Broad Phase Collision Detection: New Methodology and Solution for Standardized Analysis of Algorithms

Ygor Rebouças Serpa[1], Maria Andréia Formico Rodrigues (Orientadora)
Programa de Pós-Graduação em Informática Aplicada (PPGIA)
Universidade de Fortaleza (UNIFOR) – 60811-905 – Fortaleza-CE – Brasil
{ygor.reboucas,andreia.formico}@gmail.com

***Abstract*—Collision detection is a computational problem focused on the identification of geometric intersections between objects and, in general, proximity relationships among them. Despite its notorious relevance and applications in various computing fields, few authors have proposed solutions that are both general and scalable. Additionally, until the time of publication of the results of this work, there was no standard methodology for the analysis of algorithms, neither in academia nor in the industry: only proprietary scenes and comparative studies had been developed, making it difficult to reproduce and compare results. To tackle the issues previously mentioned, we present a new general and scalable solution for broad phase collision detection and a new methodology for comparative analysis of algorithms, named Broadmark, whose open-source code is publicly available, with the goal of transferring knowledge to academia, industry, and society, so far lacking in the scientific literature. Thus, by doing so, we aim to contribute to the generation of robust and multi-faceted solutions applied to various scenarios and, consequently, to greater transparency, ease of modification/extension and reproducibility of results.**

I. Introduction

Collision detection can be seen as a generalization of the k-Nearest Neighbour problem for dynamic scenes with non-punctiform objects, introducing complexities such as the shape and behavior of objects [1]. Although collision detection has been studied for many years, the field still features relevant and active research that has not yet been resolved, introducing additional challenging problems, such as simulation of massive dynamic scenes, deformable bodies, solid-liquid coupling, and robust constraint solving, among others [2], [3]. As a whole, despite its large success, the task is known to be unstable at massive scales and to vary widely in performance whenever scenes deviate from their expected states [4].

Over the years, several efficient solutions have been proposed [5]–[8], however, most works employ a limited set of scenes and algorithms for their comparisons, failing to provide a strong foundation to support their claims. Collectively, these works lack a shared representative methodology, an effort attempted only by Woulfe and Manzke [9], to limited success. On top of that, many authors sacrifice generality over scalability, narrowing their solution to either the static case, in which only a fraction of objects moves, or the dynamic one, in which all of them do. Combined, these problems reveal how challenging it is to weigh the strengths of each work and to faithfully reproduce their results, concerns of paramount importance given the current reproducibility crisis [10].

This work addresses the main research questions fully detailed in the M.Sc dissertation [11] available at ⟨https://1drv.ms/b/s!Aq35PBOZWmsjhppQxK3ut_ILqDvLfA⟩:

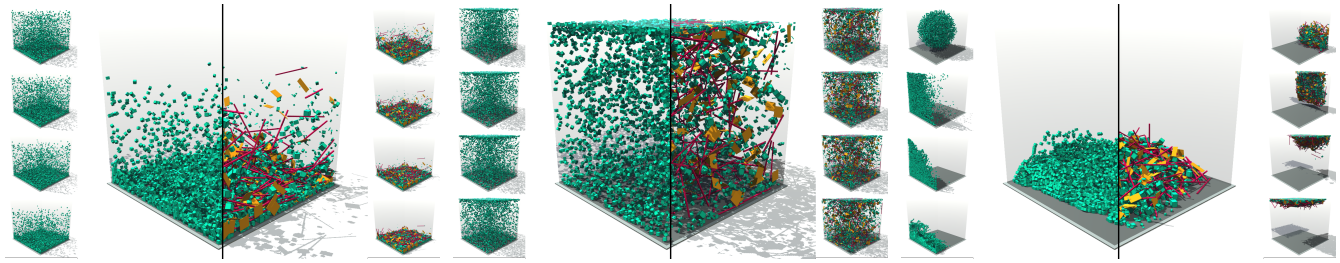
1) To propose **an open and extensible standard methodology**, yet non-existing, to the development and study of broad phase collision detection algorithms [12],
2) To create **a both general-purpose and scalable novel algorithmic solution** to the broad phase collision detection field [4], and
3) To make **all source code of the developed tools**, necessary for this research and evaluation of algorithms, **publicly available on the GitHub repository** ⟨https://github.com/ppgia-unifor/Broadmark⟩, so that anyone interested can inspect, learn from it, test it, develop it, and build on it, thus, in an effort also contributing **to the transfer of the knowledge base to the academy, industry and society.**

More specifically, we have developed the **Broadmark** methodology [12], a research development environment containing 12 algorithm implementations, plus variants, for both CPU and GPU (Table I), as well as three standardized testing scenarios (Figure 1), representatives of the static, dynamic, and uniform cases, with either same sized or randomly sized objects [12]. As part of this system, we have also developed a novel hybrid and adaptive solution based on KD-trees, the Sweep-and-Prune (SAP) algorithm, and the incremental detection paradigm, competitive on all tested scenarios [4].

II. Related Work

To date, the dominating approach for assessing collision detection algorithms is to design your own environment. This includes defining the test cases, searching for baseline algorithms and timing each solution [5]–[8], [13]–[21]. This methodology has a number of flaws, to name a few, results are difficult to reproduce, baselines are not state-of-the-art, and no external guarantees of fairness are given. For instance, the solutions proposed by Lo *et al.* [13], Avril *et al.* [17] and *Liu et al.* [6], although similar and contemporary, share almost nothing in their validation strategies. This problem is relevant even within a same institution [17]–[20]. Regarding fairness, it is not uncommon for results to be presented under favorable situations, such as grid-based algorithms on homogeneously distributed scenes [6], [13] or temporal-optimized algorithms on nearly static scenes [5], [6], [16]. Conversely, algorithms devoid of temporal optimizations are usually tested on fully dynamic scenes only [13], [21], [22]. The work of Woulfe and Manzke [9], presented in 2009, is the most recent attempt to introduce an open space for collision detection research.

[1]M.Sc. dissertation.



(a) *Free Fall.* (b) *Brownian.* (c) *Gravity.*

Figure 1: In (a), (b) e (c), respectively, created scenes using the **Broadmark** [12] framework, representatives of the nearly-static, uniformly-distributed, and fully dynamic cases.

However, it did not get traction within the community and, as of writing, is unfortunately no longer available.

In this context, the **Broadmark** system [12], presented in the M.Sc. dissertation, is meant to (1) reduce the barrier-of-entry to the field by exposing a set of ready-to-use tools, (2) be a repository of state-of-the-art collision detection algorithms, and (3) benchmark a wide range of solutions on a representative set of scenes. As of writing, the system includes algorithms from five different sources that span single and multi-threaded algorithms on both CPU and GPU, shown in Table I. Additionally, representative scenes of the most frequently seen setups in the literature are also bundled: the nearly static (Figure 1.a), the uniformly distributed (Figure 1.b) and the fully dynamic (Figure 1.c), up to a million objects. Finally, the system is designed towards extensibility, featuring a common interface for all new algorithms and a *Unity* [23] based editor for authoring new massive scenes.

The **Broadmark** framework, in its early stages, revealed that most solutions are biased towards one of the three developed scenarios, with none being significantly competitive on all three simultaneously [4]. For a solution to be efficient on all these three scenes, we hypothesized that it had to (1) have temporal optimizations, for the static case; (2) not be impaired by these optimizations on dynamic scenes; and (3) be flexible enough to handle uniformly and non-uniformly distributed objects similarly.

To jointly address these three propositions, we have developed an adaptive hybrid algorithm based on an efficient KD-tree with a SIMD-optimized SAP implementation. Both algorithms run under the incremental detection regime, for the temporal optimization, but can be adaptively toggled to perform the full detection when a specified threshold is met. While KD-trees, SAP and incremental detection, in isolation, are well-established techniques, the synergy between them had not yet been explored in the literature. In the series of tests we carried out, our solution out-performed well-known state-of-the-art algorithms [6], [7], [14], [22] in all three scenarios, being competitive among multi-core and GPUs in several instances.

III. Broadmark

The **Broadmark** system is composed of two independent modules: (1) the simulation generator, developed using the *Unity* game engine, and (2) the algorithm runner, developed purely using the C++ language. The former is accessible either via a pre-compiled wizard, designed for those with no *Unity* expertise, or via the *Unity* project itself, through the engine's editor, while the latter is a command-line tool, for maximum flexibility. As an extra convenience, we also provide a Python tool to design large benchmark schedules.

A. The Simulation Generation

The simulation generation tool was designed using *Unity* to ease the creation and maintenance of massive scenes with beautiful real-time visualizations. When ran, simulations are baked to disk in binary format, completely decoupling the scene generation from the benchmark. Thus, the time spent on generating scenes is constant with regard to the number of algorithms. Moreover, it ensures that each test uses the exact same Axis-Aligned Bounding Boxes (AABBs), improving fairness. Finally, we have designed the system to modulate each physical aspect of the scene to the number of objects used. This way, the ratio between the scene volume and the total volume occupied by objects is constant, regardless of how many objects are used.

Within this system, three scenes have been developed: *Free Fall*, *Brownian*, and *Gravity*, representatives of the static, uniform and dynamic cases, respectively, and designed to be run from one thousand objects to a million. Figure 1 shows the three scenarios with four thousand same sized objects (left half, in green) and varied-sized objects (right half, in assorted-colors).

B. The Algorithm Runner

In the second module, we have included everything related to running and analyzing the algorithms, as well as the scene reader and the logging functionality. Within this module, we gathered 12 sets of algorithms, including original implementations and known algorithms from the literature and industry, spanning serial, parallel, CPU, and GPU algorithms [4], [6], [7], [14], [22]. Table I enumerates each set of algorithms and their most important features. While some sets have only one algorithm, others, such as BF and SAP, include several variants, for instance, SIMD, parallel, and SIMD + parallel implementations. Other families, such as the DBVT family, have unique features, such as being able to run on a forward pass (DBVT F) or in deferred mode (DBVT D), respectively, optimized for the static and dynamic cases.

IV. Hybrid Algorithm

To jointly satisfy the criteria of generality and efficiency, we hypothesized that three properties are needed: **flexibility**, to be versatile enough to handle different object distributions; **efficiency**, to be able to prune huge portions of the search space quickly; and **adaptivity**, to avoid worst-case scenarios and bias towards one kind of simulation over another. In practical terms, to be simultaneously efficient on both static and dynamic scenes.

For the first and second concerns, we have developed a two-tier approach based on the KD-trees and the SAP technique. The former is responsible for dividing the set of objects into smaller, independent, groups, and the latter for processing each individual subproblem using sorting, yielding the set of colliding objects for each sub-problem. Both algorithms complement each other, as the KD-tree is allowed to be shallower, mitigating its overhead, and the SAP is performed over smaller sets, reducing its $\mathcal{O}(n^{\frac{5}{3}})$ time complexity. To efficiently build and update this structure, we have developed an update algorithm which runs in linearithmic time, is idempotent, and is able to work efficiently for both minor adjustments and major tree changes, thus,

Table I: Bundled algorithms within the *Broadmark* system. Time complexity derived for the uniformly distributed case [11].

Algorithms	IMPLEMENTATION					COMPLEXITY	
	Principle	Optimizations	Temporal	Remarks	Source	Time	Space
BF	BF	SIMD + Parallel	-	Naive	Original	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$
SAP	SAP	SIMD + Parallel	-	STL Sort	Original	$\mathcal{O}(n^{5/3})$	$\mathcal{O}(1)$
Grid BF	Grid	Parallel	-	T objects/cell	Original	$\mathcal{O}(n^2/t)$	$\mathcal{O}(nt)$
Grid SAP	Grid + SAP	Parallel	-	T objects/cell	Original	$\mathcal{O}(n^2/t)$	$\mathcal{O}(nt)$
AxisSweep	iSAP	-	Yes	Insertion Sort	Bullet 2	$\mathcal{O}(n+s)$	$\mathcal{O}(n)$
DBVT	BVH	-	Optional	Persistent Tree	Bullet 2	$\mathcal{O}(nlog(n))$	$\mathcal{O}(n)$
CGAL	Tree + SAP	-	-	Stateless	CGAL	$\mathcal{O}(nlog^3(n))$	$\mathcal{O}(n)$
Tracy	Grid + iSAP	Parallel	Yes	Insertion Sort	Authors	$\mathcal{O}(n+s)$	$\mathcal{O}(n)$
KD-Tree	Tree + SAP	SIMD	Adaptive	Adaptive, Persistent Tree	M.Sc Dissertation	$\mathcal{O}(nlog(n))$	$\mathcal{O}(n)$
GPU Grid	Grid	GPU	-	OpenCL	Bullet 3	N/A	N/A
GPU LBVH	BVH	GPU	-	OpenCL	Bullet 3	N/A	N/A
GPU SAP	SAP	GPU	-	OpenCL	Bullet 3	N/A	N/A

being optimized for both static and dynamic scenes. Orthogonal to these developments, we explored the use of SIMD instructions to accelerate the SAP algorithm and an efficient memory layout custom-tailored to reduce the costs of the most expensive operations carried out during the tree update algorithm.

To efficiently cope with static scenes, it is paramount to have algorithms whose time complexity is dependant on the number of dynamic objects, instead of the total. To introduce reasoning within our algorithm, we employ the incremental detection, which consists in detecting only new collisions and ceased collisions. For this to work, we start from the set of collisions from the previous frame, which are screened to remove ceased collisions. Afterwards, we only need to check collisions that involve dynamic objects. In other words, all static-static pairs can be safely pruned from testing. Finally, we label objects with speed below/above 0.05, in any direction, as static/dynamic, respectively. Empirically, we found that the incremental detection is faster than the plain algorithm whenever more than half of the objects are static. To have the best of both worlds regarding temporal optimizations and the lack of thereof, we alternate between both approaches in run-time based simply on the number of static objects labeled on the current frame.

In the full text of the M.Sc. dissertation, detailed analyses of the empirical observations used to guide the design of our solution are presented, as well as the technical and implementation details of each algorithm used.

V. Tests and Results

All implemented algorithms were tested and their average time per frame from one thousand to one million objects, as presented numerically in Table II, and graphically in Figure 2. For brevity, we show the results for same-sized objects only and the worst and best variants of each family, when more than one algorithm is available. To scope our analysis to the best solutions available, we chose 0.5 seconds per frame as a cutoff threshold. More results and their detailed descriptions are provided in the M.Sc. dissertation [11].

In the following, we summarise the findings and contributions made. To the best of our knowledge, this is the most extensive and up-to-date comparative study, in both depth and breadth, published in the field using the same hardware/software setup and test scenes [4], [12]. In a nutshell, on the Brownian scene (Figure 1.b), solutions based on grids are favored, on Free Fall (Figure (1.a), temporal optimized solutions dominate and, finally, on the Gravity scene (Figure 1.c), the best results are obtained by solutions optimized for the dynamic case. In all three, the developed solutions during the M.Sc (KD-tree) performed comparably or better to the best state-of-the-art solutions for each case. In special, our solution is the fastest for the static case (Free Fall), surpassing both multi-core CPU and GPU solutions, performs similarly as the best CPU parallel solutions on the dynamic case (Gravity), and is on par with the best grid-based serial solutions on the uniform case (Brownian), all that being a single-threaded CPU algorithm.

VI. Conclusion and Future Work

We are confident that this work may improve knowledge in the broad phase collision detection area.

In the M.Sc. dissertation [11] we answer the, until then, open question "is it possible to have a solution that is both general and scalable for the broad phase collision detection problem?".

In addition, we contribute to the advancement of the field by releasing the **Broadmark** methodology as an open-source tool for the community.

With these initiatives, this work may have many implications for research into the collision detection area. More specifically, we hope for new research to be developed using the framework and that, in turn, for their results to be more easily reproducible and validated by others, increasing both the significance and impact of each individual contribution. Moreover, with **Broadmark**, we hope to contribute with lowering the barrier of entry for new researchers to the field, to raise the quality of following comparatives and to encourage the research for novel, general-purpose, solutions.

Finally, in the near future, we plan to investigate novel parallel CPU and GPU solutions while retaining the joint focus on generality and speed, and to further develop the **Broadmark** framework to include new algorithms and scenes, as well as to support other related tasks, such as collision queries and continuous detection. Related to the topic, we are interested in applying deep learning models to collision detection research, on topics such as cloth and volume deformations.

VII. Awards and Publications

The two main results of this work, *i.e.*, the new general and scalable solution for broad phase collision detection and the new methodology for the comparative analysis of algorithms named **Broadmark**, have been individually published in the *Computer Graphics Forum* (CGF) journal [4], [12] (Qualis A1), and come from the exclusive effort of the student and advisor, not being part of an overarching project.

In addition, during the M.Sc, two other journal publications related to this work have been published in the *Computers in Entertainment* [24] (Qualis B1) and ACM *Entertainment Computing* [25] (Qualis B1) journals.

Moreover, the authors have been formally invited by the program Co-Chairs of the *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA'19)* (Google Scholar h5-index 20) to present the results published on the CGF journal [4] as a journal first entry at *UCLA – University of California*, in Los Angeles, USA, testifying the quality and relevance of the research to the scientific community.

Table II: Results for each algorithm and their variants (s/frame) [11].

BROWNIAN

	BF		SAP		Grid BF		Grid SAP		Axis	DBVT		Tracy				GPU		
# objects	**ST**	**MT**	**ST**	**MT**	**ST**	**MT**	**ST**	**MT**	**Sweep**	**F**	**D**	**ST**	**MT**	**CGAL**	**KD**	**SAP**	**LBVH**	**Grid**
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.11	0.00	0.01	0.00	0.02	0.01	0.00	0.00	0.02	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00
16	0.42	0.01	0.02	0.00	0.03	0.01	0.00	0.00	0.06	0.01	0.01	0.02	0.01	0.01	0.00	0.00	0.00	0.00
32	1.60	0.04	0.06	0.01	0.04	0.02	0.01	0.00	0.23	0.04	0.02	0.05	0.02	0.03	0.01	0.00	0.00	0.00
64		0.16	0.18	0.02	0.09	0.04	0.02	0.01	1.09	0.09	0.05	0.12	0.05	0.08	0.01	0.01	0.00	0.00
128		0.60	0.57	0.04	0.17	0.07	0.03	0.02		0.25	0.14	0.26	0.11	0.18	0.03	0.02	0.01	0.01
256				0.13	0.36	0.13	0.06	0.03		1.42	0.44	0.70	0.31	0.41	0.06	0.06	0.02	0.01
512				0.34	0.78	0.23	0.14	0.08			1.04		1.19	0.95	0.13	0.19	0.05	0.04
768				0.63		0.32	0.21	0.14							0.20	N/A	0.08	0.07
1,024						0.39	0.28	0.19							0.28		0.10	0.08

FREE FALL

	BF		SAP		Grid BF		Grid SAP		Axis	DBVT		Tracy				GPU		
# objects	**ST**	**MT**	**ST**	**MT**	**ST**	**MT**	**ST**	**MT**	**Sweep**	**F**	**D**	**ST**	**MT**	**CGAL**	**KD**	**SAP**	**LBVH**	**Grid**
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.11	0.00	0.01	0.00	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
16	0.42	0.01	0.01	0.00	0.07	0.02	0.01	0.00	0.01	0.00	0.01	0.01	0.00	0.02	0.00	0.00	0.00	0.00
32	1.65	0.05	0.04	0.00	0.15	0.05	0.02	0.01	0.04	0.01	0.03	0.01	0.01	0.05	0.00	0.01	0.00	0.00
64		0.16	0.14	0.01	0.41	0.14	0.05	0.02	0.18	0.02	0.08	0.04	0.02	0.12	0.00	0.01	0.01	0.01
128		0.68	0.41	0.04	1.05	0.37	0.10	0.05	1.09	0.05	0.18	0.09	0.05	0.30	0.01	0.03	0.02	0.01
256			1.30	0.10		0.94	0.24	0.11		0.10	0.42	0.23	0.12	0.65	0.03	0.06	0.04	0.03
512				0.24			0.60	0.22		0.22	1.09	0.57	0.29		0.06	0.16	0.08	0.07
768				0.46				0.35		0.36			0.47		0.08	N/A	0.12	0.10
1,024				0.85				0.51		0.52			0.71		0.11		0.15	0.14

GRAVITY

	BF		SAP		Grid BF		Grid SAP		Axis	DBVT		Tracy				GPU		
# objects	**ST**	**MT**	**ST**	**MT**	**ST**	**MT**	**ST**	**MT**	**Sweep**	**F**	**D**	**ST**	**MT**	**CGAL**	**KD**	**SAP**	**LBVH**	**Grid**
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.13	0.00	0.01	0.00	0.07	0.01	0.01	0.00	0.09	0.02	0.01	0.01	0.08	0.01	0.00	0.00	0.00	0.00
16	0.51	0.01	0.02	0.00	0.14	0.06	0.02	0.01	0.36	0.04	0.02	0.03	0.15	0.02	0.01	0.00	0.00	0.00
32		0.05	0.05	0.01	0.34	0.15	0.03	0.02	0.89	0.15	0.05	0.29	0.52	0.04	0.01	0.01	0.00	0.00
64		0.17	0.16	0.01	0.84	0.36	0.08	0.03		0.57	0.15	2.90		0.09	0.03	0.01	0.01	0.01
128		0.63	0.51	0.04		0.76	0.17	0.06			0.44			0.22	0.06	0.03	0.01	0.01
256				0.11			0.37	0.14			0.99			0.51	0.14	0.06	0.02	0.03
512				0.27			0.82	0.33							0.31	0.17	0.05	0.06
768				0.48				0.51							0.49	N/A	0.09	0.09
1,024				0.72											0.68		0.13	0.12

The SCA'19 program schedule and invitation letter can be found, respectively, at ⟨https://sca2019.kaist.ac.kr/wordpress/program/⟩ and ⟨https://1drv.ms/b/s!Aq35PBOZWmsjhsUyoq9eevMYa5pD-w⟩.

In parallel, still in 2019, the field of artificial intelligence started to be studied as a complementary tool. This study led to other two scientific contributions: the presentation of a tutorial on the *Conference on Graphics, Patterns and Images (SIBGRAPI'19)* [26], and a First Place Award-Winning full paper in the Computing Track of the *Brazilian Symposium on Games and Digital Entertainment (SBGAMES'19)* [27].

Finally, this work has also recently won a prestigious award from the *XXXIII Concurso de Teses e Dissertações (CTD)* of the *XL Congresso da Sociedade Brasileira de Computação (CSBC)*: It is among the Top 10 finalists chosen from 73 M.Sc. dissertations in the Computing area in Brazil, concluded in 2019. The 3 winners will be announced next November, during the *CSBC* conference, after online presentation of the 10 finalist works.

REFERENCES

[1] D. M. Ming C. Lin and Y. J. Kim, "Collision and proximity queries," in *Handbook of Discrete and Computational Geometry*, 3^{rd} ed. CRC Press, 2017, ch. 39.

[2] C. Ericson, *Real-time Collision Detection*. CRC Press, 2004.

[3] D. H. Eberly, *Game physics*. CRC Press, 2010.

[4] Y. R. Serpa and M. A. F. Rodrigues, "Flexible use of temporal and spatial reasoning for fast and scalable CPU broad-phase collision detection using KD-Trees," *Computer Graphics Forum (CGF)*, vol. 38, no. 1, pp. 1–14, 2017.

[5] G. Capannini and T. Larsson, "Adaptive collision culling for massive simulations by a parallel and context-aware Sweep and Prune algorithm," *TVCG*, vol. 24, no. 7, pp. 2064–2077, 2018.

[6] F. Liu, T. Harada, Y. Lee, and Y. J. Kim, "Real-time collision culling of a million bodies on graphics processing units," *ACM Trans. on Graphics (TOG)*, vol. 29, no. 6, pp. 1–8, 2010.

[7] L. Kettner, A. Meyer, and A. Zomorodian, "Intersecting sequences of dD iso-oriented boxes," in *CGAL User and Reference Manual*, 5.0 ed., 2019.

[8] R. G. Luque, J. a. L. D. Comba, and C. M. D. S. Freitas, "Broad-phase collision detection using semi-adjusting BSP-trees," in *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games (I3D)*. ACM, 2005, pp. 179–186.

[9] M. Woulfe and M. Manzke, "A framework for benchmarking interactive collision detection," in *Proc. of the* 25^{th} *Conf. on Computer Graphics*. ACM, 2009, pp. 205–212.

[10] M. Baker, "Reproducibility crisis?" *Nature*, vol. 533, no. 26, pp. 353–66, 2016.

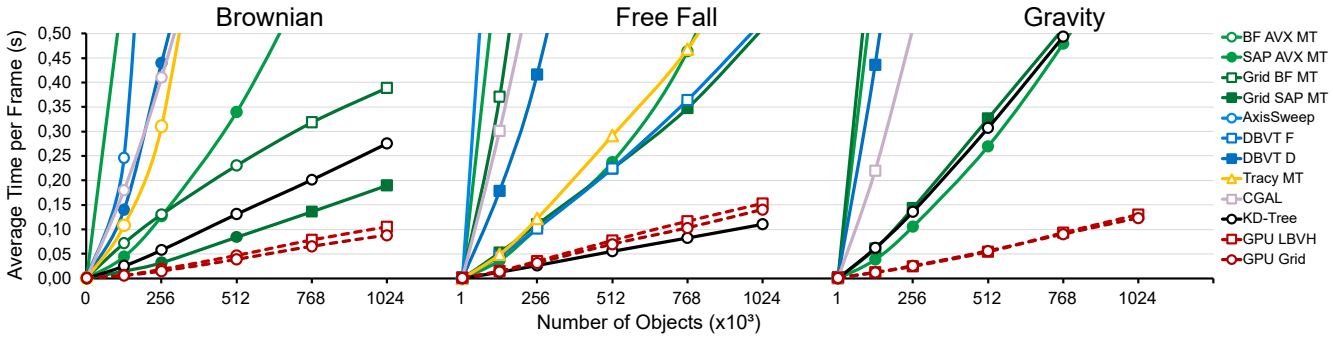


Figure 2: Scalability analysis up to one million objects.

[11] Y. R. Serpa, "Detecção de colisão broad phase: Nova solução e metodologia implementadas para análise padronizada de algoritmos," Master's thesis, Programa de Pós-Graduação em Informática Aplicada (PPGIA). Universidade de Fortaleza (UNIFOR). Defesa: 19/12/2019, 2019.

[12] Y. R. Serpa and M. A. F. Rodrigues, "Broadmark: A testing framework for broad-phase collision detection algorithms," *Computer Graphics Forum (CGF)*, vol. 39, no. 1, pp. 436–449, 2019.

[13] S.-H. Lo, C.-R. Lee, I.-H. Chung, and Y.-C. Chung, "Optimizing pairwise box intersection checking on GPUs for large-scale simulations," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 23, no. 3, pp. 1–22, 2013.

[14] D. J. Tracy, S. R. Buss, and B. M. Woods, "Efficient large-scale Sweep and Prune methods with AABB insertion and removal," in *Proceedings of the 2009 IEEE Virtual Reality Conference (VR)*. Lafayette, LA, USA: IEEE, 2009, pp. 191–198.

[15] G. Capannini and T. Larsson, "Efficient collision culling by a succinct bi-dimensional Sweep and Prune algorithm," in *Proceedings of the* 32^{nd} *Spring Conference on Computer Graphics (SCCG)*. Smolenice castle, Slovakia: ACM, 2016, pp. 25–32.

[16] D. J. Tracy and S. Brown, "Accelerating physics in large, continuous virtual environments," *Concurrency and Computation: Practice & Experience*, vol. 24, no. 2, pp. 125–134, 2012.

[17] Q. Avril, V. Gouranton, and B. Arnaldi, "Fast collision culling in large-scale environments using GPU mapping function," in *Proc. of the 2012 Eurographics Symposium (EG)*. Cagliari, Italy: The Eurographics Association, 2012, pp. 71–80.

[18] ——, "Collision detection: broad phase adaptation from multi-core to multi-GPU architecture," *Journal of Virtual Reality and Broadcasting*, vol. 6, no. 11, pp. 1–13, 2014.

[19] ——, "Dynamic adaptation of broad phase collision detection algorithms," in *2011 IEEE International Symposium on VR Innovation*, March 2011, pp. 41–47.

[20] ——, "Synchronization-free parallel collision detection pipeline," in *Proceedings of the* 20^{th} *International Conference on Artificial Reality and Telexistence*, Adelaide, Australia, Dec. 2010, pp. 1–7.

[21] A. Zomorodian and H. Edelsbrunner, "Fast software for box intersections," in *Proceedings of the* 16^{th} *Annual Symposium on Computational Geometry (SoCG)*. Clear Water Bay, Hong Kong: ACM, 2000, pp. 129–138.

[22] E. Coumans, "Bullet Physics," github.com/bulletphysics/bullet3, 2018.

[23] Unity Technologies, "Unity 2019.2," unity.com, 2019.

[24] D. V. Macedo, Y. R. Serpa, and M. A. F. Rodrigues, "Fast and realistic reflections using screen space and GPU ray tracing—a case study on rigid and deformable body simulations," *ACM Computers in Entertainment (CIE)*, vol. 16, no. 4, p. 5, 2018.

[25] Y. R. Serpa, M. B. Nogueira, H. Rocha, D. V. Macedo, and M. A. F. Rodrigues, "An interactive simulation-based game of a manufacturing process in heavy industry," *Entertainment Computing (ENTCOM)*, vol. 34, pp. 1–11, 2020.

[26] Y. R. Serpa, L. A. Pires, and M. A. F. Rodrigues, "Milestones and new frontiers in deep learning," in *Proceedings of the SIBGRAPI-T 2019*. IEEE, 2019, pp. 22–35.

[27] Y. R. Serpa and M. A. F. Rodrigues, "Towards machine-learning assisted asset generation for games: A study on pixel art sprite sheets," in *Anais do SBGames'19*. IEEE, 2019, pp. 182–191.