Combining Motion Planning and Optimization for Flexible Robot Manipulation

Jonathan Scholz & Mike Stilman, ICHR 2010

Motivation:

Robots face two challenges in natural environments:

- *Underspecified goals:* no human to specify exact goal configuration
- *Uncertain dynamics:* Effects of robot's actions on novel objects is uncertain

Approach:

- For underspecified goals:
  - *Pose task as a constrained optimization problem over a set of reward or cost terms.*
  - *Can be defined manually or modeled from human*
- For uncertain dynamics:
  - *Quickly approximate dynamics for a set of actions*
  - *Plan efficiently using sampling-based techniques*

Our algorithm:

- Searches in *object* configuration space using Rapidly-exploring Random Trees (RRT)
- Adds leaves to search tree by forward-simulating the learned dynamics for each object-action pair
- Uses directGD heuristic to quickly search optimization landscape
- Returns a plan from the starting state to the most optimal reachable state, given cost function

```
TASK_SPACE_RRT(s_init, A)
for i ← 1 to |O|
do Model ← LEARN_MODEL(O_i, A, s_init)
T.init(s_init)
for i ← 1 to max_nodes
do s_GD ← DIRECTGD(T)
    if RAND() > ϵ
        then s_samp ← s_GD
        else  s_samp ← RANDOM_CONFIG()
    s_near ← NEAREST_NEIGHBOR(s_samp)
    a* ← ARGMIN_a(ρ (MODEL(s_near, a), s_samp))
    s_new ← MODEL(s_near, a*)
    if not IN_COLLISION(s_new)
        then ADD_VERTEX(s_new)
             ADD_EDGE(s_near →(a*) s_new)
```

Pseudo code for TS-RRT algorithm

Manipulation under uncertainty

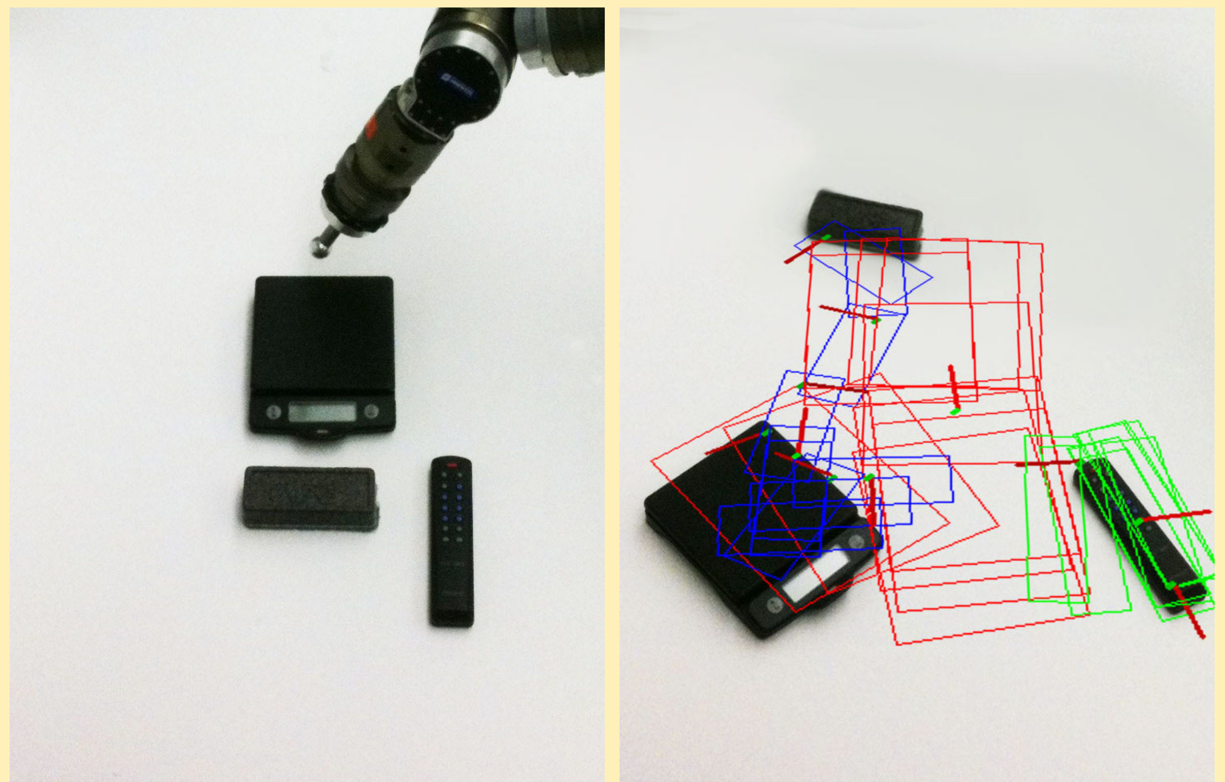
Initial state

- Robot begins with a workspace containing three unfamiliar objects
- Robot provided a cost function expressing the following desiderata:
  - *Orthogonality*
  - *Cicumscribed area*
  - *Distance from edge of workspace*

Solution

- All objects pushed to orthonormal orientations in the center of the workspace
- All paths free of collisions and redundant actions
- Robot monitored error and replanned as necessary

Random initial configuration

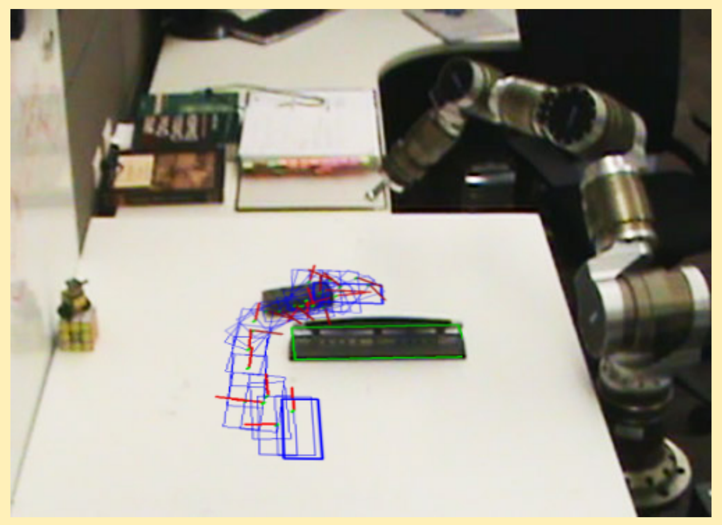


Optimized final configuration

Generalization to other manipulation tasks

Appropriate for tasks naturally expressed as optimization of a cost function:

- *Arranging clutter on a surface*
- *Multiple object placement*
- *Table setting*



In the presence of fixed obstacles

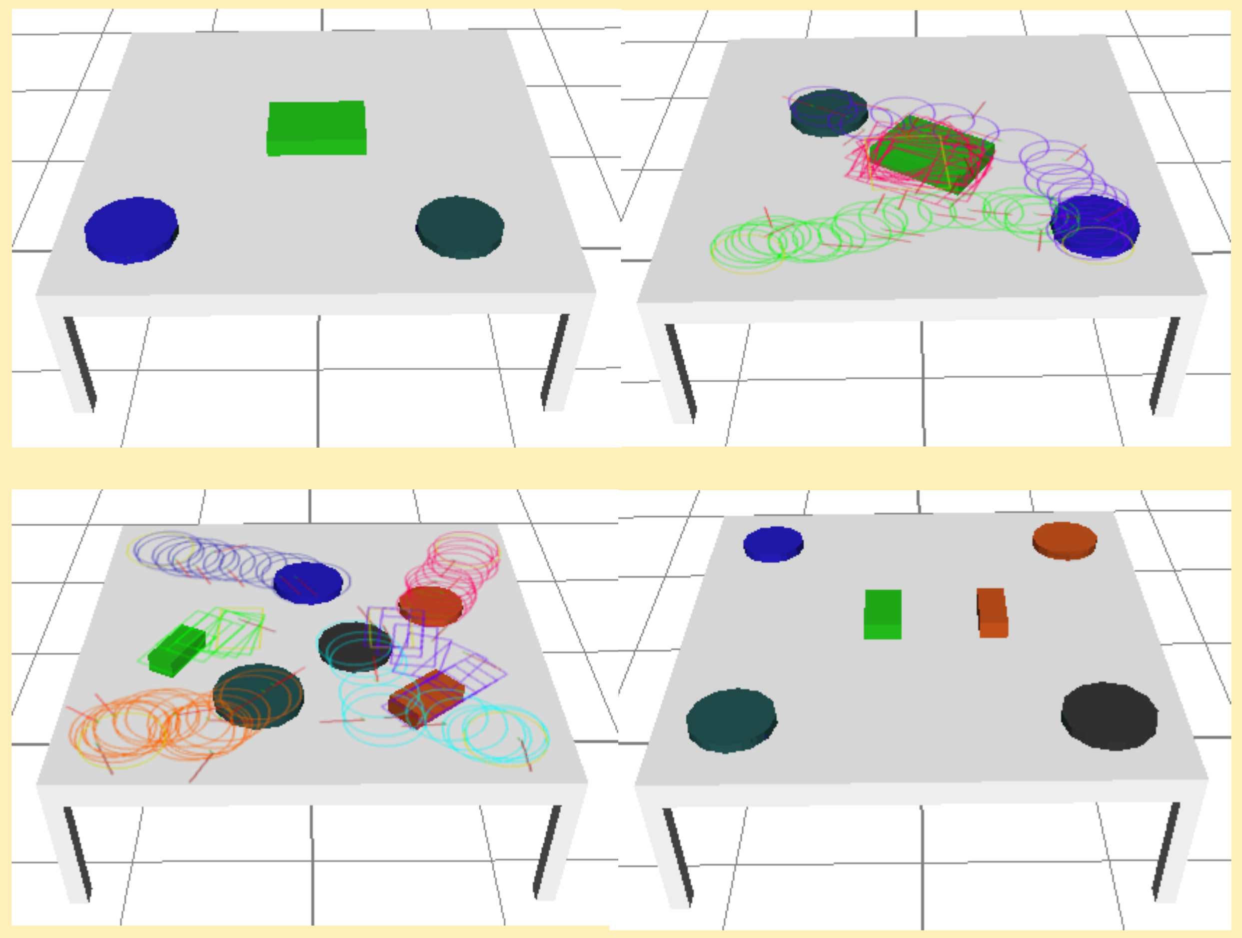
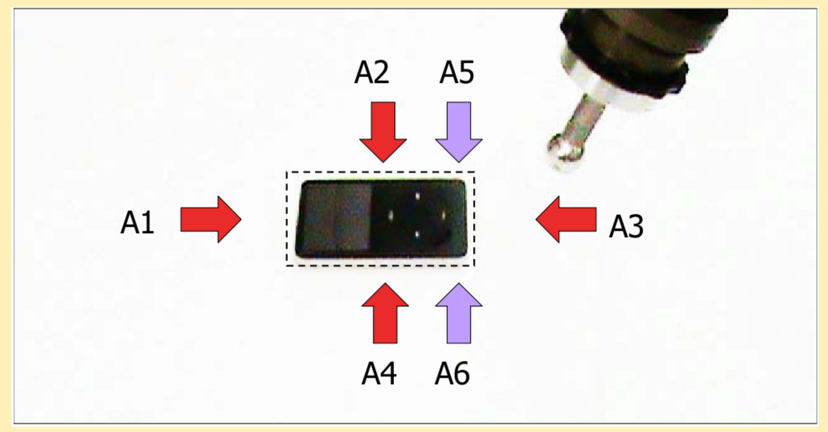


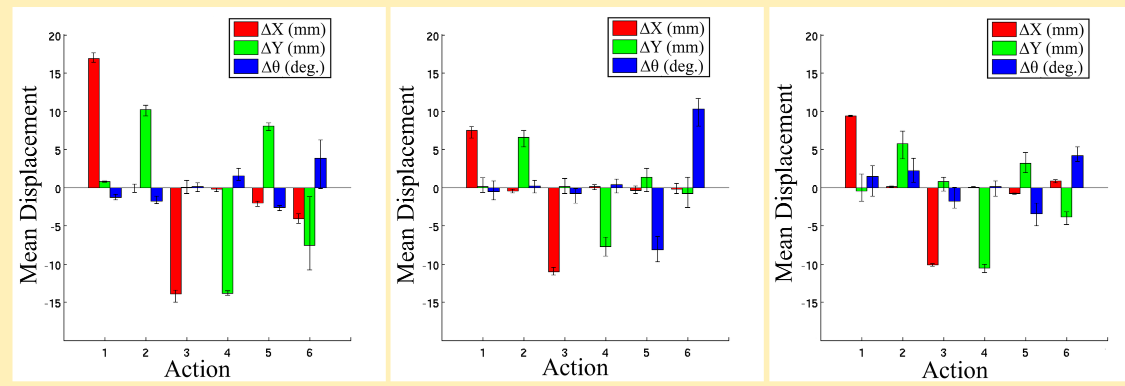
Table-setting with arbitrary numbers of objects

Model Learning

Goal: discover the dynamics of each object class over a set of action primitives



Six action primitives defined for interacting with objects in the robot's workspace



Model learning results from three object types, depicting mean and 95% CI for object displacement in each workspace dimension

```
LEARN_MODEL(o, A, s_init)
s = s_init
for i ← 1 to |A|
do while σ² > σ²_ref
    do (Δs, s) = APPLY_ACTION(a_i, O)
        P(Δs|a_i, o) = UPDATEDISTR(P(Δs|a_i, o), a_i, Δs)
        σ² ← VARIANCE(P(Δs|a_i, o))
```

Model-learning procedure

Advantages

Appropriate for tasks naturally expressed as optimization of a cost function:

- Unlike conventional single-shot methods, doesn't require user specified goals
- Always guaranteed to return reachable solution
- Favorable anytime characteristics
- Feasible for real-time planning in high DOF problems

Similar to Reinforcement Learning formalism, but trades path optimality for realtime feasibility

- RL can require many full-passes through configuration space to converge to optimal policy
- Handling continuous features requires discretization, tiling, or other appoaches
- RL better suited for problems with sparse reward landscape, but optimizations offer a gradient (like shaping reward) which allows fast heuristic search with RRT