Title:
Evaluation of the OptiX Ray Tracing Engine for Machining Simulation

Authors:
Marc Jachym, marc.jachym@ens-paris-saclay.fr, Paris-Saclay University
Sylvain Lavernhe, sylvain.lavernhe@ens-paris-saclay.fr, Paris-Saclay University
Christophe Tournier, christophe.tournier@ens-paris-saclay.fr, Paris-Saclay University
Charly Euzenat, charly.euzenat@ens-paris-saclay.fr, Paris-Saclay University
Pierre-Alain Boucard, pierre-alain.boucard@ens-paris-saclay.fr, Paris-Saclay University

Keywords:
Machining simulation, GPU computing, CUDA architecture, OPTIX, ray tracing.

DOI: 10.14733/cadconfP.2017.138-142

Introduction:

In molds and dies industry, simulation of machining process is mandatory to validate the tool path generated with the CAM software before launching the production of parts with very high added value. Indeed, machining operations including roughing, reworks and finishing are particularly long, especially for large size parts as for example in the automotive industry. Thus, the appearance of defects in the final stages of the process has a dramatic impact on manufacturing companies. CAM software editors therefore provides cutting simulation applications that allow to validate the paths from a macroscopic point of view, i.e. to test the presence of collisions. However, these simulations don't incorporate any features of the actual process likely to deteriorate the surface finish during machining operations. Finally, these simulations do not provide the accuracy required within a reasonable time or to select an area in which the user would like to have greater precision. On the other hand, high-performance simulation prototype software covering the preceding shortcomings are developed in laboratories, but requires significant computer resources.

Many methods have been published in the literature to perform machining simulations. Some of them are based on a partition of the space whether by lines [6], by voxels [5] or by planes [10], other are based on meshes [3]. Previous works have shown that it is possible to simulate the resulting geometry of the surface with Z- or N-buffer methods applied to a realistic description of both the tools and the machining path in a few minutes [7]. Simulation results are very close to experimental results but the simulated surfaces have an area of a few square millimeters with micrometer resolution. Therefore, to overcome the limits in terms of computing capacity, some works deal with the use of GPGPU (general-purpose computing on graphics processing units) and especially Nvidia GPU (graphics processing units) and CUDA (Compute Unified Device Architecture) technology in the field of manufacturing simulation [4,8]. In this context we have developed a software called "SIMSURF1" in order to simulate very quickly a selected machined area at different scales chosen by the user [1]. This tool is very fast and relies on GPU/Cuda technology or many CPU cores [2]. However, the development of such applications requires an extended and deep knowledge of these architectures. Low-level CUDA library has to be used and every aspect of the multi GPU-core architecture on which CUDA is based has to be managed, from the distribution of the parallel calculations on the cores to the memory exchanges between the CPU and the GPU.

This is why the use of an engine such as Optix, also offered by Nvidia, would facilitate the development of high-performance ray tracing applications [9]. Thus, the proposed "SIMSURF2" approach aims at taking advantage of the Optix engine in order to facilitate the writing of a machining simulation software based on the GPU parallel calculation platform. With Nvidia Optix, and with the specialized subset Optix Prime, gains are expected regarding the software development speed as well as regarding the optimization in the use of the CUDA architecture which, in turn, could accelerate the whole machining simulation process. We propose in this article to compare the performances of the OptiX engine with the developments previously achieved in "SIMSURF1" and updated here as an implementation for NVIDIA Tesla® K40 GPU. The rest of the paper is organized as follows: the computation algorithm and the low-level approach used in "SUMSURF1" is resumed in section 2, Optix engine is described in section 3 and section 4 is dedicated to the experimental investigations and benchmarking of both approaches.

Computation algorithm and CUDA architecture:

The computation algorithm relies on the Zbuffer method which consists in partitioning the space around the surface to be machined in a set of lines, which are equally distributed in the x-y plane and oriented along the z-axis. The machining simulation is carried out by computing the intersections between the lines and the tool along the tool path. The geometry of the tool is modeled by a triangular mesh including cutting edges, which allows to simulate the effect of the rotation of the tool on the surface topography. The tool path is either a 3-axis tool path with a fixed tool axis orientation or a 5-axis tool path with variable tool axis orientations. In order to simulate the material removal, all the intersections with a given line are compared and the lowest is registered. The complete simulation requires the computation of the intersections between the N lines (~1.e6) and the T triangles (~1.e4) of the tool mesh at every tool posture P (~1.e6) on the tool path. Thus simulations with 1.e16 potential intersections to compute are commonly encountered.

The strength of the CUDA programming model lies in its capability to achieve high performance through its massively parallel architecture. In order to achieve high throughput, the algorithm must be divided into a set of tasks with minimal dependencies. Tasks are mapped into lightweight threads, which are scheduled and executed concurrently on the GPU. The 32 threads within a same warp are always executed simultaneously; maximum performance is therefore achieved if all the 32 threads execute the same instruction at each cycle. Warps are themselves grouped into virtual entities called blocks; the set of all blocks forms the grid, representing the parallelization of the algorithm. Threads from the same block can be synchronized and are able to communicate efficiently using a fast on-chip memory, called shared memory, whereas threads from different blocks are executed independently and can only communicate through global (GDDR) memory of the GPU. The number of threads executed simultaneously can be two orders of magnitude larger than on a classical CPU architecture. As a consequence, task decomposition should be fine-grained opposed to the traditional coarse-grained approach for CPU parallelization.

The basic algorithm consists in determining whether there is an intersection between a line and a triangle associated to a tool posture. Given these three variables on which the algorithm iterates during the sequential computation, there are numerous possible combinations to affect threads and browse the set of lines, triangles and positions. Only one possibility is used hereafter which is the most appropriate for macro scale simulations [1]. Each thread is assigned to a position of the tool and applies the Z-buffer algorithm for every triangle of the tool mesh for this position. The granularity of tasks is high: if the number of triangles to be processed is large, each thread will run for a long time. If the computation time between threads is heterogeneous, some threads of a warp may no longer be active, and therefore the parallelism is lost. A thread may affect the cutting height of several lines so a line can be updated by multiple threads and global memory access conflicts appear. Atomic operations proposed by CUDA are then used to allow concurrent update of the height of the lines.

Optix ray tracing engine:

Nvidia Optix is an engine for ray tracing 3D-rendering. It allows the developer to concentrate on the objects in a scene whose geometry is defined by the algorithms for the ray-object intersections and on the behavior of the light when it encounters some material. Those elements are the entry points to the ray-tracing parallel calculation engine that executes on the CUDA architecture (Fig. 1). The Optix engine is based on acceleration structures, which are hierarchies of bounding boxes, to determine which of the scene areas are empty and do not need any calculation. Optix Prime is an Optix's subset which is dedicated to the high-speed calculation of intersections between rays and triangle meshes. There is no notion of material properties in Optix Prime and thus, it has nothing to do with optic rules and 3D object rendering. Rather, it provides a hopefully optimized way to use a hidden acceleration structure suited to triangle meshes and to perform a high-speed rays-triangles intersection on the underlying CUDA architecture. By hidden, we mean hidden to the software developer who is freed from researching methods for reducing the number of possible intersections that the GPU will have to calculate.

Within “SIMSURF1”, the software programmer has to devise by himself clever methods to determine empty areas in the scene in order to avoid that the GPU would have to calculate every possible intersection between any ray and any triangle. Within “SIMSURF2”, the programmer has to choose between different possibilities regarding acceleration structures and traversal methods, whether he has to manage static vs dynamic scenes or whether his objects are defined with geometric formulas or meshes. Optix Prime simplifies this greatly because the best possible choices, regarding Nvidia experience in acceleration structures and traversal algorithms, have been made for a static scene based on triangle meshes.

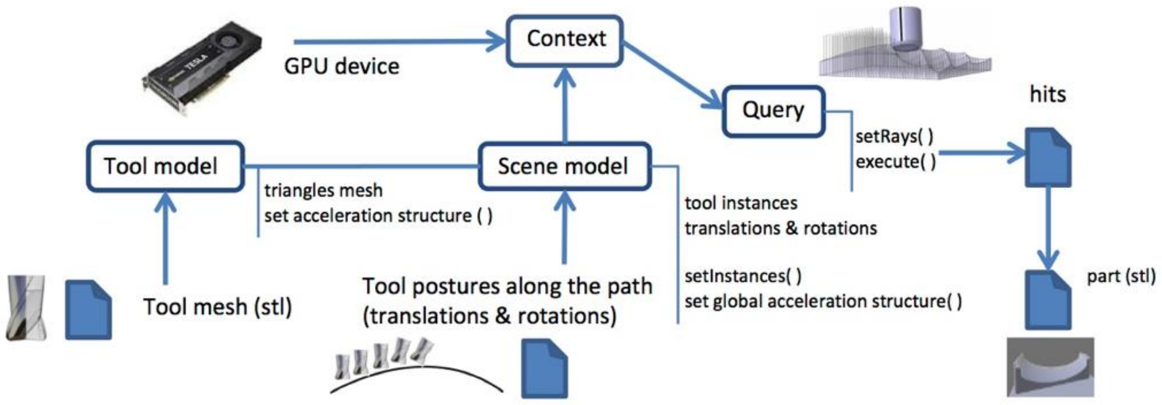

Fig. 1: OptiX engine process overview.

The calculation of the acceleration structures is the slowest stage of the process and, with previous Optix Prime versions, an acceleration structure has to be built at every step of the loop even if the geometry of the tool will not be changed but is simply moved along the planned path. This problem has been addressed with Optix Prime 3.9 which offers a new possibility called ‘instancing’. From a model object; in the sense of Object Oriented Programming; which associates a triangle mesh and its dedicated acceleration structure, ‘instancing’ composes complex scenes using existing triangle models. Then Optix Prime is able to create a global acceleration structure for the whole scene without duplicating the elementary models' description. The programmer has to create a memory structure to associate each instance of a model object in the scene with a transformation descriptor, i.e. a translation, a rotation or/and a scaling matrix. The fact that the basic model description is not duplicated in memory allows to process much bigger path buffers.

Experimental investigations:

Several test cases (fig. 2) have been investigated in 3 or 5-axis, in roughing (numerous air paths) and finishing (numerous postures) with variations in the number of tool postures on the tool path and triangles in the mesh. The Z-buffer is computed with a grid of 1024x1024 lines covering the X-Y trajectory range. NC simulations have been carried out with a Nvidia K40 GPU card with the following technical characteristics: 0.745 GHz, 2880 CUDA cores, 1.43 Tflops in double precision, 4.29 Tflops in single precision; bandwidth 288GB/s. Simulations have also been run on a CAD/CAM commercial software in order to establish a reference. Results are gathered in table 1.

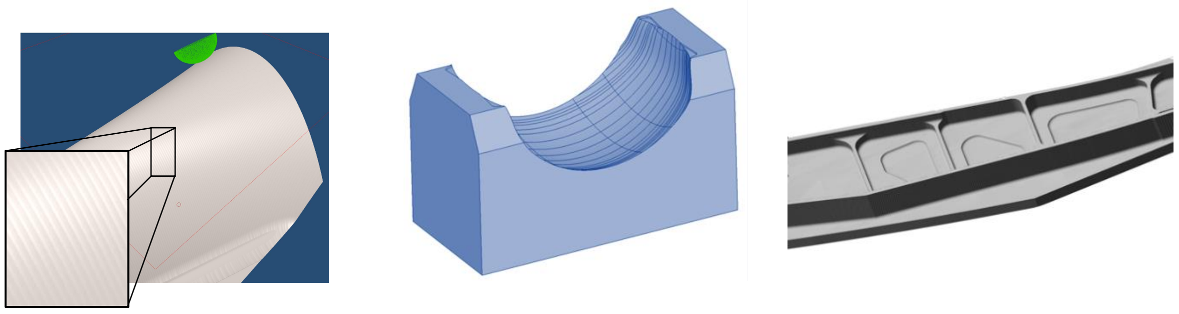
Fig. 2: Test parts (blade, ski mask mold and structural part).

In cases 1 and 3 which are 3-axis roughing simulations, 'SIMSURF2' is faster which is a positive outcome. In case 4 and 5, the results of 'SIMSURF2' are disappointing compared to other 3-axis cases. In case 4, the ratio between the machined surface area and the tool dimension is low. This implies that a lot of intersections will be computed between triangles and lines. Here 'SIMSURF1' is more than twice as fast as 'SIMSURF2'. In case 5 the ratio machined surface area / tool dimension is very high and the path is of about 27M cutter locations. We could therefore describe this simulation as macroscopic. In this case, the number of positions is almost 1000 times larger than the maximum number of threads that could theoretically be executed (30720) which is rarely the case due to memory limitations. This involves additional workload for the task scheduler CUDA as well as serialization of the computation. Two different tests have been carried out. One with our usual sampling density of 1024 x1024 rays and another one, more realistic, of 10 000 x 10 000 rays (Table 2). In both cases 'SIMSURF1' is more than five times faster than 'SIMSURF2'.

For 5-axis simulations including translations and rotations of the tool (case 2), 'SIMSURF1' is still 3 times faster than 'SIMSURF2'. It seems that the generation within Optix of the scene including rotation of the instances of the tool is time consuming.

Case	*Tool geom.*	*Triangles T*	*Postures P*	*Simsurf1 (ms)*	*Simsurf2 (ms)*	*Speed-Up*	*CAM (ms)*
1. Blade roughing	Torus	25904	62576	1166	812	0.7	25000
2. Blade finish. 5 axis	Sphere	12482	53667	1703	4925	2.95	245000
3. Mask roughing	Torus	25904	345848	2986	1272	0.42	380000
4. Mask finish.	Sphere	12482	3015072	11314	24671	2.5	450000
5. Aero finishing	Sphere	12482	27425026	3698	21865	5.88	2520000

Tab. 1: 32 bits computation times on test cases for a 1024x1024 Zbuffer with K40 GPU.

5. Aero finishing	Sphere	12482	27425026	30961	163565	5.28	2520000

Tab. 2: 32 bits computation times on test case for a 10000x10000 Zbuffer with K40 GPU.

Conclusions:

This paper presents a comparison of two ray-tracing GPU implementations for NC simulations. The first approach is based on the direct use of CUDA which requires a rather steep learning curve and expertise to achieve high performances. The second one is based on the Optix ray tracing engine which provides simple application programming interfaces to compute rendering of machining scenes. This approach is the simplest to implement and seems to be very competitive in 3-axis machining for macroscale simulations. However, 5-axis configurations and large machining scenes remain a problem for the Optix engine even if simulation times are quite reasonable. These results are very promising and will be confirmed by additional tests.

Acknowledgements:
We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research as well as the support of the Farman Institute (CNRS FR3311).

References:
[1] Abecassis, F.; Lavernhe, S.; Tournier, C.; Boucard, P-A.: Performance evaluation of CUDA programming for 5-axis machining multi-scale simulation, Computers in Industry, 71, 2015, 1-9. http://dx.doi.org/10.1016/j.compind.2015.02.007
[2] CUDA C Programming Guide, NVIDIA, 2012 http://developer.nvidia.com/cuda/
[3] He, W.; Bin, H.: Simulation model for CNC machining of sculptured surface allowing different levels of detail, The International Journal of Advanced Manufacturing Technology, 33, 2007, 1173-1179. http://doi.org/10.1007/s00170-006-0543-1
[4] Inui, M.; Umezu, N.; Shinozuka, Y.: A comparison of two methods for geometric milling simulation accelerated by GPU. Transactions of the institute of systems, Control and Information Engineers, 6 (3) 2013 95–102.
http://doi.org/10.5687/iscie.26.95
[5] Jang, D.; Kim, K.; Jung, J.: Voxel-based virtual multi-axis machining, International Journal of Advanced Manufacturing Technology, 16, 2000, 709-713.
http://doi.org/10.1007/s001700070022
[6] Jerard, R.; Drysdale, R.; Hauck, K.; Schaudt, B.; Magewick, J.: Methods for detecting errors in numerically controlled machining of sculptured surfaces, IEEE Computer Graphics and Applications, 9 (1), 1989, 26-39. http://doi.org/10.1109/38.20331
[7] Lavernhe, S.; Quinsat, Y.; Lartigue, C.; Brown, C.: Realistic simulation of surface defects in 5-axis milling using the measured geometry of the tool, International Journal of Advanced Manufacturing Technology, 74 (1-4), 2014, 393-401. http://doi.org/10.1007/s00170-014-5689-7
[8] Morell-Gimenez, V.; Jimeno-Morenilla, A.; Garcia-Rodrõguez, J.: Efficient toolpath computation using multi-core GPUs, Computers in Industry 64 (1), 2013, 50-56.
http://doi.org/10.1016/j.compind.2012.09.009
[9] Parker, S.; Bigler, J.; Dietrich, A.; et al, OptiX: a general purpose ray tracing engine, ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH, 2010, 29(4), 2010.
http://doi.org/10.1145/1778765.1778803
[10] Quinsat, Y.; Sabourin, L.; Lartigue, C.: Surface topography in ball end milling process: description of a 3D surface roughness parameter, Journal of Materials Processing Technology, 195 (1–3), 2008, 135-143. http://doi.org/10.1016/j.jmatprotec.2007.04.129