Geometry Processing (601.458/658)

Misha Kazhdan

Assignment 4: Goal

Implement gradient-domain processing for:

1. Smoothing/sharpening geometry
2. Solving the geodesics in heat problem

Assignment 4: Where and What

`Include/RiemannianMesh.inl`:

Implement the member function:

`Eigen::SparseMatrix< double >RiemannianMesh::CotangentMass( … )`

Input:

`size_t`: The number of vertices in the mesh (can ignore)

`const std::vector< SimplexIndex< K > > &`: The mesh triangles

`MetricFunctor &&`: A functor taking a triangle index and returning the metric.

Output:

The expression for the mass, $\overline{\mathbf{M}} \in \mathbb{R}^{2|\mathcal{T}|\times 2|\mathcal{T}|}$, w.r.t. to the cotangent vector field basis $\{\eta_1^1, \eta_1^2, \ldots, \eta_{|\mathcal{T}|}^1, \eta_{|\mathcal{T}|}^2\}$.

Assignment 4: Where and What

`Include/RiemannianMesh.inl`:

Implement the member function:

`Eigen::SparseMatrix< double >RiemannianMesh::Differential( ... )`

Input:

`size_t`: The number of vertices in the mesh

`const std::vector< SimplexIndex< K > > &`: The mesh triangles

Output:

The expression for the stiffness, $\mathbf{D} \in \mathbb{R}^{2|\mathcal{T}|\times|\mathcal{V}|}$, w.r.t. to the scalar and vector field bases $\{\phi_1, \ldots, \phi_{|\mathcal{V}|}\}$ and $\{\eta_1^1, \eta_1^2, \ldots, \eta_{|\mathcal{T}|}^1, \eta_{|\mathcal{T}|}^2\}$.

Assignment 4: Where and What

`Include/RiemannianMesh.inl`:

Eigen::SparseMatrix< double >RiemannianMesh::CotangentMass(…)

The function takes an `rvalue reference` to a functor:

MetricFunctor && metricFunctor

with MetricFunctor a `template` parameter.

The behavior of a MetricFunctor object should match that of:

std::function< Eigen::Matrix< double , 2 , 2 > (size_t t) > >

E.g. To get the metric $\mathbf{g}_\tau \in \mathbb{R}^{2\times 2}$ defined by triangle τ, call:

metricFunctor(τ).

Assignment 4: Where and What

Incl

You will not need create the functor yourself, that is done for you in the implementation of the member functions:

RiemannianMesh::CotangentMass(const Mesh & mesh)

.)

The fun

As long as you implemented the member function Mesh::g, the functor will be generated correctly for you.

wi

Feel free to add helper function definitions/declarations to either:

Include/RiemannianMesh.[h/inl]

The behavior of a MetricFunctor object should match that of:

std::function< Eigen::Matrix< double , 2 , 2 > (size_t t) > >

E.g. To get the metric $\mathbf{g}_\tau \in \mathbb{R}^{2\times 2}$ defined by triangle τ, call:

metricFunctor(τ).

Assignment 4: Where and What

Note:

The `GradientDomainFiltering` code will perform some basic sanity-testing by invoking the function:

RiemannianMesh::Validate

The mass of the function that is constantly one should equal the area of the mesh.

The stiffness of constant functions is zero.

The stiffness matrix is the pull-back, under the differential, of the inner-product on the space of cotangent vector fields.

Filtering: Where and What

`GradientDomainFiltering/GradientDomainFiltering.cpp`:

The executable supports applying **local** smoothing/sharpening edits to the x-, y-, and z-coefficients of the mesh vertices using a spray-can interface.

Weights are defined **per-triangle**, prescribing the scale factor to be applied to the differentials of the input (original) vertex coordinates within that triangle.

Filtering: Where and What

`GradientDomainFiltering/GradientDomainFiltering.cpp`:

In the constructor:

GDFilteringViewer::GDFilteringViewer(…)

Implement the functionality initializing the solver member data:

GDFilteringViewer::_solver

using:

- GDFilteringViewer::_scalarMass: The mass matrix associated with scalar functions
- GDFilteringViewer::_cotangentMass: The mass matrix associated with vector fields
- GDFilteringViewer::_differential: The differential matrix
- GDFilteringViewer::_differentialTranspose: The transpose of the differential matrix
- GDFilteringViewer::differentialFittingWeight: The weight of fitting the target differential

These matrices will have been initialized for you.

Filtering: Where and What

`GradientDomainFiltering/GradientDomainFiltering.cpp`:

In the constructor:

`GDFilteringViewer::_updateNumericalFactorization( … )`

Implement the functionality to update the solver when the system changes:

If the user changes the differential fitting weight

Recall that `Eigen`'s simplicial factorization has a member function `SimplicialLDLt::factorize` which only updates the numerical factorization.

Since the sparsity structure of the system matrix hasn't changed, there is no need to expend efforts recomputing the symbolic factorization.

Filtering: Where and What

`GradientDomainFiltering/GradientDomainFiltering.cpp`:

In the constructor:

GDFilteringViewer::animate(…)

Compute the modulated differential and solve the gradient-domain problem for the best-fitting x-, y-, and z-coefficients.

- If the user changes the weight associated with a triangle.
- If the user changes the differential fitting weight.

The original vertex positions are stored in the member data:

GDFilteringViewer::_originalVertices

The modified vertex positions should be updated in the mesh:

_mesh.vertices

Filtering: Command Line Arguments

The executable takes the following parameters:

--in <input geometry>:
The file-name of the mesh (mandatory)

--dWeight <differential fitting weight>:
The importance given to fitting the modulated differentials.
(The weight associated to value-fitting is 1.)

Filtering: User Interface

You can interface with the running executable using:

[space bar]: Start/stop the animation by hitting.
(Animation is on by default.)

‘[‘: Decrease differential fitting weight

‘]’: Increase differential fitting weight

Filtering: User Interface

You can interface with the running executable using:

‘s‘: Enter/exit selection mode

‘{‘: Decrease selection radius

‘}’: Increase selection radius

[left mouse]: Decrease modulation weight

[right mouse]: Increase modulation weight

‘v‘: Toggle modulation weight visualization

Geodesics: Where and What

`GeodesicsInHeat/GeodesicsInHeat.cpp`:

The executable solving the single-source geodesic problem using the “heat” method. This requires solving two linear systems:

1. Short time-step heat diffusion
2. Gradient fitting

Geodesics: Where and What

`GeodesicsInHeat/GeodesicsInHeat.cpp`:

In the constructor:

GeodesicsInHeat::GeodesicsInHeat(…)

Implement the functionality initializing the solver member data:

GeodesicsInHeat::_diffusionSolver

GeodesicsInHeat::_distanceSolver

using:

GeodesicsInHeat::_scalarMass: The mass matrix associated with scalar functions

GeodesicsInHeat::_scalarStiffness: The stiffness matrix associated with scalar fields

GeodesicsInHeat::diffusionTime: The time for diffusing the delta

GeodesicsInHeat::regularizationWeight: The value-fitting weight used to regularize the gradient-fitting system.

The system matrices will have been initialized for you.

Geodesics: Where and What

`GeodesicsInHeat/GeodesicsInHeat.cpp`:

In the constructor:

`GeodesicsInHeat::animate( … )`

Set up the constraints for, and solve the associated systems:

- Initialize the delta:
  the index of the source vertex is `GeodesicsInHeat::_sourceIndex`
- Diffuse the delta
- Compute and adjust the diffused delta's differential:
  Recall that the inner-product on **tangent** vectors is available through `Mesh::g`
- Fit a scalar field to the target differential (using value regularization to make the system positive-definite)
- Adjust the constant term

Geodesics: Command Line Arguments

The executable takes the following parameters:

--in <input geometry>:
The file-name of the mesh (mandatory)

--dTime <diffusion time>:
The time-step for diffusing the delta.

--rWeight <regularization weight>:
The weight of the regularization term encouraging the values to be centered around 0 (should be small)

Geodesics: User Interface

You can interface with the running executable using:

[left click]: Set the source vertex

‘[‘: Decrease the number of bands

‘]’: Increase the number of bands

Assignment 4: Questions

Filtering:

What happens as you increase/decrease the differential fitting weight?

Geodesics:

What happens as you increase/decrease the diffusion time?