relevant multidisciplinary environments, and *Projects*. As an open access software, significant interest in its usage has been shown from all over the world (evident by the number of hits on the websites) and several scientists from different domains (e.g. Nuclear Physics) have produced visualisations of their data (simulations or observational datasets).

The suite is maintained by INAF-Astrophysical Observatory of Catania and is continuously enriched with several international collaborations.

CHAPTER

TWO

PROJECTS

VisIVO technologies have been demonstrated as success stories in numerous relevant multidisciplinary environments, e.g. EU FP7 projects such as EDGI, EGI-Inspire, SCI-BUS, ViaLactea; EU H2020 projects such as NEANIAS; and national projects such as MuonPortal and CIRASA.

CHAPTER

THREE

PUBLICATIONS

Antoniou, Varvara, et al. "Integrating Virtual Reality and GIS Tools for Geological Mapping, Data Collection and Analysis: An Example from the Metaxa Mine, Santorini (Greece)." Applied Sciences 10.23 (2020): 8317.

Sciacca, Eva, et al. "VIALACTEA science gateway for Milky Way analysis." Future Generation Computer Systems 94 (2019): 947-956.

Krokos, Mel, et al. "Workflows for virtual reality visualisation and navigation scenarios in earth sciences." 5th International Conference on Geographical Information Systems Theory, Applications and Management. SciTePress, 2019.

Gerloni, Ilario Gabriele, et al. "Immersive virtual reality for earth sciences." 2018 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE, 2018.

Vitello, F., et al. "Vialactea visual analytics tool for star formation studies of the galactic plane." Publications of the Astronomical Society of the Pacific 130.990 (2018): 084503.

Sciacca, Eva, et al. "VIALACTEA science gateway for Milky Way analysis." Future Generation Computer Systems (2017).

Becciani, Ugo, et al. "Science gateway technologies for the astrophysics community." Concurrency and Computation: Practice and Experience 27.2 (2015): 306-327.

Castelli, Giuliano, et al. "VO-compliant workflows and science gateways." Astronomy and Computing 11 (2015): 102-108.

Vitello, Fabio, et al. "Mobile application development exploiting science gateway technologies." Concurrency and Computation: Practice and Experience 27.16 (2015): 4361-4376.

Sciacca, Eva, et al. "An integrated visualization environment for the virtual observatory: Current status and future directions." Astronomy and Computing 11 (2015): 146-154.

Sciacca, Eva, et al. "Visivo gateway and visivo mobile for the astrophysics community." Science Gateways for Distributed Computing Infrastructures. Springer, Cham, 2014. 181-194.

Vitello, Fabio, et al. "Developing a mobile application connected to a science gateway." Science Gateways (IWSG), 2014 6th International Workshop on. IEEE, 2014.

Becciani, Ugo, et al. "Creating gateway alliances using WS-PGRADE/gUSE." Science Gateways for Distributed Computing Infrastructures. Springer, Cham, 2014. 255-270.

Sciacca, Eva, et al. "Towards a big data exploration framework for astronomical archives." High Performance Computing & Simulation (HPCS), 2014 International Conference on. IEEE, 2014.

Sciacca, Eva, et al. "VisIVO Science Gateway: a Collaborative Environment for the Astrophysics Community." IWSG. 2013.

Sciacca, Eva, et al. "Visivo workflow-oriented science gateway for astrophysical visualization." Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on. IEEE, 2013.

Costa, A., et al. "VisIVO: A Web-based, Workflow-enabled Gateway for Astrophysical Visualization." Astronomical Data Analysis Software and Systems XXII. Vol. 475. 2013.

Massimino, P., et al. "Learning astrophysics through mobile gaming." Astronomical Data Analysis Software and Systems XXII. Vol. 475. 2013. Riggi, S., et al. "A large area cosmic ray detector for the inspection of hidden high-Z materials inside containers." Journal of Physics: Conference Series. Vol. 409. No. 1. IOP Publishing, 2013.

Becciani, Ugo, et al. "Visivo: A library and integrated tools for large astrophysical dataset exploration." Astronomical Data Analysis Software and Systems XXI. Vol. 461. 2012.

Petta, C., et al. "Track reconstruction and VisIVO visualization of the cosmic secondary charged radiation paths in the detection of heavy nuclear materials using muon tomography." Astronomical Data Analysis Software and Systems XXI. Vol. 461. 2012.

Becciani, Ugo, et al. "Cosmological simulations and data exploration: a testcase on the usage of Grid infrastructure." Journal of Grid Computing 10.2 (2012): 265-277.

Presti, D. Lo, et al. "Design of a large area tomograph to search for high-Z materials inside containers by cosmic muons." Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE. IEEE, 2012.

Costa, A., et al. "Visivoweb: a www environment for large-scale astrophysical visualization." Publications of the Astronomical Society of the Pacific 123.902 (2011): 503.

Costa, A., et al. "VisIVODesktop 3.0: an interactive desktop environment for astrophysical visualization." Astronomical Data Analysis Software and Systems XX. Vol. 442. 2011.

Becciani, U., et al. "Large-scale astrophysical visualization on smartphones." Astronomical Data Analysis Software and Systems XX. Vol. 442. 2011.

Becciani, U., et al. "Visivo–integrated tools and services for large-scale astrophysical visualization." Publications of the Astronomical Society of the Pacific 122.887 (2010): 119.

Becciani, U., M. Comparato, and C. Gheller. "VisIVO: a VO enabled tool for Scientific Visualization and Data Analysis." Memorie della Societa Astronomica Italiana Supplementi 9 (2006): 427.

Becciani, Ugo, et al. "VisIVO: an interoperable visualisation tool for Virtual Observatory data." Proceedings of the International Astronomical Union 2.14 (2006): 622-622.

Becciani, U., et al. "VisIVO: a tool for the virtual observatory and grid environment." Astronomical Data Analysis Software and Systems XVI. Vol. 376. 2007.

Comparato, Marco, et al. "Visualization, exploration, and data analysis of complex astrophysical data." Publications of the Astronomical Society of the Pacific 119.858 (2007): 898.

Costa, A., et al. "The tvo archive for cosmological simulations: Web services and architecture." Publications of the Astronomical Society of the Pacific 120.870 (2008): 933.

Becciani, U., et al. "VisIVO: new integrated services." Astronomical Data Analysis Software and Systems XVIII. Vol. 411. 2009.

Manzato, P., et al. "An Archive and Tools for Cosmological Simulations inside the Virtual Observatory." Astronomical Data Analysis Software and Systems XVII. Vol. 394. 2008.

Costa, A., et al. "VisIVO: a Visualization Toolkit towards Grid Environment." Astronomical Data Analysis Software and Systems XV. Vol. 351. 2006.

Caniglia, G., et al. "Visual discovery in large-scale astrophysical datasets; experiences using the sloan digital sky survey." Visualisation, 2009. VIZ'09. Second International Conference in. IEEE, 2009.

Costa, A., et al. "Theoretical Virtual Observatory and Grid Web services: VisIVO and new capabilities." Highlights of Astronomy14 (2007): 627.

Comparato, M., and U. Becciani. "VisIVO. Data exploration." Memorie della Societa Astronomica Italiana Supplementi 13 (2009): 29.

Caniglia, G., et al. “VisIVO: data exploration of complex data.” Memorie della Societa Astronomica Italiana 80 (2009): 441.

Comparato, Marco, et al. “Visualization and Data Mining in the Virtual Observatory Framework.” Modelling And Simulation In Science. 2007. 295-299.

Gheller, C., M. Comparato, and U. Becciani. “VisIVO: interoperable visualization and data analysis for the VO.” Astronomical Data Analysis Software and Systems XV. Vol. 351. 2006.

Molinaro, M., et al. “VO Compliant Visualization of Theoretical Data.” Astronomical Data Analysis Software and Systems XIX. Vol. 434. 2010.

CHAPTER

FOUR

TEAM

- Ugo Becciani
- Fabio Vitello
- Eva Sciacca
- Giuseppe Tudisco
- Valentina Cesare
- Alessandro Costa
- Simone Riggi
- Marilena Bandieramonte
- Gabriella Caniglia
- Marco Comparato
- Pietro Massimino

CHAPTER

FIVE

FORMER TOOLS

5.1 VisIVO Desktop

VisIVODesktop provides a multi-platform desktop environment for the interactive visualization of cosmological simulations.

5.2 VisIVO and Vialactea Science Gateway

The VisIVO Science Gateway aims to create an astrophysical portal based on the generic-purpose gUSE/WS-PGRADE portal family to access VisIVO software tools. It has been developed thanks to the EU FP7 Sci-Bus Project.

The VIALACTEA Science Gateway operates as a central workbench for the VIALACTEA community in order to allow astronomers to process the new-generation surveys (from Infrared to Radio) of the Galactic Plane to build and deliver a quantitative 3D model of our Milky Way Galaxy. The adopted agile software development process allowed to fulfill the community needs in terms of required workflows and underlying resource monitoring. The science gateway has been developed in the context of the EU FP7 VIALACTEA project.

5.3 VisIVO Mobile

The VisIVO Mobile application is connected to a WS-PGRADE/gUSE gateway [4] and accesses to VisIVO Server tools, enabling execution of a comprehensive collection of modules for data exploration (including processing and visualization) of astrophysical datasets on DCIs. A number of customized workflows have been configured by default, e.g. to allow local or remote upload of datasets and creation of scientific movies. These workflows are provided with specific user interfaces to enable easy parameter setting for the end-users while hiding the complexity of the underlying system and infrastructures. The mobile application employs the same user accounts of the SG and provides a platform for astrophysical communities to share results, analysis experiences and exploration of their datasets.

Vitello, Fabio, et al. "Mobile application development exploiting science gateway technologies." Concurrency and Computation: Practice and Experience 27.16 (2015): 4361-4376.

Vitello, Fabio, et al. "Developing a mobile application connected to a science gateway." Science Gateways (IWSG), 2014 6th International Workshop on. IEEE, 2014..

Sciacca, Eva, et al. "Visivo gateway and visivo mobile for the astrophysics community." Science Gateways for Distributed Computing Infrastructures. Springer, Cham, 2014. 181-194..

CHAPTER

SIX

INTRODUCTION

VisIVO Server is a suite of software tools for creating customized views of 3D renderings from astrophysical data tables. Their defining characteristic is that no fixed limits are prescribed regarding the dimensionality of data tables input for processing, thus supporting very large scale datasets.

VisIVO Server consists of three core components: *VisIVO Importer*, *VisIVO Filter* and *VisIVO Viewer* respectively. Their functionality and usage is described in the following sections.

To create customized views of 3D renderings from astrophysical data tables, a two-stage process is employed. First, VisIVO Importer is utilized to convert user datasets into *VisIVO Binary Table* (VBT). Then, VisIVO Viewer is invoked to display customized views of 3D renderings.

As an example, consider displaying views from only three columns of an astrophysical data table supplied in ascii form, say col_1, col_2 and col_3, by using the commands

```
$ VisIVOImporter --fformat ascii UserDataSet.txt
$ VisIVOViewer -x col_1 -y col_2 -z col_3 --scale --glyphs pixel VBT.bin
```

VisIVOServer is distributed with GNU General Public License v2.0 License for NON COMMERCIAL use.

VisIVOServer source code is on GitHub.

6.1 Installation

To install VisIVO Server, follow the instructions indicated in the INSTALL.md file in the GitHub repository https://github.com/VisIVOLab/VisIVOServer.

6.2 VisIVO Binary Table

A VisIVO Binary Table (VBT) is a highly-efficient data representation used by VisIVO Server internally. A VBT is realized through a header file (extension `.bin.head`) containing all necessary metadata, and a raw data file (extension `.bin`) storing actual data values.

For example, the header may contain information regarding the overall number of fields and number of points for each field (for point datasets) or the number of cells and relevant mesh sizes (for volume datasets). The raw data file is typically a sequence of values, e.g. all X followed by all Y values.

6.2.1 Header

The header file contains the following fields:

```
float | double
n1
n2 [ GeoX GeoY GeoZ DX DY DZ ]
little | big
X
Y
Z
Vx
Vy
Vz
```

Where:

- **float | double** It is the data type of the storage variables used.
- **n1** It denotes the number of columns (fields) in the VBT (e.g. 6).
- **n2** It denotes the number of rows in the VBT (e.g. 262144).
- **GeoX GeoY GeoZ DX DY DZ** They are employed only if the VBT represents volumetric datasets.

  In that case GeoX, GeoY and GeoZ represent the mesh geometry, while DX, DY and DZ represent the x, y and z size of volumetric cells (e.g. 64 64 64 1.0 1.0 1.0).
- **little | big** It denotes the endianism employed in the VBT. After this field there exist n1 rows that indicate the VBT columns as positions (X, Y, Z) and velocities (Vx, Vy, Vz).

CHAPTER

SEVEN

VISIVO IMPORTER

VisIVO Importer converts user-supplied datasets into a VBT. VBTs are used by VisIVO Filter for data processing and VisIVO Viewer for visualization. The conversion is independent of data dimensionality.

General VisIVO Importer Syntax:

```
$ VisIVOImporter --flag1 --flag2 ... --flagn FileName
```

VisIVO Importer General Options:

--fformat Specifies the format of input datasets.

--out <name> (Optional) Output FileName. The path specified with this flag is the VisIVO Importer output directory.

If no path is specified the VisIVO Importer output directory is the current directory, and the default name for OutputFileName is `VisIVOServerBinary.bin`.

--volume (Optional) Used to create volumes.

--compx <value>, --compy <value>, --compz <value> (Optional) Used for entering geometry for volumetric datasets.

If data size fits in a cubic mesh, values are computed automatically.

--sizex <value>, --sizey <value>, --sizez <value> (Optional) Used for specifying volumetric cell dimensions; the default values are 1.0, 1.0 and 1.0 respectively.

--userpwd <username:password> (Optional) Used to prescribe a username and password for accessing remote files.

Ambiguous characters for the shell (i.e. $ >, < etc..) must be given with escape character "". For example `guest$09` must be given as `guest\$09`.

Note: Escape characters MUST NOT BE GIVEN using the parameterFile.

--binaryheader <headerfilename> (Optional) Used to specify the file name of the header of VBTs; this flag is ignored in case of other formats.

--missingvalue <value> (Optional) Used to set the missing data to a fixed value. If not present the default value is: -1.0918273645e+23.

--textvalue <value> (Optional) Used to set the textual data to a fixed value. If not present a default value is given: -1.4536271809e+15.

--bigendian (Optional) Used only for big endian Gadget and FLY input files, otherwise by default this flag is set to little endian.

--double (Optional) Used only for the double data type of FLY input files, otherwise by default this flag is set to float.

--npoints <value> (Optional) Used only for FLY input files to specify the number of data points.

--history (Optional) Create an XML file which contains the history of operations performed.

--historyfile <filename> (Optional) Change output history file name. Default: `hist.xml`.

The FileName is the local (or remote) file containing the data to be converted into a VBT. If FileName starts with `http://`, `ftp://` or `sftp://` the remote file is downloaded automatically. However if the –userpwd option is specified, the prescribed username and password are employed for remote access. The sftp syntax requires the remote directory where data are located.

Note: sftp is not allowed if VisIVO is compiled with the LIGHT option. The server must have libcurl with ssl support to enable the sftp functionality and VisIVO must be compiled without the LIGHT option.

All downloaded files are copied temporarily into the directory given in –out option and are deleted automatically at the end of the import process. Under the current file directory, the file `DownLoadedFilename_VisIVO_List` contains information on all download operations. If the fformat option is binary VisIVOImporter will attempt to download two remote files, a binary table (given as remote filename) and its associated header file (same name +``.head`` extension).

7.1 Parameter file

Alternatively, to run the importer with options specified in the parameter file:

```
$ VisIVOImporter <parameterFile>
```

Lines starting with # are comments.

Options with one or more parameters must be all specified after the "=" sign in the same line (e.g. field=X Y Z).

Options that do not require parameters must be given with "true" keyword (e.g. double=true, bigendian=true).

The input filename has the keyword file (ex. file=myInputFilename).

Examples of parameter files are the following:

```
fformat=ascii
out=outFilename.bin
file=asciinputFile
```

```
fformat=votablefast
out=/home/user/dataNewTable.bin
missingvalue=0.0
file=myVOTable.xml
```

```
fformat=fly
out=FlyData.bin
double=true
npoints=1000
bigendian=true
userpwd=myusername:mypassword
file=http://remotehosts.domain.eu/directory/InputDataFile
```

7.2 Formats

The following file formats are supported:

- *ASCII*
- *CSV*
- *BINARY*
- *FLY*
- *FITS Table*
- *FITS Image*
- *GADGET*
- *HDF5*
- *MUPORTAL*
- *RAMSES*
- *RAW Binary*
- *RAW Grid*
- *VOTable*

7.2.1 ASCII

ASCII files are expected to be in tabular form. An ASCII file may contain values for N variables organized in columns. The columns are typically separated by whitespace characters, e.g. spaces or tabs.

The first row of an ASCII file lists the N variables names explicitly. As an example, the command below produces `NewTable.bin`, `NewTable.bin.head` from `ASCIIUserFileName.txt`.

Usage:

```
$ VisIVOImporter --fformat ascii --out /home/user/data/NewTable ASCIIUserFileName.txt
```

7.2.2 CSV

CSV is a delimited data format that has fields/columns separated by the comma character and records/rows separated by newlines.

Fields that contain a special character (such as comma, newline, or double quote) must be enclosed in double quotes. However, if a line contains a single entry that happens to be the empty string, it may be enclosed in double quotes. If a field's value is a double quote character, this is dealt with by placing another double quote character next to it.

The CSV file format does not require a specific character encoding, byte order, or line terminator format.

As an example, the command below produces the files `NewTable.bin` and `NewTable.bin.head` from the user-supplied CSV file `CSVUserFileName.txt`.

Usage:

```
$ VisIVOImporter --fformat csv --out /home/user/data/NewTable CSVUserFileName.txt
```

Note: The Importer automatically skips all the lines starting with # character. If the first line contains column names starting with #, this character will be removed, and the columns names will be given without it.

7.2.3 BINARY

The binary format is supposed to be the Internal Binary Table.

Usage:

```
$ VisIVOImporter --fformat binary --out /home/user/data/NewTable BinaryUserFilename.bin
```

Assuming that a VBT is to be processed, the previous command produces `NewTable.bin` and `NewTable.bin.head` from `VBTUserFileName.bin`. Note that the files `VBTUserFileName.bin` and `VBTUserFileName.bin.head` must be (either local or remote) existing files. Also, if the –binaryheader option is prescribed, a header file with the specified filename must exist. In case this header file resides remotely, a complete copy is created within the VisIVO Server output directory.

Note: This command is useful for changing the endianism of a binary table. An input big endian table is transformed in a little endian table if system where VisIVOImporter is running is a little endian system and viceversa.

7.2.4 FLY

FLY is code that uses the tree N-body method, for three-dimensional self-gravitating collisionless systems evolution. FLY is a fully parallel code based on the tree Barnes-Hut algorithm; periodical boundary conditions are implemented by means of the Ewald summation technique.

FLY is based on the one-side communication paradigm for sharing data among processors, accessing remotely private data without synchronism. The FLY output format is a binary sequence of values of n data points as follows: X1,Y1,Z1,X2,Y2,Z2 ,.... Xn,Yn,Zn, Vx1,Vy1,Vz1, Vx2,Vy2,Vz2, ... Vxn,Vyn,Vzn.

As an example, the command below produces `NewTable.bin` and `NewTable.bin.head` from `FLYUserFile` which is a FLY file using double data types, containing a total of 2000000 data points.

Usage:

```
$ VisIVOImporter --fformat fly --out /home/user/data/NewTable --double --npoints 2000000␣
↪FLYUserFile
```

The FLY format also allows the download of elements given in a descriptor file (.desc extension):

```
flyDesc             (type of the Fly descriptor)
2m_test             (ID)
double              (data type)
time                (ID for snapshot sequence)
2097152             (number of points on each snapshot)
50 50 50            (box dimension in the proper unit)
l                   (“l” or “b” for data endianism)
0.0 out_scdm_0.0000 (time tag and snapshot filename sequence)
```

(continues on next page)

(continued from previous page)

```
1.0 out_scdm_1.0000
2.0 out_scdm_2.0000
```

In this case the flags –npoints and –double must no be given as values are read automatically from the descriptor file. Each listed file (out_scdm_# in this case) produces a VBT.

```
$ VisIVOImporter --fformat fly --out /home/user/data/NewTable FLYUserFile.desc
```

Note: The names of output files will be determined by using the –out parameter concatenated first by a "_" and then by the listed filename. As an example, if –out /tmp/pippo is prescribed, output filenames will be `/tmp/pippo_out_scdm_0.000.bin` (and one with the extension .bin.head). On the other hand, if –out option is not prescribed, output filenames will be `./VisIVOServerBinaryout_scdm_0.000.bin` (and one with extension .bin.head).

7.2.5 FITS Table

The definition of FITS is a codification into a formal standard, by the NASA/Science Office of Standards and Technology (NOST), of the FITS rules (http://fits.gsfc.nasa.gov) endorsed by the IAU.

FITS supports tabular data with named columns and multidimensional rows. Both binary and ASCII FITS table versions have been specified. The data in a column of a FITS table can be in a different format from the data in other columns. Together with the ability to string multiple header/data blocks together, by using FITS files it is possible to represent entire relational databases.

As an example, the command below produces `NewTable.bin`, `NewTable.bin.head` from `FITSTableUserFile`.

Usage:

```
$ VisIVOImporter --fformat fitstable --out /home/user/data/NewTable FITSTableUserFile
```

7.2.6 FITS Image

The Definition of FITS is a codification into a formal standard, by the NASA/Science Office of Standards and Technology (NOST), of the FITS rules endorsed by the IAU.

FITS image headers can contain information about one or more scientific coordinate systems that are overlain on the image itself. Images contain an implicit Cartesian coordinate system that describes the location of each pixel in the image, but scientific uses generally require working in 'world' coordinates, for example the celestial coordinate system.

As an example, the command below produces `NewTable.bin`, `NewTable.bin.head` from `FITSImageUserFile`.

Usage:

```
$ VisIVOImporter --fformat fitsimage --out /home/user/data/NewTable FITSImageUserFile
```

7.2.7 GADGET

GADGET is freely-available code for cosmological N-body/SPH simulations on massively parallel computers with distributed memory. GADGET uses an explicit communication model that is implemented with the standardized MPI communication interface. The code can be run on essentially all supercomputer systems presently in use, including clusters of workstations or individual PCs.

VisIVO Importer will produce a VBT for each species in the gadget file supplied. As an example, the command below will produce the files `NewTableHALO.bin`, `NewTableHALO.bin.head`, and `NewTableGAS.bin`, `NewTableGAS.bin.head` from the GADGET file `GadgetUserFile` if we assume that it contains only two species, namely halo and gas particles.

Usage:

```
$ VisIVOImporter --fformat gadget --out /home/user/data/NewTable GadgetUserFile
```

7.2.8 HDF5

The hierarchical Data Format is a library and multi-object file format for the transfer of graphical and numerical data between computers. It was created by the NCSA, but is currently maintained by The HDF Group. The freely available HDF distribution consists of the library, command-line utilities, test suite source, Java interface, and the Java-based HDF Viewer (HDFView).

HDF supports several different data models, including multidimensional arrays, raster images, and tables. Each defines a specific aggregate data type and provides an API for reading, writing, and organizing data and metadata. New data models can be added by the HDF developers or users.

```
$ VisIVOImporter --fformat hdf5 --datasetlist dataset1 dataset2 --hyperslab dataset 210,
↪10 100,100 --out /home/user/data/NewTable HDF5UserFile
```

```
$ VisIVOImporter --fformat hdf5 --datasetlist vol1 vol2 --hyperslab vol1 0,0,0 10,10,10␣
↪vol2 5,5,5 10,10,10 --compx 10 --compy 10 --compz 10 --out /home/user/data/NewTable␣
↪HDF5UserFile
```

Options

Dataset

This importer can be used with the datasetlist option. When the datasetlist option is not given all datasets in the hdf5 will be considered forming the VBT.

```
--datasetlist nameOfDatset1 nameOfDatset2 nameOfDatset3 ...
```

If the parameterfile is used the file can contain more rows listing datasets. The parameter file must have only one dataset in the datasetlist parameter. More rows with datasetlist parameter must be given to import more than one dataset.

Hyperslab

The hyperslab option must be given as follows:

```
--hyperslab nameOfDataset offset count
```

where offset and counts are the same of the hdf5 file and must be comma separated values. If count exceeds the size of dataset, it is automatically adjusted up to the end of hyperslab.

If offset and/or count contains lower values than the dataset dimension, not specified offset are put to 0 and count to the dimension of the dataset. But this fact could give some unexpected behaviors and it is strongly recommended giving all parameters of offset and dimension.

The offset and dimension values must be equal to the number of rank. If rank=3 offset/dimension must contain 3 separated comma values (one for each rank) if offset/dimension contains only two values the third value is assumed to be equal to the size of the dataset.

Example rank=3, dataset dimension (200,200,200). –hyperslab datasetname 10,10 20,20 is considered as –hyperslab datasetname 10,10,200 20,20,200.

The parameter file must have only one hyperslab in the hyperslab parameter. More rows with an hyperslab parameter must be given to describe more than one dataset.

Note in case of volume with more than one hyperslab: users are strongly suggested to givethe same hyperlab extension (count). In case of volume with specified hyperslab the –compx-y-z values are ignored. Datasets with different numbers of rows will produce columns with the number of rows equal to the maximum number of rows. The rows will be pads with missingvalue parameter.

Datasets with rank greather than 1000 cannot be read.

Datasets

Datasets can represent tables or volumes. If a dataset represents a volume the dataset rank must be equal to 3 and the –volume option must be given.

Tables

A dataset with rank > 1 will produce different columns in the VBT. If a dataset is a table and it has rank=3 and hyperslab offset=0,0,0 count=15,10,1000, it will produce 15*10 columns each having 1000 elements. The columns names will be datasetname_0_0 datasetname_0_1 datasetname_0_2.... datasetname_14_9.

Volume

The dataset rank must be equal to 3. If the dataset represents a volume the –volume option must be given. In this case the hyperslab dimension (if given) represents the volume dimension and the –compx –compy – compz options are ignored. More datasets can be given, but in this case they must have the same hyperslab dimension (the first hyperslab sets the volume resolution).

Examples

1. Tables

Reading three datasets from a file, the dataset1 has rank = 2, the dataset2 has rank =3 and hyperslabs are specified. The dataset 3 is totally imported.

```
fformat=hdf5
datasetlist=dataset1
datasetlist=dataset2
datasetlist=dataset3
out=filename_out.bin
hyperslab=dataset1 20,45,65 32,38,49
hyperslab=dataset2 30,45,65 32,38,49
file=myFile.h5
```

2. Volume

Reading two datasets (volumes) from a file, each dataset must have rank = 3 and the same hyperslab extension.

```
fformat=hdf5
datasetlist=vol1
datasetlist=vol2
out=filename_out.bin
hyperslab=vol1 0,0,0 50,45,50
hyperslab=vol2 100,100,100 50,45,50
volume=true
compx=50
16
compy=45
compz=50
sizex=1.0
sizey=1.0
sizez=1.0
file=myFile.h5
```

7.2.9 MUPORTAL

MUPORTAL files are expected to be in tabular form. They are produced from the experiment Muon Portal. It is an ASCII file containing rows with 10 values space separated. Each row represent an event (muon track). The column names are automatically added by the importer. The first row of the file contain the first event. The columns are typically separated by whitespace characters, e.g. spaces or tabs.

The 10 columns represent: Event number, X_A Y_B X_C Y_D X_E Y_F X_G Y_H (8 values coordinates in cm at the planes of the system), Pulse energy in GeV/C. As an example, the command below produces `NewTable.bin`, `NewTable.bin.head` from `MuPortal.in`.

Usage:

```
VisIVOImporter --fformat muportal --out /home/user/data/NewTable MuPortal.in
```

Note: The Importer automatically skips all lines starting with # character.

7.2.10 RAMSES

RAMSES files contain many type of data. This importer reads the particles positions only. The input is the root filename of a sequence of files (normally equal to the number of processes that generate the ramses output). Each file contains a set of particles (e.g. `part_seq.out00001` .... `part_seq.out00064`). The importer add .outXXXXX where XXXXX are 5 numbers in a sequence from 00001 to the number of processes that is automatically read from files.

The ramses file can have 1, 2 or 3 dimension. The generated output in any case will have 3 dimensions. In case of 1 or 2 dimensions the second and/or third dimension assume the missingvalue (–missingvalue option).

Usage:

```
$ VisIVOImporter --fformat ramses --out /home/user/data/NewTable dir/output/part_seq
```

7.2.11 RAW Binary

Raw files are simply a binary dump of the memory for data points. The content of the Raw Binary data points file is a sequence of x,y and z coordinate for each point, then a sequence of fields, one scalar for each data point.

General binary file structure:

```
1, Y1, Z1
X2, Y2, Z2
.........
Xn, Yn, Zn
Field0_1
Field0_2
........
Field0_n
Field1_1
Field1_2
........
Field1_n
........
Fieldm_1
Fieldm_2
........
Fieldm_n
```

VisIVOImporter reads a descriptor file. More than one raw data file name can be described. The descriptor file has the following structure:

- rawPointDesc
- Variable name
- Variable type
- Time Variable (at the moment not used, but needed in the descriptor file)
- Number of particles
- The size of the box
- Endianism type (b=big endian or l=little endian)
- List of Ids of the data files (a number representing the order or the time) and names of the data files

Example of Descriptor file:

```
rawPointsDesc
20
dark
Float
time
130000
50 50 50
b
0.0 16ml_096
0.5 16ml_104
1.0 16ml_112
```

All the files listed in the descriptor file must be given. Each file will be converted and an internal binary table will be created for each listed file. Output files will have the same name of –out parameter +listedfilename+".bin" and ".bin.head". If a filename start with http:// or ftp:// the remote file is downloaded. If the –userpwd option is given the username and password are used for remote access to the file. Downloaded files are temporarily copied in the same directory given in –out option and cleaned at the end of the import phase. The file downLoadedFilename_VisIVO_list in the current directory contains information on the download operations.

Usage:

```
$ VisIVOImporter --fformat rawpoints --out /home/user/data/NewTable rawpointsUserFile.
↪desc
```

7.2.12 RAW Grid

Raw files are simply a binary dump of the memory. For volume data, only one quantity is expected to be stored in each file. The content of a volume file is a sequence of values: one value for each mesh point.

VisIVOImporter reads a descriptor file. More than one raw data file name can be described. The descriptor file has the following structure:

- rawGridDesc
- Variable Name
- Number of spatial dimensions
- Variable type
- Number of cells in the first dimension
- Number of cells in the second dimension
- Number of cells in the third dimension
- Time variable (in the present release not used, but required in the descriptor file)
- Endianism type (b=big endian or l=little endian)
- List of Ids of the data files (a number representing the order or the time) and names of the data files

Example of Descriptor file:

```
rawGridsDesc
density
3
```

(continues on next page)

(continued from previous page)

```
Float
64
64
64
time
1
0.0 JET8Xhj.f064.dat.pff
0.7 JET8Xeh.f064.dat.pff
2.1 JET8Tat.f064.dat.pff
```

All the files listed in the descriptor file must be given. Each file will be converted and an internal binary table will be created for each listed file. Output files will have the same name of –out parameter +listedfilename+".bin" and ".bin.head". If a filename start with http:// or ftp:// the remote file is downloaded. If the –userpwd option is given the username and password are used for remote access to the file.

Downloaded files are temporarily copied in the same directory given in –out option and cleaned at the end of the import phase. The file `downLoadedFilename_VisIVO_list` in the current directory contains information on the download operations.

Usage:

```
$ VisIVOImporter --fformat rawgrids --out /home/user/data/NewTable rawpointsUserFile.desc
```

7.2.13 VOTable

The VOTable format is an XML standard for the interchange of data represented as a set of tables. In this context, a table is an unordered set of rows, each having a uniform format, as specified in the table metadata information. Each row in a table is a sequence of table cells, and each of these contain either a primitive data type or an array of such primitives. It can also contain a link to an external file, that the XML part describes. No VOTables with binary values are supported in VisIVO.

The file sizes that can be processed by the VisIVO Server are only limited by the underlying parsing libraries. As an example, the command below produces `NewTable.bin`, `NewTable.bin.head` from `VOTableUserFileName.xml`. This reader has no limit on VOTable size, but can read only ascii data.

Usage:

```
$ VisIVOImporter --fformat votable --out /home/user/dataNewTable.bin VOTableUserFilename.
↪xml
```

CHAPTER

EIGHT

VISIVO FILTER

VisIVO Filter is a component that converts data from a VisIVO Binary Table (VBT) into a new one or can create a Volume from a table.

To get a general help:

```
$ VisIVOFilter --help
```

To get a specific operation help:

```
$ VisIVOFilter --op <operation> --help
```

To run the operation:

```
$ VisIVOFilter --op <operation> <parameters> [--file] InputFile
```

Note: The InputFile must be a valid VBT. `InputFile.bin` and `InputFile.bin.head` must exist.

8.1 Global options

The following option can be given on any filter operation.

--memsizelimit <percentage> This option reduces the memory request of the percentage value given with this option.

Depending on the specific filter request and on the system where VisIVOFilter runs, the allocated memory could exceed the available size and the application could be aborted or can use a significant portion of the system swap area, with a dramatic lost of performance.

This parameter can be given to reduce the allocated space avoiding this effect.

A Warning message will be given when this option is used. The allowed value is a float greater than 0. and lower than 95.0.

--history Create an XML file which contains the history of operations performed.

--historyfile <file.xml> Change output history file name. Default: `hist.xml`.

8.2 Parameter file

Alternatively, to run the operation with options specified in the parameter file:

```
$ VisIVOFilter <parameterFile>
```

Lines starting with # are comments.

An example of this file is the following (for the randomizer filter):

```
op=randomizer
#memsizelimit=30        This is a commented line
perc=50.0
iseed=1
out=VBT_rand.bin
file=VBT.bin
```

8.3 Operations

The following operations are available:

- *Add ID*
- *Append*
- *Cartesian2Polar*
- *Change Column name*
- *Cut*
- *Decimator*
- *Extraction*
- *Extract List*
- *Extract Subvolume*
- *Grid2Point*
- *Include*
- *Interpolate*
- *Math Operations*
- *Merge*
- *Module*
- *Multi Resolution*
- *Poca*
- *Point Distribute*
- *Point Property*
- *Randomizer*

8.3.1 Add ID

This operation adds a new column with a sequence of Ids in the input data table.

Usage:

```
$ VisIVOFilter --op addId [--outcol col_name] [--start start_number] [--file] inputFile.
↪bin
```

Options:

--outcol	Column name of the new id column. Default name is Id.
--start	Starting Id. Default value is 0. Only an int value can be given.
--file	Input table filename.

8.3.2 Append

This operation creates a new table appending data from a list of existing tables. Append Filter can append up to 100 tables with the same number of Columns.

Usage:

```
$ VisIVOFilter --op append [--out filename_out.bin] [--filelist] table_list.txt
```

Options:

--out	Output table filename. Default name is given.
--filelist	Input filename containing the table list.

`table_list.txt` is a file that contains a list of valid table names. The ".bin" extension is automatically added if the listed filename does not contain it.

```
tab1
tab2
tab3
```

Note: The column names are copied from the first table. An error is given if tables contain different numbers of columns.

8.3.3 Cartesian2Polar

This operation creates three new fields in a data table as the result of the spherical polar transformation of three existing fields.

Usage:

```
$ VisIVOFilter --op cartesian2polar --field X Y Z [--append] [--outcol rho theta phi] [--
↪out filename_out.bin] [--file] inputFile.bin
```

Options:

--field	Three valid columns name used as cartesian coordinates.
--append	No new table will be created. The original table will have new fields. Default options: a new table with only the new field is produced.
--outcol	Column name of the new fields. Default names are: rho, theta and phi.
--out	Name of the new table. Default name is given. Ignored if –append is specified.
--file	Input table filename.

8.3.4 Change Column name

This operation changes the column names in an existing table.

Usage:

```
$ VisIVOFilter --op changecolname --field column_names --newnames new_names [--file]␣
↪inputFile.bin
```

Options:

--field	Valid columns names.
--newnames	Valid new columns names.
--file	Input table filename.

8.3.5 Cut

This operation fixes column values included in an interval to a threshold.

Usage:

```
$ VisIVOFilter --op cut [--field columns_list] --limits limitsfile.txt [--threshold␣
↪value] [--operator AND/OR] [--out filename_out.bin] [--file] inputFile.bin
```

The `limitsfile.txt` file must have the following structure. A valid column name and an interval indicating the extraction limits:

```
X 20.0 30.0
Y 10.0 20.0
Z 0.0 10.0
```

Note: The unlimited word can be used to indicate the infinite value.

Options:

--field	It is a valid columns name list to be reported in the new table. Default: all columns will be reported.
--limits	A file that has three columns: a valid column name and an interval indicating the extraction limits.
--threshold	Value to be used to cut data. Default value is 0.
--operator	Limits on all fields listed in the –limits option file are combined by default with logic AND operator. If this option is given with the OR value the field limits are combined with logic OR operator.
--out	Output table filename. Default name is given.
--file	Input table filename.

The example file and the following command:

```
$ VisIVOFilter --op cut --field A B C --limits limitsfile.txt --operator AND --out␣
↪filename_out.bin --threshold 1.0 --file inputFile.bin
```

produce a new table that contains all the data points and columns of the input table. In any row where $X \in [20.0, 30.0]$ AND $Y \in [10.0, 20.0]$ AND $Z \in [0.0, 10.0]$ the fields A B and C will be changed with the threshold value 1.0. Other fields will not be changed.

8.3.6 Decimator

This operation creates a sub-table as a regular subsample from the input table.

Usage:

```
$ VisIVOFilter --op decimator --skip step [--list parameters] [--out filename_out.bin] [-
↪-file] inputFile.bin
```

Options:

--skip	An integer that represent the number of elements to skip.
--list	Valid columns names of the input table. Default: all columns are included.
--out	Output table filename. Default name is given.
--file	Input table filename.

Values are extracted in a regular sequence, skipping step element every time. The skip value is an integer number > 1 and represents the number of skipped values. The output table must fit the available RAM.

8.3.7 Extraction

This operation creates a new table from a sub-box or a sphere.

Note: Operation not allowed on volumes.

Usage:

```
$ VisIVOFilter --op extraction --geometry geometry.txt [--out filename_out.bin] [--file]␣
↪inputFile.bin
```

Options:

--geometry The `geometry.txt` file must have four rows and two columns. The first three rows must have a valid column name and a value for each column that indicates the extraction coordinates. The fourth field means the extraction mode and the sub-volume size:

- RADIUS, a sphere centered in the given values will be extracted;
- CORNER, a rectangular region having the lower corner at the given values will be extracted;
- BOX, a rectangular region centered in the given values will be extracted.

--out Output table filename. Default name is given.

--file Input table filename.

Geometry file examples:

```
X 25.0
Y 25.0
Z 25.0
RADIUS 5.0
```

```
X 0.0
Y 0.0
Z 0.0
CORNER 10.0
```

```
X 25.0
Y 25.0
Z 25.0
BOX 5.0
```

8.3.8 Extract List

This operation creates a new table from an input table with the elements (rows) listed in a given multi-list file. A multi-list is given in ascii or binary format (unsigned long long int).

Note: Operation not allowed on volumes.

Usage:

```
$ VisIVOFilter --op extractlist --multilist filename_list [--binaryint] [--asciilist] [--
↪numberlists NL] [--listelements N0] [--onelist] [--out filename_out.bin] [--file]␣
↪inputFile.bin
```

Options:

--multilist	The multi-list file name.
--binaryint	If this parameter is specified the multi-list file is in binary int. Defalt format: binary unsigned long long int.
--asciilist	If this parameter is specified the multi-list file is an ascii text.
--numberlists	The multi-list file format is just a sequence of NL lists specified in this option. Each list starts with the number of elements in the list.
--listelements	The multi-list file format is just a sequence of NL lists. Each list has the same number of N0 elements. This option requires that the –numberlists option is specified, otherwise it is ignored.
--onelist	If this option is given, the multi-list file is considered as only one list. Each element is the ID of the particle to be extracted. The –numberlists and –listelements options will be ignored.
--out	Name of the new table. Default name is given.
--file	Input table filename.

The multi-list has the following structure:

```
Number NL of lists
NL sequences:
    1) Number N0 of elements in the list,
    2) N0 elements
```

Option can be given to provide the NL number. In this case the multi-list file must not contain this information. Option can be given to provide the N0 number. In this case the multi-list file must not contain this informations but it is a multi-list, each list must contain N0 elements.

If the onelist option is given the multi-list file is only a sequence of rows to be extracted.

Multi-list ascii file example:

```
2       # N of lists
4       # N of elments of the 1st list
1       # Start sequence of the 1st list
7
27
100     # End of the 1st list
```

(continues on next page)

(continued from previous page)

```
6       # N of elments of the 2nd list
4
8
15
16
23
42
```

8.3.9 Extract Subvolume

This operation extract a table which is a sub-volume from the original volume.

Usage:

```
$ VisIVOFilters --op extractsubvolume --startingcell X Y Z --resolution x_res y_res z_
↪res [--field column_names] [--out filename_out.bin] [--help] [--file] inputFile.bin
```

As an example the following command:

```
$ VisIVOFilters --op extractsubvolume --startingcell 8 8 8 --field Mass Temperature --
↪resolution 16 16 16 --out mysubvolume.bin --file inputFile.bin
```

produces a new table volume (`mysubvolume.bin` and `mysubvolume.bin.head`) that is a sub-volume of resolution 16x16x16 from the original volume and starting from the cell (8,8,8) of the original mesh. Only Mass and Temperature fields will be reported in the new table.

Options:

--startingcell	X Y Z number of the first cell to be extracted: 0 0 0 is the first cell of the original grid.
--resolution	Grid size (3D) of the new subgrid.
--field	Valid columns name list to be reported in the new table.
--out	Name of the new table. Default name is given.
--file	Input table filename.

8.3.10 Grid2Point

This operation distributes a volume property to a point data set on the same computational domain using a field distribution (CIC/NGP/TSC algorithm) on a regular mesh. CIC is the default adopted algorithm. The Cell geometry is considered only to compute the cell volume value in this operation.

This filter produces a new table or adds a new field to the input table.

The operation performs the following:

1. It loads a volume (input volume data table) and a table with a point distribution in the same volume;
2. It computes, using a CIC or NGP or TSC algorithm, a value (assumed density) for each data point, considering the cells value where the point is spread. The grid points density values are multiplied for the cell volume and assigned to the point. If the density option is given the cell volume is assumed =1;
3. It saves the property in a new table or adds the field to the original input table.

Usage:

```
$ VisIVOFilter --op grid2point --points x_col y_col z_col [--field column_name] [--
↪density] [--append] [--out filename_out.bin] [--outcol col_name] [--tsc] [--ngp] --
↪volume inputVolmeData.bin [--gridOrigin xg0 xg1 xg2] [--gridSpacing sg0 sg1 sg2] [--
↪box length] [--periodic] [--file] inputFile.bin
```

Options:

--points	Columns to be assumed for points coordinates.
--field	Valid Volume Column Name. Default value is the first column name.
--density	Cell volume is not considered (cell volume=1).
--append	No new table will be created. The original table will have the new field.
--out	Name of the new table. Default name is given. Ignored if –append is specified.
--outcol	Column name of the new field.
--tsc	The TSC algorithm is adopted.
--ngp	The NGP algorithm is adopted.
--volume	Input data volume filename (a VisIVO Binary Table).
--gridOrigin	It specifies the coordinate of the lower left corner of the grid. Default values are assumed from the box of `inputFile.bin`.
--gridSpacing	It specifies the length of each cell dimension in arbitrary unit. This parameter is ignored if the box option is given. Default values are assumed from the box of `inputFile.bin`.
--box	It specifies the length of a box. Default value is assumed from the box of `inputFile.bin` if the gridSpacing option is not given.
--periodic	Applies a periodical boundary condition.
--file	Input table filename with point distribution.

8.3.11 Include

This operation produces a new table or adds a new field to the input table. Points inside the sphere (given with center and radius) will have the value invalue, otherwise outvalue.

Usage:

```
$ VisIVOFilter --op include --center x_coord y_coord z_coord --radius radius [--field x_
↪col y_col z_col] [--append] [--out filename_out.bin] [--outcol col_name] [--outvalue␣
↪outvalue] [--invalue invalue] [--file] inputFile.bin
```

Options:

--center	Coordinates of the sphere center.
--radius	Radius of the sphere.
--field	Three valid columns names. Default values are the first three columns.
--append	No new table will be created. The original table will have the new field.
--out	Name of the new table. Default name is given. Ignored if –append is specified.

--outcol	Column name of the new field.
--outvalue	Value given to points outside the sphere. Default value is 0.
--invalue	Value given to points inside the sphere. Default value is 1.
--file	Input table filename.

8.3.12 Interpolate

This operation creates new tables from two existing data tables (mainly used to produce intermediate frames of a dynamical evolution).

Usage:

```
$ VisIVOFilter --op interpolate [--field columns_name] [--numbin numberbin] [--periodic]␣
↪[--interval from to] [--out filename_out] --infiles file_start.bin file_end.bin
```

Note: The two table must have the same structure. The infiles tables must have the listed columns in the –field option in the same corresponding order. The input tables must have the same number of rows and the interpolated elements are considered in the same order. No index is currently supported.

Options:

--field	A valid list of columns names that must exist on both input tables. Default: all columns in infile files are considered.
--numbin	Is the number of bins between the starting and ending input files or the interval given in the –interval option. The default value is 10. The number of created tables is equal to numberbin-1.
--periodic	Applies a periodical boundary condition.
--interval	VisIVO assumes a distance of 1.0 between the starting frame and ending frame. This option produces the intermediate frames (tables) in a subinterval between the two input frames. The value 0.5 is the medium point of the interval. If the from value is lower than 0.0 it is considered 0.0. If the to value is greater than 1.0 it is considered 1.0. If the from value is equal to to value the operation is not performed. Default value from=0.0 to=1.0
--out	It is the root name of the new tables. The default name is given. The new name is given by the `filename_out#.bin` where # is the number of created tables.
--infiles	It contains the names of the input tables of the interpolation process.

8.3.13 Math Operations

The operation creates a new field in a data table as the result of a mathematical operation between the existing fields.

Usage:

```
$ VisIVOFilter --op mathop [--expression math_expression.txt] [--compute <<expression>>]␣
↪[--append] [--outcol col_name] [--out filename_out.bin] [--file] filename.bin
```

Options:

--expression A file with only one row having any valid mathematical expression with Valid Column names. Ignored if compute option is given.

--compute A valid mathematical expression with Valid Column names. The expression must start with << and finish with >> characters. It has the priority on the expression option.

The expression must contain the escape character control for the << and >> symbols and the parentheses. For example, to evaluate $(A/B) * C$ the correct syntax will be `--compute \<\<\(A/B\)*C\>\>`.

Note: The << , >> and escape characters must not be given if the parameter file is used.

--append No new table will be created. The original table will have the new field. Default options: a new table with only the new field is produced.

--outcol Column name of the new field

--out Output table filename. Default name is given. Ignored if –append is specified.

--file Input table filename.

`math_expression.txt` is a file that contains only one row with a mathematical expression, for example:

```
sqrt(VelX*VelX+VelY*VelY+VelZ*VelZ)
```

Arithmetic float expressions can be created from float literals, variables or functions using the following operators in this order of precedence:

()	expressions in parentheses first
A unit	a unit multiplier (if one has been added) exponentiation (A raised to the power B)
A^B	exponentiation (A raised to the power B)
-A	unary minus
!A	unary logical not (result is 1 if int(A) is 0, else 0)
A*B A/B A%B	multiplication, division and modulo
A+B A-B	addition and subtraction
A=B A!=B A<B A<=B A>B A>=B	comparison between A and B (result is either 0 or 1)
A&B	result is 1 if int(A) and int(B) differ from 0, else 0
A|B	result is 1 if int(A) or int(B) differ from 0, else 0

Since the unary minus has higher precedence than any other operator, the following expression is valid: `x*-y`.

The comparison operators use an epsilon value, so expressions which may differ in very least-significant digits should work correctly.

The following operations can be used:

abs(A)	Absolute value of A. If A is negative, returns -A otherwise returns A.
acos(A)	Arc-cosine of A. Returns the angle, measured in radians, whose cosine is A.
acosh(A)	Same as acos() but for hyperbolic cosine.
asin(A)	Arc-sine of A. Returns the angle, measured in radians, whose sine is A.
asinh(A)	Same as asin() but for hyperbolic sine.
atan(A)	Arc-tangent of (A). Returns the angle, measured in radians, whose tangent is (A).
atan2(A,B)	Arc-tangent of A/B. The two main differences to atan() is that it will return the right angle depending on the signs of A and B (atan() can only return values between -pi/2 and pi/2), and that the return value of pi/2 and -pi/2 are possible.
atanh(A)	Same as atan() but for hyperbolic tangent.
ceil(A)	Ceiling of A. Returns the smallest integer greater than A. Rounds up to the next higher integer.
cos(A)	Cosine of A. Returns the cosine of the angle A, where A is measured in radians.
cosh(A)	Same as cos() but for hyperbolic cosine.
cot(A)	Cotangent of A (equivalent to 1/tan(A)).
csc(A)	Cosecant of A (equivalent to 1/sin(A)).
exp(A)	Exponential of A. Returns the value of e raised to the power A where e is the base of the natural logarithm, i.e. the non-repeating value approximately equal to 2.71828182846.
floor(A)	Floor of A. Returns the largest integer less than A. Rounds down to the next lower integer.
if(A,B,C)	If int(A) differs from 0, the return value of this function is B, else C. Only the parameter which needs to be evaluated is evaluated, the other parameter is skipped; this makes it safe to use eval() in them.
int(A)	Rounds A to the closest integer. 0.5 is rounded to 1.
log(A)	Natural (base e) logarithm of A.
log10(A)	Base 10 logarithm of A.
max(A,B)	If A>B, the result is A, else B.
min(A,B)	If A<B, the result is A, else B.
sec(A)	Secant of A (equivalent to 1/cos(A)).
sin(A)	Sine of A. Returns the sine of the angle A, where A is measured in radians.
sinh(A)	Same as sin() but for hyperbolic sine.
sqrt(A)	Square root of A. Returns the value whose square is A.
tan(A)	Tangent of A. Returns the tangent of the angle A, where A is measured in radians.
tanh(A)	Same as tan() but for hyperbolic tangent.

8.3.14 Merge

This operation creates a new table from two or more existing data tables. Up to 100 tables can be merged. Volumes can be merged but they must have the same geometry.

Usage:

```
$ VisIVOFilter --op merge  [--size HUGE/SMALLEST] [--pad value] [--out filename_out.bin]␣
↪[--filelist] tab_selection_file.txt
```

Options:

--size Produce a new table having the size of the smallest (or larger) table. Default option: SMALLEST.

--pad	Pad the table rows of smaller table with the given value if HUGE size is used. Default value: 0.
--out	Output table filename. Default name is given.
--filelist	Input filename containing the table list.

The `tab_selection_file.txt` is a file that contain a list of tables and valid columns. Wildcard "*" means all the columns of the given table. An example file is the following:

```
tab1.bin Col_1
tab1.bin Col_2
tab5.bin Col_x
tab4.bin *
```

This file a new table having columns Col_1 and Col_2 from `tab1.bin`, Col_x from `tab5.bin` and all the columns of `tab4.bin`.

8.3.15 Module

This operation creates a new table (or a new field) computing the module of three fields of the input data table.

Usage:

```
$ VisIVOFilter --op module --field parameters [--append] [--outcol colname] [--out␣
↪filename_out.bin] [--file] inputFile.bin
```

Options:

--field	Three valid columns name lists used to compute the module.
--append	No new table will be created. The original table will have the new field. Default options: a new table with only the new field is produced.
--outcol	Column name of the new field.
--out	Name of the new table. Default name is given. Ignored if –append is specified.
--file	Input table filename

Example:

```
$ VisIVOFilter --op module --field x y z --outcol Module --append --file inputFile.bin
```

This command appends a new field named Module to the `inputFile.bin` file that represents the module of the fields x, y and z: $\sqrt{x^2 + y^2 + z^2}$.

8.3.16 Multi Resolution

This operation creates a new VBT as a random subsample from the input table, with different resolutions.

Starting from a fixed position, that represents the center of inner sphere, concentric spheres are considered. Different level of randomization can be given, creating more detail table in the inner sphere and lower detail in the outer regions, or vice versa. The region that is external to the last sphere represents the background.

Note: Operation not yet allowed on volumes.

Usage:

```
$ VisIVOFilter --op mres --points x_col y_col z_col [--pos values] [--geometry layer_
↪file.txt] [--background value] [--out filename_out.bin] [--file] inputFile.bin
```

Options:

--points	Columns to be assumed for points coordinates.
--pos	Camera point coordinates. Default value is the center of the domain.
--geometry	A file that contains a radius and a randomization value: 1.0 all values included; 0.1 means 1 per cent of values included in the layer. Each row of this file determine a layer. A default geometry is created with three spheres and different levels of randomization, depending on the input dataset.
--background	A randomizator value for points outside the geometry. Default value is maximum 100000 values from input VBT.
--out	Name of the new table. Default name is given.
--file	Input table filename.

For example, the following command and file:

```
$ VisIVOFilter --op mres --points X Y Z --pos 10.0 10.0 10.0 --geometry layer.txt --
↪background 0.0001 --out pos_layer.bin --file pos.bin
```

```
5.0 1.0
10.0 0.1
30.0 0.01
```

produce a new table.

The geometry file in this example, has the following values:

- 5.0 is the radius of the inner sphere. The 1.0 is the percentage of randomization inside the inner sphere: all points that are inside the inner sphere will be reported in the output VBT.
- 10.0 is the radius of the second sphere. The value 0.1 is the percentage of randomization inside this sphere: only 10% of points that are inside this sphere will be reported in the output VBT.
- 30.0 is the radius of the last sphere. The value 0.01 is the percentage of randomization inside this sphere: only 1% of points that are inside this sphere will be reported in the output VBT.

8.3.17 Poca

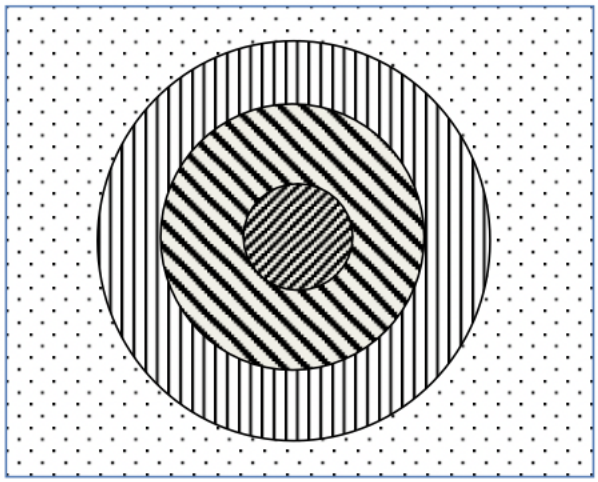
This operation produces two tables. The first one contains scattering points and angles. The second is a volume: each mesh point contains the square sum of the scattering angle.

The input file must be the output from the muportal importer with 10 column: Event number, X_A Y_B X_C Y_D X_E Y_F X_G Y_H (8 values coordinates in cm at the planes of the system), Energy pulse in GeV/C.

Usage:

```
$ VisIVOFilter --op poca [--resolution x_res y_res z_res] [--dimvox voxel_size] [--
↪trackplanedist distance] [--innerdist distance] [--outpoints points.bin] [--outvol vol.
↪bin] [--file] inputFile.bin
```

Options:



--resolution	3D mesh size in cm. Default value 600 300 300.
--dimvox	Cubic voxel dimension in cm. Default value 10.
--trackplanedist	Distance in cm between planes 1 - 2 and planes 3 - 4. Default value 100.
--innerdist	Distance in cm between planes 3 - 4. Default value 300.
--outpoints	Name of the new table containing poca points.
--outvol	Name of the new table containing the volume having the theta square value sum in each voxel.
--file	Input table filename.

For example, the following command:

```
$ VisIVOFilter --op poca --resolution 600 300 300 --dimvox 10 --outpoins points --outvol␣
↪vol.bin --file inputFile.bin
```

Produces a points file VBT with scatter points inside the volume of reference a 60x30x30 and a multi-volume.

The input file is a VBT with 10 columns that represent: Event number, X_A Y_B X_C Y_D X_E Y_F X_G Y_H (8 values coordinates in cm at the planes of the system), Pulse energy in GeV/C.

The filter gives the track Id and the Energy of each points X Y Z. This is the output of the Importer “muportal”. It produces two tables (`points.bin` and `vol.bin`) containing:

- The points of the POCA algorithm and angles with the following fields: event number, X, Y and Z, theta, theta square, energy.
- A multi-volume (7 volumes) with cell resolution/dimvox on each directiom. Each voxel contatins quantities of all scattering points inside the voxel: sum of the theta (scattering angle), sum of theta square, theta square average, sigma, error and number of scatter points.

8.3.18 Point Distribute

This operation creates a table which represents a volume from selected fields of the input table that are distributed using NGP, CIC (default) or TSC algorithm. The filter produces (by default) a density field: the field is distributed and divided for the cell volume.

Usage:

```
$ VisIVOFilter --op pointdistribute  --resolution x_res y_res z_res --points x_col y_col␣
↪z_col [--field column_names] [--constant value] [--nodensity] [--avg] [--out filename_
↪out.bin] [--tsc] [--ngp] [--gridOrigin xg0 xg1 xg2] [--gridSpacing sg0 sg1 sg2] [--box␣
↪length] [--periodic] [--file] inputFile.bin
```

Options:

--resolution	3D mesh size.
--points	Columns to be assumed for points coordinates.
--field	Valid columns name list to be distributed in the grid.
--constant	Assign a constant value to all points, to be distributed in the grid. Ignored if field option is given. Default value is 1.0 for all points.
--nodensity	Overrides the default behavior. The field distribution is not divided for the cell volume.
--avg	Distributes the first field on the volume grid and computes the arithmetic average value on each cell of the first field. The output volume table will have three fields. For each cell there will be: the number of total elements in the cell (NumberOfElements), the sum of total field value (fieldSum), and the arithmetic average value (fieldAvg). Only the ngp algorithm will be applied.
--out	Name of the new table. Default name is given.
--tsc	The TSC algorithm is adopted.
--ngp	The NGP algorithm is adopted.
--gridOrigin	It specifies the coordinates of the lower left corner of the grid. Default values are assumed from the box of `inputFile.bin`.
--gridSpacing	It specifies the length of each cell dimension in the arbitrary unit. This parameter is ignored if the box option is given. Default values are assumed from the box of the `inputFile.bin`.
--box	It specifies the length of a box. Default values is assumed from the box of `inputFile.bin` if the gridSpacing option is not given.
--periodic	It specifies the box is periodic. Particles outside the box limits are considered inside on the other side.
--file	Input table filename.

8.3.19 Point Property

This operation assigns a property to each data point on the table.

The operation performs the following:

1. It creates a temporary volume using a field distribution (CIC algorithm) on a regular mesh. The temporary file will have the filename given in –out option + `_tempPDOp.bin` or `_tempPDOp.bin`, and it is automatically cleaned by the operation itself;
2. It computes, with the same CIC algorithm, the property for each data point, considering the cells where the point is spread on the volume;
3. It saves the property in a new table or adds the field to the original input table. This operation cannot be applied to volumes.

Usage:

```
$ VisIVOFilter --op pointproperty --resolution x_res y_res z_res --points x_col y_col z_
↪col [--field column_name] [--constant value] [--append] [--out filename_out.bin] [--
↪outcol col_name] [--periodic] [--file] inputFile.bin
```

Options:

--resolution	3D mesh size.
--points	Columns to be assumed for points coordinates.
--field	valid column name list to be distributed in the grid.
--constant	Assign a constant value to all points, to be distributed in the grid. Ignored if field option is given. Default value is a 1.0 for all points.
--append	the input table will contain the new field.
--out	Name of the new table. Default name is given. Ignored if –append is given.
--outcol	New field column name.
--periodic	Applies a periodical boundary condition.
--file	Input table filename.

For example, the following command:

```
$ VisIVOFilter --op pointproperty --resolution 16 16 16 --points X Y Z --field Mass --
↪append --outcol distribute --file inputFile.bin
```

distributes the Mass field of points X Y Z and it produces a temporary volume. Then it calculates a new field representing the weight value of the nearest mesh-cells where the point is distributed using the same CIC algorithm.

8.3.20 Randomizer

This operation creates a random subset from the original data table.

Usage:

```
$ VisIVOFilter --op randomizer --perc percentage [--field parameters] [--iseed iseed] [--
↪out filename_out.bin] [--file] inputFile.bin
```

Options:

--perc Percentage (from 0.0 to 100.0) of the input file obtained in the output file.

> **Note:** Only the first decimal place is considered.

--field Valid columns names of the input table. Default: all columns are included.

--iseed Specify the seed for the random generation. Default value 0.

--out Output table filename. Default name is given.

--file Input table filename.

8.3.21 Select Columns

This operation creates e new table using (or excluding) one or more fields of a data table. The default case produces the output table including only listed fields.

Usage:

```
$ VisIVOFilter --op selcolumns --field parameters [--delete] [--out filename_out.bin] [--
↪file] inputFile.bin
```

Options:

--field Valid columns names of the input table. Default: all columns are included.

--delete Produce output table excluding only field listed in the –field option.

--out Output table filename. Default name is given.

--file Input table filename.

8.3.22 Select Fields

This operation creates a new table setting limits on one or more fields of a data table. Optionally it creates a list of elements satisfying the requested condition.

Usage:

```
$ VisIVOFilter --op selfield  --limits limitsfile.txt [--operator AND/OR] [--outlist␣
↪list_filename] [--format uns/int/ascii] [--out filename_out.bin] [--file] inputFile.bin
```

Options:

--limits A file that has three columns: a valid column name and an interval indicating the extraction limits.

--operator Limits on all fields listed in –limits option file are combined by default with logic AND operator. If this option is given with OR value the field limits are combined with logic OR operator.

--outlist Output list filename containing the number of the elements satisfying the requested condition. Default name is given.

--format Data format in the outlist filename. Default value unsigned long long int.

--out Output table filename. Default name is given.

--file Input table filename.

The `limitsfile.txt` file must have the following structure. A valid column name and an interval indicating the extraction limits:

```
X 20.0 30.0
Y 10.0 20.0
Z 0.0 10.0
```

This file produces a new table that contains all the data points of the input table (all columns will be reported) where $X \in [20.0, 30.0]$ AND $Y \in [10.0, 20.0]$ AND $Z \in [0.0, 10.0]$.

Note: The unlimited word can be used to indicate the infinite value.

8.3.23 Show Table

Produce an ASCII table with selected field of the first number of rows as specified in the –numrows parameter.

Usage:

```
$ VisIVOFilter --op showtable [--field column_name] [--numrows num_of_rows] [--rangerows␣
↪fromRow toRow] [--width format_width] [--precision format_precision] [--out filename_
↪out.txt] [--file] inputFile.bin
```

Options:

--field	Valid columns names. Default value of all columns will be reported.
--numrows	Number of rows in the ASCII output file. Default value is equal to the number of rows of the input table.
--rangerows	Rows range of the inputFile that will be reported in the ASCII output file. Default range is equal to all the rows of the input table. It is ignored if numrows is specified.
--width	Field width in the ASCII output file. Default value is given.
--precision	Field precision in the ASCII output file. Default value is given.
--out	Output ASCII filename. Default name is given.
--file	Input table filename.

8.3.24 Show Volume

This operation writes on an output ascii file, the volume cells and their values that satisfy given limits.

Usage:

```
$ VisIVOFilter --op showvol [--field column_name] --limits limits.txt [--operator AND/
↪OR] [--numcells value] [--out filename_out.txt] [--file] inputFile.bin
```

The `limits.txt` file must have the following structure: a valid column name and an interval that indicate the limits.

```
density 20.0 30.0
mass 10.0 20.0
```

Note: The keyword unlimited can be used to indicate the infinite value.

Options:

--field	Valid columns names. Default value of ALL columns will be reported.
--limits	A file that has three columns: a valid column name and an interval that indicate the limits.
--operator	Limits on all field listed in the –limits option file are combined by default with logic AND operator. If this option is given with OR value the field limits are combined with logic OR operator.
--numcells	Set the maximum number of cells that will be reported in the output.
--out	Output ascii filename. Default name is given.
--file	Input table filename.

The example file and the following command:

```
$ VisIVOFilter --op showvol --field density mass --limits limitsfile.txt --operator AND -
↪-out filename_out.txt --file inputFile.bin
```

produce an ascii file that lists all cells where limits are satisfied.

The output ascii file will report on each row the cell (X, Y and Z) and the field value.

8.3.25 Sigma Contours

This operation creates a new table where one or more fields of a data table have values within (or outside) N sigma contours. For the selected fields, the filter prints in the stdout the average and the sigma values of the distributions.

Note: The filter can be applied on fields that have a Gaussian distribution.

Usage:

```
$ VisIVOFilter --op sigmacontours [--nsigma nOfSigma] [--field columns_list] [--exclude]␣
↪[--allcolumns] [--out filename_out.bin] [--file] filename.bin
```

Options:

--nsigma	Number of sigma used in the variable selection. Default value: 1 sigma countour.
--field	List of columns that must have values within N sigma contours. Default value: all the columns in the table.
--exclude	Values outside N sigma contours will be reported.
--allcolumns	All columns of the input tables will be saved in the output table. But only the corresponding rows for the –field selected columns are reported.
--out	Name of the new table. Default name is given.
--file	Input table filename.

Example:

```
$ VisIVOFilter --op sigmacontours --nsigma 2 --field F1 F2 F5 --allcolumns --out␣
↪ncontours.bin --file example.bin
```

The command produces a new table where columns F1, F2 and F5 have (all of them) values included in 2 sigma contours. The option –allcolumns creates a new table with all the columns of the input table, otherwise only F1, F2 and F5 will be reported in the output table. The command also prints in the stdout the average values and the sigma values for F1, F2 and F5 columns.

8.3.26 Split Table

This operation splits an existing table into two or more tables, using a field that will be used to divide the table.

Usage:

```
$ VisIVOFilter --op splittable [--field column] [--volumesplit direction] [--numoftables␣
↪numberOfTable] [--maxsizetable MaxMbyteSize] [--hugesplit] [--out filename_out.bin] [--
↪file] inputFile.bin
```

Options:

--field	A valid column name along which the table will be split. Must be given to split a table.
--volumesplit	Direction (1, 2 or 3) along which the volume will be split. Must be given to split a volume.
--numoftables	The number of tables in which the input table will be split. It must be greater than 1.
--maxsizetable	Indicates the maximum size of the split table. VisIVO Filter will compute how many tables will be created. This option is ignored if –numoftable option is given.
--hugesplit	Must be given to force the generation of more than 100 tables from the input table, avoiding errors.
--out	Output prefix root table filename. A suffix `_split_#.bin` is added to each generated table, to this prefix. Default name is given.
--file	Input table filename.

8.3.27 Statistic

This operation produces average, min and max value of field and creates an histogram of fields in the input table.

Usage:

```
$ VisIVOFilter --op statistic [--list columns_name] [--histogram [bin]] [--range min␣
↪max] [--out result.txt] [--file] inputFile.bin
```

Options:

--list	A valid list of columns name. Default value all columns.
--histogram	Produces an histogram ASCII file with the given number of bins. If the bins number is not specified, the default value is fixed to 10% of the total rows of the input table.

--range	Produces the results only inside the specified interval.
--out	Output ASCII filename with histogram.
--file	Input table filename.

Note: An error is given if there are no data in the specified range.

For example the following command:

```
$ VisIVOFilter --op statistic --list X Y --histogram 1000 --range 10.0 100.0 --out␣
↪result.txt --file inputFile.bin
```

produces min,max and average values printed in the standard output. The command also produces a `result.txt` file that gives the histogram values of X and Y in the range $[10.0, 100.0]$ with 1000 bins.

8.3.28 Swap

This operation produces the Endianism swap: little Endian to big Endian data format and viceversa of a VisIVO Binary Table. It can produce a new swapped table or data can be overwritten.

Usage:

```
$ VisIVOFilter --op swap [--override] [--out filename_out.bin] [--file] inputFile.bin
```

Options:

--override	The same input table is swapped. Data are overwritten.
--out	Name of the new table. Default name is given. Ignored if –override is specified.
--file	Input table filename.

8.3.29 Visual

This operation creates an eventually randomized new table from one or more input tables. All the input tables must have the same number of rows.

Note: The operation cannot be applied to volume tables.

Usage:

```
$ VisIVOFilter --op visual [--size number_of_elements] [--out filename_out.bin] [--
↪filelist] tab_selection_file.txt
```

Options:

--size	Number of max rows in output table. Default is the minimum between 8000000 and the number of rows of input tables. **Note:** Input table must have the same number of rows.
--out	Output table filename. Default name is given.

--filelist Input text file with a list of tables and columns.

The `tab_selection_file.txt` is a text file that contain a list of tables and valid columns. Wildcard "*" means all columns of the given table. For example:

```
tab1.bin Col_1
tab5.bin Col_x
tab4.bin *
tab1.bin Col_2
```

This file produces a new table having columns Col_1 and Col_2 from tab1.bin, Col_x from tab5.bin and all the columns of tab4.bin. If the row number of the input tables exceeds 8000000 elements, the output file will be limited to 8000000 randomized sampled rows.

The column names in the output table will have the suffix `_visual_#` where # represent the number order listed in the txt file. The output table will contain the columns of the listed tables in alphabetic order. In the above example, the header of the VBT (if tab4.bin contains two columns A and B and there are 8000000 rows) will be:

```
float
5
8000000
little
Col_1_visual_1
Col_2_visual_5
A_visual_3
B_visual_4
Col_x_visual_2
```

8.3.30 Write VOTable

This operation produces a VOTable 1.2 with selected field.

Usage:

```
$ VisIVOFilter --op wrvotable [--field column_name] [--force] [--out filename_out.xml] [-
↪-file] inputFile.bin
```

Options:

--field Valid columns names. Default values of ALL columns will be reported.

--force Must be given to force the VOTable creation when the input table has more than 1000000 rows.

--out Output ascii filename. Default name is given.

--file Input table filename.

CHAPTER

NINE

VISIVO VIEWER

VisIVOViewer creates views from the input data file. The input data file must be in the VBT format. The input data file must fit the available RAM.

VisIVOViewer produces five png images. The first four default images are generated with the following fixed values:

```
Azimuth     0   90  0   45
Elevation   0   0   90  45
Zoom        1
```

The last image is given with values fixed by the user.

Note: The camera of the Viewer at the default position (Azimuth=0, Elevation=0) is looking the box from the top (z plane).

To get a general help:

```
$ VisIVOViewer --help
```

To run the Viewer:

```
$ VisIVOViewer <options> inputFile
```

9.1 Global options

--out <filename> Default output filenames are `VisIVOServerImage0.png`, `VisIVOServerImage1.png`, ...

Note: The output name is always completed with `0.png`, `1.png`, `2.png` and `3.png` if default images are produced. The output file name (image) is completed with `.png` extension.

--nodefault Default images are not created.

--cycle <filename> (Optional) VisIVO Viewer will produce a sequence of images reading data of azimuth elevation and zooming from the file given with this parameter. This option will prevent the production of default images while the camera position and zoom factor of the line command will be ignored.

The file for the cycle must contain three ascii values for each row. Blank line or rows with less than three values will be ignored. VisIVO will produce one image for each valid row in the file. This file can be created with the tool VisIVOUtils. The output filenames will have the same root-name as those given in –out parameter plus a progressive number (with six digit) starting from 0 (or the value given in the –cycleoffset option), with png file extension. A new file listing all images is created. The file has the fixed name `VSCycleImage#.txt` in the output director, where # is the value of –cycleoffset option.

The cycle file for camera position must be of four types:

- Four values for each line: Azimuth, Elevation, Zoom and Roll;
- Seven values for each line: Azimuth, Elevation, Zoom, Focal Point (three values) and Roll;
- Eight values for each line: Zoom, Camera Position (three values), Focal Point (three values) and Roll;
- Ten values for each line: Azimuth, Elevation, Zoom, Camera Position (three values), Focal Point (three values) and Roll;

The keyword NULL is accepted for Camera position and/or Focal Point and/or Roll. In this case the previous camera setting is maintained. The NULL keyword cannot be used for Azimuth, Elevation and Zoom.

Examples:

Azimuth, Elevation, Zoom and Roll

```
0.0 0.0 1.0 0
.....
60.0 10.0 1.0 NULL
60.0 10.0 1.2 NULL
60.0 10.0 1.4 NULL
60.0 10.0 1.6 NULL
```

Azimuth, Elevation, Zoom, Focal Point and Roll

```
0.0 0.0 1.0 35 35 35 0
.....
60.0 10.0 1.0 20 20 20 0
60.0 10.0 1.2 20 20 20 0
60.0 10.0 1.4 20 20 20 0
60.0 10.0 1.6 20 20 20 0
```

Zoom, Camera Position, Focal Point and Roll

```
1.0 35 35 200 35 35 35 0
.....
1.0 35 35 70 NULL NULL NULL 60
```

Azimuth, Elevation, Zoom, Camera Position, Focal Point and Roll

```
0 0 1.0 35 35 200 35 35 35 0
10 0 1.0 35 35 200 35 35 35 0
10 0 1.5 35 35 200 35 35 35 0
```

Note: Fixing the camera position, the azimuth and the elevation are computed starting from the camera position set.

If slices are required, the cycle file must contain the sequence of positions in the volume of the planes. This file can be created with the tool VisIVOUtils.

--cycleoffset <value> (Optional) The value for the progressive number of files produced with the cycle option. The default value is 0.

--cycle_skip_from <value> (Optional) Skips the first number of lines in the cycle file.

--cycle_skip_to <value> (Optional) Reads up to the given line in the cycle file.

--camazim <double> (Optional) Fixes the camera azimuth position from the camera position.

--camelev <double> (Optional) Fixes the camera elevation position from the camera position.

The allowed range for the camera elevation is [-90, 90]. If the given camera elevation value is out of the valid range, the elevation is set at the boundary. E.g.: –camelev 100 will automatically be changed with –camelev 90.

--zoom <double> (Optional) Zooming factor. A value greater than 1 is a zoom-in, a value less than 1 is a zoom-out (–zoom 2.0).

--camfov <value> (Optional) Fixes the zooming factor.

--campos <value> (Optional) Fixes the camera position in the system coordinate: 3 values (x, y and z) must be given.

--camfp <value> (Optional) Fixes the focal point position in the system coordinate: 3 values (x, y and z) must be given. The default focal point is the center of the system.

--camroll <double> (Optional) Fixes the camera roll factor.

--imagesize (Optional) Fixes the size of the image. It may assume the following values: small, medium, large. Default value is medium.

--backcolor (Optional) Fixes the background color. It may assume one of the following values: yellow, red, green, blue, white, black, cyan, violet. Default value is black.

--onecolor (Optional) Fixes the color of points and isosurfaces (ignored if –-color option is given). It may assume one of the following values: yellow, red, green, blue, white, black, cyan, violet. Default color is white.

--color (Optional) Uses the palette.

--colortable <name> (Optional) Selects the palette (–colortable default) or (–colortable paletteFilename).

The following predefined palette can be given: default, default_step, efield, glow, gray, min_max, physics_contour, pure_red, pure_green, pure_blue, run1, run2, sar, temperature, tensteps, volren_glow, volren_green, volren_rg, volren_twolevel, all_yellow, all_red, all_green, all_blue, all_white, all_black, all_cyan, all_violet.

Note: If the –colortable option does not contain a predefined palette, VisIVOViewer assumes that an external filename is given as for this parameter. The file must exist in the current directory or the path must be specified. The palette is an ASCII file. The table must contain [Id+] RGB [+A] or Id+HSV+A comma or space separated values. HSV values are converted into RGB. The Id is an integer

that represents the number of points in the palette. RGB (or HSV) and A must be given in [0.0, 1.0] range.

The palette file can have one of the following formats:

- Only RGB values (or HSV) are given. Opacity (A) is equal to 1.0

```
0.2 0.1 0.7
0.7 0.5 1.0
0.4 1.0 0.2
```

The palette table will have the number of points (3 in this case) equal to the number of rows.

- RGB+A values (or HSV+A) are given.

```
0.2 0.1 0.7 0.2
0.7 0.5 1.0 0.8
0.4 1.0 0.2 1.0
```

The palette table will have the number of points (3 in this case) equal to the number of rows.

- Id+RGB+A values (or id+HSV+A) are given.

```
0 0.2 0.1 0.7 0.2
100 0.7 0.5 1.0 0.8
299 0.4 1.0 0.2 1.0
```

The palette table will have the number of points equal to the last Id+1 (300 in this case).

Intermediate values (not given) are automatically generated with a linear interpolation between the given values. The table must have increasing Ids. Tables with not ordered Ids are discarded. If the starting 0 point is not given (first row) it is assumed (by default) given as follows:

```
0 0.0 0.0 0.0 0.0
```

The first row indicates the table RGB or HSV. In the case of the RGB table this row could not be given.

--colorrangefrom (Optional) Sets the lower limit of the palette.

--colorrangeto (Optional) Sets the upper limit of the palette.

--stereo (Optional) Produces stereoscopic images. May assume the following values:

- RedBlue, it produces an image for use with red-blue glasses;
- CrystalEyes, it uses frame-sequential capabilities available in OpenGL to drive LCD shutter glasses and stereo projectors; it produces two images with suffixes _r and _l for the Right and Left eyes.
- Anaglyph, it is a superset of RedBlue mode, but the color output channels can be configured using the anaglyphmask option and the color of the original image can be (somewhat) maintained using anaglyphsat option.

The default colors for Anaglyph mode is red-cyan. Stereoscopic visualization option is ignored if the slice view is required.

--anaglyphsat (Optional) Sets the anaglyph color saturation factor. This number ranges from 0.0 to 1.0: 0.0 means that no color from the original object is maintained, 1.0 means all of the color is maintained. The default value is 0.65.

Note: Too much saturation can produce uncomfortable 3D viewing.

--anaglyphmask (Optional) Sets the anaglyph color mask values. These two numbers are bits mask that control which color channels of the original stereo images are used to produce the final anaglyph image. The first value is the color mask for the left view, the second the mask for the right view. If a bit in the mask is on for a particular color, that color is passed on to the final view; if it is not set, that channel for that view is ignored. The bits are arranged as r, g, and b, so r = 4, g = 2, and b = 1. By default, the first value (the left view) is set to 4, and the second value is set to 3.

--showlut (Optional) visualizes the colorbar.

--showbox (Optional) visualizes the box.

--showaxes (Optional) visualizes the axes.

--cliplarge (Optional) Fixes the clipping plane from 0 to 1.0e+13. The cliplarge option can be used if a black image is obtained. To fix specific range use the cliprange option. This option is ignored if cliprange is given.

--cliprange <value> (Optional) Fixes the clipping plane in the system coordinate: 2 values must be given (i.e. 0.0 1.0e+4). The user can set this values using the statistic filter and checking the user data value in the field of view.

--history (Optional) create an XML file which contains the history of operations performed. Default create `hist.xml` file.

--historyfile <filename> (Optional) Change default history file name and or directory.

9.2 Parameter file

This alternative command allows VisIVOViewer to read all options from a parameter file. Lines starting with # are comments.

An example of this file is the following:

```
############## VisIVO SECTION
######## Sect 1 General
volume=no
vector=no
input=u2.bin
out=outFilename
showbox=yes
showaxes=no
imagesize=medium
#cycle=cicleFilename
#cycleoffset=0
#cycle_skip_from=0
#cycle_skip_to=0
######## Sect 2 Points and vectors
x=X
```

(continues on next page)

(continued from previous page)

```
y=Y
z=Z
#vx=VX
#vy=VY
#vz=VZ
#scale=yes
#######
########### Sect 3 Volume
#vrendering=yes
#isosurface=no
#slice=no
#shadow=no
#vrenderingfield=ColumnName
#slicefield=ColumnName
#sliceplane=x
#sliceposition=0
#sliceplanenormal= 1 1 1
#sliceplanepoint= 10 10 10
#isosurfacefield=ColumnName
#isosurfacevalue=120
#wireframe=no
#isosmooth=none
################## Sect 4 Camera
camazim=20
camelev=20
campos= 35 35 200
camfp= 35 35 35
zoom=1.5
nodefault=yes
#largeimage=no
################## Sect 5 Colour
color=yes
colorscalar=X
colortable=default
#colorrangefrom=0
#colorrangeto=100
#onecolor=white
#backcolor=black
#showlut=yes
opacity=0.666
#logscale=no
########### Sect 5 Glyphs
#glyphs=sphere
#scaleglyphs=no
#radius=1.0
#radiusscalar=ColumnName
#height=1.0
#heightscalar=ColumnName
#vectorline=yes
#vectorscalefactor=1.0
#vectorscale=1
```

9.3 Visualization

The following kinds of visualizations are available:

9.3.1 Data Points

VisIVO Viewer creates data points views from the input data file. The Input data file must be in VBT format. The input data file must fit the available RAM.

Options:

-x <field> (Optional) Selects the first coordinate.

-y <field> (Optional) Selects the second coordinate.

-z <field> (Optional) Selects the third coordinate.

--scale (Optional) Enables data normalization. It always allows you to visualize a cubic region even if the coordinates system has different scales. The field names containing X,Y,Z or RA,DE and Mag are assumed to be default values for the x y z system, or the first three table columns, if these options are not given.

Note: It is strongly recommended to fix these parameters to prevent unpredictable behavior.

--colorscalar <field> (Optional) Selects the field for the palette.

--logscale (Optional) Uses the logarithmic scale for the palette. If the select field has values less than or equal to 0 this option is ignored and the linear scale will be used.

--glyphs <name> (Optional) Data points are displayed with different geometrical form. The following forms are available: pixel, sphere, cone, cylinder, cube. This option has no effect if the data point number is more than 1000.

--radius <vaulue> (Optional) Radius of the geometrical form.

--height <value> (Optional) Height of the geometrical form (where applicable).

--opacity <double> (Optional) Data points opacity. Default value 0.66.

--opacityTF <three double values> (Optional) Data smoothed points opacity representation. They fix the curve slope for opacity transfer function when smoothed representation is given. The three values must not be negative. Suggested ranges are [3-10] [1-5] [2-5]. Default values are 5.0 3.0 2.5.

--scaleglyphs (Optional) Enables the geometrical form to be scaled with a scalar field.

--scenario (Optional) In a smoothed representation it gives the colors for data point. Each scenario is represented by a string name. The active current scenario is called etna. This is the default value.

--radiusscalar <field> (Optional) Sets the scalar field for radius scaling.

--heightscalar <field> (Optional) Sets the scalar field for height scaling.

Examples:

Palette usage

```
$ VisIVOViewer --x X --y Y --z Z --color --colorscalar scalar0 --colortable temperature -
↪-logscale /home/user/inputFile.bin
```

Normal glyphs

```
$ VisIVOViewer --x X --y Y --z Z --glyphs cone --radius 1 --height 2 /home/user/
↪inputFile.bin
```

Scaled glyphs

```
$ VisIVOViewer --x X --y Y --z Z --glyphs cone --scaleglyphs --radiusscalar scalar0 --
↪heigthscalar scalar1 /home/user/inputFile.bin
```

9.3.2 Volumes

VisIVOViewer creates a volume view of data points from the input data file that contains a volume. The input data file must be in VBT format and must have the number of mesh elements on each dimension. The input data file must fit the available RAM.

A volume can be visualized with the volume rendering technique, with an isosurface or with slices. A color table must be given. The default color table will be used if the colortable option is not given.

Specific volume options:

--volume (Optional) Enables volume visualization.

--vrendering (Optional) Enables volume rendering view. The volume rendering view is the default when –volume is given.

--vrenderingfield <field> Sets the scalar to be represented in the view.

--shadow (Optional) Enables shadow view in the rendering view.

Example

```
$ VisIVOViewer --volume --vrendering --vrenderingfield density -colortable temperature /
↪home/user/inputFile.bin
```

Specific isosurface options:

--isosurface (Optional) Enables isosurface view.

--isosurfacefield <field> (Optional) Sets the scalar to be represented in the view.

--isosurfacevalue <field> (Optional) Fixes the isocontur value: from 0 to 255.

--wireframe (Optional) Visualizes the isosurface with wireframe.

--isosmooth (Optional) Smoothes the isosurface visualization. It may assume the following values: none (default), medium, high.

Example

```
$ VisIVOViewer --volume --isosurface --isosurfacefield density --isosurfacevalue 200 /
↪home/user/inputFile.bin
```

Specific slider options:

--slice (Optional) Enables slice view.

--slicefield <field> (Optional) Sets the scalar to be represented in the slice view.

--sliceplane <plane> (Optional) Sets the plane to be represented in the view. It must be one of the following: x, y, z. The camera is always positioned in front of the plane.

--sliceposition <position> (Optional) Sets the plane coordinate position to be represented in the view. It must be an integer value from 0 to the maximum number of cells in the selected direction. Ignored if cycle option is given.

VisIVOViewer can also visualize oblique planes. In this case sliceplane and sliceposition options must not be given. The camera is positioned using azimuth and elevation options.

--sliceplanepoint (Optional) Sets the three coordinates of a point in the plane. It is ignored in case of cycle file.

--sliceplanenormal (Optional) Sets the three coordinates of a point belonging to the normal axes to the slice. The sliceplanepoint and the sliceplanenormal fix the points and anthe axis in the space. The slice is normal to this axis and the point in sliceplanepoint is a point of this plane. It is ignored in case of cycle file.

Important Remarks. One of the following options must be specified: sliceplane, sliceplanenormal and/or sliceplanepoint. If sliceplane is selected orthogonal slices will be produced. If sliceplane is not given but sliceplanenormal and/or sliceplanepoint are given, generic slices will be produced. In case of cycle the specific values of sliceposition, sliceplanenormal and sliceplanepoint are ignored and the cycle file values will be used, even if the options must be given to select the type of slice visualization.

Note: The stereoscopic visualization is ignored in case of slice.

Note: Cycle can be given for Orthogonal Normal planes (x, y or z). In this case the cycle file must contain a sequence of integers (one for each row) inside the volume range (e.g 0-64).

Note: Cycle can be given for point-planenormal slice. In this case the cycle file must contain a sequence of six values: three point coordinates and three plane normal coordinates. Lines with less than 6 values are ignored. In this case the showbox option is recommended.

Example:

```
$ VisIVOViewer --volume --slice --slicefield density --sliceplane x -sliceposition 3␣
↪-color --colortable default /home/user/inputFile.bin
```

9.3.3 Vectors

VisIVOViewer creates a view of vectors created from the input data file that contains data points. The input data file must fit the available RAM.

Options:

--vector (Optional) Enables vector visualization.

--x <field> (Optional) First component of the vector application point.

--y <field> (Optional) Second component of the vector application point.

--z <field> (Optional) Third component of the vector application point.

--vx <field> (Optional) First component of the vector.

--vy <field> (Optional) Second component of the vector.

--vz <field> (Optional) Third component of the vector.

--colorscalar <field> (Optional) Selects the field of the VBT to be used for the palette. The vectors are displayed with the color palette based on the value of the active scalar given in this option. If this option is not given, the palette is based on is the magnitude of the vector. This option set the active scalar.

--vectorline (Optional) Enables the vector representation with a line. Default is arrows.

--vectorscalefactor <field> (Optional) Scale factor for vector representation.

--vectorscale <field> (Optional) Assumes the following values. Value 0: the scale of the vector dimension is given by the active scalar (colorscalar option). Value 1: the scale of the vector dimension is given by the vector magnitude. Value -1 (default): the vectors are not scaled.

Example:

```
$ VisIVOViewer --x X --y Y --z Z --vx Vx --vy Vy --vz Vz --color --colorscalar scalar0 --
↪colortable temperature --vectroscalefactor 1.3 --vectorscale 0 /home/user/inputFile.bin
```

CHAPTER

TEN

VISIVO UTILS

VisIVO Utilities is a tool that creates intermediate data that will be used by the other components of VisIVO Server. These data could consist of a sequence of values or files extracted from VisIVO Binary Tables.

To get general help:

```
$ VisIVOUtils --help
```

To get a specific utility help:

```
$ VisIVOUtils --op utilitycode --help
```

To run the utility:

```
$ VisIVOUtils --op utilitycode <options>
```

10.1 Utilities

The following utilities are available:

10.1.1 Create path

This utility append or creates an ascii files containing 4, 7 or 8 values for each row for the camera position. The file can be used by VisIVOViewer –cycle to produce a sequence of png images to be mounted in a movie.

The camera position, focal point and roll, when not specified, will contain NULL that will allow VisIVO Viewer to maintain the last used setting.

Usage:

```
$ VisIVOUtils --op createpath --type value [--azimuth from to] [--elevation from to] [--
↪zoom from to] [--zoomend [stepframe]] [--campos from to] [--camfp from to] [--camroll␣
↪from to] [--framesec value] [--length value] [--out filename] [--help]
```

(continues on next page)

(continued from previous page)

Options:

--type	0 Create path for azimuth, elevation, zoom and roll. Default value. 1 Create path for azimuth, elevation, zoom, focal point and roll. 2 Create path for zoom, camera position, focal point and roll. 3 Create path for azimuth, elevation, zoom, camera position, focal point and roll.
--azimuth	Movement from to. Default values 0.0 and 0.0.
--elevation	Movement from to. Default values 0.0 and 0.0. Valid range [-90,90]. Values outside this interval are automatically set to the near extreme: Ex.: `--elevation -85 100` will be modified with `--elevation -85 90`.
--zoom	Zoom from to. Default values 1.0 and 1.0. A zooming factor <1 represents a zoom in a zooming factor >1 represent a zoom out. Negative value are ignored.
--zoomend	The zoom is given at the end. The value step-frame represent the step for zooming . Default step-frame is 0.2 If this option is given priority with zoom will be ignored. The final zooming is added to global the length.
--campos	Movement from to. Three vale for starting point and three value for ending point are expected.
--camfp	Movement from to. Three vale for starting point and three value for ending point are expected.
--camroll	Movement from to. Three vale for starting point and three value for ending point are expected.
--framesec	Number of frame values for each second. Default value is 10.
--length	Value in seconds. Default value is 10 sec.
--out	Output filename. Default filename cycle.par. The file is opened in append mode.

Example:

```
$ VisIVOUtils --op createpath --type 0 --azimuth 0.0 60. --elevation 0.0 10.0 --zoom 1.0␣
↪1.5 --zoomend --length 20 --out my_cycle.par
```

The utility produce 10 values (default value) for each second. The file `my_cycle.par` will contain: azimuth, elevation, zooming and roll:

```
0.0 0.0 1.0 NULL
......
60.0 10.0 1.0 NULL
60.0 10.0 1.2 NULL
60.0 10.0 1.4 NULL
60.0 10.0 1.6 NULL
```

10.1.2 Orthogonal Slices

This utility append or creates an ascii file containing the slice poisition in the volume table. The file can be used by VisIVOViewer –cycle to visualize slices and to produce a sequence of png images to be mounted in a movie.

Usage:

```
$ VisIVOUtils --op orthoslices --pos from to [--xplane] [--yplane] [--zplane] [--step␣
↪stepvalue] [--out filename] [--help] [--file inputFile.bin]
```

Options:

--pos	Sets the slice position from-to in the volume. Values outside the volume size are ignored.
--xplane	Sets the direction x to be considered. Default is x.
--yplane	Sets the direction y to be considered. It is ignored if –xplane is given.
--zplane	Sets the direction z to be considered. It is ignored if –xplane or –yplane is given.
--step	Step increment for slice position (integer). Default value 1.
--out	output filename. Default filename cycle.par. The file is opened in append mode.
--file	(optional) Input Volume table.

Example:

```
$ VisIVOUtils --op orthoslices --pos 0 64 --step 1 --out my_cycle.par --file inputFile.
↪bin
```

The utility produces 64 values as follows:

```
0
1
2
...
64
```

10.1.3 Generic Slices

This utility append or creates a file with six columns. The point position (plane point) has increased (decreased) of step_size for n steps. The plane point is moved along the normal axis. The product step*size determines the movement of the plain point. If step*size is equal to 1, at the end the plane point will be at the same point of the normal point.

The file can be used by VisIVOViewer `--cycle` to visualize generic slices and to produce a sequence of png images to be mounted in a movie.

Usage:

```
$ VisIVOUtils --op genericslices --point x y z --normal x y z --step n [--size step_
↪size] [--movedown] [--out filename] [--help]
```

Options:

--point	The three coordinates of a point in the plane.
--normal	The three coordinates fixing the normal axis to the plane.

--step	Number (int) of generated new point positions along the normal axis.
--size	Value of increased (decreased) point coordinates. Default value 1.
--movedown	The plane point is moving to the opposite side of the normal point.
--out	Output filename. Default filename cycle.par. The file is opened in append mode.

Example:

```
$ VisIVOUtils --op genericslices --point 1 1 1 --normal 10 10 10 --step 20 --size 0.05 --
↪out cyclefile
```

The utility produces 21 values (including the starting point) as follows:

```
1 1 1 10 10 10
1.45 1.45 1.45 10 10 10
1.9 1.9 1.9 10 10 10
2.35 2.35 2.35 10 10 10
2.8 2.8 2.8 10 10 10
3.25 3.25 3.25 10 10 10
3.7 3.7 3.7 10 10 10
4.15 4.15 4.15 10 10 10
4.6 4.6 4.6 10 10 10
5.05 5.05 5.05 10 10 10
5.5 5.5 5.5 10 10 10
5.95 5.95 5.95 10 10 10
6.4 6.4 6.4 10 10 10
6.85 6.85 6.85 10 10 10
7.3 7.3 7.3 10 10 10
7.75 7.75 7.75 10 10 10
8.2 8.2 8.2 10 10 10
8.65 8.65 8.65 10 10 10
9.1 9.1 9.1 10 10 10
9.55 9.55 9.55 10 10 10
10 10 10 10 10 10
```

10.1.4 Load History

This utility, starting from a history xml file creates a bash script for re-execution.

Usage:

```
$ VisIVOUtils --op loadhistory --file <hist.xml> [--help]
```

Options:

--out	Output filename. Default filename VisIVO.sh
--file	Input History file.

Example:

```
$ VisIVOUtils --op loadhistory --file hist.xml
```

The utility produces the script `VisIVO.sh` from the file `hist.xml`.

10.1.5 Text Column

Starting from an ascii file file, extract the value of a column as string.

Usage:

```
$ VisIVOUtils --op textcol --file <table.ascii> --colname <column_name> [--help]
```

Options:

--colname	The name of the requested column.
--out	Output filename. Default filename VisIVO.sh
--file	Input ASCII file.

Example:

```
$ VisIVOUtils --op textcol --file table.txt --colname X
```

The utility extracts the column X from file `table.txt`.

CHAPTER

ELEVEN

INTRODUCTION

The ViaLactea Visual Analytics (VLVA) tool combines different types of visualization to perform the analysis exploring the correlation between different data, for example 2D intensity images with 3D molecular spectral cubes. All underlying data are managed by the ViaLactea Knowledge Base (VLKB). The VLKB includes 2D and 3D (velocity cubes) surveys, numerical model outputs, point-like and diffuse object catalogues and allows for retrival of all the available datasets as well as cutouts on the positional and/or velocity axis and some merging capabilities on adjacent datasets.

For more details about the ViaLactea Visual Analytics tool and its aspects please refere to F. Vitello et al. 2018 PASP 130 084503.

Please also refer to the DOI 10.5281/zenodo.7274312 as a citation when using VLVA for publications.

CHAPTER

TWELVE

INTRODUCTION

The ViaLactea Web (VLW) solution is a simplified web version of the ViaLactea Visual Analytic (VLVA) tool. It is developed as a collaborative web solutions for multi-user support underpinned by efficient remote server CPU and GPU rendering, and support of mobile and desktop devices.

All underlying data is managed by a dedicated data service, namely the ViaLactea Knowledge Base, that provides object catalogs and Spectral Energy Distribution model outputs to carry out correlation analysis workflows for studying the star formation process in our Galaxy.

The overall performance experiences is defined by remote GPU and CPU visualisation server performances.