Appendix A
Proofs

Here, we provide the proof of each mathematical claim in the paper, plus a short explanation of how each claim is useful.

First, we examine the structure of the JRS zonotope representation. This structure enables the creation of *fully-k-sliceable* generators when we use the JRS to produce rotatotopes. That is, this lemma enables us to slice the arm's RS to find subsets corresponding to particular trajectory parameters.

Lemma 2. *There exist $J_i : \mathbb{N}_T \to \mathcal{P}(\mathbb{R}^2 \times K)$ that overapproximate $\mathscr{J}_i$ as in (9) such that, for each $t \in T$, $J_i(t)$ has only one generator with a nonzero element, equal to Δk_i^{v}, in the dimension corresponding to k_i^{v}, and only one (distinct) generator with a nonzero element, Δk_i^{a}, for k_i^{a}.*

Proof: Given $J_i(0)$, the subsequent zonotope $J_i(\Delta t)$ is computed as $J_i(\Delta t) = e^{F\Delta T} J_i(0) + E$, where F is found by linearizing the dynamics (5) at $t = 0$ and E is a set that overapproximates the linearization error and the states reached over the interval $[0, \Delta t]$ [26, Section 3.4.1]. This linearized forward-integration and error-bounding procedure is applied to $J_i(\Delta t)$ to produce $J_i(2\Delta t)$, and so on, to compute all $J_i(t)$ in (9). Since $\dot{k} = 0$, we have that $e^{F\Delta} \tilde{g}_i^{\mathrm{v}}$ equals $\tilde{g}_i^{\mathrm{v}}$ in the k dimensions (and therefore each $g_i^{\mathrm{v}}(t)$ does as well, and similarly for $g_i^{\mathrm{a}}(t)$). Since the zero dynamics have no linearization error, one can define E to have zero volume in the k dimensions [26, Proposition 3.7], meaning no generator of any $J_i(t)$ has a nonzero element in the k dimensions, except for $g_i^{\mathrm{v}}(t)$ and $g_i^{\mathrm{a}}(t)$ (which are defined with such nonzero elements). ∎

We now note that all rotation matrices are contained in the matrix zonotopes $M_i(t)$, which are produced by slicing and reshaping the JRS zonotopes. This enables us to conservatively approximate the forward occupancy map.

Lemma 3. *For any parameterized trajectory $q : T \to Q$ with $k_i^{\mathrm{v}} = \dot{\tilde{q}}$, every $R_i(q_i(t;k)) \in M_i(t)$.*

Proof: By [26, Thm. 3.3 and Prop. 3.7], all values attained by the sines and cosines of the joint angles are contained in each $J_i(t)$. By Alg. 1 and (11), each $S_i(t)$ only contains the values of sine and cosine of $q(t;k)$ for which $k_i^{\mathrm{v}} = \dot{\tilde{q}}$. Since $M_i(t)$ only reshapes $S_i(t)$, the proof is complete. ∎

The following lemma confirms that the product of multiple matrix zonotopes times a zonotope is still a rotatotope. This is necessary to overapproximate the forward occupancy map, wherein the arm's joint rotation matrices are multiplied together (and, analogously, the matrix zonotopes are multiplied together).

Lemma 5. *A matrix zonotope times a rotatotope is a rotatotope.*

Proof: This follows from the rotatotope definition. ∎

We use the Minkowski sums of zonotopes and rotatotopes to enable stacking, which is how we build an RS of the entire arm from the low-dimensional JRSs. We also use the Minkowski sum to dilate obstacles, which is necessary to check for intersection with our arm's RS per Lem. 8 (which we do not prove here, as it is proven in [29]).

Lemma 6 (Minkowski sum of zonotopes and rotatotopes)**.** *Consider two zonotopes $X = (x, g_X^i, \langle \zeta^i \rangle)^n$ and $Y = (y, g_Y^j, \langle \psi^j \rangle)^m$. Then $X \oplus Y = (x+y, \{g_X^i, g_Y^j\}, \{\langle \zeta^i \rangle, \langle \psi^j \rangle\})_{i=1,j=1}^{i=n,j=m}$, which is a zonotope centered at $x+y$ with all the generators and indeterminates of both X and Y. Similarly, for two rotatotopes, $V = (v, g_V^i, \langle \mu^i \rangle)^n$ and $W = (w, g_W^j, \langle \omega^j \rangle)^m$),*

$$V \oplus W = \left(v + w, \{g_V^i, g_W^j\}, \{\langle \mu^i \rangle, \langle \omega^j \rangle\}\right)_{i=1,j=1}^{i=n,j=m}. \tag{15}$$

Proof: This follows from the zonotope definition (7) and rotatotope definition (Def. 4). ∎

The following lemma confirms that the sliced and stacked rotatotopes $V_i(t)$ overapproximate the forward occupancy map FO for each i^{th} link.

Lemma 7. *For any $t \in T$ and $k \in K$, $\mathrm{FO}_i(q(t;k)) \subseteq V_i(t)$, where*

$$V_i(t) = \bigoplus_{j<i} \left(\prod_{n \le j} M_n(t) \{l_j\} \right) \oplus \left(\prod_{n \le i} M_n(t) L_i \right) \subset W. \tag{16}$$

Proof: First, note (16) is defined analogously to (1). We have $R_i(q_i(t;k)) \in M_i(t)$ from Lem. 3. The product of matrix zonotopes multiplied by a zonotope is a rotatope by Def. 4, and the Minkowski sum of rotatotopes are given exactly using Lem. 6. Therefore, all sets and operations in (16) are exact or conservative (note, we can overapproximate L_i with a zonotope), so $\mathrm{FO}_i(q(t;k)) \subseteq V_i(t)$. ∎

We use this next lemma to overapproximate the swept volume of the arm (represented with rotatotopes); the overapproximation means that ARMTD is provably conservative, which enables safety guarantees.

Lemma 9. *Any rotatotope MZ as in (13) can be overapproximated by a zonotope.*

Proof: Consider the components of the indeterminate coefficients of $MZ = (x, g^r, \langle \gamma^r \rangle)^s$ that can be written as $\langle \beta^i \lambda^j \rangle$. When evaluated, $\beta^i \lambda^j \in [-1, 1]$. Consider a zonotope $\hat{Z} = (x, g^r, \langle \sigma^r \rangle)^s$ with the same center and generators as MZ, but where each product $\langle \beta^i \lambda^j \rangle$ is replaced with a single new symbolic coefficient $\langle \sigma^r \rangle$. If $z \in MZ$, $\exists\ \sigma^r \in [-1, 1]$ such that $z \in \hat{Z}$. ∎

The following lemma confirms that our unsafe set representation (the function h_{obs}) is conservative.

Lemma 10. *If $k^{\mathrm{a}} \in K_{\text{obs}}$, then there exists $i \in \mathbb{N}$, $t \in T$, and $O \in \mathscr{O}$ such that $h_{\text{obs}}(i, t, O, k^{\mathrm{a}}) \ge 0$.*

Proof: This follows from Lems. 7 and 8; h_{obs} is positive when the zonotope produced by slicing $V_i(t)$ intersects O, and $V_i(t)$ provably contains all points in workspace reachable by the arm under the trajectory parameterized by k^{a}. ∎

The following theorem, the main result in this paper, confirms that feasible parameters for the constraints we generate are collision free and obey joint limits. Note, we consider self-intersection below, in Appx. C.

Theorem 11. *Any feasible solution to (24) parameterizes a trajectory that is collision-free and obeys joint limits over the time horizon T.*

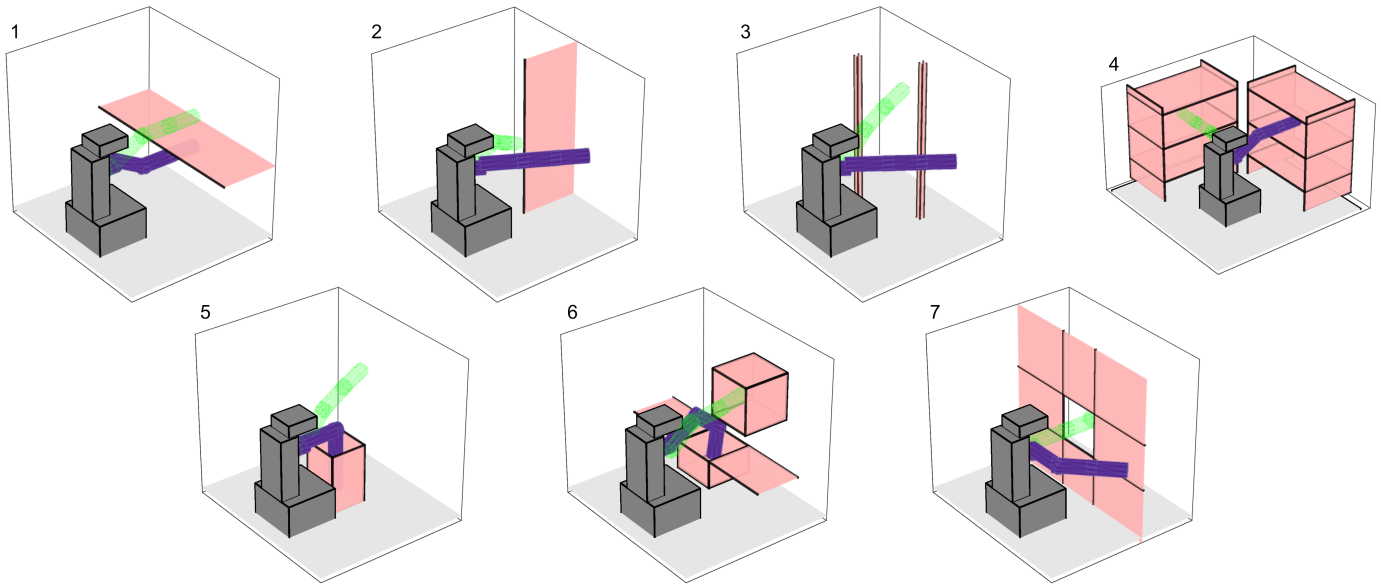

Fig. 4: The set of seven Hard Scenarios (number in the top left), with start pose shown in purple and goal pose shown in green. There are seven tasks in the Hard Scenarios set: (1) from below to above a table, (2) from one side of a wall to another, (3) between two vertical posts, (4) from one set of shelves to another, (5) from inside to outside of a box on the ground, (6) from a sink to a cupboard, (7) through a small window.

Proof: The conservatism of h_{obs} follows from Lem. 9, since each $J_i(t)$ is conservatively transformed into $V_i(t)$; h_{lim} is conservative by construction. ■

Appendix B
Safe Receding-Horizon Planning

ARMTD uses Alg. 3 at each planning iteration. Recall that, without loss of generality, each iteration generates a plan over the time horizon $T = [0, t_{\text{f}}]$ (by shifting the current time to 0). Also recall, at each iteration, we allot t_{plan} s within which to find a new plan. The initial position and velocity of each joint in each iteration is the position and velocity, at time t_{plan}, of the trajectory plan of the previous iteration. ARMTD attempts to find a safe trajectory within t_{plan} by optimizing over a set of safe trajectory parameters; Thm. 11 ensures that any feasible solution is actually collision-free. If no safe trajectory is found within the allotted time, the arm executes the braking maneuver specified by the previous safe trajectory. Assuming the arm does not start in collision, this algorithm ensures that the arm is always safe (see [13, Remark 70] or [11, Theorem 1]).

Appendix C
Self-Intersection Constraints

Typical arms must avoid self-intersection between their links. We specify $I_{\text{self}} \subset \mathbb{N}^2$ as a set of joint index pairs for which the links can intersect. That is, for $(i,j) \in I_{\text{self}}$, there exist $q \in \mathcal{Q}$ such that $\text{FO}_i(q) \cap \text{FO}_j(q) \neq \emptyset$. For example, one may have $I_{\text{self}} = \{(1,3),(1,4),(2,4)\}$ for an arm with 4 links and three possible self-intersections.

We represent self-intersection constraints similarly to how we represent collision-avoidance constraints, with a function $h_{\text{self}} : \mathbb{N}\times\mathbb{N}\times T \times K^{\text{a}} \to \mathbb{R}$. Suppose $(i,j) \in I_{\text{self}} \subset \mathbb{N}^2$ indexes a pair of links that could intersect, whose volume is overapproximated by $V_i(t)$ and $V_j(t)$. In analogy to (19), define

$$V_{\text{self}}(i,j,t) = V_{i,\text{slc}}(t) \oplus (-V_{j,\text{slc}}(t)) \quad \text{and} \tag{25}$$

$$V_{\text{buf}}(i,j,t) = V_{i,\text{buf}}(t) \oplus V_{j,\text{buf}}(t), \tag{26}$$

where $-V_{j,\text{slc}}(t)$ means the center and generators are multiplied by -1. Let $A_{\text{self}}(i,j,t)$ and $b_{\text{self}}(i,j,t)$ return the half-space representation of $V_{\text{buf}}(i,j,t)$. Then, using $*$ in place of the arguments (i,j,t) for space,

$$h_{\text{self}}(*,k^{\text{a}}) = -\max\left(A_{\text{self}}(*)\texttt{eval}(V_{\text{self}}(*),k^{\text{a}}) - b_{\text{self}}(*)\right). \tag{27}$$

As with h_{obs}, h_{self} is a max of a linear combination of polynomials in k^{a}, so we can take the subgradient with respect to k^{a}. Note one can prove a similar result to Lem. 10 for h_{self}.

With these self-intersection constraints, we again implement (23) as a nonlinear program, denoted `optTraj` in Alg. 3.

$$\begin{aligned} \underset{k^{\text{a}}\in K^{\text{a}}}{\text{argmin}} \quad & f(k^{\text{a}}) \\ \text{s.t.} \quad & h_{\text{obs}}(i,t,O,k^{\text{a}}) < 0 \quad \forall\ i \in \{1,\cdots,n_q\},\ t \in T,\ O \in \mathcal{O} \\ & h_{\text{self}}(i,j,t,k^{\text{a}}) < 0 \quad \forall\ (i,j) \in I_{\text{self}},\ t \in T \\ & h_{\text{lim}}(k^{\text{a}}) < 0 \quad \forall\ i \in \{1,\cdots,n_q\}. \end{aligned} \tag{28}$$

Appendix D
Additional Explanations

A. Forward Occupancy Example

For an arm with $n_q > 2$, FO_i as in 1 can be written:

$$\begin{aligned} \text{FO}_i(q) \ = \ & \Big\{ R_1(q_1)l_1 + R_1(q_1)R_2(q_2)l_2 + \cdots \\ & \cdots + \prod_{j\le(i-1)} R_j(q_j)l_{i-1} \Big\} \oplus \left(\prod_{j\le i} R_j(q_j) L_i \right). \end{aligned} \tag{29}$$

Notice that in this example, the rotated link volume of the i^{th} given by $\left(\prod_{j\leq i} R_j(q_j)L_i\right)$ is "stacked" on top of the sum of the positions of all predecessor joints.

B. Slicing

ARMTD uses zonotopes and rotatotopes to represent RSs of parameterized trajectories of an arm. In Sec. IV, our trajectory optimization implementation requires obtaining subsets of the RS corresponding to a particular choice of trajectory parameters. We call this operation *slicing*, because it takes in a zono/rotatotope, evaluates some (or all) of its coefficients, and returns a zono/rotatotope that is a subset of the original with potentially fewer (or no) generators.

We define the `slice` function in Alg. 1 using indeterminate evaluation and removal. This function takes in a zono/rotatotope $Z = (x, g^i, \langle\beta^i\rangle)^p$, a set of indeterminate coefficients $\{\langle\sigma^j\rangle\}_{j=1}^m$, and a set of values for the indeterminate coefficients $\{\sigma^j\}_{j=1}^m$, and outputs a sliced zono/rotatotope. For each generator in Z, if $\langle\sigma^j\rangle$ is a factor of that generator's coefficients (as in (14)), then the generator is multiplied by the value σ^j. If a generator becomes fully-sliced (and therefore has no more indeterminate coefficients), it is added to the center of the output zono/rotatotope, and removed from the set of generators. For zonotopes, each generator becomes fully-sliced if its coefficient is evaluated because each coefficient has only one factor. If a rotatotope is sliced until each generator has only one coefficient factor, the rotatotope becomes a zonotope.

To understand slicing, consider a particular choice of the trajectory parameter k_i^{v}, as in (11). We want to obtain the subset of the JRS representing reachable joint angles corresponding to this particular trajectory parameter. For each $J_i(t)$, only the generator $g_i^{\text{v}}(t)$ is non-zero in the dimension corresponding to this trajectory parameter, meaning this k^{v}-sliceable generator is solely responsible for the volume of the reachable set in this dimension. Choosing a particular value of the trajectory parameter means fixing this generator's indeterminate $\langle\kappa_i^{\text{v}}(t)\rangle$ to a particular value. Since the k^{v}-sliceable generator is (generally) non-zero in the cosine and sine dimensions as well, the slicing operation returns a subset of the JRS in those dimensions (that is, by fixing the value of this indeterminate, we do not lose all of the JRS's volume in the cosine/sine dimensions, whereas the volume in the k-dimensions goes to zero). For example, $\texttt{slice}\left(J_i(t), \left\{\langle\kappa_i^{\text{v}}(t)\rangle\right\}, \left\{\frac{\pi}{6}\right\}\right)$ returns the subset of $J_i(t)$ corresponding to setting $k_i^{\text{v}} = \frac{\pi}{6}$ rad/s (provided that $\frac{\pi}{6} \in K_i^{\text{v}}$).

C. Matrix Zonotope Example

This example constructs $M_i(t)$ from $S_i(t)$ by reshaping the center and generators. Suppose joint i rotates about the 3-axis of link $i-1$. Then:, by [5, (3.39)], we have

$$M_i(t) = R_i(\tilde{q}_i)\left(X_i^{\text{v}}(t), \left\{G_i^{\text{a}}(t), G_i^j(t)\right\}, \left\{\langle\kappa_i^{\text{a}}(t)\rangle, \langle\beta_i^j(t)\rangle\right\}\right)^{p(t)}, \quad (30)$$

$$X_i^{\text{v}}(t) = \begin{bmatrix} c_i^{\text{v}} & -s_i^{\text{v}} & 0 \\ s_i^{\text{v}} & c_i^{\text{v}} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; G_i^{\text{a}}(t) = \begin{bmatrix} c_i^{\text{a}} & -s_i^{\text{a}} & 0 \\ s_i^{\text{a}} & c_i^{\text{a}} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad (31)$$

$$G_i^j(t) = \begin{bmatrix} c_i^j & -s_i^j & 0 \\ s_i^j & c_i^j & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (32)$$

Since each $S_i(t)$ is computed assuming $q_i(0;k) = 0$, we include $R_i(\tilde{q}_i)$ when constructing $M_i(t)$ to correct for each initial joint angle. For different joint axes, $M_i(t)$ can be constructed accordingly [5, Chapter 3.2.3].

D. Reduction of Generators

Creating the rotatotopes $V_i(t)$ in (16) requires multiplying generators together and storing their product. For example, a matrix zonotope described by 10 matrices (a center and 9 generators) multiplied by a zonotope described 2 vectors (a center and 1 generator) yields a rotatotope described by 20 vectors (a center and 19 generators). Because this process is repeated for each joint, the number of generators theoretically required to represent each rotatotope grows exponentially with the number of joints. In practice, many of these generators are very small, and their effect can be overapproximated without adding much conservatism to the RS.

We conservatively approximate (16) by reducing the number of generators after each product, with a `reduce` function implemented as in [26, Proposition 2.2 and Heuristic 2.1]. The `reduce` function keeps the largest n_{red} generators according to a user-defined metric (we used the L^2-norm), then overapproximates the rest of the generators with an axis-aligned box. This ensures the number of rotatotope generators never exceeds a user-specified size. From Lem. 2, each $M_i(t)$ has k^{a}-sliceable generators. If a k^{a}-sliceable generator is chosen for reduction, we no longer consider it k^{a}-sliceable. This is a conservative approach, because slicing reduces the volume of a rotatotope in Alg. 1. A generator that is no longer k^{a}-sliceable cannot decrease the volume of the RS for any choice of k^{a}.

E. Hard Scenarios

The set of Hard Scenarios is shown in Fig. 4.

F. Design Choices and Hyperparameters

ARMTD has several design choices and hyperparameters, all of which can impact the time required for online planning, but none of which impact the strict safety guarantees. That is, **ARMTD guarantees safety independent of design choices**.

The first design choice to consider is the trajectory parameterization. While we provide a generic definition (Def. 1), we find that parameterizing velocities and accelerations in our implementation provides a physical intuition for the planned trajectories. For future work, we plan to explore other parameterizations that provide, for example, smoother motion profiles.

The next design choice to consider are those that define the user-specified cost function, generated at each receding-horizon planning iteration. A more non-convex cost function can slow down online planning. In this work, we generate the cost function by using a high-level planner (HLP) such as an RRT* to generate waypoints between the robot's current location and the global goal. Importantly, the waypoints need not be collision-free; they are used to create a cost function that rewards reaching the waypoint, but ARMTD's safety constraints take care of collision-avoidance. Therefore, **ARMTD provides a safety layer on top of RRT*** or any other HLP (e.g., PRM, or simply picking a waypoint along a straight-line between the robot and the goal).

There are two hyperparameters that determine a tradeoff between conservatism and online planning speed (without impacting safety). The first is the density of the time partition for the JRS. That is, if we partition time more finely to generate the zonotope JRS, then it takes longer to generate and evaluate constraints at runtime (because we have to consider more zonotopes), but the JRS is also less conservative (so, the robot has more free space to move through).

The second hyperparameter is the range of parameters in the trajectory parameterization. A larger range produces a more conservative JRS, because the same number of zonotopes (determined by the time partition) must contain a larger range of joint angles achieved by all parameterized trajectories. We mitigate this problem in practice by precomputing many JRSs (in this work, we used 400), each of which has a narrow range of initial velocity parameters K_i^{v}. We choose the range of acceleration parameters K_i^{a} to vary with the velocity parameters, so that at higher speeds, there is a larger range of available control actions. This reduces conservatism at lower speeds so that ARMTD can maneuver tightly around obstacles.

Note, each JRS only takes around 1 s to compute, since it is only for a single joint, and for the low-dimensional cosine/sine dynamics. At runtime, to construct the RS of the entire arm, we first select the JRS (for each joint) containing the current initial velocity within its narrow range. Then, we slice by the exact initial velocity to produce the RS, and the corresponding collision-avoidance constraints.

G. Seeding CHOMP with RRT*

CHOMP performs better when seeded with a path output by RRT*, as opposed to the default straight-line initialization [6], [34]. Given that ARMTD uses RRT* to generate waypoints at each receding-horizon planning iteration, one may wonder why we do not use the same RRT* to seed CHOMP. However, ARMTD and CHOMP use RRT* in fundamentally different ways. ARMTD plans in a receding-horizon way, so its runtime and safety guarantees are not dependent on the RRT* output. On the other hand, CHOMP would require the RRT* to run for some (unknown) duration, then perform trajectory optimization. In other words, CHOMP requires additional planning time for seeding, whereas ARMTD does not. So, in terms of the most important metric in this work (finding a collision-free trajectory in under $t_{\mathrm{plan}} = 0.5$ s), it is unclear how much time to dedicate for RRT* and how much for CHOMP.

The challenge of generating a fair comparison is compounded by the fact that ARMTD does not require the output of the RRT* to be collision-free. One could potentially use CHOMP in a receding-horizon way, by attempting to reach an intermediate waypoint generated by RRT* in each planning iteration. But, the available open-source CHOMP implementation (via MoveIt! [34]) requires the goal (i.e., intermediate waypoint) to be collision-free. Implementing CHOMP in a more generalized receding-horizon framework is outside the scope of the present work.

References

[1] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, Apr. 1987.

[2] T. Kunz and M. Stilman, "Time-Optimal Trajectory Generation for Path Following with Bounded Acceleration and Velocity," in *Robotics: Science and Systems*, 2012.

[3] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[5] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[6] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[7] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[8] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental Trajectory Optimization for Real-Time Replanning in Dynamic Environments," 2012.

[9] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. Konidaris, "Robot Motion Planning on a Chip," in *Robotics: Science and Systems*, 2016.

[10] T. Kunz, U. Reiser, M. Stilman, and A. Verl, "Real-time path planning for a robot arm in changing environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 5906–5911.

[11] K. Hauser, "On responsiveness, safety, and completeness in real-time motion planning," *Autonomous Robots*, vol. 32, no. 1, pp. 35–48, Jan. 2012.

[12] A. Majumdar and R. Tedrake, "Funnel libraries for real-time robust feedback motion planning," *arXiv preprint arXiv:1601.04037*, 2016.

[13] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the Gap Between Safety and Real-Time Performance in Receding-Horizon Trajectory Design for Mobile Robots," *ArXiv e-prints arXiv:1809.06746*, Sep. 2018.

[14] S. Vaskov, S. Kousik, H. Larson, F. Bu, J. R. Ward, S. Worrall, M. Johnson-Roberson, and R. Vasudevan, "Towards Provably Not-At-Fault Control of Autonomous Robots in Arbitrary Dynamic Environments," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, Jun. 2019.

[15] S. Kousik, P. Holmes, and R. Vasudevan, "Safe, Aggressive Quadrotor Flight via Reachability-Based Trajectory Design," Dynamic Systems and Control Conference, vol. 3, Oct. 2019, V003T19A010.

[16] M. Althoff, A. Giusti, S. B. Liu, and A. Pereira, "Effortless creation of safe robots from modules through self-programming and self-verification," *Science Robotics*, vol. 4, no. 31, 2019.

[17] T. G. A. A. A. Singletary P. Nilsson, "Online Active Safety for Robotic Manipulators," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019.

[18] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "FaSTrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec. 2017, pp. 1517–1522.

[19] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "Decomposition of Reachable Sets and Tubes for a Class of Nonlinear Systems," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3675–3688, Nov. 2018.

[20] T. Fraichard and J. J. Kuffner, "Guaranteeing motion safety for robots," *Autonomous Robots*, vol. 32, no. 3, pp. 173–175, 2012.

[21] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control and verification of high-dimensional systems with dsos and sdsos programming," in *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 394–401.

[22] B. Paden and R. Panja, "Globally asymptotically stable 'PD+'controller for robot manipulators," *International Journal of Control*, vol. 47, no. 6, pp. 1697–1712, 1988.

[23] A. Giusti and M. Althoff, "Efficient Computation of Interval-Arithmetic-Based Robust Controllers for Rigid Robots," in *2017 First IEEE International Conference on Robotic Computing (IRC)*, Apr. 2017, pp. 129–135.

[24] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*, Springer, 2005, pp. 291–305.

[25] M. Althoff, "An Introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[26] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," PhD thesis, Technische Universität München, 2010.

[27] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[28] J.-M. Lien and N. M. Amato, "Approximate convex decomposition of polyhedra," in *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, 2007, pp. 121–131.

[29] L. J. Guibas, A. Nguyen, and L. Zhang, "Zonotopes as bounding volumes," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2003, pp. 803–812.

[30] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, pp. 2004–2005, 2003.

[31] E. Polak, *Optimization: algorithms and consistent approximations*. Springer Science & Business Media, 2012, vol. 124.

[32] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on Autonomous Mobile Service Robots*, 2016.

[33] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[34] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study," *CoRR*, vol. abs/1404.3785, 2014.

[35] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[36] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.