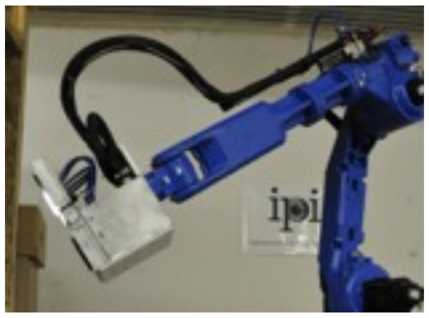
Finding Locally-Optimal, Collision-Free Trajectories with Sequential Convex Optimization

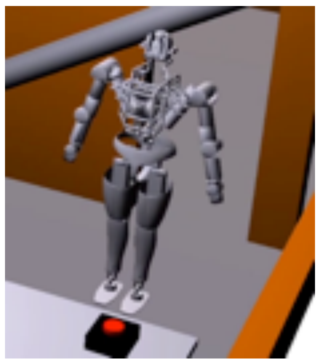
John Schulman, Jonathan Ho, Alex Lee,
Ibrahim Awwal, Henry Bradlow, Pieter Abbeel

I’m going to tell you about our work on using trajectory optimization for motion planning

Motion Planning


Industrial robot arm (6 DOF)

Mobile manipulator (18 DOF)


Humanoid (34 DOF)

Leading motion planning methods, based on RRT and A*, slow down a lot as the dimensionality of the state space increases.

Fast planning is especially important in situations with uncertainty, where you need to do frequent replanning

There's another class of methods based on optimization.

In robotics some of the most notable early work was on potential fields and elastic bands.

Then in 2009, CHOMP showed that trajectory optimization can work surprisingly well for planning from scratch,

We present a new trajectory optimization method.

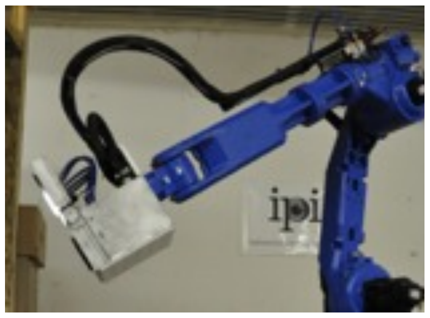
In our benchmarks we outperform both chomp and sampling based methods under all criteria: success rate, path length, and computation time.

Motion Planning

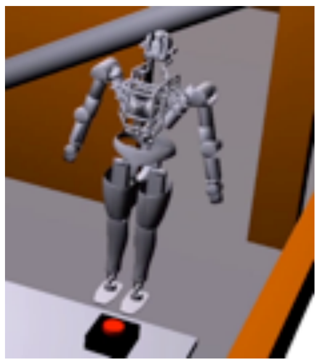
- Sampling-based methods like RRT
- Graph search methods like A*

} slow down as dimensionality increases

- Optimization based methods
  - Reactive control
    - Potential-based methods for high-DOF problems (Khatib, '86)
  - Optimize over the entire trajectory
    - Elastic bands (Quinlan & Khatib, '93)
    - CHOMP (Ratliff, et al. '09) & variants (STOMP, ITOMP)


Industrial robot arm (6 DOF)

Mobile manipulator (18 DOF)


Humanoid (34 DOF)

Leading motion planning methods, based on RRT and A*, slow down a lot as the dimensionality of the state space increases.

Fast planning is especially important in situations with uncertainty, where you need to do frequent replanning

There's another class of methods based on optimization.

In robotics some of the most notable early work was on potential fields and elastic bands.

Then in 2009, CHOMP showed that trajectory optimization can work surprisingly well for planning from scratch,

We present a new trajectory optimization method.

In our benchmarks we outperform both chomp and sampling based methods under all criteria: success rate, path length, and computation time.

Trajectory Optimization

$$\min_{\boldsymbol{\theta}_{1:T}} \sum_t \|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2 + \text{ other costs}$$

subject to

no collisions $\longleftarrow$ non-convex

joint limits

other constraints

Here's the problem formulation.

The variables are the degrees of freedom of the robot at every timestep, and we want to minimize the path length plus other costs, subject to no collisions, joint limits, and other constraints.

We use this formulation to solve motion planning problems from scratch, starting with infeasible initial trajectories

This problem is challenging because of the no collisions constraint, which is non-convex.

We use sequential convex optimization, which works by repeatedly constructing a convex approximation to the problem and solving it.

The key challenge is to approximate the collision constraint in a convex way.

Trajectory Optimization

$$\min_{\boldsymbol{\theta}_{1:T}} \sum_t \|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2 + \text{ other costs}$$

subject to

no collisions $\longleftarrow$ non-convex

joint limits

other constraints

- Sequential convex optimization
  - Repeatedly solve local convex approximation
- Challenge
  - Approximating collision constraint

Thursday, July 4, 13

Here's the problem formulation.

The variables are the degrees of freedom of the robot at every timestep, and we want to minimize the path length plus other costs, subject to no collisions, joint limits, and other constraints.

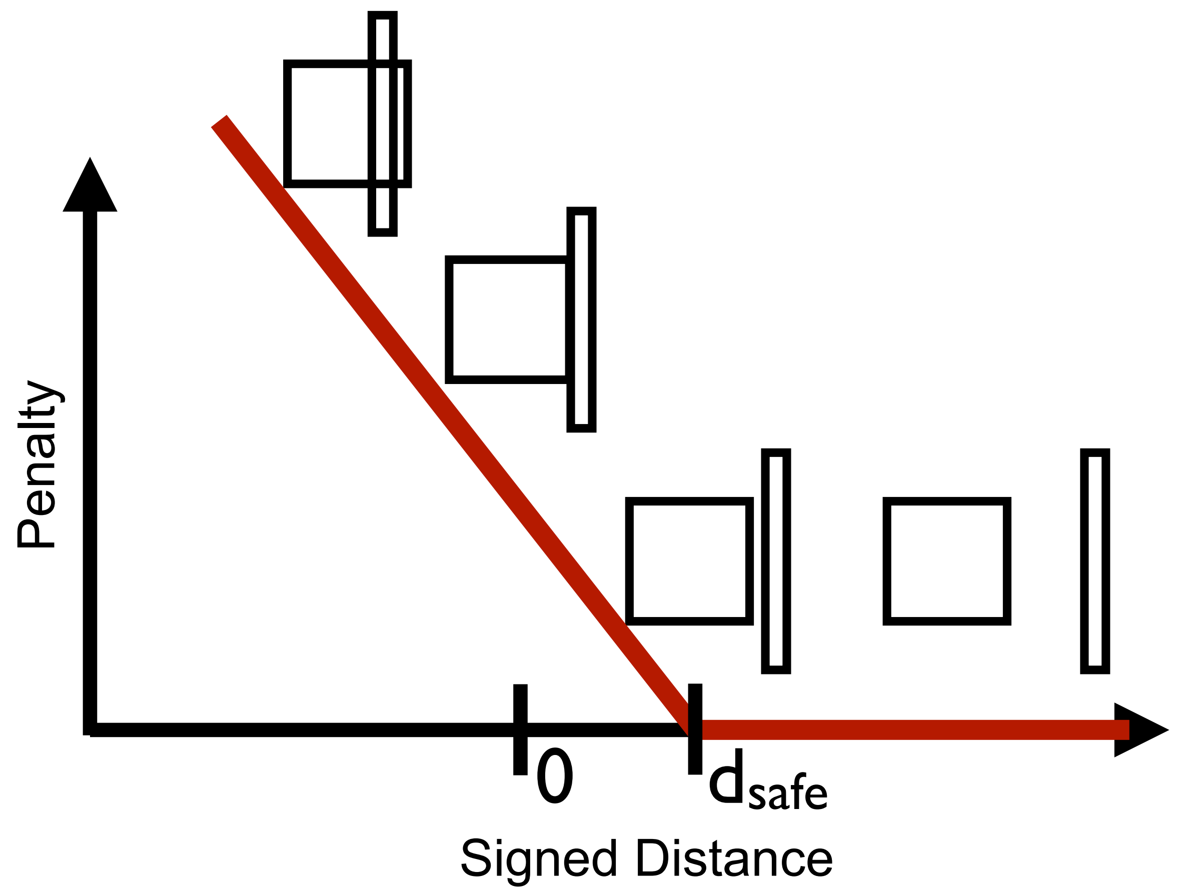
We use this formulation to solve motion planning problems from scratch, starting with infeasible initial trajectories

This problem is challenging because of the no collisions constraint, which is non-convex.

We use sequential convex optimization, which works by repeatedly constructing a convex approximation to the problem and solving it.

The key challenge is to approximate the collision constraint in a convex way.

Collision Constraint as L1 Penalty



Our algorithm is based on convex collision detection, which calculates the signed distance between pairs of primitive shapes using the GJK and EPA algorithms.

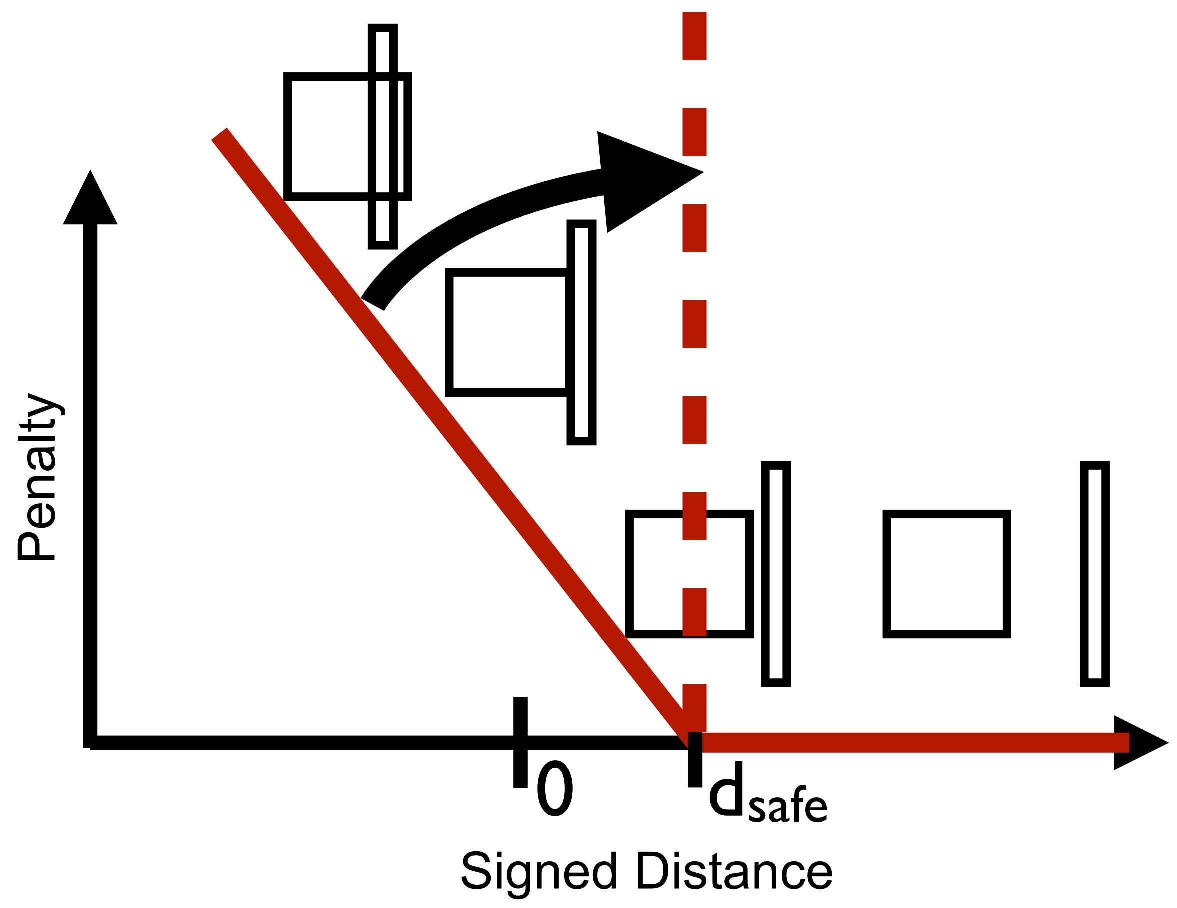
For two shapes that are separated, as shown on the right, the signed distance is positive, and for two overlapping shapes, as shown on the left, the signed distance is negative.

Our optimization procedure adds a penalty for each collision based on the signed distance; the penalty is shown by the red curve, and becomes nonzero when the distance is below a safety margin .

In our sequential convex optimization procedure, we linearize the signed distance of every overlapping pair of shapes with respect to the robot's degrees of freedom, and then we add the resulting hinge loss terms to the objective.

The slope of the penalty function is increased as needed in an outer loop of the algorithm.

Collision Constraint as L1 Penalty



Linearize w.r.t. degrees of freedom

$$\mathrm{sd}_{AB}(\boldsymbol{\theta}) \approx \mathrm{sd}_{AB}(\boldsymbol{\theta}_0) + \hat{\mathbf{n}}^T J_{\mathbf{p}_A}(\boldsymbol{\theta}_0)(\boldsymbol{\theta} - \boldsymbol{\theta}_0)$$

Our algorithm is based on convex collision detection, which calculates the signed distance between pairs of primitive shapes using the GJK and EPA algorithms.

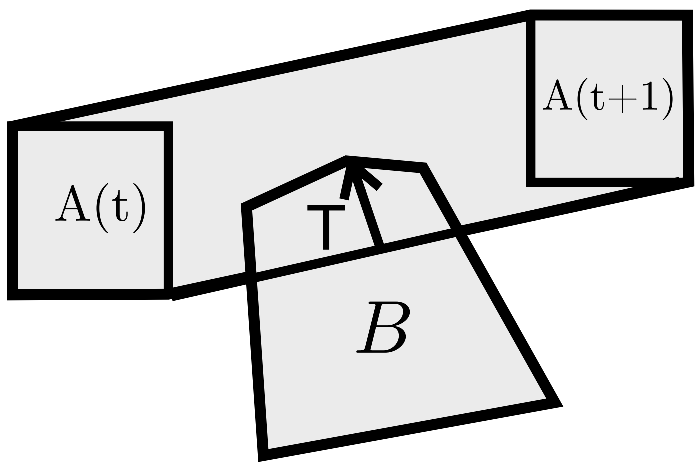
For two shapes that are separated, as shown on the right, the signed distance is positive, and for two overlapping shapes, as shown on the left, the signed distance is negative.

Our optimization procedure adds a penalty for each collision based on the signed distance; the penalty is shown by the red curve, and becomes nonzero when the distance is below a safety margin .

In our sequential convex optimization procedure, we linearize the signed distance of every overlapping pair of shapes with respect to the robot's degrees of freedom, and then we add the resulting hinge loss terms to the objective.

The slope of the penalty function is increased as needed in an outer loop of the algorithm.

Continuous-Time Safety



Collision check against swept-out volume

- Continuous-time collision avoidance
- Allows coarsely sampling trajectory
  - overall faster
- Finds better local optima

To ensure the continuous-time safety of the trajectory, we consider the shape swept out by the robot between timesteps, and we compute the signed distance between this swept shape and the obstacles.

That way, we can use a coarsely sampled trajectory but still ensure that it is safe in continuous time.

This type of collision checking is only slightly more expensive than the usual discrete-time version, but overall, the algorithm is much faster because we can use fewer timesteps.

As an added benefit, this formulation is very good at getting out of collision.

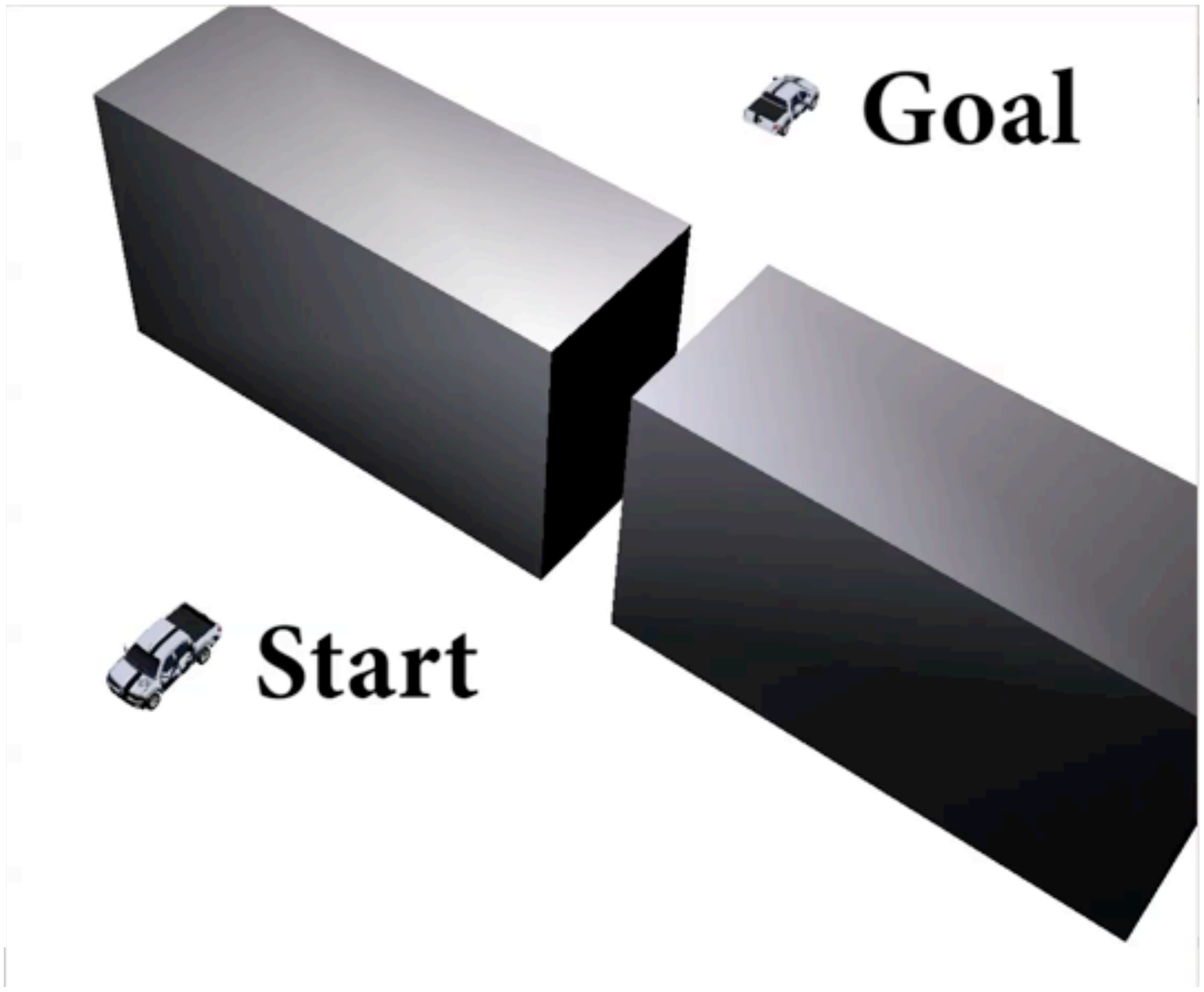
Optimization: Toy Example

Here's a toy example of a dubins car problem.

We've initialized with a straight line in configuration space that's both in collision and dynamically infeasible

The optimization gets it out of collision and fixes the constraint violation

Optimization: Toy Example



Here’s a toy example of a dubins car problem.

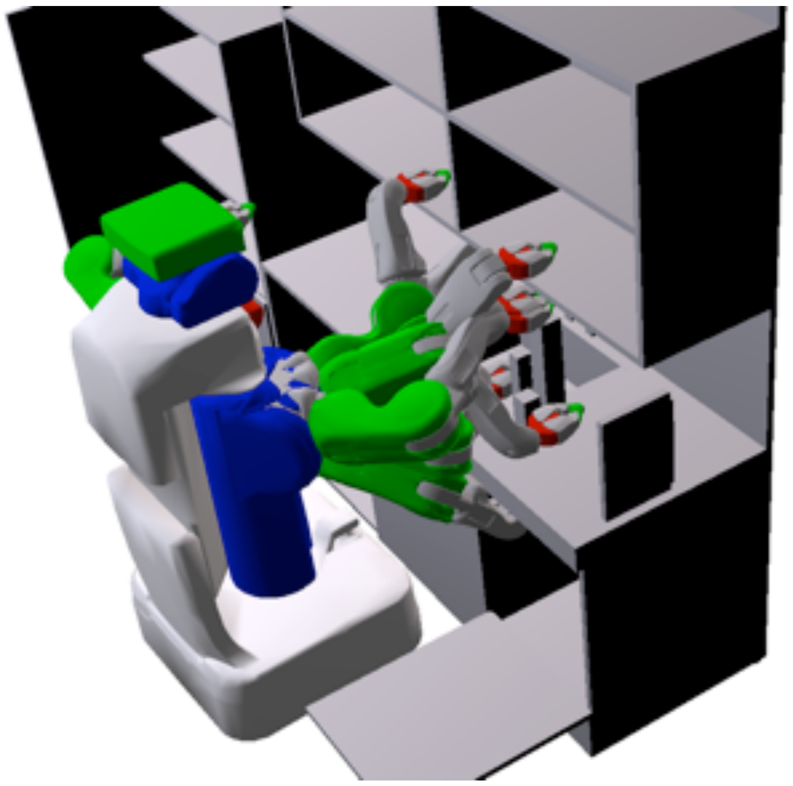
We’ve initialized with a straight line in configuration space that’s both in collision and dynamically infeasible

The optimization gets it out of collision and fixes the constraint violation

Benchmark: Example Scenes

7 DOF (one arm)

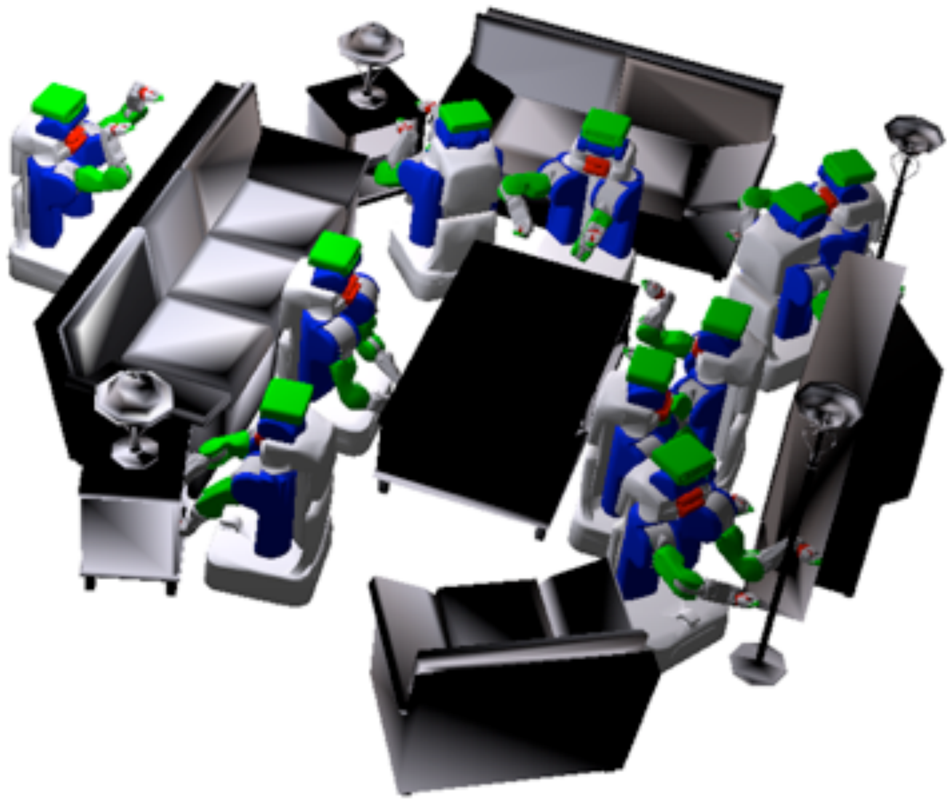
198 problems



example scene (taken from MoveIt collection)

18 DOF (two arms + base + torso)

96 problems



example scene (imported from Trimble 3d Warehouse / Google Sketchup)

Our benchmark consists of 198 7DOF arm planning problems and 96 18-DOF two-arm-base-torso planning problems with the pr2.

Benchmark Results

Arm planning (7 DOF) 10s limit			
	Trajopt	BiRRT (*)	CHOMP
success	99%	97%	85%
time (s)	0.32	1.2	6.0
path length	1.2	1.6	2.6

Full body (18 DOF) 30s limit			
	Trajopt	BiRRT (*)	CHOMP (**)
success	84%	53%	N/A
time (s)	7.6	18	N/A
path length	1.1	1.6	N/A

(*) Top-performing algorithm from MoveIt/OMPL
(**) Not supported in available implementation

Our algorithm, called “Trajopt” here, solved 99% of the 198 problems, in an average of .3 secs

The Bi-directional RRT in this table is a state of the art implementation from MoveIt and OMPL and includes a smoothing phase.

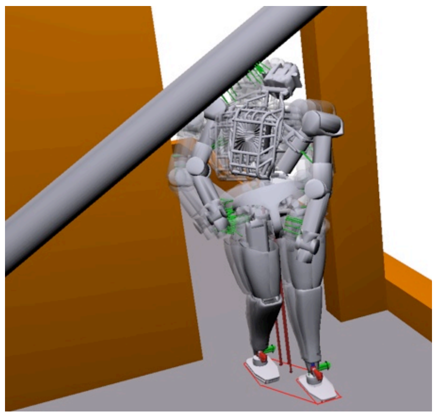
It solved about the same number of problems, but was more than 3 times slower, and generated longer paths.

CHOMP solved only 85% of the problems, was slower than our algorithm, and produced paths that are, on the average, more than twice as long.

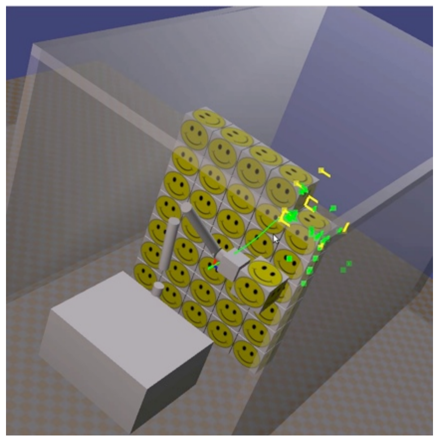
Our algorithm performed much better than the others others on the full body planning problems.

Other Experiments -- Videos at Interactive Session

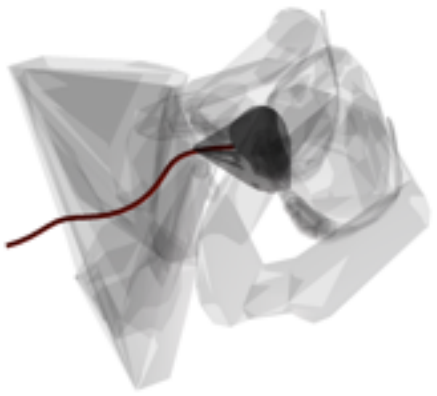
- Planning for 34-DOF humanoid (stability constraints)



- Box picking with industrial robot (orientation constraints)



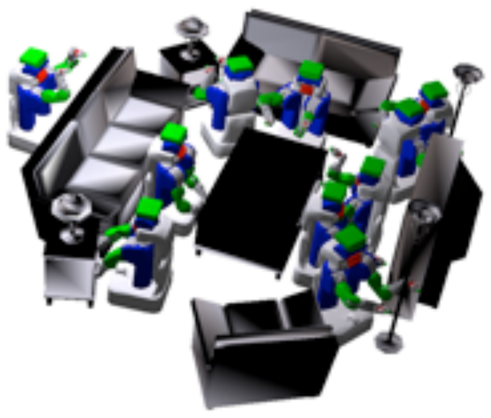
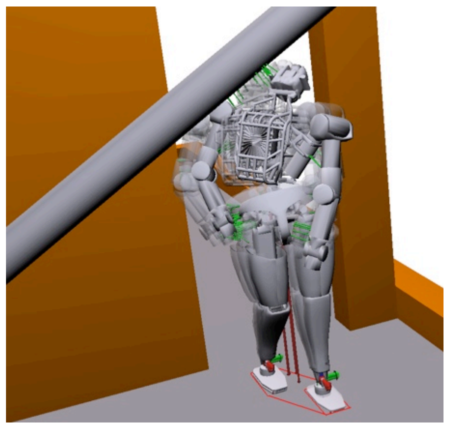
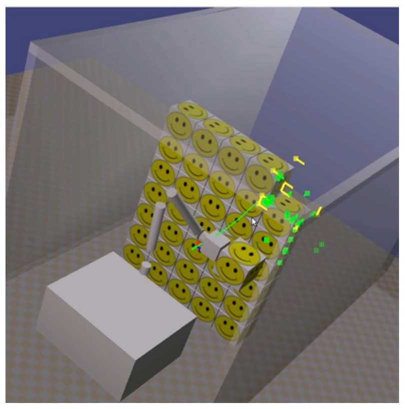
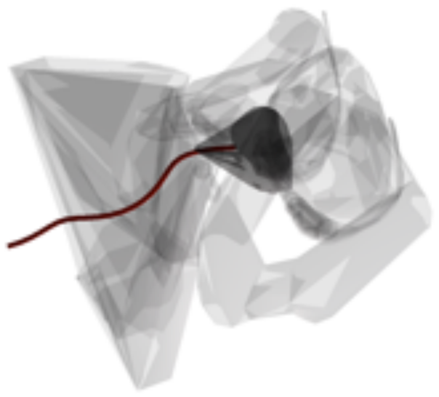
- Constant-curvature 3D needle steering (non-holonomic constraint)



We also applied our approach to a variety problems that include
stability constraints as was needed for a humanoid,
maintaining orientation constraints during manipulation,
and non-holonomic dynamics constraints in 3D needle steering.

Try it out yourself!

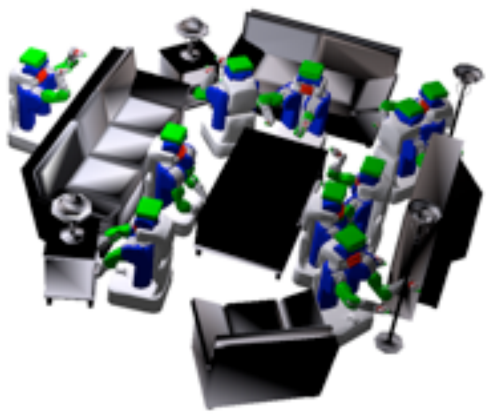
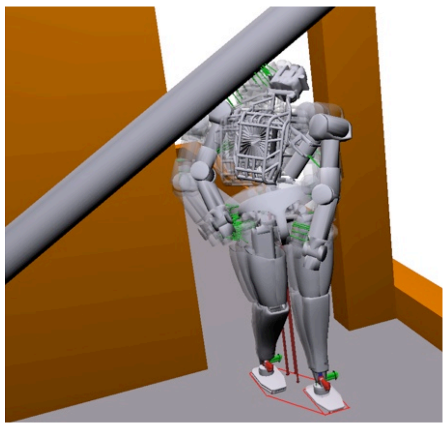
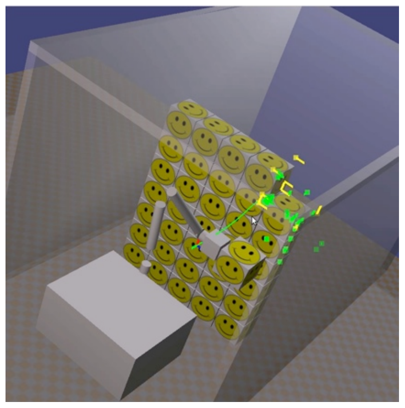
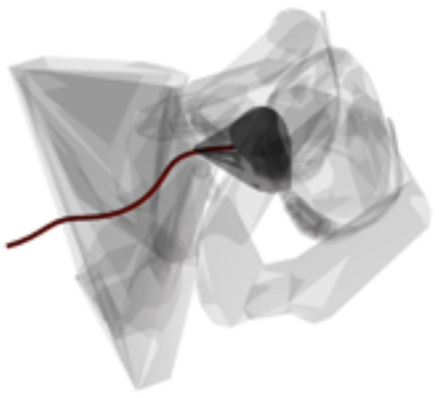
- Code and docs: rll.berkeley.edu/trajopt
- Run our benchmark: github.com/joschu/planning_benchmark

our complete source code for both trajopt and the benchmarks is available online with tutorials

Thanks!

- Code and docs: rll.berkeley.edu/trajopt
- Run our benchmark: github.com/joschu/planning_benchmark

Thursday, July 4, 13

thanks for your attention