A visualization for Discrete Stratified Morse Theory

Yulong Liang

April 30, 2018

1 Introduction

This project is a visualization for Discrete Stratified Morse Theory. In this project, a web-based visualization tool was developed to illustrate the discrete gradient field after implementing Discrete Stratified Morse Theory on a simplicial complex and to demonstrate the process of performing simplification with Discrete Stratified Morse Theory on this discrete gradient field to reduce data complexity without changing homotopy type.

2 Related work

Classical Morse Theory Classical Morse Theory is a mathematical technique which was introduced by Marston Morse. It studies Morse Function on a manifold to enable one to find CW-complices and handle decompositions to analyze the topology of that manifold.

Discrete Morse Theory is a combinatorial adaptation of Morse Theory developed by Robin Forman, which focuses on a discrete simplicial complex with Discrete Morse Function instead of a continuous manifold with Morse Function. The idea of Discrete Morse Theory is to recognize critical simplicial complices and non-critical simplicial comlice pairs and reduce data complexity by removing noncritical pairs while keeping the homotopy type of the original data.

Discrete Stratified Morse Theory Discrete Stratified Morse Theory is a expansion of Discrete Morse Theory proposed by Kevin Knudson and Bei Wang. It expands the usage of Discrete Morse Theory from a simplicial complex with Discrete

Morse Function to a simplicial complex with any function by isolating violators into separate strata before normal procedure, which enables simplifications as well as many practical applications.

Visualization There are only few implementations of the visualization of Discrete Morse Theory and Discrete Stratified Morse Theory. Attila Gyulassy, Joshua A. Levine, and Valerio Pascucci developed a Visualization of Discrete Gradient Construction which shows the discrete gradient field of a 2d triangulation. However, there are no existing visualization for using Discrete Morse Theory/Discrete Stratified Morse Theory to perform data simplification.

3 Contribution overview

The project supports the following functions:

- Reading an `OFF` file of a 2d simplicial complex with function value on each simplex (vertex, edge and face) and illustrating on the canvas.
- Calculating and marking violator simplices, critical simplices and non-critical simplex pairs, namely discrete gradient field.
- Demonstrating the process of removing non-critical simplices and the result after running Discrete Stratified Morse Theory.

This project can serve as an educational tool for researchers and college students to study and understanding the underlying process of Discrete Morse Theory and Discrete Stratified Morse Theory.

4 Technical Background

Two Important Sets For a simplicial complex K, let $\alpha^{(p)}$ denote a simplex with dimension p. Let $\alpha < \beta$ denote that simplex α is a face of β, or simplex β is a coface of α. Let $f : K \to \mathcal{R}$ denote the function over the simplical complex K. Then we can define the following sets,

$$U(\alpha) = \beta^{(p+1)} > \alpha | f(\beta) \leq f(\alpha)$$

$$L(\alpha) = \gamma^{(p1)} < \alpha | f(\gamma) \geq f(\alpha)$$

That is to say, $U(\alpha)$ represents the cofaces of α (the adjacent simplices of *alpha* which have dimensions that 1 dimension greater than α and values less than or equal to α) and $L(\alpha)$ represents the faces of α (the adjacent simplices of *alpha* which have dimensions that 1 dimension less than α and values greater than or equal to α).

Discrete Morse Function A function $f : K \rightarrow \mathcal{R}$ is a Discrete Morse Function if $\forall \alpha^p \in K$,

$$|U(\alpha)| \leq 1 \quad and \quad |L(\alpha)| \leq 1$$

Stratified simplicial comlex A stratified simplicial complex is a simplicial complex K with a stratification equipped. A stratification of a simplicial complex K is a collection of subsets of K that can denote the original simplicial complex K.

Discrete Stratified Morse Function A function $f : K \rightarrow \mathcal{R}$ over a simplicial complex with stratification s is a Discrete Stratified Morse Function if within each stratum,

$$U(\alpha) = \beta^{(p+1)} > \alpha | f(\beta) \leq f(\alpha)$$

$$L(\alpha) = \gamma^{(p1)} < \alpha | f(\gamma) \geq f(\alpha)$$

Violator A simplex is a violator if it satifies any one of the following conditions,

$$|U(\alpha)| \geq 2$$

$$|L(\alpha)| \geq 2$$

$$|U(\alpha)| = 1 \cap |L(\alpha)| = 1$$

Critical A simplex is critical if it satifies the following conditions,

$$|U(\alpha)| = 0 \cap |L(\alpha)| = 0$$

Non-critical A simplex is non-critical if it satifies either of the following conditions,

$$|U(\alpha)| = 1 \cap |L(\alpha)| = 0$$

$$|U(\alpha)| = 0 \cap |L(\alpha)| = 1$$

5 Methods

5.1 Marking

Given a simplicial complex with any function value on its simplices, we can calculate the two sets $U(\alpha)$, $L(\alpha)$ for each of its simplices. We can categorize the simplices into different groups, namely violators, criticals, or non-criticals by the cardinalities of the two sets.

Typically, there are two algorithms for simplex categorization. The idea of the first algorithm is to first identify the violators. Then remove the violators from consideration. Finally identify the criticals and non-critical pairs.

Algorithm 1

```
function DSMT1(K)
    for all simplex $\alpha \in k$ do
        Calculate set $U(\alpha)$
        Calculate set $L(\alpha)$
    for all simplex $\alpha \in k$ do
        if $|U(\alpha)| + |L(\alpha)| \geq 2$ then
            Mark $\alpha$ as violator
    Remove violators from the two sets
    for all simplex $\alpha \in k$ do
        if $|U(\alpha)| + |L(\alpha)| = 0$ then
            Mark $\alpha$ as critical
        if $|U(\alpha)| + |L(\alpha)| = 1$ then
            Mark $\alpha$ as non-critical
    Pair the non-critical one with another by relationship
    return all the violators, criticals, and non-critical pairs
```

The second algorithm is a refinement of the first algorithm. It deviates in the process of eliminating the violators. Instead of marking and removing all the violators simultaneously, this algorithm adopts a iterative approach where it first eliminates the violators with the lowest dimension and calculate the two sets, then eliminates the violators with the second lowest dimension and calculate the two sets again. The algorithm will repeat until there are no more violators left. This algorithm can keep as many non-critical pairs as possible comparing to the former algorithm so that it can simplify the simplical complex to the largest extent.

Algorithm 2

```
function DSMT2(K)
    for all simplex $\alpha \in k$ do
        Calculate set $U(\alpha)$
        Calculate set $L(\alpha)$
    for dimension d = 1 to N do
        for all simplex $\alpha \in k$ which has dimension = d do
            if $|U(\alpha)| + |L(\alpha)| \geq 2$ then
                Mark $\alpha$ as violator
        Remove violators from the two sets
        if there are no violators then
            Break
    for all simplex $\alpha \in k$ do
        if $|U(\alpha)| + |L(\alpha)| = 0$ then
            Mark $\alpha$ as critical
        if $|U(\alpha)| + |L(\alpha)| = 1$ then
            Mark $\alpha$ as non-critical
    Pair the non-critical one with another by relationship
    return all the violators, criticals, and non-critical pairs
```

Specifically, all violators will be marked as red, all criticals will be marked as yellow and all non-critical pairs will be marked by an arrow starting from the lower-dimensional simplex pointing to the higher-dimensional simplex to denote the gradient.

5.2 Removing

In this 2d edition, non-critical pairs involve two types,

- Face-edge pair
- Edge-vertex pair

5.2.1 Removing Face-edge Pair

Face-edge pairs will be removed first. The method involves no animations. The algorithm is as follows,

Algorithm 3 Removing Face-edge Pair

```
function feRemoval(K)
    for all face-edge pairs do
        Remove from canvas
        Remove from $U(\alpha)$
        Remove from $L(\alpha)$
```

5.2.2 Removing Edge-vertex pair

Edge-vertex are removed in a iterative pattern. We treat each violator/critical simplex as the pivot of the graph. For each violator/critical simplex, we perform a breadth first search for non-critical pairs and then remove each pair and pull its connecting component to the pivot. We do this until there are no more non-critical pairs around that pivot and then proceed to the next pivot. The algorithms is as follows,

Algorithm 4 Removing Edge-vertex Pair

```
function evRemoval(K)
    for all violator/critical simplices $p$ do
        repeat
            for all adjacent simplices $\alpha$ do
                if $\alpha$ is an element of a pair $\alpha - \beta$ then
                    Pull the connecting component of $\beta$ to $p$
                    Remove $\alpha - \beta$
        until no adjacent simplices of $p$ are non-critical
```

Moreover, animation is performed when doing the Edge-vertex pair removal. It takes two steps to finish the removal animation,

1. Move the connecting component one the other side to the pivot.
2. Update the adjacency information of the connecting component.
3. Remove the non-critical pair.

5.3 Corner Case

Some corner cases might show up during the non-critical pairs removal. The solutions are as follows,

- **Self-looping Edge**
  Pre-define the path as two curves connecting together.
- **Collinear Edges**
  Pre-calculate the number of collinear edges, pre-define the path as a left curve for the first colliner edge and a right curve for the second colliear edge.
- **Degenerated Face as a line**
  Pre-define the path as two curves connecting the two distinct points with colored fill and none stroke.
- **Degenerated Face as a point**
  Pre-define the path as two curves connecting the distinct point and a point below that point with colored fill and none stroke.

6 Visual Design

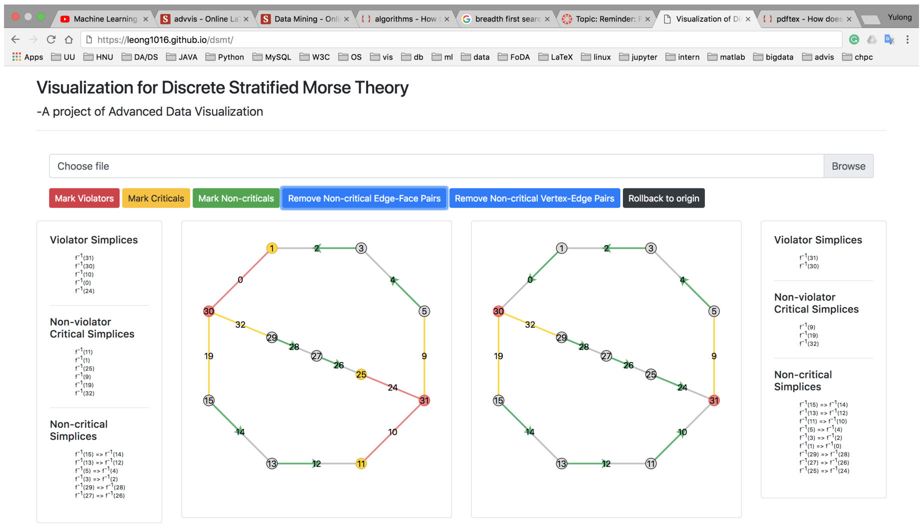
The main interface has three parts,

1. The above panel for selecting files and controlling progress pipeline.
2. The left part of main space is the illustration for algorithm 1.
3. The right part of main space is the illustration for algorithm 2.

With each illustration, both text description and graph demonstration are shown synchronously. Moreover, the interface is responsive to the resolution of the device screen.

7 Results

Below is a screen capture of the main interface,


Visualization for Discrete Stratified Morse Theory
-A project of Advanced Data Visualization
Choose file
Browse
Mark Violators
Mark Criticals
Mark Non-criticals
Remove Non-critical Edge-Face Pairs
Remove Non-critical Vertex-Edge Pairs
Rollback to origin

Violator Simplices

$f^{-1}(31)$
$f^{-1}(30)$
$f^{-1}(10)$
$f^{-1}(0)$
$f^{-1}(24)$

Non-violator Critical Simplices

$f^{-1}(11)$
$f^{-1}(1)$
$f^{-1}(25)$
$f^{-1}(9)$
$f^{-1}(19)$
$f^{-1}(32)$

Non-critical Simplices

$f^{-1}(15) => f^{-1}(14)$
$f^{-1}(13) => f^{-1}(12)$
$f^{-1}(5) => f^{-1}(4)$
$f^{-1}(3) => f^{-1}(2)$
$f^{-1}(29) => f^{-1}(28)$
$f^{-1}(27) => f^{-1}(26)$

Violator Simplices

$f^{-1}(31)$
$f^{-1}(30)$

Non-violator Critical Simplices

$f^{-1}(9)$
$f^{-1}(19)$
$f^{-1}(32)$

Non-critical Simplices

$f^{-1}(15) => f^{-1}(14)$
$f^{-1}(13) => f^{-1}(12)$
$f^{-1}(11) => f^{-1}(10)$
$f^{-1}(5) => f^{-1}(4)$
$f^{-1}(3) => f^{-1}(2)$
$f^{-1}(1) => f^{-1}(0)$
$f^{-1}(29) => f^{-1}(28)$
$f^{-1}(27) => f^{-1}(26)$
$f^{-1}(25) => f^{-1}(24)$

The following series of screen captures shows the process of simplying a simplicial complex,

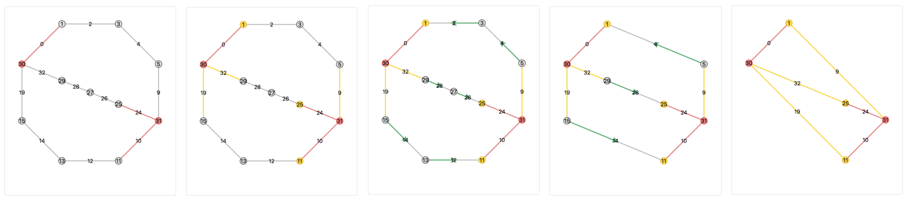


Figure 1: Algorithm 1

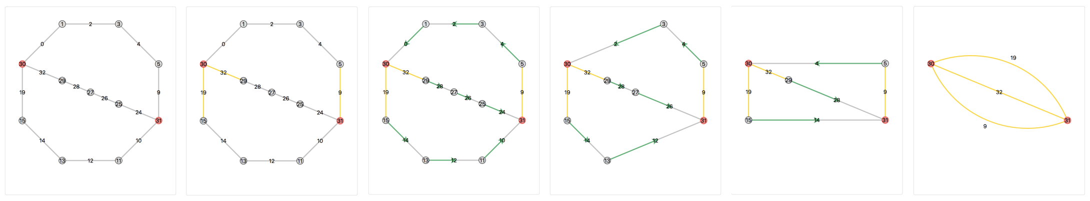


Figure 2: Algorithm 2

This visualization tool can be visited via the link `https://leong1016.github.io/dsmt/`.

8 Conclusion and discussions

This visualization tool supports most of the 2d simplicial complex with any function on the simplices. It can serve the purpose of education as well as demonstration. However, there are also something to be improved,

- Increasing the robustness of the visualization to support every 2d simplicial complex.
- Using a slider with scales to control the steps in the pipeline of simplification.
- Using 3d instead of 2d visualization, treating function value as elevation.
- Extending the support for 3d tetrahedrons.