**Post-Processing on Manipulation Trajectories**, i.e. on trajectories output by classical, sampling-based manipulation planner (e.g. Philipp’s)

**KLICK**

**Why Post-Processing** reasonable/necessary?

* Manipulation planners output kinematic path consisting of sequence of robot configurations
* Lead subsequently from start to goal
* Typically, no dynamics   
  -> post-processing steps to transform geometric path to trajectory respecting dynamic constraints  
  -> as we will soon see: most basic approach to do so is not very fast, all pp steps aim for doing that better
* Output in general not optimal (not even locally) -> post-processing aimed for local improvements

**KLICK**

**What PP-steps?**

* Implemented and evaluated a couple of post-processing steps

**KLICK**

**Hauser’s Shortcutting Approach**

* Given geometric path, from configuration to configuration -> transform to dynamic trajectory
* How?
  + At each configuration point: change in direction -> stop there, velocity = 0
  + In between: move as fast as limits on velocity and acceleration allow
  + **b)**
* Hauser’s Idea: faster since often collision-free shortcuts possible s.t.

1. Shorter distance
2. No stopping

* How?
  + sample 2 random configuration (i.e. joint position + velocity for each axis)
  + find shortcut motion connecting configurations
  + check for collisions -> discard or accept

**KLICK**

* most difficult: how compute shortcuts?
* given: start/end configuration = 14 positions + velocities

**KLICK**

* Hauser’s basic idea:
  + consider problem for each axis separately   
    -> compute for each fastest motion  
    -> 14 easy to solve problems
  + find “bottleneck” axis, i.e. most time-consuming
  + “synchronize” all axes to bottleneck time,

**KLICK**  
i.e. for all other axes: find slowlier motion profile s.t. end point is reached only after end-time of bottleneck axis

**KLICK**

* Problem:
  + synchronization not always possible
  + i.e. cannot decelerate motion to arbitrary point in time
  + in fact, inoperative time intervals

**KLICK**

* instead:
  + Reflexxes (company run by Thorsten Kroeger)
  + method to compute inoperative time intervals via decision trees
  + synchronize to earliest possible point in time
* too time-consuming to implement that ourselves  
  -> integrated open-source code from reflexxes to implement Shortcut-Approach

**Smooth Interaction Approach**

* interaction = movement when robot is approaching object
* for several reasons: want robot approach object from above, following a linear movement
* current solution:
  + follow the trajectory up to point above object
  + stop there
  + perform linear movement, grasp, linear movement back up
  + stop again
  + perform rest
* stopping takes a lot of time

**KLICK**

* instead: slide smoothly into linear movement
* sample random point along trajectory before/after linear movement
* compute orange motion connecting these to point somewhere in middle
* Reflexxes

**KLICK**

**Sampling of New Transitions**

* What is transition?
  + recall how manipulation planners typically work:
    - sample set of grasps and placements (PGP)
    - for each build configuration roadmap, consisting of arbitrary sampled robot configurations
    - manifolds connected by transition areas (here green)  
      -> transitions are configurations in both roadmaps

**KLICK**

* when planner finds path through this cluster of roadmaps -> chooses certain transitions
* idea: instead of stick to these transitions:
  + sample new transition in each transition area
  + re-plan trajectories preceeding and succeeding this transition

**KLICK**

* How do such new transitions look like?
  + For active arms currently involved in grasping/releasing object  
    -> sample inverse kinematic  
    -> s.t. grasp stays same
  + For passive arms NOT involved  
    -> sample arbitrary collision-free configuration

**KLICK**

* Re-Planning:
  + red transition chosen instead
  + need to replan motion before and afterwards
  + sample point on previous trajectory
  + connect using reflexxes to obtain new, red trajectory
  + same for succeeding trajectory
* same for transition corresponding to smooth interaction  
  -> 4 candidates for new trajectory
* update if faster
* entire procedure: combination of 3 steps
  + iterate, in each loop:
  + Hauser Shortcut in each sub-trajectory
  + sample new transition for each two adjacent motions,  
    here try to include smooth interaction

**KLICK**

**Sampling of New Grasps**

* manipulation planner samples fixed grasps
* so far: stick to these

1. maybe approximating good solution not possible, e.g. when longish object

**KLICK**

* idea: also sample new grasps and re-plan (similar to idea of transition sampling)
* however: difficult to integrate in current algorithm:
  + new grasp changes entire set-up
  + changes collision checks for all later actions
  + expensive updates necessary

**KLICK**

* 2 video clips:
  + example task, execution before and after PP
  + trajectory post-processed with iteration containing first 3 steps
* stop before/after linear movement
* stop in middle of trajectory

**KLICK**

* also show differences graphically: 4 plots of velocity diagrams

**KLICK**

* motion profile for all 14 axes of 2 robot arms
* linear movements
* several points where both robots stop

**KLICK**

* before and after linear movement -> smooth interaction

**KLICK**

* when grasping/releasing object -> no way around

**KLICK**

* stops within trajectory -> Hauser’s shortcutting

**After Hauser’s Shortcutting:**

* stopping points within movement are gone
* only one motion per sub-task
* shorter execution time
* still stopping before/after linear movement

**After *additionally* Smooth Interaction:**

* even faster
* no smooth interaction in last part: not possible without collisions since box in the way

**After new Transition (combined with Hauser and Smooth Interaction):**

* fastest motion
* since new transitions: also new velocity profiles

**KLICK**

**Statistical Comparison**

* in order to generate this comparison plot: executed
  + for one test scenario
  + each PP step with increasing number of iterations
  + 20 times each
  + plotted average relative improvement compared to solution without any PP
* green: Hauser
  + number of iterations = how often sampled 2 points and tried to shortcut
  + improvement of about 17-18%
  + could be even more if geometry of example seen not rather easy  
    -> initial trajectory already pretty well
* red: Hauser + Smooth Interaction
  + improvements up to almost 25%
* blue: Transition Sampling
  + in each iteration:
    - Hauser Shortcut for each sub-trajectory (with iteration number 5)
    - new transition at each transition area
    - Smooth Interaction
  + most effective
  + but also most computation time
* Internship: hand-crafted shortcutting/speed-up approaches
  + - * improvements, but no convergence to local optima
* Now MA: more systematic approach

**KLICK**

* Idea 1:
  + complete new approach: model dynamic manipulation task as MINLP
  + MINLP = mixed integer nonlinear program  
    -> nonlinearity: robot dynamics  
    -> integer: binary decision variables
  + Include dynamic constraints into optimization, no longer:   
    first geometric, then “post-dynamize”

**KLICK**

* systematic approach for local, time-optimal solutions
* potentially: global solutions if find globalization methods
* Idea 2:
  + develop algorithm working as systematic source of information for informed search, usable in sampling-based manipulation planners
  + exist similar ideas for motion planning
  + but: for manipulation planning additional transitions   
    -> more challenging to find good information algorithm for informed search

**KLICK**

* hope: gain insight into basic problem structure having MINLP solution at hand
* find more effective sampling strategies