第五章 可视化技术

前面的章节中我们已经介绍了一些数据渲染和交互的基本工具。本章中我们主要介绍一下可视化技术。这些技术（实现为Filters）根据其操作数据类型来组织。其中部分Filters是能够处理任何数据类型的一般性Filters-他们能够接收vtkDataSet（或者任意的子类）。而更多的Filters则是依赖于其处理的数据类型（如vtkPolyData）的专门性filters。其中处理vtkImageData（或者其子类vtkStructuredData）数据类型的一类Filters，本章中暂不涉及，在下一章中将具体讨论。（103页“图像处理和可视化”）

阅读本章时，有两点需要铭记在心。一是filters会产生多种输出数据类型，而这些类型并不一定与输入数据类型一致。第二，filters会通过组合来创建复杂的数据处理管线。下面的例子你会注意到一些常见的filters组合。

5.1 Visualizing vtkDataSet（and Subclasses）

vtkDataSet（及其子类）可视化

本章节主要讲解vtkDataSet数据对象的一些常用可视化操作。注意vtkDataSet是vtk中所有数据类型的超类类型（见图3-2）。因此，这里讲到的操作都适用于所有的数据类型。（换句话说，所有接收vtkDataSet类型的filters都可以接收vtkPolyData，vtkImageData，vtkStructured，vtkRectilinearGrid和vtkUnstructuredGrid类型。）

Working With Data Attribute 数据属性操作

数据属性是与数据结构相关的信息（如25页“数据管线”所述）。VTK中，属性数据分为点属性数据和单元属性数据。属性数据与结构数据一起被许多的filters处理产生新的结构和属性。关于属性数据的详解不是本章的重点，这里给出了一个简单的例子来讲述基本的思想。（更多信息请参阅362页“场和属性数据”和图16-1。）

数据属性是简单的vtkDataArrays类型，它可以是标量scalars，向量vectors，张量tensors，法向normals，纹理坐标texture coordinate，全局ids （global ids，如标识大量元素），或者pedigree ids（用来追踪管线中的元素历史）。vtkDataSet中的一个点或者单元都有独立的属性数据。数据属性可以关联vtkDataSet中的点或者单元。与vtkDataSet关联的vtkDataArray都是vtkDataArray的一个具体子类，例如vtkFloatArray或者vtkIntArray。这些数据数组可以看做连续的、线性的内存块。在内存块中，数据数组可以看做由子数据或者元组组成。创建属性数据即是根据类型实例化数组内存，制定元组大小，插入数据并与数据dataset关联，如下面Tcl脚本所示。属性数据类型可以指定为标量，向量，张量，纹理坐标或者法向。例如：

vtkFloatArray scalars

scalars InsertTuple1 0 1.0

scalars InsertTuple1 1 1.2

...etc...

vtkDoubleArray vectors

vectors SetNumberOfComponents 3

vectors InsertTuple3 0 0.0 0.0 1.0

vectors InsertTuple3 1 1.2 0.3 1.1

...ect...

vtkIntArray justAnArray

justAnArray SetNumberOfComponents 3

justAnArray SetNumberOfTuples $numberOfPoints

justAnArray SetName "Solution Attributes"

justAnArray SetTuple2 0 1 2

justAnArray SetTuple2 1 3 4

...etc...

vtkPolyData polyData; #A concrete type of vtkDataSet

[polyData GetPointData] SetScalars scalars

[polyData GetCellData] SetVectors vectors

[polyData GetPointData] AddArray justAnArray

这里创建了三种类型的属性数据，float，double和int。第一个数组scalars实例化后，默认的元组大小为1。InsertTuple1()方法用来向数组中插入数据（所有Insert\_\_\_()方法负责分配足够的内存来保存数据）。下一个数组vectors的元组数为3，因此vectors定义为含有三个分量，InsertTuple3用来向数组中添加数据。最后创建的是元组数为2的数组，通过SetNumberOfTuples()分配内存。接着通过SetTuple2()添加数据；该方法使用的前提是内存已经分配，因此速度要明显快于Insert\_\_()方法。当将属性数据关联到点数据或者单元数据时，注意区别设置类型的方法（SetScalars()和SetVectors()）。注意点属性个数必须与数据结构中的点个数一致，单元属性与数据结构的单元个数一致。

类似的，采用如下方法访问属性数据

set scalars [[polyData GetPointData] GetScalars]

set vectors [[polyData GetCellData] GetVectors]

许多的filters需要专门的属性数据进行工作。例如，vtkElevationFilter依赖于相应的高度数据产生标量值。其他的filters只依赖于结构数据，并忽略传来的属性数据。还有一些filters需要结构数据和属性数据来工作，如vtkMarchingCubes。它利用输入的标量属性数据和结构数据来产生轮廓结构。其他类型的属性数据，例如向量，在计算轮廓时进行差值计算并输出出来。

另一个与属性数据相关的重要问题是，有些filters只出来一种类型的属性，忽略其他的类型。例如，当你想采用一个不能处理输入数据的属性数据类型的filter来处理数据时，或者你想直接将其从一种类型转换至另一个类型。有2个filters可以帮你实现：vtkPointDataToCellData和vtkCellDataToPointData。下面例子演示来怎样使用他们（代码来源VTK/Examples/DataManipulation/Tcl/pointToCellData.tcl）。

vtkUnstructuredGridReader reader

reader SetFileName "$VTK\_DATA\_ROOT/Data/blow.vtk"

reader SetScalarsName "thickness9"

reader SetVectorsName "displacement9"

vtkPointDataToCellData p2c

p2c SetInputConnection [reader GetOutputPort]

p2c PassPointDataOn

vtkWarpVector warp

warp SetInputConnection [p2c GetOutputPort]

vtkThreshold thresh

thresh SetInputConnection [warp GetOutputPort]

thresh ThresholdBetween 0.25 0.75

thresh SetAttributeModeToUseCellData

该例子演示了怎样转换属性数据类型，以及一个可以任意处理单元数据或者点数据的filter（vtkThreshold）。PassPointDataOn()方法设置vtkPointDataToCellData来创建单元数据，并将点数据输出。SetAttributeModeToUseCellData()设置vtkThreshold使用单元数据来进行处理。

点数据和单元数据之间的转换采用的是一个平均算法。将一个给定单元的所有点的数据求平均值即可将点的属性数据转换为单元数据；而单元数据转换为点数据则是通过计算所有用到该点的单元的数据的平均值。

　Color mapping 颜色映射

最常用的可视化技术可能是通过标量值或者颜色映射来对物体着色。着色技术的思想比较简单，将标量值映射到一个颜色查找表来获取颜色，然后在渲染时使用颜色来改变点或者单元的外观。在阅读本节前，确保理解了怎样控制Actor的颜色。（详见54页“Actor Color”一节）

VTK中颜色映射主要由用户生成或者数据文件中的标量数据和vtkMapper实例执行颜色映射使用的颜色查询表来控制。也可以使用任意的数据数组通过ColorByArrayComponent()方法来控制。如果没有指明，mapper会生成一个默认的颜色查询表，你也可以自己创建（下例取自VTK/Examples/Rendering/Tcl/

Rainbow.tcl见图5-1）。

vtkLookupTable lut

lut SetNumberOfColors 64

lut SetHueRange 0.0 0.667

lut Build

for {set i 0} {$i<16} {incr i 1} {

eval lut SetTableValue [expr $i\*16] $red 1

eval lut SetTableValue [expr $i\*16+1] $green 1

eval lut SetTableValue [expr $i\*16+2] $blue 1

eval lut SetTableValue [expr $i\*16+3] $black 1

}

vtkPolyDataMapper planeMapper

planeMapper SetLookupTable lut

　　planeMapper SetInputConnection [plane GetOutputPort]

　　planeMapper SetScalarRange 0.197813 0.710419

vtkActor planeActor

　　planeActor SetMapper planeMapper

如该例所示，操作查询表有两种方式。一是指定一个HSVA范围然后在HSVA空间中插值计算颜色表中颜色（实际由build()函数计算颜色）。第二种方法是在根据颜色表的位置人为指定颜色。注意颜色表中的颜色数目可以设置。本例中利用HSVA范围生成颜色表，然后利用SetTableValue()函数替换掉相应的颜色。

Mapper的SetScalarRange()函数控制标量值与颜色表的映射方式。大于最大值的所有标量值都被映射为最大值，小于最小值的所有标量值都映射为最小值。标量范围设置，可以通过在某一数据范围内映射更多的颜色来达到“扩展”的目的。

很多情况下标量数据就是颜色值，这样就不需要通过颜色查找表进行映射。映射器Mapper提供了多种方法来控制颜色映射。

\* SetColorModeToDefault()设置使用默认Mapper映射方法。默认方法直接将三个unsigned char类型标量数据作为颜色值而不需要进行映射；而其他的类型数据则是通过颜色查找表映射。

\* SetColorModeToMapScalars()通过颜色查找表来映射所有类型的标量数据。如果每个元组由多于1个分量组成，那么通过第0个分量来执行颜色映射。

vtkMapper另一个重要特征就是控制执行颜色渲染使用的数据，例如点或者单元数据，或者是数据数组。该功能可以由如下方法完成。需要注意的是，这些方法会产生非常不同的渲染结果：在渲染时点标量数据模式下，几何体表面所有的点数据通过插值后映射颜色，而单元标量数据模式下则将每个单元渲染为一个颜色。

\* SetScalarModeToDefault()触发默认mapper行为。Mapper默认采用点标量数据渲染；如果点标量数据不可用，则采用可用的单元数据进行渲染。

\* SetScalarModeToUsePointData()利用点标量数据进行颜色映射完成渲染。如果没有点标量数据可用，那么标量数据将对物体颜色没有任何影响。

\* SetScalarModeToUseCellData()利用单元标量数据进行颜色映射渲染物体。如果没有单元标量数据可用，那么标量数据将对物体颜色没有任何影响。

\* SetScalarModeToUsePointFieldData()利用点属性数据中的数据数组，而不是点标量数据和单元标量数据。该函数需要结合ColorByArrayComponent()函数来指定数据数组和作为标量数据的分量。

\* SetScalarModeToUseCellFieldData()利用单元属性数据中得场数据，而不是点或者单元标量数据。该方法也需要结合ColorByArrayComponent()函数来指定数据数组和作为标量数据的分量。

正常情况下Mapper默认行为都会正常工作，除非点和单元标量数据都可用。这时，你需要显示指明用点或者面标量数据来对物体着色。

Contouring 轮廓

另一个常见的可视化技术是轮廓生成。轮廓是指具有相同标量数据的线或者面。VTK中vtkContourFilter执行轮廓生成，如下面的Tcl例子所示。示例代码取自VTK/Examples/VislizationAlgorithms/Tcl/VisQuad.tcl，如图5-2所示。

#Create 5 surfaces in range specified

vtkContourFilter contours

contours SetInputConnection [ Sample \

GetOutputPort]

Contours GenerateValues 5 0.0 1.2

vtkPolyDataMapper contMapper

contMapper SetInputConnection [contours GetOutputPort]

contMapper SetScalarRange 0.0 1.2

vtkActor contActor

contActor SetMapper contMapper

有两种方式来指定轮廓值。最简单的方式是通过SetValue()方法来指定轮廓数和对应的轮廓值（可以指定多个值）。

Contours SetValue 0 0.5

示例代码中采用的是第二种方法，即GenerateValues()方法。通过该函数，你可以指定数据范围和该范围内要生成的轮廓个数（包含结束数据）。

注意在VTK中有多个对象来针对特定数据类型来执行轮廓生成。如vtkSynchronizedTemplates2D和vtkSynchronizedTemplates3D。如果利用vtkContourFilter的话，你不需要直接实例化这些类型的对象，该Filter会根据数据类型自动生成相应的轮廓生成函数。

Glyphing 字形化

字形化是一种利用字符或字形来表示数据的可视化技术（图5-3）。这些符号可以简单，也可以复杂。简单的如利用有向锥体来表示向量数据，复杂的有多元字形，如Chernoff faces（由数据来控制表情的人脸字形表示）。VTK中利用vtkGlyph3D类来产生可以放缩、着色和具有方向的字形。输入数据的每个点都会拷贝一个字形。字形本身则是通过该Filter的第二个输入函数接收（接收vtkPolyData类型数据）。下面代码说明了vtkGlyph3D的使用。（代码取自VTK/Examples/VisualizationAlgorithms/Tcl/spikeF.tcl）

　　vtkPolyDataReader fran

　　 fran SetFileName "$VTK\_DATA\_ROOT/Data/fran\_cut.vtk"

　　vtkPolyDataNormals normals

　　 normals SetInputConnection [fran GetOutputPort]

　　 normals FlipNormalsOn

　　vtkPolyDataMapper franMapper

　　 franMapper SetInputConnection [normals GetOutputPort]

　　vtkActor franActor

　　 franActor SetMapper franMapper

　　 eval [franActor GetProperty] SetColor 1.0 0.49 0.25

　　vtkMaskPoints ptMask

　　 ptMask SetInputConnection [normals GetOutputPort]

　　 ptMask SetOnRatio 10

　　 ptMask RandomModeOn

　　# In this case we are using a cone as a glyph. We transform the cone so

　　# its base is at 0,0,0. This is the point where glyph rotation occurs.

　　vtkConeSource cone

　　 cone SetResolution 6

　　vtkTransform transform

　　 transform Translate 0.5 0.0 0.0

　　vtkTransformPolyDataFilter transformF

　　 transformF SetInputConnection [cone GetOutputPort]

　　 transformF SetTransform transform

　　vtkGlyph3D glyph

　　 glyph SetInputConnection [ptMask GetOutputPort]

　　 glyph SetSourceConnection [transformF GetOutputPort]

　　 glyph SetVectorModeToUseNormal

　　 glyph SetScaleModeToScaleByVector

　　 glyph SetScaleFactor 0.004

　　vtkPolyDataMapper spikeMapper

　　 spikeMapper SetInputConnection [glyph GetOutputPort]

　　vtkActor spikeActor

　　 spikeActor SetMapper spikeMapper

　　 eval [spikeActor GetProperty] SetColor 0.0 0.79 0.34

这段代码演示了怎样用有向小锥体来表示曲面法向量。首先读入并显示一个数据集（由Cyberware激光数字化系统采集），然后vtkMastPoints用来对输入数据点进行降采样，采样结果（附带属性数据）作为vtkGlyph3D的输入。VtkConeSource用来生成字形实例。注意锥体采用vtkTransformPolyDataFilter平移来使原点位于(0, 0, 0) （因为vtkGlyph3D绕原点旋转）。

vtkGlyph3D对象glyph设置用点属性法向量作为方法向量。（SetVcetorModeToUseVector()设置使用向量数据来替代法向量）它利用向量模值来对锥体进行放缩。（当然也可以利用SetScaleModeToScaleByScalar()函数设置通过标量数据对字形进行放缩或者利用SetScaleModeToDataScalingOff()函数关闭放缩功能。）

当然也可以利用标量数据、向量数据或者放缩因子对字形进行着色。还可以创建一个字形表，采用标量或者向量数据来尽力索引。参考在线文档来获取更多信息。

Streamlines 流线

流线可以看做无重量粒子在向量场（如速度场）中的移动路径。流线可以表达向量场的结构。通常可以创建多个流线来探索向量场中的感兴趣特征。如图5-4。流线可以通过数值积分来计算，因此只能近似的模拟真实的流线。

创建流线需要指定起始点，方向（沿着或者反着流向），以及其他的控制前进的参数。下面代码说明了如何创建一条流线。该流线由一个管道表示，管道半径正比于速度模值的倒数。当流比较小时，管道会比较粗；反之亦然。Tcl代码取自VTK/Exmaples/VisualizationAlgorithms/Tcl/OfficeTube.tcl。

vtkStructuredGridReader reader

reader SetFileName "$VTK\_DATA\_ROOT/Data/office.binary.vtk"

reader Update;#force a read to occur

vtkRungeKutta4 integ

vtkStreamTracer streamer

streamer SetInputConnection [reader GetOutputPort]

streamer SetStartPosition 0.1 2.1 0.5

streamer SetMaximumPropagation 500

streamer SetMaximumPropagationUnitToTimeUnit

streamer SetInitialIntegrationStep 0.05

streamer SetInitialIntegrationStepUnitToCellLengthUnit

streamer SetIntegrationDirectionToBoth

　　streamer SetIntegrator integ

vtkTubeFilter streamTube

streamTube SetInputConnection [streamer GetOutputPort]

streamTube SetInputArrayToProcess 1 0 0 vtkDataObject::FIELD\_ASSOCIATION\_POINTS vectors

streamTube SetRadius 0.02

streamTube SetNumberOfSides 12

streamTube SetVaryRadiusToVaryRadiusByVector

vtkPolyDataMapper mapStreamTube

mapStreamTube SetInputConnection [streamTube GetOutputPort]

eval mapStreamTube SetScalarRange \

[[[[reader GetOutput] GetPointData] GetScalars] GetRange]

vtkActor streamTubeActor

streamTubeActor SetMapper mapStreamTube

　　[streamTubeActor GetProperty] BackfaceCullingOn

在该例中我们设置起始点为(0.1，2.1, 0.5)。也可以通过指定单元id，单元子id和参数坐标来指定起始坐标。MaximumPropagation变量用来控制流线的最大长度（度量单位由MaximumPropagationUnit变量指定）。如果需要更高的精度（代价是更多的计算时间），设置InitialIntegrationsetp变量为一个更小的值。（这里该参数指定为单元长度，也可以选择时间或者距离）另外，精度也可以通过选择一个vtkInitialValueProblemSolver的子类如vtkRungeKutta2或者vtkRungeKutta45（可以设置自适应的步长）来控制。默认情况下，流线追踪类利用vtkRungeKutta2来执行数值积分运算。积分方向可以由以下三个函数控制。

\* SetIntegrationDirectionToForward()

\* SetIntegrationDirectionToBackward()

\* SetIntergrationDirectionToBoth()

由于线通常难以观察，因为我们采用一个管道filter来表示。管道filter设置为半径正比于速度模值的倒数，通过函数SetVaryRadiusToVaryRadiusByVector()开启该功能。也可以通过SetVaryRadiusToVaryRadiusByScalar()设置根据标量控制半径，或者取消半径可变（SetVaryRadiusToVaryRadiusOff()）。注意需要通知管道filter利用哪个数组来控制半径，这里“vectors”数组通过SetInputArrayToProcess()函数通知管道filter。

有时候我们需要产生同步地产生许多流线。其中一个方法是利用SetSourceConnection()方法来指定一个vtkDataSet的实例，利用这个实例的点来追踪流线。下面是该应用的一个例子，代码取自VTK/Examples/VisualizationAlgorithms/Tcl/OffcicesTubes.tcl。

vtkPointSource seeds

seeds SetRadius 0.15

eval seeds SetCenter 0.1 2.1 0.5

seeds SetNumberOfPoints 6

vtkRungeKutta4 integ

vtkStreamTracer streamer

streamer SetInputConnection [reader GetOutputPort]

streamer SetSourceConnection [seeds GetOutputPort]

streamer SetMaximumPropagation 500

streamer SetMaximumPropagationUnitToTimeUnit

streamer SetInitialIntegrationStep 0.05

streamer SetInitialIntegrationStepUnitToCellLengthUnit

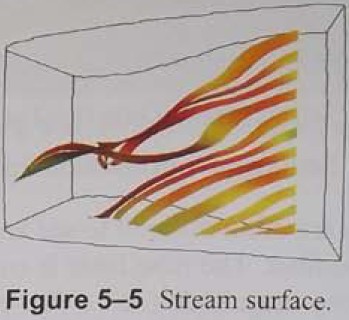
streamer SetIntegrationDirectionToBoth

streamer SetIntegrator integ

注意该例中采用vtkPointSource对象来创建球状点云作为streamer的输入。对于输入的每一个点都会计算一条流线。

Stream Surfaces 流面

高级用户可能想利用VTK的流面计算功能。流面计算分为两步，首先用一个有序点集生成一个流线序列，然后再由vtkRuledSurfaceFilter来创建流面。vtkRuledSurfaceFilter假设每条流线是有序排列的，而且每条流线与左右相邻流线在指定距离内，因此输入点集保持有序十分重要，否则计算结果将会非常糟糕。下面代码演示了如何创建一个流面（代码取自VTK/Examples/VisualizationAlgorithms/Tch/streamSurface.tcl,结果如图5-5）。

vtkLineSource rake

rake SetPoint1 15 -5 32

rake SetPoint2 15 5 32

rake SetResolution 21

vtkPolyDataMapper rakeMapper

rakeMapper SetInputConnection [rake GetOutputPort]

vtkActor rakeActor

rakeActor SetMapper rakeMapper

vtkRungeKutta4 integ

vtkStreamTracer sl

sl SetInputConnection [pl3d GetOutputPort]

sl SetSourceConnection [rake GetOutputPort]

sl SetIntegrator integ

sl SetMaximumPropagation 0.1

sl SetMaximumPropagationUnitToTimeUnit

sl SetInitialIntegrationStep 0.1

sl SetInitialIntegrationStepUnitToCellLengthUnit

sl SetIntegrationDirectionToBackward

vtkRuledSurfaceFilter scalarSurface

scalarSurface SetInputConnection [sl GetOutputPort]

scalarSurface SetOffset 0

scalarSurface SetOnRatio 2

scalarSurface PassLinesOn

scalarSurface SetRuledModeToPointWalk

scalarSurface SetDistanceFactor 30

vtkPolyDataMapper mapper

mapper SetInputConnection [scalarSurface GetOutputPort]

eval mapper SetScalarRange [[pl3d GetOutput] GetScalarRange]

vtkActor actor

actor SetMapper mapper

vtkRuledSurfaceFilter一个优点是当输入多个流线时可以关闭带功能，这样有助于理解曲面的结构。

Cutting 切割

VTK中切割或者切面化数据集意味着在数据集中利用任意类型的隐式函数来创建交叉区域。例如我们可以采用一个平面来创建一个平面切面对数据集进行切面显示。切面进行切割时对数据进行插值，然后采用任何一个标准可视化技术来显示。切割结果通常是vtkPolyData类型（n维对象的切割结果是一个n-1维几何体。例如，切割一个四面体将产生一个三角形或者四边形）。

下面Tcl实例中演示了一个燃烧室被平面切割的结果，如图5-6。代码取自VTK/Graphics/Tesing/probe.tcl。

vtkPlane plane

eval plane SetOrigin [[pl3d GetOutput] GetCenter]

plane SetNormal -0.287 0 0.9579

vtkCutter planeCut

planeCut SetInputConnection [pl3d GetOutputPort]

planeCut SetCutFunction plane

vtkProbeFilter probe

probe SetInputConnection [planeCut GetOutputPort]

probe SetSourceConnection [pl3d GetOutputPort]

vtkDataSetMapper cutMapper

cutMapper SetInputConnection [probe GetOutputPort]

eval cutMapper SetScalarRange \

[[[[pl3d GetOutput] GetPointData] GetScalars] GetRange]

vtkActor cutActor

cutActor SetMapper cutMapper

vtkCutter需要指定完成切割的隐式函数。另外，你可能希望指定一个或者多个切割值，可以通过SetValue()或者GenerateValues()函数实现。这些数值指定了执行切割的隐式函数值。（默认情况下切割值为0 ，即切面精确地位于隐式函数曲面，小于或者大于0值的曲面则位于该隐式曲面的下面和上面。切割值可以看做是到隐式曲面的“距离”。）

Merging Data 合并数据

到目前为止我们已经了解了简单的线性可视化管线。然后，管线还可以存在分支，合并甚至是循环情况。接下来我们介绍两个filters，它们可以利用其他的数据集来构建新的数据集。现在从vtkMergeFilter开始。

vtkMergeFilter将多个数据集中的数据片段合并为一个新的数据集。例如，可以将一个数据集的结构（拓扑和几何），第二个数据集的标量数据，第三个数据集的向量数据合并为一个数据集。这里是该应用的一个实例。（代码取自VTK/Examples/VisualizationAlgorithms/Tcl/imageWarp.tcl）。（主要关注vtkMergeFilter，其他的不熟悉的可以先忽略。在第106页我们将更加详细的进行描述。）

vtkBMPReader reader

reader SetFileName $VTK\_DATA\_ROOT/Data/masonry.bmp

vtkImageLuminance luminance

luminance SetInputConnection [reader GetOutputPort]

vtkImageDataGeometryFilter geometry

geometry SetInputConnection [luminance GetOutputPort]

vtkWarpScalar warp

warp SetInputConnection [geometry GetOutputPort]

warp SetScaleFactor -0.1

vtkMergeFilter merge

merge SetGeometryConnection [warp GetOutputPort]

merge SetScalarsConnection [reader GetOutputPort]

vtkDataSetMapper mapper

mapper SetInputConnection [merge GetOutputPort]

mapper SetScalarRange 0 255

mapper ImmediateModeRenderingOff

vtkActor actor

actor SetMapper mapper

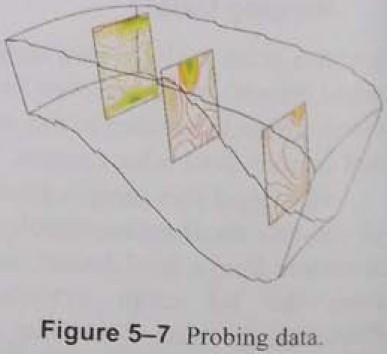
这里所做的是将vtkWarpScalar（输出是vtkPolyData类型）的输出与vtkBMPReader标量数据合并到一起。管线先分后合，因为几何结构需要利用标量数据来单独处理。

当合并数据时，数组中的元组数必须与点的个数一致，单元数据也是一样。

Appending Data追加数据

类似vtkMergeFilter，vtkAppendFilter以及vtkAppendPolyData通过追加数据集来产生新数据集。追加Filter接收一系列输入，这些输入的类型必须一致。在追加操作中，只有那些共同的数据数据才会合并在一块。下一节中演示了一个比较好的实例。

Probing 探测

探测是利用其它数据集对数据集进行采样的过程。在VTK中，可以利用任意的数据集来作为探测几何，其点属性则由其它数据集映射而来。例如下例中创建了三个平面作为探测几何来对一个结构网格数据集进行采样。然后这些平面通过vtkContourFilter来计算等值线。代码取自VTK/Examples/VisualizationAlgorithms/Tcl/probeComb.tcl。

vtkPLOT3DReader pl3d

pl3d SetXYZFileName "$VTK\_DATA\_ROOT/Data/combxyz.bin"

pl3d SetQFileName "$VTK\_DATA\_ROOT/Data/combq.bin"

pl3d SetScalarFunctionNumber 100

pl3d SetVectorFunctionNumber 202

pl3d Update

vtkPlaneSource plane

plane SetResolution 50 50

vtkTransform transP1

transP1 Translate 3.7 0.0 28.37

transP1 Scale 5 5 5

transP1 RotateY 90

vtkTransformPolyDataFilter tpd1

tpd1 SetInputConnection [plane GetOutputPort]

tpd1 SetTransform transP1

vtkOutlineFilter outTpd1

outTpd1 SetInputConnection [tpd1 GetOutputPort]

vtkPolyDataMapper mapTpd1

mapTpd1 SetInputConnection [outTpd1 GetOutputPort]

vtkActor tpd1Actor

tpd1Actor SetMapper mapTpd1

[tpd1Actor GetProperty] SetColor 0 0 0

vtkTransform transP2

transP2 Translate 9.2 0.0 31.20

transP2 Scale 5 5 5

transP2 RotateY 90

vtkTransformPolyDataFilter tpd2

tpd2 SetInputConnection [plane GetOutputPort]

tpd2 SetTransform transP2

vtkOutlineFilter outTpd2

outTpd2 SetInputConnection [tpd2 GetOutputPort]

vtkPolyDataMapper mapTpd2

mapTpd2 SetInputConnection [outTpd2 GetOutputPort]

vtkActor tpd2Actor

tpd2Actor SetMapper mapTpd2

[tpd2Actor GetProperty] SetColor 0 0 0

vtkTransform transP3

transP3 Translate 13.27 0.0 33.30

transP3 Scale 5 5 5

transP3 RotateY 90

vtkTransformPolyDataFilter tpd3

tpd3 SetInputConnection [plane GetOutputPort]

tpd3 SetTransform transP3

vtkOutlineFilter outTpd3

outTpd3 SetInputConnection [tpd3 GetOutputPort]

vtkPolyDataMapper mapTpd3

mapTpd3 SetInputConnection [outTpd3 GetOutputPort]

vtkActor tpd3Actor

tpd3Actor SetMapper mapTpd3

[tpd3Actor GetProperty] SetColor 0 0 0

vtkAppendPolyData appendF

appendF AddInputConnection [tpd1 GetOutputPort]

appendF AddInputConnection [tpd2 GetOutputPort]

appendF AddInputConnection [tpd3 GetOutputPort]

vtkProbeFilter probe

probe SetInputConnection [appendF GetOutputPort]

probe SetSourceConnection [pl3d GetOutputPort]

vtkContourFilter contour

contour SetInputConnection [probe GetOutputPort]

eval contour GenerateValues 50 [[pl3d GetOutput] GetScalarRange]

vtkPolyDataMapper contourMapper

contourMapper SetInputConnection [contour GetOutputPort]

eval contourMapper SetScalarRange [[pl3d GetOutput] GetScalarRange]

vtkActor planeActor

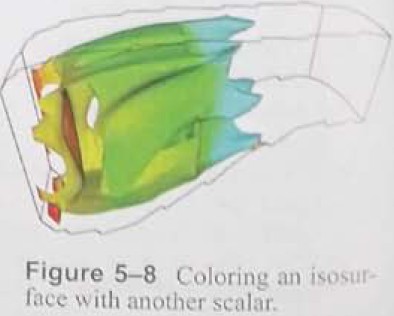
planeActor SetMapper contourMapper

注意通过SetInputConnection()方法来设置探测器，而待探测数据集是通过SetSourceConnection()函数设置。

探测的另外一个应用是重采样数据。例如，如果你有一个无结构网格数据，而你想通过专门可视化vtkImageData的工具来显示（例如体绘制，139页）。那么可以采用vtkProbeFilter对无结构网格数据采用为一个体数据，然后再可视化。同样也可以采用直线来探测数据，并将结果绘制成X-Y曲线。

最后值得注意的是，切割和探测都能得到类似的结果，除了在分辨率上有所差别外。类似于98页的切割实例，vtkProbeFilter可以利用vtkPlaneSource来产生一个平面，而该平面的属性数据来自于结构网格数据。然而，切割产生的曲面分辨率要依赖于输入数据。而探测产生的曲面分辨率则独立于输入数据。因此在探测数据时，需要特别注意低采用或者过采样。低采样会导致显示错误，过采样则会占用过多的计算时间。

Color An Isosurface With Another Scalar 用其他标量着色等值面

计算等值面并用另一个标量进行着色是一个常见的可视化操作。可能你会想到用探测器，但是如果你等值面如果包含你想用来着色的数据时，会有一个更加有效的方法。因为vtkContourFilter（实际用来产生等值面）在计算过程中会对所有数据进行插值。插值后的数据在映射过程中用来进行着色。下面例子取自VTK/Examples/VisualizationAlgorithms/Tcl/ColorIsosurface.tcl。

vtkPLOT3DReader pl3d

pl3d SetXYZFileName "$VTK\_DATA\_ROOT/Data/combxyz.bin"

pl3d SetQFileName "$VTK\_DATA\_ROOT/Data/combq.bin"

pl3d SetScalarFunctionNumber 100

pl3d SetVectorFunctionNumber 202

pl3d AddFunction 153

pl3d Update

vtkContourFilter iso

iso SetInputConnection [pl3d GetOutputPort]

iso SetValue 0.24

vtkPolyDataNormals normals

normals SetInputConnection [iso GetOutputPort]

normals SetFeatureAngle 45

vtkPolyDataMapper isoMapper

isoMapper SetInputConnection [normals GetOutputPort]

isoMapper ScalarVisibilityOn

isoMapper SetScalarRange 0 1500

isoMapper SetScalarModeToUsePointFieldData

isoMapper ColorByArrayComponent "VelocityMagnitude" 0

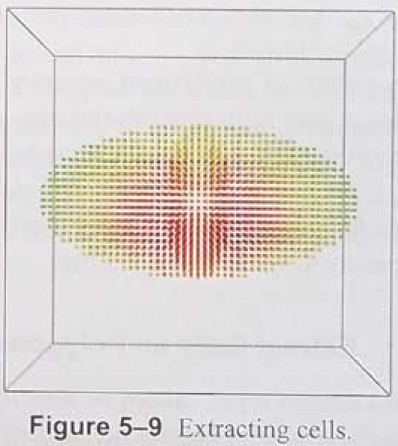
vtkLODActor isoActor

isoActor SetMapper isoMapper

isoActor SetNumberOfCloudPoints 1000

首先通过vtkPLOT3DReader读取数据集。这里我们添加一个要读的函数（函数号153），函数名字是“Velocity Magnitude”。计算等值面时，也会对所有输入数据包括速度场数据进行插值。然后我们调用SetScalarModeToUsePointFieldData()函数来利用速度模值来对等值面着色，ColorByArrayComponent()方法用来指定着色所用的数据数组。

Extract Subset of Cells 提取子单元

可视化数据的数据量一般都非常大，处理这样的数据往往要耗费大量的时间和内存。因此，提取部分数据功能显示非常重要。多数情况下，只有部分数据包含有意义的信息，或者在不影响精度的条件下对数据进行消减。

VTK中提供了许多工具来提取部分数据或者对数据降采样。我们已经了解到vtkProbeFilter可以用来进行降采样（见100页“Probing”）。其他工具包含降采样类，还有一些工具能够实现从空间区域中提取单元（降采样工具针对于特定数据，详见105页“降采样图像数据”，和113页“降采样规则网格数据”）。本节中我们主要讲述怎样在空间区域中提取数据片段。

vtkExtractGeometryl类提取数据集中位于隐式函数曲面vtkImplicitFucntion内部或者外部的所有单元(注意，隐式函数可以是多个隐式函数的二值组合)。下面代码创建了两个椭球的二值组合用来提取区域。vtkShrinkFilter用来对单元进行收缩以便能观察被提取的数据。（代码取自VTK/Examples/VisualizationAlgorithms/Tcl/ExtractionGeometry.tcl.）

vtkQuadric quadric

quadric SetCoefficients .5 1 .2 0 .1 0 0 .2 0 0

vtkSampleFunction sample

sample SetSampleDimensions 50 50 50

sample SetImplicitFunction quadric

sample ComputeNormalsOff

vtkTransform trans

trans Scale 1 .5 .333

vtkSphere sphere

sphere SetRadius 0.25

sphere SetTransform trans

vtkTransform trans2

trans2 Scale .25 .5 1.0

vtkSphere sphere2

sphere2 SetRadius 0.25

sphere2 SetTransform trans2

vtkImplicitBoolean union

union AddFunction sphere

union AddFunction sphere2

union SetOperationType 0;#union

vtkExtractGeometry extract

extract SetInputConnection [sample GetOutputPort]

extract SetImplicitFunction union

vtkShrinkFilter shrink

shrink SetInputConnection [extract GetOutputPort]

shrink SetShrinkFactor 0.5

vtkDataSetMapper dataMapper

dataMapper SetInputConnection [shrink GetOutputPort]

vtkActor dataActor

dataActor SetMapper dataMapper

vtkExtractGeometry的输出通常是vtkUnstructuredGrid类型。因为在提取过程中，数据集的拓扑结构往往被破坏，因此必须采用最普遍的数据格式来表示输出。

注意，隐式函数可以通过分配一个vtkTransform来进行变换。如果指定了变换，vtkTransform就被用来改变隐式函数的取值。你可能希望对该功能进行测试。

Extract Cells as Polygonal Data 提取多边形单元数据

大部分数据类型不能直接被图形硬件或者图形库渲染。渲染系统中只有多边形数据（vtkPolyData）被广泛的支持。规则数据集，特别是图像以及体数据也可以被图形系统支持。而其他的数据则需要特别的处理才能被渲染。在VTK中，渲染非多边形数据的一个方法是将其转换为多边形数据。vtkGeometryFilter可实现该功能。

vtkGEometryFilter接收任意vtkDataSet类型数据并输出vtkPolyData数据。其执行转换时，遵循如下规则。所有二维及以下拓扑单元（如多边形，直线，顶点）直接传递至输出。位于数据集边界的三维单元面会被传递至输出结果中。（如果一个面只属于一个单元，那么这个面位于边界。）

vtkGeometryFilter常被用来进行数据格式转换。下面代码中利用vtkGeometryFilter来将二维不规则网格转换为多边形数据，这些多边形数据接下来会作为其他接收vtkPolyData的Filter所接收。代码取自VTK/Examples/DataManipulation/Tcl/pointToCellData.tcl。这里vtkConnectivityFilter提取vtkUnstructedGrid数据然后通过vtkGeometryFilter将其转换为多边形数据。

vtkConnectivityFilter connect2

connect2 SetInputConnection [thresh GetOutputPort]

vtkGeometryFilter parison

parison SetInputConnection [connect2 GetOutputPort]

vtkPolyDataNormals normals2

normals2 SetInputConnection [parison GetOutputPort]

normals2 SetFeatureAngle 60

vtkLookupTable lut

lut SetHueRange 0.0 0.66667

vtkPolyDataMapper parisonMapper

parisonMapper SetInputConnection [normals2 GetOutputPort]

parisonMapper SetLookupTable lut

parisonMapper SetScalarRange 0.12 1.0

vtkActor parisonActor

parisonActor SetMapper parisonMapper

实际上vtkDataSetMapper内部使用vtkGeometryFilter来将任意数据类型转换为多边形数据。（该Filter可以直接将输入的vtkPolyData传递到输出。）

另外vtkGeometryFilter提供了一系列函数来根据点id集合，单元ids集合或者判断是否位于一个特定的矩形空间区域中来提取数据单元。利用点或者面id集合提取数据片段时，使用的函数有PointClippingOn(),SetPointMinimum(),SetPointMaximum()和CellClippingOn(),SetCellMinimum(),SetCellMaximum()。最小值和最大值指定了提取的id 的范围。同样还可以指定一个空间矩形区域来限制提取的范围。用ExtentClippingOn()和SetExtent()来开始空间切割和指定范围。Extent包含了六个参数来定义一个包围盒-（,）。可以使用这三种方式的任意组合来提取数据。在调试数据时会非常有用，或者当你只想观察部分数据时。

**5.2 多边形数据可视化**

多边形数据（vtkPolyData）是一种重要的可视化数据类型。因为它是图像硬件或者渲染引擎的几何接口。其他的数据必须转换为多边形数据才能被渲染，除了vtkImageData外。VtkImageData使用的时特别图像或者体绘制技术。可以参考104页“提取多边形单元数据”来了解怎样转换数据类型。

多边形数据主要由点和点集，直线和直线组合，三角形，四边形，多边形和三角形带组合而成。大部分接收vtkPolyData的filters都会处理这些数据组合。当然也有一些filters如vtkDecimatePro和vtkTubeFilter会只处理部分数据类型（三角网格和直线）。

Manually Create vtkPolyData手动创建vtkPolyData数据

多边形数据可以通过多种方式构造。通常需要创建一个vtkPoints来存储点集，然后创建4个vtkCellArrays来分别表示点，线，多边形和三角带连接。下面例子取自VTK/Examples/DataManipulation/Tcl/CreateStrip.tcl。它创建了一个具有一个三角形带的vtkPolyData数据。

vtkPoints points

points InsertPoint 0 0.0 0.0 0.0

points InsertPoint 1 0.0 1.0 0.0

points InsertPoint 2 1.0 0.0 0.0

points InsertPoint 3 1.0 1.0 0.0

points InsertPoint 4 2.0 0.0 0.0

points InsertPoint 5 2.0 1.0 0.0

points InsertPoint 6 3.0 0.0 0.0

points InsertPoint 7 3.0 1.0 0.0

vtkCellArray strips

strips InsertNextCell 8;#number of points

strips InsertCellPoint 0

strips InsertCellPoint 1

strips InsertCellPoint 2

strips InsertCellPoint 3

strips InsertCellPoint 4

strips InsertCellPoint 5

strips InsertCellPoint 6

strips InsertCellPoint 7

vtkPolyData profile

profile SetPoints points

profile SetStrips strips

vtkPolyDataMapper map

map SetInput profile

vtkActor strip

strip SetMapper map

[strip GetProperty] SetColor 0.3800 0.7000 0.1600

这里有另外一个C++例子来演示怎样创建一个立方体（VTK/Examples/DataManipulation/Cxx/Cube.cxx）。这次我们创建6个四边形和每个顶点对应的标量值。

int i;

static float x[8][3]={{0,0,0}, {1,0,0}, {1,1,0}, {0,1,0},

{0,0,1}, {1,0,1}, {1,1,1}, {0,1,1}};

static vtkIdType pts[6][4]={{0,1,2,3}, {4,5,6,7}, {0,1,5,4},

{1,2,6,5}, {2,3,7,6}, {3,0,4,7}};

vtkPolyData \*cube = vtkPolyData::New();

vtkPoints \*points = vtkPoints::New();

vtkCellArray \*polys = vtkCellArray::New();

vtkFloatArray \*scalars = vtkFloatArray::New();

// Load the point, cell, and data attributes.

for (i=0; i<8; i++) points->InsertPoint(i,x[i]);

for (i=0; i<6; i++) polys->InsertNextCell(4,pts[i]);

for (i=0; i<8; i++) scalars->InsertTuple1(i,i);

// We now assign the pieces to the vtkPolyData.

cube->SetPoints(points);

points->Delete();

cube->SetPolys(polys);

polys->Delete();

cube->GetPointData()->SetScalars(scalars);

scalars->Delete();

vtkPolyData可以由点，线，多边形和三角形带的任意组合构建。同时，vtkPolyData支持一系列扩展的操作来对数据结构进行编辑和修改。参考345页“多边形数据”。

Generate Surface Normals计算表面法向量

当你渲染多边形网格时，你可能会发现渲染图像中清晰的显示了网格面片（图5-10）。采用Gouraud着色技术可以提高渲染结果（见53页“Actor属性”）。然而Gouraud着色技术依赖于网格每个顶点的法向量。因此需要利用vtkPolyDataNoramals filter来计算网格顶点法向量。217页“Extrusion”、94页“Glyphing”和102页“Color An Isosurface With Another Scalar”都是用了该类计算法向量。



该filter有两个重要的参数，splitting和FeatureAngle。如果splitting开启，那么特征边被分裂（如果一条边任意一端的顶点法向与该边所成夹角大于或者等于特征角FeatureAngle的话，那么该边定义为特征边）。也就是沿着该条边复制顶点，网格在特征边的任意一边被分割。这样虽然产生了新的顶点，但是能够清晰的渲染尖锐角处。另外一个重要的参数是FlipNormals。调用函数FlipNormalsOn()将使法向量反向（同时多边形连接表也会反向）。



Decimation抽取

多边形数据，特别是三角形网格是很常用的图形数据格式。Filters如vtkContourFilter产生的结果就是三角形网格。但是这些三角形网格往往会比较大，因此在一些交互应用中难以快速的处理。抽取技术常被用来解决这个问题。抽取也被称作多边形消减，网格简化或者多分辨率建模。其在保持近似原始网格条件下对三角网格中的三角形数量进行消减。

VTK支持三种消减方法，vtkDecimatePro,vtkQuadricClustering和vtkQuadricDecimation。虽然每一种都有各自的优缺点，但在应用上基本都一致。

* vtkDecimatePro相对来说速度比较快，而且在消减过程中可以修改拓扑结构。它采用边塌陷方法来去除顶点和三角形。它的误差度量是基于到平面或者边的距离。该方法的一个优点是可以完成任意程度的消减，因为该算法一开始就将网格分裂为小的碎片来完成消减目的（如果允许拓扑修改的话）。
* vtkQuadricDecimation采用的Siggraph97论文“Surface Simplification Using Quadric Error Metrics”中提出的二次误差度量。它采用边塌陷方法来剔除顶点和三角面片。二次误差度量通常被认为是比较好的误差度量。
* vtkQuadricClustering是最快的方法。它的思想基于Siggraph2000中的论文“Out-of-Core Simplification of Large Polygonal Models”。它能够快速的消减大网格模型，并且支持网格片段消减（利用StartAppend(), Append()和EndAppend()方法）。这样可以避免读取整个模型到内存中。对于大网格模型，该方法有较好的效果；但是当网格变小时，三角化过程效果不是很好（需要结合其他的算法会有较理想的效果）。

下面是一个vtkDecimatePro 的一个实例。代码改自VTK/Examples/VisualizationAlgorithms/Tcl/deciFram.tcl（如图5-11）。

vtkDecimatePro deci

deci SetInputConnection [fran GetOutputPort]

deci SetTargetReduction 0.9

deci PreserveTopologyOn

vtkPolyDataNormals normals

normals SetInputConnection [deci GetOutputPort]

normals FlipNormalsOn

vtkPolyDataMapper franMapper

franMapper SetInputConnection [normals GetOutputPort]

vtkActor franActor

franActor SetMapper franMapper

eval [franActor GetProperty] SetColor 1.0 0.49 0.25

vtkDecimatePro有两个重要的参数：TargetReduction和PreserveTopology。TargetReduction是用户的消减量（例如0.9意味着我们希望减少网格中90%的面片。）根据你是否允许修改拓扑结构（PreserveTopologyOn/Off()）来得到相应的消减结果。

最后注意的是消减filter的输入为三角形数据。如果你的是多边形数据，那么需要利用vtkTriangleFilter 来进行转换。

Smooth Mesh网格平滑

多边形网格往往由于噪声或者过于粗糙，从而影响渲染的结果。例如，低分辨率数据的等值面会有走样的效果。数据平滑是处理这种问题的方法之一。平滑是通过改变顶点位置来减少曲面噪声数据的的过程。

VTK中提供了两种平滑对象：vtkSmoothPolyDataFilter和vtkWindowSincPolyDataFilter。两者中，vtkWindowSincPolyDataFilter处理效果最好，而且处理速度也较快。下面例子（代码来自VTK/Examples/VisualizationAlgorithms/Tcl/smoothFran.tcl）演示了怎样利用该平滑filter。该例与上节例子内容相同，只是多了一个平滑filter处理。图5-12显示了简化模型的平滑效果。

vtkDecimatePro deci

deci SetInputConnection [fran GetOutputPort]

deci SetTargetReduction 0.9

deci PreserveTopologyOn

vtkSmoothPolyDataFilter smoother

smoother SetInputConnection [deci GetOutputPort]

smoother SetNumberOfIterations 50

vtkPolyDataNormals normals

normals SetInputConnection [smoother GetOutputPort]

normals FlipNormalsOn

vtkPolyDataMapper franMapper

franMapper SetInputConnection [normals GetOutputPort]

vtkActor franActor

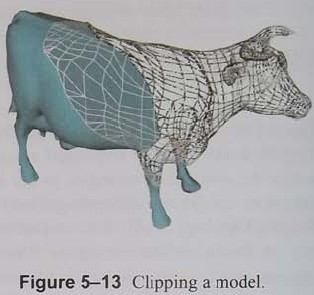
franActor SetMapper franMapper

eval [franActor GetProperty] SetColor 1.0 0.49 0.25



两种平滑Filter的使用方法类似。他们提供了一些函数来控制特征边或者边界处的平滑效果。可查阅在线文档或者 .h头文件了解相关信息。

Clip Data数据裁剪

裁剪类似于切割（98页“切割”）利用一个隐式函数来定义裁剪面。裁剪将多边形模型分解为片段，如图5-13所示。裁剪在裁剪面的任意一边将多边形几何体分解为单独的部分。如切割一样，裁剪可以设置一个裁剪值来定义隐式裁剪函数的值。

下例中利用一个平面来裁剪一个牛的三维多边形模型。切割值用来定义切割平面沿着法向量移动距离的大小，这样便可以在不同的位置对模型进行裁剪。代码取自（VTK/Examples/VisualizationAlgorithms/Tcl/ClipCow.tcl）。

vtkBYUReader cow

cow SetGeometryFileName "$VTK\_DATA\_ROOT/Data/Viewpoint/cow.g"

vtkPolyDataNormals cowNormals

cowNormals SetInputConnection [cow GetOutputPort]

vtkPlane plane

plane SetOrigin 0.25 0 0

plane SetNormal -1 -1 0

vtkClipPolyData clipper

clipper SetInputConnection [cowNormals GetOutputPort]

clipper SetClipFunction plane

clipper GenerateClipScalarsOn

clipper GenerateClippedOutputOn

clipper SetValue 0.5

vtkPolyDataMapper clipMapper

clipMapper SetInputConnection [clipper GetOutputPort]

clipMapper ScalarVisibilityOff

vtkProperty backProp

eval backProp SetDiffuseColor $tomato

vtkActor clipActor

clipActor SetMapper clipMapper

eval [clipActor GetProperty] SetColor $peacock

clipActor SetBackfaceProperty backProp

vtkPolyDataMapper restMapper

restMapper SetInputConnection [clipper GetClippedOutputPort]

restMapper ScalarVisibilityOff

vtkActor restActor

restActor SetMapper restMapper

[restActor GetProperty] SetRepresentationToWireframe

GenerateClippedOutputOn()函数计算裁剪过程中被裁剪掉得数据。在该例中这部分数据采用线框方式显示。如果通过SetValue()函数改变了裁剪值，那么裁剪面将移动至初始裁剪面前或者后并且平行与初始裁剪面的位置。（同样改变vtkPlane的定义也可实现同样的效果。）

Generate Texture Coordinates计算纹理坐标

常用的计算纹理坐标的filters是vtkTextureMapToPlane，vtkTextureMapToCylinder和vtkTextureMapToSphere。他们分别基于平面，圆柱和球面坐标系计算纹理坐标。另外，vtkTransformTextureCoordinates通过平移和缩放纹理坐标在曲面上定义纹理映射。下例中演示了利用vtkTextureMapToCylinder来为一个由vtkDelaunay3D计算产生的不规则网格数据计算纹理坐标（参考218页“Delaunay Triangulation”）。完整代码可以参考VTK/Examples/VisualizationAlgorithms/Tcl/GenerateTextureCoords.tcl。

vtkPointSource sphere

sphere SetNumberOfPoints 25

vtkDelaunay3D del

del SetInputConnection [sphere GetOutputPort]

del SetTolerance 0.01

vtkTextureMapToCylinder tmapper

tmapper SetInputConnection [del GetOutputPort]

tmapper PreventSeamOn

vtkTransformTextureCoords xform

xform SetInputConnection [tmapper GetOutputPort]

xform SetScale 4 4 1

vtkDataSetMapper mapper

mapper SetInputConnection [xform GetOutputPort]

vtkBMPReader bmpReader

bmpReader SetFileName "$VTK\_DATA\_ROOT/Data/masonry.bmp"

vtkTexture atext

atext SetInputConnection [bmpReader GetOutputPort]

atext InterpolateOn

vtkActor triangulation

triangulation SetMapper mapper

triangulation SetTexture atext

该例中首先对一个单位球上的随机点进行三角剖分，并为三角网格计算纹理坐标。接着在纹理坐标i-j方向上对纹理坐标进行缩放以便使纹理能够重复。最后读入纹理图像并指定到Actor上。

（注意，vtkDataSetMapper可以接收任意数据类型的输入。其内部vtkGeometryfilter对象将数据转换为多边形数据然后再传入到渲染引擎中。参考104页“提取多边形单元数据”）

如果需要深入学习纹理坐标，可以运行VTK/Examples/VisualizationAlgorithms/Tcl/TransformTexturecoords.tcl实例。该例GUI可以选择多边形模型，纹理图像，纹理计算方法和纹理变换方法。

**5.3 规则网格可视化**

规则网格在拓扑结构上规则，而几何上是不规则的，如图3-2所示。规则网格常被用来进行数据分析（如流体动力学计算）。规则网格vtkStructuredGrid数据集主要有六面体（vtkHexahedral）单元或者是四边形（vtkQuad）单元组成。

Manually Create vtkStructedGrid手动创建vtkStructedGrid

创建规则网格需要指定网格的维数（定义拓扑结构）和定义x-y-z坐标的vtkPoints对象（定义几何结构）。下面代码改自VTK/Examples/DataManiulation/Cxx/SGrid.cxx。

vtkPoints points

points InsertPoint 0 0.0 0.0 0.0

…etc…

vtkStructedGrid sgrid

sgrid SetDimensions 13 11 11

sgrid SetPoints points

主要vtkPoints对象中点的个数要与网格i，j和k三个方向的维数乘积保持一致。

Extract Computational Plane 提取计算平面

大多数情况下规则网格由接收vtkDataSet类型输入的filter来处理（见89页“可视化技术”）。而直接接收vtkStructuredGrid类型输入的其中一个filter是vtkStructuredGridGeometryFilter。该Filter根据其Extent变量值来提取网格中的局部点、线段或者平面（Extent是一个描述区域的六维向量（,））。

下例中我们读取了一个规则网格数据，提取其中的三个平面，并利用相关的向量数据对平面进行Warp处理（代码取自VTK/Examples/VisualizationAlgorithms/Tcl/warpComb.tcl）。

vtkPLOT3DReader pl3d

pl3d SetXYZFileName "$VTK\_DATA\_ROOT/Data/combxyz.bin"

pl3d SetQFileName "$VTK\_DATA\_ROOT/Data/combq.bin"

pl3d SetScalarFunctionNumber 100

pl3d SetVectorFunctionNumber 202

pl3d Update

vtkStructuredGridGeometryFilter plane

plane SetInputConnection [pl3d GetOutputPort]

plane SetExtent 10 10 1 100 1 100

vtkStructuredGridGeometryFilter plane2

plane2 SetInputConnection [pl3d GetOutputPort]

plane2 SetExtent 30 30 1 100 1 100

vtkStructuredGridGeometryFilter plane3

plane3 SetInputConnection [pl3d GetOutputPort]

plane3 SetExtent 45 45 1 100 1 100

vtkAppendPolyData appendF

appendF AddInputConnection [plane GetOutputPort]

appendF AddInputConnection [plane2 GetOutputPort]

appendF AddInputConnection [plane3 GetOutputPort]

vtkWarpScalar warp

warp SetInputConnection [appendF GetOutputPort]

warp UseNormalOn

warp SetNormal 1.0 0.0 0.0

warp SetScaleFactor 2.5

vtkPolyDataNormals normals

normals SetInputConnection [warp GetOutputPort]

normals SetFeatureAngle 60

vtkPolyDataMapper planeMapper

planeMapper SetInputConnection [normals GetOutputPort]

eval planeMapper SetScalarRange [[pl3d GetOutput] GetScalarRange]

vtkActor planeActor

planeActor SetMapper planeMapper

Subsampling Structured Grids规则网格数据降采样

如图像数据一样，规则网格数据也可以进行降采样（见105页“图像数据降采样”）。vtkExtractGrid用来执行数据降采样和提取。

vtkPLOT3DReader pl3d

pl3d SetXYZFileName "$VTK\_DATA\_ROOT/Data/combxyz.bin"

pl3d SetQFileName "$VTK\_DATA\_ROOT/Data/combq.bin"

pl3d SetScalarFunctionNumber 100

pl3d SetVectorFunctionNumber 202

pl3d Update

vtkExtractGrid extract

extract SetInputConnection [pl3d GetOutputPort]

extract SetVOI 30 30 -1000 1000 -1000 1000

extract SetSampleRate 1 2 3

extract IncludeBoundaryOn

在该例子中初始规则网格（维数为57X33X25）按照（1,2,3）的采样率进行采样，生成一个维数（1,17,9）的子集。函数IncludeBoundaryOn()保证了在没有选择边界的采样操作下也将边界提取出来。

**5.4 线性网格可视化**

线性网格在拓扑上规则，但是在几何上半规则（如图3-2（b））。线性网格也常用于数值分析。vtkRectilinearGrid由体素（vtkVoxel）或者像素（vtkPixel）单元组成。

Manually Create vtkRectilinearGrid

手动创建vtkRectlinearGrid

线性网格创建需要指定网格维数定义网格拓扑，以及三个标量数组定义x-y-z三个方向坐标来定义几何。下面代码改自VTK/Examples/DataManipulation/Cxx/RGrid.cxx。

vtkFloatArray \*xCoords = vtkFloatArray::New();

for (i=0; i<47; i++) xCoords->InsertNextValue(x[i]);

vtkFloatArray \*yCoords = vtkFloatArray::New();

for (i=0; i<33; i++) yCoords->InsertNextValue(y[i]);

vtkFloatArray \*zCoords = vtkFloatArray::New();

for (i=0; i<44; i++) zCoords->InsertNextValue(z[i]);

vtkRectilinearGrid \*rgrid = vtkRectilinearGrid::New();

rgrid->SetDimensions(47,33,44);

rgrid->SetXCoordinates(xCoords);

rgrid->SetYCoordinates(yCoords);

rgrid->SetZCoordinates(zCoords);

确保x，y，z三个方向的标量个数要与拓扑i,j,k三个方向维数保持一致。

Extract Computational Plane 提取计算平面

大多数情况下线性网格由接收vtkDataSet类型输入的filter来处理（见89页“可视化技术”）。而直接接收vtkRectilinearGrid类型输入的其中一个filter是vtkRectilinearGridGeometryFilter。该Filter根据其Extent变量值来提取网格中的局部点、线段或者平面（Extent是一个描述区域的六维向量（,））。

下面例子还是紧接上面代码VTK/Examples/DataManipulation/Cxx/RGrid.cxx，我们如下提取一个平面：

vtkRectilinearGridGeometryFilter \*plane =

vtkRectilinearGridGeometryFilter::New();

plane->SetInput(rgrid);

plane->SetExtent(0,46, 16,16, 0,43);

**5.5 不规则网格可视化**

不规则网格无论在拓扑上还是几何上都是不规则的（如图3-2（f））。不规则网格常用来进行数值分析（如有限元分析）。任何一中单元类型都可以用来表示不规则网格。

Manually Create vtkUnstructuredGrid

不规则网格定义是通过vtkPoints来定义几何，通过插入单元来定义拓扑。（下面代码取自VTK/Examples/DataManipulation/Tcl/BuildUGrid.tcl.）

vtkPoints tetraPoints

tetraPoints SetNumberOfPoints 4

tetraPoints InsertPoint 0 0 0 0

tetraPoints InsertPoint 1 1 0 0

tetraPoints InsertPoint 2 .5 1 0

tetraPoints InsertPoint 3 .5 .5 1

vtkTetra aTetra

[aTetra GetPointIds] SetId 0 0

[aTetra GetPointIds] SetId 1 1

[aTetra GetPointIds] SetId 2 2

[aTetra GetPointIds] SetId 3 3

vtkUnstructuredGrid aTetraGrid

aTetraGrid Allocate 1 1

aTetraGrid InsertNextCell [aTetra GetCellType] [aTetra GetPointIds]

aTetraGrid SetPoints tetraPoints

…insert other cells if any…

在为vtkUnstructuredGrid实例插入单元数据前，必须先执行Allocate()函数来分配空间。该函数的两个参数分别表示初始分配的数据空间大小，和需要额外内存时扩展的内存空间大小。一般情况大比较大的值会有较好的效果（因为需要较少的内存重分配操作）。

Extract Portions of the Mesh 提取局部网格

大部分情况下，不规则网格由接收vtkDataSet类型输入的filters处理（见89页“可视化技术”）。直接接收vtkUnstructuredGrid类型输入的其中一个filter是vtkExtractUnstructuredGrid。这个filter根据指定的点ids，单元ids和几何边界来提取局部网格（Extent参数定义一个包围盒）。下例代码取自VTK/Examples/VisualizationAlgorithms/Tcl/ExtractUGrid.tcl。

vtkDataSetReader reader

reader SetFileName "$VTK\_DATA\_ROOT/Data/blow.vtk"

reader SetScalarsName "thickness9"

reader SetVectorsName "displacement9"

vtkCastToConcrete castToUnstructuredGrid

castToUnstructuredGrid SetInputConnection [reader GetOutputPort]

vtkWarpVector warp

warp SetInput [castToUnstructuredGrid GetUnstructuredGridOutput]

vtkConnectivityFilter connect

connect SetInputConnection [warp GetOutputPort]

connect SetExtractionModeToSpecifiedRegions

connect AddSpecifiedRegion 0

connect AddSpecifiedRegion 1

vtkDataSetMapper moldMapper

moldMapper SetInputConnection [reader GetOutputPort]

moldMapper ScalarVisibilityOff

vtkActor moldActor

moldActor SetMapper moldMapper

[moldActor GetProperty] SetColor .2 .2 .2

[moldActor GetProperty] SetRepresentationToWireframe

vtkConnectivityFilter connect2

connect2 SetInputConnection [warp GetOutputPort]

connect2 SetExtractionModeToSpecifiedRegions

connect2 AddSpecifiedRegion 2

vtkExtractUnstructuredGrid extractGrid

extractGrid SetInputConnection [connect2 GetOutputPort]

extractGrid CellClippingOn

extractGrid SetCellMinimum 0

extractGrid SetCellMaximum 23

vtkGeometryFilter parison

parison SetInputConnection [extractGrid GetOutputPort]

vtkPolyDataNormals normals2

normals2 SetInputConnection [parison GetOutputPort]

normals2 SetFeatureAngle 60

vtkLookupTable lut

lut SetHueRange 0.0 0.66667

vtkPolyDataMapper parisonMapper

parisonMapper SetInputConnection [normals2 GetOutputPort]

parisonMapper SetLookupTable lut

parisonMapper SetScalarRange 0.12 1.0

vtkActor parisonActor

parisonActor SetMapper parisonMapper

该例中我们采用单元裁剪（利用单元id号）和一个连通 filter来提取局部网格。vtkConnectivityFilter(以及vtkPolyDataConnectivityFilter)用来提取数据中的连通部分（当单元共享点时，他们是连通的）。SetExtractionModeToSpecifiedRegions()方法设置提取哪个连通区域。连通filter默认提取最大的连通区域。也可以如该例中那样提取一个特定的区域，不过需要一些额外的实验来对应各个区域。

Contour Unstructured Grids　不规则网格等值线提取

vtkContourGrid用来提取不规则网格的等值线。与一般的vtkContourFilter类相比，这个类更加高效。通常你并不需要直接实例化该类，因为当vtkContourFilter识别到输入数据时vtkUnstructuredGrid类型时，vtkContourFilter内部会自动创建一个vtkContourGrid实例。

以上是可视化技术概述。接下来你可能希望学习图像处理和体绘制技术。也可以参考444页“Filters总结”部分了解VTK中的Filters。