Real-time Adaptive Non-holonomic Motion Planning in Unforeseen Dynamic Environments

RAMP-H

S. McLeod, J. Xiao, Department of Computer Science
University of North Carolina, ICRA 2016

Presented by: Alexander Holston

KAIST

Objectives

- **Paper aims for dynamic path planning considering:**
  - **Global search and real-time adaptation of non-holonomic paths**
  - **Smooth switching/changing of trajectories**
  - **Adaptation and prediction of motion error**

- **To do this it utilizes:**
  - **Genetic Algorithm inspired approach**
  - **Bezier Curve – For online adaptation of paths**
  - **Obstacle movement prediction**

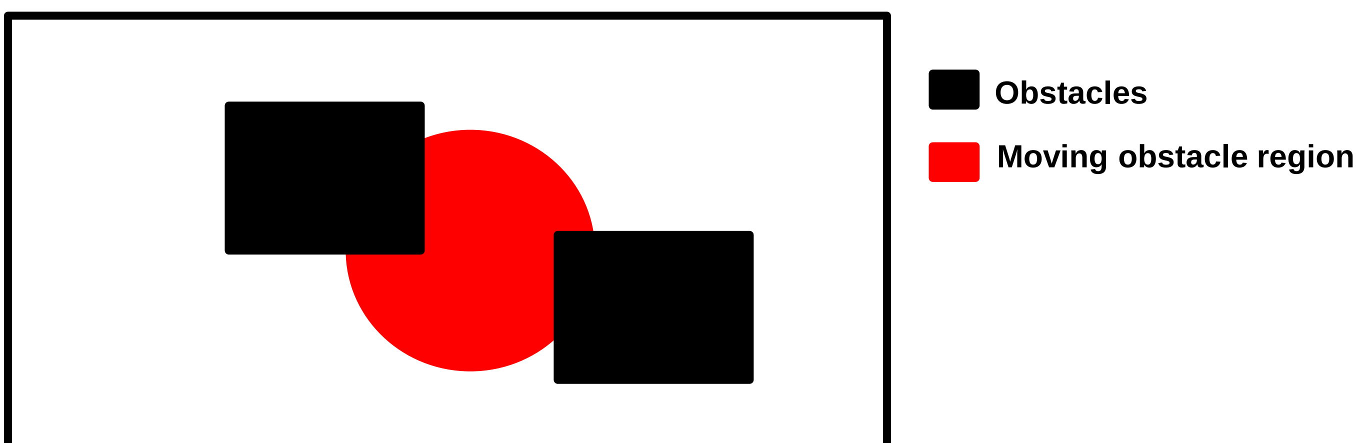
KAIST

Dynamic Environment Approaches

- **Most previous algorithms do not consider unknown motion**
- **Gaussian Artificial Potential Fields**
  - **We know these fail in global motion planning (local minima)**
- **No algorithm is fully complete**
  - **Although we may try to move in a currently optimal path, there is always a chance of collision**

Real-time Adaptive Motion Planning (RAMP)

- **Originally created for efficient navigation through unforeseen dynamic obstacles for manipulators with high DoF**
- **Generates initial population of paths from start to goal**
  - **Initiated randomly to goal**
  - **Forced paths – can help avoid homotopic paths**

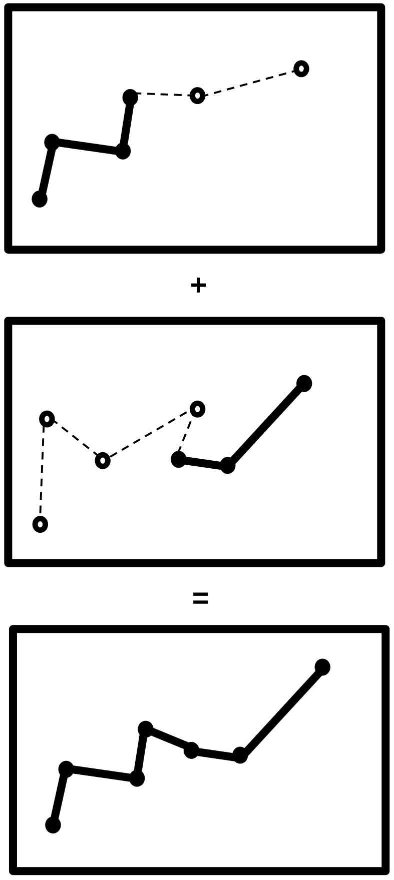


Real-time Adaptive Motion Planning (RAMP)

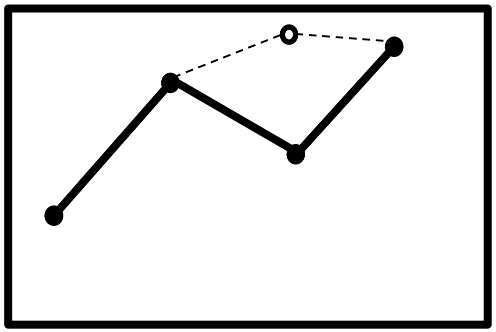
- **Genetic Algorithm modifications**
  - **Add – Add node**
  - **Delete – Delete node**
  - **Change – Change node location**
  - **Swap – Swap nodes**
  - **Crossover – Mix two**

- **Replace non-best fitness (random)**

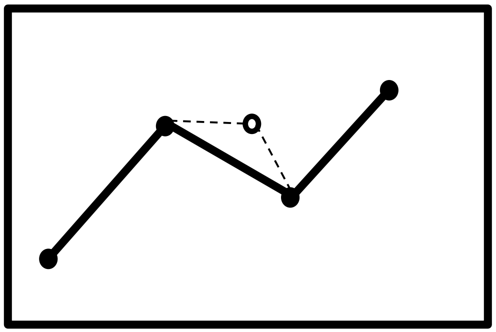
Crossover



Change



Add

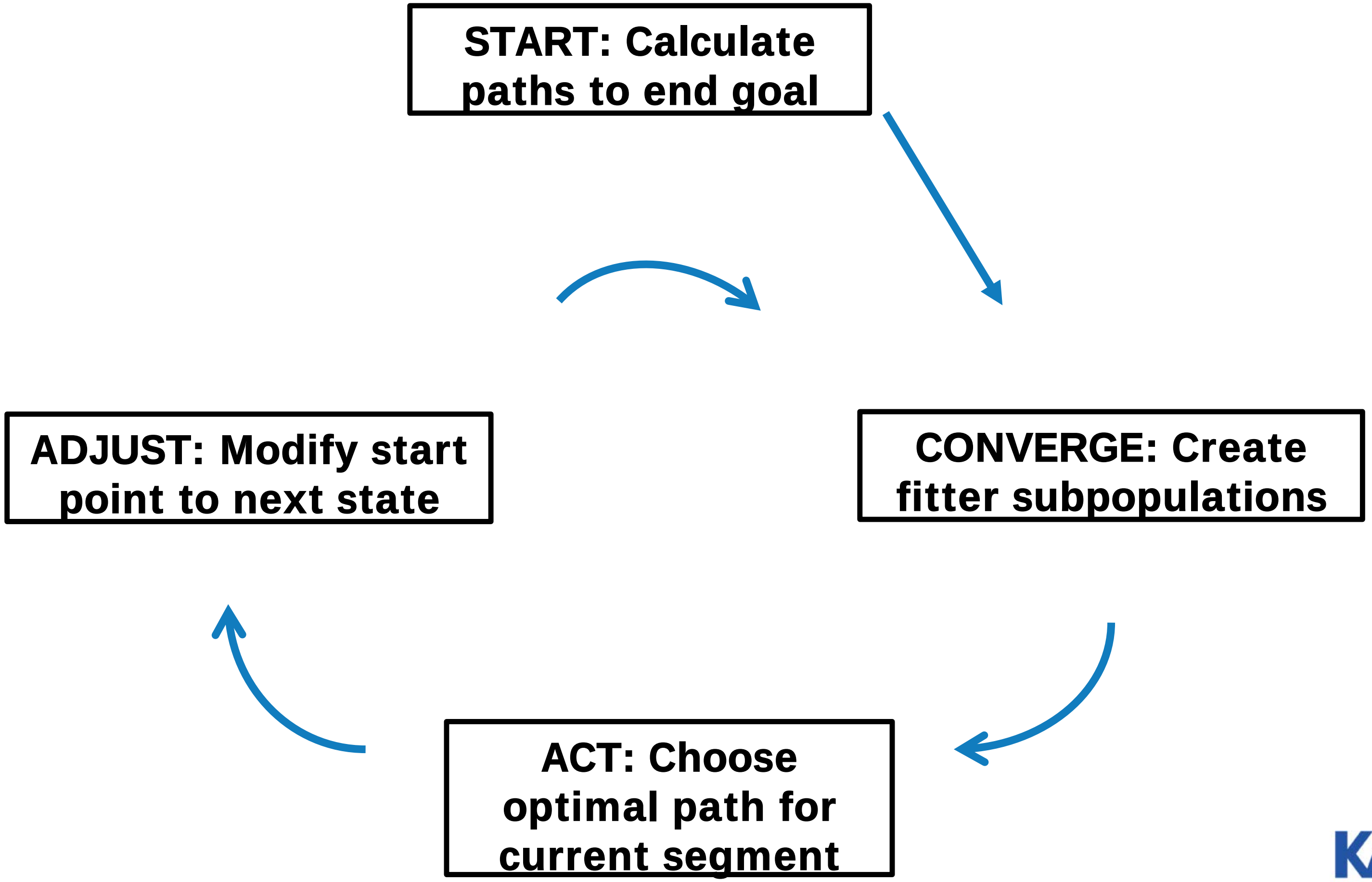


Real-time Adaptive Motion Planning (RAMP)

- **Path choosing is based on:**
  - **Feasibility**
    - **Will the robot collide with an object?**
  - **Minimal Cost**
    - **Time**
    - **Energy**
    - **Manipulability**
    - **Includes cost to ‘switch’ paths**
      - **Decelerating, changing direction etc.**
      - **Ensures stable switching**
  - **Infeasible trajectories are calculated by total feasible cost + penalty**

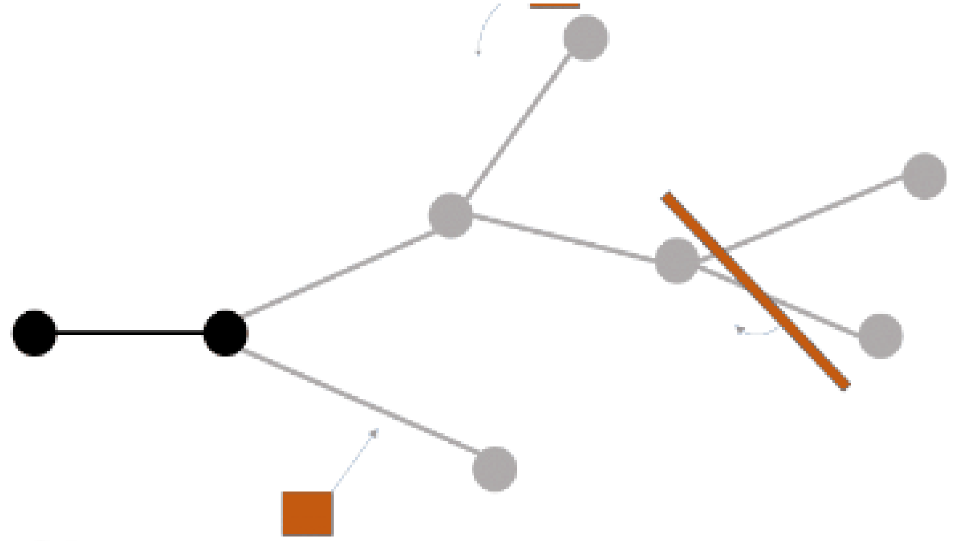
Real-time Adaptive Motion Planning (RAMP)

- **Planning continually checks for infeasibility and optimal path for next control cycle**



Real-time Adaptive Motion Planning (RAMP)

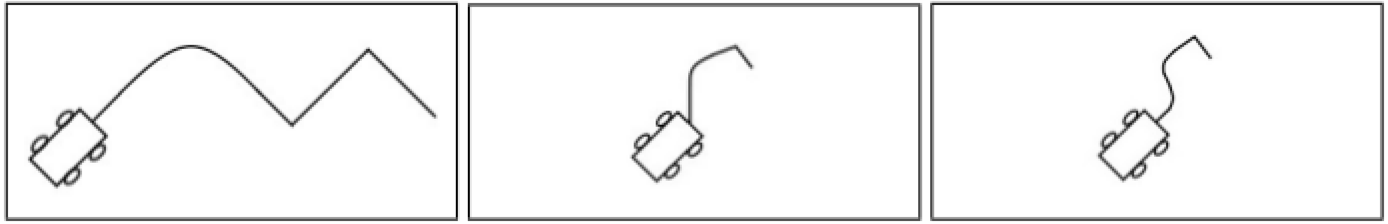
- **Approximates rough future trajectory based on previous measurements**
  - **Sorts into 4 types of movements, depending on values of velocity and angular rotation and their directions**
- **Predicts next time-step movement (sensing cycle) and checks collision**
  - **Given no fully feasible path, the most feasible can be chosen (It could clear in future)**



Non-holonomic Extension

- **Non-holonomic robots suffer from additional constraints**
  - **Original RAMP paths have vertices requiring axial rotation**

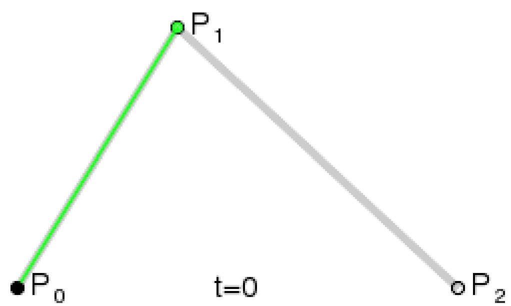
- **RAMP-H adds the capability of adapting these paths to allow for smooth switching**

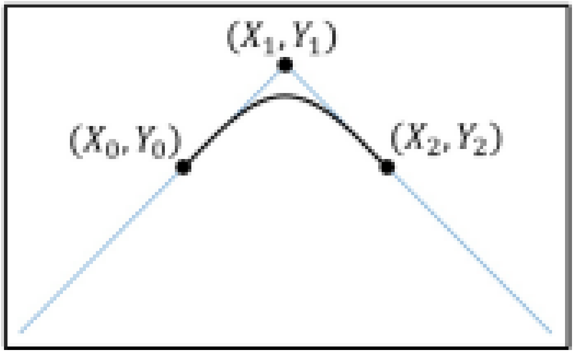


(a) Example hybrid trajectory (b) Robot has incorrect orientation (c) Bezier curve added to enable moving on trajectory

Non-Holonomic RAMP (RAMP-H)

- **Given a 2D example of a non-holonomic car**
  - **We know a car is limited only by a turning circle, and changing speed accordingly**
  - **Given three points a quadratic Bezier curve can be created**





Improved Trajectory Sensing

- **After each cycle:**
  - **Population for next cycle is taken from previous**
  - **Starting state (pos/vel) is updated to current**

- **Due to error in real-world circumstances**
  - **Algorithm polls the robot sensors to get actual actuator states**
  - **Subtracts the difference**

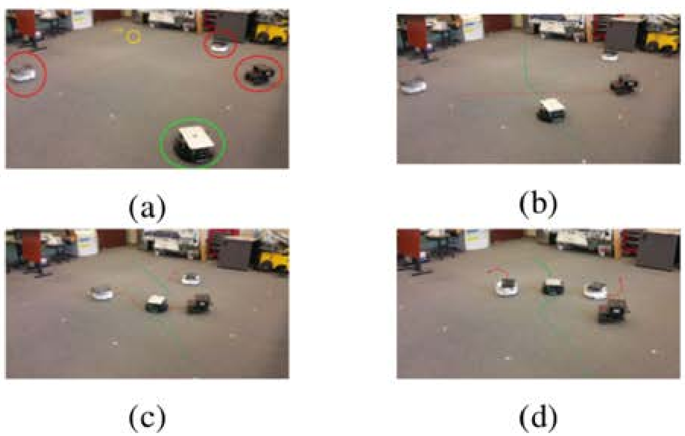
- **Simple way of updating for next cycle while removing actuator inaccuracies**

KAIST

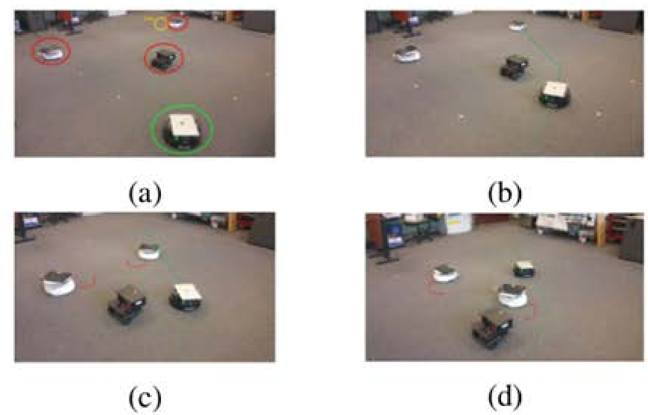
Results

- **One robot setup using RAMP-H**
- **Others have ‘unforeseen’ movements**

- **No external sensors or cameras are incorporated**



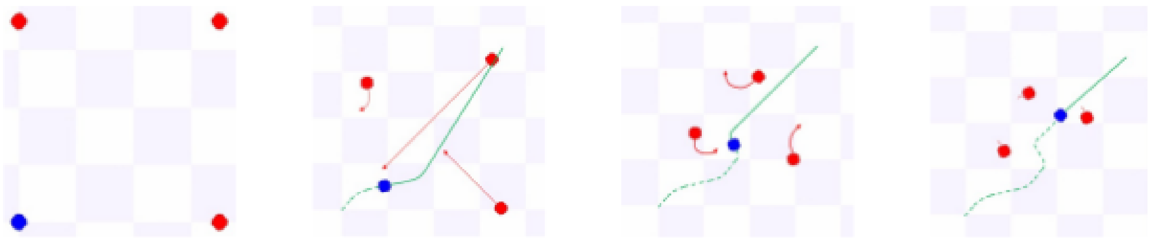
The RAMP-H robot (circled in green) moving among three dynamic obstacles (circled in red)



The RAMP-H robot (circled in green) moving among two dynamic obstacles and one static obstacle (all obstacles circled in red)

Results

- **We see robot moves in a general direction with priority to avoid nearby moving obstacles**



(a) (b) (c) (d)

The RAMP-H robot (blue) and three dynamic obstacles (red) in simulation

Results

- **The ability to switch paths without rotating on axis reduced execution time**

TABLE I: Execution Time

	Hybrid Traj.	Holonomic Traj.
Mean execution time	22.72s	35.31s
Standard dev.	3.09s	7.64s

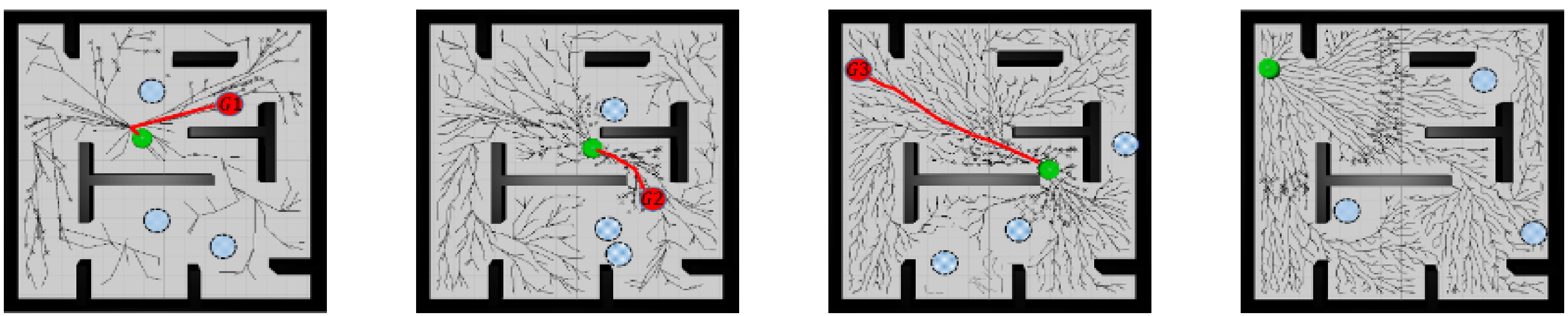
TABLE II: Cycle Time and # Cycles

	Planning Cycle	Control Cycle
Mean time	44ms	1.7s
Standard dev.	27ms	0.32s
Mean # per run	221	12
Standard dev.	39	1.5

- **Note the planning time is small, especially considering the control time**
  - **Plenty of time for a number of genetic iterations before each leg**

Final Intuitive Comparison

- **Why use RAMP-H?**
- **Compare:**
  - **Genetic Algorithm Approach**
    - **Multiple paths, switching, inefficient computation (Best solution from given pop)**
  - **Real-Time RRT* Approach**
    - **Multiple paths, constant rewiring around obstacles (Converges on best solution)**



(a) (b) (c) (d)

Final Intuitive Comparison

- **Low-DoF**
  - **RT-RRT* could allow online convergence in simple problems**
  - **RAMP(-H) uses essentially as much computation as possible to converge the population**

- **High-DoF**
  - **RRT* known to lack reasonable convergence speed at increased DoF**
  - **RAMP(-H) could still find a path online, given limitation to underlying path population**

Conclusion

- **RAMP originally created for feasible high DoF dynamic obstacles avoidance**
  - **Given amount of wasted computation, at low DoF perhaps there are better methods**
  - **Perhaps a better solution lies somewhere between**

- **Regardless, RAMP-H gives an approach to slowly find better solutions to avoid objects of unforeseen movement while online**
  - **Gives solution to switch smoothly using Bezier curves**

Questions????

KAIST