图像处理与可视化 Homework-8 报告

林子开 21307110161

2023 年 12 月 21 日

目录

1 在 VTK 中实现面绘制和体绘制

在 VTK 中进行数据可视化的流程如 b 下：

$$\text{Source/Reader} \rightarrow \text{Filter} \rightarrow \text{Mapper} \rightarrow \text{Actor} \rightarrow \text{Renderer} \rightarrow \text{Render Window} \rightarrow \text{Interactor}$$

在本次试验中，统一使用老师提供的 image_lr.nii.gz 数据。

1.1 面绘制

面绘制的 Python 代码如下：

Listing 1: 面绘制的 Python 代码

```python
"""面绘制"""

import nibabel as nib
import vtk


img1 = nib.load('image_lr.nii') # load and save
img1_data = img1.get_fdata() # 获取标量场数据
dims = img1.shape #[124,124,73] # 数据场的维度
spacing = (img1.header['pixdim'][1],img1.header['pixdim'][2],img1.header['pixdim'][3]) # 间隔

image = vtk.vtkImageData() # 生成vtkImageDate对象
image.SetDimensions(dims[0],dims[1],dims[2]) # 设置vtkImageData对象的维度
image.SetSpacing(spacing[0],spacing[1],spacing[2]) # 设置间隔
image.SetOrigin(0,0,0)

if vtk.VTK_MAJOR_VERSION <= 5:
```

```
    image.SetNumberOfScalarComponents(1) # vtkImageData schalarArray tuple size
    image.SetScalarTypeToDouble()
else:
    image.AllocateScalars(vtk.VTK_DOUBLE,1)

# fill every entry of the image data
for z in  range(dims[2]):
    for y in  range(dims[1]):
        for x in  range(dims[0]):
            # 将图像标量场数据填入vtkImageData对象的scalar属性中
            scalarData = img1_data[x][y][z]
            image.SetScalarComponentFromDouble(x,y,z,0,scalarData)

Extractor = vtk.vtkMarchingCubes() # 移动立方体算法对象，得到等值面
Extractor.SetInputData(image) # 输入数据
Extractor.SetValue(0,150) # 设置value，求value=150的等值面

stripper = vtk.vtkStripper() # 建立三角带对象
stripper.SetInputConnection(Extractor.GetOutputPort()) # 输入数据，将生成的三角片连接成三角带

mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(stripper.GetOutputPort())
# mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

actor.GetProperty().SetColor(1,1,0)
actor.GetProperty().SetOpacity(0.95)
actor.GetProperty().SetAmbient(0.05)
actor.GetProperty().SetDiffuse(0.5)
actor.GetProperty().SetSpecular(1.0)

ren = vtk.vtkRenderer()
ren.SetBackground(1,1,1)
ren.AddActor(actor)
renWin = vtk.vtkRenderWindow()

renWin.AddRenderer(ren)
renWin.SetSize(750,750)

iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
renWin.Render()
iren.Start()
```

面绘制的效果如下：

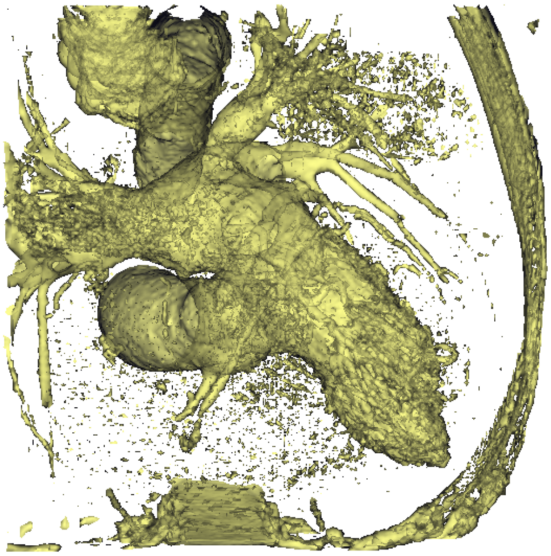


图 1: 面绘制效果

可以看出，图中有很多碎片。

1.2 体绘制

体绘制的 Python 代码如下：

Listing 2: 体绘制的 Python 代码

```
"""体绘制"""

import nibabel as nib
import vtk
import numpy as np


img1 = nib.load('image_lr.nii') # load and save
img1_data = img1.get_fdata() # 获取标量场数据
dims = img1.shape #[124,124,73] # 数据场的维度
spacing = (img1.header['pixdim'][1],img1.header['pixdim'][2],img1.header['pixdim'][3]) # 间隔

image = vtk.vtkImageData() # 生成vtkImageDate对象
image.SetDimensions(dims[0],dims[1],dims[2]) # 设置vtkImageData对象的维度
image.SetSpacing(spacing[0],spacing[1],spacing[2]) # 设置间隔
image.SetOrigin(0,0,0)
image.SetExtent(0, dims[0]-1, 0, dims[1]-1, 0, dims[2]-1)

image.AllocateScalars(vtk.VTK_UNSIGNED_SHORT, 1)

intRange = (20,500) # 设置感兴趣的灰度区域
max_u_short = 128
const = max_u_short / np.float64(intRange[1]-intRange[0])
for z in  range(dims[2]):
    for y in  range(dims[1]):
        for x in  range(dims[0]):
```

```
            scalarData = img1_data[x][y][z]
            scalarData = np.clip(scalarData,intRange[0],intRange[1]) # 超出范围的部分进行截断
            scalarData = const * np.float64(scalarData - intRange[0])
            image.SetScalarComponentFromFloat(x,y,z,0,scalarData)

## 设置传输函数的参数

# create transfer mapping scalar value to opacity
opacityTransferFunction = vtk.vtkPiecewiseFunction()
opacityTransferFunction.AddPoint(0, 0.0)
opacityTransferFunction.AddSegment(23, 0.3, 128, 0.5)
opacityTransferFunction.ClampingOff()

# create transfer mapping scalar value to color
colorTransferFunction = vtk.vtkColorTransferFunction()
colorTransferFunction.AddRGBSegment(0, 0.0, 0.0, 0.0, 20, 0.2, 0.2, 0.2)
colorTransferFunction.AddRGBSegment(20, 0.1, 0.1, 0, 128, 1, 1, 0) # intensity between 0

# grad to opacity transfer
gradientTransferFunction = vtk.vtkPiecewiseFunction()
gradientTransferFunction.AddPoint(0, 0.0)
gradientTransferFunction.AddSegment(100, 0.1, 1000, 0.3)


# the property dexcribes how the data will work
volumeProperty = vtk.vtkVolumeProperty()
volumeProperty.SetScalarOpacity(opacityTransferFunction)
volumeProperty.SetColor(colorTransferFunction)
volumeProperty.SetGradientOpacity(gradientTransferFunction)
volumeProperty.ShadeOn()
volumeProperty.SetInterpolationTypeToLinear() # 采样时使用线性插值，性价比高
volumeProperty.SetAmbient(1)
volumeProperty.SetDiffuse(0.9) # 漫反射
volumeProperty.SetSpecular(0.8) # 镜面反射
volumeProperty.SetSpecularPower(10) # 用于描述镜面反射的强度

# The mapper / ray cast function know how to render the data.
volumeMapper = vtk.vtkFixedPointVolumeRayCastMapper()
volumeMapper.SetInputData(image)
volumeMapper.SetImageSampleDistance(5.0)

# the volume holds the mapper and the property and can be used to position/orient the volume
volume = vtk.vtkVolume()
volume.SetMapper(volumeMapper)
volume.SetProperty(volumeProperty)

ren = vtk.vtkRenderer()
ren.SetBackground(1,1,1)
ren.AddVolume(volume)
renWin = vtk.vtkRenderWindow()

light = vtk.vtkLight()
```

```
light.SetColor(0,1,1)
ren.AddLight(light)

renWin.AddRenderer(ren)
renWin.SetSize(750,750)
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

renWin.Render()
iren.Initialize()
iren.Start()
```

体绘制的效果如下：

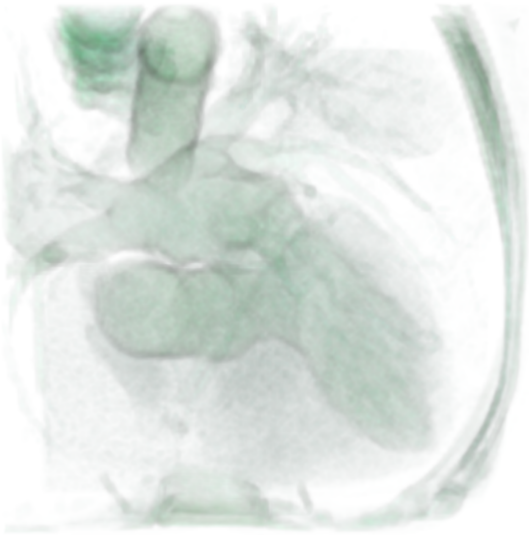


图 2: 体绘制效果

2 消除碎片

使用 `vtk.vtkSmoothPolyDataFilter()` 可以非常方便地消除碎片。消除碎片的 Python 代码如下：

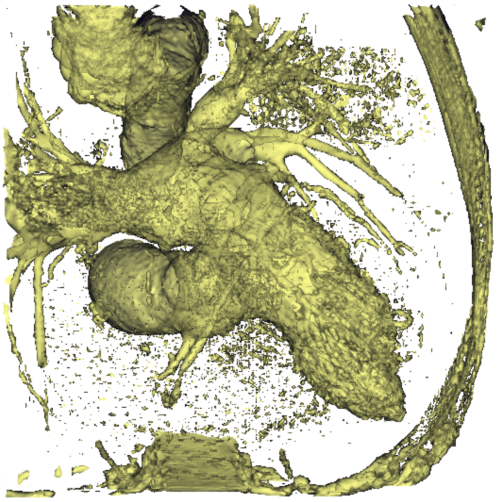
Listing 3: 消除碎片的 Python 代码

```
"""面绘制并消除碎片"""

import nibabel as nib
import vtk


img1 = nib.load('image_lr.nii') # load and save
img1_data = img1.get_fdata() # 获取标量场数据
dims = img1.shape #[124,124,73] # 数据场的维度
spacing = (img1.header['pixdim'][1],img1.header['pixdim'][2],img1.header['pixdim'][3]) # 间隔

image = vtk.vtkImageData() # 生成vtkImageDate对象
image.SetDimensions(dims[0],dims[1],dims[2]) # 设置vtkImageData对象的维度
```

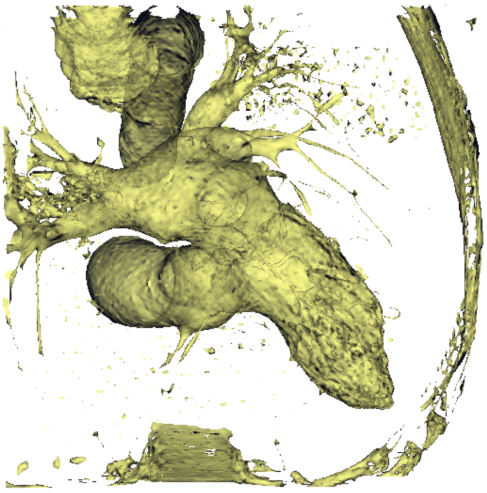
```
14 image.SetSpacing(spacing[0],spacing[1],spacing[2]) # 设置间隔
15 image.SetOrigin(0,0,0)
16
17 if vtk.VTK_MAJOR_VERSION <= 5:
18     image.SetNumberOfScalarComponents(1) # vtkImageData schalarArray tuple size
19     image.SetScalarTypeToDouble()
20 else:
21     image.AllocateScalars(vtk.VTK_DOUBLE,1)
22
23 # fill every entry of the image data
24 for z in  range(dims[2]):
25     for y in  range(dims[1]):
26         for x in  range(dims[0]):
27             # 将图像标量场数据填入vtkImageData对象的scalar属性中
28             scalarData = img1_data[x][y][z]
29             image.SetScalarComponentFromDouble(x,y,z,0,scalarData)
30
31
32 Extractor = vtk.vtkMarchingCubes() # 移动立方体算法对象，得到等值面
33 Extractor.SetInputData(image) # 输入数据
34 Extractor.SetValue(0,150) # 设置value，求value=150的等值面
35
36 # ----------------这部分是消除碎片的关键-------------------
37 # Smoothing
38 smoother = vtk.vtkSmoothPolyDataFilter() # 用于消除碎片
39 smoother.SetInputConnection(Extractor.GetOutputPort())
40 smoother.SetNumberOfIterations(1000)
41 # ---------------------------------------------------------
42
43 stripper = vtk.vtkStripper() # 建立三角带对象
44 stripper.SetInputConnection(smoother.GetOutputPort()) # 输入数据，将生成的三角片连接成三角带
45
46 mapper = vtk.vtkPolyDataMapper()
47 mapper.SetInputConnection(stripper.GetOutputPort())
48 # mapper.ScalarVisibilityOff()
49
50 actor = vtk.vtkActor()
51 actor.SetMapper(mapper)
52
53 actor.GetProperty().SetColor(1,1,0)
54 actor.GetProperty().SetOpacity(0.95)
55 actor.GetProperty().SetAmbient(0.05)
56 actor.GetProperty().SetDiffuse(0.5)
57 actor.GetProperty().SetSpecular(1.0)
58
59 ren = vtk.vtkRenderer()
60 ren.SetBackground(1,1,1)
61 ren.AddActor(actor)
62 renWin = vtk.vtkRenderWindow()
63
64 renWin.AddRenderer(ren)
65 renWin.SetSize(750,750)
```

```
66
67 iren = vtk.vtkRenderWindowInteractor()
68 iren.SetRenderWindow(renWin)
69 iren.Initialize()
70 renWin.Render()
71 iren.Start()
```

消除碎片前后的效果对比图如下：



(a) 消除碎片前



(b) 消除碎片后