Post Processing

1 Four Steps of Post Processing

of the plans that are generated e.g. by RMR* [Philipp's paper] or PTR (probabilistic tree of roadmaps):

1. smoothing, ShortCutting (`OptimizeTrajectory`)
2. smooth in linear movement, object grasping (`OptimizeTrajectory`)
   - now: try to use `alternate_start` by generating time optimal spline. If this fails, don't use it at all.
   - idea: if time optimal spline fails, try another point above `alternate_start`
3. use different contacts within transition regions between C_{free,σ_1} and C_{free,σ_2}
4. copy planned path in a within-contact roadmap in C_{free,σ_1} into another area C_{free,σ_2} of the robot's configuration space, corresponding to another contact σ_2
   $\rightarrow$ in practice, this corresponds for instance to a slightly different grasp of the object and therefore a only slightly shifted motion

2 Pseudocode

First, consider only post-processing steps 1-3.

---

for all `adjacent contacts` σ_1, σ_2 **do**
  for $i = 1 : 100$ **do**
    `sampleTransition(`σ_1, σ_2`);`
    `findShortestPath(`σ_1`), findShortestPath(`σ_2`);` \\σ_1`.start and` σ_2`.end is fixed`
    `optimizeTrajectory(`σ_1`), optimizeTrajectory(`σ_2`);` \\`shortcutting and smooth docking`
    if `faster(`σ_1, σ_2`)` **then**
      `update path(`σ_1`), path(`σ_2`), transition(`σ_1, σ_2`);`
    end if
  end for
end for

If any of the procedures fails, `continue` to next iteration.

```
for i = 1 : iterations_ do
   for contact σ_1 do
      σ_2 = successor(σ_1)
      sampleTransition(σ_1, σ_2);
      findShortestPath(σ_1), findShortestPath(σ_2);        \\σ_1.start and σ_2.end is fixed
      optimizeTrajectory(σ_1), optimizeTrajectory(σ_2);        \\shortcutting and smooth
docking
      if faster(σ_1, σ_2) then
         update path(σ_1), path(σ_2), transition(σ_1, σ_2);
      end if
   end for
end for
```

```
initializeSpline()
dock alternate_global_start and alternate_global_end
for i = 1 : iterations_ do
   for all contact σ do
      optimizeSpline in σ
   end for
   for all contact σ do
      σ' = neighbor(σ);
      sampleTransition(σ, σ');
      compute alternate_transition;
      compute four candidate vectors of splines;
      if faster(σ_1, σ_2) then
         update optimized_splines(σ), optimized_splines(σ');
      end if
   end for
end for
```

2.1 Problems

- fast trajectories for first two contacts might entail long trajectory in later contacts

2.2 Open Tasks

- How do we `sampleTransitions` systematically?
  - Use SAFT-Strategy for mode expansion (i.e. assign priority to each pair (σ, σ') and decrease it if sampling in $\mathcal{F}_{\sigma,\sigma'}$ fails)
  - Use utility-based strategy like Hauser (i.e. come up with some utility function/heuristic for contacts)
- How do we include post processing step 4?