The paper presents a framework for multithreaded RRT-based motion planning introducing four different parallelization strategies. Results comparing the time efficiency of these strategies for different planning problems are presented.

The authors address a relevant but also well-studied problem. Most reviewers generally appreciate the presented framework but also point out several issues that need to be resolved prior to publication.

The literature review needs to be complemented with related work on parallelization schemes as pointed out in the reviews.

The clarity of presentation in the algorithms and the experimental section needs to be be improved. To avoid confusion, the established CS term 'critical section' should be used instead of 'critical region'. Superlinear speedups in Fig.7 and 8 should be discussed. See reviewers' comments for further details.

----------------------------------------

Comments on Video Attachment:

Reviewer 3 of IROS 2019 submission 806

Comments to the author

======================

In this paper, authors presented a parallel mechanism for RRT-based algorithms, such as RRT, Bi-RRT, RRT\*. By parallel searching, the computation load is separately distributed and the total computation time is effectively reduced. Detailed comments are listed as follows..

1. In section II.A, the computational time analysis is decided by two main factors, i.e., optimal trajectory

connecting two states and required collision checks. According to eq(1), the obstacles avoiding checks is $O(It\_V)$. But in the connecting process, intermediate states along $\tau$ also require to be checked whether admissible in configuration space $X$. Authors are suggested to describe this part more clearly.

2. Why two same procedures "threads barrier" are called in line 13-14 of Alg.5. It seems an editing error.

3. The description of Alg.4-Alg.6 is poor. For the purpose of reading this paper, the key procedures in each

algorithms are supposed to be described clearly.

4. In section III-B and III-C, critical regions are designed to achieve the parallel searching. Authors are

supposed to present the detailed settings of critical regions.

Reviewer 8 of IROS 2019 submission 806

Comments to the author

======================

The pseudo code is somewhat hard to follow. In Algorithm 1, Steer is not defined.

The example figures are also hard to interpret. The red jagged line in Figures 3 and 4 looks like a path but I

think it is the robot.

In the caption to Figure 3 there seems to be text missing. "On the left the poses xo and xf, while..." is not a

complete phrase and some text needs to be added. Also it is impossible to see where xo and xf are.

Problems like this have been pursued for decades. They are all members of the class of shortest path algorithms. As a user of such algorithms for robot task planning over 50 years ago it is amazing to me to see the challenges still remaining and the similarities to methods we used (forward and backward chaining, for example) still being explored. I used the Hart, Nilsson, Raphael A\* algorithm.

Reviewer 14 of IROS 2019 submission 806

Comments to the author

======================

This work describes a multithreading implementation to speed up RRTs. I have 2 major comments:

1) The references section is not complete. A better coverage of RRTs should be carried out.

2) As this is an experimental work, a thorough analysis of performance of the 4 strategies is required. A table showing the performance of the multithreaded strategies in terms of computational time as well as

computational space. The results should be compared with the original RRT and fast implementations of RRT such as RRG, RRT\*, and LBT-RRT.

Reviewer 16 of IROS 2019 submission 806

Comments to the author

======================

This paper addresses multithreading strategies for RRT, bidirectional RRT or RRT\*. In detail, it presents the

efficiency of parallelization (normalized speed up) for four cases with four multithreading strategies. The

multithreading strategies consist of 1) querying activities, 2) a shared tree, 3) copied trees and 4)

blinded multiagents explorations. Its result is that the multithreading strategy with blinded multiagents

explorations is most efficient.

Comments

1) In the bidirectional version, expansions are made in forward and backward trees. So, it seems that Eq. (2) is the same as Eq. (1).

2) In parallel exploration on a shared tree, authors use the term, critical region, which is not clearly

defined. It seems like the critical region is shown as the relation, || xs – xf || <= Threshold in Algorithm 4.

3) In the results, authors present methods for the simulations to compare the efficiency of multithreading strategies. In Fig. 5, 6, 7 and 8, it seems like the bars in those graphs represent the standard deviation. So, authors should indicate the meaning of the bars. If they indicate the standard deviation, authors should present how many simulations are made with 2500 iterations.

4) It is more desirable that authors additionally present a mean computation time (including a standard deviation) required to get a solution for each strategy.