VSRS Software Manual

Version: VSRS 4.0 (SVN tag: VSRS_4)

Last update: October 28, 2013

Summary:

This document contains a detailed description of the usage and configuration of the VSRS (*V*iew *S*ynthesis *R*eference *S*oftware) for the 3D Video (3DV) and Free viewpoint TeleVision (FTV) project of the Free Viewpoint Television of the ISO/IEC Moving Pictures Experts Group (MPEG). The name of version 1 of this software was View Synthesis software.

It provides information how to build the software on Windows32 and Linux platforms. It contains a description of the usage and configuration for the binaries built from the software package, including examples for view synthesis scenarios.

Date	Contributors	Brief Description
Monday, 28 Octiober 2013	Krzysztof Wegner (Poznan Univeristy of Technology) kwegner@multimedia.edu.pl	Supports for 16bit depth maps allowing to use depth maps with 65k disparities.
Thursday, 28 April 2009	Lu Yu and Yin Zhao (Zhejiang Univeristy)	Improvement in 1D mode - Temporal Improvement Method - Enhanced warping process to clean noise
Friday, 3 April 2009	Dong Tian (Thomson) dong.tian@thomson.net Kazuyoshi Suzuki (Nagoya Univ.) Cheon Lee (GIST)	Updated with two new features (SVN tag: VSRS_3) - ViSBD was merged to VSRS (more precisely, 1D-parallel view synthesis mode of VSRS was replaced with view generation method in ViSBD) - Boundary Noise Removal was implemented.
Monday, 17 November 2008	Kazuyoshi Suzuki (Nagoya University) and Shinya Shimizu (NTT) (suzuki-kz@tanimoto.nuee.nagoya-u.ac.jp)	Initial version (SVN tag: VSRS_2)

Table of Contents

1 General Information

The VSRS (*V*iew *S*ynthesis *R*eference *S*oftware) is the reference software for the 3D Video and FTV project of the Free Viewpoint Television Team of the ISO/IEC Moving Pictures Experts Group (MPEG). Since the 3D Video and FTV project is still under development, the VSRS as is also under development and changes frequently.

The VSRS is written in C++ and is provided as source code. Section 1.1 describes how the VSRS can be obtained via an SVN server. Information about the structure of the MPEG SVN repository is presented in section 1.2. Section 1.3 describes how the VSRS can be built on Win32 and Linux platforms, and section 1.4 gives basic information about the binaries that are contained in the VSRS software package.

1.1 Accessing the latest VRSR

In order to keep track of the changes in software development and to always provide an up-to-date version of the VSRS, a SVN server for the VSRS has been set up at the MPEG sc29wg11 website. The SVN server can be accessed using Tortoise SVN or any other SVN client. The server is configured to allow read access only using the parameters specified in Table 1. Write access to the VSRS server is restricted to the VSRS coordinators group.

Table 1: SVN access parameters

trunk url address:	http://wg11.sc29.org/svn/repos/MPEG-4/test/trunk/3D/view_synthesis/VSRS
tag url address:	http://wg11.sc29.org/svn/repos/MPEG-4/test/tags/3D/view_synthesis/VSRS_xx
user name:	sc29wg11
password:	same password as MEPG account, may be updated every few months

Example 1 shows how the VSRS can be accessed by using a command line SVN client.

Example 1: Accessing the VSRS with a command line SVN client

```
svn co http://wg11.sc29.org/svn/repos/MPEG-4/test/trunk/3D/view_synthesis/VSRS VSRS
```

In Example 2, it is shown how a specific VSRS software version – specified by a tag (VSRS_2 in Example 2) – can be obtained using a command line SVN client. Note that *co* represents an abbreviation for the command *checkout*, which was used in Example 1.

Example 2: Accessing the VSRS software version with the tag VSRS_3 with a command line SVN client

```
svn co http://wg11.sc29.org/svn/repos/MPEG-4/test/tag/3D/view_synthesis/VSRS_3 VSRS_3
```

1.2 Structure of the MPEG SVN Repository

After accessing the VSRS as described in section 1.1, a folder *VSRS* is created. The directory structure of this folder is summarized in Table 2. The folder *VSRS* contains all files that are required for building and running the software.

The folder *VSRS/CommonLibStatic* is structured into sub-folders for all source (*.cpp) and include (*.h) files for the common libraries of the VSRS and DERS. The folder *VSRS/ViewSynLibStatic* is structured into sub-folders for all source and include files for the libraries of the VSRS. The folder

VSRS/ViewSyn is structured into sub-folders for all source and include files for the application of the VSRS.

A log file describing the main changes from one VSRS software version to the next is given by *vsrs_changes.txt*. Note that this log files starts with the View Synthesis version 0 (SVN tag VS0).

Table 2: Structure of the MPEG SVN repository for the VSRS software

Folder	***Content***
VSRS	*source code and project files for the VSRS* All files that are required for building and using the VSRS software are contained in this folder.
VSRS/windows	*workspaces* Workspaces are provided for Microsoft Visual Studio 6 and Microsoft Visual Studio .NET.
VSRS/linux	*makefiles* Makefiles are provided for Linux
VSRS/ViewSyn	*source code and project files for the VSRS applicaton* All files that are required for building the VSRS application are contained in this folder.
VSRS/ViewSynLibStatic	*source code and project files for the VSRS library* All files that are required for building the VSRS libraries are contained in this folder.
VSRS/CommonLibStatic	*source code and project files for the VSRS and DERS common library* All files that are required for building the common libraries of the VSRS and DERS are contained in this folder.
VSRS/camera_parameter_files	*camera parameter files*
VSRS/configuration_files	*configuration files*
VSRS/bin	*location of binaries after building the software* More information about the binaries are given in section 1.4
VSRS/lib	*location of libraries after building the software* More information regarding the libraries are given in section 1.4
VSRS/doc	*changes log file and software manual* *vsrs_changes.txt* file described the (main) changes from one SVN version to the next. It starts with View Synthesis version 0 (SVN tag: VS0).

1.3 Building the VSRS

It shall be possible to build the VSRS on a Windows32 platform with Microsoft Visual Studio .NET and Visual Studio 6 and on a Linux platform with gcc. For information on how to build the software on a Windows32 platform with Microsoft Visual Studio .NET and Visual Studio 6 refer to section 1.3.1, and for information on how to build the software on a Linux platform with gcc refer to section 1.3.2.

Since the VSRS is written in C++, it should also be possible to build the software on other platforms, which provide a C++ compiler. However, it is only guaranteed that the software can be built by using Microsoft Visual Studio .NET and Visul Studio 6 or the gcc compiler.

1.3.1 Windows32 platform with Microsoft Visual Studio

The folder *VSRS/windows* contains a Microsoft Visual Studio .NET 2003 workspace *VSRSVC7.sln*. In order to build the software, open this workspace with Microsoft Visual Studio .NET, and build the project files by selecting *Build → Batch Build*, which opens a new dialog window. Then press the buttion *Select All* and *Rebuild*.

After building the software the folders *bin* shall contain the executable binaries *VSRS/bin/ViewSyn.exe* and *ViewSynd.exe*. The folders *CommonLibStatic/Debug* and *CommonLibStatic/Release* contain libraries *VSRS/CommonLibStatic/Debug/CommonLibStatic.lib* and *VSRS/CommonLibStatic/Release/CommonLibStatic.lib*, respectively. The folders *ViewSynLibStatic/Debug* and *ViewSynLibStatic/Release* contain libraries *VSRS/ViewSynLibStatic/Debug/ViewSynLibStatic.lib* and *VSRS/ViewSynLibStatic/Release/ViewSynLibStatic.lib*, respectively. Note that there exist two different versions for the binary, one with and one without a "d" before the dot. The version with a "d" before the dot represent binary that has been built in debug mode, while the version without a "d" before the dot represent binary that has been built in release mode. The folder *VSRS/windows* also contains a Microsoft Visual Studio 6 workspace *VSRSVC6.dsw*. However, the compilation under Microsoft Visual Studio 6 is only occasionally checked, and thus it is not guaranteed that each software version can be build using Microsoft Visual Studio 6. In order to build the software with Microsoft Visual Studio 6, open the workspace with Microsoft Visual Studio 6.0, and build the project file by selecting *Build→ Batch Build*, which opens a new dialog window. Then press the buttion *Select All* and *Rebuild*.

1.3.2 Linux platform with gcc compiler

Makefiles for the Linux with gcc compiler is provided in the folder *VSRS/linux* and corresponding sub-folders. For example, if the currect folder is the folder *VSRS/lunux* of the VSRS repository (see section 1.2), the command specifies in Example 3 should be executed to build all project files.

Example 3: Building the VSRS under Linux with a gcc compiler.

```
make
```

By replacing *make* with *make release* or *make debug* in the Example 3, it can be specified that only the release or debug versions of the libraries and executables should be built.

After building the software the folders *bin* and *lib* shall contain the binaries and libraries *VSRS/bin/ViewSyn, ViewSynd* and libraries *VSRS/lib/libViewSynLibStatic.a, libViewSynLibStaticd.a, VSRS/lib/libCommonLibStatic.a, libCommonLibStaticd.a,* respectively. Note that there exist two different versions for each binary or library, one with and one without a "d" before the dot. The versions with a "d" before the dot represent binaries or libraries that have been built in debug mode, while the versions without a "d" before the dot represent binaries or libraries that have been built in release mode. When the command *make release* or *make debug* was used, only the debug or release version are present, respectively.

1.4 Information on binary and library

Table 3 and Table 4 give information on the library and executable that are contained in the VSRS software package. Note that – as described in sections 1.3.1 and 1.3.2 – the naming of the actual library and executable files is dependent on the platform and on whether the library and executable have been built in release or debug mode.

Detailed information on command line options and configuration parameters for the executables are given in section 2. And in section 2 and examples for using the VSRS software are given in the form of a tutorial. And in section 0 an example perl script is given to run batch simulation tasks with VSRS.

Table 3: Library provided by the VSRS software

library	***description***
ViewSynLibStatic	*View synthesis algorithm lib* This library provides classes that are used to do view synthesis.
CommonLibStatic	*common lib* This library provides classes that are used by both VSRS and DERS softwares.

Table 4: Executable provided by the VSRS software

executable	***description***
ViewSyn	*View synthesis application* The application to run view synthesis tasks. More information on the usage of the program is provided in section 2.

1.5 OpenCV

To compile VSRS, the OpenCV library must be installed on your computer. The version of the OpenCV has to be 1.0.0 or more. If you are not familiar with OpenCV, refer

http://www.intel.com/technology/computing/opencv/index.htm

If opencv-devel is installed using yum of Fedora Core 6.0 or more, opencv ver.1.0.0 or more is installed automatically. If opencv ver.1.0.0 or more is not installed, please download the source files from

http://sourceforge.net/projects/opencvlibrary/

and compile and install them.

The header files of opencv have to be set by user. If pkg-config is installed and opencv-devel is configured in user's PC, makefiles of VSRS do not need any changes.

2 Usage and configuration of the VSRS

This section provides information on usage and configuration of the binary contained in the VSRS package. Additionally, examples for using the software are provided in section 0.

2.1 VSRS

The VSRS can be used for generating intermediate views. The basic View Synthesis call is illustrated in Example 4 and Example 5. At this *cfg* represents the filename of the configuration file. The configuration file shall be specified for each VSRS call.

Example 4: Using the VSRS with Visual Studio .NET

```
VSRSVC7.exe <cfg>
```

Example 5: Using the VSRS with Visual Studio 6

```
VSRSVC6.exe <cfg>
```

Generally, the configuration files present a collection of configuration parameters. Each configuration parameter is specified in one line of the configuration files. Comments are started by the character '#'. The order of configuration parameters inside a configuration file can be arbitrarily selected. Each configuration parameter has a default value, and when the configuration parameter is not present in the configuration file, the default value is taken instead.

Thus, it is generally not required to specify all configuration parameters in a configuration file. Finally, it should be noted that the configuration parameter values specified in the configration file can be replaced by the values specified via command line options.

Two example configuration files are provided in the following sub-sections.

2.2 Configuration file

All available configuration file parameters for the view synthesis together with a brief description are summarized in Example 6. Additional information about the configuration parameters is given below.

Example 6: View Synthesis configuration file

```
#================ Input Parameters ================
DepthType                       0                       # 0...Depth from camera,
1...Depth from the origin of 3D space
SourceWidth                     1280                     # Input frame width
SourceHeight                    960                     # Input frame height
StartFrame                      0                       # Starting frame
TotalNumberOfFrames             300                     # Total number of input
frames
LeftNearestDepthValue           3907.725727             # Nearest depth value of
left image from camera or the origin of 3D space
LeftFarthestDepthValue          8221.650623            # Farthest depth value of
left image from camera or the origin of 3D space
RightNearestDepthValue          3907.725727             # Nearest depth value of
right image from camera or the origin of 3D space
RightFarthestDepthValue         8221.650623            # Farthest depth value of
right image from camera or the origin of 3D space
CameraParameterFile             ..\\camera_parameter_files\\cam_param_dog.txt
# Name of text file which includes real and virtual camera parameters
LeftCameraName                  param_dog38             # Name of real left camera
VirtualCameraName               param_dog39             # Name of virtual camera
RightCameraName                 param_dog41             # Name of real right camera
LeftViewImageName               C:\\YUV\\Dog\\dog038.yuv          # Name of left
input video
RightViewImageName              C:\\YUV\\Dog\\dog041.yuv          # Name of right
input video
LeftDepthMapName                depth_dog038.yuv        # Name of left depth map
video
RightDepthMapName               depth_dog041.yuv        # Name of right depth map
video
OutputVirtualViewImageName      dog_virtual039.yuv      # Name of output virtual
view video

SynthesisMode                   0                       # 0...General, 1...1D
parallel

ColorSpace                      0                       # 0...YUV, 1...RGB
Precision                       4                       # 1...Integer-pel,
2...Half-pel, 4...Quater-pel
Filter                          1                       # 0...(Bi)-linear,
1...(Bi)-Cubic, 2...MPEG-4 AVC

BoundaryNoiseRemoval   1   # Boundary Noise Removal: Updated By GIST

SynthesisMode        0         # 0...General, 1...1D parallel

#---- General mode ------
ViewBlending          0        # 0...Blend left and right images, 1...Not Blend

#---- 1D mode      ------
```

```
#---- In this example, all parameters below are commented and default values will
be taken ----
SplattingOption       2         # 0: disable; 1: Enable for all pixels; 2: Enable
only for boundary pixels. Default: 2
BoundaryGrowth        40                # A parameter to enlarge the boundary area
with SplattingOption = 2. Default: 40
MergingOption         2                 # 0: Z-buffer only; 1: Averaging only; 2:
Adaptive merging using Z-buffer and averaging. Default: 2
DepthThreshold        75                # A threshold is only used with
MergingOption = 2. Range: 0 ~ 255. Default: 75
HoleCountThreshold    30                # A threshold is only used with
MergingOption = 2. Range: 0 ~ 49. Default: 30
TemporalImprovementOption  1            # Temporal improvement method, 0: Disable;
1: Enable. Default 1.
WarpEnhancementOption      1            # Clean noise in 1D warping. 0: Disable; 1:
Enable;  Default 1
CleanNoiseOption           1            # a method to clean boundary noise during
warping process. 1: On, 0 off. Default 1.
```

DepthType

Bool (0 or 1), default: 0

Specifies the depth type. The input value 0 means the view synthesis mode by using depth from a camera. The input value 1 means the view synthesis mode by using depth from the origin of 3D space.

SourceWidth

Unsigned Int, default: 1280

Specifies the width of the input images. *SourceWidth* shall be a positive.

SourceHeight

Unsigned Int, default: 960

Specifies the height of the input images. *SourceHeight* shall be a positive.

StartFrame

Unsigned Int, default: 0

Specifies the start frame number. *StartFrame* shall be a nonnegative.

TotalNumverOfFrames

Unsigned Int, default: 300

Specifies the number of frames of the input sequence. TotalNumverOfFrames shall be a positive integer.

LeftNearestDepthValue

Double, default: 0.0

Specifies the nearest depth value of left image from camera or the origin of 3D space.

LeftFarthestDepthValue

Double, default: 0.0

Specifies the farthest depth value of left image from camera or the origin of 3D space.

RightNearestDepthValue

Double, default: 0.0

Specifies the nearest depth value of right image from camera or the origin of 3D space.

RightFarthestDepthValue

Double, default: 0.0

Specifies the farthest depth value of right image from camera or the origin of 3D space.

CameraParameterFile

String, default: cam_param_dog.txt

Specifies the filename (with the .txt) of the file which includes intrinsic and extrinsic parameters of all cameras at real and virtual views.

LeftCameraName

Sting, default: param_dog38

Specifies the name of real camera at the left hand side of a specified virtual viewpoint camera.

VirtualCameraName

Sting, default: param_dog39

Specifies the name of virtual viewpoint camera to be generated view images.

RightCameraName

Sting, default: param_dog41

Specifies the name of real camera at the right hand side of a speficied virtual viewpoint camera.

LeftViewImageName

Sring, default: dog038.yuv

Specifies the filename (with the .yuv) of the original video sequence corresponds to left camera to be used for view synthesis.

RightViewImageName

String, default: dog041.yuv

Specifies the filename (with the .yuv) of the original video sequence corresponds to right camera to be used for view synthesis.

LeftDepthMapName

Sring, default: depth_dog038.yuv

Specifies the filename (with the .yuv) of the depth map video sequence corresponds to left camera to be used for view synthesis.

RightDepthMapName

String, default: depth_dog041.yuv

Specifies the filename (with the .yuv) of the depth map video sequence corresponds to right camera to be used for view synthesis.

OutputVirtualViewImageName

String, default: dog_virtual039.yuv

Specifies the filename (with the .yuv) of the output depth map video sequence.

SynthesisMode

Unsigned int (0 or 1), default: 0

Specifies the mode of view synthesis. 0 means the method has no restriction and 1 means the method is only works on 1D parallel multi-view sequences. 1D mode was replaced with the method in ViSBD.

ColorSpace

Unsigned int (0 or 1), default: 0

Specifies the color space where synthesis is operated. 0 means YUV and 1 means RGB.

Precision

Unsigned int (1 or 2 or 4), default: 2

Specifies the level of precision to find correspondances. 1 means interger-pixel precision, 2 means half-pixel precision, and 4 means quarter-pixel precision.

Filter

Unsigned int (0-2), default: 1

Specifies the upsampling filter to generate image signals on sub-pixel positions. 0 means (bi-) linear filter, 1 means (bi-) cubic filter, and 2 means filter which used in MPEG-4 AVC.

BoundaryNoiseRemoval

Unsigned int (0 or1), default: 1

Specifies the usage of boundary noise removal. The value 1 represents the use of the boundary noise removal and the value 0 represents no use.

ViewBlending

Unsigned int (0 or1), default: 0

Specifies blending left and right view images or primarily using either left or right view image that is near virtual viewpoint: 0 means blending left and right view images, 1 means primarily using either left or right view image. In mode 1, if virtual viewpoint is near to left viewpoint, left view image is used primarily and holes are filled with right image, and vice versa.

SplattingOption

Flag (0, 1 or 2), default: 2

Specifies if splatting is enabled. Value 0 means that splatting is disabled. Value 1 means that splatting is enabled for the whole picture. Value 2 means that splatting is enabled for boundary area only.

BoundaryGrowth

Unsigned int, default: 40

Invalid when SplattingOption is not equal to 2. When SplattingOption is equal to 2, once a boundary pixel is detected, all pixels fall within ±BoundaryGrowth of the detected boundary pixel are marked as boundary and then the splatting will be enabled in the corresponding 3D warping.

MergingOption

Flag (0, 1 or 2), default: 2

Specify the blending (merging) method. When a pixel gets mapped from both the left and the right reference view, a blending method need be applied to decide the final pixel value. With MergingOption equal to 0, the blending process relies solely on the z-buffer, which means that the pixel closer to camera is always selected. Value 1 means that an averaging is always performed. Value 2 means a more complicated method, described in MPEG document M15883.

DepthThreshold

Unsigned int, range [0, 255], default: 75

A valid parameter only when MergingOption is set to 2. It is used in the heuristic blending algorithm. A larger value means more pixels are averaged. Generally, a larger value is expected with a depth map of poor quality. See MPEG document M15883 for more details.

HoleCountThreshold

Unsigned int, range [0, 49], default: 30

A valid parameter only when MergingOption is set to 2. It is used in the heuristic blending algorithm. A larger value means less pixels are averaged. See MPEG document M15883 for more details.

TemporalImprovementOption

Bool (0 or 1), default: 1

Specifies the usage of Temporal Improvement Method (TIM, see more details in m16041). The value 0 represents that TIM is disabled in 1D mode, while 1 represents that TIM is applied.

WarpEnhancementOption

Bool (0 or 1), default: 1

Specifies the usage of enhancement method in 1D Warping, including constraint on pixel mapping and specific warping orders for pixels on different views. The value 0 represents that the method is disabled in 1D mode, while 1 represents that it is applied.

CleanNoiseOption

Bool (0 or 1), default: 1

Specifies the usage of Clean Noise method in 1D Warping. The boundary on texture is always not a sharp as the edge boundary, and boundary area seems to be separated after warping because of different depth value. This method detects major boundaries of the object, and disables the pixel warping around narrow area at left side of a rising boundary and at right side of a falling boundary. After warping, the area appears as hole and will be filled with merge algorithm or hole filling algorithm, and boundary noise is cleaned. The value 0 represents that the method is disabled in 1D mode, while 1 represents that it is applied.

Given the configuration files in Example 6, the encoder can be started using the call in Example 7 and Example 8 with the configuration file.

Example 7: View Synthesis call with Visual Studio .NET

```
>VSRSVC7.exe ViewSynthesis.cfg
```

Example 8: Example Depth Esimationt call with Visual Studio 6

```
>VSRSVC6.exe ViewSynthesis.cfg
```

2.3 Camera parameter file format

All available configuration file parameters for the depth estimation together with a brief description are summarized in Example 6. Additional information about the camera parameters is given below.

Specification of Camera Parameters

Camera parameters shall be specified as rotation matrix **R**, translation vector **t**, intrinsic matrix **A** for each camera i. Values shall be given in floating point precision as accurate as possible.

The extrinsic camera parameters **R** and **t** shall be specified according to a right-handed coordinate system. The upper left corner of an image shall be the origin for corresponding image/camera coordinates, i.e., the (0,0) coordinate, with all other corners of the image having non-negative coordinates.

The rotation matrix **R**(i) for i-th camera is represented as follows.

r_11[i]	**r_12**[i]	**r_13**[i]
r_21[i]	**r_22**[i]	**r_23**[i]
r_31[i]	**r_32**[i]	**r_33**[i]

The translation vector **t**(i) for i-th camera is represented as follows:

t_1[i]
t_2[i]
t_3[i]

The rotation matrix **R** and the translation vector **t** define the position and orientation of the corresponding camera with respect to the world coordinate system. The components of the rotation matrix **R** are function of the rotations about the three coordinate axes.

The intrinsic matrix **A**(i) for i-th camera is represented as follows:

focal_length_x[i]	**radial_distortion**[i]	**principal_point_x**[i]
0.0	**focal_length_y**[i]	**principal_point_y**[i]
0.0	0.0	1.0

focal_length_x[i] specifies the focal length of the i-th camera in the horizontal direction units of pixels.

focal_length_y[i] specifies the focal length of the i-th camera in the vertical direction in units of pixels.

principal_point_x[i] specifies the principal point of the i-th camera in the horizontal direction units of pixels.

principal_point_y[i] specifies the principal point of the i-th camera in the vertical direction in units of pixels.

radial_distortion[i] specifies the radial distortion coefficient of the i-th camera.

- Radial distortion coefficients are estimated up to first order
- Radial distortion centre is the same as the principal point.
- Tangential distortion coefficients have not been estimated, they are to be assumed zero.

Example 9: Example camera parameter file

```
'camera name of the real 0th view'
focal_length_x[0] radial_distortion[0] principal_point_x[0]
0.0               focal_length_y[0]    principal_point_y[0]
0.0               0.0                  1.0
0.0
0.0
r_11[0]           r_12[0]              r_13[0]          t_1[0]
r_21[0]           r_22[0]              r_23[0]          t_2[0]
r_31[0]           r_32[0]              r_33[0]          t_3[0]

'camera name of the virtual intermediate view of real 0th and 1st views'
focal_length_x[0_1] radial_distortion[0_1] principal_point_x[0_1]
0.0                 focal_length_y[0_1]    principal_point_y[0_1]
0.0                 0.0                    1.0
0.0
0.0
r_11[0_1]           r_12[0_1]              r_13[0_1]          t_1[0_1]
r_21[0_1]           r_22[0_1]              r_23[0_1]          t_2[0_1]
r_31[0_1]           r_32[0_1]              r_33[0_1]          t_3[0_1]

'camera name of the real 1st view'
focal_length_x[1] radial_distortion[1] principal_point_x[1]
0.0               focal_length_y[1]    principal_point_y[1]
0.0               0.0                  1.0
0.0
0.0
r_11[1]           r_12[1]              r_13[1]          t_1[1]
r_21[1]           r_22[1]              r_23[1]          t_2[1]
r_31[1]           r_32[1]              r_33[1]          t_3[1]

'camera name of the virtual intermediate view of real 1st and 2nd views'
focal_length_x[1_2] radial_distortion[1_2] principal_point_x[1_2]
0.0                 focal_length_y[1_2]    principal_point_y[1_2]
0.0                 0.0                    1.0
0.0
0.0
r_11[1_2]           r_12[1_2]              r_13[1_2]          t_1[1_2]
r_21[1_2]           r_22[1_2]              r_23[1_2]          t_2[1_2]
r_31[1_2]           r_32[1_2]              r_33[1_2]          t_3[1_2]

'camera name of the real 2nd view'
focal_length_x[2] radial_distortion[2] principal_point_x[2]
0.0               focal_length_y[2]    principal_point_y[2]
0.0               0.0                  1.0
0.0
0.0
r_11[2]           r_12[2]              r_13[2]          t_1[2]
r_21[2]           r_22[2]              r_23[2]          t_2[2]
r 31[2]         r_32[2]              r_33[2]          t_3[2]
```

Example 10: Example camera parameter file "cam_param_dog.txt"

```
param dog37
```

```
2979.0        0.0           -170.905600
0.0           2979.0        452.7121
0.0           0.0           1.0
0.0
0.0
1.0           0.0           0.0           -125.0
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0

param_dog38
2979.0        0.0           -152.788800
0.0           2979.0        452.7121
0.0           0.0           1.0
0.0
0.0
1.0           0.0           0.0           -75.0
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0

param_dog39
2979.0        0.0           -134.672000
0.0           2979.0        452.7121
0.0           0.0           1.0
0.0
0.0
1.0           0.0           0.0           -25.0
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0

param_dog40
2979.0        0.0           -116.555200
0.0           2979.0        452.7121
0.0           0.0           1.0
0.0
0.0
1.0           0.0           0.0           25.0
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0

param_dog41
2979.0        0.0           -98.438400
0.0           2979.0        452.7121
0.0           0.0           1.0
0.0
0.0
1.0           0.0           0.0           75.0
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0

param_dog42
2979.0        0.0           -80.321600
0.0           2979.0        452.7121
0.0           0.0           1.0
0.0
0.0
1.0           0.0           0.0           125.0
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0

param_dog43
2979.0        0.0           -62.204800
0.0           2979.0        452.7121
0.0           0.0           1.0
```

```
0.0
0.0
1.0          0.0          0.0          175.0
0.0          1.0          0.0          0.0
0.0          0.0          1.0          0.0

param_dog44
2979.0       0.0          -44.088000
0.0          2979.0       452.7121
0.0          0.0          1.0
0.0
0.0
1.0          0.0          0.0          225.0
0.0          1.0          0.0          0.0
0.0          0.0          1.0          0.0

param_dog45
2979.0       0.0          -25.971200
0.0          2979.0       452.7121
0.0          0.0          1.0
0.0
0.0
1.0          0.0          0.0          275.0
0.0          1.0          0.0          0.0
0.0          0.0          1.0          0.0

param_dog46
2979.0       0.0          -7.854400
0.0          2979.0       452.7121
0.0          0.0          1.0
0.0
0.0
1.0          0.0          0.0          325.0
0.0          1.0          0.0          0.0
0.0          0.0          1.0          0.0

param_dog47
2979.0       0.0          10.262400
0.0          2979.0       452.7121
0.0          0.0          1.0
0.0
0.0
1.0          0.0          0.0          375.0
0.0          1.0          0.0          0.0
0.0          0.0          1.0          0.0

param_dog48
2979.0       0.0          28.379200
0.0          2979.0       452.7121
0.0          0.0          1.0
0.0
0.0
1.0          0.0          0.0          425.0
0.0          1.0          0.0          0.0
0.0          0.0          1.0          0.0

param_dog49
2979.0       0.0          46.496000
0.0          2979.0       452.7121
0.0          0.0          1.0
0.0
0.0
1.0          0.0          0.0          475.0
```

```
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0

param_dog50
2979.0        0.0           64.612800
0.0           2979.0        452.7121
0.0           0.0           1.0
0.0
0.0
1.0           0.0           0.0           525.0
0.0           1.0           0.0           0.0
0.0           0.0           1.0           0.0
```

3 Information for Software Integration

At the MPEG meetings, the integration of proposals into the VSRS is decided. The software coordinators collect information on the integration complexity of all proposals and arrange a schedule for the integration period after the meeting. As input for this procedure, proponents should prepare to give an estimate of the time required for the integration of their proposal by the end of the meeting. The integration time includes the integration work itself and validation testing by the proponent.

The following software integration process is proposed:

- Please check out the latest software version from the MPEG SVN.
- Do your integration.
  - Please try hard to stick to the given time slot!
  - Please obey the guidelines in section 3.1.
- In case you might be running into trouble meeting the deadline please contact the software coordinators as soon as possible.
- Propose new tests that are designed to ensure that the tool you have implemented is not broken. Therefore, the tests do not need to demonstrate the performance of the tool but should rather validate its operability.
- Update the file "*vsrs_changes.txt"* as well as the software manual.

To verify the validity of the claimed gains of the proposal, proponents are encouraged to provide verification results with the integrated VSRS software for their adopted tool at the next meeting.

3.1 Software integration guidelines and rules

When integrating adopted proposals into the VSRS, please obey the following basic principles and recommendations:

- **The integrated software shall compile without warnings when using the provided VC6 and VS .NET workspaces as well as linux makefiles (i.e. using an up-to-date g++ compiler).**
- Do not use variable declarations inside the header of for-loops (the scope for for-loops is not correctly supported with all compilers).
- Follow the coding style of the View Synthesis software. Use 2 (two) spaces for indentation, no tabs.
- Re-use code and integrate functionality as possible. Try to avoid redundant code.
- Do not change the meaning of existing input parameters but define new ones if necessary (and applicable).
- Make sure that new parameters have meaningful default values. Tools should not be switched on by default (if not decided different by the MPEG).

- Do not re-structure the output of the compiled binaries (if not decided different by the MPEG).
- Please change the VSRS version number VERSION located in the file “version.h” to be inline with your integration tag (e.g. “2.0”).

3.2 Information to be provided to the software coordinators group

The following information shall be provided after integration of a proposal into the VSRS.

- The corresponding proposal numbers (for reference),
- A zipped software package containing only the modified files, but in the correct directory structure *without SVN directories*,
- Proposals for modified and/or additional test algorithms for existing tools.
- Proposals for new tests (one or more) that validate the functionality of the new tool.
- A brief list of changes (to maintain the changes file “*vsrs_changes.txt*” in the root directory of the VSRS – the changes should be directly written to the file “*vsrs_changes.txt*”),
- An updated version of the software manual (when parameters or tools have been added).