DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

GPU-Accelerated Pressure Solver for Deep Learning

Martin Vincent Wepner

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

GPU-Accelerated Pressure Solver for Deep Learning

GPU-Beschleunigter Druck Löser für Deep Learning

Author:	Martin Vincent Wepner
Supervisor:	Prof. Dr. Nils Thürey
Advisor:	Philipp Holl
Submission Date:	15. March 2019

I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15. March 2019 Martin Vincent Wepner

Acknowledgments

todo

Abstract

TODO: Abstract

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**
- 1.1 PhiFlow 1
- 1.2 Problem 2
- 1.3 Pressure Solve and Conjugate Gradient 2
- 1.4 Challenges (Part of introduction or leave it out?) 2

2 Related Work **3**

3 Cuda Pressure Solver **4**

4 Results **5**

List of Figures **6**

List of Tables **7**

1 Introduction

The animation of fluids has become a key element of visual effects in the past years increasing the immersion of movies and computer games. Two main approaches have arisen to achieve realistic results: The Lagrangian method, which computes the dynamics of individual particles in respect to each other and the Eulerian method, that uses a fixed grid and computes how mass is moved within the grid. Existing solutions are often highly computational intense leading to trade-offs between realtime-capable simulations and accuracy, density or size and thus believability of the simulation. Newer approaches tackling this issue by leveraging the power of Deep Neural Networks and transforming it into a data-driven unsupervised learning problem. Until recently only Eulerian methods were fully differentiable making them suitable for deep-learning. However recent work by Schenk and Fox [SPnet] has shown a fully differentiable particle-based approach. Using simulation data of each timestep, Deep Neural Networks can be trained to mimic the concepts of the Navier-Stokes equation for fluid dynamics [Paper of Jonathan Tompson, Ken Perlin]. For larger simulations, the amount of physical memory needed to store the training data gets huge quickly. To avoid this, training data generation and the training process itself can be run in parallel: After every timestep is calculated the network's weights can be adjusted.

1.1 PhiFlow

PhiFlow is a toolkit written entirely in Python and currently under development by Philipp Holl at the TUM Chair of Computer Graphics and Visualization for solving and visualizing n-dimensional fluid simulations. It focuses to keep every simulation step differentiable which makes it suitable for backpropagation which is required to calculate the gradient of the loss function of Neural Networks to adjust the weight of its neurons. Being developed on top of NumPy/SciPy and TensorFlow it is not only capable to run on CPU, but also completely on GPU. TensorFlow simplifies the implementation of Machine Learning algorithms including the use of Neural Networks for predictions and classification. This makes PhiFlow a powerful tool to train Neural Networks on fluid simulations and also visualize it by its interactive GUI running in the browser.

1.2 Problem

This work proposes an implementation to solve the pressure equation - the most computational intense part of Eulerian fluid simulations. Being as efficient as possible is crucial for fast simulations. PhiFlow already comes with its own pressure solver running directly in TensorFlow. However, it is not clear/easy to understand how TensorFlow builds the dataflow graph exactly which may bring overhead to the computation. TensorFlow offers the possibility to write custom operations ("Custom ops") in C++ to tackle exactly this issue: Writing efficient code to solve a specific problem within the dataflow graph. Furthermore, the CUDA API by Nvidia makes it possible to write low-level C++ code that runs natively on GPUs. Combining Custom Ops and CUDA kernels lead to a solution that is both efficient and transparent and also compliant to TensorFlow's Tensors which is required to be used within PhiFlow. It can then be compared to the solution that comes with PhiFlow.

1.3 Pressure Solve and Conjugate Gradient

The Pressure Solve Problem is basically a system of linear equations whose solution is the exact pressure value of each cell in order to be compliant with the incompressibility constraint of fluids. Luckily the matrix of the linear system is symmetric and positive-definite which makes it suitable to be solved by the Conjugate Gradient method. Consisting of only some vector operations and Matrix-Vector multiplications, the cg-method is perfect to be solved in parallel on a GPU. Being a sparse matrix gives additional room for improvement, by not only keeping the required memory low but also decreasing the amount of data, that needs to be sent between the GPU's global memory to the on-chip local memory and vice versa. Also, no changes to the boundary conditions of the simulation mean no changes to the matrix itself, so it can be kept on GPU memory and don't have to be transferred between CPU and GPU, which is also pretty costly.

1.4 Challenges (Part of introduction or leave it out?)

The fact that the whole documentation for Custom TensorFlow operations consists of one inchoate single page made it hard to build the first working skeleton. Besides having a lot of code examples available, finding the one specific example to extract the information needed was hard, which led to lots of trial and error. [Also debugging on GPU...]

2 Related Work

TODO: Related Work

3 Cuda Pressure Solver

TODO: Cuda Pressure Solver

4 Results

TODO: Results

List of Figures

List of Tables