Final Project Report
Smoke simulation in box-splines spaces
CPSC 503

Estelle Altazin

Department of Computer Science
University of Calgary
ealtazin@cpsc.ucalgary.ca

Abstract

Your text here...

Key words: Your text here...
Your text here...
Your text here...
Your text here...
Your text here...
Your text here...
Your text here...
Your text here...
Your text here...
Your text here...

1 Introduction

1.1 Goal

What did we try to do?

The modeling and simulation of smoke is challenging and allows us to apprehend various mathematical methods specific to natural phenomenas or fluids simulation. The main goal of this project was to understand and implement all these methods in order to model and visualize smoke. This work was supposed to be done on both Cartesian and body-centered cubic (BCC) grids, so as to compare the results. The result is expected to be better on BCC grids.

Who would benefit?

Smoke simulation could be use for special effects, or for games for example. More generally, this work is a contribution to the computer graphics community.

1.2 Previous Work

What related work have other people done?

Fedkiw et al. [**?**] have worked on Cartesian grids and obtained good results of smoke simulation. They design a model for smoke and gases, using numerical methods that are both fast and efficient. Solving the incompressible Euler equations to compute smoke's velocity, they then introduce a vorticity confinement term to model rolling features in order to make the smoke more realistic. Their model uses a finite difference approximation for the divergence and Laplacian operators with the goal of making the solution accurate at the grid points. Alim [**?**] worked on smoke simulation and describe the mathematical methods and different experiments in his master thesis.

For the BCC part, we can rely on the work of Theul et al. [**?**] that clearly describes BCC grids and their assets in terms of accuracy and efficiency in computer graphics simulations. They explain how to construct and implement the BCC grids in the context of scalar date approximation and visualization.

Unser's work [**?**] provides us a general description of shift-invariant spaces, while Entezari et al. [**?**] work more specifically on interpolation on BCC grids. Finally, Alim [**?**] introduced data processing techniques on BCC grids, implementing divergence and the Poisson equation, ingredients that are essential for a numerical solution of the Euler equations.

When do previous approaches fail/succeed?

The main previous approach use finite difference for the advection part, and this approach does not allow to use large time steps. Our approach consists in tracing the streamline of every gridpoint given the velocity fields, it is much more stable and thus this allows to use whatever time step we want.

1.3 Approach

What approach did we try?

Our approach was based on five main steps for both Cartesian or BCC grids, implemented in Matlab. The first three steps concern solving the incompressible Euler equations of fluid flow to model smoke's velocity. We then need to be able to interpolate the values of a grid at any given point different than a gridpoint. This is an essential step to implement advection then. The second

step would be to implement advection so that our smoke particules follows the velocity field they are exposed to. This velocity field is defined by our force model, which will be described later in this part.

Finally, we need to make sure that the smoke's velocity is a divergence-free field to satisfy the incompressible Euler equations. We solve the Poisson equation with Neumann boudary condition to obtain an unique solution.

After soving the Euler equations of fluid flow, we can start to make a simulation : we need a grid, a source of smoke and forces to create a velocity field. At each time step, we will solve the Euler equations and update the velocity fields with the forces present.

The last part is rendering. We will use Matlab built-in function to render slices of the grid in 2D.

Under what circumstances do we think it should work well?

We think that the simulation will work well for grids with spacing between 32 and 128, for bigger grids, the computational time will be too big.

For the methods used, the best result should be obtained with cubic interpolation, Runge-Kutta advection and with vorticity confinement, given that these methods give the most accurate results. However, we will also implement other methods to be able to compare the results.

2 Methodology

2.1 What pieces had to be implemented to execute my approach?

First, we needed to implement the resolution of Euler equation of the fluid flow. To do this, we first of all need to understand how to use the code already existing for cubic interpolation. Therefore, we need to test it before using it during the advection part.

Then, we need to implement the advection of a grid given a velocity field, that is to say to implement a way to determine, for each gridpoint, where it was dt ago. We need to implement an integration scheme and we could use Euler integration or the Runge-Kutta method for example.

The last step in solving the fluid flow equations is to implement a Poisson solver. We will seek guidance from the work of Alim [**?**] to solve the Poisson equation in 3D.

Next, we need to implement a proper simulation, and thus we need to define a force model, as the one in the paper of Fedkiw et al. [**?**]. We will define different parameters in order to run different simulation easily and quickly.

The last step of rendering will not need much implementing work since we will use Matlab's function.

2.2 Interpolation

In order to implement the advection, we need to be able to determine, for any scalar grid, the value that this grid would take at any point whose coordinates are contained in the grid. So we need to interpolate the values of the closest gridpoints to figure out the value of the grid at the concerned point.

There is various order interpolation : linear and cubic for example. With linear interpolation, the error on a grid of size $N \times N \times N$ should be twice the error on a grid of size $2N \times 2N \times 2N$. Cubic interpolation allows to increase this ratio up to 16, so as soon as the grid size increases, the interpolation error decreases very fast.

For our use of these functions, the main difference will be that the linear interpolation will give a result much more blurred than the cubic interpolation, as we see on these pictures :

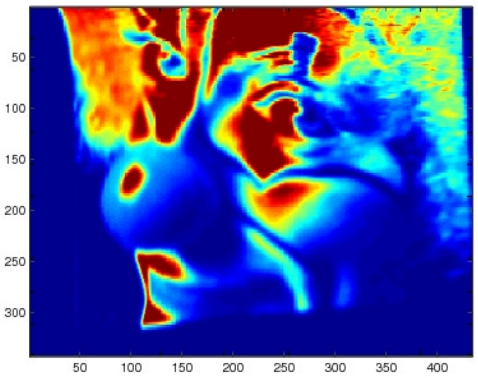

Figure 1: Clown image interpolated 25 times with linear interpolation

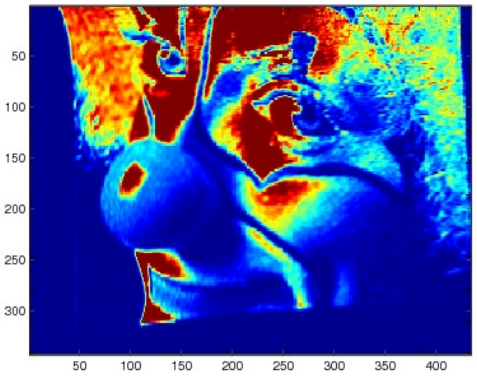

Figure 2: Clown image interpolated 25 times with cubic interpolation

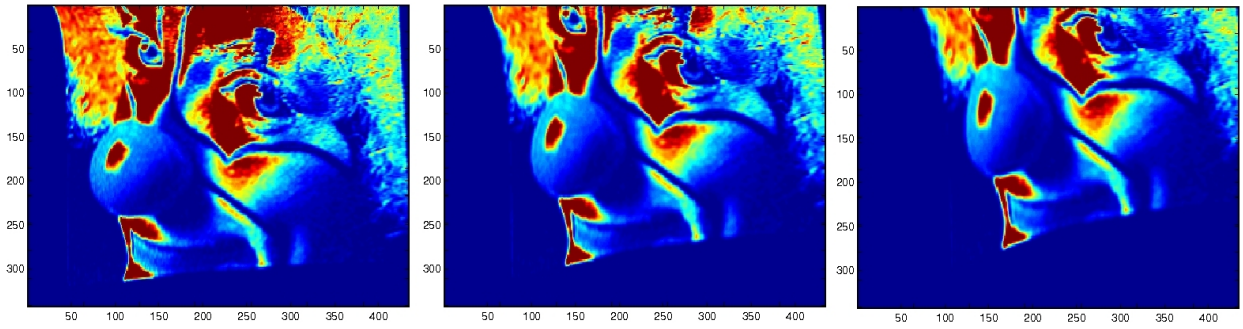

Figure 3: Advection at time steps 25, 50, 75 and 100

2.3 Advection

Were there several possible implementations?

The main part of this piece is to implement an integration scheme. Thus there is different possibilities of implementations, such as Euler integration, second order integration or the Runge-Kutta method.

If there were several possibilities, what were the advantages/disadvantages of each?

Euler integration is really easy to implement and require just one call to the interpolation function, thus this method would be very fast. Yet, the results would certainly be more blurred than with another method of higher order integration.

In contrast, the Runge-Kutta method will probably be much more accurate, but with 13 calls to the interpolation function, it will probably be much slower than the Euler integration.

Which implementation(s) did we do? Why?

We choose to implement both Euler and Runge-Kutta integration methods in order to have both quick and accurate results and be able to compare them.

What did we implement? – Include detailed descriptions

Advection is about determine the movement of the gridpoints of a given grid T depending on the velocity field. We want, for each gridpoint, to determine its streamline given the velocity field.

At each time step dt, we integrate the velocity field at each grid point, thus we have the points corresponding to the positions of the gridpoints dt ago. Then we interpolate T at these new points and update T with the interpolated values.

The only difference between the two integration methods rely on the way to compute the position of the gridpoints dt ago where to interpolate T.

Indeed, if X, Y and Z are the grids containing the coordinates, and V_x, V_y, V_z the grids containing the velocity fields, the new positions according to Euler integration would be :

$$\begin{cases} X_1 &= X - dt * V_x \\ Y_1 &= Y - dt * V_y \\ Z_1 &= Z - dt * V_z \end{cases}$$

For the Runge-Kutta method, we first need to compute the coefficients k_1, k_2, k_3 and k_4 for each component :

k_1 is the values of V_x, V_y, V_z at the coordinates X,Y,Z;

k_2 at the coordinates

$$\begin{cases} X + dt/2 * k_1(x) \\ Y + dt/2 * k_1(y) \\ Z + dt/2 * k_1(z) \end{cases}$$

k_3 at the coordinates

$$\begin{cases} X + dt/2 * k_2(x) \\ Y + dt/2 * k_2(y) \\ Z + dt/2 * k_2(z) \end{cases}$$

k_4 at the coordinates

$$\begin{cases} X + dt * k_3(x) \\ Y + dt * k_3(y) \\ Z + dt * k_3(z) \end{cases}$$

Then, the new positions where to interpolate T are given by :

$$\begin{cases} X_1 &= X + \frac{dt}{6}\left(k_1(x) + 2k_2(x) + 2k_3(x) + k_4(x)\right) \\ Y_1 &= Y + \frac{dt}{6}\left(k_1(y) + 2k_2(y) + 2k_3(y) + k_4(y)\right) \\ Z_1 &= Z + \frac{dt}{6}\left(k_1(z) + 2k_2(z) + 2k_3(z) + k_4(z)\right) \end{cases}$$

Finally, the last step is the same for both Euler and Runge-Kutta methods : interpolate the grid T at the

points of coordinates X_1, Y_1, and Z_1, and update T with these new values.

The pictures 3 is a example of the clown image advected with the Runge-Kutta integration, with a velocity field defined as

$$\begin{cases} V_x &= Y \\ V_y &= -X \\ V_z &= 0 \end{cases}$$

2.4 Poisson solver

The last step to solve the incompressible Euler equations is to transform the velocity field ω into a divergence-free velocity field $\mathbf{u}$ by solving the Poisson equation :

$$\nabla^2 p = -\nabla \cdot \omega$$

And then, $\mathbf{u}$ is given by

$$\mathbf{u} = \omega - \nabla p$$

We want to simulate smoke evolving in a box, we will thus use the Neumann boundary condition

$$\mathbf{u} \cdot \hat{\mathbf{n}} = 0$$

As explain by Alim[**?**], this condition prevent smoke to move orthogonal to the walls.

Were there several possible implementations?

With our domain as a grid, the Poisson problem is a system of linear equation. There are many methods to solve it [**?**, **?**], some are direct and some other iterative. For example the conjugate gradient method is iterative and refine a solution at each iteration step until some convergence criterion is satisfy. On the other side, a direct method get an exact solution of the problem in one iteration.

The conjugate-gradient method is practical because it works on every type of domains, though it is often quite expensive. But our domain is rectangular and simple, and thus we will use a rapid fast Fourier transform (FFT) solver.

What did we implement? – Include detailed descriptions

We seek guidance from the work of Alim [**?**] to implement the FFT method in 3D : We first compute the 3D discrete cosine transform $\hat{\omega}$ of the velocity field ω using an existing function. Then, we divide $\hat{\omega}$ by the sum of the eigenvalues of the matrix describing the second-order central diffrence approximation te the Poisson equation. These eigenvalues are defined as :

$$\begin{cases} \lambda_x &= 2 - 2cos(\frac{\pi(j-1)}{M}) \;\; j \in 1..M \\ \lambda_y &= 2 - 2cos(\frac{\pi(j-1)}{N}) \;\; j \in 1..N \\ \lambda_z &= 2 - 2cos(\frac{\pi(j-1)}{P}) \;\; j \in 1..P \end{cases}$$

Where M, N and P are the dimensions of X, Y and Z. Finally, we compute the inverse cosine transform of

$$\frac{\hat{f}}{\lambda_x + \lambda_y + \lambda_z}$$

The result is p, and we then deduce $\mathbf{u}$ easily from it.

2.5 Simulation

To simulate smoke, we need to define the meshgrids X, Y and Z for the coordinates, and two grids `density` and `temp`. The first one, `density`, represents the density of smoke in the grid, while `temp` represents the temperature at each gridpoint.

We create a source of smoke : a sphere at the bottom of the grid where the density is set to 100 and the temperature to 100 as well, these two grids are set to 0 everywhere else. We will update this source at every timestep in order to have a continuous flow of smoke. We then need a velocity field : initially there will be no velocity along the X and Y axis, and an inital value V_0 for the smoke's velocity along Z. The velocity field will then be update at every time step according to the force model.

Force model

Our force model relies on three forces, as explained in the work of Fedkiw et al. [**?**] : the buoyancy, the temperature effect and vorticity confinement.

The buoyancy force describes the natural tendance of smoke to rise :

$$f_{buoy} = \alpha * density * \mathbf{z}$$

The temperature effect depends on the difference of temperature between each gridpoint and the ambient temperature, which we have set at $T_{amb} = 25$:

$$f_{temp} = \beta(temp - T_{amb})\mathbf{z}$$

Finally, to make the result more realistic, we add a term of vorticity confinement, as explain in Alim and in Fedkiw [**?**, **?**]. Vorticity is defined as the curl of the velocity field and describes the rotational behaviour of the smoke. That is what cause the swirls and curls of the smoke. Vorticity confinement has been developed by Fedkiw et al. [**?**], it locates the area where small vorticity features

should be added. The vorticity of a incompressible flow $\mathbf{u}$ is :

$$\omega = \nabla \times \mathbf{u}$$

We then computed normalized vorticity location vectors $\mathbf{N}$, they point from lower to higher vorticity concentrations :

$$\mathbf{N} = \frac{\mu}{\|\mu\|} \qquad \mu = \nabla \|\omega\|$$

And then the vorticity confinement force is computed as :

$$f_{conf} = \epsilon h (N \times \omega)$$

Where h is the spacing of the grid and ϵ is a positive parameter used to control the amout of confinement to add.

3 Results

3.1 How did we measure success?

Your text here...

3.2 What experiments did we execute?

Your text here...

3.3 Provide quantitative results

Your text here...

3.4 What do my results indicate?

Your text here...

4 Discussion

4.1 Overall, is the approach we took promising?

Your text here...

4.2 What different approach or variant of this approach is better?

Your text here...

4.3 What follow-up work should be done next?

Your text here...

4.4 What did we learn by doing this project?

Your text here...

5 Conclusion

Your text here...

Acknowledgements

Your text here...