Surfdiver Users Guide
Version 3.0.0

February 27, 2014

www.illinoisrocstar.com

License

Surfdiver sources, executables, and this document are the property of Illinois Rocstar LLC. Licensing and support of the software package, including full source access for government, industrial, and academic partners, are arranged on an individual basis. Please contact Illinois Rocstar at

- **tech@illinoisrocstar.com**
- **sales@illinoisrocstar.com**

for support and licensing.

Contents

1 Introduction

Surfdiver is an executable program built on top of Rocface for constructing a *common refinement* or *common tessellation* of two nonmatching surface meshes, which can then be used to support transferring data between these meshes. Rocface was developed by Xiangmin Jiao at the Center for Simulation of Advanced Rockets (CSAR) at University of Illinois, as a framework component of the Rocstar suite. It uses the Roccom framework for data management and uses Rocin and Rocout of the framework for high-level I/O support (in HDF or CGNS). Most of the documents of these supporting libraries can be found at http://www.cse.uiuc.edu/~jiao/Roccom.

Figure 1 shows the data flow of Surfdiver. It reads two surface meshes in HDF files, detects the features (such as ridges and corners) in the meshes with user-provided parameters, and then computes the common refinement of input meshes. Note that the input meshes are assumed to model the same surface geometry, but discretization errors may be present so that the two discrete surfaces need not coincide. The nodes and elements of the comment refinement, referred to as *sub-nodes* and *sub-elements* (*sub-faces*), compose a mesh that is "nested" in the elements of both input meshes. We determine two separate copies of this common refinement by assigning different coordinates and numbering systems based on the two input meshes. These resulting two copies of the common refinement are then written into two groups of HDF files, one corresponding to each input mesh, in a user-specified directory. The input and output files of Surfdiver can be visualized using Rocketeer, or be converted into Tecplot or VTK format using Roccom's utility tools hdf2plt and hdf2vtk to be visualized using other tools.

The purpose of this documentation is to describe the file formats of Surfdiver, the procedure for building and running Surfdiver, and advanced procedures for tuning feature-detection parameters. This document uses some basic terminology of Roccom, including *windows*, *panes*, and *attributes*. A *window* is a distributed object composed of a mesh and its associated field attributes. A window may have zero, one or more *panes*, which may correspond to partitions of a mesh (for parallel computations) or correspond to patches with different boundary conditions. The attributes are referenced by character strings, including some reserved names (such as nodal coordinates "nc", connectivity tables ":t3", ":q4", etc.) and user-defined attribute names. Please refer to *Roccom's Users Guide* for more detail.

2 Input File Format

Surfdiver takes two separate groups of HDF files as input. Each group of input files stores a *surface window*, composed of a finite-element style *surface* mesh (including the nodal coordinates (double-precision) and element connectivity (integers)), which can be either a triangular, quadrilateral, or mixed unstructured mesh, or a multi-block structured mesh. For a partitioned mesh, the file may provide the connectivity among shared nodes along boundaries between different partitions (referred to as "pconn"). If this pconn is not provided, then Rocface will determine it automatically by comparing the differences of nodal coordinates

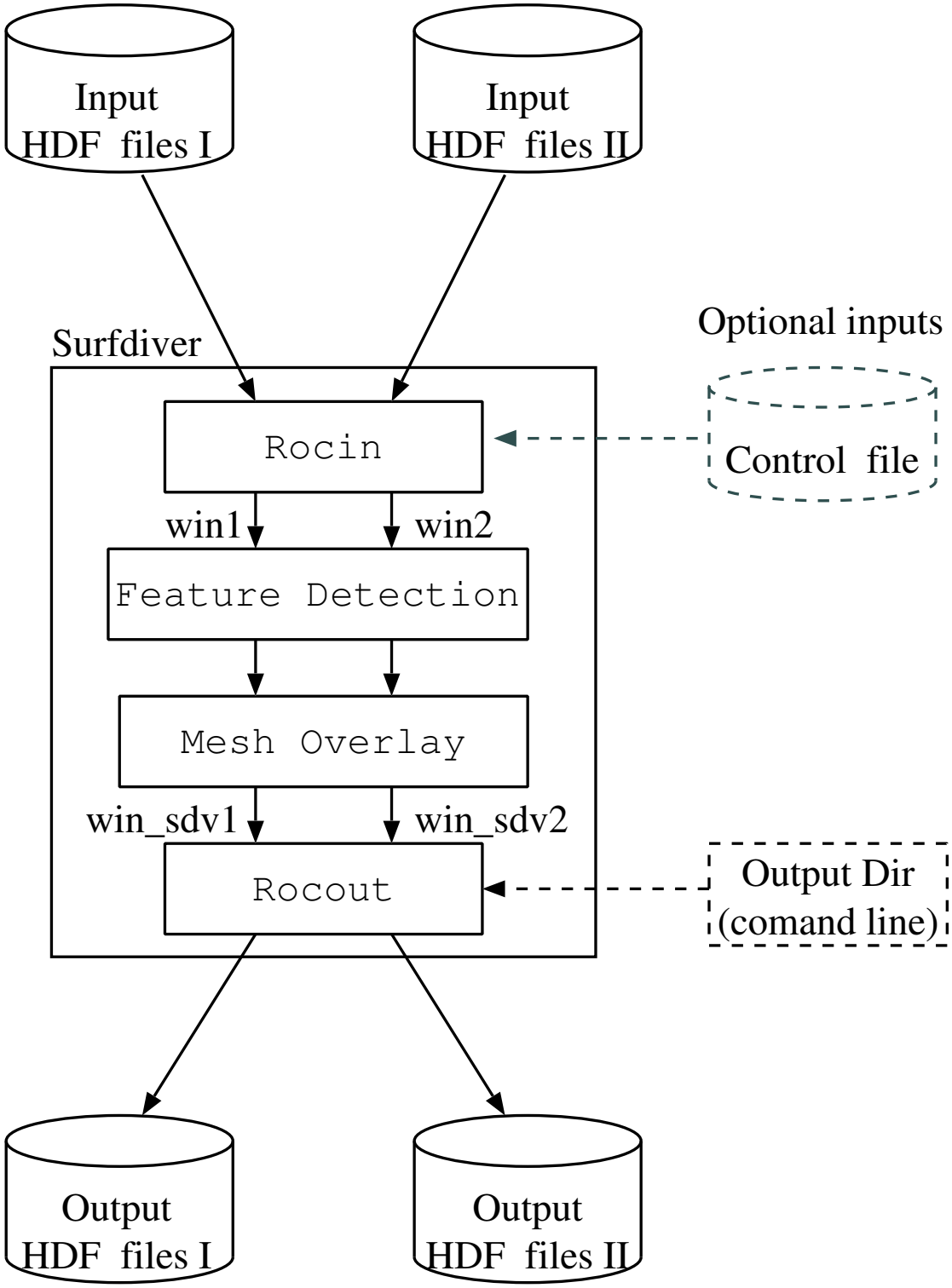

Figure 1: Data flow of Surfdiver.

against an automatically determined tolerance. These HDF files can be generated easily by first creating a window using Roccom's API and then calling Rocout.

Special note for Rocstar users: When using Surfdiver within Rocstar, each pane may have an integer-type attribute associated with it, named "bcflag". If this attribute is present, then Surfdiver will obtain only the panes with bcflag equal to 0 or 1 and ignore all the other panes. This convention was adopted to support rocket simulations with Rocstar, for which the panes with bcflag equal to 0 are interacting but not burning, those with value 1 are interacting and burning, and those with value 2 are non-interacting.

3 Output File Format

Surfdiver creates two groups of files to store the common refinement of surface meshes, one for each input window. The number of files in each input window is equal to the number of panes of the window. In the special case of sequential meshes, there will be one output file for each window. In the following, we will describe the contents in these files.

3.1 Fields for Sequential Meshes

If the input file for one window has prefix "*win*" (See Section 4 for details how the prefix is determined), as illustrated in Figure 2, the output file contains the following key information:

1. A window "*win*", including its coordinates, and element connectivity. This window essentially duplicates the mesh of the input window so that the output file is self contained.

2. A window "*win*_sdv", including its coordinates, element connectivity (composed of triangles). The sub-nodes are numbered arbitrarily. The sub-faces are sorted based on the IDs of their parent faces, so the sub-faces contained within a face are numbered consecutively. In addition, the following attributes are defined:

   (a) mapping from each sub-node to its parent face, or more precisely, one of its host faces that contain the sub-node ("sn_parent_fcID", one integer per sub-node)

   (b) mapping from each sub-node to the natural coordinates within its parent face ("sn_parent_ncs", two *single-precision* numbers per sub-node. Note that *all these local coordinates are between* 0 *and* 1.);

   (c) mapping from sub-nodes to their counterparts in the other window ("sn_cntpt_ndID", one integer per sub-node);

   (d) mapping from each node to its corresponding sub-node ID ("sn_subID", one integer per node);

   (e) mapping from each sub-face to its parent face ("sf_parent", one integer per sub-face);

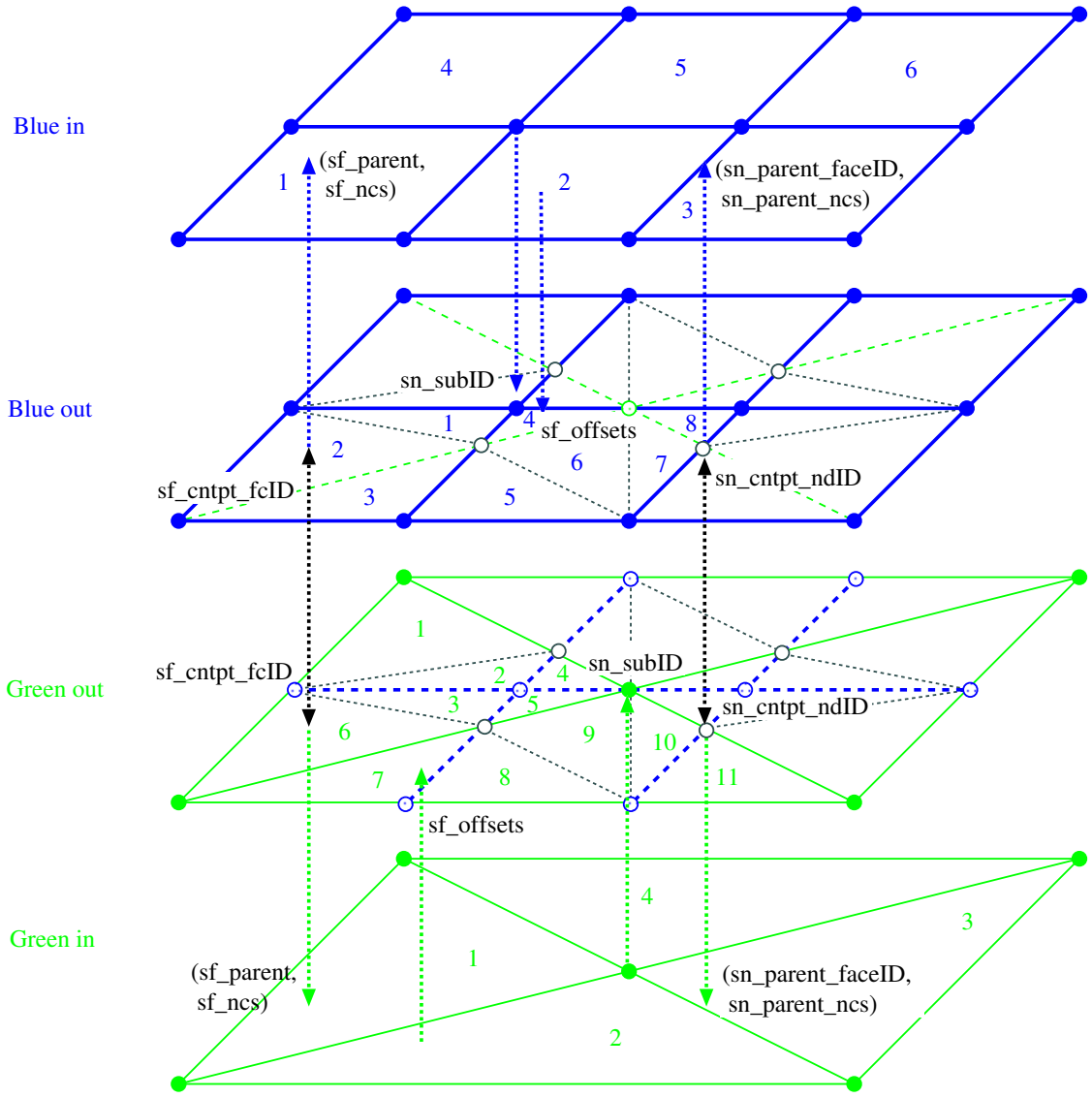

Figure 2: Schematic of common refinement of surface meshes.

(f) mapping from each sub-face to the local parametric coordinates of its sub-nodes in its parent face ("sf_ncs", six *single-precision* numbers per sub-face);

(g) mapping from each input face to the *offset* of its *first* sub-face (i.e., the sub-face ID minus 1, "sf_offset", one integer per sub-face);

(h) mapping from each sub-face to their counterparts in the other window ("sf_cntpt_ndID", one integer per sub-face);

(i) Additional fields (typically not needed by applications): the local coordinates ("sn_permu_ncs", two single-precision numbers per sub-node) of each sub-node in a permutation of its parent face (indicated by the edge "sn_permu_edID", one integer per sub-node), so that the sub-nodes at vertices have natural coordinates (0,0) and those at edges have natural coordinates $(\alpha, 0)$ for $0 < \alpha < 1$.

All the attributes with prefix "sn_" correspond to data associated with sub-nodes, and those with prefix "sf_" correspond to data associated with sub-faces. Except for "sn_subID" and "sf_offsets", which are *panel* attributes of lengths equal to the number of nodes and the number of faces of the input window, respectively, all other "sn_" attributes are *nodal* attributes and "sf_" attributes are *elemental* (*facial*) attributes of window "win_sdv". In addition, note that all the node and element IDs start from 1, except for sf_offset stores array index starting from 0.

The easiest way to read in these attributes is to use Rocin. A sample C++ code for reading in these attributes can be found at Rocface/test/readsdv.C.

3.2 Additional Fields for Partitioned meshes

If an input mesh has multiple panes (or partitions), then the panes of window "win_sdv" follow exactly the same partitioning as that of window "win". In addition, the following contents in the output HDF files become important as well. (Note that these additional fields are written out for sequential meshes as well, but they can be ignored for sequential meshes and hence be ignored.)

1. Pane connectivity "pconn" in window "win".

2. Window "win_sdv" contains the following additional fields

   (a) mapping from sub-faces to the pane IDs of their counterparts in the other window ("sf_cntpt_pnID");

   (b) mapping from sub-nodes to the pane IDs of their counterparts in the other window ("sn_cntpt_pnID");

   (c) auxiliary communication information useful for parallel transfer ("b2v" and "v2b").

A sub-face may have different element IDs in the output windows, and their owner panes may have different pane-IDs. The array "sf_cntpt_pnID" stores the pane ID of the sub-face

in the other window and "sf_cntpt_ndID" contains the sub-element-ID in the owner pane of the other window. Similarly, "sn_cntpt_pnID" stores the pane ID of the sub-node in the other window and "sn_cntpt_ndID" contains the sub-node-ID in the owner pane of the other window.

4 Building and Running Surfdiver

You can obtain Surfdiver as part of Rocstar suite from CSAR's CVS repository. A CVS account is needed. Talk to Mark Brandyberry to get account. Check out Rocstar's code using from directory "genx/Codes".

Surfdiver uses many modules of the Roccom framework and Rocface. Surfdiver is built automatically as part of the Rocstar suite. You may also build it separately by first building Roccom and then build Rocface by invoking the make command. Note that you must use the GNU make. You may run "gmake help" (or "make help" on Linux or Mac OS X) to see different compilation options. After successfully built, Surfdiver will reside at genx/bin/surfdiver.

To invoke Surfdiver, use the following command line:

surfdiver <Surf-mesh1> <Surf-mesh2> [output_dir] [Control File]

It is important that the prefixes of the input files are different. The prefix is defined as the alphabetical part of a base file name, e.g., the prefix of "dir/ifluid_1.hdf" or "dir/ifluid1.hdf" is "ifluid". Each surface-mesh is an HDF file name (e.g., ifluid.hdf), a HDF filename pattern (e.g., "ifluid_*.hdf", note that the quotation marks are required in this case to protect '*'), or a Rocin control file listing all the files and panes. Note that these prefixes (such as "ifluid") and their extensions with "_sdv" (such as "ifluid_sdv") are used as the "material" names in the HDF files are important when reading the HDF files. See the example code Rocface/test/readsdv.C to see how the prefix is used.

The third argument specifies the directory for the output files, and when not present, the default is the current directory. If the input file for one window has a prefix "win", then the output file names have the pattern "win_<pane_ID>_sdv.hdf". The fourth argument may specify a control file. The control file is a text file, and each line has the format of

option=value

Currently, the supported options include:

snap_tolerance The relative tolerance for merging subvertices (such as 0.01) along an edge if the distance between them is smaller than the tolerance times the edge length.

print_features Create HDF files to include the detected features to help debugging.

verbosity the level of verbosity. Value is an integer.

5 Advanced Features

Rocface automatically detects the corners and ridges of a surface mesh. For most models, the default parameters set by Rocface would work. For some complex models, one can control the parameters of feature detection by providing a control file <window name>.fc. This file should have five lines:

1. The first line contains four parameters for face angle: cosine of the upper bound, cosine of the lower bound, the signal-to-noise ratio, and cosine of an open-end of a 1-feature.
2. The second line contains three parameters for angle defect: upper bound, lower bound, and signal-to-noise ratio.
3. The third line contains three parameters for the turning angle: cosine of the upper bound, cosine of the lower bound, and the signal-to-noise ratio.
4. The fourth line contains four parameters controlling the filtration rules: the minimum edge length for open-ended 1-features, whether to apply the long-falseness filtration rule, whether to apply the strong-end filtration rule, and whether to snap 1-features of input meshes on top of each other.
5. The fifth line controls the verbosity level. The default value is one. Setting verbosity level to greater than one will instruct Rocface to write out HDF files "*_fea.hdf", which contain the feature information which is very helpful for tuning feature detection.

The following is a sample control file containing the default values.

```
0.76604444 0.98480775  3 0.98480775 # Feature angles
1.3962634  0.314159265 3            # Angle defects
0.17364818 0.96592583  3            # Turning angles
6 1 0 0                            # Filtration rules
2                                  # Verbosity level
```

If the control file is missing, then the default values (as shown above) will be used. These default values should work for most cases. If it ever becomes necessary to fine-tune the feature-detection parameters, adjusting only the parameters of feature angles (i.e., the first line of the .fc files) should suffice most of time. The following procedure is useful in determining the proper values of feature angles:

1. Find the ifluid_fea*.hdf and isolid_fea*.hdf files in the output directory. These files are generated by surfdiver if the ".fc" files are present and the verbosity level in these files are greater than 1.
2. Convert these HDF files into Tecplot files using the utility hdf2plt. Typically, the commands look like (note that the quotation marks are important):

Table 1: Troubleshooting procedure.

Problem	Potential causes	Suggested solutions
Mismatch bounding-boxes.	Different units.	Check dimensions of input meshes.
	Incorrect boundary conditions.	Check application configuration files.
Features do not match.	Incorrect boundary conditions.	Check application configuration files.
	Different geometric models.	Check geometry of input meshes.
	Weak geometric features.	Tune feature-detection parameters.
After feature matching.	Geometry mismatch.	Switch two windows in command line.
	Geometric features not detected.	Visualize features and tune parameters.
Other errors.	Bugs or unknown issues?	Contact xjiao@acm.org.

(a) hdf2plt "ifluid_fea*.hdf" ifluid.plt

(b) hdf2plt "isolid_fea*.hdf" isolid.plt

3. Load the ifluid.plt and isolid.plt into two separate Tecplot sessions. Look at the contour of "frank" (stands for feature rank) to eyeball the discrepancies of the detected features in the input meshes.

4. Use the "probe" tool of Tecplot to look at the values of "face_angle", and adjust the feature-angle thresholds based on the values "face_angle" of false features. Typically, if some edges are marked as features in correctly, then one should increase the feature-angle thresholds; if some feature edges are missing, then one should decrease the feature-angle thresholds.

6 Troubleshooting and Testing

Table 1 lists the most commonly seen problems and their potential causes and suggested solutions.

6.1 Visualization

The HDF files read and written by Surfdiver can be visualized using Rocketeer. In the input files, the most useful values to visualize is "bcflags". In the output files, it may be useful to visualize the meshes to see their sizes. For feature detection, it may be helpful to visualize the feature ranks ("franks") and face angle ("face_angle").