LAB REPORT: LAB 5

TNM079, MODELING AND ANIMATION

Linnea Bergman
linbe810@student.liu.se

Saturday 14th July, 2018

Abstract

In this lab a level set framework was implemented. A level set is an implicit surface which can be deformed by solving a set of Partial Differential Equations, abbreviated PDEs. PDEs for performing dilation, erosion, advection and smoothing was implemented.

1 Introduction

This lab uses temporal and spatial discretization of the necessary equation of motion. The first determines how the surface is evolved in time using discrete time steps, while the latter determines how the discrete differentials (e.g. gradients, curvature) are computed in space.

Two different classes of Partial Differential Equations (abbreviated PDEs), hyperbolic and parabolic, give use to spatial discretization strategies, ”upwinding” and ”central differencing”, which are used.

The level set function is kept as close as possible to the signed distance function, which describes the closest distance to the surface for each point. The sign of the distance determines whether it is outside or inside the surface.

In the lab PDEs for performing operations such as dilation, erosion, advection and smoothing the surface were constructed.

2 Assignments

This sections contains a thorough explanation and description of the computed assignments.

2.1 Assignment Implement differential calculations

The different differentials, ϕ_x^+, ϕ_x^-, $\phi_x^\pm$, $\phi_{xx}^\pm$, $\phi_{xy}^\pm$, were implemented in the class LevelSet, int the file Levelset.cpp. The functions are called Diff + x, y, z + p, m, pm where x, y, z denotes the dimension for the derivative and $p = plus(\phi_x^+)$, $m = minus(\phi_x^-)$ and $pm = centraldifference(\phi_x^\pm)$. There were one function for each differential, a total of 15 function, and all had the grid coordinates (i, j, k) as input values. There were converted to world coordinates (x, y, z) before the calculations. The equation used for respective differential were:

$$\phi_x^+ = \frac{\phi_{i+1,j,k} - \phi_{i,j,k}}{\Delta x}, \qquad if V_x < 0 \tag{1}$$

$$\phi_x^+ = \frac{\phi_{i,j,k} - \phi_{i-1,j,k}}{\Delta x}, \qquad if V_x > 0 \tag{2}$$

$$\phi_x^\pm = \frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x} \tag{3}$$

$$\phi_{xx}^\pm = \frac{\phi_{i+1,j,k} - 2\phi_{i,j,k} + \phi_{i-1,j,k}}{(\Delta x)^2} \tag{4}$$

$$\phi_{xy}^\pm = \frac{\phi_{i+1,j+1,k} - \phi_{i+1,j-1,k} + \phi_{i-1,j-1,k} - \phi_{i-1,j+1,k}}{4\Delta x \Delta y} \tag{5}$$

All of the equations above are for the x-axis, but the same equations were used for y- and z-axis as well, with the small difference of alternating (i, j, k)-variables. ϕ_x equals $\frac{\partial \phi}{}$ respectively, so $\phi_{xx} = \frac{\partial^2 \phi}{^2}$ and $\phi_{xy} = \frac{\partial^2 \phi}{\partial x \partial y}$. To get the world coordinates (x, y, z) for the point GetValue() was used on the variable mGrid for the desired point. Then the respective equation (1-5) was used and the computed value was returned. The gradient was studied for a sphere for $\phi^+, \phi^-, \phi^\pm$.

To visualize the gradient, a vector cut plane was added to the level set, which gets the gradient from the GetGradient-function in LevelSet.cppp. This was used to verify the normals for each of the first order differentials.

2.2 Assignment The signed distance property

A level set object was added to the scene as well as a scalar cut plane, which maps the scalar values of the LevelSet-function to colours with the selected colourmap. The selected colourmap was the "Iso contour", which maps green to red in a wrap-around cycle. Each red-green cycle represented a change in the level set-function at 0,1.

The reinitialize function was used multiple times to be able to study what happens to the levelset. Both the scalar cut plane and the vector cut plane was studied.

2.3 Assignment Implement erosion and dilation

The operator for erosion and dilation was implemented in the file OperatorDilateErode.h. For it to function properly the function ComputeTimestep() was modified to compute a stable timestep.

$$\Delta t < \frac{min\{\Delta x, \Delta y, \Delta z\}}{|F|}$$

Δt is the desired timestep and F the constant speed function. The speed function differ for dilation and erosion and are the following:

$$Dilation : F(\mathbf{x}) = c_1, \qquad c_1 > 0$$

$$Erosion : F(\mathbf{x}) = c_2, \qquad c_2 < 0$$

where c_1 and c_2 are constants. The speed function is used in the function Evaluate() which returns the rate of change ($\frac{\partial \phi}{\partial t}$) for a grid point (i, j, k). Input values were the grid point (i, j, k). These along with the normalized speed function and some variables of type float used to store the partial derivatives ($\frac{\partial^2 x}{\partial t}$, $\frac{\partial^2 y}{\partial t}$, $\frac{\partial^2 z}{\partial t}$) were used as input values to the given function Godunov().

Godunov uses an upwinding scheme to calculate the gradient squared. The return value of Evaluate() was the speed function multiplied with the square root of the sum of the partial derivatives. To see the result add a level set object in the GUI and do dilation and erosion.

2.4 Assignment Implement advection from an external vector field

In this assignment the file OperatorAdvect.h was modified with an advection operator. First and foremost the function ComputeTimestep() and Evaluate() was improved.

$$\Delta t < min\left\{ \frac{\Delta x}{|V_x|}, \frac{\Delta y}{|V_y|}, \frac{\Delta z}{|V_z|} \right\} \tag{6}$$

Equation 6 was used as the basis for calculating the timestep. Δt is the desired timestep and V a vector field representing an external velocity field. Δx is acquired with the use of GetDx() and the velocity field is acquired from the variable mVectorField, where the length of the max value was used. The return value was calculated with Equation 6 along one axis and multiplied with a number close to, but smaller than, one since the timestep should be smaller according to Equation 6.

For Evaluate() the grid coordinates (i, j, k) were the input values. First, these were transformed into world coordinates (x, y, z) and then the corresponding vector field was extracted from mVectorField with (x, y, z). Each axis in the velocity field was tested to see if the value was positive or negative. For positive values the differential for ϕ^+ (Equation 1) was used and for negative ϕ^- (Equation 2). The gradient was equal to the vector containing the differentials for $x-, y-$, and $z-$axis. Return value of Evaluate() was the negative velocity field multiplied with the gradient. To test advection, add a level set object to the scene and used the advection functionality.

2.5 Assignment Implement mean curvature flow

In OperatorMeanCurvatureFlow.h an operator that computes the mean curvature flow was implemented. When used, the mean curvature flow generates a smoother surface. Step on was to compute the timestep.

$$\Delta t = \frac{\Delta x^2}{6\alpha} \tag{7}$$

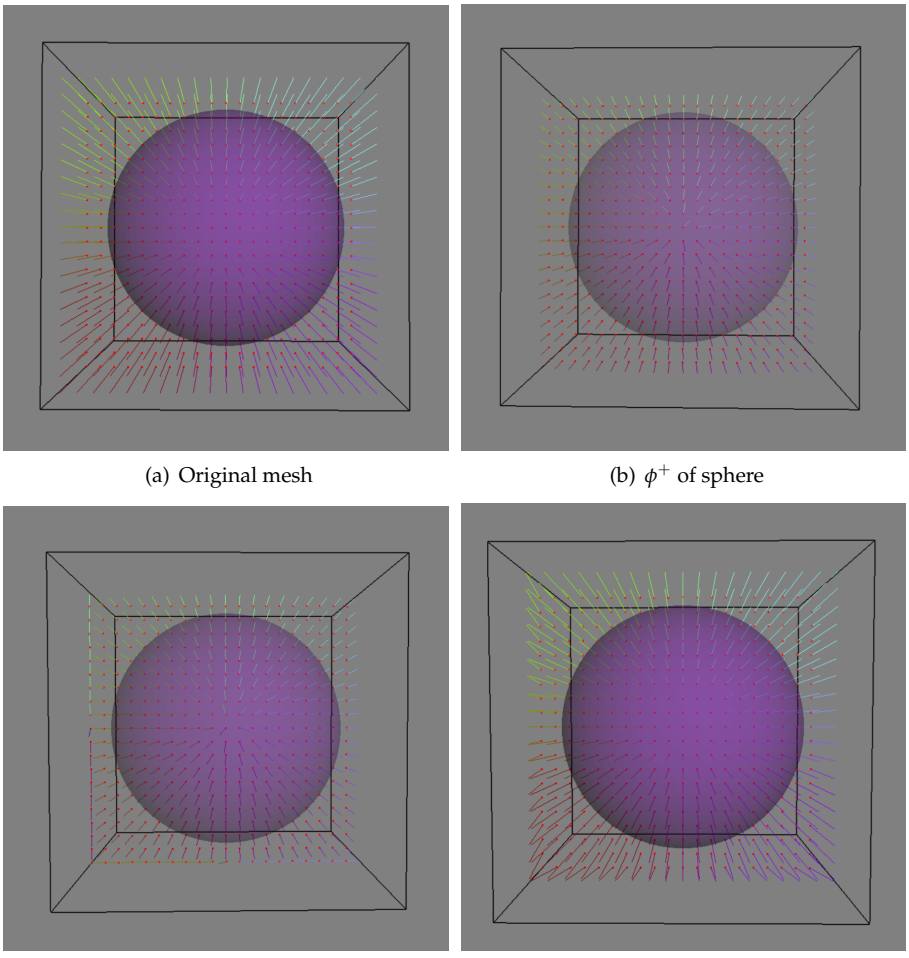
α is a small value and Δt is the desired timestep.

The next step was to implement the function Evaluate(). As in previous assignments, the grid coordinates (i, j, k) was the input values. All partial derivatives, Equation 1-5, for each axis was stored in local variables for later use. The the curvature, $\kappa_{i,j,k}$ was computed.

$$\begin{aligned} \kappa = \frac{1}{2}\Delta\frac{\Delta\phi}{|\Delta\phi|} = & \frac{\phi_x^2(\phi_{yy}+\phi_{zz}) - 2\phi_y\phi_z\phi_{yz}}{2(\phi_x^2+\phi_y^2+\phi_z^2)^{\frac{3}{2}}} \\ & + \frac{\phi_y^2(\phi_{xx}+\phi_{zz}) - 2\phi_y\phi_z\phi_{xz}}{2(\phi_x^2+\phi_y^2+\phi_z^2)^{\frac{3}{2}}} \\ & + \frac{\phi_z^2(\phi_{xx}+\phi_{yy}) - 2\phi_y\phi_z\phi_{xy}}{2(\phi_x^2+\phi_y^2+\phi_z^2)^{\frac{3}{2}}} \end{aligned} \tag{8}$$

Equation 8 is the curvature, κ, for the point (i, j, k). The norm of the gradient, $|\Delta\phi|_{i,j,k}$, was computed using a central difference scheme, Equation 3. Written in code, this simply becomes the length of the gradient. Lastly the rate of change $(\frac{\partial \phi_{i,j,k}}{\partial t})$ was calculated as the product between the scaling parameter, the curvature and the norm of the gradient of the level set. This value was returned from Evaluate(). To test the mean curvature flow a more complex mesh was used with the smoothing operator.

3 Results



(a) Original mesh

(b) ϕ^+ of sphere

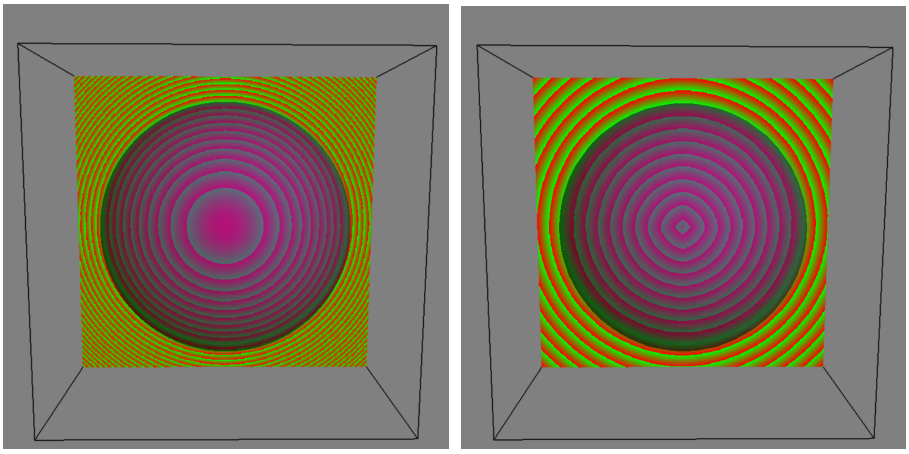
(c) ϕ^- of sphere

(d) $\phi^\pm$ of sphere

Figure 1
Result from assignment 2.1 showing the original mesh and its gradients, 1(a), and the different differentials computed, 1(b)-1(d). There is no reason for behind the two spheres that have a lower opacity.

The original mesh, a sphere, that was used in assignment 2.1 is visualized in Figure 1(a)

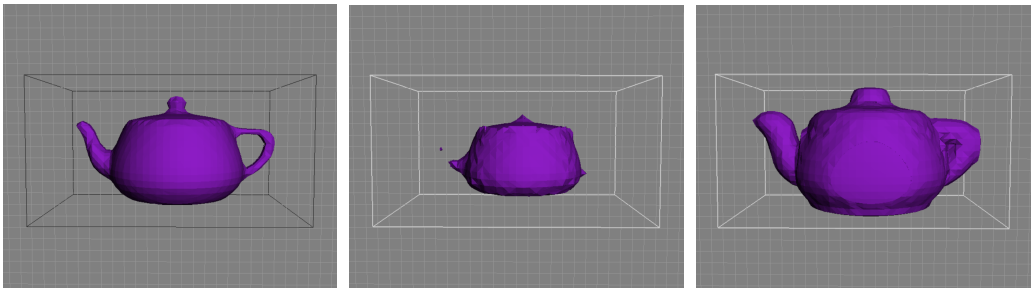
along with a plane showing the vectors of the gradient. In the other three images in Figure 1, a vector cut plane has been added that displays the gradients, or normals, for each differential. 1(b) shows the normal for ϕ^+, 1(c) for ϕ^- and 1(d) for $\phi_\pm$.



(a) Signed distance property of mesh (b) Re-initialized signed distance

Figure 2
Result from assignment 2.2 showing the original signed distance property without re-initialization, 2(a), and the result of the re-initialized signed distance property, 2(b). Signed distance is visualized with red-green cycles.

The signed distance property can be viewed in Figure 2, with the unmodified version in Figure 2(a) and the re-initialized version in 2(b). Colourmap visualized "Iso contour" where each red-green cycle represents a change in the level set function by 0.1. This visualizes all level sets and not just the zero level set (the border or surface of the mesh).

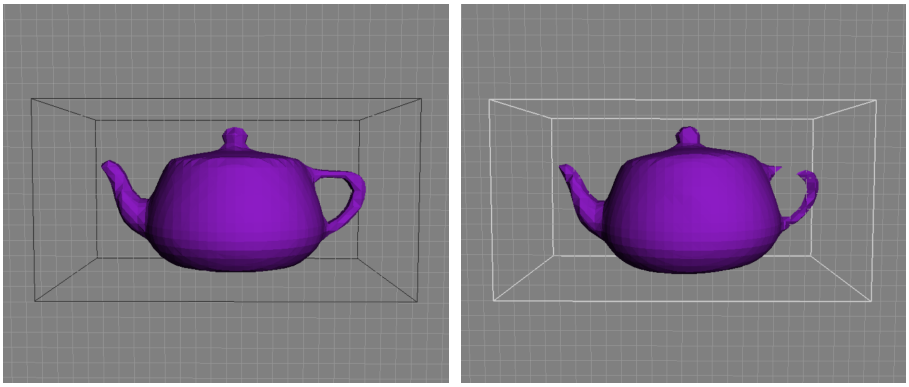


(a) Original mesh, a teapot (b) Dilated mesh (c) Eroded mesh

Figure 3
Result from assignment 2.3 showing the original mesh, a teapot, in Figure 3(a) to the left, a dilated teapot in the center 3(b) and an eroded teapot to the right 3(c).

Figure 3 shows the mesh, a teapot, used in the assignment 2.3. To the left is the original mesh 3(a), without any modifications. The result of the dilation operator can be seen in the center 3(b) and erosion to the right 3(c). In both cases the mesh was dilated or eroded 10 times.

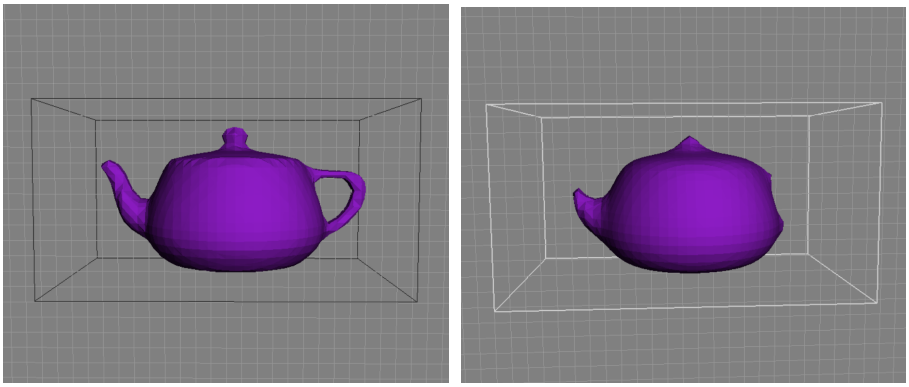
For advection the same mesh as dilation and erosion was used, see Figure 4(a). An advected



(a) Original mesh, a teapot (b) Advected mesh

Figure 4
Result from assignment 2.4 showing the original mesh, a teapot, in Figure 4(a) to the left and an advected teapot to the right 4(b).

version of the mesh is shown in Figure 4(b). The mesh was advected 10 times.



(a) Original mesh, a teapot (b) Smoothed mesh

Figure 5
Result from assignment 2.5 showing the original mesh, a teapot, in Figure 5(a) to the left and a smoothed teapot to the right 5(b).

Lastly was the mean curvature flow, which creates a smoothing effect on the mesh. Once again the same mesh as in assignment 2.3 was used, see Figure 5(a), to test the operator. Figure 5(b) shows the smoothed version of the mesh.

The original mesh and the modified versions are placed next to each other for easier comparisons.

4 Conclusion

The vector cur plane was primary used to test if the implemented code was working correctly. As can be seen in Figure 1(b)-1(d), the gradients are different since the schemes used are different,

bur for all the gradients points out from the center. The main difference between ϕ^+ and ϕ^-, 1(b) and 1(c), is the center. For ϕ^+ 1(b), the center gradient points upward and for ϕ^- it point downwards which corresponds respectively to the positive and negative part of an upwind scheme. In the central difference scheme 1(d), the center gradient point nowhere, it is equal to zero. This is how it should behave since the central difference scheme is a combination between the positive and negative upwind scheme.

In Figure 2 the colourmap that visualizes the "Iso contour", changes gradually for each re-initialization step. After each re-initialization the width of each red-green cycle moves towards the same width, which can clearly be seen in Figure 2(b) where the red-green cycles are almost the same width. If the level set function is properly re-initialized then the breadth of each red-green cycle should be approximately the same. The value of the level set function moves close and closer to 0.1. On a grid, if the gradient of the level set function is too small it should result in the location of the zero level set being sensitive to perturbation. On the other hand, if the gradient is too large with respect to the grid spacing, one loses accuracy in the interface representation. Another reason for re-initialization is to control numerical errors resulted in discretization of the level set PDE. The error is typically dependant on the gradient of the level set function.

Dilation, or an outward advection, should result in a larger mesh were holes have been filled in and thin lines or parts have become thicker. If the original mesh, Figure 3(a), and the dilated mesh, Figure 3(b), are compared some significant differences are to be noted. The dilated mesh is overall thicker, the hole in the teapots ear if filled in and the nose is very thick. This concurs with how dilation should behave, which mean that it was a success. There is room for improvements though. After the dilation the mesh became less smooth.

Erosion, which is an advection inward, should result in a smaller mesh where spiky and loose parts are removed. As can be seen in Figure 3(c), compared to the original mesh in Figure 3(a), the mesh have reduces in size and parts that sticks out have vanished to some degree. The erosion could be improved though to create a smoother version that does not have remains of eroded parts floating around. The resulting mesh, Figure 3(c), became very square and spiky with a part of the teapots nose floating in the front. The simplicity of the equations used made it easy to implement with an ok result.

An advected mesh both dilates and erodes. An example of that is shown in Figure 4. The advected mesh, Figure 4(b), are as smooth as the original mesh, Figure 4(a). The main difference between the original and the advected mesh is the details sticking out of the body. The nose of the teapot is fairly similar but the ear in the advected version have become fragmented. The performed advection most likely consists of a dilation followed by an erosion. In Figure 4(b) the original mesh have been advected 10 times. The result will vary on the number of times the advection is performed.

To easier see what the mean curvature flow does to a mesh a complex mesh was used, see Figure 5(a). Figure 5(b) shows a smooth version of the original mesh. The curves are smoother and both the lid and the nose of the teapot have changed shape. They are smoother but at the cost of a deformed shape. How the shape changes depends on the algorithm for the mean curvature flow. One thing that was a bit odd is that the whole ear of the teapot disappeared, and a crescent shape was removed from the teapots body. The ear could be missing because it was too thin and therefore completely removed during the computations. This does not explain the missing crescent part, which is most likely a effect of the ear being holy. A better smoothing operator could most likely fix these small flaws. All in all, the mean curvature flow algorithm could be improved but gives an ok result.

5 Lab partner and grade

This lab was done together with Rebecca Cedermalm, rebca973. The tasks for grade 3 and 4 was completed an therefore I think grade 4 would be a fair grade.