可视化7

18300290007 加兴华

实验环境

python 3.9

vtk 9.0.20210612.dev0

python需第三方库： vtk， matplotlib.pyplot ， numpy， nibabel ， tqdm

文件说明

等值面渲染.py对应第一题（1）

体渲染.py对应对应第一题（2）

渲染去噪.py对应第二题

运行时需将image_lr.nii.gz文件置于python工作文件夹根目录

第一题

阅读了解VTK（VTK - The Visualization Toolkit, www.vtk.org），学习某个编程环境下调用VTK库进行可视化。调用可视化渲染引擎库VTK，实现三维体数据完整的渲染过程（如光照模型，颜色设置等）。需要实现的渲染过程包括：(1) 等值面渲染，(2) 体渲染。请自己找一个体数据进行测试和结果展示或使用image_lr.nii.gz。

提交作业需要对使用数据进行说明，如果使用自己的数据，请一并提交源数据（或数据下载的网上链接）。

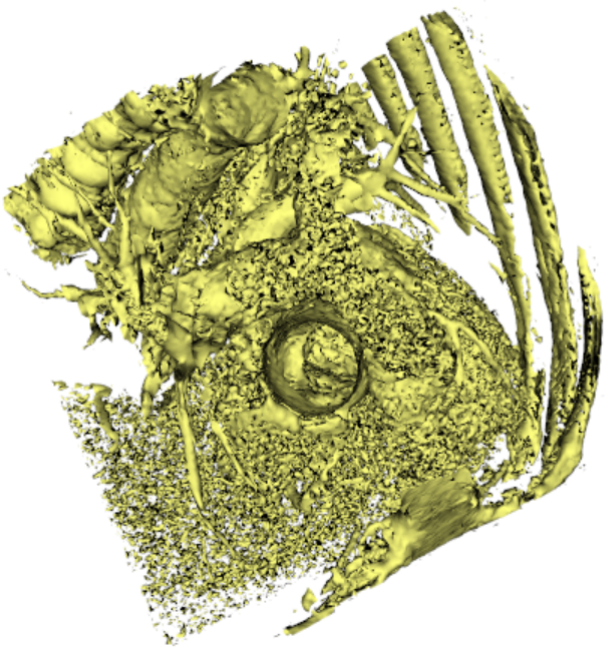
(1)

运行等值面渲染.py：

```
import vtk
import matplotlib.pyplot as plt
import numpy as np
import nibabel as nib

#使用nibabel读入图像数据img_data中
img=nib.load('image_lr.nii.gz')
img_data = img.get_fdata()
dims = img.shape
spacing = (img.header['pixdim'][1], img.header['pixdim'][2],
img.header['pixdim'][3])

#vtk的image data对象
image = vtk.vtkImageData()
image.SetDimensions(dims[0], dims[1], dims[2])
image.SetSpacing(spacing[0], spacing[1], spacing[2])
```

```
image.SetOrigin(0,0,0)

if vtk.VTK_MAJOR_VERSION<=5:
    image.SetNumberOfScalarComponents(1)
    image.SetScalarTypeToDouble()
else:
    image.AllocateScalars(vtk.VTK_DOUBLE,1)

#逐点输入3d数据
for z in range(dims[2]):
    for y in range(dims[1]):
        for x in range(dims[0]):
            scalardata = img_data[x][y][z]
            image.SetScalarComponentFromDouble(x,y,z,0,scalardata)

#使用Marching Cude算法进行面渲染
Extractor = vtk.vtkMarchingCubes()
Extractor.SetInputData(image)

#设置等值面
Extractor.SetValue(0, 100)
#Extractor.SetValue(1, 200)

# 先建立三角条带对象
stripper = vtk.vtkStripper()
stripper.SetInputConnection(Extractor.GetOutputPort()) #连接三角片
# 设置mapper，acter，renderer等
mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(stripper.GetOutputPort())
mapper.ScalarVisibilityOff()
actor = vtk.vtkActor()
actor.SetMapper(mapper)
actor.GetProperty().SetColor(1,1,0.5)
# actor.GetProperty().SetOpacity(0.9)
# actor.GetProperty().SetAmbient(0.25)
# actor.GetProperty().SetDiffuse(0.6)
# actor.GetProperty().SetSpecular(1.0)

#生成交互式窗口
ren = vtk.vtkRenderer()
ren.SetBackground(1,1,1)
ren.AddActor(actor)
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetSize(500, 500)
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
renWin.Render()
iren.Start()
```

结果如下（经过旋转调整）：



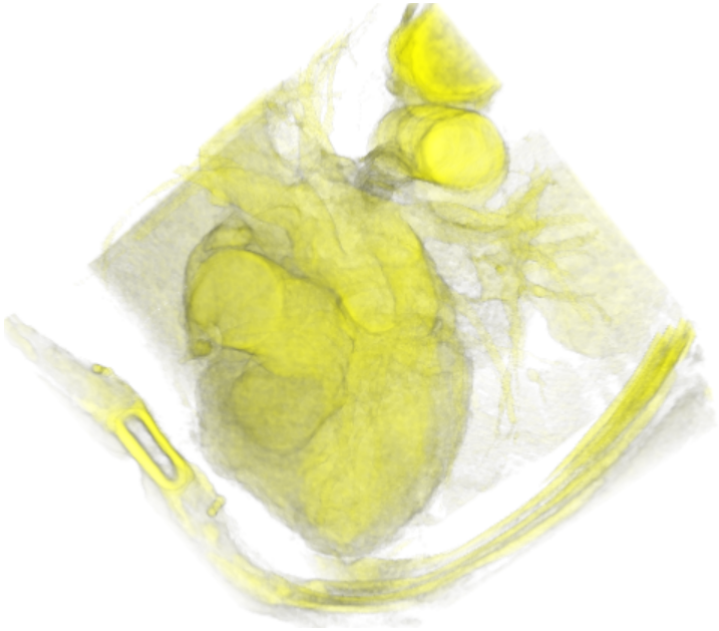
(2)

运行体渲染.py:

```
import nibabel as nib
import vtk
import numpy as np

#数据读取和定义
img = nib.load("image_lr.nii.gz")
img_data = img.get_fdata()
dims = img.shape
spacing = (img.header['pixdim'][1], img.header['pixdim'][2],
img.header['pixdim'][3])

#vtk的image对象声明
image = vtk.vtkImageData()
image.SetDimensions(dims[0], dims[1], dims[2])
image.SetSpacing(spacing[0], spacing[1], spacing[2])
image.SetOrigin(0,0,0)

#逐点赋值
image.AllocateScalars(vtk.VTK_DOUBLE, 1)
for z in range(dims[2]):
    for y in range(dims[1]):
        for x in range(dims[0]):
            scalardata = img_data[x][y][z]
            image.SetScalarComponentFromDouble(x,y,z,0,scalardata)

volumeProperty = vtk.vtkVolumeProperty()

#-------------设置传输函数参数，进行体渲染
#创建将标量值转换为不透明度的映射
compositeOpacity = vtk.vtkPiecewiseFunction()
compositeOpacity.AddSegment(0, 0, 10, 0)
compositeOpacity.AddSegment(10, 0.2, 120, 0.2)
#compositeOpacity.AddSegment(120, 0.2, 128, 0.4)
```

```
volumeProperty.SetScalarOpacity(compositeOpacity)

# 上色
colorFunction = vtk.vtkColorTransferFunction()
colorFunction.AddRGBSegment(0, 0, 0, 0, 20, 0.2, 0.2, 0.2)
#colorFunction.AddRGBSegment(10, 0.94, 0.9, 0.55, 120, 0.94, 0.9, 0.55)
colorFunction.AddRGBSegment(20, 0.1, 0.1, 0, 128, 1, 1, 0)
volumeProperty.SetColor(colorFunction)

# 标量值+梯度模
gradientTransferFunction=vtk.vtkPiecewiseFunction()
gradientTransferFunction.AddPoint(0,0.0)
gradientTransferFunction.AddSegment(100, 0.1, 1000, 0.3)
volumeProperty.SetGradientOpacity(gradientTransferFunction)

# 光学模拟
volumeProperty.SetInterpolationTypeToLinear()
volumeProperty.SetAmbient(1)
volumeProperty.SetDiffuse(0.9) # 漫反射
volumeProperty.SetSpecular(0.5) # 镜面反射
volumeProperty.SetSpecularPower(10)

# 光线投射量绘制
volumeMapper = vtk.vtkFixedPointVolumeRayCastMapper()
volumeMapper.SetInputData(image)
volumeMapper.SetImageSampleDistance(5.0)
# volume包含映射器和属性以及可用于定位体积
volume = vtk.vtkVolume()
volume.SetMapper(volumeMapper)
volume.SetProperty(volumeProperty)

#----------------结果生成
ren = vtk.vtkRenderer()
ren.SetBackground(1,1,1)
ren.AddActor(volume)
light=vtk.vtkLight()
light.SetColor(0,1,1)
ren.AddLight(light)
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(ren)
renWin.SetSize(500, 500)
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)
iren.Initialize()
renWin.Render()
iren.Start()
```

结果如下（经旋转调整）：



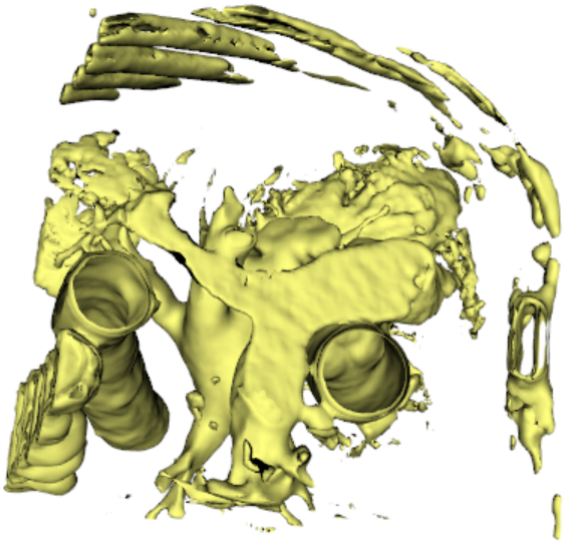
第二题

请设计一个方法消除心脏CT图像（image_lr.nii.gz）等值面渲染结果中的碎片化的面单元，如下图所示。并用代码实现和展示结果。

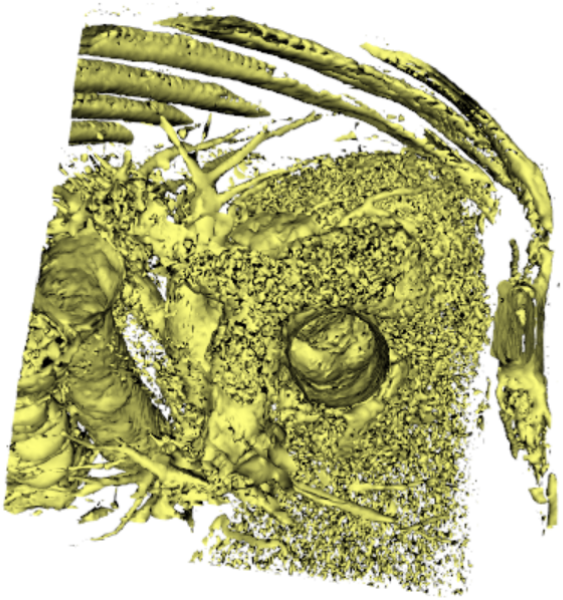
类似二维图片去噪，我考虑在等值面渲染过程中使用广义的均值算法进行平滑处理。

```
#------------------使用类似2d平滑滤波的方式进行3d平滑
#padding--零填充
m=np.zeros(shape=(dims[0]+2,dims[1]+2,dims[2]+2))
print('零填充进度')
for z in tqdm.tqdm(range(dims[2])):
    for y in range(dims[1]):
        for x in range(dims[0]):
            m[x+1][y+1][z+1]=img_data[x][y][z]
#平滑处理--均值法（3*3*3滑窗）
print('平滑处理进度')
for z in tqdm.tqdm(range(dims[2])):
    for y in range(dims[1]):
        for x in range(dims[0]):
            sumt=0
            for i in [-1,0,1]:
                for j in [-1,0,1]:
                    for k in [-1,0,1]:
                        sumt=sumt+m[x+k+1][y+j+1][z+i+1]
            sumt=sumt/27
            scalardata=int(sumt)
            image.SetScalarComponentFromDouble(x,y,z,0,scalardata)
```

运行渲染去噪.py，结果如下（经旋转调整）：



对比第一题中等值面法输出的相近角度：



可以很明显地观察到碎片化的面单元基本被清除了，说明基于广义平滑处理的算法可行。