Motion Planning - Overview

May 5, 2017

1 Collection of Definitions

- **degrees of freedom**: maximum number of independent parameters needed to completely characterize the transformation applied to a robot
- **pose** = position + orientation
- **Manipulation Planning**: Solving the combination of all planning problems in one batch (instead of sequentially planning grasps, placements, inverse kinematics and motions)
- **Kinematics**: describes the motion of points, bodies and systems of bodies without considering the mass of each or the forces that caused the motion, "geometry of motion".
  Dynamics: considers forces required to cause motion, motion equations.
  Kinodynamic planning: class of problems for which velocity, acceleration, and force/torque bounds must be satisfied, together with kinematic constraints such as avoiding obstacles
- **Cartesian Space** position, orientation $\xleftarrow{\text{forward kinematics}}$ $\xrightarrow{\textbf{inverse kinematics}}$ **Configuration Space** joint angles

2 Transformation

transformation = rotation + translation

Important: Rotation and translation do not commute! Rotations take place about axes of the world frame, not the robot frame.

In 2D:

- transformation matrix T

$$T = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & x_t \\ \sin(\theta) & \cos(\theta) & y_t \\ 0 & 0 & 1 \end{pmatrix}$$

- represents a counterclockwise rotation *followed* by translation

In 3D:

- yaw (z), pitch (y), roll (x) (depending on the axis the rotation is about)
- rotation matrix = multiplication of yaw, pitch, roll rotation matrices
- $SE(3)$ is the set of transformations in $\mathbb{R}^3$

3 Sampling-Based Motion Planning

Basic Idea: Avoid explicit construction of $\mathcal{C}_{obs}$, instead conduct a search that probes the C-space with a sampling scheme. Use collision detector.

Probabilistic Completeness: Algorithm is *complete* if for any input it correctly reports whether there is a solution or not in a finite amount of time. Algorithm is *probabilistically complete* if with enough points, the probability that it finds an existing solution converges to one.

Probabilistic Roadmaps (PRM): For multiple queries, we preprocess the model in order to construct a topological graph $\mathcal{G}$ (the so-called *roadmap*) which represents a rich set of collision-free trajectories. This method is probabilistically complete.

Preprocessing Phase: Use a dense sequence α of random vertices (configurations) which are inserted step by step. If $\alpha(i) \in \mathcal{C}_{free}$, it is inserted as a vertex and all already inserted vertices within a certain neighborhood around $\alpha(i)$ in order of increasing distance are connected to $\alpha(i)$ - as long as this is possible without any collisions occurring.

As the algorithm mainly aims at establishing connectivity, connections within the same component are avoided. The resulting roadmap thus is a forest.

Query Phase: We assume that the graph obtained in the preprocessing phase is sufficiently complete to answer many queries. In a query, an initial configuration q_I and a goal configuration q_G are given. First, we connect them to $\mathcal{G}$ as we did for the vertices in α in the preprocessing phase. If q_I and q_G are successfully connected to other vertices in the graph, a search is performed for a path that connects the vertex q_I to the vertex q_G.

Rapidly-exploring Random Trees (RRT): Incrementally construct a topological graph $\mathcal{G}$ which is a search tree of feasible trajectories and is used for a single query. This method is probabilistically complete.

Construction: Along a dense and random sequence α of samples construct a search tree that gradually improves the resolution. When a new vertex $\alpha(i)$ is added, $\alpha(i)$ is connected to the nearest point among all those reached by $\mathcal{G}$ (if the nearest point lies in an endge, the edge is split into two).

With obstacles: When considering $\alpha(i)$, connect nearest point q_n to (newly created vertex) q_s which is the nearest configuration possible to the boundary of $\mathcal{C}_{free}$ along the direction towards $\alpha(i)$.

Planning:

- **Single-tree search:** grow a tree from q_I and periodically check whether it is already possible to connect the tree to q_G, e.g. by setting every 100^{th} sample to be q_G or setting every sample to be q_G with probability $1/100$.
- **Balanced, bidirectional search:** Grow two trees, one from q_I and the other from q_G. Need to make sure that the trees meet. Additionally, one can make sure that the search is balanced, i.e. both trees are of the same size.