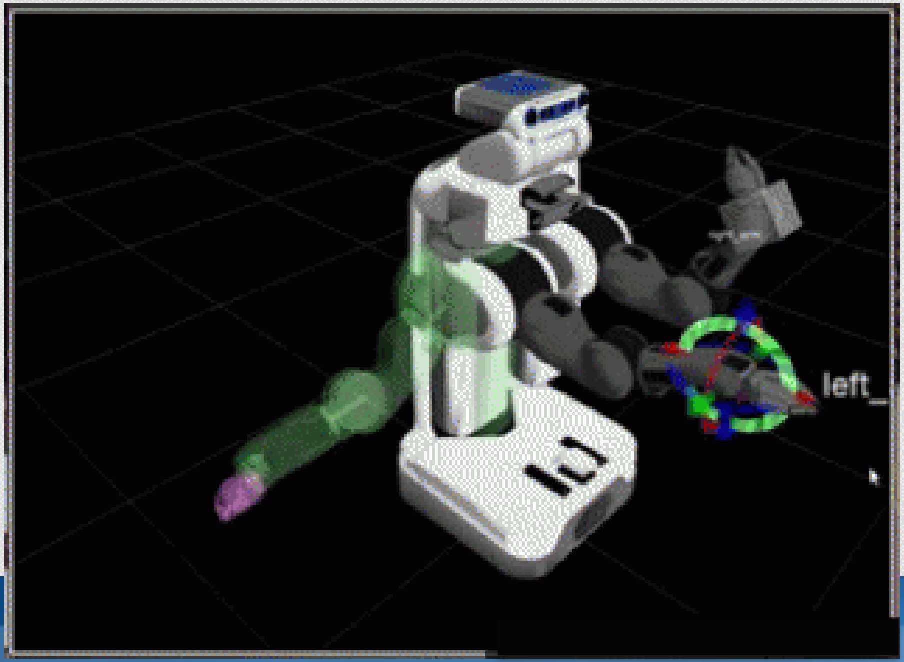
ITOMP: Incremental Trajectory Optimization for Real-time Replanning in Dynamic Environments

Chonhyon Park and **Jia Pan** and **Dinesh Manocha**
University of North Carolina at Chapel Hill

Motion Planning

- Find a continuous, collision-free motion trajectory from an initial pose to a goal pose
- An important problem in many robotics, virtual prototyping, gaming, CAD/CAM, etc.

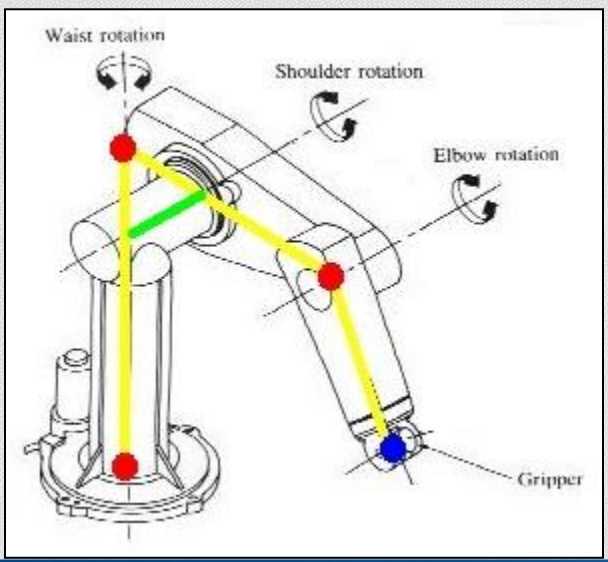


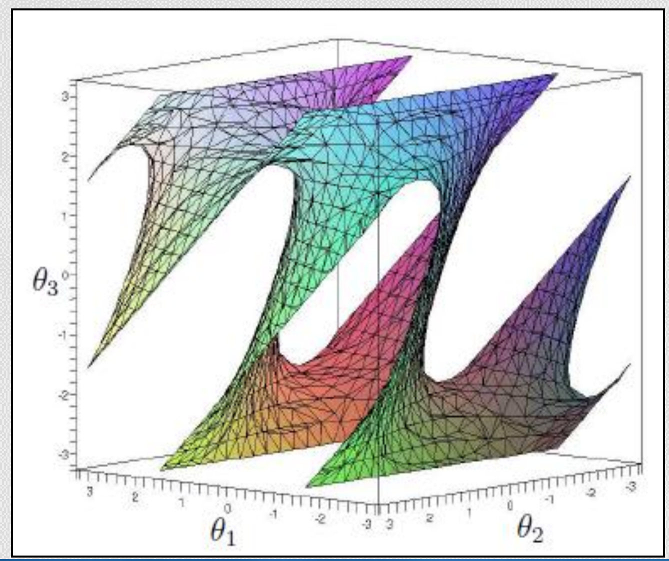
Motion Planning in Dynamic Environments

- Increasing use of robots in human-like environments & real-world scenarios
- Hard to make assumptions about the motion of obstacles
- Need real-time approaches that can adapt to the environment
- Need to compute smooth paths and satisfy (dynamics) constraints

Configuration Space

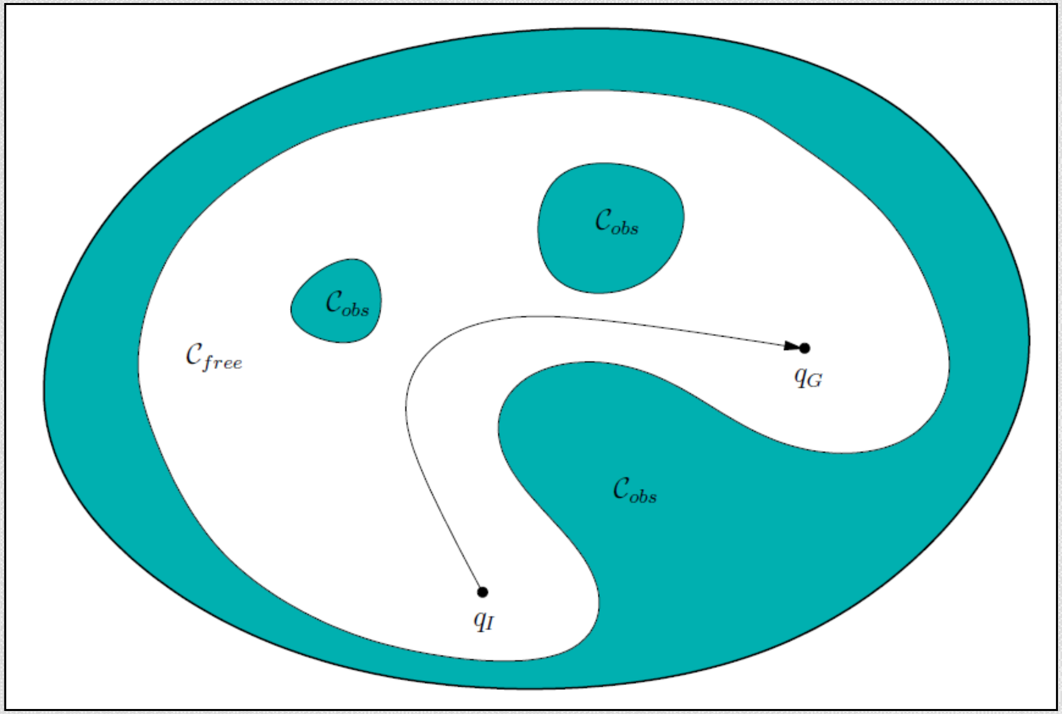
- Motion planning reduces to path computation in configuration spaces (C-space)
- For a robot with k DOFs, C-space is a k-dimensional space





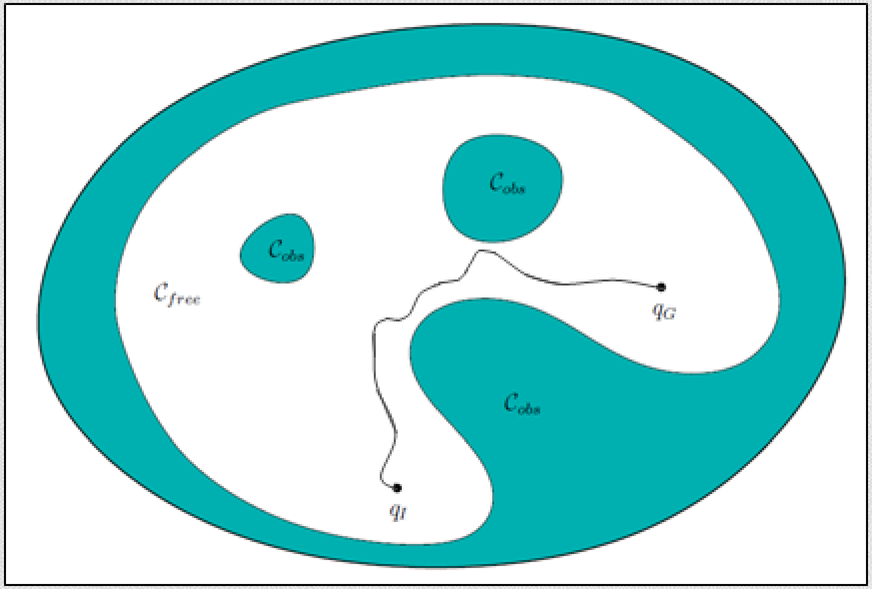
Configuration Spaces

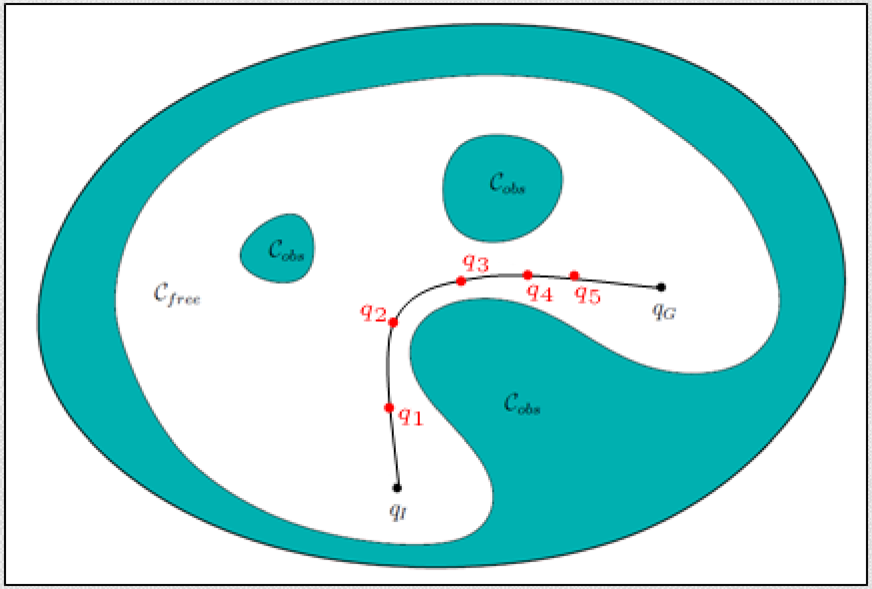
Find a collision-free path from an initial point to a goal point, which lies in the same connected component in C-space



Motion Planning Algorithms

- Random sampling-based algorithms
- Optimization-based algorithms





Motion Planning: Prior work

- Random sampling-based algorithms
  - PRM based methods [Kavraki et al. 1996]
  - RRT based methods [Kuffner and LaValle 2000]
  - Widely used in many real applications
  - Involves preprocessing or limited dynamic scenes

Motion Planning: Prior work

- Optimization-based Planning Algorithms
  - Based on earlier techniques based on potential field methods
  - Can handle dynamic obstacles (in low dimensions)
  - Generate smooth trajectories

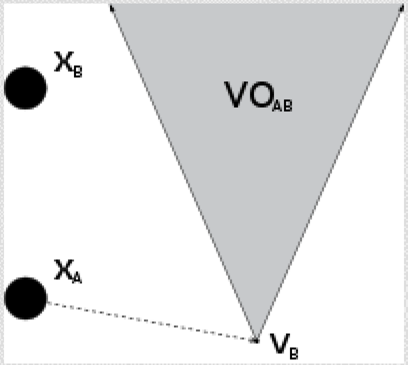
Optimization-Based Planning: Recent work

- Gradient Optimization [Ratliff et al. 2009]
  - Discretize the trajectory to waypoints
  - Compute costs of each waypoint
  - Use gradient to move waypoints to minimize total cost
  - Repeat iteration until find a solution

- Stochastic Optimization[Kalakrishnan et al. 2011]
  - Use stochastic gradient
  - Any value can be a cost factor
    - Torque, orientation constraints, etc.

Motion Planning in Dynamic Environments

- Velocity obstacles
  - [Fiorini and Shiller 1998]
  - [Wilkie, van den Berg, and Manocha 2009]



- Inevitable collision states
  - [Petti and Fraichard 2005]
- Real-time replanning
  - [Bekris and Kavraki 2007]
  - [Hauser 2011]

Optimization-based Planning Algorithm

- Objective function for optimization

$$\min_{\mathbf{q}_1,\ldots,\mathbf{q}_N} \left[\sum_{i=1}^{N} c(\mathbf{q}_i) + \frac{1}{2}\|\mathbf{A}\mathbf{Q}\|^2 \right]$$

  - $c(q_i)$: Cost for each **waypoint**
    - Collision cost for static obstacles : computed by the distance field
  - $\|\mathbf{AQ}\|^2$: Cost for the **smoothness** of trajectory
    - $$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & & 0 & 0 & 0 \\ -2 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & -2 & 1 & & 0 & 0 & 0 \\ & \vdots & & \ddots & & \vdots & \\ 0 & 0 & 0 & & 1 & -2 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -2 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{pmatrix}$$
    - Represents the sum of squared accelerations

Dynamic Environments: Issues

- Previous algorithms assume static environments
  - Distance fields for collision avoidance
    - Uses precomputation methods
    - Lookup is fast, but dynamic updates are slow
  - Planning before execution
    - Can lead to long delays in movement
    - Not safe in uncertain dynamic environments

Optimization-based Motion Planning

- Pros
  - Smooth trajectory computation
  - Other constraints (dynamics) can be handled
- Cons
  - **Environments** – Assumes static environment
  - **Performance** – Slow!
  - **Quality** – Local minima may prevent planner to find a collision-free or good solution

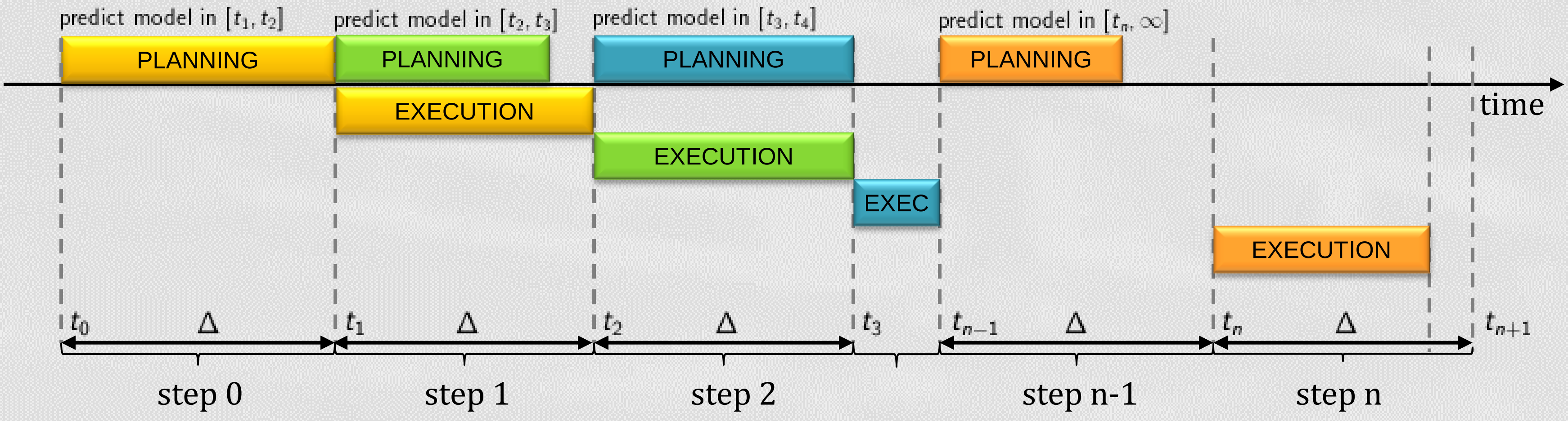
Our Approach

- Interleave planning with execution
  - Handle dynamic environments
  - A general scheme for collision avoidance and smooth path computation
  - Improves Safety
- Parallelize trajectory optimization
  - Reduces cost computation time
  - Improves the search: larger coverage of C-space
  - Better Performance & Quality

Motion Planning in Dynamic Environments

- Future motions of obstacles are unknown
- Use local estimates based on recent position of the obstacles
- Planner cannot estimate exact motions
  - Recent position data from sensor has noise
  - Obstacles may change their trajectory during planning computation

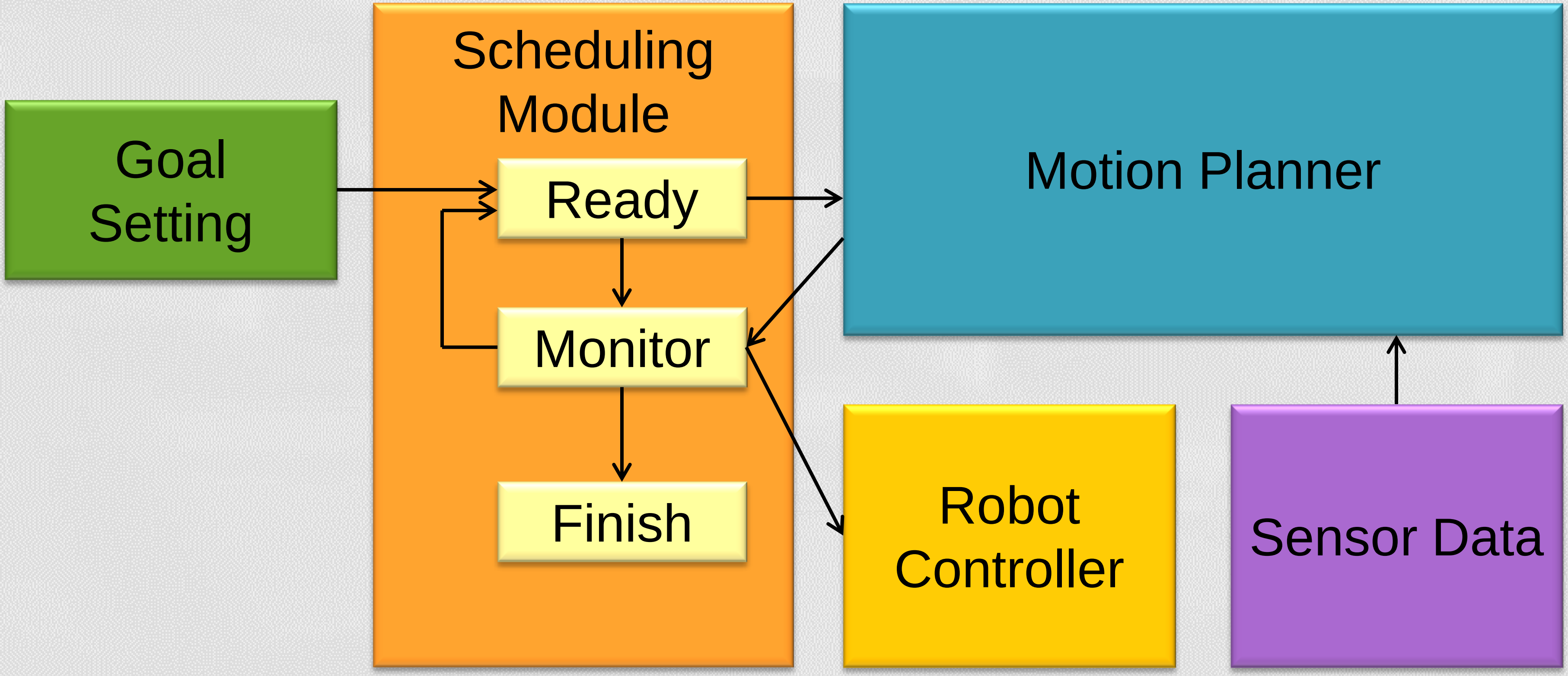
Real-time Replanning



- Interleave planning with execution
  - Compute partial plan for the next execution step
  - Improve the trajectory while execution
  - Use the latest information about the dynamic environment

Real-time Replanning

- Overall Pipeline: Dynamic Environments



Handling Dynamic Obstacles

- Modified objective function

$$\min_{\mathbf{q}_1,\dots,\mathbf{q}_N} \sum_{i=1}^{N} \left[c_s(\mathbf{q}_i) + c_d(\mathbf{q}_i) + \frac{1}{2}\|\mathbf{A}\mathbf{Q}\|^2 \right]$$

  - $c_s(\mathbf{q}_i)$: Costs for static obstacles use precomputed distance fields

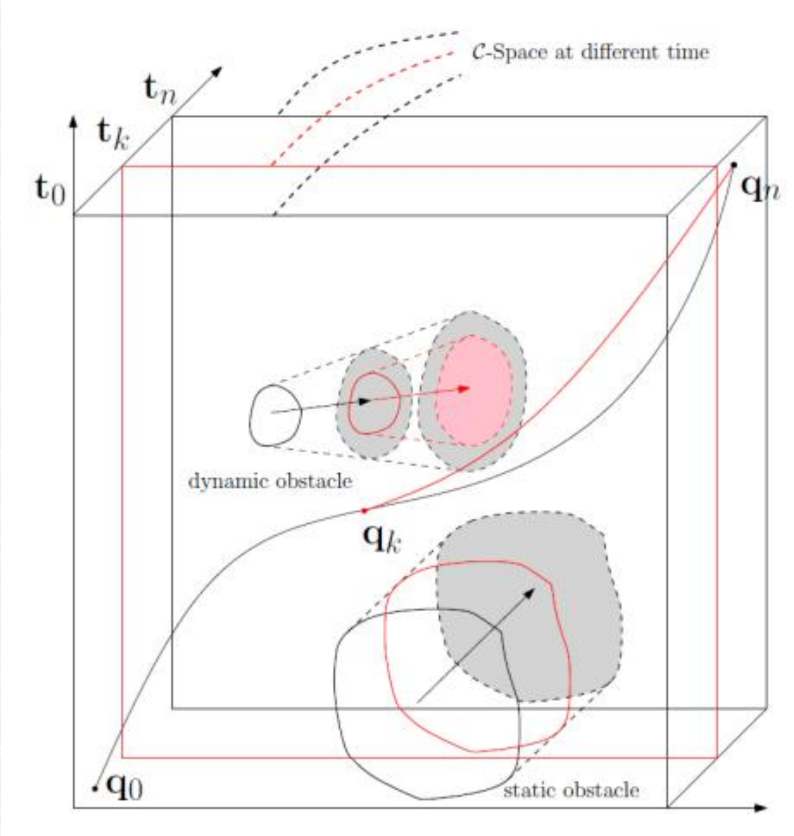
  - $c_d(\mathbf{q}_i)$: Costs for dynamic obstacles use the collision detection between the robot and obstacles

Dynamic Obstacles: Collision Checking

- Compute motion bounds on the local trajectories of dynamic obstacles
- Use bounding volumes and hierarchies for fast collision checking
- Hierarchies are computed/updated incrementally

Handling Dynamic Obstacles

- Use conservative bounds
  - The predicted position of obstacles may not be accurate
  - Use conservative bounds on obstacles for collision checking



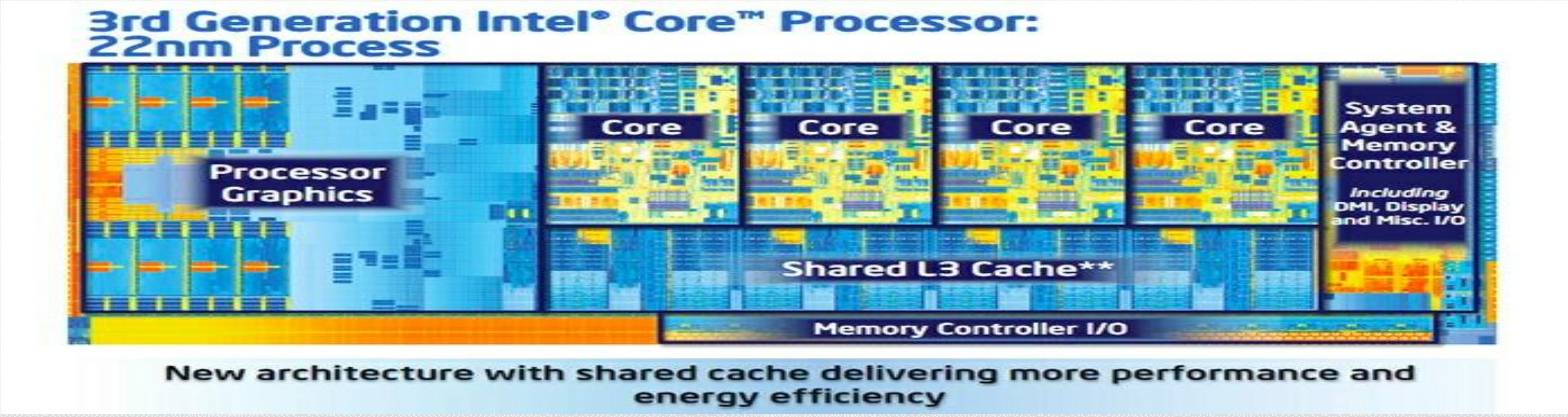
Parallel Trajectory Optimization

- Limitations of optimization-based algorithms
  - Performance – slow
  - Quality – Local optima may prevent planner to find a collision-free solution
- In real-time replanning, the performance is critical
  - Limited time to perform planning computations

Parallel Trajectory Optimization

- Parallel optimization of multiple trajectories
  - Use Multiple threads
    - Start from different initial trajectories
    - Trajectories are generated by quasi-random sampling
  - Exploits the multiple CPU cores (multi-cores) or GPU-based cores (many-cores)

Multi-Core CPUs



NVIDIA & AMD GPU Compute Accelerators

AMD Radeon 7970	NVIDIA GTX 680
3.79 Single Tflops	3.09 Single Tflops
947 Double Gflops	1.1 Double Tflops
2048 Stream Cores	1536 CUDA Cores

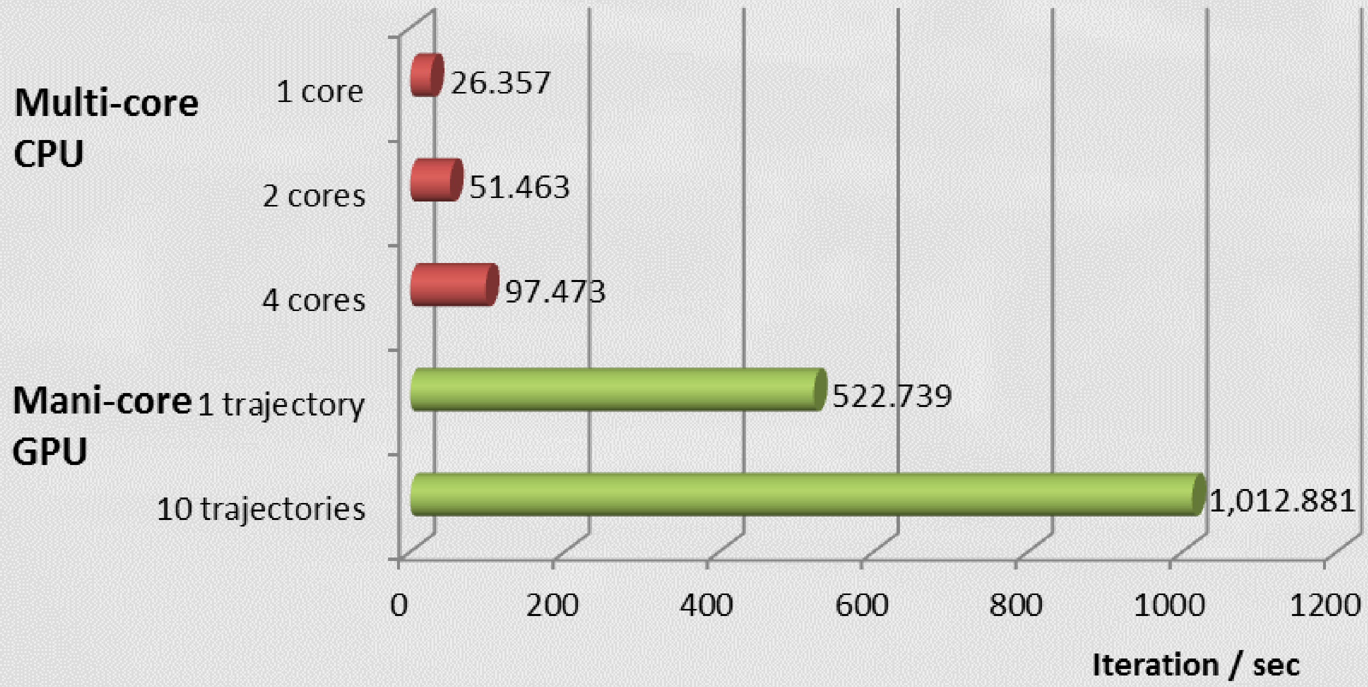
Commodity Tera-Flop Processor (peak performance)

Parallel Trajectory Optimization

- Parallelization improves the performance
  - Reduce the iteration time of the single optimization
  - Parallel optimization of multiple trajectories reduces t he time to compute the first collision-free solution

Parallel Trajectory Optimization

Performance improvement with number of cores

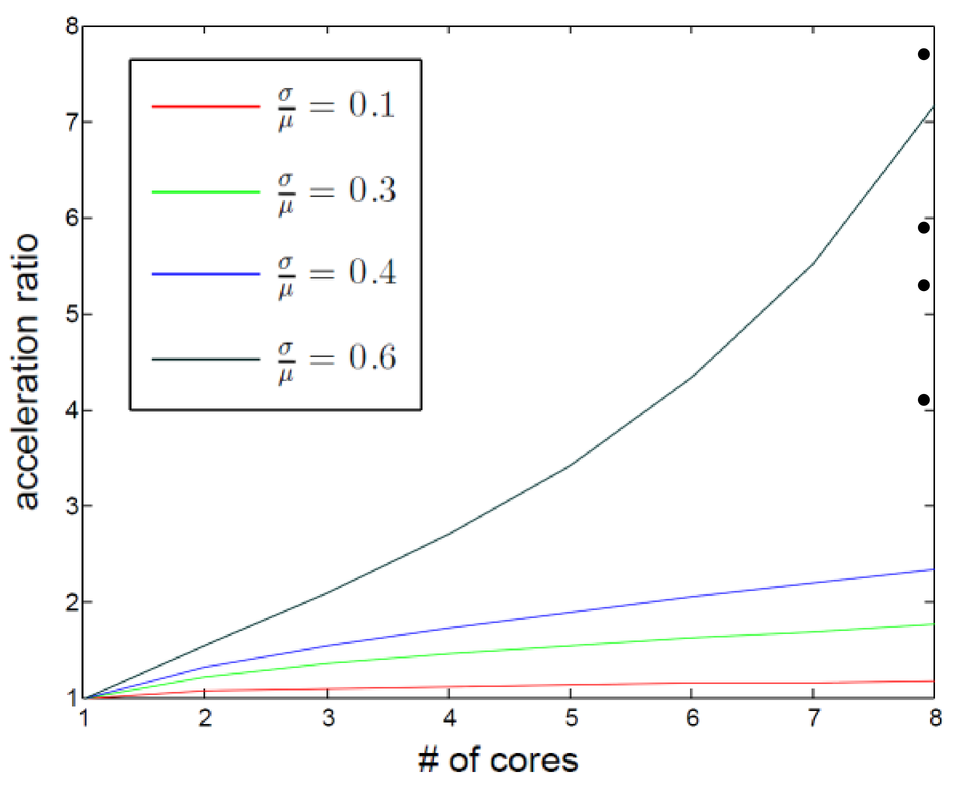


Parallel Trajectory Optimization

- Parallelization also improves the success rate
  - Each local minima is constrained to a subset of C-space
  - With more trajectories, the algorithm can explore a larger subset of C-space

Parallel Trajectory Optimization

- Acceleration in varying environments



- Assumes the time costs to compute a solution f ollow normal distribution.

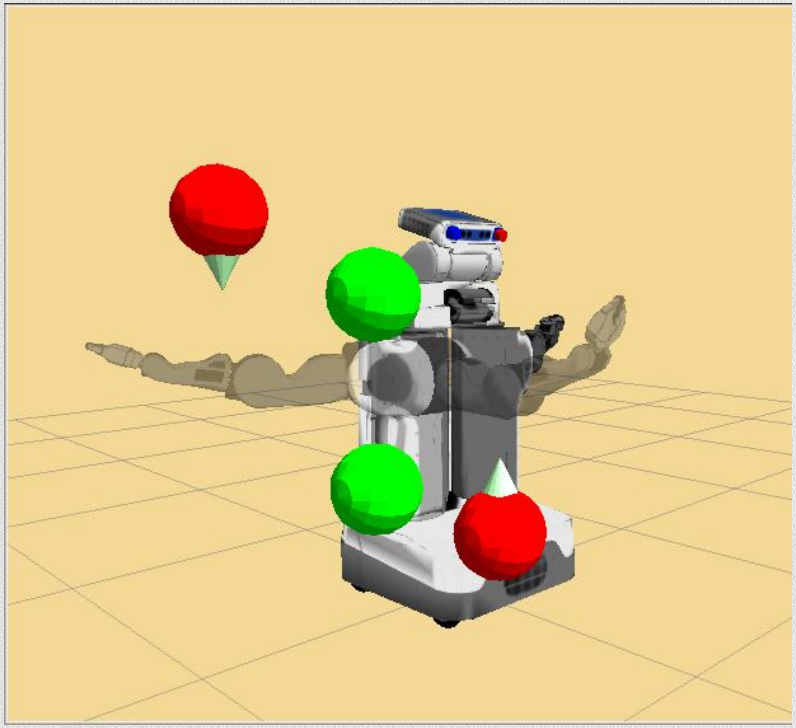
- Large μ : the environment is challenging
- Large σ^2 : the solver is sensitive to the initial va lues
- → Acceleration is large when the solver is more sensitive to the initial values.

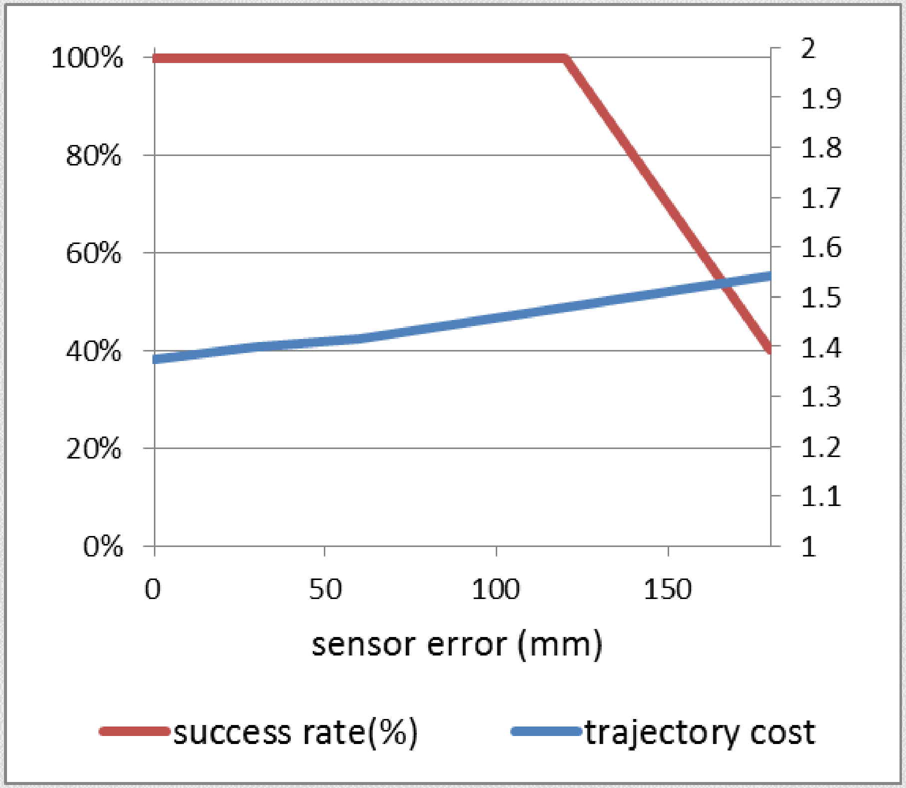
Results

- Implemented in ROS simulator
  - Willow Garage's PR2 robot model (two 7-DOF arms)
  - LIDAR sensor accuracy : 30mm
  - Update on dynamic obstacles (position and velocity): every 200ms

Results: Varying Sensor error

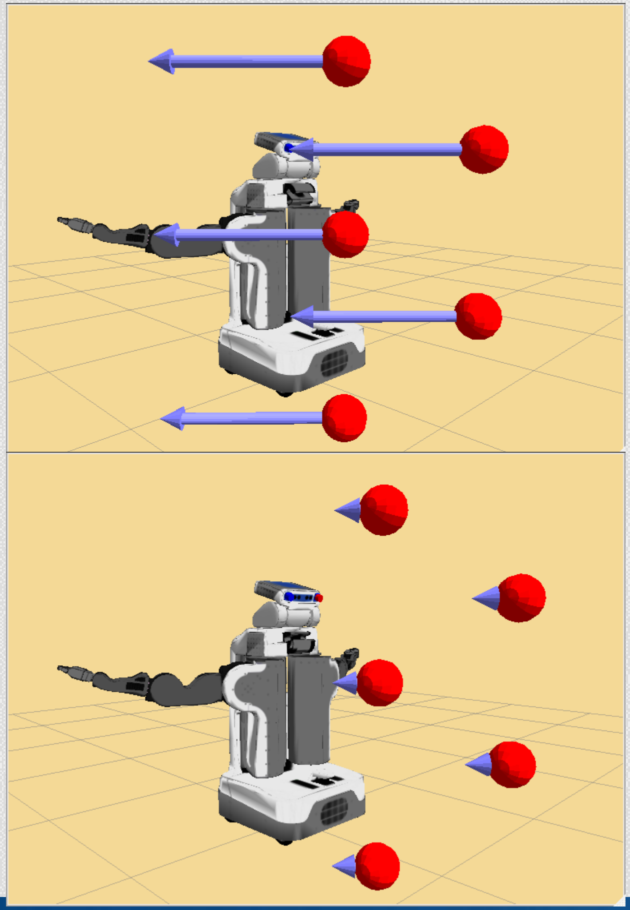
- Increase the sensor error in our simulation

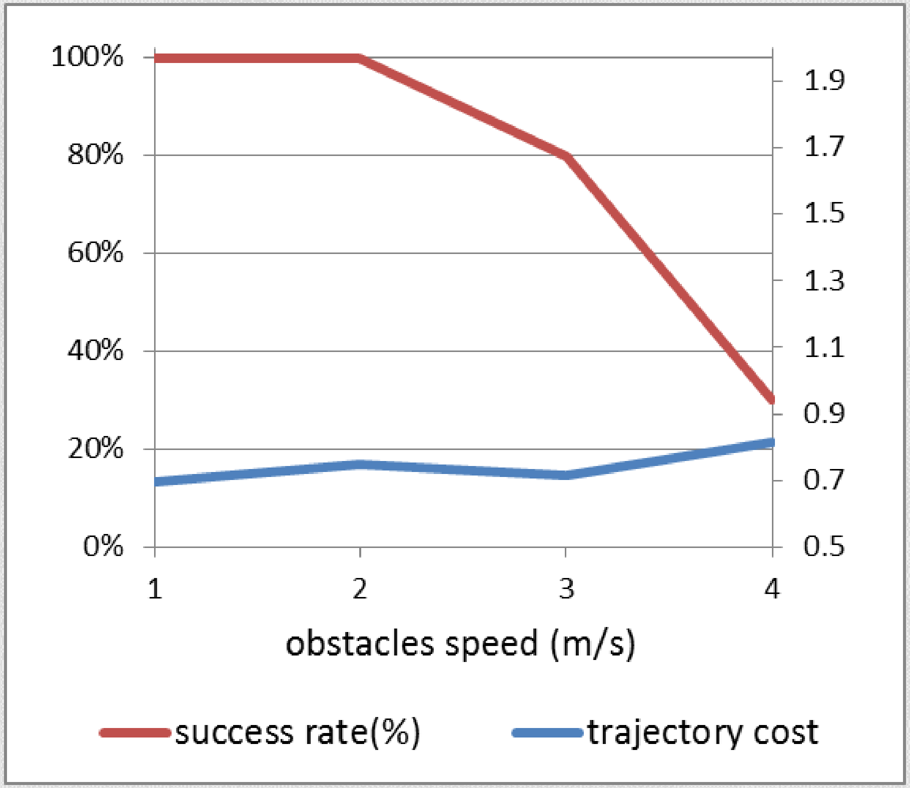




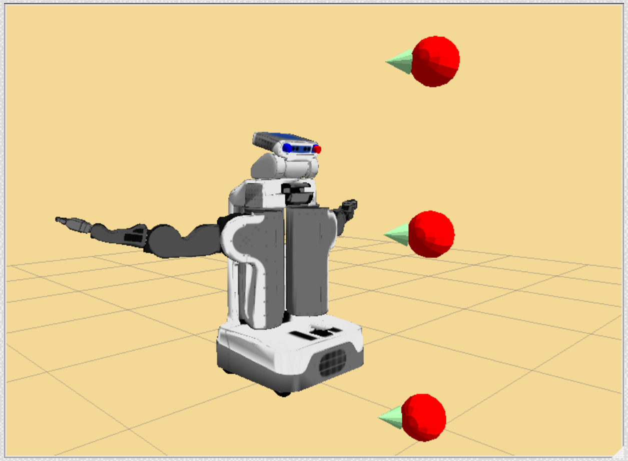
Varying the Obstacle Motion

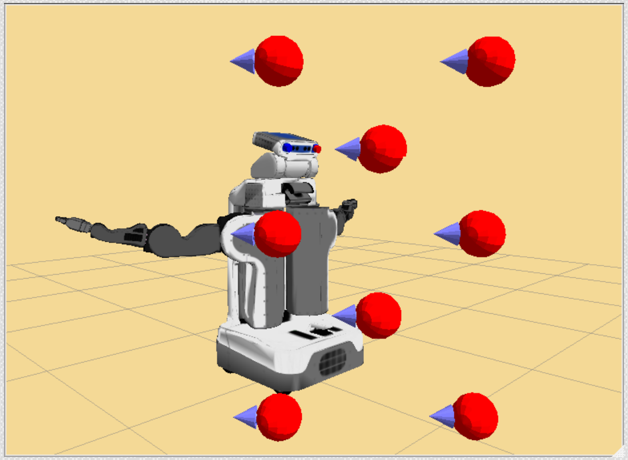
- Planning with varying obstacles speed

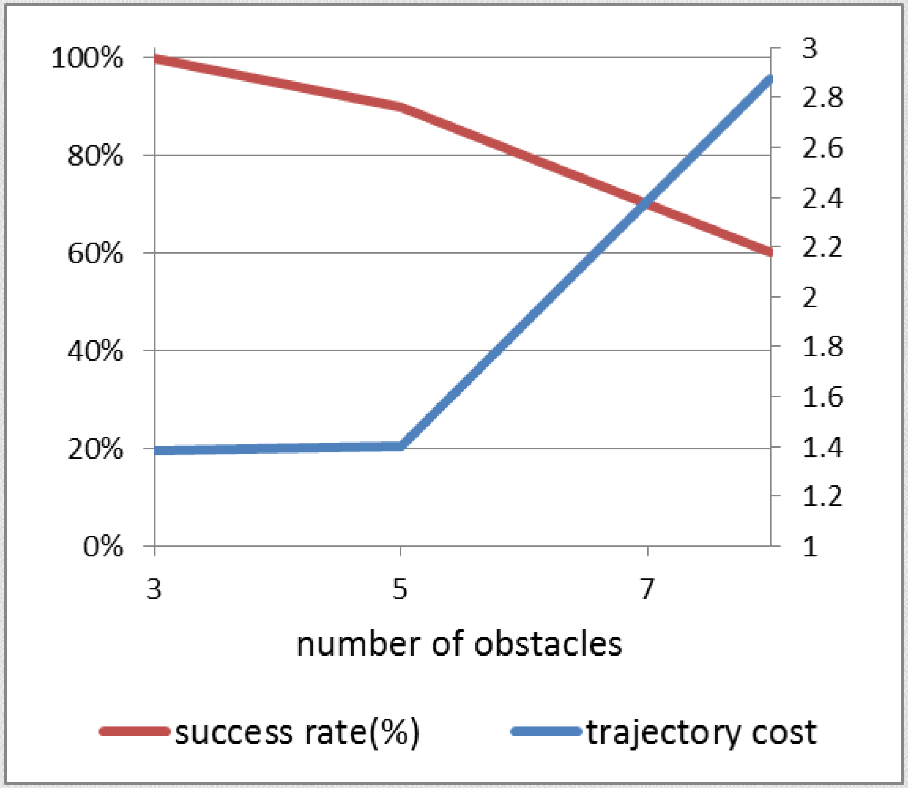




Varying the Number of Obstacles







Human-Like Environment: Simulation

Limitations

- Does not account for all sources of uncertainty
- Bounds on dynamic trajectory tend to be conservative
- Can't guarantee global optimal solutions
  - Sensitive to the choice of initial seed values

Conclusions

- Optimization-based motion planning algorithm for dynamic environments
  - General approach to compute smooth paths
  - No assumptions on obstacle motion
  - Real-time collision avoidance
  - Parallelization on multiple cores
  - Improved performance and path quality

Ackowledgements

- Army Research Office
- National Science Foundation
- Willow Garage