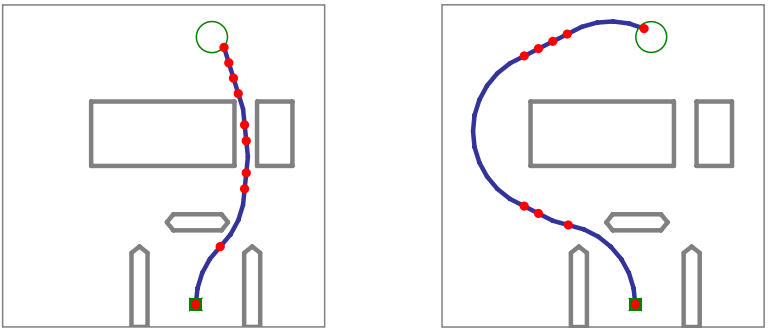
6 The Stochastic Motion Roadmap: A Sampling-Based Framework for Planning with Motion Uncertainty

In many applications of motion planning, the motion of the robot in response to commanded actions cannot be precisely predicted. Whether maneuvering a vehicle over unfamiliar terrain, steering a flexible needle through human tissue to deliver medical treatment, guiding a micro-scale swimming robot through turbulent water, or displaying a folding pathway of a protein polypeptide chain, the underlying motions cannot be predicted with certainty. But in many of these cases, a probabilistic distribution of feasible outcomes in response to commanded actions can be experimentally measured. This stochastic information is fundamentally different from a deterministic motion model. Though planning shortest feasible paths to the goal may be appropriate for problems with deterministic motion, shortest paths may be highly sensitive to uncertainties: the robot may deviate from its expected trajectory when moving through narrow passageways in the configuration space, resulting in collisions.

In this chapter, we develop a new motion planning framework that explicitly considers uncertainty in robot motion at the planning stage. Because future configurations cannot be predicted with certainty, we define a plan by actions that are a function of the robot's current configuration. A plan execution is successful if the robot does not collide with any obstacles and reaches the goal. The idea is to compute plans that maximize the probability of success.

The approach we develop here is a generalization of the motion planning algorithm developed in chapter 5 to consider a wider range of robot motions and uncertainty models. Our framework builds on the highly successful approach used in Probabilistic Roadmaps (PRM's): a learning phase followed by a query phase [118]. During the learning phase, a random (or quasi-random) sample of discrete states is selected in the configuration space, and a roadmap is built that represents their collision-free connectivity. During the query phase, the user specifies initial and goal states, and the roadmap is used to find a feasible path that connects the initial state to the goal, possibly optimizing some criteria such as minimum length. PRM's have successfully solved many path planning problems for applications such as robotic manipulators and mobile robots [51, 139]. The term "probabilistic" in PRM comes from the random sampling of

R. Alterovitz and K. Goldberg: Motion Planning in Medicine, STAR 50, pp. 75–89, 2008.
springerlink.com



(a) Minimizing path length

$p_s = 27\%$

(b) Maximize probability of success using SMR

$p_s = 57\%$

Fig. 6.1. The expected results of two plans to steer a Dubins-car mobile robot with left-right bang-bang steering and normally distributed motion uncertainty from an initial configuration (solid square) to a goal (open circle). Dots indicate steering direction changes. The Stochastic Motion Roadmap (SMR) introduces sampling of the configuration space and motion uncertainty model to generate plans that maximize the probability p_s that the robot will successfully reach the goal without colliding with an obstacle. Evaluation of p_s using multiple randomized simulations demonstrates that following a minimum length path under motion uncertainty (a) is substantially less likely to succeed than executing actions from an SMR plan (b).

states. An underlying assumption is that the collision-free connectivity of states is specified using boolean values rather than distributions.

In this chapter, we relax this assumption and combine a roadmap representation of the configuration space with a stochastic model of robot motion. The input to our method is a geometric description of the workspace and a motion model for the robot capable of generating samples of the next configuration that the robot may attain given the current configuration and an action. We require that the motion model satisfy the Markovian property: the distribution of the next state depends only on the action and current state, which encodes all necessary past history. As in PRM's, the method first samples the configuration space, where the sampling can be random [118], pseudo-random [140], or utility-guided [47]. We then sample the robot's motion model to build a *Stochastic Motion Roadmap (SMR)*, a set of weighted directed graphs with vertices as sampled states and edges encoding feasible state transitions and their associated probability of occurrence for each action.

The focus of our method is not to find a *feasible* motion plan, but rather to find an *optimal* plan that maximizes the probability that the robot will successfully reach a goal. Given a query specifying initial and goal configurations, we use the SMR to formulate a Markov Decision Process (MDP) where the "decision" corresponds to the action to be selected at each state in the roadmap. We solve

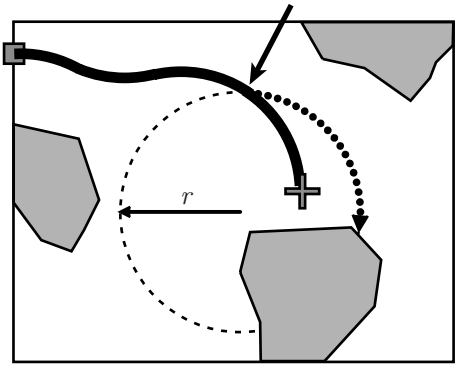


Fig. 6.2. The solid line path illustrates a motion plan from the start square to the goal (cross) for a nonholonomic mobile robot constrained to follow paths composed of continuously connected arcs of constant-magnitude curvature with radius of curvature r. If a deflection occurs at the location of the arrow, then the robot is unable to reach the goal due to the nonholonomic constraint, even if this deflection is immediately detected, since the robot cannot follow a path with smaller radius of curvature than the dotted line.

the MDP in polynomial time using Infinite Horizon Dynamic Programming. Because the roadmap is a discrete representation of the continuous configuration space and transition probabilities, the computed optimal actions are approximations of the optimal actions in continuous space that converge as the roadmap increases in size. Although the plan, defined by the computed actions, is fixed, the path followed by the robot may differ each time the plan is executed because different state transitions may occur due to motion uncertainty. As shown in figure 6.1, plans that explicitly consider uncertainty to maximize the probability of success can differ substantially from traditional shortest path plans.

In SMR, "stochastic" refers to the motion of the robot, not to the sampling of states. PRM's were previously extended to explore stochastic motion in molecular conformation spaces [21, 22], but without a planning component to optimize actions. Our SMR formulation is applicable to a variety of decision-based robotics problems. It is particularly suited for nonholonomic robots, for which a deflection in the path due to motion uncertainty can result in failure to reach the goal, even if the deflection does not result in an immediate collision. The expansion of obstacles using their Minkowski sum [139] with a circle corresponding to an uncertainty tolerance is often sufficient for holonomic robots for which deflections can be immediately corrected, but this does not address collisions resulting from a nonholonomic constraint as in figure 6.2. By explicitly considering motion uncertainty in the planning phase, we hope to minimize such failures.

Although we use the terms robot and workspace, SMR's are applicable to any motion planning problem that can be modeled using a continuous configuration space and discrete action set with uncertain transitions between configurations. In this chapter, we demonstrate a SMR planner using a variant of a Dubins car

with bang-bang control, a nonholonomic mobile robot that can steer its wheels far left or far right while moving forward but cannot go straight. This model can generate motion plans for steerable needles, a new class of flexible bevel-tip medical needles introduced in chapter 4 that clinicians can steer through soft tissue around obstacles to reach targets inaccessible to traditional stiff needles [15, 208]. As in many medical applications, considering uncertainty is crucial to the success of medical needle insertion procedures: the needle tip may deflect from the expected path due to tissue inhomogeneities that cannot be detected prior to the procedure. Due to uncertainty in predicted needle/tissue interactions, needle steering is ill-suited to shortest-path plans that may guide the needle through narrow passageways between critical tissue such as blood vessels or nerves. By explicitly considering motion uncertainty using an SMR, we obtain solutions that result in possibly longer paths but that improve the probability of success.

6.0.1 Related Work

Motion planning can consider uncertainty in *sensing* (the current state of the robot and workspace is not known with certainty) and *predictability* (the future state of the robot and workspace cannot be deterministically predicted even when the current state and future actions are known) [139]. Extensive work has explored uncertainty associated with robot sensing, including Simultaneous Localization and Mapping (SLAM) and Partially Observable Markov Decision Processes (POMDP's) to represent uncertainty in the current state [51, 202]. In this chapter, we assume the current state is known (or can be precisely determined from sensors), and we focus on the latter type of uncertainty, predictability.

Predictability can be affected by uncertainty in the workspace and by uncertainty in the robot's motion. Previous and ongoing work addresses many aspects of uncertainty in the workspace, including uncertainty in the goal location, such as in pursuit-evasion games [139, 142], and in dynamic environments with moving obstacles [147, 204, 206]. A recently developed method for grasp planning uses POMDP's to consider uncertainty in the configuration of the robot and the state of the objects in the world [105]. In this chapter we focus explicitly on the case of uncertainty in the robot's motion rather than in goal or obstacle locations.

Apaydin et al. previously explored the connection between probabilistic roadmaps and stochastic motions using Stochastic Roadmap Simulation (SRS), a method designed specifically for molecular conformation spaces [21, 22]. SRS, which formalizes random walks in a roadmap as a Markov Chain, has been successfully applied to predict average behavior of molecular motion pathways of proteins and requires orders of magnitude less computation time than traditional Monte-Carlo molecular simulations. However, SRS cannot be applied to more general robotics problems, including needle steering, because the probabilities associated with state transitions are specific to molecular scale motions and the method does not include a planning component to optimize actions.

Considering uncertainty in the robot's response to actions during planning results in a stochastic optimal control problem where feedback is generally required for success. Motion planners using grid-based numerical methods and geometric analysis have been applied to robots with motion uncertainty (sometimes combined with sensing uncertainty) using cost-based objectives and worst-case analysis [39, 141, 143]. MDP's, a general approach that requires explicitly defining transition probabilities between states, have also been applied to motion planning by subdividing the workspace using regular grids and defining transition probabilities for motions between the grid cells [63, 81, 139]. This was the motion planning approach taken in chapter 5. These methods differ from SMR's since they use grids or problem-specific discretization.

Many existing planners for deterministic motion specialize in finding feasible paths through narrow passageways in complex configuration spaces using specialized sampling [20, 38] or learning approaches [158]. Since a narrow passageway is unlikely to be robust to motion uncertainty, finding these passageways is not the ultimate goal of our method. Our method builds a roadmap that samples the configuration space with the intent of capturing the uncertain motion transition probabilities necessary to compute optimal actions.

We apply SMR's to needle steering, a type of nonholonomic control-based motion planning problem. Nonholonomic motion planning has a long history in robotics and related fields [51, 139]. Past work has addressed deterministic curvature-constrained path planning with obstacles where a mobile robot's path is, like a car, constrained by a minimum turning radius [36, 111, 138, 152, 188]. For steerable needles, Park et al. applied a numeric diffusion-based method but did not consider obstacles or motion uncertainty [171]. Alterovitz et al. proposed an MDP formulation to find a stochastic shortest path for a steerable needle to a goal configuration, subject to user-specified "cost" parameters for direction changes, insertion distance, and collisions [15]. Because these costs are difficult to quantify, Alterovitz et al. introduced the objective function of maximizing probability of success [7]. These methods use a regular grid of states and an ad-hoc, identical discretization of the motion uncertainty distribution at all states. The methods do not consider the advantages of sampling states nor the use of sampling to estimate motion models.

6.0.2 SMR Contributions

SMR planning is a general framework that combines a roadmap representation of configuration space with the theory of MDP's to explicitly consider motion uncertainty at the planning stage to maximize the probability of success. SMR's use sampling to both learn the configuration space (represented as states) and to learn the stochastic motion model (represented as state transition probabilities). Sampling reduces the need for problem-specific geometric analysis or discretization for planning. As demonstrated by the success of PRM's, sampling states is useful for modeling complex configuration spaces that cannot be easily represented geometrically and extends well to higher dimensions. Random or quasi-random sampling reduces problems associated with regular grids of states,

including the high computational complexity in higher dimensions and the sensitivity of solutions and runtimes to the selection of axes [139]. Sampling the stochastic motion model enables the use of a wide variety of motion uncertainty representations, including directly sampling experimentally measured data or using parameterized distributions such as a Gaussian distribution. This greatly improves previous Markov motion planning approaches that impose an ad-hoc, identical discretization of the transition probability distributions at all states (as was done in chapter 5).

Although SMR is a general framework, it provides improvements for steerable needle planning compared to previously developed approaches specifically designed for this application. Previous planners do not consider uncertainty in needle motion [171], or apply simplified models that only consider deflections at decision points and assume that all other motion model parameters are constant (as was done in chapter 5). Because we use sampling to approximate the motion model rather than a problem-specific geometric approximation, we eliminate the discretization error at the initial configuration and can easily include a more complex uncertainty model that considers arbitrary stochastic models for both insertion distance and radius of curvature. SMR's increase flexibility and decrease computation time; for problems with equal workspace size and expected values of motion model parameters, a query that requires over a minute to solve using a grid-based MDP due to the large number of states needed to bound discretization error (as in chapter 5) requires just 6 seconds using an SMR.

6.1 Algorithm

6.1.1 Input

To build an SMR, the user must first provide input parameters and function implementations to describe the configuration space and robot motion model. A configuration of the robot and workspace is defined by a vector $x \in C = \Re^d$, where d is the number of degrees of freedom in the configuration space C. At any configuration x, the robot can perform an action from a discrete action set U of size w. The bounds of the configuration space are defined by B_i^{min} and B_i^{max} for $i = 1, \ldots, d$, which specify the minimum and maximum values, respectively, for each configuration degree of freedom i. The functions `isCollisionFree`(x) and `isCollisionFreePath`(x, y) implicitly define obstacles within the workspace; the former returns `false` if configuration x collides with an obstacle and `true` otherwise, and the latter returns `false` if the path (computed by a local planner [118]) from configuration x to y collides with an obstacle and `true` otherwise. (We consider exiting the workspace as equivalent to colliding with an obstacle.) The function `distance`(x, y) specifies the distance between two configurations x and y, which can equal the Euclidean distance in d-dimensional space or some other user-specified distance metric. The function `generateSampleTransition`(x, u) implicitly defines the motion model and its probabilistic nature; this function returns a sample from a known probability distribution for the next configuration given that the robot is currently in configuration x and will perform action u.

Procedure 1. `buildSMR`

Require:
n: number of nodes to place in the roadmap
U: set of discrete robot actions
m: number of sample points to generate for each transition
Ensure:
SMR containing states V and transition probabilities (weighted edges) E^u for each action $u \in U$

$V \leftarrow \emptyset$
for all $u \in U$ **do**
  $E^u \leftarrow \emptyset$
end for
while $|V| < n$ **do**
  $q \leftarrow$ random state sampled from the configuration space
  if `isCollisionFree`(q) **then**
    $V \leftarrow V \cup \{q\}$
  end if
end while
for all $s \in V$ **do**
  for all $u \in U$ **do**
    for all $(t, p) \in$ `getTransitions`(V, s, u, m) **do**
      $E^u \leftarrow E^u \cup \{(s, t, p)\}$
    end for
  end for
end for
return weighted directed graphs $G^u = (V, E^u)\ \forall\ u \in U$

6.1.2 Building the Roadmap

We build the stochastic motion roadmap using the algorithm `buildSMR` defined in Procedure 1. The roadmap is defined by a set of vertices V and sets of edges E^u for each action $u \in U$. The algorithm first samples n collision-free states in the configuration space and stores them in V. In our implementation, we use a uniform random sampling of the configuration space inside the bounds defined by (B_i^{min}, B_i^{max}) for $i = 1, \ldots, d$, although other random distributions or quasi- random sampling methods could be used [51, 140]. For each state $s \in V$ and an action $u \in U$, `buildSMR` calls the function `getTransitions`, defined in Procedure 2, to obtain a set of possible next states in V and probabilities of entering those states when action u is performed. We use this set to add to E^u weighted directed edges (s, t, p), which specify the probability p that the robot will transition from state $s \in V$ to state $t \in V$ when currently in state s and executing action u.

The function `getTransitions`, defined in Procedure 2, estimates state transition probabilities. Given the current state s and an action u, it calls the problem-specific function `generateSampleTransition`(x, u) to generate a sample configuration q and then selects the state $t \in V$ closest to q using the problem-specific `distance` function. We repeat this motion sampling m times and then

Procedure 2. `getTransitions`

Require:
V: configuration space samples
s: current robot state, $s \in V$
u: action that the robot will execute, $u \in U$
m: number of sample points to generate for this transition

Ensure:
List of tuples (t, p) where p is the probability of transitioning from state $s \in V$ to state $t \in V$ after executing u.

```
R ← ∅
for i = 1 to m do
  q = generateSampleTransition(s, u)
  if isCollisionFreePath(s, q) then
    t ← arg min_{t∈V} distance(q, t)
  else
    t ← obstacle state
  end if
  if (t, p) ∈ R for some p then
    Remove (t, p) from R
    R ← R ∪ {(t, p + 1/m)}
  else
    R ← R ∪ {(t, 1/m)}
  end if
end for
return R
```

estimate the probability of transitioning from state s to t as the proportion of times that this transition occurred out of the m samples. If there is a collision in the transition from state s to t, then the transition is replaced with a transition from s to a dedicated "obstacle state," which is required to estimate the probability that the robot collides with an obstacle.

This algorithm has the useful property that the transition probability from state s to state t in the roadmap equals the fraction of transition samples that fall inside state t's Voronoi cell. This property is implied by the use of nearest neighbor checking in `getTransitions`. As $m \to \infty$, the probability p of transitioning from s to t will approach, with probability 1, the integral of the true transition distribution over the Voronoi cell of t. As the number of states $n \to \infty$, the expected volume V_t of the Voronoi cell for state t equals $V/n \to 0$, where V is the volume of the configuration space. Hence, the error in the approximation of the probability p due to the use of a discrete roadmap will decrease as n and m increase.

6.1.3 Solving a Query

We define a query by an initial configuration s^* and a set of goal configurations T^*.

Using the SMR and the query input, we build an $n \times n$ transition probability matrix $P(u)$ for each $u \in U$. For each tuple $(s, t, p) \in E^u$, we set $P_{st}(u) = p$ so

$P_{st}(u)$ equals the probability of transitioning from state s to state t given that action u is performed. We store each matrix $P(u)$ as a sparse matrix that only includes pointers to a list of non-zero elements in each row and assume all other entries are 0.

We define $p_s(i)$ to be the probability of success given that the robot is currently in state i. If the position of state i is inside the goal, $p_s(i) = 1$. If the position of state i is inside an obstacle, $p_s(i) = 0$. Given an action u_i for some other state i, the probability of success will depend on the response of the robot to the action and the probability of success from the next state. The goal of our motion planner is to compute an optimal action u_i to maximize the expected probability of success at every state i:

$$p_s(i) = \max_{u_i} \{E[p_s(j)|i, u_i]\}, \tag{6.1}$$

where the expectation is over j, a random variable for the next state. Since the roadmap is a discrete approximation of the continuous configuration space, we expand the expected value in Eq. 6.1 to a summation:

$$p_s(i) = \max_{u_i} \left\{ \sum_{j \in V} P_{ij}(u_i) p_s(j) \right\}. \tag{6.2}$$

We observe that Eq. 6.2 is an MDP and has the form of the Bellman equation for a stochastic shortest path problem [34]:

$$J^*(i) = \max_{u_i} \sum_{j \in V} P_{ij}(u_i) \left(g(i, u_i, j) + J^*(j) \right). \tag{6.3}$$

where $g(i, u_i, j)$ is a "reward" for transitioning from state i to j after action u_i. In our case, $g(i, u_i, j) = 0$ for all i, u_i, and j, and $J^*(i) = p_s(i)$.

Stochastic shortest path problems of the form in Eq. 6.3 can be optimally solved using infinite horizon dynamic programming. For stationary Markovian problems, the configuration space does not change over time, which implies that the optimal action at each state is purely a function of the state without explicit dependence on time. Infinite horizon dynamic programming is a type of dynamic programming (DP) in which there is no finite time horizon [34]. Specifically, we use the value iteration algorithm [34], which iteratively updates $p_s(i)$ for each state i by evaluating Eq. 6.3. This generates a DP look-up table containing the optimal action u_i and the probability of success $p_s(i)$ for each $i \in V$.

The algorithm is guaranteed to terminate in n (the number of states) iterations if the transition probability graph corresponding to some optimal stationary policy is acyclic [34]. Violation of this requirement can occur in rare cases in which a cycle is feasible and deviating from the cycle will result in imminent failure. To remove this possibility, we introduce a small penalty γ for each transition by setting $g(i, u_i, j) = -\gamma$ in Eq. 6.3. Increasing γ has the effect of giving preference to shorter paths at the expense of a less precise estimate of the probability of success, where the magnitude of the error is (weakly) bounded by γn.

6.1.4 Computational Complexity

Building an SMR requires $O(n)$ time to create the states V, not including collision detection. Generating the edges in E^u requires $O(wn)$ calls to `generateTransitions`, where $w = |U|$. For computational efficiency, it is not necessary to consolidate multiple tuples with the same next state t; the addition $p + 1/m$ can be computed automatically during value iteration. Hence, each call requires $O(mdn)$ time using brute-force nearest neighbor checking. For certain low-dimensional configuration spaces, this can be reduced to $O\left(m \exp\left(d\right) \log\left(n\right)\right)$ using kd-trees [219]. Hence, the total time complexity of building an SMR is $O\left(wmdn^2\right)$ or $O\left(wm \exp\left(d\right) n \log\left(n\right)\right)$. This does not include the cost of n state collision checks and nm checks of collision free paths, which are problem-specific and may increase the computational complexity depending on the workspace definition.

Solving a query requires building the transition probability matrices and executing value iteration. Although the matrices $P_{ij}(u)$ each have n^2 entries, we do not store the zero entries as described above. Since the robot will generally only transition to a state j in the spatial vicinity of state i, each row of $P_{ij}(u)$ has only k nonzero entries, where $k << n$. Building the sparse matrices requires $O(wkn)$ time. By only accessing the nonzero entries of $P_{ij}(u)$ during the value iteration algorithm, each iteration for solving a query requires only $O(wkn)$ rather than $O(wn^2)$ time. Thus, the value iteration algorithm's total time complexity is $O(wkn^2)$ since the number of iterations is bounded by n. To further improve performance, we terminate value iteration when the maximum change ϵ over all states is less than some user-specified threshold ϵ^*. In our test cases, we used $\epsilon^* = 10^{-7}$, which resulted in far fewer than n iterations.

6.2 SMR for Medical Needle Steering

We assume the workspace is extracted from a medical image, where obstacles represent tissues that cannot be cut by the needle, such as bone, or sensitive tissues that should not be damaged, such as nerves or arteries. As in chapter 5, we consider motion plans in an imaging plane since the speed/resolution trade-off of 3-D imaging modalities is generally poor for 3-D interventional applications. We assume the needle follows paths of radius of curvature r and moves a distance δ between image acquisitions that are used to determine the current needle position and orientation. We do not consider motion by the needle out of the imaging plane or needle retraction, which modifies the tissue and can influence future insertions. When restricted to motion in a plane, the bevel direction can be set to point left ($b = 0$) or right ($b = 1$) [208]. Due to the nonholonomic constraint imposed by the bevel, the motion of the needle tip can be modeled as a bang-bang steering car, a variant of a Dubins car that can only turn its wheels far left or far right while moving forward [7, 208].

Clinicians performing medical needle insertion procedures must consider uncertainty in the needle's motion through tissue due to patient differences and the difficulty in predicting needle/tissue interaction. Bevel direction changes further

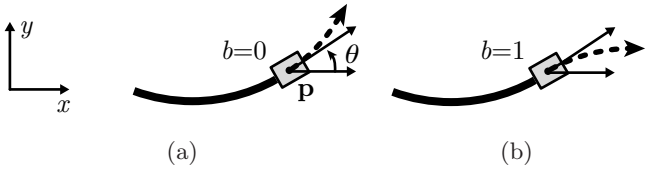


Fig. 6.3. The state of a bang-bang steering car is defined by point $\mathbf{p}$, orientation θ, and turning direction b (a). The car moves forward along an arc of constant curvature and can turn either left (a) or right (b).

increase uncertainty due to stiffness along the needle shaft. Medical imaging in the operating room can be used to measure the needle's current position and orientation to provide feedback to the planner [55, 67], but this measurement by itself provides no information about the effect of future deflections during insertion due to motion uncertainty.

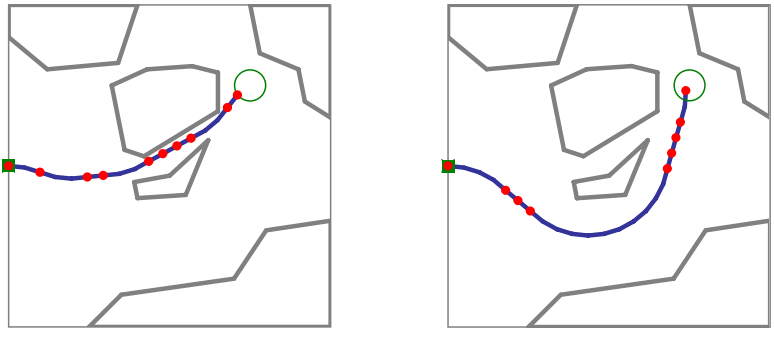
Stochastic motion roadmaps offer features particularly beneficial for medical needle steering. First, SMR's explicitly consider uncertainty in the motion of the needle. Second, intra-operative medical imaging can be combined with the fast SMR queries to permit control of the needle in the operating room without requiring time-consuming intra-operative re-planning.

6.2.1 SMR Implementation

We formulate the SMR for a bang-bang steering car, which can be applied to needle steering. The state of such a car is fully characterized by its position $\mathbf{p} = (x, y)$, orientation angle θ, and turning direction b, where b is either left ($b = 0$) or right ($b = 1$). Hence, the dimension of the state space is $d = 4$, and a state i is defined by $s_i = (x_i, y_i, \theta_i, b_i)$, as illustrated in figure 6.3. We encode b_i in the state since it is a type of history parameter that is required by the motion uncertainty model. Since an SMR assumes that each component of the state vector is a real number, we define the binary b_i as the floor of the fourth component in s_i, which we bound in the range $[0, 2)$.

Between sensor measurements of state, we assume the car moves a distance δ. The set U consists of two actions: move forward turning left ($u = 0$), or move forward turning right ($u = 1$). As the car moves forward, it traces an arc of length δ with radius of curvature r and direction based on u. We consider r and δ as random variables drawn from a given distribution. In this chapter, we consider $\delta \sim N(\delta_0, \sigma_{\delta_a})$ and $r \sim N(r_0, \sigma_{r_a})$, where N is a normal distribution with given mean and standard deviation parameters and $a \in \{0, 1\}$ indicates direction change. We implement `generateSampleTransition` to draw random samples from these distributions. Although the uncertainty parameters can be difficult to measure precisely, even rough estimates may be more realistic than using deterministic transitions when uncertainty is high.

We define the workspace as a rectangle of width x_{max} and height y_{max} and define obstacles as polygons in the plane. To detect obstacle collisions,



(a) Minimizing path length $p_s = 35\%$

(b) Maximizing p_s using SMR $p_s = 83\%$

Fig. 6.4. Explicitly considering motion uncertainty using an SMR planner improves the probability of success

we use the zero-winding rule [99]. We define the distance between two states s_1 and s_2 to be the weighted Euclidean distance between the poses plus an indicator variable to ensure the turning directions match: $\texttt{distance}(s_1, s_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + \alpha(\theta_1 - \theta_2)^2} + M$, where $M \to \infty$ if $b_1 \neq b_2$, and $M = 0$ otherwise. For fast nearest-neighbor computation, we use the CGAL implementation of kd-trees [195]. Since the `distance` function is non-Euclidean, we use the formulation developed by Atramentov and LaValle to build the kd-tree [23]. We define the goal T^* as all configuration states within a ball of radius t^r centered at a point $\mathbf{t}^*$.

6.2.2 Results

We implemented the SMR planner in C++ and tested the method on workspaces of size $x_{max} = y_{max} = 10$ with polygonal obstacles as shown in shown in figure 6.1 and figure 6.4. We set the robot parameters $r_0 = 2.5$ and $\delta_0 = 0.5$ with motion uncertainty parameters $\sigma_{\delta_0} = 0.1$, $\sigma_{\delta_1} = 0.2$, $\sigma_{r_0} = 0.5$, and $\sigma_{r_1} = 1.0$. We set parameters $\gamma = 0.00001$ and $\alpha = 2.0$. We tested the motion planner on a 2.2 GHz AMD Opteron PC. Building the SMR required approximately 1 minute for $n = 50{,}000$ states, executing a query required 6 seconds, and additional queries for the same goal required less than 1 second of computation time for both example problems.

We evaluate the plans generated by SMR with multiple randomized simulations. Given the current state of the robot, we query the SMR to obtain an optimal action u. We then execute this action and compute the expected next state. We repeat until the robot reaches the goal or hits an obstacle, and we illustrate the resulting expected path. Since the motion response of the robot to actions is not deterministic, success of the procedure can rarely be guaranteed. To estimate p_s, we run the simulation 100 times, sampling the next state from the transition probability distribution rather than selecting the expected

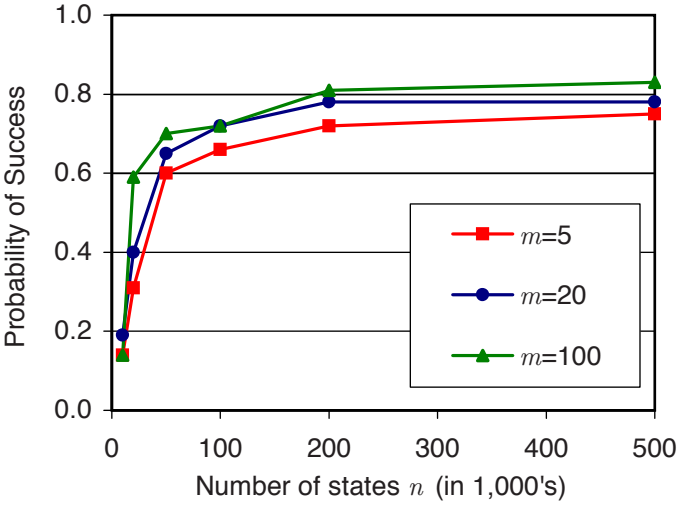


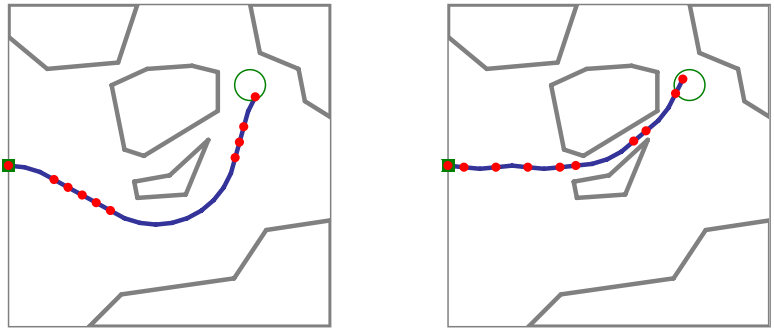
Fig. 6.5. Effect of the number of states n and number of motion samples m on the probability of success p_s

value, and we compute the number of goal acquisitions divided by the number of obstacle collisions.

In figure 6.4(b), we illustrate the expected path using an SMR with $m = 100$ motion samples and $n = 500{,}000$ states. As in figure 6.1(b), the robot avoids passing through a narrow passageway near the goal and instead takes a longer route. The selection of the longer path is not due to insufficient states in the SMR; there exist paths in the SMR that pass through the narrow gaps between the obstacles. The plan resulting in a longer path is selected purely because it maximizes p_s.

The probability of success p_s improves as the sampling density of the configuration space and the motion uncertainty distribution increase, as shown in figure 6.5. As n and m increase, $p_s(s)$ is more accurately approximated over the configuration space, resulting in better action decisions. However, p_s effectively converges for $n \geq 100{,}000$ and $m \geq 20$, suggesting the inherent difficulty of the motion planning problem. Furthermore, the expected path does not substantially vary from the path shown in figure 6.4(b) for $n \geq 50{,}000$ and $m \geq 5$. The number of states required by the SMR planner is far smaller than the 800,000 states required for a similar problem using a grid-based approach with bounded error [7].

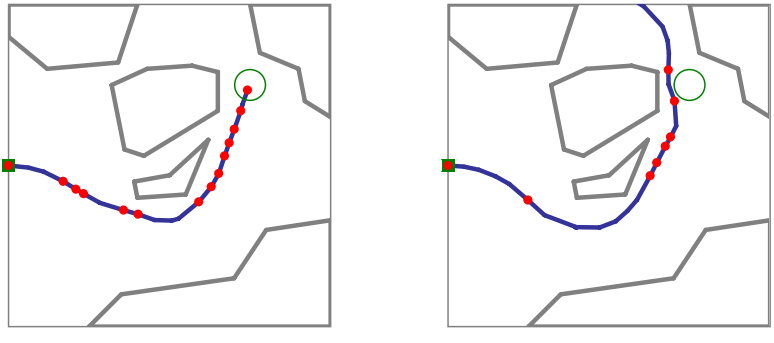
In figure 6.4(a), we computed the optimal shortest path assuming deterministic motion of the robot using a fine regular discrete grid with 816,080 states for which the error due to discretization is small and bounded [7]. We estimate p_s using the same simulation methodology as for an SMR plan, except that we compute the shortest path for each query. The expected shortest path passes



(a) High uncertainty
$p_s = 78\%$

(b) Low uncertainty
$p_s = 87\%$

Fig. 6.6. The level of uncertainty affects SMR planning results. In cases of low uncertainty (with 75% reduction in distribution standard deviations), the expected path resembles a deterministic shortest path due to the small influence of uncertainty on p_s and the effect of the penalty term γ. In both these examples, the same $n = 200{,}000$ states were used in the roadmap.



(a) Successful procedure

(b) Unsuccessful procedure

Fig. 6.7. Two simulated procedures of needle steering, one successful (a) and one unsuccessful due to effects of uncertain motion (b), using an SMR with $n = 50{,}000$ states

through a narrow passage between obstacles and the resulting probability of success is substantially lower compared to the SMR plan. The result was similar for the example in figure 6.1; explicitly considering motion uncertainty improved the probability of success.

To further illustrate the importance of explicitly considering uncertainty during motion planning, we vary the standard deviation parameters σ_{δ_0}, σ_{δ_1}, σ_{r_0}, and σ_{r_1}. In figure 6.6, we compute a plan for a robot with each standard deviation parameter set to a quarter of its default value. For this low uncertainty case,

the uncertainty is not sufficient to justify avoiding the narrow passageway; the penalty γ causes the plan to resemble the deterministic shortest plan in figure 6.4(a). Also, p_s is substantially higher because of the lower uncertainty.

In figure 6.7, we execute the planner in the context of an image-guided procedure. We assume the needle tip position and orientation is extracted from a medical image and then execute a query, simulate the needle motion by drawing a sample from the motion uncertainty distribution, and repeat. The effect of uncertainty can be seen as deflections in the path. In practice, clinicians could monitor $p_s(s)$ for the current state s as the procedure progresses.

6.3 Conclusion and Open Problems

In many motion planning applications, the response of the robot to commanded actions cannot be precisely predicted. We introduce the Stochastic Motion Roadmap (SMR), a new motion planning framework that explicitly considers uncertainty in robot motion to maximize the probability that a robot will avoid obstacle collisions and successfully reach a goal. SMR planners combine the roadmap representation of configuration space used in PRM with the theory of MDP's to explicitly consider motion uncertainty at the planning stage.

To demonstrate SMR's, we considered a nonholonomic mobile robot with bang-bang control, a type of Dubins-car robot model that can be applied to steering medical needles through soft tissue. Needle steering, like many other medical procedures, is subject to substantial motion uncertainty and is therefore ill-suited to shortest-path plans that may guide medical tools through narrow passageways between critical tissues. Using randomized simulation, we demonstrated that SMR's generate motion plans with significantly higher probabilities of success compared to traditional shortest-path approaches.

Because SMR is a framework, extensions to the underlying methods can be applied to wide variety of problems. Several extensions that would expand the applicability of SMR include considering actions in a continuous range rather than solely from a discrete set, investigating more sophisticated sampling methods for generating configuration samples and for estimating transition probabilities, and integrating the effects of sensing uncertainty. We hope that SMR can be applied to new biomedical and industrial problems where uncertainty in motion should be considered to maximize the probability of success.