LAB REPORT: LAB 5

TNM079, MODELING AND ANIMATION

Lucas Hägg
lucha567@student.liu.se

Monday 10th June, 2024

Abstract

In this laboratory report, the focus is on the application of level set methods in numerical simulations, particularly aimed at modeling and animating deformable interfaces. The level set approach offers a robust framework for handling topological changes. The Key methods are by solving partial differential equations (PDEs) that govern the interface dynamics. These are advection, reinitialization, and curvature motion processes. Initial results confirm that the level set method is effective in preserving the interface's integrity while allowing complex topological transformations, making it good for detailed simulations. Overall, this report goes through some of the theory and implementations for a level set framework.

1 Introduction

This lab report examines the use of level set methods for modeling and animating deformable interfaces. The lab aims to deepen understanding and practical skills in applying level set techniques to simulate dynamic changes in interfaces. The specific purpose is to implement how level set methods manage complex surface deformations.

2 Background

In this laboratory, several tasks were implemented to understand the dynamics of level set methods for simulating deformable interfaces. Each task focuses on different parts of the level set framework.

2.1 Differential Calculations

The first task was to implement differential calculations for understanding changes in the level set function over the grid. Using discrete differential operators, changes across the interface were calculated. These operators, such as forward, backward, and central differences, allow for the approximation of derivatives at grid points and are needed for the accurate simulation of motion and deformation in level sets.

- Forward Difference in x-direction:

$$\frac{\partial \phi}{\partial x} \approx \frac{\phi(i+1,j) - \phi(i,j)}{\Delta x} \tag{1}$$

- Backward Difference in x-direction:

$$\frac{\partial \phi}{\partial x} \approx \frac{\phi(i,j) - \phi(i-1,j)}{\Delta x} \tag{2}$$

- Central Difference in x-direction:

$$\frac{\partial \phi}{\partial x} \approx \frac{\phi(i+1,j) - \phi(i-1,j)}{2\Delta x} \tag{3}$$

2.2 Advection Implementation

Advection plays a critical role in moving the level set interface according to a velocity field. In this task, advection was implemented by solving the level set equation:

$$\frac{\partial \phi}{\partial t} - \mathbf{V} \cdot \nabla \phi = 0 \tag{4}$$

Where **V** represents an external velocity field. This equation describes how the interface moves with the flow defined by **V**, requiring the calculation of the gradient and ensuring that the movement respects the speed and direction of the underlying flow.

2.3 Erosion and Dilation

Erosion and dilation are fundamental operations for modifying the shape and size of the level set interface. Erosion reduces the interface, pulling it inward, while dilation expands it outward. the implementation focuses on how the level set function ϕ evolves under the influence of a predefined speed function **F** (speed in the normal direction).

$$\frac{\partial \phi}{\partial t} + F \cdot |\nabla \phi| = 0 \tag{5}$$

Here, $\nabla\phi$ represents the gradient of the level set function, and $|\nabla\phi|$ is its magnitude. F determines the motion's speed and direction, which can either expand the interface (dilation) if positive or shrink it (erosion) if negative.

3 Tasks

This section of the lab report outlines the specific tasks that were executed and the methods

3.1 Differential Calculations

The functions in the class LevelSet were called $Diff + \{X, Y, Z\} + \{p, m, pm\}$ where $\{X, Y, Z\}$ denotes dimension along which the derivative should be taken and p=plus, m=minus and pm=central difference. In total, 15 differential operators were implemented.

The level set grid values were accessed using: $mGrid.GetValue()$ which retrieves the function value at specific grid indices. Below are two examples, showing the implementation of DiffXm and Diff2Xpm.

if $GV()$ is short for $mGrid.GetValue()$:

$$DiffXm = (GV(i,j,k) - GV(i-1,j,k))/Dx$$

$$Diff2ZXpm = GV(i+1,j,k+1) - GV(i+1,j,k-1) + GV(i-1,j,k-1) - GV(i-1,j,k+1))/(4 * pow(Dx,2))$$

Note that the grid spacing were uniform, making $dx = dy = dz$.

3.2 Advection Implementation

The core of this implementation involved solving the advection equation, and solving a stable time step.

3.2.1 ComputeTimestep():

The purpose of this function is to calculates a stable time step for the numerical simulation to ensure stability. The time step depends on the grid spacing and the maximum velocity in the velocity field. The time step is computed as:

$\Delta t = 0.9\Delta x / \max(|V|)$

where Δx is the gridspace, multiplied with a safety decimal. $\max(|V|)$ is the maximum speed at which the inteface could possibly move across the grid.

3.2.2 Evaluate():

The Evaluate() function calculates the rate of change of the level set function at each grid point based on equation 4. First step was to transform the grid coordinates to world coordinates, and then get the velocity from the vector field at that position.

- TransformGridToWorld(x, y, z);
- V = VectorField.GetValue(x,y,z);

The second step was then to calculate the gradient $\nabla\phi$. This was done by using the differential operations already implemented in 3.1. The gradient calculation was discretized by an upwind scheme, which means that the finite difference is evaluated on points that have

been "touched" by the numerical wave. Depending on the direction of the velocity field, either a positive or a negative differential operation was called. This was checked using an if-else statement. See below for implementation in x-direction:

if $V.x > 0$ **then**
  $gradient.x \leftarrow DiffXm$
else
  $gradient.x \leftarrow DiffXp$
end if

At last, a final computation was done, returning the value of equation 5:

$$-1.0f * glm :: dot(V, gradient)$$

3.3 Erosion and Dilation

Below describes the implementation of dilation (advection outwards) and erosion (advection inwards).

3.3.1 ComputeTimestep():

The timestep Δt is calculated as below:

$$\Delta t = \min\left(\frac{\Delta x}{|F|}, 1.0\right) \tag{6}$$

where Δx is the grid spacing and F is the speed function. The min value was chosen using $std :: min()$.

3.3.2 Evaluate():

The Evaluate() function calculates the rate of change similar to the Advection implementation. Here, the gradient norm was calculated using Godunov Scheme, which calculates the squares of partial derivatives in each direction, choosing the appropriate upwind or downwind difference based on the sign of F. The Godunov Scheme was already implemented in the given code structure and updated local the local variables ddx2, ddy2 and ddz2 (squares of partial derivatives).
Then, the square root of the sum of the partial derivatives retrieved $|\nabla\phi|$ in equation 5:

$$-1.0f * F * std :: sqrt(ddx2 + ddy2 + ddz2)$$

4 Results

Results of dilation and erosion are presented on a sphere:

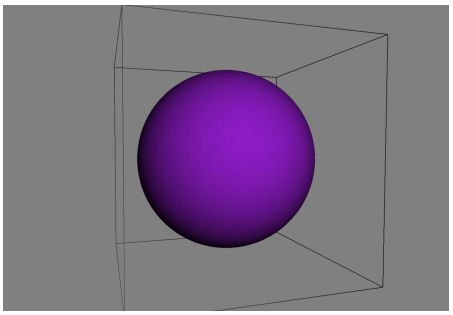
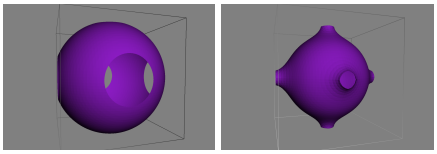


Figure 1: Level set sphere.

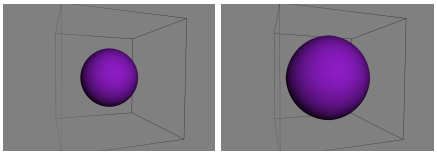
Dilation, followed by erosion:



(a) (b)

Figure 2: (a) Sphere dilated with 30 iterations, (b) Sphere dilated with 30 iterations, then 30 iterations of erosion.

Erosion, followed by dilation:



(a) (b)

Figure 3: (a) Sphere eroded with 30 iterations, (b) Sphere eroded with 30 iterations, then 30 iterations of dilation.

Results of advection are presented on a plane using a vortex vector field as V in equation 4:

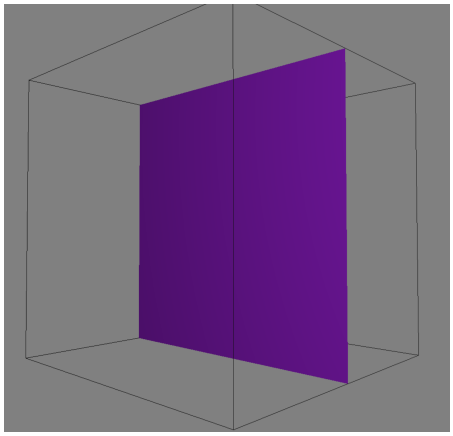
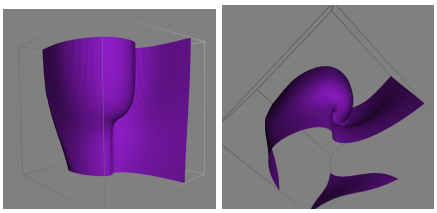


Figure 4: Level set plane, camera angle slight to the right.



(a) (b)

Figure 5: (a) camera angle slight to the right, (b) camera angle from below.

5 Conclusion

This lab demonstrated the effective use of level set methods to model dynamic interfaces, implementing advection, erosion, and dilation.

6 Lab partner and grade

Lab partner was Kenton Larsson and the grade I am for is 3.

References

[1] Mark Eric Dieckmann. *MESH DATA STRUCTURES*. Linkoping university, 2023.