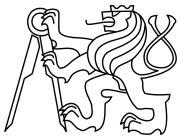
CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF INFORMATION TECHNOLOGY



ASSIGNMENT OF BACHELOR'S THESIS

Title:	Multi-level, Parallel Algorithms for Shape Interpolation on Modern Architectures
Student:	Šimon Hrabec
Supervisor:	Patrick Sanan, Ph.D.
Study Programme:	Informatics
Study Branch:	Computer Science
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2017/18

Instructions

Interpolation between shapes is a common task in geometry processing, fundamentally involving non-trivial geometric considerations. Aiming towards real-time, robust applications, we investigate how state-of-the-art methods may be accelerated and made robust with the introduction of multi-level methods and modern parallel and hybrid computer architectures.
1. Get familiar with the sub-field of shape interpolation within the field of geometry processing.
2. In C/C++, implement a variant of a shape-interpolation method from the literature and assess its performance.
3. Produce a prototype interactive application relying on this implementation.
4. Implement a multi-level extension of the method and assess its performance in terms of
i) scalability of time to solution for a family of relevant interpolation problems,
ii) robustness to mesh complexity.
5. Implement one of the computational kernels for execution on a GPU.

References

Will be provided by the supervisor.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

prof. Ing. Pavel Tvrdík, CSc.
Dean

Prague October 20, 2016

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical computer science

Bachelor's thesis

Multi-level, Parallel Algorithms for Shape Interpolation on Modern Architectures

Šimon Hrabec

Supervisor: Dr. Patrick Sanan

17th February 2017

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on 17th February 2017 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Hrabec, Šimon. *Multi-level, Parallel Algorithms for Shape Interpolation on Modern Architectures.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2017.

Abstrakt

Interpolace tvarů se zabývá problémem, v němž se hledá sekvence tvarů, které aproximují v čase pravděpodobný pohyb objektu. Jadna z nedávných metod je v práci analyzována, popsána a implementována v jazyce Matlab a s využitím numerických knihoven i v jazycích C++ a CUDA za účelem využití vysokého výpočetního výkonu GPU pro paralelní výpočty

Klíčová slova Tvar, optimizace, interpolace, paralelní algoritmy, CUDA, numerické metody

Abstract

The problem of shape interpolation is to find a sequence of meshes that approximates in time the likely movement of the represented object. A recent method is analysed, described and implemented in Matlab and with the help of numerical libraries in C++ and CUDA exploiting high performance of GPU devices for parallel computations.

Keywords Shape, Mesh, Optimisation, Interpolation, Parallel Algorithms, CUDA, Numerical Methods

Contents

List of Figures

Introduction

With the rapid progress during last few decades in the area of computer hardware, devices with high computational power has become widespread and hand in hand with that goes an increased demand for visually appealing graphics, including 3D graphics. In such area, objects are commonly represented as a set of polygons, formed by a connected network of points, forming a mesh. Despite the exponential (as it has been predicted by Moore and his law) growth of performance efficient algorithms are still required... With the increasing requirements for high quality graphics ... Shape interpolation is a problem where a sequence of shapes - represented as meshes - are given and intermediate ones are desired to obtain.

Chapter 1

Motivation

For a long time there has been an exponential growth of performance where frequency increase has played a major role. However with the exponential growth of frequency the power consumption grew in similar manner. The unsuccessful NetBurst architecture of INTEL showed us that forever going frequency increase is not an option and during the last decade we could observe a stall in the frequencies of CPUs [1]. This phenomenon has lead to a paradigm shift in the area of computing. The limitation set by the infeasible frequency cap has been overcome by supplying multiple computational cores, allowing multiple things to be computed at the same time. Intel reported that lowering a frequency of a core by 20 percent saved about 50 percent of the energy. The aspect of power consumption is empathised in HPC, where the power required to run such computer can be over years higher than the computer itself [2] and also in the area of mobile computing, where power supply is limited.

But having a different paradigm at the side of hardware requires a similar change in the software that is run on such architecture. In last decade GPGPU computing has become a popular option - using GPU as a highly parallel SIMD device [3]. CPUs are designed in a way to have low latency and for execution of various code due to which a significant area of the CPU is designated for control flow (branch prediction) and caching. GPUs used for GPGPU computing do not have sophisticated caching and branch prediction, but rather high amount of computational units, due to which they provide high throughput with higher delay, making them optimal for homogeneous computation - where one instruction is applied on multiple data (also called single instruction multiple data - SIMD). The different architecture requires problems of interests to be formulated in different, parallel manner. Parallel and scalable algorithms are of interest.

Chapter 2

Problem description

Objects represented as meshes move, rotate and scale in time and even some of its parts do. A shape interpolation is a problem where given two meshes (or possibly more) we want to have intermediate shape with a parameter t (1,0) specifying how where in between them (in terms of shape space) the interpolated shape should be $t * s1 + (1 - t) * s2$. In fact the shape interpolation problem consist of 2 parts - vertex corespondence problem, where we need mark which vertices correspond to each other between the two given meshes (we can also interpolate between two same meshes), which can be done in a user assisted manner or automatically. Second problem is vertex path problem - computing the location of the vertices of the intermediate shape. For shapes that are under the effect of translation, scale or shear a simple linear interpolation can be used, however such approach might lead to shape distortion for rotation due to its non-Euclidian nature. A linear interpolation for rotating object will result in shrinkage of the object, which is not desired. For shape interpolation algorithms we are interested how well the algorithm interpolates - how an object is distorted, how a human would perceive such shapes as natural - but also the computational speed. In the movie industry a high quality solution is desired, but in the case of mobile computing a real-time interpolation is desired despite its worse quality. There is a trade-off between the quality and processing time/number of shapes interpolated per second.

For the purpose of shape interpolation it is useful to have a metric how much the interpolated shape has been distorted. We can specify how much energy would have been required to deform s shape into a position in different ways - angle or edge length modification is an option. For the chosen metric we are trying to find a shape that minimizes the energy.

Chapter 3

Energy minimisation

Killian et el [4] proposed a metric, that gives and energy (an indicator how much energy would be required to deform the mesh into such shapes) for a mesh in a following way:

$$E(P) := \sum_{i=0}^{n} (\langle\langle X_i, X_i \rangle\rangle_{P_i} + \langle\langle X_i, X_i \rangle\rangle_{P_{i+1}}) \tag{3.1}$$

Given the following definitions of the riemannian metric,

$$\langle\langle X, Y \rangle\rangle_{M,\lambda} := \langle\langle X, Y \rangle\rangle_M + \lambda \langle\langle X, Y \rangle\rangle_M^{L^2} \tag{3.2}$$

the regularisation term,

$$\langle\langle X, Y \rangle\rangle_M^{L^2} := \sum_{p \in M} \langle X_P, Y_P \rangle A_P \tag{3.3}$$

the semi Riemannian part of the metric (as isometric as possible),

$$\langle\langle X, Y \rangle\rangle_M^{I} := \sum_{(p,q) \in M} \langle X_p - X_q, p - q \rangle \langle Y_p - Y_q, p - q \rangle \tag{3.4}$$

we got an equation:

$$\begin{aligned} E(P) := \sum_{i=0}^{n} (& \sum_{(p,q) \in P_i} \langle X_p - X_q, p - q \rangle^2 + \lambda \sum_{p \in P_i} \langle X_P, X_P \rangle A_P + \\ & \sum_{(p,q) \in P_{i+1}} \langle X_p - X_q, p - q \rangle^2 + \lambda \sum_{p \in P_{i+1}} \langle X_p, X_p \rangle A_P) \end{aligned} \tag{3.5}$$

The purpose of creating a metric is not only to define how the meshes are distorted in the interpolation problem, but also to be able to minimise it and make the resulting interpolated shapes as least deformed as possible (and the definition of "distorted" is given by the metric).

For the minimisation (optimisation) we will use an iterative method (improving the solution with every step), which will operate with the derivative of the metric.

The advantage of this metric is that it consist of a lot of summations, for which can use the derivative sum rule - that a derivative of sum is sum of derivatives. Meshes P_0 and P_n are the input meshes and their positions are fixed - we are looking only for the positions for the poses in between. Therefore for the gradient calculation we can drop the gradient assigned to these 2 meshes.

After expanding all sums in the equations it is possible to determine that a gradient of a mesh is determined only by itself and its neighboring meshes. To determine the derivative of mesh P_i we can drop all terms that do not contain vertices from the mesh (as their derivative is zero). We get:

$$d?d? = (\langle\langle X_{i-1}, X_{i-1}\rangle\rangle_{P_i} + \langle\langle X_i, X_i\rangle\rangle_{P_i} + \langle\langle X_i, X_i\rangle\rangle_{P_{i+1}} + \langle\langle X_{i+1}, X_{i+1}\rangle\rangle_{P_{i+1}})' \tag{3.6}$$

We can expand further to get:

$$\begin{aligned}\frac{E(x)}{\partial x_i} = &\sum_{(v,q)\in P_{i-1}} \langle X_p - X_q, p - q\rangle^2 + \lambda \sum_{p\in P_{i-1}} \langle X_P, X_P\rangle A_P + \\ &\sum_{(p,q)\in P_i} \langle X_p - X_q, p - q\rangle^2 + \lambda \sum_{p\in P_i} \langle X_P, X_P\rangle A_P + \\ &\sum_{(p,q)\in P_i} \langle X_p - X_q, p - q\rangle^2 + \lambda \sum_{p\in P_i} \langle X_P, X_P\rangle A_P + \\ &\sum_{(p,q)\in P_{i+1}} \langle X_p - X_q, p - q\rangle^2 + \lambda \sum_{p\in P_{i+1}} \langle X_p, X_p\rangle A_P\end{aligned} \tag{3.7}$$

sidenote-TODO equations above are trash and need to be rewritten, just a placeholder now

3.0.1 Derivation validity

It is desired to verify that the equations (derivations) are correct, otherwise we would not obtain a correct gradient and the minimisation could not work. For such purpose we compare the derivation (analytic, explicit formula) with an approximation via finite difference method.

$$\frac{E(x)}{\partial x_i} = \lim_{h\to 0} \frac{E(x + hv_i) - E(x)}{h} \approx \frac{E(x + \epsilon v_i) - E(x)}{\epsilon}$$

In the equation above v_i represents a vector of the same size as x, full of zeros except the ith position, which is one. ϵ can be set to square root of a machine precision.

All the equation used in the gradient generation procedure has been tested by looking at the difference between analytic gradient and the one obtained by finite difference method and its result has members smaller than $1e-7$ which lies below the numerical precision.

3.0.2 Minimisation method

We now have defined a metric - a function that that takes a series of meshes and gives a real number that denotes a level of deformation. We also have a way to compute the gradient (partial derivative for every vertex). Now wan use these in an iterative method that, in steps, modify the mesh in a way that its energy decreases. A simple iterative optimisation method is gradient descent. It is simple and requires only the function and its gradient, but the cost is slow rate of convergence. It works with an idea to take a direction in which a function decreases the most and make a step in this way:

$$X_{n+1} = X_n - \gamma \nabla f(x)$$

In this equation γ represents a step size. It can be set in advance (it is guaranteed that there exist enoug small γ that the method converges) or we can adjust it as we iterate - lower the step until $X_n < X_n - gamma \nabla f(x)$.

More sophisticated method is Netwon's method. It works with Hessian matrix, which is a matrix of second-order partial derivatives. It has faster rate of convergence but it requires more and it need more computation to apply the Hessian matrix. For such purpose quasi-Newton's method might come in use. Such method do not use Hessian matrix directly but only approximate it in some way. An example of a quasi-Newton's method is BFGS [5].

Conclusion

In the thesis I implemented a version of shape interpolation including a version in Matlab, then with help of a numerical library (numerical solvers) [6] I implemented the same interpolation in C++ and then provided a kernel for GPGPU computation in CUDA with the usage of L-BFGS library [7]. With that a graphical visualisation is included, allowing to view and examine a sequence of mesh objects or to perform a linear interpolation for comparison.

Bibliography

[1] Ross, P. E.: Why cpu frequency stalled. *IEEE Spectrum*, ročník 45, č. 4, 2008: s. 72–72.

[2] Balladini, J.; Suppi, R.; Rexachs, D.; aj.: Impact of parallel programming models and CPUs clock frequency on energy consumption of HPC systems. In *Computer Systems and Applications (AICCSA), 2011 9th IEEE/ACS International Conference on*, IEEE, 2011, s. 16–21.

[3] Rahim, M. N. I. A.; Mazalan, L.; Adnan, S. F. S.: Analysis on parallelism between CPU and GPGPU processing on cluster computing. In *Computer Applications and Industrial Electronics (ISCAIE), 2014 IEEE Symposium on*, IEEE, 2014, s. 203–207.

[4] Kilian, M.; Mitra, N. J.; Pottmann, H.: Geometric modeling in shape space. In *ACM Transactions on Graphics (TOG)*, ročník 26, ACM, 2007, str. 64.

[5] Fletcher, R.: A new approach to variable metric algorithms. *The computer journal*, ročník 13, č. 3, 1970: s. 317–322.

[6] Wieschollek, P.: CppNumericalSolvers. https://github.com/PatWie/CppNumericalSolvers, 2017.

[7] Wetzl, J.: CudaLBFGS. https://github.com/jwetzl/CudaLBFGS, 2016.

Appendix A

Acronyms

CUDA Compute Unified Device Architecture

CPU Central Processing Unit

GPU Graphics Processing Unit

GPGPU General-Purpose Computation on Graphics Processing Unit

SIMD Single instruction, multiple data

Appendix B

Contents of enclosed DVD

- `Code` ..... the directory with code
  - `Matlab` ..... the directory of source codes
    - `*.m` ..... Matlab source codes
  - `src` .... the directory with mesh visualisation application source codes
    - `*.h` ..... header files
    - `*.cpp` ..... .cpp files
  - `meshinterpolation.cu` ... CUDA code - kernel for mesh interpolation
- `text` ..... the directory with text of the thesis
  - `BP_Hrabec_Simon_2017.pdf` ..... the PDF version of the thesis
  - `BP_Hrabec_Simon_2017.tex` ..... the tex version of the thesis
  - `csn690.bst` ..... CSN citation standard
  - `cvut-logo-bw.pdf` ..... CTU logo
  - `FITthesis.cls` ..... Thesis template
  - `literature.bib` ..... Bibtex references
- `readme.txt` ..... the file with DVD contents description
- `Thesis_assignment.pdf` ..... the assignment of the bachelor thesis