A Review of Motions in Microseconds

Pravi Samaratunga

Boston University ECE

July 23, 2024

Introduction I

Sampling-based motion planners have computationally expensive subroutines:

- collision checking
- nearest neighbors search
- forward kinematics

By improving the performance of some of these planning primitives, sampling-based motion planners can acheive substantial improvements.
Sampling-based motion planners are commonly used in robotics, and speedups in this domain can improve safety and responsiveness.
Motion planning for robotic manipulation tasks often takes several seconds to execute Thomason, Kingston, and Kavraki 2023.
If motion planning can be done on the fly, the ability to plan reactively could expose whole problem domains that were previously impractical to explore.

Introduction II

By using the architectural principles of fine-grained parallelism and work-ordering, the researchers were able to achieve a 500x speedup over previous state-of-the-art approaches.

Sampling Based Motion Planning Algorithms I

Example algorithms:

- RRT [diagrams & description]
- PRM
- EST

Several common steps:

- sample the state space

building a state tree to explore
[A slide for each primitive, visual/physical intuition for each piece] This typically makes use of a *nearest neighbors* (NN) search of the state space. The state is mapped to physical space using *forward kinematics* (FK). The planner next *checks for collisions* (CC) to ensure that the selected state does not collide with any obstacles in the environment, nor does any point in the state space along the edge that is added.
This is conventionally thought to be the most computationally intensive step of the process Thomason, Kingston, and Kavraki 2023.

State of the Art I

Collision checks are performed in two sweeps: *Broadphase* approximates a set of possible collisions by modeling the environment in faster-to-compute ways. *Narrowphase* takes these possible collisions and returns the actual collisions. Historically, SBMP has been parallelized in a coarse-grained fashion, running many independent planners in parallel. There is space to optimize SBMP parallelism in terms of the planning primitives the authors identified.

Innovation I

In the paper, the authors present Vector Accelerated Motion Planning (VAMP), which provides significant motion planning speedups by conceptualizing sampling-based motion planning as a composition of primitives, and introducing vectorized implementations of those primitives. Vectorizing these operations allows for fine-grained parallelism at the instruction level, which is underexplored in the robotics domain.

Collision Checking I

One of the major innovations presented is the use of a *Struct-of-Arrays*, as opposed to the more traditional *Array-of-Structs*.

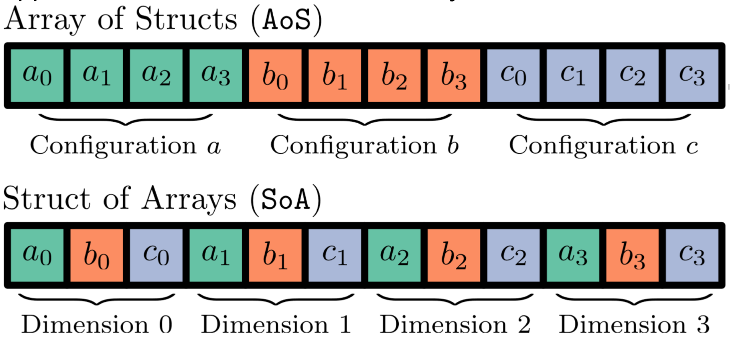

Array of Structs (AoS)
a_0 a_1 a_2 a_3 b_0 b_1 b_2 b_3 c_0 c_1 c_2 c_3
Configuration a
Configuration b
Configuration c
Struct of Arrays (SoA)
a_0 b_0 c_0 a_1 b_1 c_1 a_2 b_2 c_2 a_3 b_3 c_3
Dimension 0
Dimension 1
Dimension 2
Dimension 3

Figure duplicated from Thomason, Kingston, and Kavraki 2023

Due to CPU memory access patterns, the Struct-of-Arrays layout is more favorable for SIMD approaches. VAMP replaces the existing CC & FK

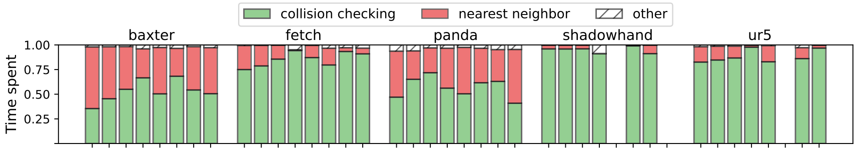
Collision Checking II

primitives with vector-oriented versions, reducing the cost of given computation. In order to vectorize FK, VAMP generates a *trace* from a URDF to create an unrolled loop. In order to vectorize CC, the robots and obstacles are represented as a composition of bounding volumes (e.g. spheres), and collisions are calculated between these known geometries.

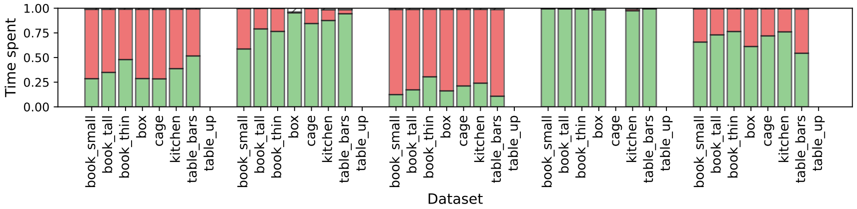
Strengths & Weaknesses I

This paper shows how some simple optimizations can yield serious empirical results, including a 500x speedup over the state of the art. However, the authors fail to pose the question of *why* they get these results. They combine several innovations, but it remains unclear how each impacts end to end performance.

?? shows a preliminary analysis of the end-to-end breakdown of motion planning performance, specifically in terms of the collision checking and nearest neighbor planning primitives.

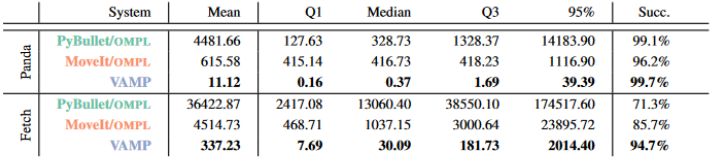


Strengths & Weaknesses II



Collision checking (CC) vs. nearest neighbors (NN) in (a) RRTConnect and (b) BiEST. Bars clustered by robot. NN (red) takes up significant time ($>30\%$) in over half (55%) of all workloads. Figure from unpublished work.

The paper also does not include much exploration on why different workloads show different amounts of improvement with the VAMP optimizations. There's less of a speedup in the 8DOF Fetch robot than the 7DOF Panda, though the speedup still very significant. *Explicitly state those ratios - redo the entire table actually*



	System	Mean	Q1	Median	Q3	95%	Succ.
Panda	PyBullet/OMPL	4481.66	127.63	328.73	1328.37	14183.90	99.1%
	MoveIt/OMPL	615.58	415.14	416.73	418.23	1116.90	96.2%
	VAMP	**11.12**	**0.16**	**0.37**	**1.69**	**39.39**	**99.7%**
Fetch	PyBullet/OMPL	36422.87	2417.08	13060.40	38550.10	174517.60	71.3%
	MoveIt/OMPL	4514.73	468.71	1037.15	3000.64	23895.72	85.7%
	VAMP	**337.23**	**7.69**	**30.09**	**181.73**	**2014.40**	**94.7%**

Table II. Planning times for PRM over problem classes *table pick*, *table under pick*, and *box*. The mean, first quantile, median, third quantile, 95% quantile, and success rate are shown. All times are in **milliseconds**.

Thomason, Kingston, and Kavraki 2023

Future Work I

I want to explore the particulars of the vectorized collision checking subroutine.
While there is significant work in the robotics community using spheres as geometric primitives Sundaralingam et al. 2023, there is research outside of the robotics domain regarding colliders using different kinds of geometric primitives, such as trimeshes and capsules n.d.

(N.d.). URL: https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/Collision/.

Sundaralingam, Balakumar et al. (2023). *cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation*. arXiv: 2310.17274 [cs.RO]. URL: https://arxiv.org/abs/2310.17274.

Thomason, Wil, Zachary Kingston, and Lydia E. Kavraki (2023). *Motions in Microseconds via Vectorized Sampling-Based Planning*. arXiv: 2309.14545 [cs.RO]. URL: https://arxiv.org/abs/2309.14545.