东京爱情故事

1991 年 1 月 7 日

Openmesh

李博

首都师范大学

October 7, 2018

Abstract

给刘铭和其他学 openmesh 的人. 封面我乐意。
openmesh 官网http://www.openmesh.org/,
openvolumemesh 官网http://www.openvolumemesh.org/, 这个 openvolumemesh 是个三维体网格数据库。
libigl 官网http://libigl.github.io/libigl/

Contents

1 openmesh 数据结构

openmesh 在数据结构与 meshlib 采取相同的半边结构。主要区别有以下几点。

- openmesh 没有 meshlib 半边为 NULL 的概念，其实 openmesh 潜在地把所谓的边界（比如人脸）看成外界面 (eterior face) 的边，就像内部多边形的边一样自然。你只需要把这张人脸贴到球面上即可。(其实这种看法很自然，数学上我们常把有界曲面看成一个亏格为 0 曲面的一部分。比如在读 tutte's theorem 的时候，定理关于调和函数空间的维数，很容易应用到有界曲面，还有调和映射是否单射的讨论。) 如此一来，openmesh 遍历起来非常自然。
  下图绿笔涂抹处看作 exterior face。那么如何判断一个半边 he 是否为 exterior face 的半边，一个面是否为 exterior face 呢? 其实你只需要判断一个面 f 是否是 exterior face 即可, 如下

```
mesh.is_valid_handle(f)\\对于半边，只需将f替换为mesh.halfedge_handle(he)即可
```

- openmesh 从给定点 v 遍半边 he，不管逆时针还是顺时针，起始半边为最逆时针的半边。从给定点遍历点也是如此。如下图，不管 v 顶点的半边迭代器方向如何，第一个半边一定是红色的半边

2 添加属性

openmesh 添加属性相当方便。

- 如下是给网格元素添加属性。如果想要调用某顶点 v 的属性 ux，只需 mesh.data(v).ux 即可。

```
struct MyTraits :public OpenMesh::DefaultTraits
{
        VertexTraits{ double ux = 0,uy = 0; int index = 0; };
        EdgeTraits{ double weight = 0; };
        HalfedgeTraits{  };
        FaceTraits{  };
};
typedef OpenMesh::TriMesh_ArrayKernelT<MyTraits> TMyMesh;
typedef OpenMesh::PolyMesh_ArrayKernelT<MyTraits> PMesh;
```

- 上面的属性是对 TMyMesh 类都添加了属性，如果我只想对某个 TMyMesh 对象 mesh 添加属性怎么办呢?

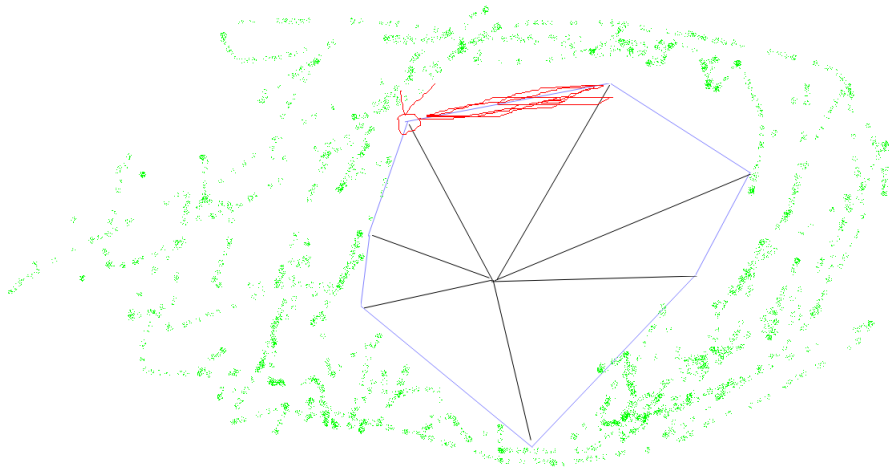

Figure 1: 数据结构

```
PMyMesh *dual = new PMyMesh();
FPropHandleT<PMyMesh::VertexHandle> primalToDual;
EPropHandleT<PMyMesh::VertexHandle> edgev;
primal.add_property(primalToDual);
primal.add_property(edgev);
for (TMyMesh::FaceIter fter = primal.faces_begin(); fter != primal.faces_end(); fter++)
{
                PMyMesh::Point p(0, 0, 0);
                primal.property(primalToDual, *fter) = dual->add_vertex(p);
}
```

上面展示了给对象 dual 添加属性的办法，可以看到 primalToDual 这是个面的属性，这个属性值是顶点元素。第四行语句给 primal 这个对象添加了该属性. 你可以用 $primal.remove_property(primalToDual)$ 删除这个属性。

3 数据的遍历

openmesh 提供了任何你需要的遍历方式。比如之前提到的给定一个顶点 v，遍历所有出去的半边，或者进来的半边。他们的调用方式相同，类名也是望文生义。如下典型几种:

```
for (TMyMesh::FaceVertexIter fvter = mesh.fv_begin(f); fvter.is_valid(); fvter++)
//遍历所有面
for (TMyMesh::VertexIter vter = mesh.vertices_begin(); vter != mesh.vertices_end(); vter++)
//遍历所有顶点
for (TMyMesh::VertexIter vter = mesh.vertices_begin(); vter != mesh.vertices_end(); vter++)
{
for (TMyMesh::VertexOHalfedgeCCWIter vhter = mesh.voh_ccwbegin(*vter); vhter.is_valid(); vhter
    ++)
{}

}
//遍历点又再遍历半边，这里是出去的半边，O表示out,同样你可以改称进的半边.如下:
for (TMyMesh::VertexIter vter = mesh.vertices_begin(); vter != mesh.vertices_end(); vter++)
{
for (TMyMesh::VertexIHalfedgeCCWIter vhter = mesh.vih_ccwbegin(*vter); vhter.is_valid(); vhter
    ++)
{}

}
//如下遍历每个面的点
for (PMyMesh::FaceIter fter = mesh.faces_begin(); fter != mesh.faces_end(); fter++)
{
for(PMyMesh::FaceVertexIter fvter = mesh.fv_begin(*fter);fvter.is_valid(); fvter++)
{}
}
```

其余同理，不再赘述。

4 部分函数表

默许如下约定：f 代表面，e 代表边，he 代表半边，v 代表顶点。

```
mesh.halfedge_handle(v)//返回以v为起点的半边
mesh.halfedge_handle(3)//返回id是3的半边
mesh.from_vertex_handle(he)//返回半边的起点
mesh.to_vertex_handle(he)//返回半边的终点
mesh.ccw_rotated_halfedge_handle(he)//he逆时针下一个半边，起点一致的半边
mesh.is_boundary(v)//顶点v是否是边界点
mesh.is_boudary(e)//e是否是边界边
mesh.is_boundary(f,false)//f是否有条边属于边界边
mesh.is_boundary(f,true)//f是否有个点属于边界点
TMyMesh::Point p//创建point类对象
mesh.point(v)//返回一个顶点v的point类对象
mesh.data(e).weight//调用边的属性，weight。承接上面第一个代码
primal.add_property(primalToDual);//一个mesh对象属性的添加
PMyMesh::Point p(0, 0, 0);
primal.property(primalToDual, *fter) = dual->add_vertex(p);//一个mesh对象属性的调用
mesh.flip(e)//翻转边e
mesh.is_flip_ok(e)//检查网格是否能够翻转边e
mesh.s_halfedge_handle(e,0)//返回e的一条半边
mesh.s_halfedge_handle(e,1)//返回e的另一条半边
TMyMesh::VertexOHalfedgeCCWIter hter = primal.voh_ccwbegin(v);//声明了一个逆时针遍历的半边迭代
    器
he = mesh.next_halfedge_handle(he);//半边的下一个半边
he = mesh.prev_halfedge_handle(he);//半边的前一个半边
mesh.opposite_halfedge_handle(he)//半边的对陈
mesh.s_edge_handle(he)\\半边的边
mesh.edge_handle(he)\\同上
mesh.calc_edge_length(e)\\计算边的长度
mesh.calc_sector_angle(he)\\计算半边和它下一个半边的角度
mesh.calc_sector_area(he)\\计算半边所在面的面积
mesh.add_vertex(TMyMesh::Point(0,0,0))//添加顶点
mesh.add_face(vector<Vertexhandle>)//添加面
mesh.request_face_status();
mesh.request_edge_status();
mesh.request_vertex_status();
mesh.delete_face(f, false);\\标记删除面，不删除因此产生的孤立点
mesh.garbage_collection();\\删除
mesh.release_edge_status();
mesh.release_face_status();
mesh.release_vertex_status();
```

5 程序展示

当我们参数化曲面时，即把一个曲面展开到二维平面，一个经典的方法是调和映射。我们已经知道固定边界点，内部的点唯一确定，而且完全可以通过求解线性方程组得到。可以得出，一个曲面的调和参数化，每个点的坐标由一组向量线性组合. 我们可以做这样的工作（这是有益的）：求出一个曲面调和坐标值空间的一组基底（这里的调和坐标值空间，并非你们已知或将来了解的曲面调和函数空间），只要我们给出边界点的坐标，立马得出所有点的坐标值。

```
#pragma once
#include <iostream>
#include <list>
#include<stack>
#include<nanogui\screen.h>
#include<nanogui\formhelper.h>
#include<igl\readOFF.h>
#include<igl\viewer\Viewer.h>
#include<igl\jet.h>
//#include<Eigen/Core>
#include<Eigen/IterativeLinearSolvers>
//#include <Eigen/SparseLU>
#include <OpenMesh/Core/IO/MeshIO.hh>
```

```
#include <OpenMesh/Core/Mesh/PolyMesh_ArrayKernelT.hh>
#include<OpenMesh\Core\Mesh\TriMesh_ArrayKernelT.hh>
#include <OpenMesh/Core/System/config.h>
#include <OpenMesh/Core/Mesh/Status.hh>
#include<GenEigsSolver.h>
#include<MatOp/SparseGenMatProd.h>
#include "igl/cotmatrix.h"
#ifndef PI
#define PI 3.1415926
#endif
struct MyTraits :public OpenMesh::DefaultTraits
{
        VertexTraits{ double ux = 0,uy = 0; int index = 0; };
        EdgeTraits{ double weight = 0; };
        HalfedgeTraits{  };
        FaceTraits{  };
};
typedef OpenMesh::TriMesh_ArrayKernelT<MyTraits> TMyMesh;
typedef OpenMesh::PolyMesh_ArrayKernelT<MyTraits> PMesh;
class Harmonic {
public:
        Harmonic(TMyMesh &me,igl::viewer::Viewer &vie):mesh(me),viewer(vie)
        {
                printf_s("create harmonic\r\n");
        }
        ~Harmonic()
        {}
        void init()
        {

                int i = 0, j = 0;
                for (TMyMesh::VertexIter vter=mesh.vertices_begin();vter!=mesh.vertices_end();
                    vter++)
                {
                //      printf_s("kaishi\r\n");
                        if (mesh.is_boundary(*vter))
                        {
                                mesh.data(*vter).index = j;
                                j++;
                        }
                        else
                        {
                                mesh.data(*vter).index = i;
                                i++;
                        }
                }

                //printf_s("%d  %d",mesh.n_vertices(),j);
                H.resize(i,i);
                H.setZero();
        }
        void init_weight()
        {
                Eigen::SparseMatrix<double> L;
                igl::cotmatrix(V, F, L);
                TMyMesh::HalfedgeHandle he;
                printf_s("kaishi\r\n");
                for (TMyMesh::EdgeIter eter = mesh.edges_begin(); eter != mesh.edges_end();
                    eter++)
                {
                        //printf_s("kaishi\r\n");
                        he = mesh.s_halfedge_handle(*eter, 0);
                        mesh.data(*eter).weight = L.coeff(mesh.to_vertex_handle(he).idx(),
                            mesh.from_vertex_handle(he).idx());
                        /*if (mesh.is_boundary(*eter))
                        {
                        printf_s(" %lf    %lf      %lf\r\n",mesh.data(*eter).weight,1/tan(mesh.
```

```
                        calc_sector_angle(mesh.next_halfedge_handle(he))), 1 / tan(mesh.
                        calc_sector_angle(mesh.prev_halfedge_handle(he))));
                }*/
                //printf_s("%f,  %f\r\n", mesh.data(*eter).weight, weight(*eter));
                if (L.coeff(mesh.to_vertex_handle(he).idx(), mesh.from_vertex_handle(
                    he).idx()) != L.coeff(mesh.from_vertex_handle(he).idx(), mesh.
                    to_vertex_handle(he).idx()))
                {
                        printf_s("´íÎó\r\n");
                }
        }

}
void construct_LL()
{
        Eigen::MatrixXd v(H.rows(),mesh.n_vertices()-H.rows());

        //int i = 0, j = 0;
        TMyMesh::HalfedgeHandle he; TMyMesh::VertexHandle v1, v2;
        v.setZero();
        LL = v;
        for (TMyMesh::EdgeIter eter=mesh.edges_begin();eter!=mesh.edges_end();eter++)
        {
                //printf_s("he\r\n");
                if (mesh.is_boundary(*eter))
                {
                }
                else
                {

                        he = mesh.s_halfedge_handle(*eter,0);
                        v1 = mesh.to_vertex_handle(he);
                        v2 = mesh.from_vertex_handle(he);
                        if (!mesh.is_boundary(v1)&&!mesh.is_boundary(v2))
                        {
                                H.coeffRef(mesh.data(v1).index, mesh.data(v2).index)
                                    -= mesh.data(*eter).weight;
                                H.coeffRef(mesh.data(v2).index, mesh.data(v1).index)
                                    -= mesh.data(*eter).weight;
                                H.coeffRef(mesh.data(v2).index, mesh.data(v2).index)
                                    += mesh.data(*eter).weight;
                                H.coeffRef(mesh.data(v1).index, mesh.data(v1).index)
                                    += mesh.data(*eter).weight;
                        }
                        else if (mesh.is_boundary(v1)&&!mesh.is_boundary(v2))
                        {
                                //printf_s("cunzai\r\n");
                                H.coeffRef(mesh.data(v2).index, mesh.data(v2).index)
                                    += mesh.data(*eter).weight;
                                v.coeffRef(mesh.data(v2).index,mesh.data(v1).index)=
                                    mesh.data(*eter).weight;
                        }
                        else if (!mesh.is_boundary(v1)&&mesh.is_boundary(v2))
                        {
                                H.coeffRef(mesh.data(v1).index, mesh.data(v1).index)
                                    += mesh.data(*eter).weight;
                                v.coeffRef(mesh.data(v1).index, mesh.data(v2).index) =
                                     mesh.data(*eter).weight;
                        }
                        else
                        {

                                printf_s("cuowullll\r\n");
                        }

                        //H.coeffRef()
                }
```

```
                }
                Eigen::SimplicialLDLT<Eigen::SparseMatrix<double>>llt;
                llt.compute(H);
                for (int j=0;j<v.cols();j++)
                {
                        LL.col(j) = llt.solve(v.col(j));
                }
        }
        void map()
        {

                Eigen::VectorXd ux(LL.cols()), uy(LL.cols()),x(LL.cols()),y(LL.cols()); int i
                    = 0;
                for (TMyMesh::VertexIter vter=mesh.vertices_begin();vter!=mesh.vertices_end();
                    vter++)
                {
                        if (mesh.is_boundary(*vter))
                        {
                                ux.coeffRef(i) = mesh.data(*vter).ux;
                                uy.coeffRef(i) = mesh.data(*vter).uy;
                                i++;
                        }
                }
                x = LL * ux;
                y = LL * uy;
                //i = 0;
                for (TMyMesh::VertexIter vter = mesh.vertices_begin(); vter != mesh.
                    vertices_end(); vter++)
                {
                        if (mesh.is_boundary(*vter))
                        {
                        }
                        else
                        {
                                mesh.data(*vter).ux = x.coeff(mesh.data(*vter).index);
                                mesh.data(*vter).uy = y.coeff(mesh.data(*vter).index);
                                i++;
                        }
                }
        }

        Eigen::SparseMatrix<double > H;
        Eigen::MatrixXd V,LL;
        Eigen::MatrixXi F;
        std::list<TMyMesh::VertexHandle> vlist;
protected:
        igl::viewer::Viewer &viewer;
        TMyMesh & mesh;


};
```

之所以说他是有益的，因为你知需要固定边界，再调用 map(), 即可得到调和映射。这里也可以轻应用 Dirichlet 边界条件, 从而得到共形映射，我们也可以改变权重.....
以上不仅用到 openmesh 还有 libigl.....