From trajectory optimization to inverse KKT and sequential manipulation

Marc Toussaint

Machine Learning & Robotics Lab – University of Stuttgart
marc.toussaint@informatik.uni-stuttgart.de

Zurich, July 2016

- Motivation:
  - Combined Task and Motion Planning
  - Learning Sequential Manipulation from Demonstration

- Approach: Optimization

- Outline
  (1) k-order Markov Path Optimization (KOMO)
  (2) Learning from demonstration – Inverse KKT
  (3) Cooperative Manipulation Learning
  (4) Logic-Geometric Programming

(1) k-order Markov Path Optimization (KOMO)

- Actually, there is nothing “novel” about this, except for the specific choice of conventions. Just Newton ($\sim$1700).
  Still, it generalizes CHOMP and many others...

Conventional Formulation

- Given a time discrete controlled system $x_{t+1} = f(x_t, u_t)$, minimize

$$\min \sum_{t=1}^{T} c_t(x_t, u_t) \quad \text{s.t.} \quad x_{t+1} = f(x_t, u_t)$$

  - Indirect methods: optimize over $u_{0:T\text{-}1}$ $\rightarrow$ shooting to recover $x_{1:T}$
  - Direct methods: optimize over $x_{1:T}$ subject to existence of u_t

- Standard approaches
  - Differential Dynamic Programming, iLQG, Approximate Inference Control
  - Newton steps, Gauss-Newton steps
  - SQP

KOMO formulation

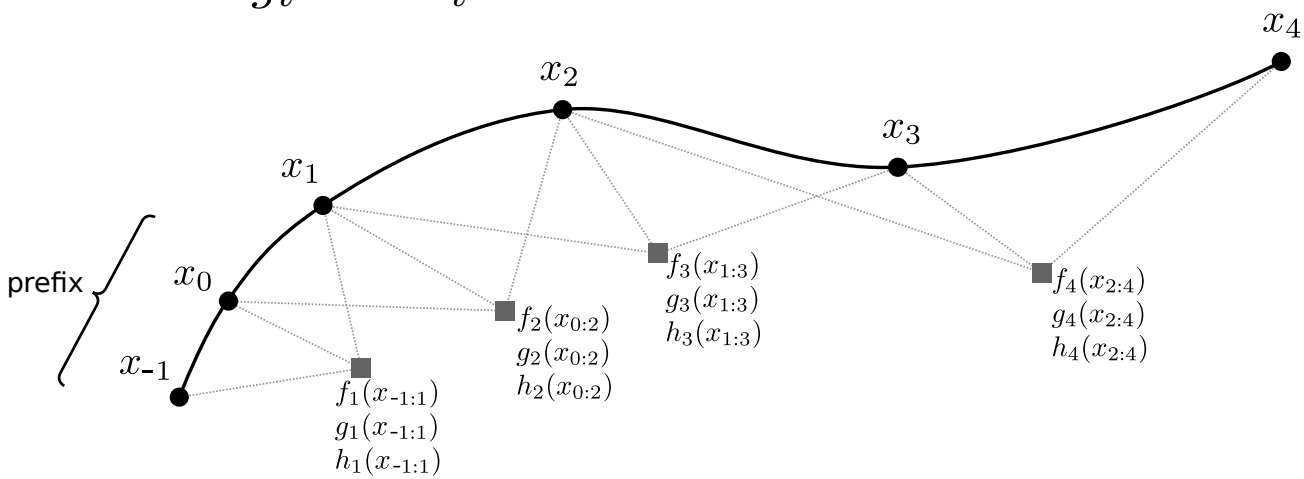
- We represent x_t in *configuration* space. $\rightarrow$ We have k-order Markov dynamics $x_t = f(x_{t-k:t\text{-}1}, u_{t\text{-}1})$

KOMO formulation

- We represent x_t in *configuration* space. $\rightarrow$ We have k-order Markov dynamics $x_t = f(x_{t-k:t\text{-}1}, u_{t\text{-}1})$
- k-order Motion Optimization (KOMO)

$$\min_x \sum_{t=1}^{T} f_t(x_{t-k:t}) \quad \text{s.t.} \quad \forall_{t=1}^{T}: \quad g_t(x_{t-k:t}) \leq 0\ , \quad h_t(x_{t-k:t}) = 0$$

for a path $x \in \mathbb{R}^{T\times n}$, prefix $x_{k\text{-}1:0}$, smooth scalar functions f_t, smooth vector functions g_t and h_t.



KOMO formulation

- The path costs are typically sum-of-squares, e.g.,
$$f_t(x_{t-k:t}) = \|M(x_t + x_{t\text{-}2} - 2x_{t\text{-}1})/\tau^2 + F\|_H^2.$$
- The equality constraints typically represent non-holonomic/non-trivial dynamics, and hard task constraints, e.g., $h_T(x_T) = \phi(x_T) - y_t^*$.
- The inequality constraints typically represent collisions & limits.

The structure of the Hessian

- The Hessian in the inner loop of a constrained solver will contain terms

$$\nabla^2 f(x) \ , \quad \sum_j \nabla h_j(x) \nabla h_j(x)^\top \ , \quad \sum_i \nabla g_i(x) \nabla g_i(x)^\top$$

- **The efficiency of optimization hinges on whether we can efficiently compute Newton steps with such Hessians!**

The structure of the Hessian

- The Hessian in the inner loop of a constrained solver will contain terms

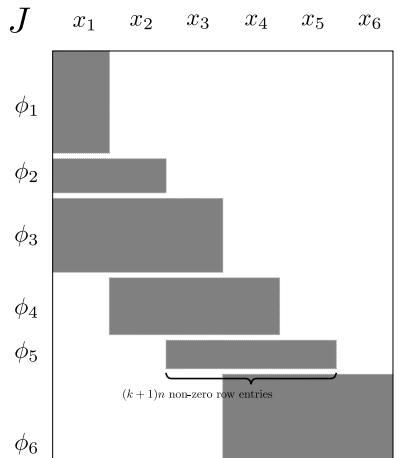
$$\nabla^2 f(x)\ , \quad \sum_j \nabla h_j(x) \nabla h_j(x)^\top\ , \quad \sum_i \nabla g_i(x) \nabla g_i(x)^\top$$

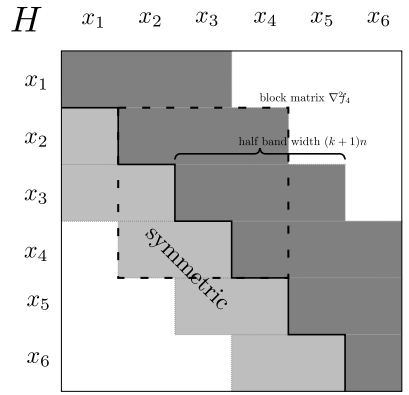
- **The efficiency of optimization hinges on whether we can efficiently compute Newton steps with such Hessians!**
- Properties:
  - *The matrix $J(x)^\top J(x)$ is banded symmetric with width $2(k+1)n-1$.*
  - *The Hessian $\nabla^2 f(x)$ is banded symmetric with width $2(k+1)n-1$.*
  - *The complexity of computing Newton steps is $O(Tk^2n^3)$.*
  - Computing a (Gauss-)Newton step in $O(T)$ is "equivalent" to a DDP (Riccati) sweep.

$$\phi_t(x_{t-k:t}) \triangleq \begin{pmatrix} f_t(x_{t-k:t}) \\ g_t(x_{t-k:t}) \\ h_t(x_{t-k:t}) \end{pmatrix}$$

$$\phi(x) = \bigotimes_{t=1}^{T} \phi_t(x_{t-k:t})$$

$$J(x) = \frac{\partial \phi(x)}{\partial x}$$





Augmented Lagrangian

- Define the Augmented Lagrangian

$$\hat{L}(x) = f(x) + \sum_j \kappa_j h_j(x) + \sum_i \lambda_i g_i(x) + \nu \sum_j h_j(x)^2 + \mu \sum_i [g_i(x) > 0]\ g_i(x)^2$$

- Centered updates:

$$\kappa_j \leftarrow \kappa_j + 2\nu h_j(x')\ , \quad \lambda_i \leftarrow \max(\lambda_i + 2\mu g_i(x'), 0)$$

(Hardly mentioned in the literature; analyzed in..)

Toussaint: *A Novel Augmented Lagrangian Approach for Inequalities and Convergent Any-Time Non-Central Updates*. arXiv:1412.4329, 2014

- In practise: typically, the first iteration dominates computational costs, which is conventional *squared penalties* $\rightarrow$ hand-tune scalings of h and g for fast convergence in practise. Later iterations do not change conditioning (!) and make constraints precise.

Toussaint: *KOMO: Newton methods for k-order Markov Constrained Motion Problems*. arXiv:1407.0414, 2014

Further Comments

- *Unconstrained* KOMO is a factor graph $\rightarrow$ solvable by standard Graph-SLAM solvers (GTSAM). This outperforms CHOMP, TrajOpt by orders of magnitude. (R:SS'16, Boots et al.)
- CHOMP = include only transition costs in the Hessian. Otherwise it's just Newton.
- We can include a large-scale ($> k$-order) smoothing objective, equivalent to a Gaussian Process prior over the path, still $O(T)$.
- Approximate (fixed Lagrangian) constrained MPC regulator (acMPC) around the path:

$$\pi_t : x_{t-k:t\text{-}1} \mapsto \underset{x_{t:t+H}}{\operatorname{argmin}} \Big[\sum_{s=t}^{t+H\text{-}1} f_s(x_{s-k:s}) + J_{t+H}(x_{t+H-k:t+H\text{-}1}) + \varrho \| x_{t+H} - x^*_{t+H} \|^2 \Big]$$
$$\text{s.t.} \quad \forall_{s=t}^{t+H\text{-}1} : \; g_s(x_{s-k:s}) \leq 0, \; h_s(x_{s-k:s}) = 0$$

Toussaint—in preparation: *A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian Process smoothing, and probabilistic inference*. Book chapter

Nathan's work

- Differential-geometric interpretation. Online MPC.

  Ratliff, Toussaint, Bohg, Schaal: *On the Fundamental Importance of Gauss-Newton in Motion Optimization*. arXiv:1605.09296

  Ratliff, Toussaint, Schaal: *Understanding the geometry of workspace obstacles in motion optimization*. ICRA'15

  Doerr, Ratliff, Bohg, Toussaint, Schaal: *Direct loss minimization inverse optimal control*. R: SS'15

Why care about this?

- Actually we care about higher-level behaviors
  - Sequential Manipulation
  - Learning/Extracting Manipulation Models from Demonstration
  - Reinforcement Learning of Manipulation
  - Cooperative Manipulation (IKEA Assembly)

- In all these cases, *KOMO became our underlying model of motion*
  - E.g., we parameterize the objectives f, and learn these parameters
  - E.g., we view sequential manipulation as logic+KOMO

(2) Learning Manipulation Skills from Single Demonstration

Research Questions

- The policy (space of possible manipulation) is high-dimensional..
  - Learning from a *single* demonstration and few own trials?

- What is the prior?

- How to generalize? What are the relevant implicit tasks/objectives? (Inverse Optimal Control)

Sample-efficient (Manipulation) Skill Learning

- Great existing work in policy search
  - Stochastic search (CMA, PI^2), “trust region” optimization (REPS)
  - Bayesian Optimization
  - Not many demonstrations on (sequential) manipulation

- These methods are good – but on what level do they apply?
  - Sample-efficient only in low dimensional policies (No Free Lunch)
  - Can’t we identify more structure in demonstrated manipulations?
  - Can’t we exploit partial models – e.g. of robot’s own kinematics? (not environment!)

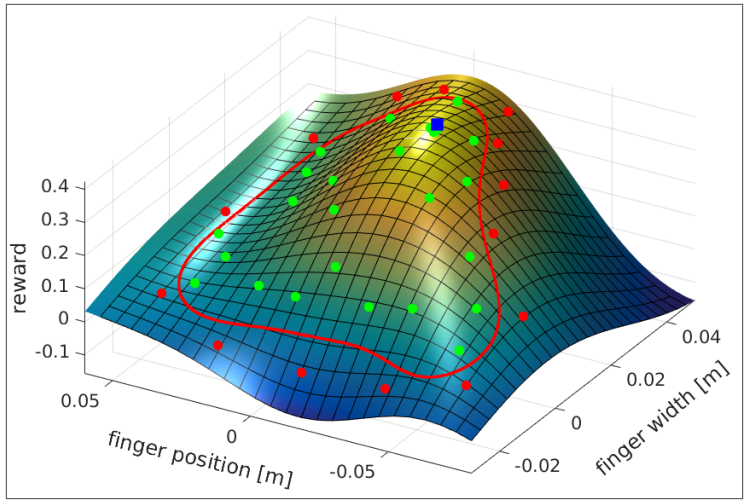
A more structured Manipulation Learning formulation

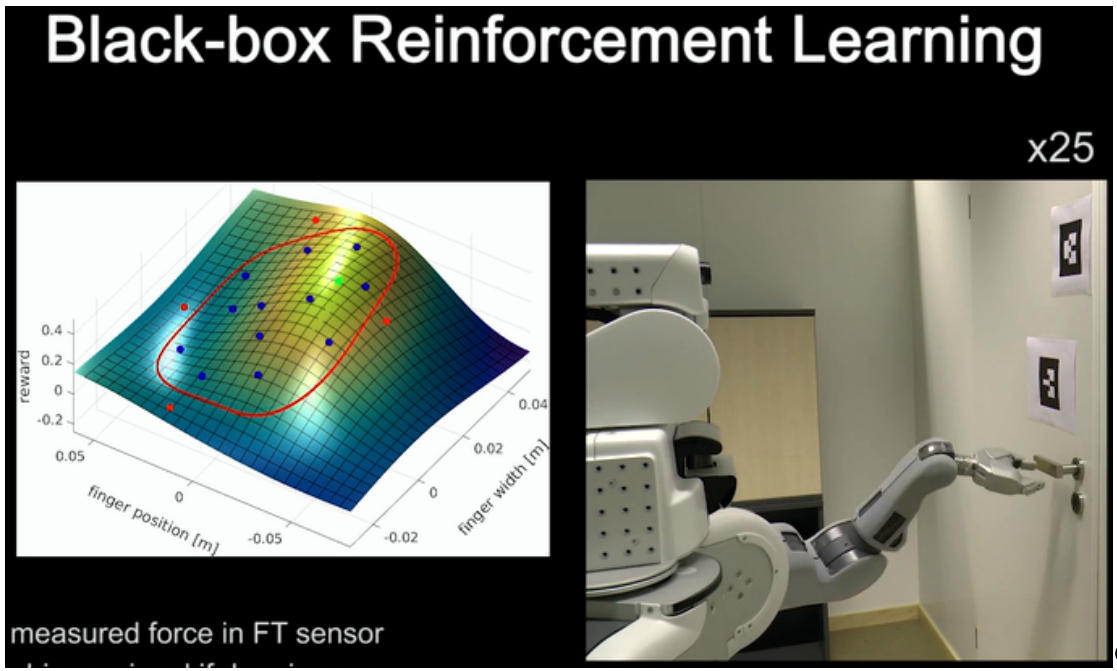
Engert & Toussaint: *Combined Optimization and Reinforcement Learning for Manipulation Skills*. R:SS'16

- CORL:
  - Policy: (controller around a) path x
  - analytically known cost function $f(x)$ in **KOMO** convention
  - **projection**, implicitly given by a constraint $h(x, \theta) = 0$
  - unknown black-box return function $R(\theta) \in \mathbb{R}$
  - unknown black-box success constraint $S(\theta) \in \{0, 1\}$
  - Problem:

$$\min_{x,\theta} f(x) - R(\theta) \quad \text{s.t.} \quad h(x, \theta) = 0,\ S(t) = 1$$

- Alternate path optimization $\min_x f(x)$ s.t. $h(x, \theta) = 0$
  with Bayesian Optimization $\max_\theta R(\theta)$ s.t. $S(\theta) = 1$


reward
0.4
0.3
0.2
0.1
0
-0.1
0.05
0
-0.05
finger position [m]
0.04
0.02
0
-0.02
finger width [m]


Black-box Reinforcement Learning
x25
reward
0.4
0.2
0
-0.2
0.05
0
-0.05
finger position [m]
0.04
0.02
0
-0.02
finger width [m]
measured force in FT sensor

Caveat

- The projection h, which defines θ, needs to be known!

- But is this really very unclear?
  - θ should capture all aspects we do not know apriori in the KOMO
  - We assume the robot's own DOFs kinematics/dynamics, and control costs are known
  - What is not known is how to **interact** with the environment
  - θ captures the interaction parameters: points of contact, amount of rotation/movement of external DOFs

And Generalization?

- The above *reinforces* a single demonstration
- Generalization means to capture/model the underlying task

Inverse KKT to gain generalization

- We take KOMO as the *generative assumption* of demonstrations

$$\min_x \sum_{t=1}^{T} f_t(x_{t-k:t}) \quad \text{s.t.} \quad \forall_{t=1}^{T}: \quad g_t(x_{t-k:t}) \leq 0 \ , \quad h_t(x_{t-k:t}) = 0$$

- Problem:
  - Infer f_t from demonstrations
  - We assume $f_t = w_t \circ \Phi_t$ (weighted features).
  - *Invert the KKT conditions* $\rightarrow$ QP over w's

Englert & Toussaint: *Inverse KKT – Learning Cost Functions of Manipulation Tasks from Demonstrations*. ISRR'15

Details

- Given a large set of potential cost features Φ, we parameterize

$$f(x_{0:T}, w) = \sum_t f_t(x_{t-k:t})^\top f_t(x_{t-k:t}) = \Phi(x_{0:T})^\top \operatorname{diag}(w)\ \Phi(x_{0:T})$$

- Lagrangian

$$L(x_{0:T}, \lambda, w) = f(x_{0:T}, w) + \lambda^\top \begin{pmatrix} g(x_{0:T}) \\ h(x_{0:T}) \end{pmatrix}$$

- 1st KKT condition

$$0 = \nabla_{x_{0:T}} L(x_{0:T}, \lambda, w) = 2J_\phi(x_{0:T})^\top \operatorname{diag}(w)\Phi(x_{0:T}) + \lambda^\top J_{gh}(x_{0:T})$$

- For the dth demonstrations we define the loss

$$\ell^{(d)}(w, \lambda^{(d)}) = \left[\nabla_{x_{0:T}} L(x^{(d)}_{0:T}, \lambda^{(d)}, w)\right]^2$$

- Choose $\lambda^{(d)}$ to minimize $\ell^{(d)}(w, \lambda)$ *s.t.* KKT complementarity

$$\partial_\lambda \ell^{(d)}(w, \lambda^{(d)}) = 0 \quad \Rightarrow \quad \lambda^{(d)} = -(\tilde{J}_{gh}\tilde{J}_{gh}^\top)^{-1}\ \tilde{J}_{gh} J_\phi^\top \operatorname{diag}(\Phi) w$$

- Reduces to

$$\min_w\ w^\top \Lambda w \quad \text{s.t.} \quad w \geq 0\ , \quad \Lambda^{(d)} = \operatorname{diag}(\Phi) J_\phi \left[I - \tilde{J}_{gh}^\top (\tilde{J}_{gh}\tilde{J}_{gh}^\top)^{-1} \tilde{J}_{gh}\right] J_\phi^\top \operatorname{diag}(\Phi)$$

Reduction to a Quadratic Program

$$\min_w \; w^\top \Lambda w \quad \text{s.t.} \quad w \geq 0$$

- Two ways to enforce a *non-singular* solution
  - Enforce positive-definiteness of Hessian at the demonstrations $\rightarrow$ maximize $\log |\nabla^2_x f(x)|$ (c.p. Levine & Koltun)
  - Add the constraint $\sum_i w_i \geq 1 \;\; \rightarrow \;\;$ Quadratic Program
- Even if $\Phi(x_{0:T}), g(x_{0:T}), h(x_{0:T})$ are a arbitrarily non-linear, this ends up a QP!

- Related work:
  - Levine & Koltun: Continuous inverse Optimal Control with Locally Optimal Examples. ICML'12
  - Puydupin-Jamin, Johnson & Bretl: A convex approach to inverse optimal control and its application to modeling human locomotion. ICRA'12
  - Albrecht et al: Imitating human reaching motions using physically inspired optimization principles. HUMANOIDS'11
  - Jetchev & Toussaint: TRIC: Task space retrieval using inverse optimal control. Autonomous Robots, 2014.

Inverse KKT

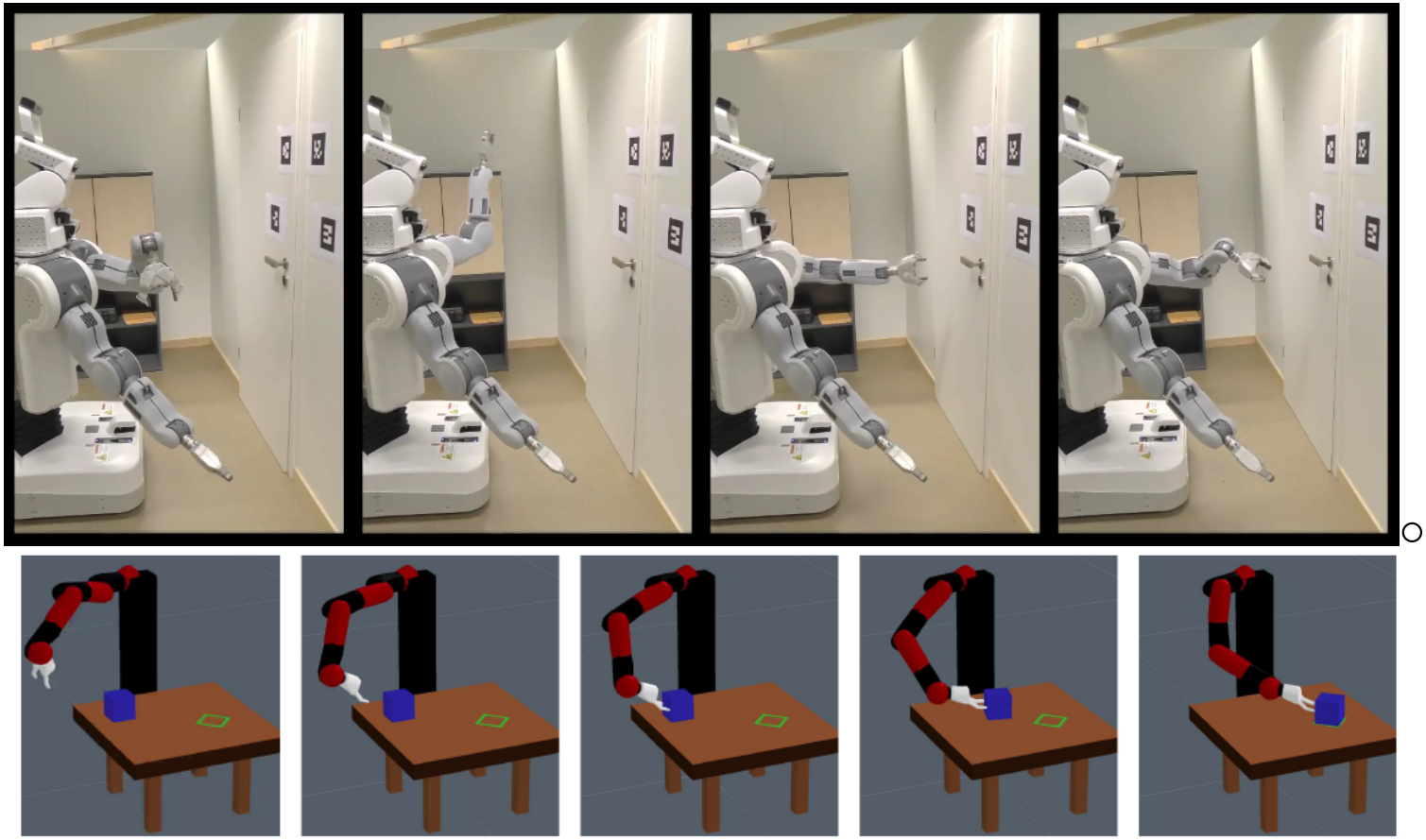


Fig. 3 These images show the box sliding motion of Section 5.2 where the goal of the task is to slide the blue box on the table to the green target region.

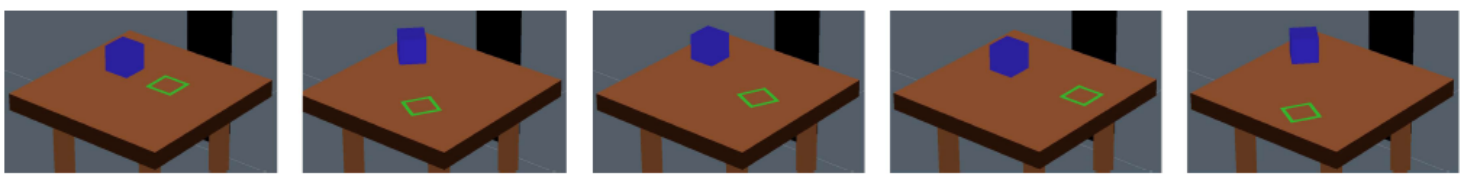
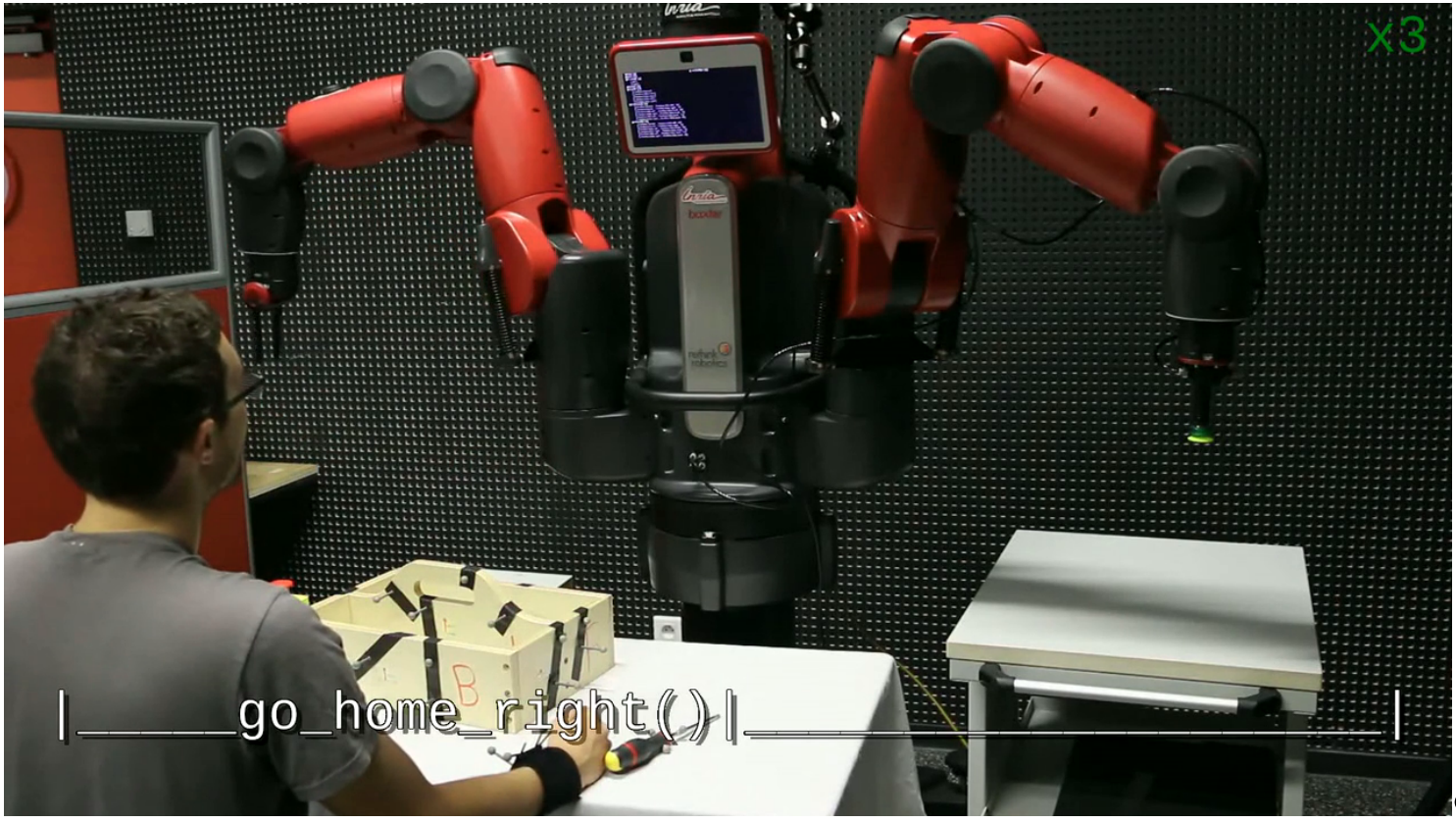


Fig. 4 Each image shows a different instance of the box sliding task. We were able to generalize to different initial box states (blue box) and to different final box targets (green area).

(3) Cooperative Manipulation Learning



(EU-Project *3rdHand*; PIs: Manuel Lopes, Jan Peters, Justus Piater, Marc Toussaint)

Research Questions

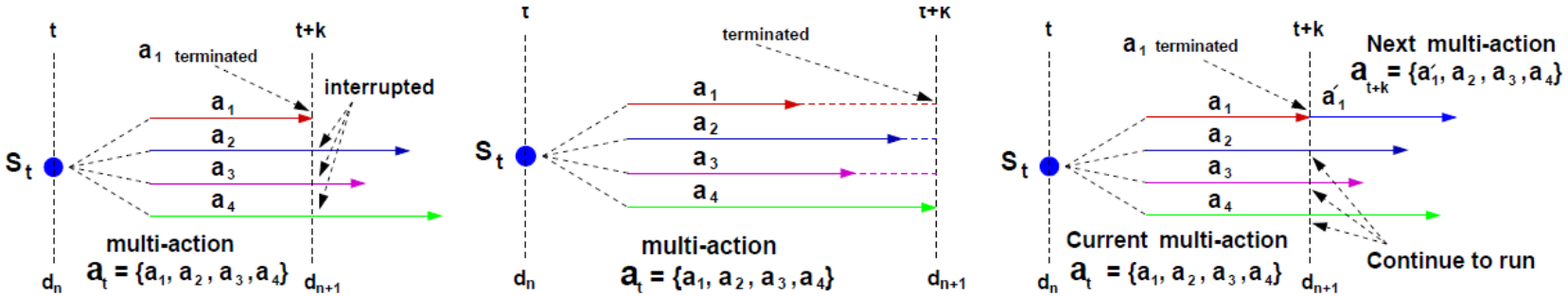
- What is a good formalization of such processes?
  - multi-agent
  - concurrent activities
  - durative actions
  - probabilistic outcomes

- What are methods for
  - anticipation (of human actions) & planning (of robot actions)?
  - learning from demonstration (imitation and inverse RL)?

- Bridging between symbolic and geometric problem formalization
- Enriching the interaction: active querying, hesitation, etc

Process formalization

- Existing formulations: semi-MDPs over multi-actions
  - Concurrent Action Models (Rohanimanesh & Mahadevan); Concurrent MDPs & Probabilistic Temporal Planning (Mausam & Weld)
  - A certain episode times, the planner makes a multi-action decision $(a_1, a_2, .., a_n)$ for all n agents; the decision space becomes combinatorial



Rohanimanesh, Mahadevan (NIPS'02)

- Issues:
  - Subjectively: Complicated, mutexing, awkward synchronization
  - Requires specialized planners (unsuccessful: IP)
  - No existing extensions to relational domains
  - No direct transfer of existing inverse RL methods

Relational Activity Processes (RAPs)

Toussaint, Munzer, Mollard & Lopes: *Relational Activity Processes for Modeling Concurrent Cooperation*. ICRA'16

- The current *state* lists the current activities (relational (1st-order logic)):

```
(object Handle), (free humanLeft), (humanLeft graspingScrew)=1.0,
(humanRight grasped Handle), (Handle held), (robot releasing Long1)=1.5, ..
```

- A planner reasons about *decisions* (for all agents!), which are
  - Initiate an activity, e.g. `(Initiate humanLeft graspingScrew)`
  - Terminate an activity, e.g. `(Terminate robot releasing Long1)`
  - Wait for a change in relational state
- Stochastic Relational Rules (cf. NDRs) determine the effects

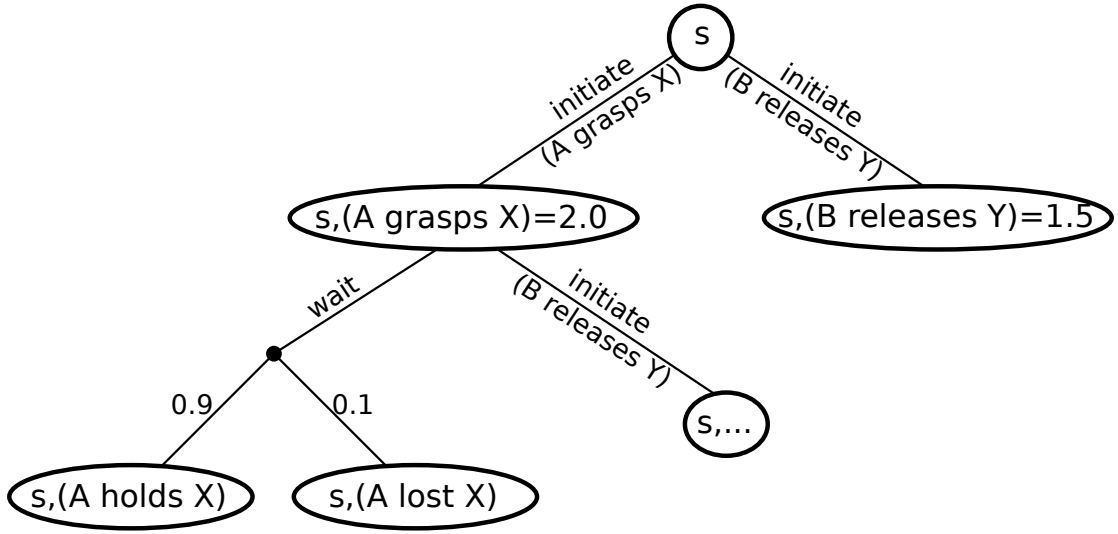
```
(Terminate X grasping Y){
  { (X grasping Y)! (X grasped Y) (X free)! (Y held) (X busy)! (Y busy)! }
  { (X grasping Y)! (X busy)! (Y busy)! }
  p=[0.9 0.1]
}
```

- This defines a decision process, which initiates, waits, and terminates activities of all agents, and predicts the effects.

Relational Activity Processes (RAPs)

- We “serialized” the decision process over concurrent activities
  - Reduction to a standard semi-MDP
  - **Standard methods for MCTS, direct policy learning & inverse RL become applicable in relational concurrent multi-agent domains**

Planning using Monte Carlo



- Every sample path gives a potential future, with rewards
- Given a set of samples, we can compute
  - a $Q(d, s)$-function over the next decision (including which agent it involves)
  - a reward-weighted probability over the future decision

  $\rightarrow$ anticipation of human action, planning of own actions

Imitation learning & inverse RL for cooperative manipulation

- Wonderful prior work:

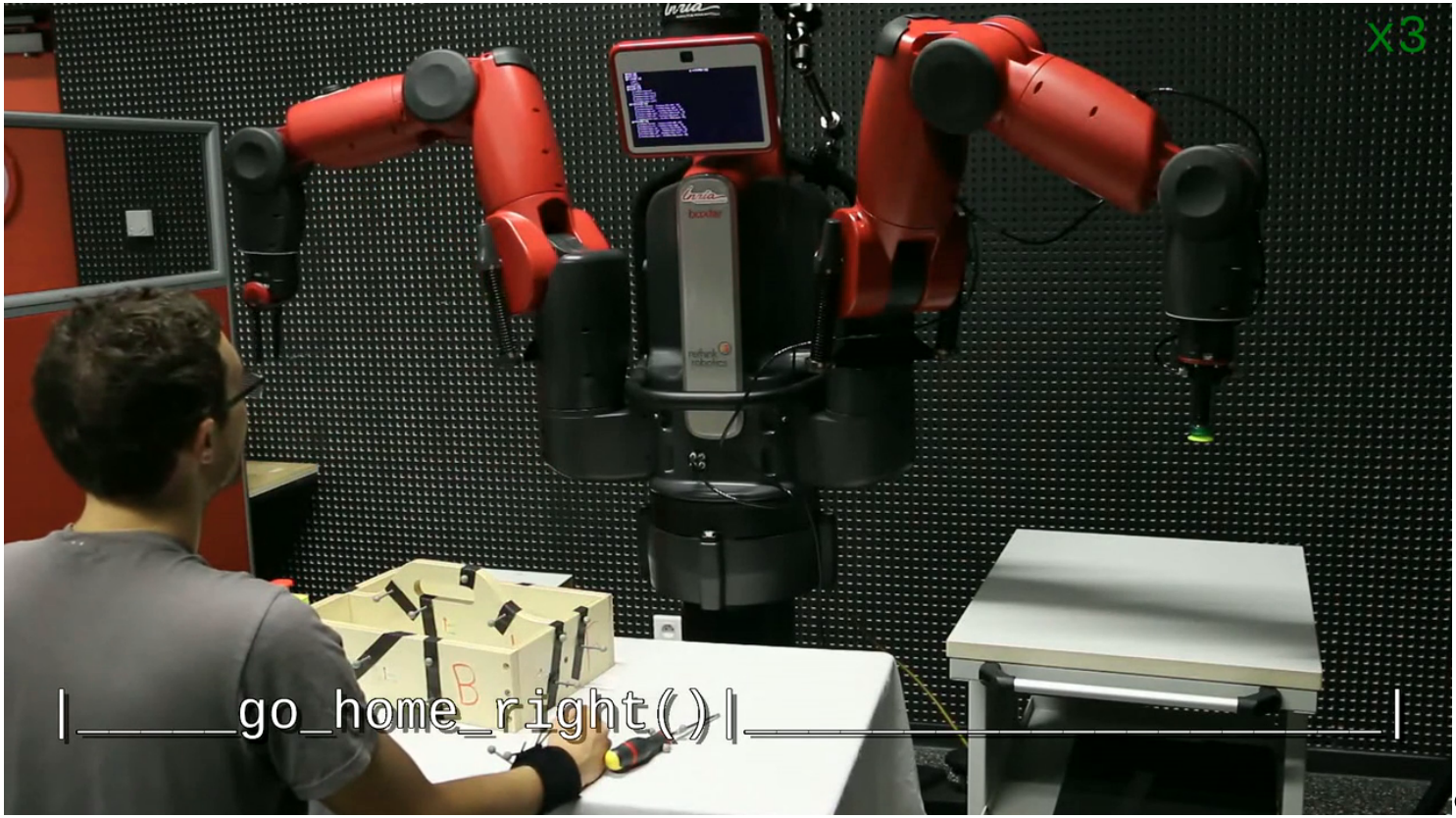
  Munzer et al.: *Inverse reinforcement learning in relational domains.* IJCAI'15

  - Imitation: Tree Boosted Relational Imitation Learning (TBRIL) to train a policy

$$\pi(a \mid s) = \frac{e^{f(a,s)}}{\sum_{a' \in \mathcal{D}(s)} e^{f(a',s)}} \ , \quad f(a,s) = \psi(a,s)^\top \beta$$

  - Use relational reward shaping and CSI to infer a relational reward function

- Directly translates to RAPs

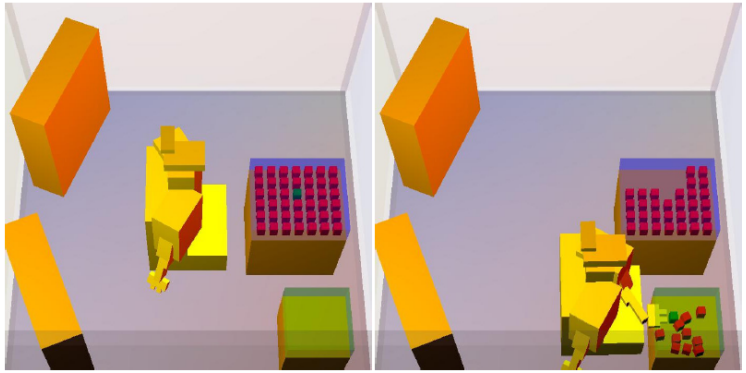


Toussaint, Munzer, Mollard & Lopes: *Relational Activity Processes for Modeling Concurrent Cooperation*. ICRA'16

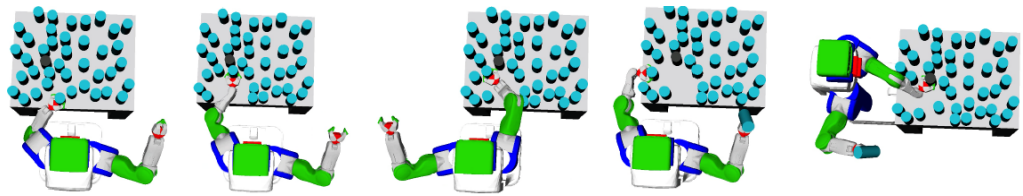
Oh no! We lost the geometry!

(4) Logic-Geometric Programming

Combined Task and Motion Planning



from Garrett et al: *FFRob...*, WAFR'14



from Srivastava et al: *Combined TAMP...*, ICRA'14

- Srivastava et al (ICRA 2014): Combined task and motion planning through an extensible planner-independent interface layer
- Siméon et al (IJRR, 2004): Manipulation planning with probabilistic roadmaps
- Lozano-Pérez et al (IROS 2014): A constraint-based method for solving sequential manipulation planning problems
- Garrett et al (WAFR 2014): An efficient heuristic for task and motion planning

- All previous work on TAMP and Rearrangement is **sample- and satisfiability-based:**
  - sample-based path finding
  - sample-based grasp finding (or pre-discretized)
  - search heuristics: which objects to try move next/before
  - backtracking

- I don't think this reflects the hybrid structure well

Sequential Manipulation

- m rigid objects, n-articulated joints, state space $\mathcal{X} \in \mathbb{R}^n \times SE(3)^m$

Sequential Manipulation

- m rigid objects, n-articulated joints, state space $\mathcal{X} \in \mathbb{R}^n \times SE(3)^m$
- Path $x : [0, T] \to \mathcal{X}$, kinematics (description of the allowed motions)

$$h_{path}(x(t), \dot{x}(t)) = 0 \ , \quad g_{path}(x(t), \dot{x}(t)) \leq 0$$

Sequential Manipulation

- m rigid objects, n-articulated joints, state space $\mathcal{X} \in \mathbb{R}^n \times SE(3)^m$
- Path $x : [0, T] \to \mathcal{X}$, kinematics (description of the allowed motions)

$$h_{path}(x(t), \dot{x}(t)) = 0 \ , \quad g_{path}(x(t), \dot{x}(t)) \leq 0$$

- First-order logic language $\mathcal{L}$ to describe kinematic structure $\to$ symbolic kinematic states $s(t) = s(x(t)) \in \mathcal{L}$

$$h_{path}(x(t), \dot{x}(t) \,|\, s(t)) = 0 \ , \quad g_{path}(x(t), \dot{x}(t) \,|\, s(t)) \leq 0 \ , \quad \textit{smooth}$$

Sequential Manipulation

- m rigid objects, n-articulated joints, state space $\mathcal{X} \in \mathbb{R}^n \times SE(3)^m$
- Path $x : [0, T] \to \mathcal{X}$, kinematics (description of the allowed motions)

$$h_{path}(x(t), \dot{x}(t)) = 0 \ , \quad g_{path}(x(t), \dot{x}(t)) \leq 0$$

- First-order logic language $\mathcal{L}$ to describe kinematic structure $\to$ symbolic kinematic states $s(t) = s(x(t)) \in \mathcal{L}$

$$h_{path}(x(t), \dot{x}(t) \,|\, s(t)) = 0 \ , \quad g_{path}(x(t), \dot{x}(t) \,|\, s(t)) \leq 0 \ , \quad \textit{smooth}$$

- Assume that $s \in \mathcal{L}$ is sufficient to describe possible successors; $s_k \in \text{succ}(s_{k\text{-}1})$ switches kinematic symbols at time t_k:

$$h_{switch}(x(t_k) \,|\, s_k, s_{k\text{-}1}) = 0 \ , \quad g_{switch}(x(t_k) \,|\, s_k, s_{k\text{-}1}) \leq 0$$

- The role of symbols is to make the remaining problem smooth

- Piece-wise smooth paths for given $s_k \in \mathcal{L}$
- Categorial decisions about $s_k \in \text{succ}(s_{k\text{-}1})$

- The logic/categorial state implies constraints on the path

A Logic-Geometric Programming Formulation

- m rigid objects, n-articulated joints, path $x: [0,T] \to \mathbb{R}^n \times SE(3)^m$

$$\min_{x, s_{1:K}, t_{1:K}} \int_0^T c(x(t), \dot{x}(t), \ddot{x}(t))\, dt + f_{goal}(x(T))$$

$$\text{s.t.} \quad h_{goal}(x(T)) = 0,\ g_{goal}(x(T)) \le 0$$

$$\forall_{t\in[0,T]}\ h_{path}(x(t), \dot{x}(t) \,|\, s_{k(t)}) = 0$$

$$\forall_{t\in[0,T]}\ g_{path}(x(t), \dot{x}(t) \,|\, s_{k(t)}) \le 0$$

$$\forall_{k=1}^{K}\ h_{switch}(x(t_k) \,|\, s_k, s_{k\text{-}1}) = 0$$

$$\forall_{k=1}^{K}\ g_{switch}(x(t_k) \,|\, s_k, s_{k\text{-}1}) \le 0$$

$$\forall_{k=1:K}\ s_k \in \mathsf{succ}(s_{k\text{-}1})$$

$$s_K \models \mathfrak{g}$$

- An LGP uses a logic state representation s_k to define the constraints on the geometric variable x

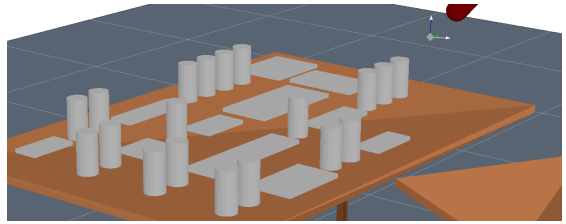
Toussaint: *Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning*. IJCAI'15

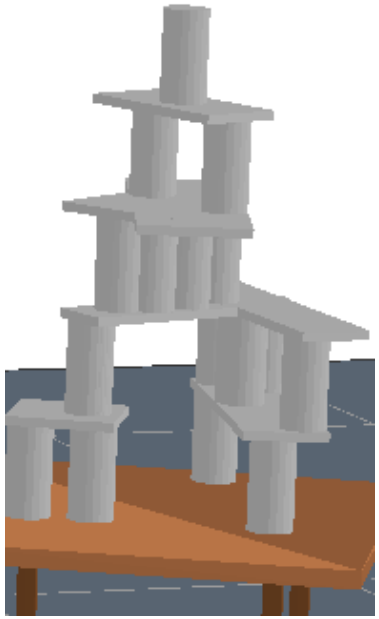
“Relational Mathematical Programming”

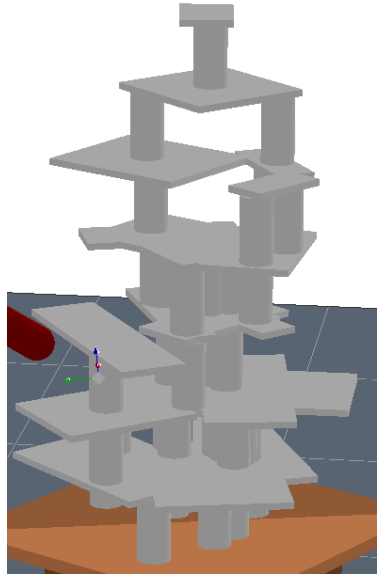
- Relational/Logic Mathematical Programs form a new, more general class of optimization problems
  - I’m influenced by discussions with Kristian Kersting & Luc de Readt
  - Very general declarative language
  - Solvers for relational LPs (e.g. Kristian Kersting)

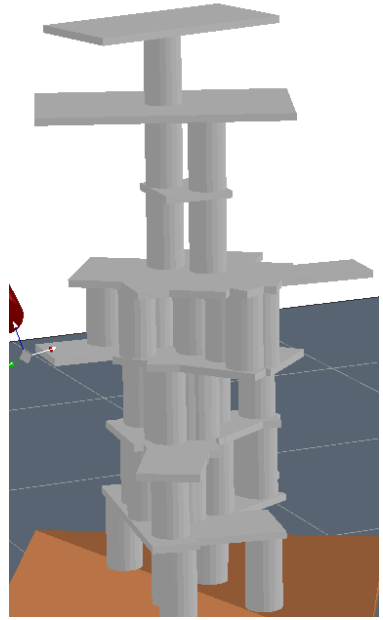
Example Domain

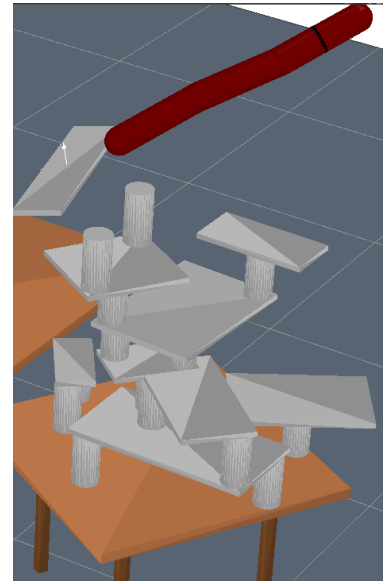
- Building high, physically stable construtions

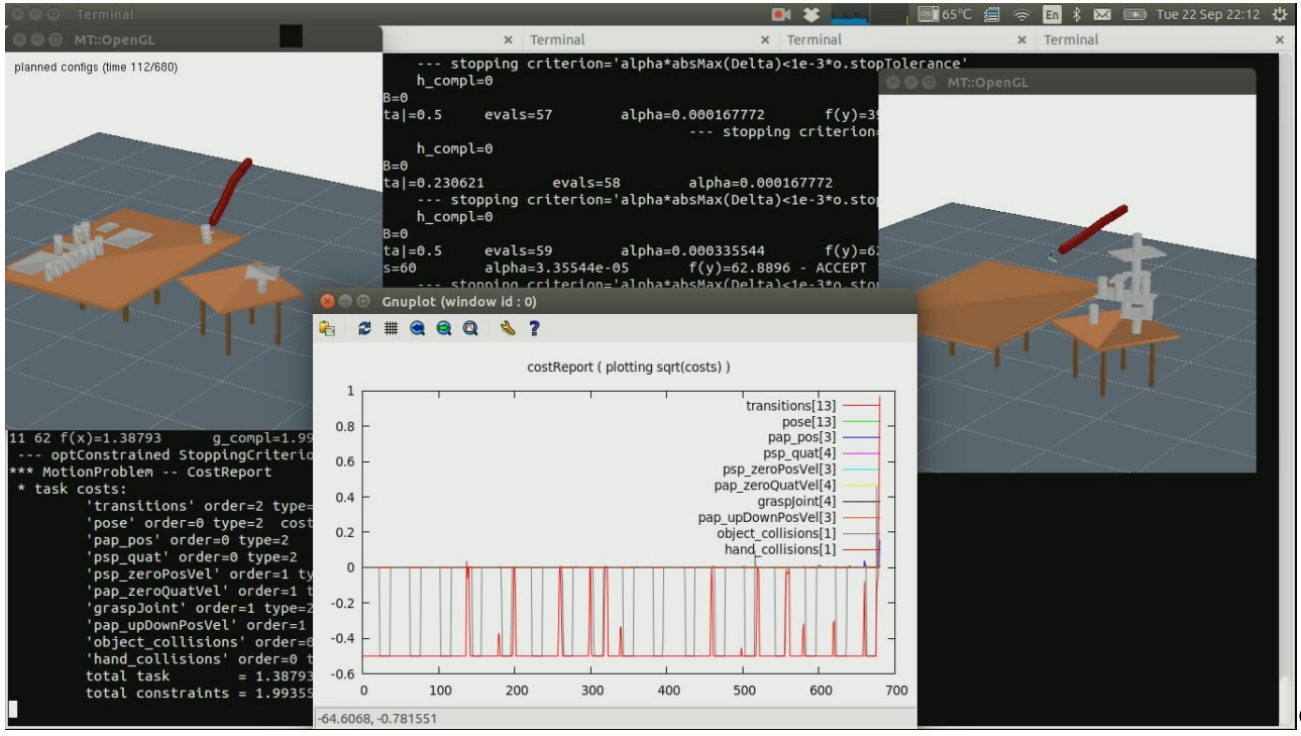











Terminal
MT::OpenGL
planned configs (time 112/680)
Terminal
--- stopping criterion='alpha*absMax(Delta)<1e-3*o.stopTolerance'
h_compl=0
B=0
ta|=0.5 evals=57 alpha=0.000167772
--- stopping criterion
h_compl=0
B=0
ta|=0.230621 evals=58 alpha=0.000167772
--- stopping criterion='alpha*absMax(Delta)<1e-3*o.sto
h_compl=0
B=0
ta|=0.5 evals=59 alpha=0.000335544
s=60 alpha=3.35544e-05 f(y)=62.8896 - ACCEPT
Gnuplot (window id : 0)
costReport (plotting sqrt(costs))
transitions[13]
pose[13]
pap_pos[3]
psp_quat[4]
psp_zeroPosVel[3]
pap_zeroQuatVel[4]
graspJoint[4]
pap_upDownPosVel[3]
object_collisions[1]
hand_collisions[1]
-64.6068, -0.781551
11 62 f(x)=1.38793 g_compl=1.99
--- optConstrained StoppingCriteri
*** MotionProblem -- CostReport
* task costs:
'transitions' order=2 type=
'pose' order=0 type=2 cost
'pap_pos' order=0 type=2
'psp_quat' order=0 type=2
'psp_zeroPosVel' order=1 ty
'pap_zeroQuatVel' order=1 t
'graspJoint' order=1 type=2
'pap_upDownPosVel' order=1
'object_collisions' order=0
'hand_collisions' order=0 t
total task = 1.38793
total constraints = 1.99359

Solver

- Use plain MC(!) to generate proposal sequences $s_{1:K}$
  - Huge possibilities for improvement, but not the bottleneck here

Fast approximate heuristics (lower bounds) to focus geometric optimization

- Three levels of approximation on the geometric side
  - **Effective Kinematics** of leaf configurations
    $\rightarrow$ Newton method over leaf configurations
    $\rightarrow$ optimistic heuristic to inform search
  - Newton-method over keyframes
  - Newton-method over the full path

- Use **KOMO** to optimize over leaf configurations, key frames, or full paths

Effective Kinematics as Optimistic Heuristic

- The set of all possible configurations $x(T) \in \mathcal{X}$ **conditional to a sequence** $s_{1:K}$

$$\mathcal{X}^*(s_{1:K}) = \{x(T) \in \mathcal{X} \,|\, s_{1:K}\}$$

- This describes a smooth manifold of all (optimistially!) “reachable” configurations after $s_{1:K}$, and a smooth NLP over such configurations, including $\psi(x(T))$
- This “defers” parameteric decisions of interactions (cf. Lozano-Perez)

$$\leftrightarrow \text{“Geometric reasoning”}$$

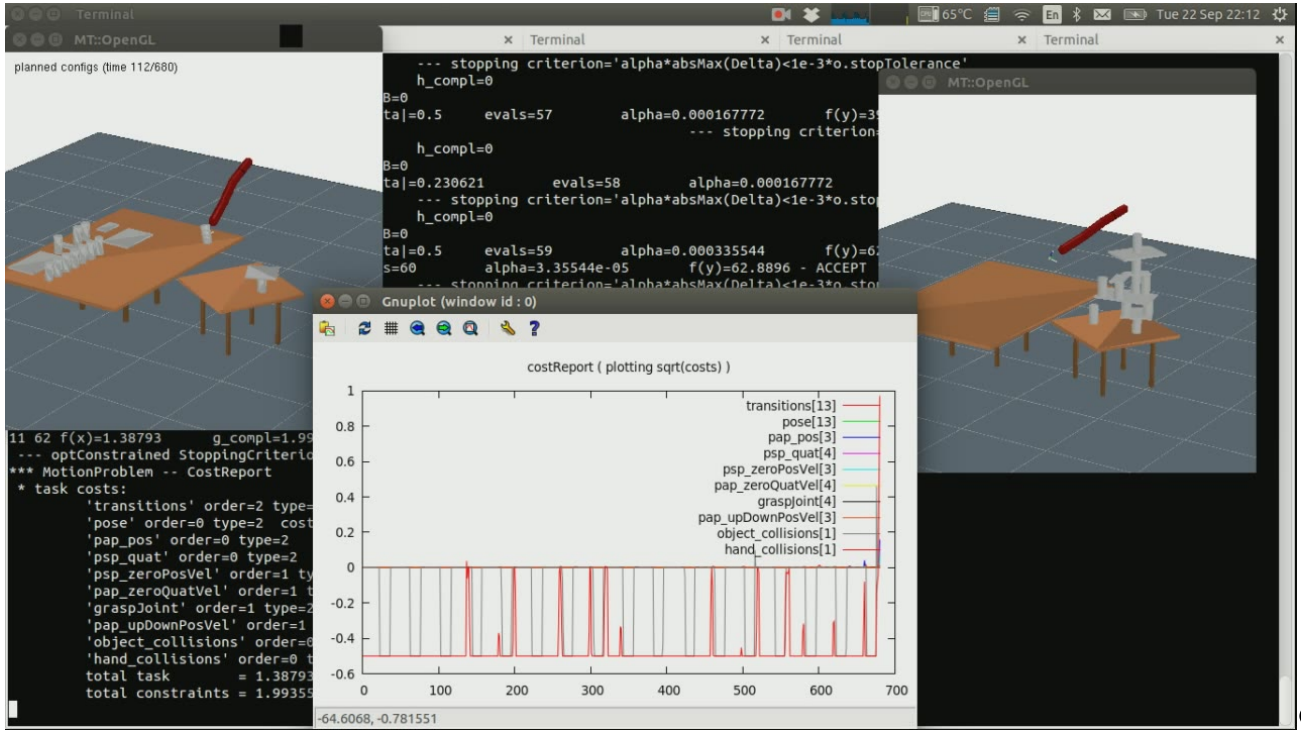
Logic to Describe *Effective* Kinematics

- We need sufficient symbols to capture the structure of $\mathcal{X}^*(s_{1:K})$. In the tower case, `supports` is sufficient:

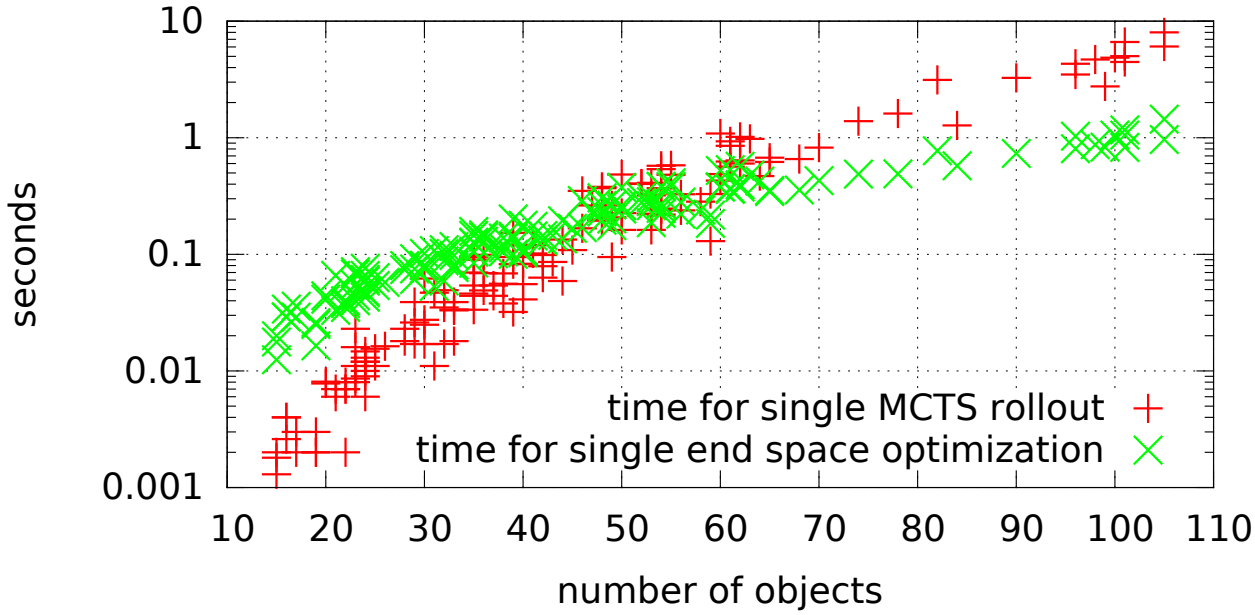
```
CylinderOnBoard(X, Y):
     Cylin(X) free(X) Board(Y)
     supports(Y X)

BoardOnCylinder(X, Y):
     Board(X) Cylin(Y) free(Y)
     free(Y)! supports(Y X)

BoardOn2Cylinders(X, Y, Z):
     Board(X) free(Y) Cylin(Y) free(Z) Cylin(Z) depth(Y)=depth(Z)
     free(Y)! free(Z)! supports(Y X) supports(Z X)
```

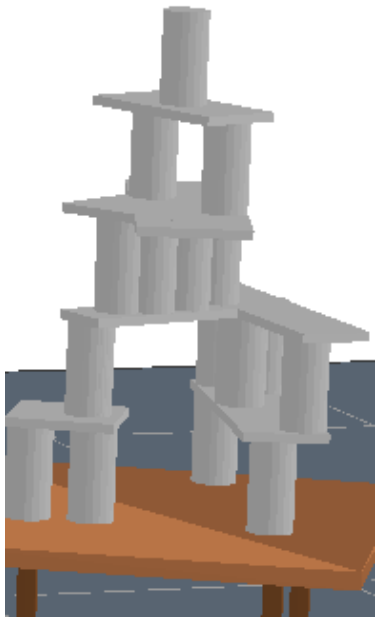
- `supports` implies constraints and costs in the *effective kinematics*:
  - Create `transXYPhi` joints
  - Inequality constraint of support being “inside”
  - Multiple support: Maximize distance to center
  - Single support: center align cost

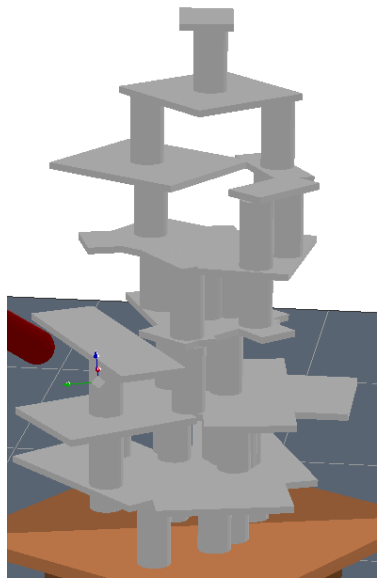
- The logic state s_k defines the path constraints & the effective kinematics
- The transitions $s_k \in \text{succ}(s_{k\text{-}1})$ are described by logic rules

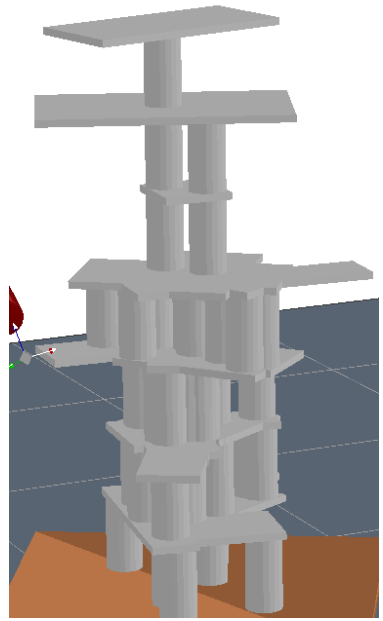

Terminal
MT::OpenGL
planned configs (time 112/680)
Terminal
Terminal
Terminal
--- stopping criterion='alpha*absMax(Delta)<1e-3*o.stopTolerance'
h_compl=0
B=0
ta|=0.5 evals=57 alpha=0.000167772
--- stopping criterion
h_compl=0
B=0
ta|=0.230621 evals=58 alpha=0.000167772
--- stopping criterion='alpha*absMax(Delta)<1e-3*o.sto
h_compl=0
B=0
ta|=0.5 evals=59 alpha=0.000335544
s=60 alpha=3.35544e-05 f(y)=62.8896 - ACCEPT
MT::OpenGL
Gnuplot (window id : 0)
costReport (plotting sqrt(costs))
transitions[13]
pose[13]
pap_pos[3]
psp_quat[4]
psp_zeroPosVel[3]
pap_zeroQuatVel[4]
graspJoint[4]
pap_upDownPosVel[3]
object_collisions[1]
hand_collisions[1]
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
100
200
300
400
500
600
700
-64.6068, -0.781551
11 62 f(x)=1.38793 g_compl=1.99
--- optConstrained StoppingCriteri
*** MotionProblem -- CostReport
* task costs:
'transitions' order=2 type=
'pose' order=0 type=2 cost
'pap_pos' order=0 type=2
'psp_quat' order=0 type=2
'psp_zeroPosVel' order=1 ty
'pap_zeroQuatVel' order=1 t
'graspJoint' order=1 type=2
'pap_upDownPosVel' order=1
'object_collisions' order=0
'hand_collisions' order=0 t
total task = 1.38793
total constraints = 1.99359

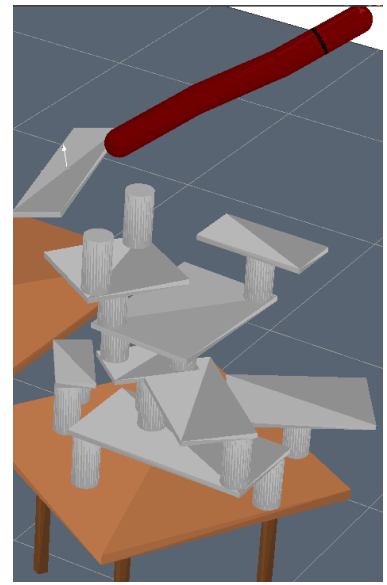
Scaling of MC & Effective Kinematics Optimization



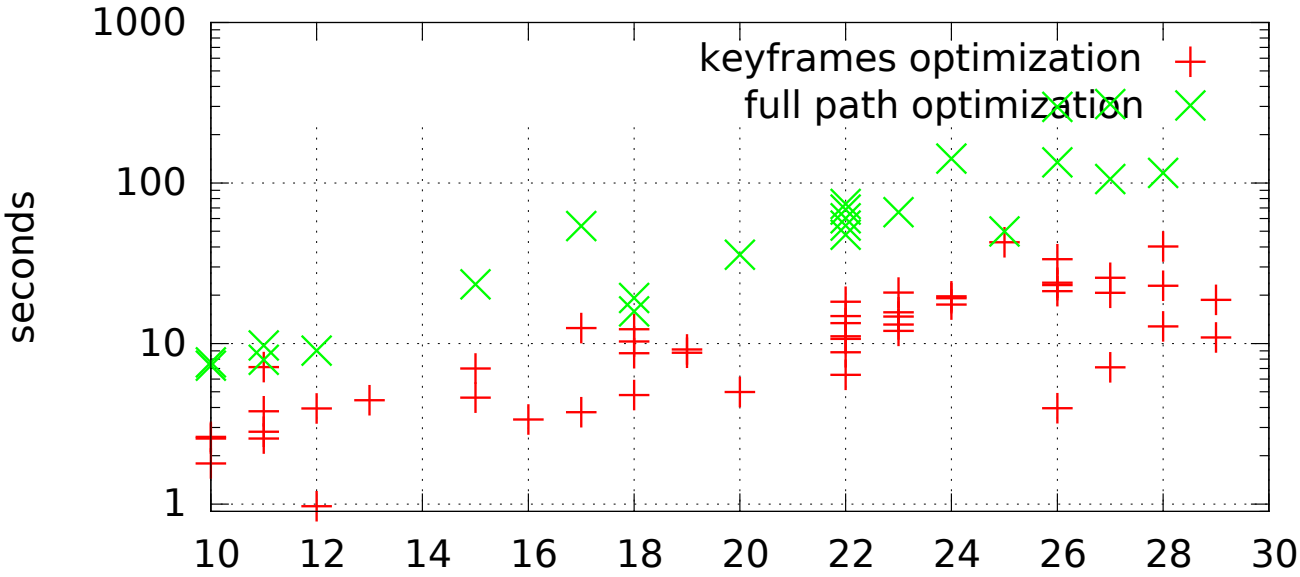




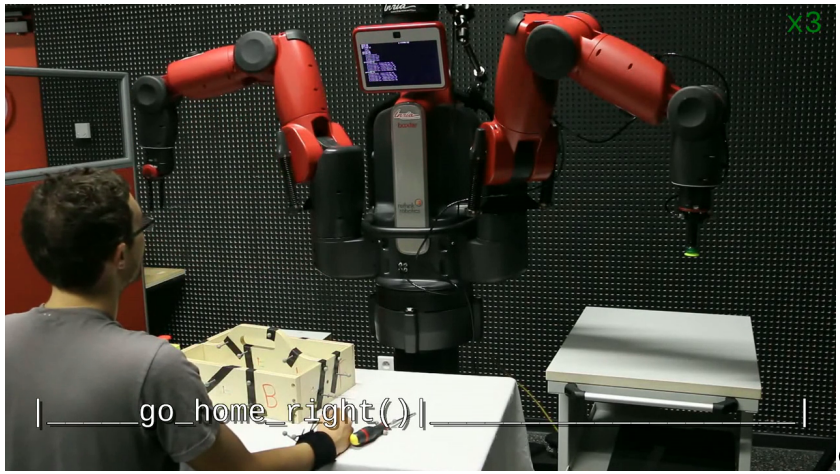




Scaling of Keyframe Sequence & Path Optimization



Ongoing Work: Cooperative Manipulation



- Ongoing:
  - Including the geometry of cooperation in the reasoning
  - **"Multi-agent Logic-Geometric Programming"**
- Technical challenges:
  - kinematic switches add and delete DOFS; $\dim(x_t)$ varies
  - handling "delayed effects" correctly
  - ...all this without breaking the banded-ness of the Hessian

Conclusions

- The challenge is to find good representations of the problems
  - Learning from single demonstration $\rightarrow$ strong priors about own motion; black-box RL w.r.t. interation
  - Cooperative manipulation $\rightarrow$ RAP to represent concurrent activities $\rightarrow$ planning, inverse RL
  - Logic-Geometric Programming: Have a unified problem formulation of the symbolic-subsymbolic levels

- Don't leave "system integration" to the software engineer!
  Instead, formalize integrated problems.

- Q: Would LGP also work for walking/climing?
  - Same categorial decisions about kinematic switches
  - Same "delayed effects" as for SeqManip

Thanks

- *for your attention!*
- to collaborators/co-authors
  - *Relational RL topics:* Tobias Lang, Manuel Lopes, Kristian Kersting
  - *Physical Exploration:* Oliver Brock
  - *Human-Robot Collaborative work:* Manuel Lopes, Thibaut Munzer, Jan Peters, Justus Piater
  - *Newton Methods for Path Optimization:* Nathan Ratliff, Jeannette Bohg, Stefan Schaal
- to colleagues
  - *Logic-Geometric Programming topic:* Tomas Lozano-Pérez, Leslie Pack Kaelbling, Kristian Kersting, Luc De Raedt, Karl Tuyls, George Konidaris
- and my lab: