

Lib/helpers.py

```
1 # =====
2 # Volume / scene helpers
3 # =====
4
5 import os
6
7 import numpy as np
8 import SimpleITK as sitk
9 import vtk
10 import vtk.util.numpy_support
11 import slicer
12
13 # TODO: look for external RAS<->IJK conversion with bounds clamping
14 # From: https://discourse.slicer.org/t/ras-to-ijk-conversion-issues/41863
15 def rasToIjk(volumeNode, rasPoint, clamp=False):
16     imageData = volumeNode.GetImageData()
17     if imageData is None:
18         return None
19     m = vtk.vtkMatrix4x4()
20     volumeNode.GetRASToIJKMatrix(m)
21     h = m.MultiplyPoint([rasPoint[0], rasPoint[1], rasPoint[2], 1.0])
22     ijk = [int(round(h[0])), int(round(h[1])), int(round(h[2]))]
23     dims = imageData.GetDimensions()
24     for axis in range(3):
25         if ijk[axis] < 0 or ijk[axis] >= dims[axis]:
26             if clamp:
27                 ijk[axis] = max(0, min(ijk[axis], dims[axis] - 1))
28             else:
29                 return None
30     return tuple(ijk)
31
32
33 def scalarArray(volumeNode):
```

```
34     """Return a (Z,Y,X) float32 copy of the volume's voxel array."""
35     arr = slicer.util.arrayFromVolume(volumeNode)
36     if arr.ndim == 4:
37         arr = arr[..., 0]
38     if arr.dtype != np.float32:
39         arr = arr.astype(np.float32)
40     else:
41         arr = arr.copy()
42     return arr
43
44
45 # TODO: Maybe use slicer.util.updateVolumeFromArray for dtype mapping, image data allocation, default display nodes.
46 # Currently claude implemented vtkImageData manually.
47 def pushArray(name, array, referenceVolume,
48              nodeClass="vtkMRMLScalarVolumeNode"):
49     """Replace any existing scene node named *name* with one holding *array*."""
50     slicer.mrmlScene.StartState(slicer.mrmlScene.BatchProcessState)
51     old = slicer.mrmlScene.GetFirstNodeByName(name)
52     if old is not None:
53         slicer.mrmlScene.RemoveNode(old)
54     node = slicer.mrmlScene.AddNewNodeByClass(nodeClass, name)
55     node.SetOrigin(referenceVolume.GetOrigin())
56     node.SetSpacing(referenceVolume.GetSpacing())
57     m = vtk.vtkMatrix4x4()
58     referenceVolume.GetIJKToRASDirectionMatrix(m)
59     node.SetIJKToRASDirectionMatrix(m)
60
61     dtypeMap = {
62         np.dtype(np.float32): vtk.VTK_FLOAT,
63         np.dtype(np.float64): vtk.VTK_DOUBLE,
64         np.dtype(np.uint8):    vtk.VTK_UNSIGNED_CHAR,
65     }
66     vtkType = dtypeMap.get(array.dtype, vtk.VTK_FLOAT)
67     if array.dtype not in dtypeMap:
68         array = array.astype(np.float32)
69
```

```
70     imageData = vtk.vtkImageData()
71     imageData.SetDimensions(*reversed(array.shape))
72     imageData.AllocateScalars(vtkType, 1)
73     v = vtk.util.numpy_support.vtk_to_numpy(
74         imageData.GetPointData().GetScalars())
75     v[:] = array.ravel()
76     imageData.Modified()
77     node.SetAndObserveImageData(imageData)
78     node.CreateDefaultDisplayNodes()
79     slicer.mrmlScene.EndState(slicer.mrmlScene.BatchProcessState)
80     return node
81
82
83 def createSegmentationFromMask(mask, referenceVolumeNode,
84                               segName="VSA_Vessels"):
85     slicer.mrmlScene.StartState(slicer.mrmlScene.BatchProcessState)
86     for n in [segName, "_VSA_temp_labelmap"]:
87         old = slicer.mrmlScene.GetFirstNodeByName(n)
88         if old is not None:
89             slicer.mrmlScene.RemoveNode(old)
90
91     labelmap = slicer.mrmlScene.AddNewNodeByClass(
92         "vtkMRMLLabelMapVolumeNode", "_VSA_temp_labelmap")
93     labelmap.SetOrigin(referenceVolumeNode.GetOrigin())
94     labelmap.SetSpacing(referenceVolumeNode.GetSpacing())
95     m = vtk.vtkMatrix4x4()
96     referenceVolumeNode.GetIJKToRASDirectionMatrix(m)
97     labelmap.SetIJKToRASDirectionMatrix(m)
98
99     imageData = vtk.vtkImageData()
100    imageData.SetDimensions(*reversed(mask.shape))
101    imageData.AllocateScalars(vtk.VTK_UNSIGNED_CHAR, 1)
102    v = vtk.util.numpy_support.vtk_to_numpy(
103        imageData.GetPointData().GetScalars())
104    v[:] = mask.astype(np.uint8).ravel()
105    imageData.Modified()
```

```
106     labelmap.SetAndObserveImageData(imageData)
107
108     segNode = slicer.mrmlScene.AddNewNodeByClass(
109         "vtkMRMLSegmentationNode", segName)
110     segNode.CreateDefaultDisplayNodes()
111     slicer.mrmlScene.EndState(slicer.mrmlScene.BatchProcessState)
112
113     slicer.modules.segmentations.logic().ImportLabelmapToSegmentationNode(
114         labelmap, segNode)
115
116     seg = segNode.GetSegmentation()
117     if seg.GetNumberOfSegments() > 0:
118         sid = seg.GetNthSegmentID(0)
119         s = seg.GetSegment(sid)
120         s.SetColor(1.0, 0.2, 0.2)
121         s.SetName("Vessels")
122
123     d = segNode.GetDisplayNode()
124     if d:
125         d.SetVisibility3D(True)
126         d.SetVisibility2D(True)
127         d.SetOpacity3D(0.6)
128
129     slicer.mrmlScene.RemoveNode(labelmap)
130     segNode.CreateClosedSurfaceRepresentation()
131     return segNode
132
133
134 # =====
135 # Export
136 # =====
137
138 def exportBinarySlices(binaryMask, outDir, prefix="slice", axis=0,
139                       extension="png"):
140     """Write each 2D slice of a binary mask as an 8-bit image (0/255).
141     Format is chosen by *extension* – SimpleITK selects the writer."""
```

```
142     os.makedirs(outDir, exist_ok=True)
143     nSlices = binaryMask.shape[axis]
144     digits = max(3, len(str(nSlices)))
145     for i in range(nSlices):
146         sl = (np.take(binaryMask, i, axis=axis).astype(np.uint8) * 255)
147         img = sitk.GetImageFromArray(np.ascontiguousarray(sl))
148         sitk.WriteImage(
149             img,
150             os.path.join(outDir, f"{prefix}_{i:0{digits}d}.{extension}"),
151         )
152     return nSlices
153
```