Providing values of adjacent voxel with vtkDiscreteMarchingCubes

Release 1.00

Roman Grothausmann[1]

July 21, 2016

[1]grothausmann.roman@mh-hannover.de,
Institute of Functional and Applied Anatomy, Hannover Medical School and
REBIRTH Cluster of Excellence, Hannover, Germany

Abstract

The contribution to VTK presented in this article is an extension to `vtk::vtkDiscreteMarchingCubes` to also create `vtk::vtkPointData` scalars containing the value of the adjacent voxel. These can be used to remove regions of the marching-cubes[1] mesh depending on the local neighborhood. The extension is based on the code of `vtkDiscreteMarchingCubes` of VTK-6.3.0 and is available on GitLab `https://gitlab.kitware.com/vtk/vtk/merge_requests/889` (and GitHub `https://github.com/Kitware/VTK/pull/18`).

Latest version available at the Insight Journal [`http://hdl.handle.net/10380/3559`]

Contents

1 Introduction

This extension to `vtkDiscreteMarchingCubes` makes it possible to let `vtkDiscreteMarchingCubes` also create `vtkPointData` scalars containing the label value of the neighboring voxel in a label image. This then allows to remove regions of the resulting mesh that are adjacent to specific labels. For example a label image that was created by running a watershed filter on a distance map followed by masking with the original binary segmentation, a common procedure to separate regions in a binary image at constrictions (see e.g. Beare and Lehmann [3]). Such a label image is shown in Fig. 1.

When creating meshes for each foreground label from such a label image the meshes are closed at the constrictions where the separation was introduced, see 3D view in Fig. 1 and Fig. 2.

With the contributed extension to `vtkDiscreteMarchingCubes` the resulting mesh contains additional points-scalars that hold the label value of the adjacent label for each point. A simple threshold that extracts the regions of the mesh that are adjacent to the background label (0) then makes it possible to remove the capping at constrictions, see Fig. 2, 3.

For example looking at label 19 and 21 (extracted with a threshold on the *cell-scalars* generated by `vtkDiscreteMarchingCubes`) which are adjacent to each other but mostly separated by the background label, Fig. 3. An additional threshold on the *point-scalars* then removes the capping at the separation, Fig. 3.

An advantage of this kind of capping removal is that it is even possible after the mesh was modified by other filters, e.g. smoothed by `vtkWindowedSincPolyDataFilter` avoiding borders. In most cases the resulting meshes are not connected on the open border edges any more, i.e. there will be gaps between them. Such gaps can be avoided with `vtkCoplanarSurfaceExtractor` [4] but it has to be applied before smoothing.

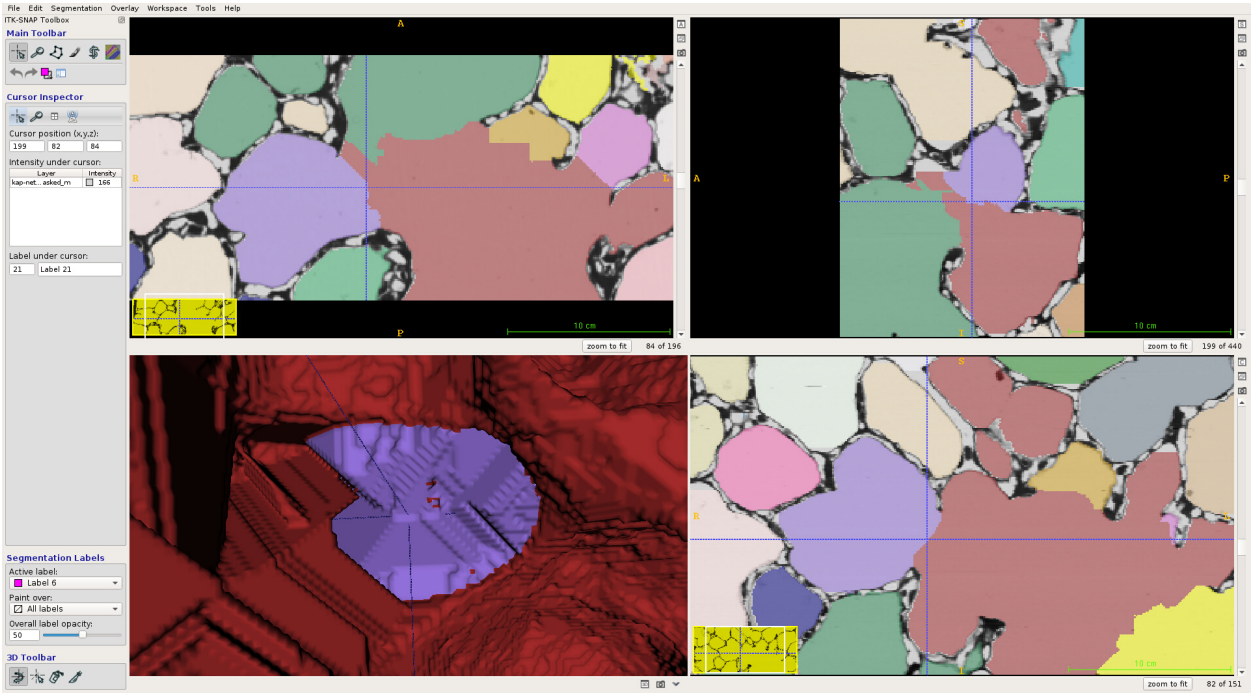

Figure 1: Segmentation with touching labels

Slices of a label image that separates a segmentation at constrictions and also contains a background label (0, rendered fully transparent). The cursor is on label 21 (light purple) which is adjacent to label 19 (dark pink). This is a screen-shot of ITKSnap [2].

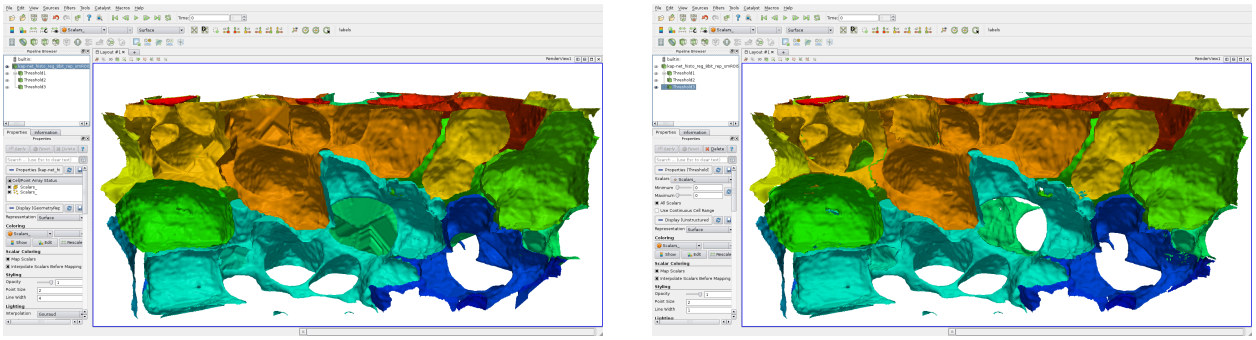

Figure 2:

left: Meshes generated by `vtkDiscreteMarchingCubes` for the label image shown above, colored differently and smoothed with `vtkWindowedSincPolyDataFilter`. Note the capping at constrictions.

right: Result only displaying those regions of the meshes that were adjacent to the background label. Note that the capping is removed.

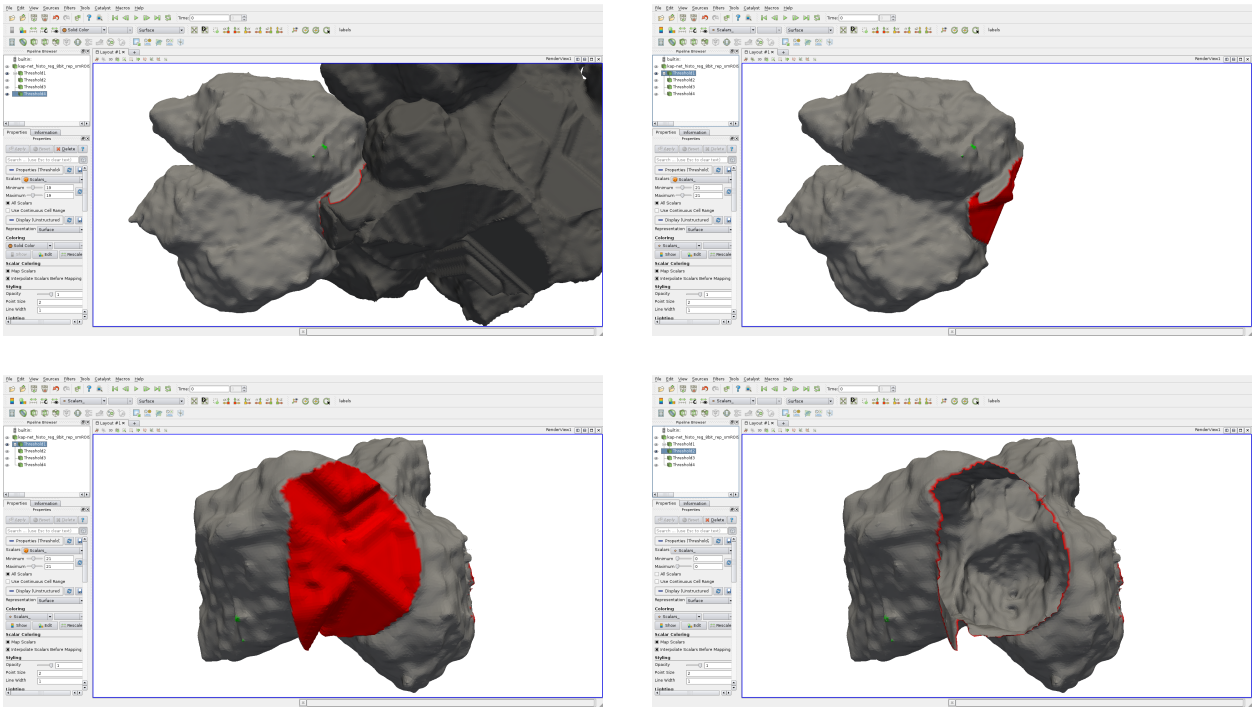

Figure 3:

Showing only the meshes of label 21 (light grey) and 19 (dark grey) (top left). Regions of the mesh of label 21 are colored if adjacent to other labels, e.g. red where 19 and 21 touch, i.e. where the separation took place. Other images show views of label 21 only, in the bottom left the capping is removed.

2 Implementation

Below the partial listing of the patch that contains the essential changes:

```
diff --git a/Filters/General/vtkDiscreteMarchingCubes.cxx b/Filters/General/
    vtkDiscreteMarchingCubes.cxx
index 3c04e25..93e04d2 100644
--- a/Filters/General/vtkDiscreteMarchingCubes.cxx
+++ b/Filters/General/vtkDiscreteMarchingCubes.cxx
@@ -217,7 +220,21 @@ void vtkDiscreteMarchingCubesComputeGradient(
               x[2] = x1[2] + t * (x2[2] - x1[2]);

               // add point
-              locator->InsertUniquePoint(x, ptIds[ii]);
+              if ( locator->InsertUniquePoint(x, ptIds[ii]) )
+                 {
+                 if (ComputeAdjacentScalars)
+                    {
+                    // check which vert holds the neighbour value
+                    if (s[vert[0]] == value)
+                       {
+                       newPointScalars->InsertTuple(ptIds[ii],&s[vert[1]]);
+                       }
+                    else
+                       {
+                       newPointScalars->InsertTuple(ptIds[ii],&s[vert[0]]);
+                       }
+                    }
+                 }
             // check for degenerate triangle
             if ( ptIds[0] != ptIds[1] &&
```

3 Testing (usage example)

A test based on `TestDiscreteMarchingCube.py` was created to monitor the newly added functionality:

```
diff --git a/Filters/General/Testing/Python/TestDiscreteMarchingCubesAdjacentScalars.
    py b/Filters/General/Testing/Python/TestDiscreteMarchingCubesAdjacentScalars.py
index 07417c0..6d9ebaf 100755
--- a/Filters/General/Testing/Python/TestDiscreteMarchingCubesAdjacentScalars.py
+++ b/Filters/General/Testing/Python/TestDiscreteMarchingCubesAdjacentScalars.py
@@ -76,10 +76,21 @@ while i < n:
 discrete = vtk.vtkDiscreteMarchingCubes()
 discrete.SetInputData(blobImage)
 discrete.GenerateValues(n, 1, n)
+discrete.ComputeAdjacentScalarsOn() # creates PointScalars
+
+thr = vtk.vtkThreshold()
+thr.SetInputConnection(discrete.GetOutputPort())
+thr.SetInputArrayToProcess(0, 0, 0, vtk.vtkDataObject.FIELD_ASSOCIATION_POINTS, vtk.
    vtkDataSetAttributes.SCALARS) # act on PointScalars created by
    ComputeAdjacentScalarsOn
+thr.AllScalarsOn() # default, changes better visible
+thr.ThresholdBetween(0, 0) # remove cells between labels, i.e. keep cells
    neighbouring background (label 0)
```

```
+
+vtu2vtp = vtk.vtkGeometryFilter()
+vtu2vtp.SetInputConnection(thr.GetOutputPort())

 mapper = vtk.vtkPolyDataMapper()
-mapper.SetInputConnection(discrete.GetOutputPort())
+mapper.SetInputConnection(vtu2vtp.GetOutputPort())
 mapper.SetLookupTable(lut)
+mapper.SetScalarModeToUseCellData() # default is to use PointScalars, which get
    created with ComputeAdjacentScalarsOn
 mapper.SetScalarRange(0, lut.GetNumberOfColors())

 actor = vtk.vtkActor()
```

4 Conclusions

The extension to `vtkDiscreteMarchingCubes` presented here allows to selectively remove the surface between adjacent labels. This is particularly useful when objects of a binary image were separated at constrictions using the common approach of a watershed filter on the distance map. Using `vtkDiscreteMarchingCubes` then to create surface meshes of the resulting label image even allows smoothing before any capping removal, which avoids unsmoothed or distorted boundaries.

5 Acknowledgement

Thanks go to all who gave feedback on this at GitLab, GitHub and the mailing-list.

References

[1] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21:163–169, July 1987. ISSN 0097-8930. doi: 10.1145/37402.37422. (document)

[2] Paul Yushkevich et al. ITK-SNAP. open source, 3.2. URL http://www.itksnap.org. see Yushkevich et al.[5]. 1

[3] 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. 1

[4] Roman Grothausmann. Extracting Intersections of Coplanar Surfaces (Boolean-operation on touching meshes). *The VTK Journal*, (949):7, December 2014. doi: http://hdl.handle.net/10380/3504. URL http://www.vtkjournal.org/browse/publication/949. 1

[5] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *NeuroImage*, 31(3):1116 – 1128, 2006. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2006.01.015. 5