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

Ygor Rebouças Serpa
Programa de Pós-Graduação em Informática Aplicada
Universidade de Fortaleza (UNIFOR)
Fortaleza, Brazil
ygor.reboucas@gmail.com

Maria Andréia Formico Rodrigues
Programa de Pós-Graduação em Informática Aplicada
Universidade de Fortaleza (UNIFOR)
Fortaleza, Brazil
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.**

***Keywords*—Collision detection, broad phase, kd-trees, rigid-body simulation, opensource**

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, robot planning and robust constraint solving, among others [1]–[5]. 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 [6]. For game developers, in particular, this is a daily issue as physics engines are optimized for near-static scenes, requiring designers to either minimize how many physical objects can interact and move to a bare minimum or make compromises elsewhere.

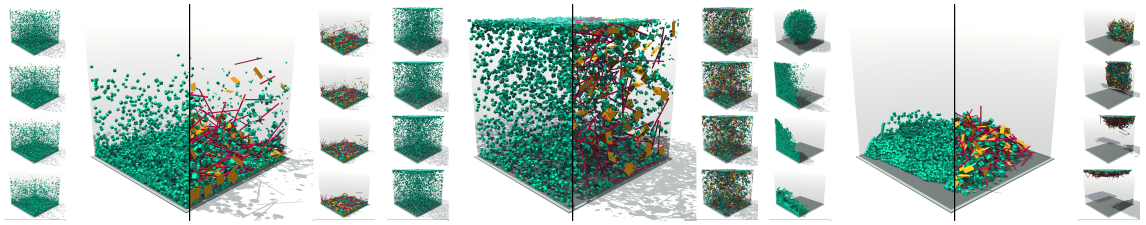
We thank FUNCAP and CNPq for the financial support.

Over the years, several efficient solutions have been proposed, 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 [7], 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 [8]. This master thesis[1] [9] addresses the following questions:

1) To propose **an open and extensible standard methodology**, yet non-existing, to the development and study of broad phase collision detection algorithms [10],
2) To create **both a general-purpose and scalable novel algorithmic solution** to the broad phase collision detection field [6], and
3) To make **all source code of the developed tools**, necessary for this research and evaluation of algorithms, **publicly available on GitHub**[2], 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 **Broadmark** [10], a research development environment containing 12 algorithm families from both CPU and GPU (Table I), as well as standardized testing scenarios (Fig. 1), representatives of the static, dynamic and uniform cases, with same sized or randomly sized objects [10]. As part of this system, we have 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 [6].

[1]Thesis: ⟨https://1drv.ms/b/s!Aq35PBOZWmsjhppQxK3ut_ILqDvLfA⟩
[2]Github: ⟨https://github.com/ppgia-unifor/Broadmark⟩



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

Fig. 1. **Broadmark** [10] built-in scenes, representatives of the nearly-static, uniformly-distributed, and fully dynamic cases.

The framework [6] and algorithm [10] contributions have been published individually on the *Computer Graphics Forum (CGF)* journal (Qualis A1) and come from the exclusive effort of the student and advisor, not being part of any overarching project. During the M.Sc., two other journal publications related to this work were accepted to the *Computers in Entertainment* [11] (Qualis B1) and *ACM Entertainment Computing* [12] (Qualis B1) journals, respectively. Additionally, the authors were formally invited by the program co-chairs of the *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*[3] (SCA'19') (Google Scholar h5-index 20) to present the results published on the *CGF* journal [6] as a journal first entry[4], on the *University of California (UCLA)*, in Los Angeles, USA, testifying the quality and relevance of the research to the scientific community. The presentation was carried out by the student on site on July 26^{th}, 2019. In parallel, still in 2019, the field of artificial intelligence started to be studied as a complementary tool. This study led to two scientific contributions: a tutorial on the *Conference on Graphics, Patterns and Images (SIBGRAPI'19)* [13] and a full research paper on the *Brazilian Symposium on Games and Digital Entertainment (SBGames'19) Computing Track* [14], awarded First Place at the SBGames Computing Track.

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. theses concluded in 2019. The 3 winners will be announced next November, during the *CSBC* conference, after online presentation of the 10 finalist works.

II. Broadmark

To date, the dominating methodology in the field is to design your own environment. This includes defining the test cases, assembling baseline algorithms and timing each solution. This methodology has a number of flaws, to name a few, results are difficult to reproduce, baselines are often not state-of-the-art, and no external guarantees of fairness are given. In this context, the proposed **Broadmark** system [10], 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. In more detail, the 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.

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 a 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 runs on the exact same input, improving fairness. Finally, we modulate each physical aspect of the scenes to the number of objects used. This way, the ratio between the scene volume and the combined objects volume is constant, regardless of object count. Finally, the system has three built-in scenes: *Free Fall*, *Brownian*, and *Gravity*, representatives of the static, uniform and dynamic cases, respectively, designed to be run from one thousand objects to a million. Fig. 1 shows the three scenarios with four thousand same sized objects (left half, in green) and varied-sized objects (right half, in assorted-colors).

The second module is concerned with loading scenes, running algorithms, and logging measurements. Within it, we assembled 12 sets of algorithms, including original implementations and known algorithms from the literature and industry, spanning serial, parallel and GPU algorithms. Table I lists each set and their distinct 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.

III. Hybrid Algorithm

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 [6]. To jointly satisfy the criteria of

[3]Invitation: ⟨https://1drv.ms/b/s!Aq35PBOZWmsjhsUyoq9eevMYa5pD-w⟩
[4]SCA'19 schedule: ⟨https://sca2019.kaist.ac.kr/wordpress/program/⟩

TABLE I. BUNDLED ALGORITHMS WITHIN BROADMARK [9].

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

generality and efficiency, we hypothesized that three properties are needed: **flexibility**, to handle different object distributions; **efficiency**, to prune huge portions of the search space quickly; and **adaptivity**, to bypass worst-case scenarios. For the first two concerns, we have developed a two-tier approach based on the KD-Tree, responsible for dividing the set of objects into smaller, independent, groups; and the SAP technique, responsible for processing each individual subproblem using sorting, yielding the set of colliding objects for each subproblem. 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, mitigating its $\mathcal{O}(n^{\frac{5}{3}})$ time complexity. While both algorithms have been extensively studied in the past, our work is the first one to study them in combination for the collision detection problem.

Regarding the tree construction and maintenance, we developed an update algorithm which runs in linearithmic time, is idempotent, and efficiently performs both minor and major adjustments, thus, 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 a custom memory layout 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 solely on the number of dynamic objects. To this end, we employ the incremental approach: to detect only new and ceased collisions. The latter are detected by screening the current set of collisions at the beginning of each frame and the former by selectively testing only collisions involving dynamic objects. This labeling is based on speed. Objects with speed below/above 0.05, in any direction, are labeled 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 are presented along with the technical and implementation details of each algorithm used.

IV. TESTS AND RESULTS

All implemented algorithms were tested and their average time per frame is presented in Table II, from one thousand to one million objects. For brevity, we show the results for same-sized objects only and the worst and best variants of each algorithm. To scope our analysis to the best solutions available, we chose 0.5 seconds per frame as a cutoff threshold. In-depth analyses are provided in the M.Sc. thesis [9].

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 published, in both depth and breadth using the same hardware, software, and scenes [6], [10]. In a nutshell, on the Brownian scene (Fig. 1.b), solutions based on grids are favored, on Free Fall (Fig. 1.a), temporal optimized solutions dominate and, finally, on the Gravity (Fig. 1.c), solutions optimized for the dynamic case perform best. In all three, the developed solutions during the M.Sc. (KD-Tree) performed comparably or better to the best state-of-the-art solutions. 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 solutions on the uniform case (Brownian), all being a single-threaded CPU algorithm.

V. CONCLUSION AND FUTURE WORK

We are confident that this work may improve knowledge in the broad phase collision detection area. In the M.Sc. thesis [9] we answer the open question "is there 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 open-sourcing **Broadmark** to 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, in turn, their results be more easily reproducible and validated by others, increasing both the significance and impact of each individual contribution. Moreover, with **Broadmark**, we hope to lower the barrier of entry for new researchers, to raise the quality of following comparatives and to encourage the research for novel, general-purpose, solutions.

TABLE II. RESULTS FOR EACH ALGORITHM AND THEIR VARIANTS (SECONDS PER FRAME) [9].

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

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 with new algorithms and scenes, as well as support for 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.

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] 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.

[5] J. S. Park and D. Manocha, "Efficient probabilistic collision detection for non-gaussian noise distributions," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1024–1031, 2020.

[6] 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, 2019.

[7] 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.

[8] M. Baker, "Reproducibility crisis?" *Nature*, vol. 533, no. 26, 2016.

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

[10] 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, 2020.

[11] 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.

[12] 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.

[13] 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.

[14] 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.