Project CUDC Proposal

OUNAN DING
Student ID: 861194909
oding001@ucr.edu

Contents

1 Abstract **1**

2 Technique Overview **2**

3 Requirements **2**
3.1 User Inputs 2
3.2 Host-Device Data Transfer 3
3.3 Function Evaluation 3
3.4 Intersection Detection 3
3.5 Hermite Data Generation 4
3.6 Quadratic Error Function 4
3.7 Mesh Topology Construction 4
3.8 Visualization 4
3.9 The Baseline and Profiling 4

4 Outputs **5**

References **6**

1 Abstract

In this document we propose a CUDA based Dual Contouring(CUDC) project. The computation model of dual contouring will be studied, then the data and computation parallelism will be exploited for a CUDA based implementation. A CPU based reference implementation will be provided as well. The profiling and benchmarking will be taken on these two implementations. Finally a conclusion will be reached based on our implementation, profiling, and analysis.

2 Technique Overview

Dual contouring is a technique that converts a implicit function [Wik15a] to mesh, which is a collection of vertices, edges, faces [Wik15b]. The general theoretical foundation of this project is outlined by [JLSW02]. We will give an overview of it here.

Given an implicit function like $x^2+y^2+z^2 = c$, we can use dual contouring to construct the surface mesh at the level set c, that is, the surface of a 3D sphere.

Concretely, in dual contouring, a function $f(x) = x^2 + y^2 + z^2$ is evaluated at discrete grid vertices in a portion of space. And for each vertex at (x_i, y_i, z_i), we can determine if it belongs to the interior or the exterior of geometry based on the sign of $f(x_i, y_i, z_i)$. After that we can use this information to construct mesh for the geometry.

In some previous research, such as marching cube [LC87], the sharp features are not preserved in constructed mesh. Whereas in dual contouring, we extract the Hermite data from the original implicit representation, and construct Quadratic Error Functions(QEF) as a more accurate model of features in original geometry.

By solving the QEFs in the grid we can get the mesh vertices. Finally we connect the adjacent vertices to build the topology of mesh.

I have already had a working example of 3D dual contouring in Python [Din15]. The running result is shown in 1, where a 3D heart function $(2x^2 + y^2 + z^2 - 1)^3 - x^2z^3/10 - y^2z^3 = 0$ is contoured.

By using CUDA, we expect a performance boost in this application.

3 Requirements

3.1 User Inputs

Users are expected to input a string of function definition ($\mathbb{R}^3 \to \mathbb{R}$ in 3D case, or $\mathbb{R}^2 \to \mathbb{R}$ in 2D case) to our application. Then our application will parse users' input and generate a kernel function at the runtime.

However, the generating of kernel function at runtime requires NVRTC, which is introduced with NVIDIA CUDA toolkit version 7.0. In our storm server the version of installed CUDA toolkit is 5.0. We work around this by hard coding the function definition in the source code.

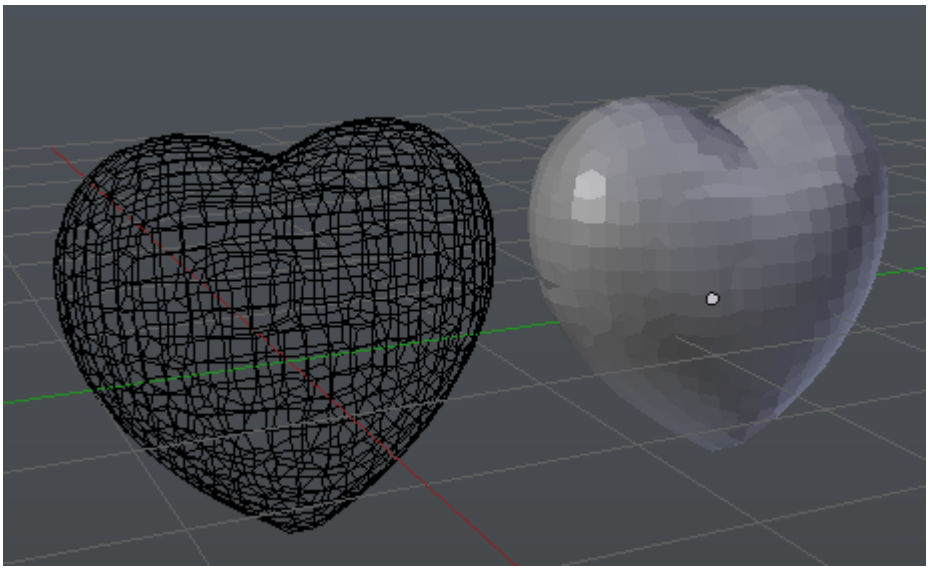

Figure 1: A working example of 3D dual contouring from [Din15]

3.2 Host-Device Data Transfer

Every cell in the computation grid may output a generated mesh vertex. We need to allocate the memory for them. Since only surface of the geometry will be contoured, we expect a space complexity of $\mathcal{O}(n^2)$ (n is the dimension of one side of the computation grid) in 3D case, or $\mathcal{O}(n)$ in 2D case.

3.3 Function Evaluation

Since the computation of input function at grid vertices is completely independent, we can utilize the CUDA for optimal performance. This should easily outperform a CPU implementation.

3.4 Intersection Detection

We need to check the 8 vertices(or 4 vertices in 2D case) to determine if an edge intersects with the geometry surface. This requires irregular memory access on the grid. It will be a challenge to achieve the maximal performance.

Besides, only a portion of the cells in the grid may intersect with the geometry. Hence a stream compaction is required here.

3.5 Hermite Data Generation

In this stage we need to compute the exact intersection point and the surface normal(i.e. the Hermite data) for each edge which exhibits an intersection.

The surface normal is computed by a finite differencing over the implicit function. And it can be fully parallelized due to the independence in computation.

However the exact interaction position is computed by a bisect search. This may cause branch(divergence).

3.6 Quadratic Error Function

To solve a QEF we need to perform a pseudo-inverse. Usually this is done through SVD or QR decomposition. We want a coherent algorithm here. This is the place where the majority work goes.

3.7 Mesh Topology Construction

The mesh topology construction usually is not a bottleneck. And this work does not fit the SIMT programming model very well. So I am thinking to implement it in host/CPU side in our first try.

However, for some applications such as real time rendering, the generated mesh data will be sent to the GPU rasterization pipeline for rendering. Thus the overhead of an extra data transfer from GPU to CPU cannot be ignored. We will optimize this in later phase of this project.

3.8 Visualization

After we fetch the result data from device, we need a program to visualize it for correctness. This can be done by using a 3D modeling software or an ad-hoc renderer using graphics API such as Direct3D or OpenGL.

3.9 The Baseline and Profiling

We also provide a CPU implementation as our reference and baseline during the benchmark. The CUDA implementation will be profiled and analyzed to reach our final conclusion.

4 Outputs

A CPU implementation will be provided as reference and baseline; a CUDA implementation will be provided, optimized, profiled, and analyzed; a report on the project process and performance analysis will be written.

References

[Din15] Ounan Ding. Fix mesh grid indexing in 3d dual contouring. `http://thebusytypist.github.io/learnblenderdev-site/2015/07/25/fix-meshgrid-indexing-in-3d-dual-contouring.html`, 2015. [Online; accessed 11-October-2015].

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Transactions on Graphics (TOG)*, 21(3):339–346, 2002.

[LC87] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[Wik15a] Wikipedia. Implicit function — wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Implicit_function&oldid=684709972`, 2015. [Online; accessed 11-October-2015].

[Wik15b] Wikipedia. Polygon mesh — wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Polygon_mesh&oldid=672906135`, 2015. [Online; accessed 11-October-2015].