Overview over Papers

Contents

1 RRT- and PRM-based Algorithms for Motion and Manipulation Planning

1.1 Sampling-based Algorithms for Optimal Motion Planning [Karaman]

- describes already existing algorithms, in particular PRM and RRT
- evaluates quality of solutions yield by PRM and RRT w.r.t. probabilistically completeness, computational complexity, optimality
- introduces new algorithms PRM* and RRT* which are still probabilistically complete and computationally efficient, but moreover asymptotically optimal
- **Optimal Probabilistic RoadMaps (PRM*):**
  - connection radius is chosen as a (decreasing) function of n, the number of samples, scaled as $\log n/n$.
  - extends "simple" PRM, i.e. disregarding the same-component-check
- **Rapidly-exploring Random Graph (RRG):**
  - incremental algorithm to build a connected roadmap, possibly containing cycles
  - similar to RRT: first attept to connect new sample to nearest node. If successful, additionally attempt connections from all vertices that are within a certain neighborhood
  - RRT and RRG have same vertex set, RRT is subgraph of RRG
- **Optimal Rapidly-exploring Random Trees (RRT*):**
  - obtained from RRG by avoiding cycles by removing edges that are not part of a shortest path from the root to a vertex
  - corresponds to "rewiring" of RRT tree, ensuring that vertices are reached through a minimum-cost path
  - In other words: new states are not only added to a tree, but also considered as replacement parents for existing nearby states in the tree
- Computational complexity: M_n^{ALG} denotes total number of calls of the collision checker by algorithm ALG in iteration n
  - $M_n^{PRM} \in \Omega(n)$
  - $M_n^{PRM^*}, M_n^{RRG}, M_n^{RRT^*} \in \mathcal{O}(\log n)$

1.2 Optimal, sampling-based Manipulation Planning [Philipp]

- Considers *manipulation* planning, not only motion planning, proposes a sampling based manipulation planner (RMR*) which is asymptotically optimal (application of 1.4 or 1.5 Multi-Modal PRM)
- uses PRM* but on joint configuration space $\mathcal{C}_r \times \mathcal{C}_o$
- **Random Manipulation Roadmap-star (RMR*):** consists of preprocessing and query phase:
  1. **roadmap construction:**
     - sample N_c contact states σ_S
     - for each σ_S build a within-contact roadmap within $\mathcal{C}_{\text{free},\sigma_S}$ using PRM* (which samples N_i random robot configurations $c \in \mathcal{C}_{\text{free},\sigma_S} \subset \mathcal{C}_r$)
     - for each pair of contact states try to connect the corresponding roadmaps by sampling N_t transitions
  2. **query phase:**
     - attempt to connect start configuration $(c_{\text{start}}, \sigma_{\text{start}})$ to previously constructed manipulation roadmap:
       * construct (docking) roadmap for σ_{start} using PRM*
       * connect this to every manipulation roadmap for each contact state σ_S
     - use standard graph search algorithm to find minimal-cost path into $\mathcal{C}_{\text{goal}}$
- addresses three challenges:
  - Continuity: instead of discretizing the contact space, sample contacts, add them to list Σ_σ, build within-contact roadmap for each using PRM* (similar to Hauser's idea of having a tree of roadmaps)
  - Incomplete motion planning: solve manipulation problem simultaneously on both levels (i.e. contact level and motion planning level). In total, we again obtain probabilistically completeness
  - Optimality: Under certain robustness assumptions it is proven that RMR* is asymptotically optimal

1.3 Task Planning with Continous Actions and Nondeterministic Motion Planning Queries [Hauser]

- two major improvements:
  - handle *continuously* parameterized actions by sampling actions u.a.r.
  - plan quickly, but *adaptively increase* time limit
- sample-based tree planner to explore tasks, probabilistic roadmap planner to explore feasible space for each task
  → i.e. tree of roadmaps
- search-based approach with continuous input requires discretization, choice is critical.
  Contribution of paper: conditions necessary for planner to find path with small number of sampled actions.
- use *forward search*, i.e. grow tree of feasbile states rooted at start state by repeatedly applying actions to states until endgame is reached. Difficulties if action is continuously parametrized:
  - planner must sample finite number of parameter instantiations
  - set of valid parameters may form lower dimensional subspace of parameter space (i.e. probability of choosing valid = 0)
  - may be no feasible continuous path achieving the action
- **Probabilistic Tree-of-Roadmaps algorithm (PTR):**
  - **components:**
    - vertices = states, edge $x \rightarrow x'$ by selecting action $A(u)$ applicable to x with resulting state x'
    - status of edge is complete or active, depending on whether corresponding motion planning query is yet finished
    - if tree contains path of complete edges leading to goal state, we have a solution
  - **construction:** repeat N-times:
    - extend tree by finding a state in tree and applicable action extending tree to a new state uniformly from (an approximation of) its reachable set
    - for all active edges: perform more computations on the corresponding motion planning query (e.g. using PRM) until total time spent on this query exceeds limit (increasing linearly in number of iterations)

1.4 Multi-Modal Motion Planning in Non-Expansive Spaces [Hauser, Latombe]

- presents a new PRM-based multi-modal planning algorithm for problems where the number of intersecting manifolds (forming the space $\mathcal{F}$ of feasible configurations) is finite
- PRMs can be slow when $\mathcal{F}$ contains "narrow passages" ($\mathcal{F}$ has poor *expansiveness*). Paper considers cases where $\mathcal{F}$ is non-expansive, i.e. consists of intersecting submanifolds of varying dimensionality ($\rightarrow$ arbitrarily thin passages)
  $\rightarrow$ **multi-modal** structure
- each submanifold $\mathcal{C}_\sigma$ corresponds to a *mode* σ, i.e. a set of contact points between robot and environment (e.g. all configurations in joint space where robot holds object in a specific grasp).

  Feasible subset $\mathcal{F}_\sigma \subset \mathcal{C}_\sigma$ corresponds to all such configurations which are collision-free.
- Planner chooses discrete sequence of modes (sequence of contacts to make and break) + continuous single-mode paths (challenging for high-dimensional submanifolds)
- **MULTI-MODAL-PRM:**
  - builds a PRM across modes by sampling configurations in each $\mathcal{F}_\sigma$ (maintaining roadmaps $\mathcal{R}_\sigma$) *and* in the transitions $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ between modes
  - build aggregate roadmap $\mathcal{R}$ by connecting roadmaps at matching transition configurations.
  - @sampling transition configuration:
    * $\mathcal{F}_\sigma \cap \mathcal{F}_{\sigma'}$ may have zero volume $\rightarrow$ sample explicitly from $\mathcal{C}_\sigma \cap \mathcal{C}_{\sigma'}$
    * existence of transition configuration is necessary for path from $q \in \mathcal{C}_\sigma$ to σ' (not sufficient, e.g. if $\mathcal{C}_\sigma$ not connected) and good indicator for existence (since less constraints on single-mode spaces)
  - If each $\mathcal{F}_\sigma$ is expansive (s.t. PRM is probabilistically complete), MULTI-MODAL-PRM finds feasible path if one exists, exponential convergence in number of samples
- **Incremental-MMPRM:**
  - using heuristics, it searches for a incrementally growing, but small candidate subset of modes which are likely to contain a solution path
  - heuristic "Search among feasible transitions" (SAFT):
    * uses the existence of a feasible transition between σ and σ' as a good indication that a feasible path exists as well
    * each mode pair (σ, σ') has priority. If sampling a transition between (σ, σ') fails (i.e. sampled transition is not valid, i.e. not in $\mathcal{F}_{\sigma,\sigma'}$), reduce priority
  - restrict MULTI-MODAL-PRM to this subset

1.5 Randomized Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task [Hauser]

- multi-modal systems occur in systems that make and break contact (because each discrete change of contact introduces or removes a closed-chain constraint on the robot's motion)
- new: continuous **infinity of modes** (e.g. corresponding to all possible contact placements of the robot against the object)
  → multi-modal planner must select finite subset of modes for single-mode planning queries
- main contribution: conditions necessary for the planner to find a path with a small number of modes sampled during planning
- **Random-MMP:**
  - adaptively **discretizes the continuous mode space** by building a search tree, where each iteration extends the tree with a mode switch sampled at random
  - at each mode switch, use sample-based planner to find continuous single-mode path
  - @implementation of `ExtendTree`:
    * k-mode extension: sample uniformly from the set of locally reachable (i.e. within k mode switches) states
    * "biasing samples towards large Voronoi regions": sample random state x, choose x_0 in tree with minimum distance towards x, plan 1-mode switch from x_0 to reach some state x_1
- under certain conditions Random-MMP is probabilistically complete, fast convergence
- performance can be improved using an informed expansion strategy, biased towards precomputed utility-tables

1.6 On the Probabilistic Foundations of Probabilistic Roadmap Planning [Hsu]

- Why don't we use algebraic planners?
  → would be overwhelmed by the cost of computing an exact representation of the feasible space
  → fast collision checker exist

- using **good sampling measures** speeds up PRM (idea: increase sampling density around narrow passages), sampling source (e.g. pseudo-random ↔ deterministic) has limited impact. Examples for sampling measures
  - workspace-based strategies (e.g. find small geometric corridors, kinematic singularities of robot)
  - filtering strategies: over-sample and reject unpromising samples using some geometric pattern
    * e.g. sample q u.a.r., sample q' w.r.t. Gaussian measure centered at q. If only one lies in $\mathcal{F}$, take it, otherwise discard both. (aims at sampling more densely around boundary)
    * e.g. estimate local visibility by testing connections among sampled configurations, discard roadmap nodes in these regions
  - adaptive strategies, e.g. first sample u.a.r., count number of (un)successful connections to nearby nodes → estimate for size of visibility set of that node

- **expansiveness** (visitibility property): $\mathcal{F}$ is $(\varepsilon, \alpha, \beta)$-expansive if each connected component $\mathcal{F}'$ satisfies:
  - each $q \in \mathcal{F}'$ sees at least an ε-fraction of $\mathcal{F}'$ (i.e. straight-line connection is in $\mathcal{F}'$)
  - for any set M of points in $\mathcal{F}'$: the fraction of points in M which sees at least a β-fraction of $\mathcal{F}' \setminus M$ is at least α

  → for $\mathcal{F}$ being expansive and two configurations in same connected component, PRM returns solution path with probability converging to 1 at an exponential rate

  → in poorly expansive $\mathcal{F}$ (i.e. α, β too small) one can construct examples for arbitrary $\varepsilon > 0, N > 0, \gamma \in (0, 1]$ s.t. PRM fails to return path with probability greater than γ

  → many spaces in practice have favorable visibility properties

  → does not directly depend on dimension s.t. PRM planners often scale up well in high dimensions

2 Heuristics for Sampling more effectively, more goal-driven

From here on:

State-space Search Methods:

- solve exact problem by sampling in state space
- use algorithmic heuristics that quickly solve approximations of the actual planning task (1) in order to estimate the distance to the goal from an arbitrary state
- heuristic can also be used (2) in order to find helpful actions to sample, direct forward search towards goal configuration

Effective approximations:

- delete relaxation (ignore delete lists for all actions) $\rightarrow$ H_{FF}-heuristic

2.1 Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic [Gammell]

- RRT* finds asymptotically optimal path from initial state to every state in space
  $\rightarrow$ expensive in high dimensions, inconsistent with single-query nature
- when sampling random states, solution only improves when adding states from *ellipsoidal* subset of space:
  - $f(x)$ = cost of optimal path from x_{start} to x_{goal} passing through x
  - subset $X_f \subset X$ of states that can improve the current solution of cost c_{best}

$$X_f = \{x \in X | f(x) < c_{best}\}$$

  - f unkown, use heuristic function $\hat{f}$, never overestimating the true cost, i.e. $\hat{f}(x) \leq f(x)$, $X_{\hat{f}} \supset X_f$
  - minimal cost $f(x)$ from x_{start} to x_{goal} through x = minimal cost $g(x)$ from x_{start} to x + minimal cost $h(x)$ from x to x_{goal}
  - if objective is minimization of path length Euclidean distance is admissible heuristic for both terms:

$$X_{\hat{f}} = \{x \in X | \|x_{start} - x\| + \|x - x_{goal}\| \leq c_{best}\}$$

    $\rightarrow$ equation of hyperellipsoid
- **Informed RRT*:**
  - behaves like RRT* until first solution is found
  - then only sample from $X_{\hat{f}}$ to possibly improve solution (using c_{best} as minimum cost among all solutions found so far)
- under strict assumptions (i.e. no obstacles) direct sampling from ellpsiodal subset results in linear convergence

2.2 The FF Planning System: Fast Plan Generation Through Heuristic Search [Nebel]

- forward state space search, hill-climbing, using a heuristic yielding a plan for relaxed problem that can be used for estimating goal distances
- each action has preconditions, add list, delete list. Planning task is triple of set of actions initial state, goals. *Relaxed* planning task obtained by ignoring delete list of all actions.
- heuristic **GRAPHPLAN:**
  - extend planning graph layer by layer (fact layer, action layer) until fact layer is reached containing all goal facts, no two of them are marked exclusive
  - then *recursive backward search*: for set of facts at layer i **choose achieving actions at layer** $i-1$ not exclusive of any actions already selected
  - on relaxed task no exclusive facts or actions, i.e. GRAPHPLAN only performs single sweep over the graph, polynomial runtime
- **heuristic value** is defined as number of all actions contained in relaxed solution, i.e. determines how many actions (if they could be performed in parallel without deletions) would be necessary to achieve the goal
- *enforced hill-climbing*:
  - search space is space of all reachable states, together with their heuristic evaluation
  - starting from initial state, repeatedly perform breadth-first search from current state s for a state s' with smaller heuristic value, add actions on path to s' at end of current plan

2.3 FFRob: An efficient heuristic for task and motion planning [Garrett]

- **extends symbolic FastForward (FF) planner** to motion planning by integrating symbolic and geometric search into one combined problem
- Problems:
  - infinitely many possible instantiations of operations
    → sample finitely many values
  - explicitly list all effects of each operation (many *reachable* literals can be affected)
    → maintain state representation containing *details* capturing geometric state (robot configuration, poses of objects, what object is in what grasp)
  - compute answers to queries in details, especially about reachability
    → use *conditional reachability graph*
- **conditional reachability graph:**
  - partial representation of connectivity of space of sampled configurations, contitioned on placeements of objects and what is in robot's hand
  - nodes = robot configurations c_i, annotated with $\{g, o, p\}$, edge iff a simple path is collision-free (checked on demand)
  - construction:
    * initialize with initial robot configuration and goal configuration
    * for each object generate set of sample poses (including initial and goal pose)
    * for each object pose and possible grasp of object use `SampleIK` to sample robot configuration. Sample additional configurations with hand near the grasp configuration
    * connect k nearest neighbors
  - query: use it to compute reachable test by constructing subgraph using only adges that are valid given the object the robot is holding, current placements of objects
- **idea:**
  - use forward search (e.g. hill-climbing) through state space with heuristic mapping a state to estimate cost to reach goal state from that state
  - for heuristic use relaxed plan graph (computed using CRG), obtained by ignoring delete list of all actions
- **FFRob heuristic:** extracts relaxed plan, returns number of actions it contains (i.e. like Nebel, only relaxation is different)
  - plan extraction works backwards starting with set of goal literals. For each literal seek "cheapest" achieving action, add it to relaxed plan
  - difference to FF heuristic: also consider reachability conditions

2.4 Backward-Forward Search for Manipulation Planning [Garrett]

- **hybrid spaces**: involve combination of *continuous* dimensions (e.g. pose of an object, configuration of a robot) and *discrete* dimensions (e.g. which object(s) a robot is holding)
- hybrid planning problems = multi-modal planning problems
- FF → FFRob (including geometric considerations, but need to pre-sample geometric roadmaps) → HBF
- **hybrid backward-forward (HBF) algorithm:**
  - **forward search** in state space:
    * use hill-climbing approach by remembering minimum heuristic value $h_{\min}$ for distance to reach state in goal space from current state
    * maintain queue, repeatedly choose state from queue, sample applicable action (from `RG`), compute resulting state
    * if there is no such action, grow `RG` for τ steps by popping a pair (C, a) and trying to find actions that can help achieve C
    * if lower heuristic value reached, pop entire queue, keep only initial state and current state
    * if no update, push current node back into queue again
  - **backward search** in relaxed space:
    * construct reachability graph working backward from goal constraints
    * focus sampling of actions towards goal, sample *useful* actions that might plausibly be contained in a plan to reach the goal (as they are contained in a plan in relaxed space)
- **reachability graph** (`RG`):
  - vertices = configurations, edges = applicable actions
  - constraint C can be *derived* from initial state s if C satisfied in s or $\exists$ action a that has C as an effect and each $C_i \in a.condition$ can be derived
  - *heuristic estimate* of distance from s to reach Γ: number of actions in derivation (found by H_{FF}-algorithm)
  - relaxation
    * goal constraints and action conditions solved independently
    * possible to plan efficiently in low-dimensional subspaces
    * plans in low-dimensional subspaces are allowed to violate constraints from full space
  - goal: iteratively construct subgraph of full `RG` that can derive goal constraints from some state s
  - subgraph will eventually contain actions that are first step towards solution. But may be that there are actions producing all condition constraints of some other action but incompatible (due to independent consideration of constraints)

3 Algorithms for Transition Sampling in Multi-Modal Settings

3.1 Multi-Modal Motion Planning for a Humanoid Robot Manipulation Task [Hauser]

- samples mode transitions at random, according to a strategy designed to distribute modes sparsely accross configuration space
- blind strategy (sampling u.a.r.) performs reasonably well
- already existing planners for tree-growing in PRM, strategies to sample in low-density areas:

  - **EST planner**: expans from node with probability inversely propertional to weight of node, e.g. number of nearby nodes
  - **RRT planner**: pick random point in configuration space, expand closest node toward that point

- **expansion strategies** to new mode:
  - *Blind*: choose adjacent mode u.a.r., sample configuration in transition region
  - *Reach/Utility-Informed*: sample according to expected reachbility/utility (precomputed in workspace of contacts)
    * reachability table R (filled by sampling and computing IK, collision check)
    * utility table stores expected distance contact can be pushed in absence of obstacles (calculated by Monte Carlo integration through reachable)
    * Reachable-Informed sampling: sample uniformly from R
    * Utility-Informed sampling: sample from R with probability proportional to Utility
  - *Push-centered*: ?