

Development and Analysis of an Interactive Physical Modeling Benchmark Set

Darlene Banta

Dept. of Computing Sciences
University of Scranton
Scranton, PA
bantad2@scranton.edu

Brenda Aldine

Dept. of Computing Sciences
University of Scranton
Scranton, PA
aldineb2@scranton.edu

Benjamin Bishop

Dept. of Computing Sciences
University of Scranton
Scranton, PA
bishop@cs.uofs.edu

Abstract—The goal of this work is to gain an understanding of the execution characteristics of certain interactive simulations. In order to achieve this, we collected a small benchmark set of physical modeling simulation applications¹. We then extended these to include a user-specified level of detail parameter. Lastly, we explored the behavior of these applications through the SimpleScalar processor simulator.

I. INTRODUCTION

Recently, there is a great deal of interest in interactive physical modeling applications. However, these simulations are generally limited to relatively simple scenes due to lack of computational performance.

Therefore, our research focuses on developing a benchmark set of interactive physical modeling applications and then ascertaining the optimal architecture resources for these applications. From this knowledge, one can better develop specialized hardware systems to run these physical modeling applications. To gather data regarding various hardware resources, we have executed the applications on an architecture simulator.

II. PRIOR WORK

There exist a variety of approaches to represent and animate physical models. This paper studies interactive simulations. Interactive simulations aim to provide stability and performance rather than accuracy. The applications included in the benchmark set we developed are a fluid modeling application, NaSt3DGP [1], a collision detection application,

SWIFT++² [2], and a deformable solid modeling application (cloth and cube) [3].

NaSt3DGP “is a parallel 3D flow solver for the incompressible, time-dependent Navier-Stokes equations” [1]. It was developed at the division of Scientific Computing and Numerical Simulation at the University of Bonn. NaSt3DGP can be used for simulations such as thermal convection, flow past an obstacle and free surface flow. Figure 1 provides screen shots from a NaSt3DGP simulation of flow past an obstacle [4].

SWIFT++ [2] was developed at the University of North Carolina at Chapel Hill as an extension of SWIFT. SWIFT++ implements collision detection, tolerance verification, approximate distance computation, exact distance computation and contact determination.

In [3], we developed a deformable solid object model with a gravity driven simulation to simulate both a cloth and a cube. The cloth and cube are composed of a series of mass points connected by deformable springs [3]. A cloth model was chosen to represent a simple two-dimensional mesh and a cube model was chosen to represent a three-dimensional mesh. In [3], results regarding the memory footprint and amount of floating point operations required for both the cloth and the cube were discussed. We have extended this research to include branch prediction and cache data.

We decided to use the SimpleScalar [5] architectural simulator developed by Todd Austin at the

¹Source code is available upon request.

²We are currently in the process of porting SWIFT++ to simple-scalar

University of Wisconsin in Madison. SimpleScalar provides both stability and flexibility. It has been in development since 1994 and has been widely used in computer architecture research.

III. CURRENT WORK

In our work we have built the SimpleScalar simulator for an i386-freebsd system and then modified and run the applications from the benchmark set on the SimpleScalar simulator.

Currently SWIFT++ is being ported to SimpleScalar and both NaSt3DGP and the cloth and cube have been successfully executed on the simulator. Thus, this paper presents the data collected regarding NaSt3DGP and the cloth and cube.

In order to understand the effect of increasing scene complexity on the execution behavior of the applications, we introduced a user-specified level of detail parameter for each application. In the case of NaSt3DGP, this scalable factor modified the resolution and the coordinates of the box. A simple NaSt3DGP simulation has fewer gridpoints and obstacle cells involved in the simulation. An increase in the amount of gridpoints and obstacle cells creates a more realistic simulation but may degrade performance. The cloth and cube application was developed with scalable complexity at runtime. The complexity factor, based on the number of subdivisions, varies the number of points and springs.

Because the computational complexity increases at varying rates between NaSt3DGP, the cloth, and the cube, we were able to collect a greater amount of data points for NaSt3DGP and the cloth than for the cube. We developed a complexity scale of 1-100 to provide a uniform measurement of complexity; 1 represents minimal complexity, 100 represents maximal complexity and all values in between are scaled accordingly.

To run the applications on the SimpleScalar simulator, they were cross-compiled using the Portable Instruction Set Architecture (PISA) SimpleScalar cross-compiler. NaSt3DGP execution includes both physical modeling calculations and saving these calculations to a file. We are interested solely in the calculation costs of the application. Therefore, the unnecessary file write commands were removed from NaSt3DGP. Additionally, NaSt3DGP requires

the math library libm.a. Therefore, we needed to build SimpleScalar's glibc-1.09. SimpleScalar's libraries do not include the OpenGL library; therefore all OpenGL calls were removed from the cloth and cube application.

Each application was run on SimpleScalar using sim-bpred to collect branch prediction data, sim-profile to find the instruction breakdown and sim-cache to gather cache and memory footprint information. Sim-bpred allows users to specify the branch predictor and predictor table size. The branch predictors available are not taken, taken, perfect, bimodal, 2-level adaptive and combined bimodal and 2-level adaptive. Within the 2-level adaptive predictor, the number of entries in the first and second level tables, the history size, and xor, which xors the history and the address in the second level table, can be modified. Therefore, using the 2-level adaptive predictor, global branch history such as GAg and GAp, local branch history such as PAg and PAp, and gshare can be simulated.

IV. RESULTS

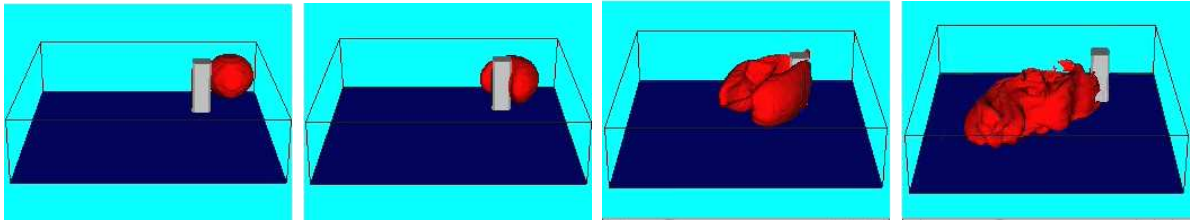
The results presented in this paper focus on branch prediction, instruction profiling, cache miss rates and memory footprints³.

Using sim-bpred, we executed applications from our benchmark set with bimodal, combined, and gshare branch predictors with table sizes from 1 to 4096 in powers of 2. However, the gshare predictor table size cannot be 1 since the table size is 2 to the W power where W is the size of the history table and W cannot equal zero. Therefore the gshare table size ranges from 2 to 4096.

The optimal branch predictor table size was the first important factor studied. Therefore, a representative complexity of each application was chosen. Then each application was executed on the simulator with a constant complexity for various branch predictors and table sizes.

For NaSt3DGP a complexity of 75 to study the optimal predictor table size was used. In Figure 2 the combined predictor behaves unexpectedly; the miss rate decreases from 8.17 percent for a table size of 1 to 6.184 percent for a table size of

³We are also working towards analyzing the maximum available parallelism in the applications.



(a) frame 0

(b) frame 1

(c) frame 2

(d) frame 3

Fig. 1. NaSt3DGP Simulation

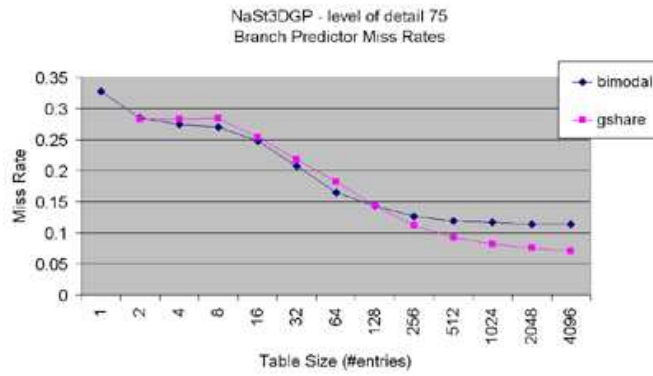
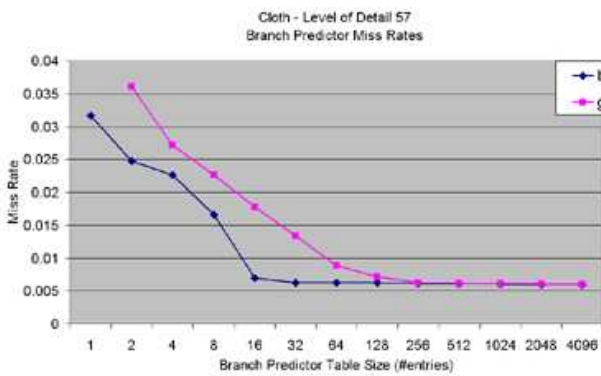
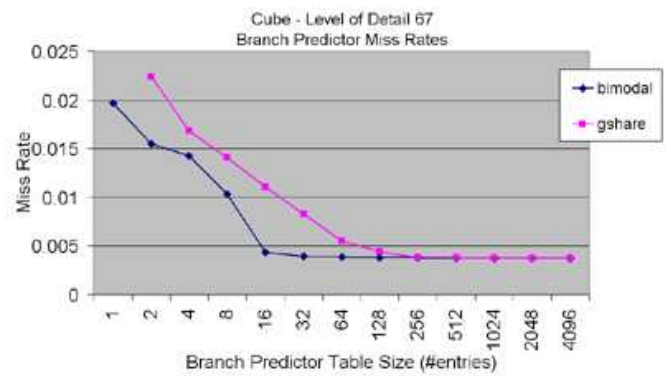


Fig. 2. NaSt3DGP Branch Prediction Increasing Table Size



(a) Cloth



(b) Cube

Fig. 3. Cloth and Cube Branch Prediction for Increasing Table Size

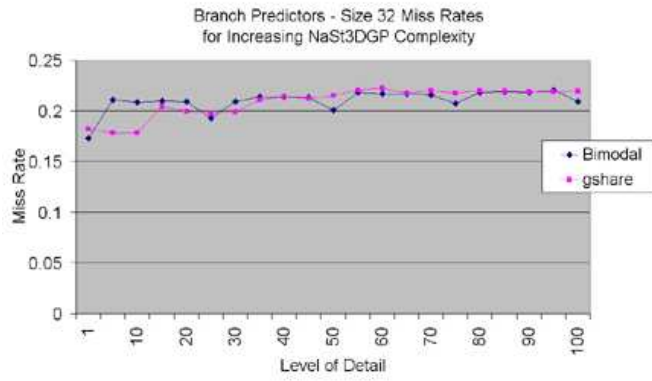
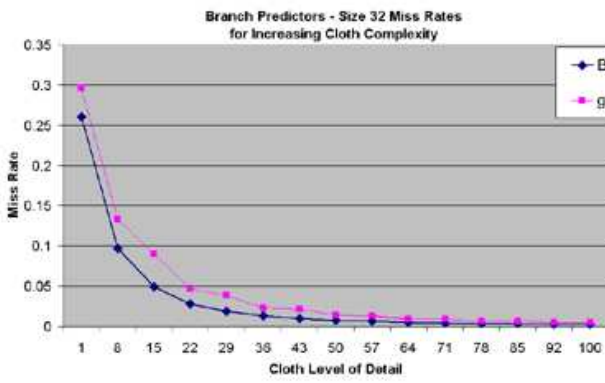
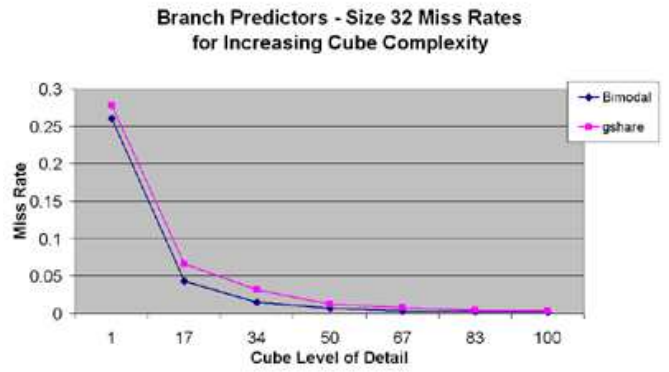


Fig. 4. Branch Prediction Increasing NaSt3DGP Complexity



(a) Cloth



(b) Cube

Fig. 5. Cloth and Cube Branch Prediction for Increasing Complexity

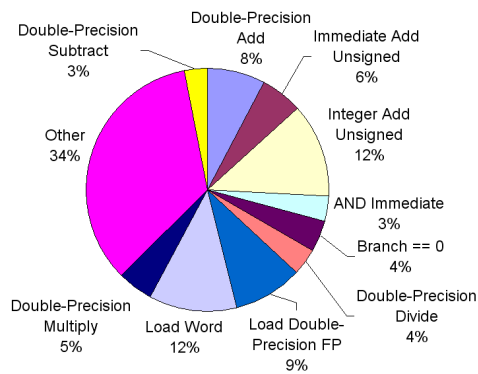


Fig. 6. NaSt3DGP Instruction Profile

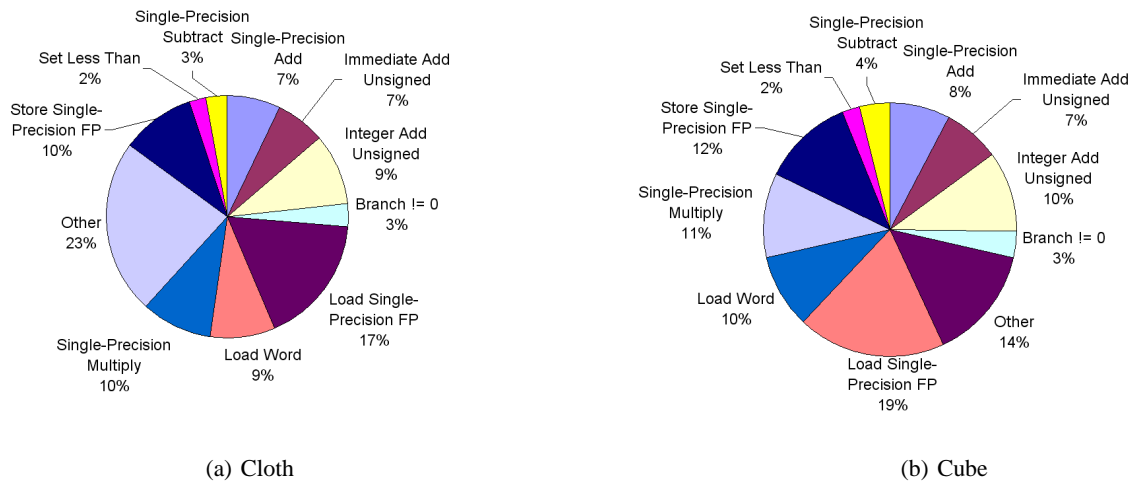


Fig. 7. Cloth and Cube Instruction Profile

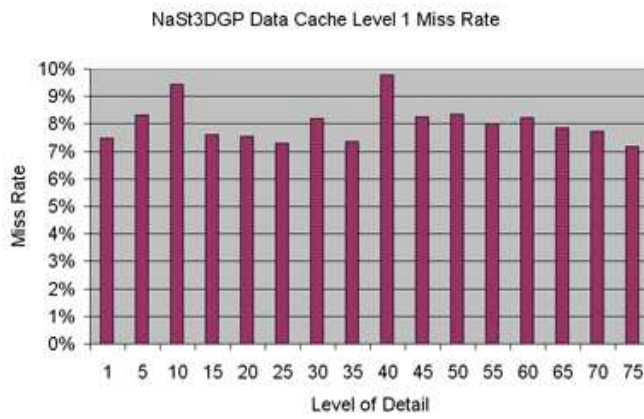


Fig. 8. Data Cache Level 1 Miss-Rate Increasing NaSt3DGP Complexity

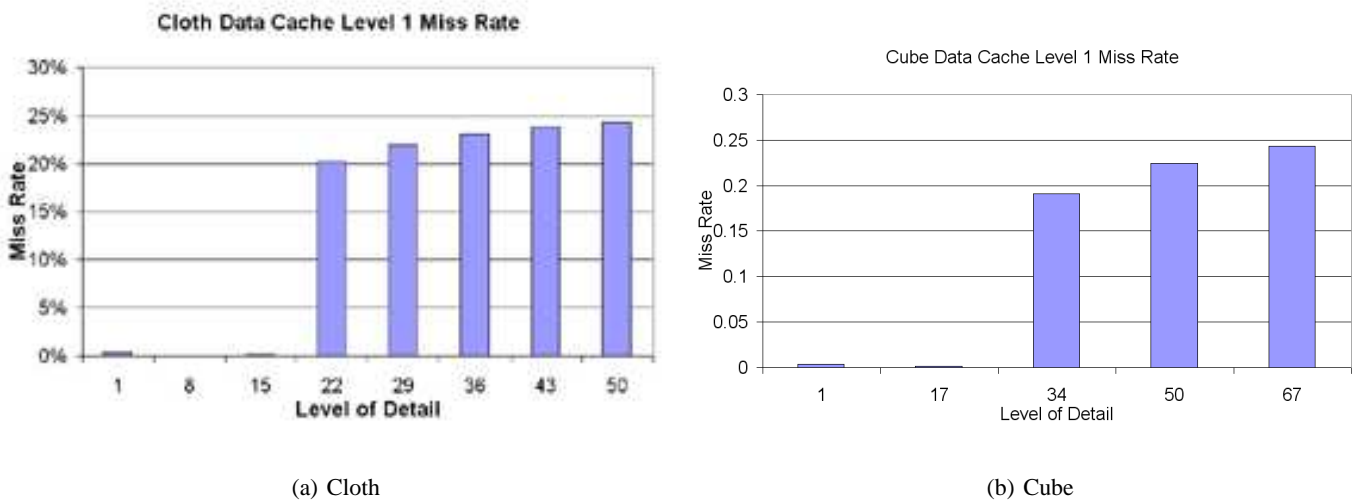


Fig. 9. Cloth and Cube Data Cache Level 1 Miss-Rate Increasing Complexity

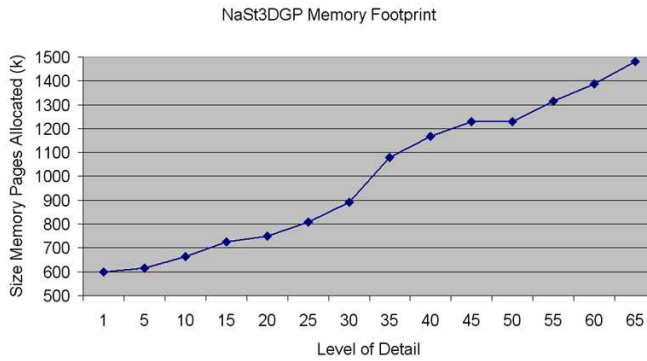


Fig. 10. Memory Footprint Increasing NaSt3DGP Complexity

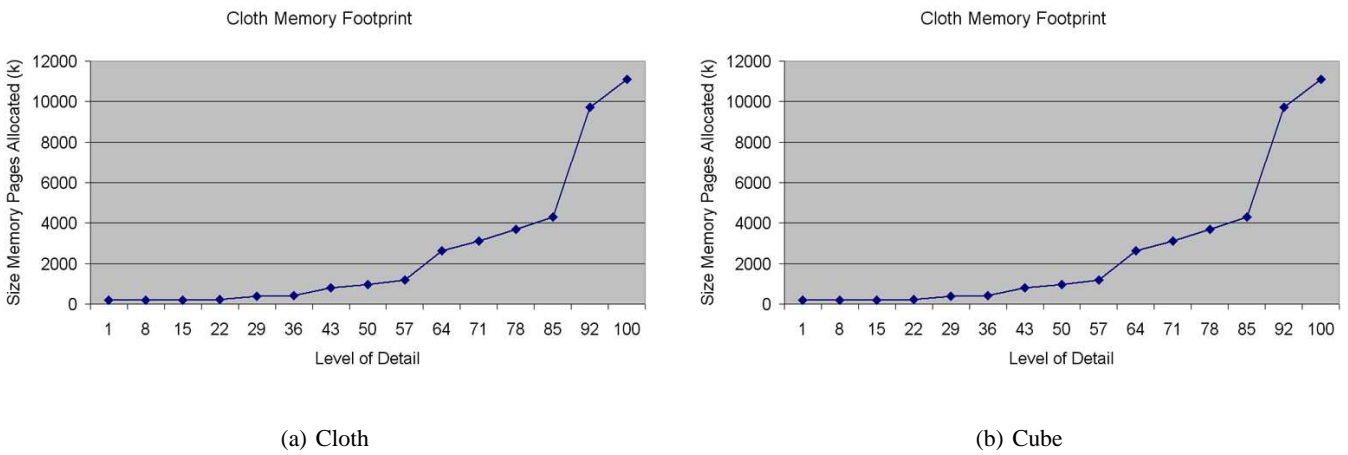


Fig. 11. Cloth and Cube Memory Footprint Increasing Cloth Complexity

4096. At this time we cannot explain why the miss rate is so low for a table size of 1. Based on Figure 2 the bimodal predictor has the greatest miss rate for predictor table sizes greater than 128 entries. However, for predictor table sizes smaller than 128 entries the bimodal predictor performs slightly better (approximately 2 percent) than the gshare predictor. The miss rate does not decrease below approximately 7 percent. We suspect that NaSt3DGP has data dependent branches; thus, the branch prediction ability is limited.

A complexity of 57 for the cloth simulation and of 67 for the cube was used. In Figure 3(a) and Figure 3(b) the results collected for the combined branch predictor seem anomalous; the miss rate changes by .002 from a table size of 1 to a table size of 4096 entries. We have verified that the simulation was correctly executed and that these are

the results produced by the simulator. As illustrated in Figure 3(a) and Figure 3(b), each predictor has a different table size for which the miss rate no longer improves. Therefore, each predictor has varying points of diminishing returns. Thus, for the cloth simulation, the optimal predictor table size for a bimodal predictor is 32 and for gshare is 256. For the cube simulation, the optimal predictor table size for a bimodal predictor is 32 and for gshare is 128. In contrast to NaSt3DGP, both the cloth and cube simulations have a table size where bimodal, gshare and combined perform similarly.

We then considered the miss rates for each application depending on various levels of detail within each application. Figure 4, Figure 5(a), and Figure 5(b) depict these results. Bimodal, combined and gshare predictors with predictor table size 32 were used.

As evidenced in Figure 4, the bimodal and gshare predictors behave differently as the complexity of NaSt3DGP increases. As the level of detail increases, at every 5th level of detail, the bimodal miss prediction rate decreases. In contrast, as the level of complexity increases, the gshare predictor reaches a steady miss prediction rate of approximately 22 percent.

In Figure 5(a) and Figure 5(b), as the level of detail increases for the cloth and cube applications the miss rate approaches zero; it remains between .1 percent and .5 percent with a level of detail of 100. We can thus conclude that the cloth and cube applications' branch prediction is rarely data dependent and easy to predict.

Another important result is the instruction profile. Figure 6 illustrates that NaSt3DGP uses double values rather than floats; we decided to leave the application as originally developed. We realize that our benchmark set ideally would contain interactive simulations; therefore, float values are expected over double values because the goal is stability over accuracy. As evidenced by Figures 7(a) and 7(b) the cloth and cube have analogous instruction profiles. Similar to NaSt3DGP, the percentage of double and integer arithmetic operations is similar.

The size of the level one data cache plays an important role in application execution. NaSt3DGP and the cloth and cube simulations were executed on sim-cache using the default cache parameters; the cache size remained constant as the complexity was increased. Figure 8 illustrates continuous fluctuation between approximately 7 percent and 10 percent in the miss rate of the level one data cache as the complexity is scaled for NaSt3DGP. However, in the cloth and cube simulations (Figure 9(a) and Figure 9(b)), the miss rate is extremely low initially but then increases to over 20 percent as the level of detail increases. With low levels of detail, the cloth and cube simulations do not produce enough data to fill the cache.

It is also interesting to study the memory footprint of these applications as the level of detail changes. Figure 10, Figure 11(a), and Figure 11(b) all illustrate that as the level of detail increases, the size of the memory footprint also increases. The size of the memory footprint increases almost linearly for NaSt3DGP; whereas the size of the

memory footprint for the cloth and the cube increase polynomially. Figure 11(a) illustrates unexpected behavior, an extreme increase in memory footprint between complexity level 57 and 64 and complexity level 85 and 92. These values were verified as the accurate values produced by the simulator; however we currently cannot explain this behavior.

V. CONCLUSION

We have made available a benchmark set for interactive simulation. It is possible to easily scale the level of detail in these simulations in order to simulate more complex scenes, and to understand how execution behavior changes as simulation complexity increases.

We have also produced detailed execution results. Interesting observations include the differences in branch predictability between the fluid dynamics simulation and deformable solid objects, the rate at which memory requirements increase with level of detail, differences in instruction execution behavior between the different applications, and the impact of level of detail on cache behavior.

REFERENCES

- [1] M. Griebel, T. Dornseifer and T. Neunhoffer "Numerical Simulation in Fluid Dynamics, a Practical Introduction," SIAM, Philadelphia, 1998.
- [2] S. Ehmann "Speedy Walking via Improved Feature Testing for Non-Convex Objects", <http://www.cs.unc.edu/~geom/SWIFT++/>.
- [3] S. Yardi, B. Bishop, T. Kelliher "An Analysis of Interactive Deformable Solid Object Modeling", *Conf. on Modeling, Simulation & Vis. Methods*, pages 95-99, 2005.
- [4] http://wissrech.iam.uni-bonn.de/research/projects/NaSt3DGP/Movies/mov_PastObstacle.mpg.
- [5] D. Burger, T. Austin "The SimpleScalar Tool Set, Version 2.0", http://simplescalar.com/docs/user_guide_v2.pdf, 1997.