2022 Sixth IEEE International Conference on Robotic Computing (IRC) | 978-1-6654-7260-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IRC55401.2022.00078

Path Smoothing with Deterministic Shortcuts

Maryam Khazaei Pool[1] Carlos Diaz Alvarenga[1] Marcelo Kallmann[1,2]

***Abstract*— Path smoothing is an important operation in a number of path planning applications. While several approaches have been proposed in the literature, a lack of simple and effective methods with quality-based termination conditions can be observed. In this paper we propose a deterministic shortcut-based smoothing method that is simple to be implemented and achieves user-specified termination conditions based on solution quality, overcoming one of the main limitations observed in traditional random-based approaches. We present several benchmarks demonstrating that our method produces higher-quality results when compared to the traditional random shortcuts approach.**

I. Introduction

Path smoothing is an important operation that appears in a number of path planning applications. Path smoothing is often used to smooth the result of a sampling-based planner, or to deform a path in order to achieve desired qualities, such as maintaining a desired distance from obstacles or controlling a given quality aspect. In applications relying on multiple paths, such as in multi-agent path finding problems, relying on a simple and efficient smoothing method with controlled quality becomes particularly important given that the quality of one smoothed path may influence the space available for smoothing the other paths.

In this paper, we propose a Deterministic Shortcut-based Smoothing (DSS) method which overcomes the main limitations of previous shortcut-based methods by being able to consider user-specified termination conditions based on solution quality. At each iteration, our method first identifies a vertex on the path that has the most potential for path improvement, and then applies one of two possible shortcut-based smoothing operations.

As a result, our prioritized shortcut selection and quality-based termination conditions result in a method that outperforms a traditional implementation of the random shortcut approach, both in terms of path length and in worst-case angle measured along the path. In this work, similarly to previous work on this area, we consider a path to be represented as a polygonal line. We present several benchmarks demonstrating that, for the same amount of smoothing time, our method produces higher-quality paths when compared to the traditional random shortcut approach [4], [6], [7], [10].

II. Related Work

Path smoothing methods are tightly related to path planning. For instance, sampling-based methods, such as the Rapidly-Exploring Random Tree (RRT), have influenced the need for efficient path smoothing algorithms. These methods have become popular over the last decade, in part given their ability to address problems in high-dimensional spaces [1], [13]. Sampling-based algorithms highly depend on collision checking in order to determine the feasibility of each search expansion, and the sampling nature of the strategy often produces paths with excessive bends and turns, leading to the need of a post-processing smoothing step.

Path smoothing and optimization are topics which have been studied by many researchers. While some methods, like ours, are based on iterative shortcuts, a wide range of different techniques have been proposed. A summary of the state of the art has been proposed by Abhijeet Ravankar et al. [11], which also highlights the importance of considering dynamic and kinematic constraints in path planning and optimization. From a broad point of view, we classify the main methods as generic optimization-based techniques and shortcut-based iterative methods.

Optimization Techniques Optimization techniques are often employed to optimize trajectories. These methods require appropriate objective functions, collision checking with obstacles, and constraints on velocity and acceleration in order to achieve smooth trajectories [3].

For instance, Jia Pan et al. presented a path optimization technique based on local spline refinement in order to compute smooth, collision-free paths in narrow passages and satisfy velocity and acceleration constraints [9]. CHOMP optimizes an initial trajectory iteratively using Covariant Hamiltonian gradient descent [14]. Hui Yang et al. presented a gradient-free optimization technique which they call the Double Layer Ant Algorithm [12]. The authors perform Turning Point Optimization and Piecewise B-spline smoothing to improve the input path.

In general the performance of optimization methods highly depend on the specific types of input trajectories. Slow convergence can be observed when the input requires significant improvement. The complexity of implementation may also be seen as a barrier for integration in practical applications. Our method is related to shortcut-based methods which have become popular given their simplicity and effective results.

Methods Based on Iterative Shortcuts Shortcut-based methods provide a simple and effective approach to path optimization [4], [7], [10]. Heuristic methods based on random shortcuts replace jerky portions of a path with shorter segments, the *shortcuts*, after checking that the segments are collision free. Shortcuts are often determined by randomly sampling its endpoints along the path. The implementation of shortcut-based techniques is simple, fast, and produces

[1]Department of Computer Science and Engineering, University of California, Merced, CA, USA. [2]Marcelo Kallmann is now a Principal Scientist at Amazon Robotics - this work was initiated before joining Amazon and is not related to Amazon work.

978-1-6654-7260-9/22/$31.00 ©2022 IEEE
DOI 10.1109/IRC55401.2022.00078

high-quality paths in many cases [4], [8].

A number of specific implementations of the approach have been proposed. For instance, David Hsu et al. describe a technique that removes redundant motions on a path [7], and terminates the procedure if the improvement falls below a threshold. In contrast our work introduces simple metrics for defining a termination condition that is quality-based, such that the produced results will meet the specified quality-based threshold.

The Cutting-triangle's-edge algorithm presented by Reda Guernane et al. [5] produces shortcuts by connecting the midpoints of adjacent path segments in the path. The dynamic and kinematic constraints of the robot are also taken into account to define cubic polynomials which smooth edge discontinuities, however difficult corners may not be possible to be improved. The shortcut-based method proposed by Mylène Campana et al. [2] is a gradient-based technique that uses backtracking when a collision is detected on the most recent iteration of the algorithm. It has advantages over random methods, however, it requires the computation of gradients. In comparison to these methods, our method is gradient-free and focuses on efficiency for 2D applications by developing specific geometric tests for generating shortcuts that can remove large unneeded portions of the input path.

One main point of our method is that it is deterministic. An important drawback of methods based on the random selection of shortcuts is that they may miss tight areas difficult to be sampled. This may lead to sharp corners left unoptimized in the resulting path. Our deterministic way of selecting where to perform the next path improvement addresses this limitation. The approach also leads to the ability to define quality-based termination conditions, which allows us to produce paths that satisfy given limits controlling smoothness and clearance. Our current work focuses on 2D applications and we show improved results when comparing against a regular implementation of the random shortcuts approach.

III. Methodology

In the scope of this work we address the particular case of 2D polygonal paths. We consider that the input polygonal path P is defined by an ordered set $P = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N\}$ containing N vertices. The path connects the starting location $\mathbf{v}_1$ to the goal location $\mathbf{v}_N$. Environments are 2-dimensional and described by a set of polygonal obstacles $O = \{O_1, O_2, ..., O_M\}$.

Our goal is to minimize the length of P and maximize the smallest angle between two adjacent P segments, while maintaining P collision-free and fixed at $\mathbf{v}_1$ and $\mathbf{v}_N$.

Our proposed Deterministic Shortcut-based Smoothing (DSS) method is based on two geometric shortcut determination procedures: the *Disk Test* and the *Corner Test*. These procedures will first test if a shortcut can be performed with respect to a given vertex, and if so, that shortcut is returned and the path is improved; otherwise, a label *done* is returned and the overall algorithm stops.

A. Disk Test

The Disk Test is outlined in algorithm 1. The Disk Test follows a greedy selection process: at every iteration, we optimize the path at the vertex with the most free space around it. First, for every non-terminal vertex $\mathbf{v}_i$, we start by calculating the minimum Euclidean distance to every obstacle in the environment:

$$d_{min}(\mathbf{v}_i) = \min_{O_j \in O} D(\mathbf{v}_i, O_j), \quad \forall i \in \{2, 3, ..., N-1\},$$

where O is the set of obstacles, O_j is an obstacle in the environment and D is a function that returns the minimum distance from the polygonal obstacles to a vertex in the path. This distance serves as a proxy for the free space available around a vertex. We then determine the vertex with the most free space $\mathbf{v}^*$, which is the vertex such that:

$$d_{min}(\mathbf{v}^*) = d^* = \max d_{min}(\mathbf{v}_i), \forall i \in \{2, 3, ..., N-1\}.$$

Distance d^* is then used as the radius of a circle centered at $\mathbf{v}^*$, which we call C. The points of intersection between C and the path form the endpoints of the shortcut with which we will update the path.

Algorithm 1: Disk Test

```
Data: input path P as a set of vertices
for every vertex v_i in P \ (v_0, v_n) do
    d_i ← min_O D(v_i, O)
end
if TerminationConditionsAreMet() then
    return done
r ← max(d_i)
v* ← P[index(max(d_i))]
s = (p_1, p_2) ← PointsOfCircleIntersection(r, v*)
return s
```

When there are fewer than two points of intersection between the circle and the path, then the circle must encompass one or both of the endpoints of the path. Therefore, in such cases, we choose those encompassed path endpoints as the endpoints of the shortcut. Additionally, if there are more than two intersections, then we choose the earliest and latest among those intersections, with respect to each path direction, as the shortcut endpoints.

The overall Disk Test procedure is summarized in Algorithm 1.

B. Corner Test

The Corner Test has similarities with the Disk Test; however, the key difference is that it only considers a subset of the obstacles. For every vertex, excluding the start and end vertices, a *corner* is defined as the triplet consisting of the previous vertex, the current vertex, and the next vertex. The Corner Test computes the same distance to the obstacles as with the Disk Test, except that now only obstacles that are inside of the *corner region* defined by the corner are considered, as defined below.

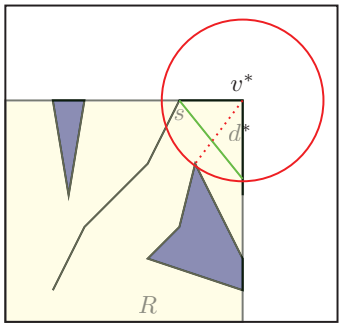

Fig. 1: *An example of the Corner test: an environment including two obstacles, v^* the vertex with the most free space (largest radius d_i called d^*), the convex region R in yellow, and shortcut s in green.*

Given a corner $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$, we define the corner region as the region in-between the rays $(\mathbf{v}_1, \mathbf{v}_0)$ and $(\mathbf{v}_1, \mathbf{v}_2)$. While the Corner Test searches for obstacles at any distance from $\mathbf{v}^*$, it limits our choice of shortcut endpoints to be inside the sub-path delimited by rays $(\mathbf{v}_1, \mathbf{v}_0)$ and $(\mathbf{v}_1, \mathbf{v}_2)$.

When the number of intersections between the circle and the path is not exactly 2, we choose the shortcut endpoints in the same way as described for the Disk Test. However, when applying the Corner Test, we limit the shortcut endpoints to not exceed the previous and next vertices of $\mathbf{v}^*$. The pseudo-code for the procedure is given in Algorithm 2.

In Figure 1, the region R, shown here in yellow, is the only area where obstacles are considered. A shortcut s is then built from the intersection of C and path P. With the Corner Test the generated shortcut is not allowed to go outside the corner region.

C. Termination Condition

We define a termination condition based on the quality of the solution, according to two criteria. When one of the two quality criteria are met for every vertex, the smoothing iterations stop and the algorithm terminates.

The first criterion determines, for a given vertex v, if v is within the threshold distance to any obstacle in the environment. This means that v is already at the limit distance to the obstacles and cannot be further optimized.

The second criterion is a measure of smoothness around v that is simply based on the angle between the two path edges sharing v. If the angle is larger than a desired threshold, then this criterion is met for v.

When these criteria are met for a given vertex it means that the vertex is not suitable for optimization, and will not be chosen as $\mathbf{v}^*$. When every vertex satisfies at least one of these criteria, then the optimization terminates.

D. DSS Method

At each iteration of our proposed DSS method one of the two shortcut determination tests presented in the previous subsections is employed. Figures 2 and 3 illustrate cases in

Algorithm 2: Corner Test

Data: input path P as a set of vertices
for *every vertex* v_i *in* $P \setminus (v_0, v_n)$ **do**
  $Y \leftarrow$ subset of O that is in corner region
  $d_i \leftarrow \min_Y D(v_i, Y)$
end
if *TerminationConditionsAreMet()* **then**
  return done
$r \leftarrow \max(d_i)$
$v^* \leftarrow P[index(\max(d_i))]$
$s = (\mathbf{p}_1, \mathbf{p}_2) \leftarrow$ PointsOfCircleIntersection(r, v^*)
return s

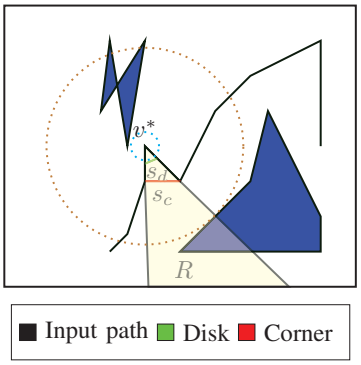

Fig. 2: *Comparative example between shortcuts $\mathbf{s_c}$ (in red) and $\mathbf{s_d}$ (in green) obtained, respectively, with the Corner Test and the Disk Test. Here the Corner Test provides the longest shortcut because it only considers obstacles inside the corner region R.*

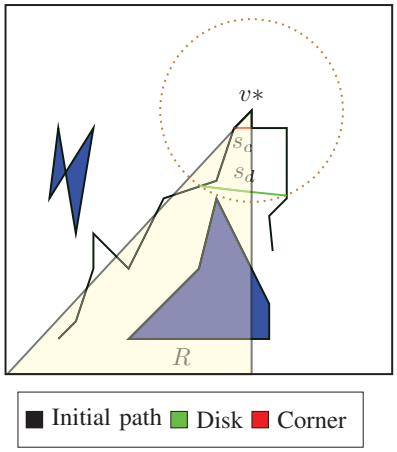

Fig. 3: *Comparative example between shortcuts $\mathbf{s_c}$ (in red) and $\mathbf{s_d}$ (in green) obtained, respectively, with the Corner Test and the Disk Test. The corner region R is shown in yellow. Here the Disk Test provides the longest shortcut.*

which either the Corner or Disk Test can be most advantageous.

We first consider the Corner Test because we have found it to be, most of the time, the better operation to perform. Considering Figure 2, it is possible to see that when the obstacles in the corner region are far away, the shortcuts tend to be longer. In this case, the Corner Test makes a more useful shortcut than the Disk Test, since there might be obstacles not inside corner region which are closer to $\mathbf{v}^*$. On the other hand, if the distance from the obstacles to $\mathbf{v}^*$ is the same on both sides of the path, or the closest obstacle is in the corner region, then it is more advantageous to employ the Disk Test.

Therefore we first check if the Corner Test provides an effective shortcut, and if not, we then compute the result of the Disk Test and compare the two obtained shortcuts in order to select the best one. The pseudocode for DSS is given in Algorithm 3 (we note here that the termination conditions are handled in the Corner Test and Disk Test functions). Finally, note that the order in which the vertices are processed may affect the determination of $\mathbf{v}^*$.

Using a single test (Corner or Disk) does not always perform well in terms of length, sharpest angle, and average angle, which are our metrics of interest. Therefore, we define parameters δ and k to specify how DSS should decide which method is better at the current iteration. Parameter k represents a factor applied to the radius of the circle used to derive the Corner Test's shortcut. When the length of the shortcut is much smaller than the radius of the circle, DSS chooses to also consider the shortcut provided by the Disk Test. Since no chord of a circle can be larger than the diameter, it does not make sense to choose $k > 2$. Parameter δ provides a similar discrimination, but according to the absolute distance of the shortcut from the Corner test, rather than its ratio to the radius. Regardless of the used parameter values, if the Disk Test is considered, then its shortcut is compared with the shortcut obtained from the Corner Test, and the shortcut with greater length is ultimately used.

Algorithm 3: DSS (δ, k)

```
Data: δ, k: Selection parameters, O: Obstacles, P:
      Current path
InitializeEnvironment(P, O)
s1 = CornerTest()
if s1.length < δ + s1.r · k then
    s2 = DiskTest()
    if s2.length > s1.length then
        UpdatePath(P, s2)
    else
        UpdatePath(P, s1)
    end
else
    UpdatePath(P, s1)
end
```

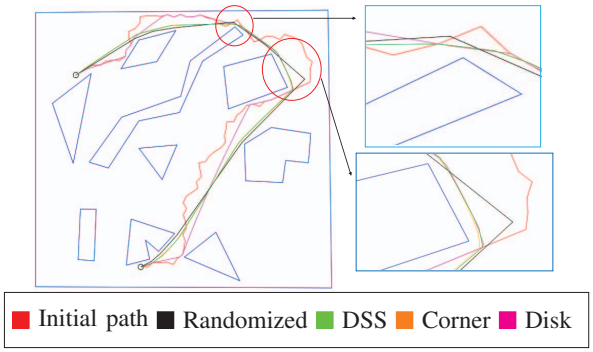

Fig. 4: *Results produced by DSS, DSS with only the Corner Test, DSS with only the Disk Test and Random Shortcuts in an environment with obstacles of diverse shapes.*

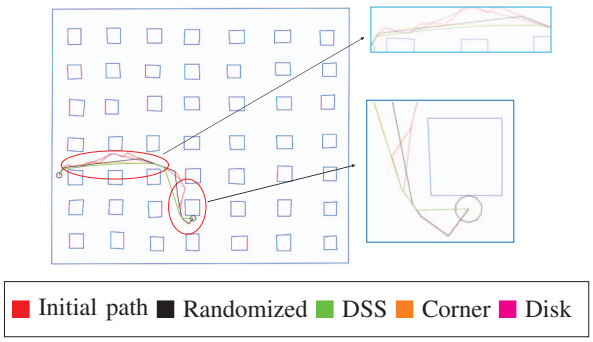

Fig. 5: *Results produced by DSS, DSS with only the Corner Test, DSS with only the Disk Test and Random Shortcuts in an environment with regularly spaced obstacles.*

IV. Evaluation and Results

We validated our DSS method and compared it against a regular implementation of the Random Shortcuts method in five different environments which contained a variety of convex and non-convex obstacles of different sizes and placements. Both methods only address static obstacles.

Our implementation of the Random Shortcuts is based on: 1) sampling random pairs of points along the current path, 2) checking if the shortcut connecting a pair of points is collision-free and respecting the given minimum clearance, and 3) if that is the case, the respective path section is replaced with the sampled shortcut. This implementation reflects how the approach is mostly used, or cited, in previous work [6].

We performed 50 trials for each environment. For every environment, a trial consisted of: (1) randomly choosing start and goal locations, (2) optimizing a path result computed with an RRT implementation, and (3) optimizing the path using the methods being compared.

The three metrics we used to compare the algorithms, which are common in the literature [2], [10], are *average angle*, *sharpest angle* and *average length*. The first two

Metric	Method	Environment				
		Mixed	Interlocked	Regular	Simple	Office
Avg Angle	Random	152.34	143.99	149.46	141.93	143.31
	Corner	174.18	175.39	173.52	166.63	173.15
	Disk	169.29	169.41	173.52	**172.74**	**173.93**
	DSS	**174.68**	**175.41**	**173.67**	166.54	172.95
Sharpest Angle	Random	88.60	80.10	81.25	83.9	83.39
	Corner	144.11	147.86	138.63	131.20	**144.95**
	Disk	127.74	132.28	123.87	**137.49**	143.14
	DSS	**145.23**	**147.93**	**139.99**	130.15	144.88
Avg Length	Random	16.34	17.47	15.10	**18.05**	**15.26**
	Corner	16.43	**17.44**	15.05	19.35	15.59
	Disk	16.97	18.16	15.47	18.34	15.57
	DSS	**16.27**	17.52	**14.98**	19.42	15.57

TABLE I: Table showing the path properties after optimizing with different methods in 5 test environments.

metrics capture how smooth the final solution is, while the last metric captures the cost of traversal for the final solution. If the average angle is larger, the path is considered smoother. These metrics are useful to determine a path that is easier and faster to traverse for robots with typical dynamic constraints [11].

For comparison, we implemented the Random Shortcut method and ran it for the same amount of time as the DSS method.

Each iteration of our method must find the distance between each vertex and the nearest obstacle which we compute by iterating over every pair. Therefore, each iteration of our method takes $O(NL)$ running time: where N is the total number of vertices in a path and L is the total number of edges of all the obstacles.

Figure 4 represents a visualization of DSS versus Random Shortcuts in one of our test environments. We circled and zoomed the regions of interest. Figure 4 shows that DSS produces a final path in green which is smoother and shorter than the final path obtained by regular Random shortcuts. Due to the sharpest angle in the final path produced by Random Shortcuts (method shown in black), we can say DSS is smoother.

Figure 5 shows a similar improvement as in figure 4 when using the DSS method.

Table I shows the performance for $\delta = 2.0$ and $k = 0.0$. In almost all scenarios, DSS performs better in terms of average angles and sharpest angles, as well as average length. DSS with only the Disk Test performs better than DSS with only Corner Test in the Simple environment, as expected according to the discussion presented in Section III.

While our current method proves to be more effective than the regular random selection of shortcuts, a number of additional combinations of the proposed deterministic tests and selection parameters can be explored which we however leave for future work.

V. Conclusion

We show that the proposed priority-based deterministic shortcut selection method, for the same amount of computation time, produces comparable and in many cases better results than the regular random selection of shortcuts in terms of path length and smoothness.

In general, the corner test is more advantageous since it can safely ignore some of the obstacles in the environment, however the disk test is more effective in particular cases. A promising future work is to include a characterization of the complexity of the environment in relation to the performance of either the corner test or the disk test.

More generally this paper shows that simple geometric tests can improve the performance of shortcut-based path smoothing techniques, motivating further developments in this area. We intend to further improve our path optimization method by employing Bézier curves as path segments in order to produce C^2 continuity and the opportunity to address curvature constraints to the resulting optimized paths.

References

[1] Sheelan Waad Adwaan et al. Proposed modified directed RRT algorithm of the basic RRT. *Al-Mansour Journal*, (33), 2020.

[2] Mylène Campana, Florent Lamiraux, and Jean-Paul Laumond. A simple path optimization method for motion planning. working paper or preprint, September 2015.

[3] Younsung Choi, Donghyung Kim, Soonwoong Hwang, Hyeonguk Kim, Namwun Kim, and Changsoo Han. Dual-arm robot motion planning for collision avoidance using B-spline curve. *International Journal of Precision Engineering and Manufacturing*, 18(6):835–843, 2017.

[4] Roland Geraerts and Mark H Overmars. Creating high-quality paths for motion planning. *The international journal of robotics research*, 26(8):845–863, 2007.

[5] Reda Guernane and Mahmoud Belhocine. A smoothing strategy for PRM paths application to six-axes MOTOMAN SV3X manipulator. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4155–4160. IEEE, 2005.

[6] Kris Hauser and Victor Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *2010 IEEE International Conference on Robotics and Automation*, pages 2493–2498. IEEE, 2010.

[7] David Hsu, J-C Latcombe, and Stephen Sorkin. Placing a robot manipulator amid obstacles for optimized execution. In *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99)(Cat. no. 99TH8470)*, pages 280–285. IEEE, 1999.

[8] Marcelo Kallmann, Amaury Aubel, Tolga Abaci, and Daniel Thalmann. Planning collision-free reaching motions for interactive object manipulation and grasping. In *ACM SIGGRAPH 2008 classes*, pages 1–11. 2008.

[9] Jia Pan, Liangjun Zhang, Dinesh Manocha, and UC Hill. Collision-free and curvature-continuous path smoothing in cluttered environments. *Robotics: Science and Systems VII*, 17:233, 2012.

[10] Joseph Polden, Zengxi Pan, Nathan Larkin, and Stephen van Duin. Adaptive partial shortcuts: Path optimization for industrial robotics. *Journal of Intelligent & Robotic Systems*, 86(1):35–47, 2017.

[11] Abhijeet Ravankar, Ankit A Ravankar, Yukinori Kobayashi, Yohei Hoshino, and Chao-Chung Peng. Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. *Sensors*, 18(9):3170, 2018.

[12] Hui Yang, Jie Qi, Yongchun Miao, Haixin Sun, and Jianghui Li. A new robot navigation algorithm based on a Double-Layer Ant algorithm and trajectory optimization. *IEEE Transactions on Industrial Electronics*, 66(11):8557–8566, 2018.

[13] Zhen Zhang, Defeng Wu, Jiadong Gu, and Fusheng Li. A path-planning strategy for unmanned surface vehicles based on an adaptive hybrid dynamic stepsize and target attractive Force-RRT algorithm. *Journal of Marine Science and Engineering*, 7(5):132, 2019.

[14] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP: Covariant Hamiltonian Optimization for Motion Planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.