Facet Analyser: ParaView plugin for automated facet detection and measurement of interplanar angles of tomographic objects

Release 1.00

Roman Grothausmann[1] and Richard Beare[2]

February 5, 2015

[1]grothausmann.roman@mh-hannover.de
Institute of Functional and Applied Anatomy, Hannover Medical School and
REBIRTH Cluster of Excellence, Hannover, Germany
[2]richard.beare@ieee.org
Department of Medicine, Monash University, Melbourne, Australia
Developmental Imaging, Murdoch Childrens Research Institute, Melbourne, Australia

Abstract

The presented ParaView plugin allows easy access to the algorithm described in Ref. 1. It enables analysis of faceted objects that exhibit distortions in their digital representation, e.g. due to tomographic reconstruction artifacts. The contributed functionality can also be used outside ParaView in e.g. command-line programs. The code, data, a test and an example program are included.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3510]

Contents

1 Introduction

Object that exhibit some kind of facets are found in many scientific fields. The word facet is used here in a general form for any kind of surfaces that – idealized – would be planar and adjacent to each other forming distinct edges, e.g. crystals[1], closely packed organic cells[2], precipitates in alloys[3] or other materials[4,5]. However, due to experimental or imaging limitations (e.g. preparation "dirt", image resolution or artifacts from tomographic reconstruction) their digital representations are often distorted, uneven and separated by rounded edges like in Fig. 1. In some cases it is not the object that is faceted but some kind of separating structure, e.g. the domain walls of magnetic domains[6] or the walls of "dry" foam cells. It is also possible that the surface of an object is only partially faceted like e.g. alveoli[7].

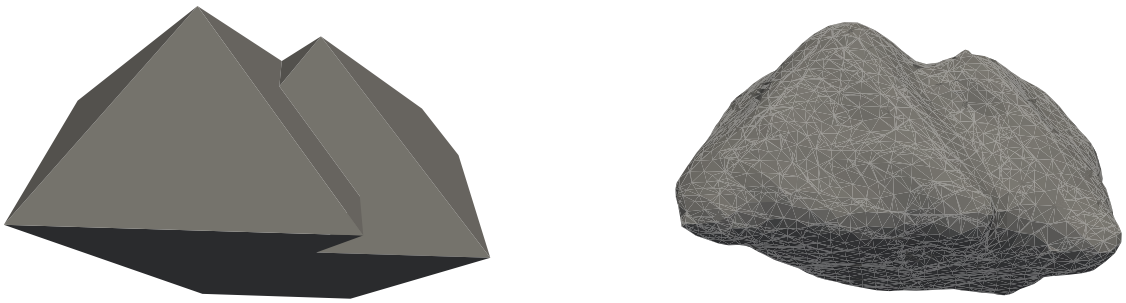

Figure 1: 3D test datasets
Left: An idealized test dataset of a crystal. Right: Dataset originating from the left one, but distorted to mimic artefacts commonly introduced by e.g. tomographic reconstruction.

It is common to characterize such objects by the angle that is formed by adjacent facets or between their normals. These interplanar angles are often measured manually, either in 2D images, ideally within a plane containing both normals, or in projections of 3D models. This is very tedious work, especially if many measurements are needed for reliable statistics. The method described in this article and its implementation in the contributed ParaView plugin makes automated identification and quantification of three-dimensional faceted objects possible. It is based on the algorithm described by Grothausmann et al.[1] and designed to even handle very distorted and rough facets. Its main parameters are based on the measurement errors of the facet normal direction and the resolution limit of the facet sizes. These define the tolerance for the roughness of ideally planar facets. Apart from the determination of the interplanar angles, the relative and absolute facet sizes, the facet normals and centers the main output also labels the faceted regions. The second output of the plugin is an idealized hull of the input, constructed only from the facets found. The third output consists of the edges of this hull. Values that belong to two adjacent facets are assigned to these edges, like for example their interplanar angle. Together, they allow a very detailed characterization and visualization of the facets of objects, see Figs. 4, 5, 6.

2 Installation and Usage

Building the FacetAnalyser plugin (using dynamic linking) requires the following steps:

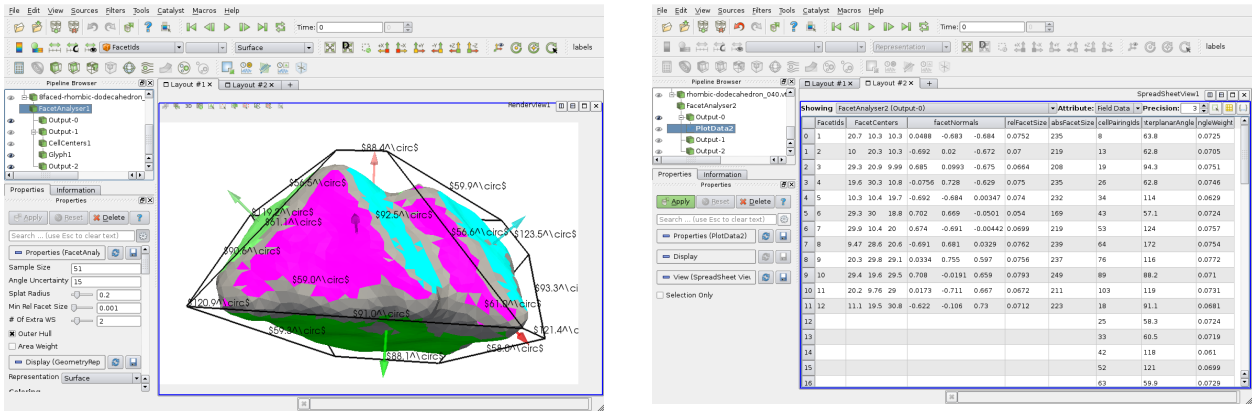

Figure 2: Screen shots of ParaView with the results of the plugin
Left: The plugin with its properties panel and the corresponding results rendered in the “3D view”. Right: The “Spread Sheet View” of the *Field Data* of the main output.

1. Build ParaView, ensuring that the shared library option is enabled.
   Create symbolic links to ParaView shared libraries so that ITK will link against appropriate versions of VTK etc.

2. Build ITK with the Review and ITKVtkGlue modules enabled, and with the compiler options that will use ParaView versions of libraries.

3. Build the FacetAnalyser plugin.

It is also possible to use static linking (might need `-fPIC`) or VTK separate from ParaView, for more details follow: http://public.kitware.com/pipermail/paraview/2015-January/033077.html.

2.1 Building ParaView

As long as this plugin is not included in releases of ParaView, it is necessary to compile ParaView (www.paraview.org) and ITK (www.itk.org) from source. General instructions for this can be found here:
www.paraview.org/Wiki/ParaView/Plugin_HowTo#Using_Plugins
www.paraview.org/Wiki/ParaView/User_Created_Plugins
www.itk.org/Wiki/ITK/Configuring_and_Building.

In short, configure ParaView with cmake as follows:

```
BUILD_SHARED_LIBS      ON
```

Compile ParaView with `make` and optionally install it. The following ITK compilation does not need ParaView to be installed.
After compilation change into the directory `/paraview-build-dir/lib/` and create symbolic links without the ParaView suffix. In a BASH for example with:

```
for i in *-pv*.so; do ln -s $i ${i%-pv*}.so; done
```

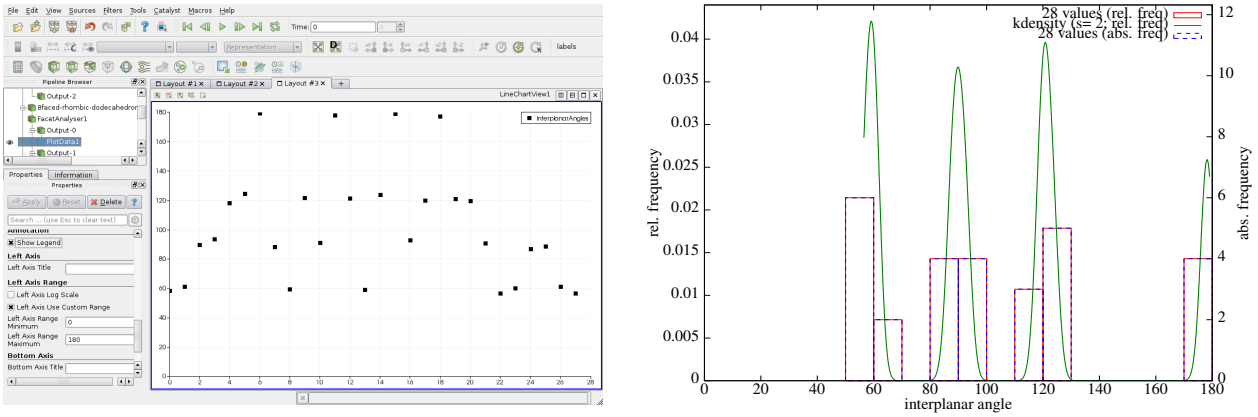

Figure 3: Plotting the interplanar angle values

Left: A ParaView scatter plot of the interplanar angles. Right: A kernel density plot of the exported interplanar angles, combined with relative and absolute frequency histograms.

2.2 Build ITK

Then configure ITK with cmake as follows, set `/paraview-build-dir/` to the build directory used for building ParaView:

```
Module_ITKVtkGlue      ON
Module_ITKReview       ON
VTK_DIR                /paraview-build-dir/VTK/
CMAKE_CXX_FLAGS        -L/paraview-build-dir/lib/
```

The two additionally enabled ITK modules are needed for the connection of VTK with ITK and for the watershed filters which still reside in ITKReview as of ITK-4.6.1. It is essential, that `VTK_DIR` is set to the build directory containing VTK shipped with ParaView.

2.3 Build FacetAnalyser plugin

Finally, create a build directory in the plugin source directory, change into it and configure the build with cmake again, specifying the `VTK_DIR` only if asked for:

```
BUILD_PLUGIN      ON
ITK_DIR           /itk-build-dir/
ParaView_DIR      /paraview-build-dir/
VTK_DIR           /paraview-build-dir/VTK/
```

Optionally choose `BUILD_EXAMPLE, BUILD_TESTING` and compile with `make`.

2.4 Load and test FacetAnalyser plugin in ParaView

If no errors occurred, start the self-compiled ParaView. To load the FacetAnalyser plugin into ParaView navigate in the main menu to *Tools→Mangage Plugins...* and click *Load New ...*. Choose the build directory of the FacetAnalyser and select the FacetAnalyser Plugin-library (*.so;*.dylib;*.dll;*.sl). The FacetAnalyser should now be listed in the plugin list where a tick can be set for auto-loading. After closing the plugin-

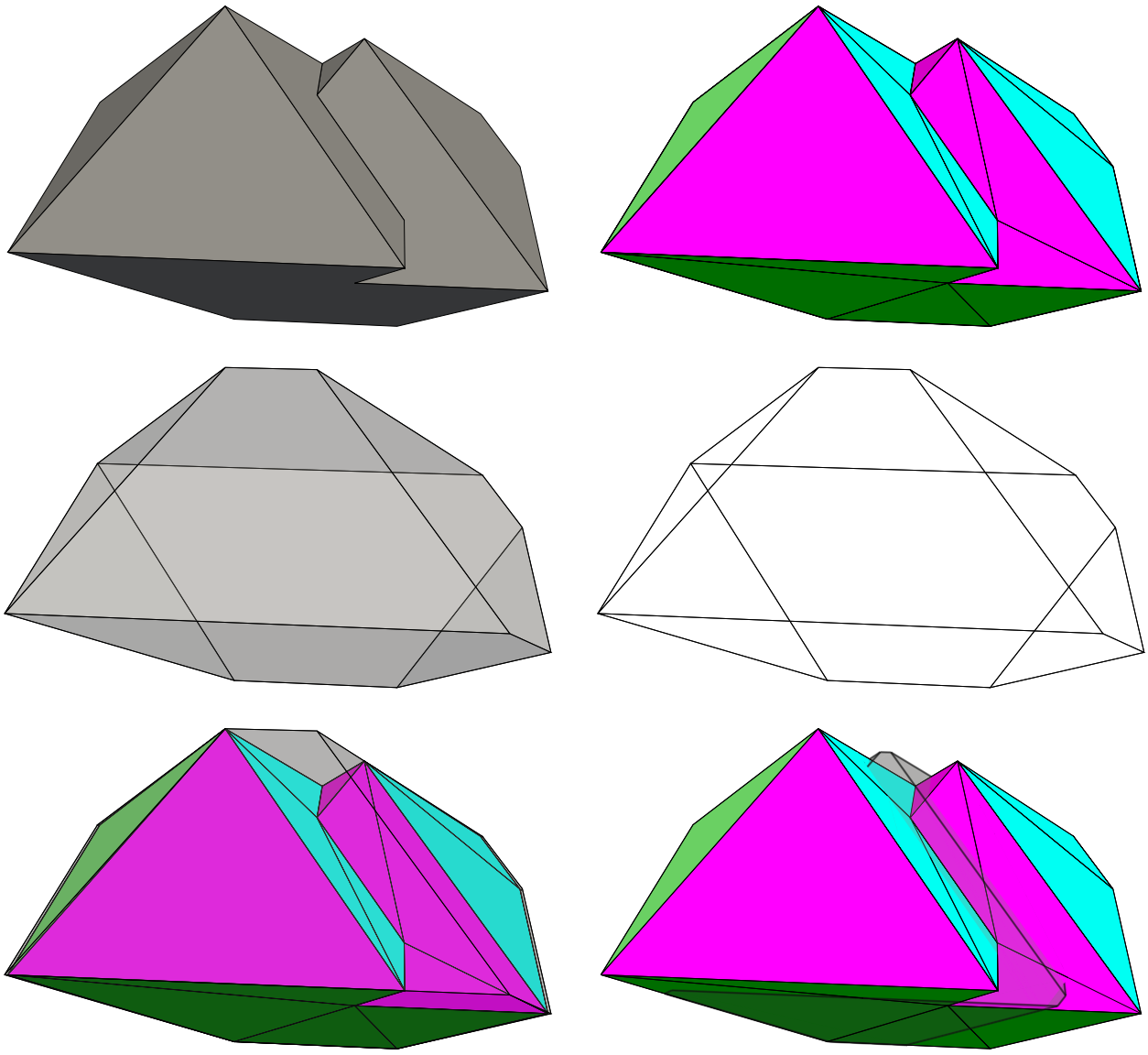

Figure 4: The idealized test dataset and its resulting outputs
Top left: The input vtkPolyData to the plugin containing also quads and higher order polygons. Top right: The main output of the plugin consisting only of triangles, faces rendered with edges and colored according to `FacetIds`. Middle left: The outer hull created by the plugin (second output). Middle right: The edges of the outer hull created by the plugin (third output). Bottom left: The outer hull rendered together with the main output. Bottom right: The number weighted intermediate hull rendered together with the main output.

dialog the FacetAnalyser should be listed under *Filters→Alphabetical*. As it expects vtkPolyData as input it can only be applied if some vtkPolyData is loaded and selected in the *Pipeline Browser*, use e.g. the STL-file in the demos directory. After a FacetAnalyser instance has appeared in the *Pipeline Browser* select it and click *Apply* or change some of the plugin parameters first. Three outputs should appear after a while depending on the amount of input faces/triangles and the chosen parameters. Decreasing the *Sample Size*, the *Angle Uncertainty*, the *Splat Radius* will reduce the execution time of the plugin most.

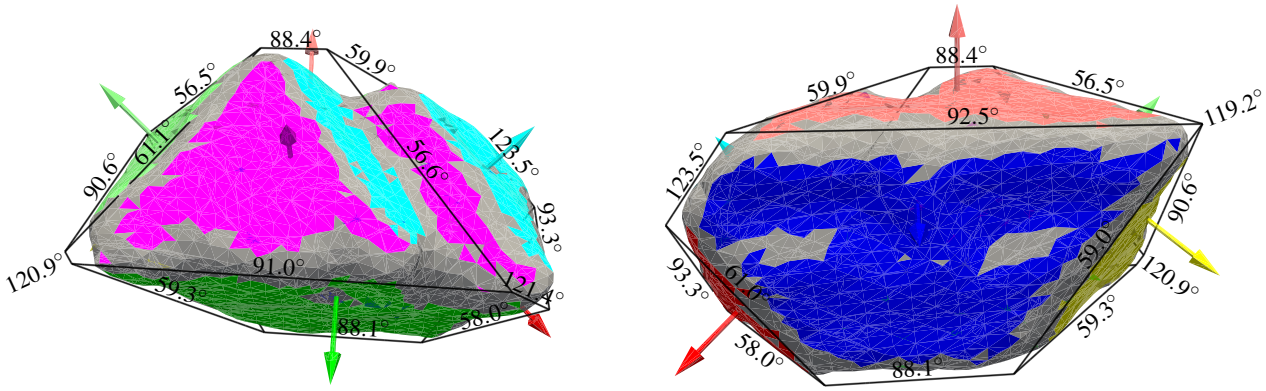

Figure 5: Visualization of the outputs of the plugin applied on the distorted mesh

Detected facets are colored according to their label, unfaceted regions are labeled with `0` and colored gray (color map included in the demos). The measured facet normals are depicted by arrow glyphs positioned at (parametric) cell centers. The interplanar angles are shown for facets that form an edge in the reconstructed area weighted hull. It can happen that a hull edge is so short, that it is hardly visible, see e.g. the interplanar angle of 119.2° in the right image. In order to find out the corresponding faces in such cases, each interplanar angle is associated with the `cellPairingId`. The complete list of interplanar angles is contained in the field data of the main output.

The *3D View* should now show a result similar to Fig. 2, left. Adding another layout tab with a *Spread Sheet View*, selecting the main output of the FacetAnalyser and choosing to display as *Attributes* the *Field Data* will reveal the analysis data as in Fig. 2, right. This data can then be plotted either within ParaView or exported as e.g. a CVS and plotted with e.g. gnuplot as in Fig. 3

3 Usage notes

When starting with a tomographic voxel dataset, a surface needs to be created first. This can be achieved in ParaView with the *Contour* filter or with e.g. `vtkDiscreteMarchingCubes`. The resulting surface (vtkPolyData) needs to be smoothed before the application of the *FacetAnalyser*, especially if `vtkDiscreteMarchingCubes` was used, see Fig. 1 in Ref. 1. Either apply the ParaView *Smooth* filter or e.g. `vtkWindowedSincPolyDataFilter`.

The most important parameters of the algorithm are adjustable in the Properties panel. The plugin comprises steps 3 to 10 and 12 described in Ref. 1. The *Sample Size*, the *Angle Uncertainty* and the *Splat Radius* allow to tune the roughness tolerance (see step 4 in Sec. 2.2 of Ref. 1).
The *Minimum Relative Facet Size* allow to reduce the amount of the finally detected facets. Note however, that it is an indirect measure and in way depends on the parameters mentioned before (step 6 in Sec. 2.2 of Ref. 1). It is not a concrete size threshold on the final analysis values, this can be achieved with a simple additional application of the ParaView *Threshold* filter.
It is also possible to choose the additional runs of watershed segmentations of higher order derivatives with the *# of Extra WS.*[8,9] This will generally lead to larger extents of the detected facets and their labeling, see Beare and Lehmann[9]. If set to `0` unfaceted regions will not be considered as in Fig. 4, otherwise unfaceted regions are colored gray as in Figs. 5 and 6). The *Field Data* the FacetAnalysis yields are based on the `itkLabelImageToStatisticsLabelMapFilter` (step 9 in Sec. 2.3 of Ref. 1).[10]
The *Outer Hull* and *Area Weight* control the type of hull and the edges created in the second and third outputs of the plugin, see Fig. 4.

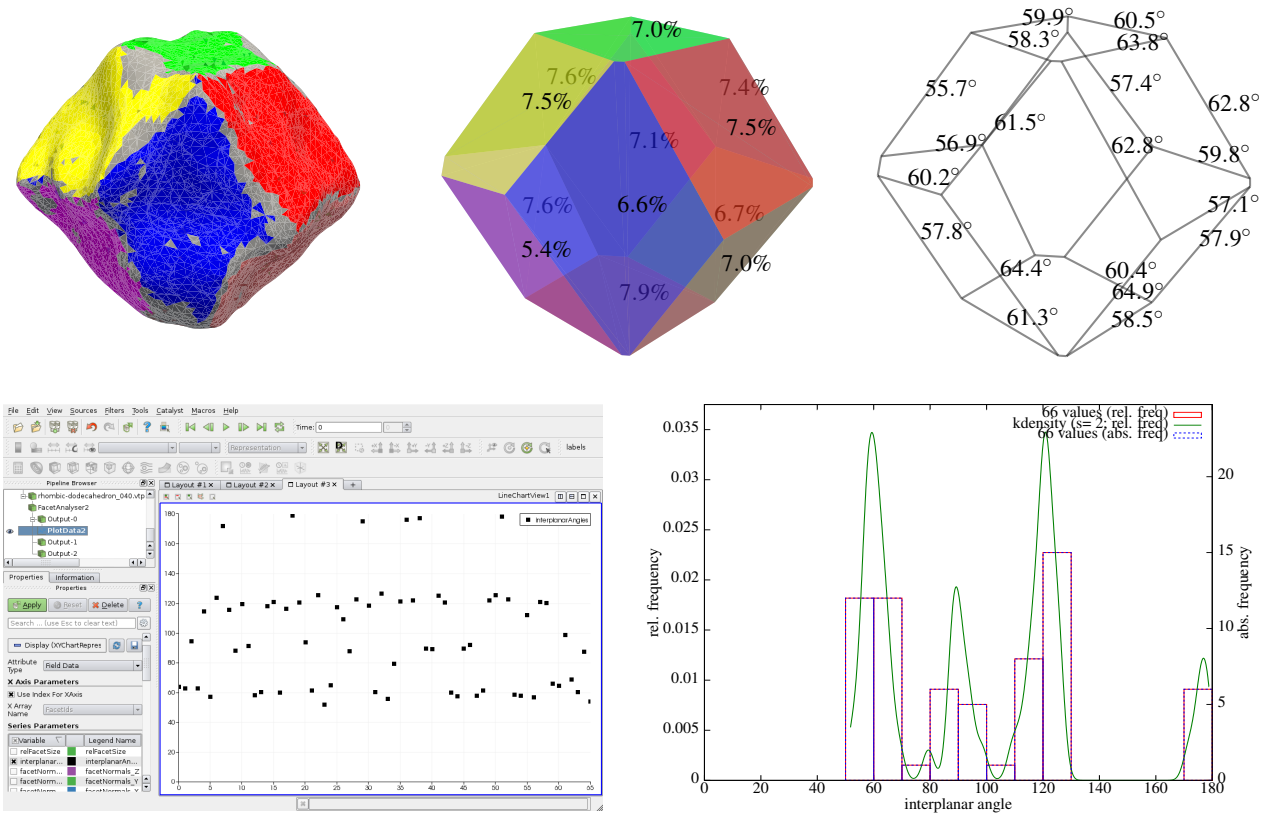

Figure 6: Rhombic-dodecahedron with 12 faces

Left: Facets detected on a distorted rhombic-dodecahedron. Middle: Generated outer hull used to display relative facet sizes of colored regions of left image. Right: Generated edges of the hull used to display all interplanar angles in the range from 55° to 65°. Bottom Left: A ParaView scatter plot of all interplanar angles. Bottom Right: A kernel density plot of the exported interplanar angles, combined with relative and absolute frequency histograms.

The presented contribution can also be used without ParaView only as a VTK filter. The included `FacetAnalyserExample` is a command-line example program for this type of usage.

The time consuming filters in the internal pipeline report their progress to `stderr` on the command line. The computation of the *FacetCenters* is based on `vtkCellCenters` which calculates the parametric center of the cell, not necessarily the geometric or bounding box center.

Splatting of single voxel is needed for the quantification of the detected facets (step 8 in Sec. 2.3 of Ref. 1). However, setting the radius of `vtkGaussianSplatter` to `0.0` results in an empty image. In order to make `vtkGaussianSplatter` to function nearly as expected in this case, the radius can be set to `1.0/(this->SampleSize+1)` to yield splats of about a voxel. Test showed that this does not work well in some cases as it can still result in multiple splatted voxels for a single input point. Therefore, a modified version `vtkGaussianSplatterExtended` is included to avoid the need for patching VTK.

4 Conclusions

A method to analyze faceted objects that exhibit distortions in their digital representation is presented in Ref. 1. The ParaView plugin described in this article allows easy access to that method. It is based on VTK

and ITK. The contributed functionality can also be used outside ParaView in e.g. command-line programs, an example and a test program are included. If you find this contribution useful, please cite Ref. 1.

5 Acknowledgement

Thanks go to Matt McCormick for his help concerning building ITK with VTK bundled with ParaView.

References

[1] Roman Grothausmann, Sebastian Fiechter, Richard Beare, Gaëtan Lehmann, Holger Kropf, Goarke Sanjeeviah Vinod Kumar, Ingo Manke, and John Banhart. Automated quantitative 3D analysis of faceting of particles in tomographic datasets. *Ultramicroscopy*, 122(0):65 – 75, 2012. ISSN 0304-3991. doi: 10.1016/j.ultramic.2012.07.024. (document), 1, 1, 3, 4

[2] Uzuki Matsushima, André Hilger, Wolfgang Graf, Simon Zabler, Ingo Manke, Martin Dawson, Gerard Choinka, and Werner B. Herppich. Calcium oxalate crystal distribution in rose peduncles: Non-invasive analysis by synchrotron X-ray micro-tomography. *Postharvest Biology and Technology*, 72 (0):27–34, October 2012. ISSN 0925-5214. doi: 10.1016/j.postharvbio.2012.04.013. 1

[3] M. Timpel, N. Wanderka, R. Grothausmann, and J. Banhart. Distribution of Fe-rich phases in eutectic grains of Sr-modified Al-10wt% Si-0.1wt% Fe casting alloy. *Journal of Alloys and Compounds*, 558: 18–25, May 2012. doi: 10.1016/j.jallcom.2012.12.009. 1

[4] C Kübel, K Gries, R Kröger, M Fritz, and A Rosenauer. Microstructure of Aragonite Platelets in Nacre. *Microscopy and Microanalysis*, 15:900–901, 2009. doi: 10.1017/S1431927609097177. 1

[5] Christian Kübel and Ute Kaiser. Electron Tomographic Characterization of ErSi2 and GexSi1-x Nanoparticles Prepared by Doping of 4H-SiC. *Microscopy and Microanalysis*, 12:1546–1547, 2006. doi: 10.1017/S1431927606064932. 1

[6] I. Manke, N. Kardjilov, R. Schäfer, A. Hilger, M. Strobl, M. Dawson, C. Grünzweig, G. Behr, M. Hentschel, C. David, A. Kupsch, A. Lange, and J. Banhart. Three-dimensional imaging of magnetic domains. *Nature Communications*, 1(125):6, Nov. 2010. doi: 10.1038/ncomms1125. 1

[7] Ewald R. Weibel. *Morphometry of the Human Lung*. Springer Berlin Heidelberg, 1963. doi: 10.1007/978-3-642-87553-3_6. 1

[8] Richard Beare. Optimization of connected component labelling. *Insight Journal*, (75), Apr 2006. URL http://hdl.handle.net/1926/168. 3

[9] Richard Beare and Gaëtan Lehmann. The watershed transform in ITK - discussion and new developments. *Insight Journal*, (92):1–24, June 2006. URL http://hdl.handle.net/1926/202. 3

[10] Gaëtan Lehmann. Label object representation and manipulation with ITK. *Insight Journal*, (176): 1–34, Aug 2008. URL http://hdl.handle.net/1926/584. 3