Soft Subdivision Search in Motion Planning

Chee K. Yap
Courant Institute, New York University
New York, NY 10012 USA
Email: yap@cs.nyu.edu

***Abstract*—The main paradigm for practical motion planning in the last two decades is Probabilistic Road Map (PRM). We propose an alternative paradigm called *Soft Subdivision Search* (SSS). The SSS approach is based on two ingredients: the standard subdivision of space, coupled with *soft predicates*. Such predicates are conservative and convergent relative to exact predicates. This leads to a new class of *resolution-exact* planners.**

We view PRM and SSS as frameworks for broad classes of planners. There are many parallels between SSS and PRM: both frameworks are versatile, practical, easy to implement, with adaptive local complexity. The critical difference is that SSS avoids the Halting Problem of PRM. We address three issues:
(1) We axiomatize some basic properties that allow resolution-exact planners to be constructed in the SSS framework.
(2) We show how soft predicates can be effectively and correctly implemented using numerical approximations.
(3) We recover exact planners by extending our framework.

The SSS framework is a theoretically sound basis for new classes of algorithms in motion planning and beyond. We discuss the prospects of SSS planners being able to solve currently challenging problems and their relation to PRM.

I. Introduction

Motion Planning is a fundamental problem in robotics. It originated as the "findpath problem" in Artificial Intelligence [3]. In the 1980s, computational geometers began the algorithmic study of motion planning [19, 9], focusing on *exact planners*: such planners return a path if any exists, and report "No Path" otherwise. It was early observed that there are two universal approaches for motion planning: Cell Decomposition [16] and Retraction [19]. After the work of Canny [4], the retraction is popularly known as the "roadmap approach". In the 1990's the roadmap approach takes another turn.

¶1. Theory. Today, exact motion planning is textbook material [6, 13] and continues to be actively investigated (e.g., [8]). Nevertheless its impact on robotics is modest: Zhang et al. [22] noted that exact implementations have been limited to 3 degrees of freedom, and for simple robots only. Exactness is costly as it implicitly requires algebraic numbers. The techniques of Exact Geometric Computation [20] can avoid direct manipulation of algebraic numbers, and is practical for many basic problems. Nevertheless, the usual expedient is to replace exact arithmetic by approximate machine arithmetic, leading to the ubiquitous problems of numerical non-robustness. Efficiency aside, there is a fundamental but less well-known barrier: *the Turing computability of exact algorithms for most non-algebraic problems is unknown* [5]. This barrier exists in most problems beyond kinematic motion planning. See [5] for a rare non-algebraic planning problem that is computable using transcendental number theory.

¶2. Practice. Since the mid 1990's, the Probabilistic Road Map (PRM) paradigm has been dominant; Kavraki et al. [11] gave the basic formulation. A more general viewpoint [12] regards PRM as the probabilistic form of "Sampling Road Maps". For simplicity, we use "PRM" as surrogate for all sampling methods. Quoting Choset et al [6, p.201]: "*PRM, EST, RRT, SRT, and their variants have changed the way path planning is performed for high-dimensional robots. They have also paved the way for the development of planners for problems beyond basic path planning.*" In his invited talk at the workshop[1] on open problems in this field, J.C. Latombe stated that the major open problem of PRM is that it does not know how to terminate when there is no path. In practice, the algorithm is timed-out, but this leads to problems such as the "Climbers Dilemma" giving rise to issues like the "Climbers Dilemma" (Bretl 2005). We will call it the **Halting Problem for PRM** (like the Halting Problem in Turing machines, it is a semi-decidable). It is the ultimate form of the "Narrow Passage Problem" [6, p. 201]. Latombe's talk suggested promising approaches such as Lazy PRM [2], but clearly a large part of the literature is devoted to this issue. The theoretical basis for PRM algorithms is probabilistic completeness [10], or more generally, "sampling completeness". But the Halting Problem is inherent in such completeness.

¶3. Common Ground. We seek a common ground for theory and practice: stronger theoretical guarantees without the inordinate demands of exactness. Fortunately, exactness is a mismatch for robotics. This is evident from the remark that all physical constants, devices and sensors have limited accuracy. Yet it does not absolve us from mathematical precision if we wish the theoretical development of robotic algorithms to thrive. This tension between practice and theory has led to their divergent paths described above. We turn to the idea of "resolution complete" methods, noting that the early work of Brooks and Lozano-Perez [3] was already on this track. It is known that resolution complete methods can avoid the Halting Problem [22]. The notion of resolution completeness is seldom scrutinized. Our companion paper [17] pointed out some untenable or lacking interpretations, and proposed

[1] IROS 2011 Workshop on Progress and Open Problems in Motion Planning, September 30, 2011, San Francisco.

the notion of **resolution-exactness**. Surprisingly, resolution-exactness has "inherent" indeterminacy, even for deterministic algorithms using exact predicates. This indeterminacy is mild compared to sampling completeness, and a good match for the needs of robotics.

Resolution-exactness is basically a numeric/analytic concept, but exactness is not an adjective we associate with numerics. Our main contribution is to explicate ideas that are intuitively known to practitioners, to provide a clear foundation for theoretical algorithm designers to ply their craft. This seems critical: without a clear foundation, many theoreticians would shun such algorithms. Exact computation has served as *the* foundation for over 2 decades, but its limitations are showing (as discussed above). What we need is a viable *exact numerical foundation* (cf. Smale's effort in this direction). We will argue that algorithms in our framework are not just theoretically-sound but implementable and useful.

¶4. Overview of Paper. Our full paper [21] aims to expose the foundations of resolution exactness. There are four themes:
(0) We take a leaf from the success of PRM research: the simplicity and generality of PRM framework ensures that implementers of this framework can get easy access to a whole family of algorithms. This led us to formulate an analogous framework called **soft subdivision search** (SSS). Curious aside: PRM and SSS correspond (resp.) roughly to the two universal approaches to motion planning, namely retraction/roadmap and cell-decomposition.
(1) Next, we axiomatize the setting for SSS planners by considering the problem of finding paths in $Y \subseteq X$ connecting given $\alpha, \beta \in Y$ where X is a normed linear space. The boxes in subdivision trees may be shapes such as simplices. The goal is to derive general principles to guide the design of SSS planners. Here we must avoid the temptation of excessive generality, leading to weak generic results about metric spaces. We aim at a balance which captures a large class of problems about which non-trivial theorems can be proved.
(2) One bane of exact algorithms is the "implementation gap". Exact primitives are typically algebraic but implementers use machine arithmetic approximation, thereby forfeiting all the guarantees of exactness. We derive principles for correct implementation of soft predicates, and derive error estimates to allow machine arithmetic filters.
(3) SSS planners take an input resolution parameter ε which must be positive. If we admit $\varepsilon = 0$, the planners become non-halting like PRM. We indicate solutions, leading to new classes of exact algorithms.

Within the constraints of this workshop presentation, we only address theme (0) and give a critical evaluation of SSS planners for solving challenging planning problems.

II. Basics

To aid further discussion, we need some definitions. Our terminology is quite standard, but we rely on readers' intuition for now. Consider the standard kinematic motion planning problem for a fixed rigid robot $R_0 \subseteq \mathbb{R}^k$ ($k = 2, 3$) which defines a configuration space $C_{space} = C_{space}(R_0)$. The planner input is $(\varepsilon, \Omega, \alpha, \beta)$ where $\varepsilon > 0$ is the **resolution parameter**, $\Omega \subseteq \mathbb{R}^k$ is a polyhedral set, and $\alpha, \beta \in C_{space}$. The planner is **resolution-exact** (or **ε-exact**) if it has a constant $K > 1$ such that on any input, it outputs an Ω-avoiding path from α to β if there exist paths with clearance $K\varepsilon$; and it outputs "No Path" if there are no paths with clearance ε/K. The role of K is critical: it causes indeterminacy, but is also the key to avoiding exact computation.

There are two ingredients in resolution-based methods: first is the subdivision of C_{space}. The subdivision may be organized as a **subdivision tree**, often called quadtrees. Tree nodes correspond to subsets $B \subseteq C_{space}$ with simple shapes such as boxes or simplices. Although grid search is often identified with resolution complete algorithms, we stress that grid methods are usually a form of sampling and inadequate for ε-exactness. The second ingredient is a **classification predicate** to decide if a node B is free or not. The obstacle set Ω defines the free space, $C_{free} = C_{free}(R_0, \Omega) \subseteq C_{space}$. Wlog, C_{free} is an open set; its boundary ∂C_{free} comprises the **semi-free** configurations. Say $\alpha \in C_{space}$ is **stuck** if it is neither free nor semi-free. The **exact classification predicate** is $C : B \mapsto C(B) \in \{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$ such that

$$C(B) = \begin{cases} \texttt{FREE} & \text{if } \ B \subseteq \overline{C_{free}}, \\ \texttt{STUCK} & \text{if } \ B \cap \overline{C_{free}} \text{ is empty}, \\ \texttt{MIXED} & \text{else}. \end{cases}$$

It is easy to construct exact planners using C. Moreover $C(B)$ *could* be computed exactly for nice B (e.g., B is a box). But our thrust is to avoid the high cost of exactness. The basic idea (ansatz) is: *in the presence of subdivision, exact predicates can be replaced by suitable approximations*. More precisely, a predicate $\widetilde{C} : B \mapsto \widetilde{C}(B) \in \{\texttt{FREE}, \texttt{STUCK}, \texttt{MIXED}\}$ is a **soft version of** C if it is **conservative** (i.e., $\widetilde{C}(B) \neq \texttt{MIXED}$ implies $\widetilde{C}(B) = C(B)$), and **convergent** (i.e., $B_i \to p \in C_{space}$ as $i \to \infty$ implies $\widetilde{C}(B_i) \to C(p)$). For analysis, we may also need some measure of "effectivity" or convergence rate.

What we know: soft predicates are relatively easy to design and to implement. Indeed, the computation of $\widetilde{C}$ can be correlated to the expansion of the subdivision tree $\mathcal{T}$. LaValle insightfully call this aspect of our work as "opening up the blackbox of collision testing". Soft predicates have nice properties like **composability**: if a polyhedral robot $R_0 \subseteq \mathbb{R}^k$ has a cover $R_0 = \cup_{i=1}^m T_i$ then we obtain a soft predicate for R_i from soft predicates for the T_i's. Thus predicates for complex robots like R_0 is reducible to simpler robots T_i. To ensure efficiency and adaptivity, the technique of filters will prove essential (a well-known phenomenon in numerical methods). We refer to related work (not just in motion planning) exemplifying these remarks (e.g., [14, 17, 18, 15]).

¶5. Critical Discussion. It is hard to claim novelty in an old idea like resolution-based methods, although we claim new theoretical foundations. It is harder to improve upon a 20-year old paradigm like PRM, where remarkable advances have been made over the years. Nevertheless, we claim a place for SSS planners in practical robotics under PRM's shadow.

First, we address a conceptual objection. Some critiques view the "No Path" outcome in SSS planners as equivalent to the "PRM time-out" but in resolution space. Hence it is no less "arbitrary". This analogy can mislead in two ways. First, the said inherent inaccuracies in sensors, actuators and physical constants mean that paths with clearance below some (calculable) resolution are as good as "No Path". So the "No Path" outcome in SSS planners may be principled, not arbitrary. Second, "time-out" in resolution search is only the most obvious way terminate (we use it below), but it is not the only way. In fact, it is a deeply interesting question to develop techniques for fast detection of "No Path" (cf. [22, 7]).

Consider our suggestion that SSS is practical. The subdivision infrastructure is well-understood and based on efficient data structures like union-find. The soft predicates we design [17] can mostly reduce to estimating distances between two obstacle features (i.e., point, line or plane). This almost seems trivial compared to exact algorithms; so SSS planners are clearly *implementable*. But will these implementations be *practically efficient*? Here, we invoke the evidence of prior resolution-based work such as Zhu and Latombe [23], Barhehenn and Hutchinson [1], and Zhang, Kim and Manocha [22]. Of course, we must reinterpret them using our new perspective; it is illuminating to revisit these papers with hindsight.

The preceding paragraph is not new except for our SSS perspective. The deeper debate among roboticists is about the ability of resolution methods to scale up in dimension. The consensus is that resolution methods can only reach medium degrees-of-freedom (DOF), while PRM reaches much higher[2] degrees. Choset [6, p. 202] suggests that state-of-art PRM can handle DOF in the range $5 - 12$. They noted that a 10 DOF planar robot from Kavraki (1995) cannot be tackled by other methods. However, we find no conceptual barriers for SSS to match PRM. Randomness is not pertinent since SSS can also expand randomly. A naive approach to resolution methods yields tree sizes that are exponential in the depth; but related subdivision work in root isolation [15] shows we can achieve tree size that is worst-case polynomial in the depth. The performance of SSS planners are highly dependent on the tree expansion strategy; this is no different from PRM. Indeed, the tree structure of subdivision seems to give SSS a great advantage. Currently, we have limited (but encouraging) experiments to support our intuition, but we feel the field is wide open for experimentation.

III. Two Frameworks for Motion Planning.

We intend to view PRM as an **algorithmic framework** for a large class of sampling-based planners. An algorithm within the framework is[3] just a specific instantiation, using particular data structures and subroutines.

¶6. The PRM Framework. There are many possible formulations, but we follow LaValle [13, Section 5.4.1]: to find a path connecting $\alpha, \beta \in C_{free}$, we maintain a graph $G = (V, E)$ where $\{\alpha, \beta\} \subseteq V \subseteq C_{free}$ and edges in E correspond to paths. We need three predicates: $Free(u)$ to test if configuration u is free; $Connect(v, u)$ to test if the (straight) motion from v to u is free; a **termination criterion** that returns "success" (a path is found) or "failure" (time-out or other condition).

```
PRM FRAMEWORK:
While (termination criterion fails):
    1.  Vertex Selection Method (VSM):
            Choose a vertex $v$ in $V$ for expansion.
    2.  Configuration Generation Method (CGM):
            Generate some $u \in C_{space}$ (perhaps near $v$)
    3.  Local Planning Method (LPM):
            If $Free(u)$,
                Add $u$ to $V$
                If $Connect(v, u)$, add $(v, u)$ to $E$.
Return success or failure accordingly.
```

Choset et al. [6, p.198] noted that PRM is practical because the predicate $Free(u)$ is relatively cheap. The large literature on collision detection is about this predicate. We offer another reason for the great success of PRM: *the framework allows one to easily modify the constituent components (VSM, CGM, LPM) to obtain a variety of algorithms for diverse needs. The basic infrastructure is relatively stable, thanks to the simplicity and generality of PRM.* We want to emulate this in SSS.

¶7. The SSS Framework. In addition to the usual input $(\varepsilon, \Omega, \alpha, \beta)$, an initial box $B_0 \subseteq C_{space}$ is given: we are interested in paths restricted to B_0. The subdivision tree $\mathcal{T}$ is rooted at B_0, and each node $B \subseteq B_0$ is classified by a soft predicate $\widetilde{C}$. We grow $\mathcal{T}$ by expanding successive MIXED-leaves until we find a path or conclude "No Path".

```
SSS FRAMEWORK
1.  While ($\widetilde{C}(Box(\alpha)) \neq$ FREE)          ◁ Initialization
        If $Box(\alpha)$ has length $< \varepsilon$, Return ("No Path")
        Else Expand($Box(\alpha)$)
    While ($\widetilde{C}(Box(\beta)) \neq$ FREE)
        ... do the same for $\beta$ ...
2.  While ($Find(Box(\alpha)) \neq Find(Box(\beta))$)       ◁ Main Loop
        If $Q$ is empty, Return("No Path")
        $B \leftarrow Q$.GetNext()
        Expand($B$)
3.  Compute a FREE channel $P$ from $Box(\alpha)$ to $Box(\beta)$
        Generate and return the "canonical path" $\overline{P}$ inside $P$.
```

A priority queue $Q = Q_{\mathcal{T}}$ holds all MIXED-leaves of radius $r(B) \geq \varepsilon$. The routine Q.**GetNext**() returns a leaf of highest priority which is split by **Expand**(B). The FREE boxes are stored in a union-find structure for the connected components: two boxes B, B' are directly connected if $\dim(B \cap B') = d - 1$. If "$Box(\alpha)$" is a leaf of $\mathcal{T}$ containing α, our algorithm halts as soon as $Box(\alpha)$ and $Box(\beta)$ are in the same component.

The performance of SSS is naturally adaptive but highly dependent on GetNext() and Expand(). For $d \geq 4$ degrees of

[2] We do not consider very highly flexible robots such as molecules, snakes, humanoids, etc., where the DOF can go much higher than discussed here. What counts as "success" is much less formal in such settings.

[3] To be sure, there are degrees of specificity. The most specific instantiation might be called "implementation" of some less specific "algorithm".

freedom, careful expansion is critical; do not always expect to split into 2^d children. The methods in [23, 1, 22] fall under our framework.

¶8. Similarities and Differences. There are many similarities between PRM and SSS, especially in their contrasts to exact algorithms. Both have two key subroutines: (i) search strategies (VSM in PRM, `GetNext`() in SSS), and (ii) freeness testing ($Free(u)$ in PRM, $\widetilde{C}(B)$ in SSS). Both SSS and PRM have the ability to find paths *before* exploring the entire C_{free}. Thus, Hsu et al. [10, p. 640] calls this a "foundational choice in PRM planning". In contrast, exact methods require expensive (non-adaptive) pre-processing to compute a description of C_{free}. Both frameworks naturally compute a path, i.e., a parametrized curve in C_{free}; exact methods require a separate subalgorithm for this.

The key difference is that SSS planners have no Halting Problem. In SSS we use a more demanding predicate $\widetilde{C}(B)$ than $Free(u)$. The benefit is that SSS can compute free channels just by checking adjacency of two `FREE` boxes; PRM needs an extra predicate $Connect(v, u)$.

IV. Conclusion

In this paper, we described the SSS framework for ε-exact planners. Ideas of resolution-limited algorithms are well-known, but to our knowledge, the simple[4] properties of soft classifiers have never been isolated. These "simple ideas" promise to create new algorithms that are practical *and* theoretically sound, not only in motion planning. There are many open questions concerning SSS. Although there are interesting theoretical questions, we feel the immediate challenge is to prove the practical power of SSS. Following up on [17], we plan to do this by designing and implementing a variety of soft predicates and search strategies, from 6DOF robots, medium DOF flexible robots (e.g., Kavraki robot), and complex robots. Like PRM, we expect many variants of SSS to arise.

References

[1] M. Barbehenn and S. Hutchinson. Toward an exact incremental geometric robot motion planner. In *Proc. Intell. Robots and Systems 95*, vol. 3, pp. 39–44, 1995.

[2] R. Bohlin and L.E. Kavraki. A randomized algorithm for robot path planning based on lazy evaluation. *Handbook on Randomized Comput.*, pp. 221–249. Kluwer, 2001.

[3] R. A. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *8th IJCAI - Vol. 2*, pp. 799–806, San Francisco, 1983.

[4] John Francis Canny. *The complexity of robot motion planning*. The MIT Press, 1988. PhD thesis, M.I.T.

[5] E.-C. Chang, S. W. Choi, D. Kwon, H. Park, and C. Yap. Shortest paths for disc obstacles is computable. *IJCGA*, 16(5-6):567–590, 2006. Special Issue.

[6] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[7] J. Denny and N. M. Amato. Toggle PRM: A coordinated mapping of C-free and C-obstacle in arbitrary dimension. In *Proc. WAFR*. MIT, Cambridge, USA. June 2012.

[8] M. Safey el Din and E. Schost. A baby steps/giant steps probabilistic algorithm for computing roadmaps in smooth bounded real hypersurface. *Discrete and Comp. Geom.*, 45(1):181–220, 2011.

[9] D. Halperin, L. Kavraki, and J.-C. Latombe. Robotics. In J. E. Goodman and J. O'Rourke, eds., *Handbook of Discrete and Comp. Geom.*, chap. 41. CRC Press, 1997.

[10] D. Hsu, J.-C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *IJRR*, 25(7):627–643, 2006.

[11] L. Kavraki, P. Švestka, C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation*, 12(4):566–580, 1996.

[12] S. LaValle, M. Branicky, and S. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *IJRR*, 23(7/8):673–692, 2004.

[13] Steven M. LaValle. *Planning Algorithms*. Cambridge U. Press, 2006.

[14] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In *Proc. SGP*, pp. 245–254, New York, 2004. ACM Press.

[15] M. Sagraloff and C.K. Yap. A simple but exact and efficient algorithm for complex root isolation. *36th ISSAC*, pp. 353–360, 2011. June 8-11, San Jose, California.

[16] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Appl. Math.*, 4:298–351, 1983.

[17] C. Wang, Y.-J. Chiang, and C. Yap. On Soft Predicates in Subdivision Motion Planning. In *29th Symp. on Comp. Geom.*. To Appear. Rio de Janeiro. Jun 17-20, 2013.

[18] Chee Yap, Vikram Sharma, and Jyh-Ming Lien. Towards Exact Numerical Voronoi diagrams. In *9th ISVD*, pp. 2–16. IEEE, 2012. Invited Talk.

[19] Chee K. Yap. Algorithmic motion planning. In *Advances in Robotics, Vol. 1: Algorithmic and geometric issues*, vol. 1, pp. 95–143. Lawrence Erlbaum Associates, 1987.

[20] Chee K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, eds., *Handbook of Discrete and Computational Geometry*, chap. 41, pp. 927–952. Chapman & Hall/CRC, Boca Raton, FL, 2nd ed., 2004.

[21] Chee K. Yap. Theory of Soft Subdivision Search and Motion Planning, 2012. Manuscript. URL `http://cs.nyu.edu/exact/`.

[22] L. Zhang, Y. J. Kim, and D. Manocha. Efficient cell labelling and path non-existence computation using C-obstacle query. *IJRR*, 27(11–12), 2008.

[23] D.J. Zhu and J.-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7:9–20, 1991.

[4] Some reviewers of our work see only the safeness part of soft classifiers. They fail to note that previous work are silent about convergence or effectivity.