RACOD: Algorithm/Hardware Co-design for Mobile Robot Path Planning

Mohammad Bakhshalipour
Mohamad Qadri
Maxim Likhachev

Seyed Borna Ehsani
Dominic Guri
Phillip B Gibbons
Executive Summary

- Robots are becoming increasingly important
  - Global market by 2025: US $210 billion

- Real-time robots have to solve complete AI problems at speed
  - Great compute capabilities

- **RACOD** accelerates mobile robot path planning by more than 40x
  - Hardware acceleration
  - Algorithm extension
Mobile Robot Path Planning

- Collision Detection
- Path Search
RACOD

- Hardware acceleration
  - Collision detection
- Algorithm extension
  - Path search algorithm
Outline

• Background and challenges

• Hardware acceleration

• Algorithm extension

• Evaluation
Mobile Robot Path Planning

Inputs:
• Start
• Goal
• Occupancy Grid

Output:
• Path

Algorithm:
• Dijkstra
• A*
• ...
Why Costly?

• Collision detection is costly
• The search algorithm is largely serial
Why Costly?

- Collision detection is costly
Why Costly?

- Collision detection is costly

\[ r = 1 \rightarrow 9 \text{ cells} \]
\[ r = 10 \rightarrow 384 \text{ cells} \]
Why Costly?

- The search algorithm is largely serial

```python
while (goal is not expanded) {
    expand s

    for successor s’ of s:
        status[s’] = check_collision(s’)
}
```
Why Costly?

• The search algorithm is largely serial

```plaintext
while (goal is not expanded) {
    expand s

    for successor s' of s:
        status[s'] = check_collision(s')
}
```
Our Proposal: RACOD

RACOD: Run-Ahead Collision Detection
Our Proposal: RACOD

RACOD: **Run-Ahead COllision Detection**

CODAcc  

RASExp
Our Proposal: RACOD

RACOD: Run-Ahead Collision Detection

CODAcc

COllision Detection Accelerator

RASExp

Run-Ahead State Exploration
Hardware Acceleration Insights

- Abundant fine-grained parallelism
- Excellent spatial locality
CODAcc Hardware Realization

Diagram showing the components of CODAcc hardware realization:
- Cache Block
- Load Queue
- $L_0$ Cache
- Scheduler
- Collision?
CODA Acc Hardware Realization

MapReduce-style address generation

- Cache Block
- Load Queue
- CMP
- Address
- Value
- Collision?
CODAcc Hardware Realization

- Cache Block
- Load Queue
- CMP
- Address
- Value
- Collision?
CODAcc Hardware Realization

- Cache Block
- Load Queue
- CMP
- Address
- Value
- Collision?
CODAcc Hardware Realization

- Cache Block
  - Address
  - Value
  - Load Queue
  - CMP
  - Scheduler
  - Configuration
  - Collision?
CODAcc Hardware Realization
CODAcc Hardware Realization
Use Multiple CODAccs?

• No!
  • Parallelism is severely limited

\[
\text{while (goal is not expanded) \{ }
  \text{remove s with minimum } f(s) \text{ for successor } s' \text{ of } s:
  \text{status}[s'] = \text{check_collision}(s')
\]

Use Multiple CODAccs?

- No!
  - Parallelism is severely limited

\[
\text{while (goal is not expanded) } \{
  \text{remove } s \text{ with minimum } f(s)
\}
\]

\[
\text{for successor } s' \text{ of } s:
  \quad \text{status}[s'] = \text{check_collision}(s')
\]

- Every point is connected to at most 4-8 nodes
- Average parallelism is way smaller than maximum
  - E.g., 2.2 vs. 8
Algorithm Extension Insights

• Paths explored in robot path planning exhibit very regular patterns.

Predict next points

+ Check their collision status speculatively

+ Memoize the result
Algorithm Extension

```
while goal is not expanded {
    remove s with minimum f(s)

    for successor s’ of s:
    `{status[s’]=check_collision(s’)}

    for node n in predicted_nodes:
    `{status[n]=check_collision(n)}
}
```
RASExp Prediction

• The last direction will repeat
  • A very simple predictor but works great in practice!

• Why does the last direction repeat?
  • Geometry principles + World’s organization
Methodology

• TSMC’s 45-nm technology

• Intel Core i3-8190U
  • A popular robotic processor

• zsim simulator
Results: Self-Driving Cars

![Self-Driving Car Diagram]

- Boston
- London: Obstacle
- Moscow
- Shanghai
Results: Pilotless Drones
Prediction Accuracy

- Boston
- London
- Moscow
- Shanghai
- Freiburg

Prediction Accuracy:

- R=2
- R=4
- R=8
- R=16
- R=32
Prediction Accuracy

Worst-case > 85%
Also In The Paper

• Stationary robotic arm planning

• GPU implementation (2.8x speedup)

• Different path planning algorithms

• Different predictors

• Synthetic environments
RTRBench

A Benchmark Suite for Real-Time Robotics

RTRBench

Real-Time Robotics Benchmark (RTRBench) is a collection of real-time robotic kernels. The kernels span the entire software pipeline of most autonomous robots (Listing of Benchmarks). RTRBench is published in ISPASS 2022 [Paper] [Slides] [Talk].

https://cmu-roboarch.github.io/rtrbench
Thanks for Your Attention!
Executive Summary

- Robots are becoming increasingly important
  - Global market by 2025: US $210 billion

- Real-time robots have to solve complete AI problems at speed
  - Great compute capabilities

- RACOD accelerates mobile robot path planning by more than 40x
  - Hardware acceleration
  - Algorithm extension