Abstract—A redundant manipulator has multiple inverse kinematics solutions per an end-effector pose. Accordingly, there can be many trajectories for joints that follow a given end-effector path in a Cartesian space. In this paper, we present a trajectory optimization of a redundant manipulator (TORM) to synthesize a trajectory that follows a given end-effector path accurately, while achieving the smoothness and collision-free manipulation. Given these desirable properties, our method optimizes a trajectory using two-stage gradient descent to reduce potential competition between different properties during the update. To further improve the performance and avoid falling into local minima, we apply the quantum annealing that iteratively randomizes various configurations of the trajectory, followed by updating the trajectory. We first show benefits of our method with environments containing external obstacles. We then compare ours with the state-of-the-art methods in their favorable setting, environments without having obstacles. Our method robustly minimizes the pose error in a progressive manner while satisfying various desirable properties.

I. INTRODUCTION

Remote control of various robots has been one of the main challenges in the robotics, while it is commonly used for cases where it is difficult or dangerous for a human to perform tasks [1], [2]. In this remote control scenario, a robot has to follow the task command accurately while considering its surrounding environment and constraints of the robot itself.

In the case of a redundant manipulator that this paper focuses on, a sequence of finely-specified joint configurations is required to follow the end-effector path in a Cartesian space accurately. Traditionally, inverse kinematics (IK) has been used to determine joint configurations given an end-effector pose. The traditional IK, however, cannot consider the continuity of configurations, collision avoidance, and kinematic singularities that arise when considering to follow the end-effector path.

Prior approaches [3], [4] solve this problem using nonlinear optimization by considering these constraints. While these approaches also avoid self-collisions using a neural network, they do not deal with collisions for external obstacles. Main Contributions. In this work, we present a trajectory optimization of a redundant manipulator (TORM) for synthesizing a trajectory that is accurately following a given path as well as smooth and collision-free against the robot itself and external obstacles (Fig. 1). Our method is based on the two-stage gradient descent, inspired by CHOMP [5], and incorporates the aforementioned properties into our trajectory optimization process. Based on the gradients of the objective functions, we iteratively update a trajectory through a two-stage update manner, making a feasible trajectory while accurately following the given end-effector path (Sec. III-B). Since this optimization process is based on local updates, it can be stuck in local minima. To avoid local minima, we adopt the quantum annealing, an effective randomized technique, into our method for progressively identifying better trajectories (Sec. III-D).

To see the benefits of each component of our method, we test our method with its ablated methods on two different scenarios containing external obstacles (Sec. IV-A). To compare ours with the state-of-the-art methods, RelaxedIK [3] and Stampede [4], we test three different scenarios that do not have any obstacles (Sec. IV-B). Overall, we are able to observe that our method robustly minimize the pose error reasonably fast in both obstacle and non-obstacle environments. The synthesized results of tested problems and real robot verifications can be seen in the attached video.

II. RELATED WORK

In this section, we discuss prior studies in the fields of inverse kinematics and optimization-based motion planning.

A. Inverse Kinematics

Inverse kinematics (IK) has been studied widely to find a joint configuration from an end-effector pose [6]. In the case of a redundant manipulator, where our target robots belong to, there can be various joint configurations from a single end-effector pose. For this problem, several techniques...
for quickly finding solutions have been proposed [7], [8]. In particular, task-priority IK algorithms [9], [10], [11] prioritize solutions based on an objective function for each purpose.

Most methods that synthesize a trajectory geometrically constrained for an end-effector pose are based on the Jacobian matrix instead of IK for finding a feasible solution among various solution candidates [12], [13], [14], [15]. Unfortunately, the Jacobian matrix is the first derivative of the vector-valued function with multiple variables and thus it can cause false-negative failures by getting stuck in local minima or convergence failures due to the limits of the joint angle [16].

Trac-IK [16] points out the problem and improves success rates by using randomly selected joint configurations and sequential quadratic programming. Nonetheless, its solutions do not guarantee continuity of a sequence of joint configurations to make a feasible trajectory [3]. In summary, the traditional IK approaches have different strengths and weaknesses for synthesizing a feasible trajectory along a given end-effector path.

To get alleviated solutions, many studies present optimization techniques with objective functions for synthesizing a feasible trajectory with matching end-effector poses. Luo and Hauser [17] use a geometric and temporal optimization to generate a dynamically-feasible trajectory from a sketch input. Recently, Rakita et al. [3] proposed a real-time approach using a weighted-sum non-linear optimization, called RelaxedIK, to solve the IK problem for a sequence of motion inputs. Since the collision checking has a relatively large computational overhead, RelaxedIK uses a neural network for fast self-collision avoidance.

Based on RelaxedIK, two studies [4], [18] are proposed for synthesizing a highly-accurate trajectory on off-line. One of them is Stampede [4], which finds an optimal trajectory using a dynamic programming algorithm in a discrete-space-graph that is built by samples of IK solutions. The other work [18] generates multiple candidate trajectories from multiple starting configurations and then selects the best trajectory with a user guide by allowing a deviation if there are risks of self-collisions or kinematic singularities [18].

Our method also seeks to find a highly-accurate solution for the redundant manipulators while avoiding self-collisions. On top of that, we additionally consider collision avoidance against external obstacles. To synthesize an accurate and feasible trajectory without any collisions against a robot itself as well as obstacles, we propose a trajectory optimization of a redundant manipulator (TORM) that extends the CHOMP [19], one of the most prominent optimization-based motion planning approaches for our problem using a redundant manipulator.

### B. Optimization-based Motion Planning

The optimization-based motion planning approach quickly finds a collision-free trajectory by incorporating an efficient collision avoidance method into optimization techniques. Some of well-known techniques include covariant gradient algorithm (CHOMP) [19], stochastic optimization without gradient information (STOMP) [20], and sequential convex optimization (Trajopt) [5]. Thanks to high-quality trajectory, these techniques have been extended to different directions (e.g., constrained motion planning) [21], [22], [23].

Our method is based on CHOMP, but we extend it by adding an additional objective term to consider a given end-effector path. The main difference over the original CHOMP is that CHOMP basically finds a collision-free trajectory from start to goal configurations, but our task is to synthesize a feasible and accurate trajectory that follows a given end-effector path. Also, CHOMP based on gradient descent is a local optimization, and thus it can yield a sub-optimal trajectory. Accordingly, the search space of our problem has many local minima by considering various objective terms. In this work, we adopt quantum annealing [24], a randomized technique, to avoid local minima.

### III. Trajectory Optimization

Our goal is to find a trajectory, $\xi$, of a redundant manipulator for a given end-effector path, $\mathcal{X}$. The trajectory $\xi$ is a sequence of joint configurations, and the end-effector path, $\mathcal{X} \subseteq \mathbb{R}^6$, is defined in the 6-dimensional Cartesian space.

We solve the problem by a waypoint parameterization [25] of the path that finely splits the given end-effector path, $\mathcal{X} \approx \mathcal{X} = (x_0, x_1, ..., x_n)$; $x$ is an end-effector pose. We then compute the joint configurations for end-effector poses $\mathcal{X}$. As a result, the trajectory is approximated as: $\xi \approx (q_0^T, q_1^T, ..., q_n^T)^T \subseteq \mathbb{R}^{(n+1) \times d}$, where $d$ is the degree of freedom (DoF) of a manipulator; $q_0$ is a start configuration and $q_n$ is a goal configuration for the computed trajectory of joint configurations. Main notations are summarized in Table I.

Our target robot is a redundant manipulator that has multiple joint configurations given an end-effector pose; if $d > 6$, it has infinitely many valid solutions. Among many candidates, we find appropriate joint configurations to generate a trajectory that is smooth and accurate as well as collision-free for the given end-effector path (Fig. 2).
To achieve our goal, we present a trajectory optimization of a redundant manipulator (TORM) by adapting an optimization-based motion planning approach, specifically, CHOMP [19], for finding a collision-free trajectory from the start to the goal configurations. Since CHOMP not only quickly optimizes a trajectory using gradient descent, but also efficiently avoids collisions by incorporating objective functions, we thus choose CHOMP as a base trajectory optimization method of our approach.

In the following section, we explain how to optimize an objective function and how to avoid getting stuck in local minima or violating constraints. To do so, we first define our own objective function, which has a new path following objective (Sec. III-A). We then update the objectives based on our two-stage gradient descent (Sec. III-B), followed by the quantum annealing that starts with our initial trajectory (Sec. III-D).

A. Objective Function

We solve our problem by modeling an objective function with three different properties that are 1) matching the end-effector pose, 2) smoothness, and 3) avoiding collisions:

\[
U(\xi) = F_{\text{pose}}(\xi) + \lambda_1 F_{\text{smooth}}(\xi) + \lambda_2 F_{\text{obs}}(\xi),
\]

where \( \lambda \) is a regularization constant. \( F_{\text{pose}} \) is introduced to minimize the pose error between the target and current end-effector poses, and is defined using the Euclidean distance:

\[
F_{\text{pose}}(\xi) = \sum_{i=0}^{n} \left\| x_i - FK(q_i) \right\|,
\]

where \( FK(q_i) \) computes the end-effector pose at the joint configurations \( q_i \) using forward kinematics (FK).

For both smooth term \( F_{\text{smooth}} \) and obstacle term \( F_{\text{obs}} \), we follow the definitions presented in CHOMP. \( F_{\text{smooth}} \) measures dynamical quantities, i.e., the squared sum of derivatives to guarantee the smoothness:

\[
F_{\text{smooth}}(\xi) = \frac{1}{\Delta t} \sum_{i=0}^{n-1} \left\| \frac{q_{i+1} - q_i}{\Delta t} \right\|^2.
\]

\( F_{\text{obs}} \) quantitatively measures a proximity to obstacles for avoiding collisions:

\[
F_{\text{obs}}(\xi) = \sum_{i=0}^{n-1} \sum_{b} \frac{1}{2} \left(c(x(q_{i+1}, b)) + c(x(q_{i}, b))\right) \left\| x(q_{i+1}, b) - x(q_{i}, b) \right\|,
\]

where \( x(q_i, b) \) is partial forward kinematics, i.e., a position of a body point \( b \in B \) in the workspace at a configuration \( q_i \), and \( c(\cdot) \) is an obstacle cost computed from a signed distance field that can be calculated analytically from a geometry of workspace obstacles (see the details in [19]). At a high level, it approximately measures the sum of penetration depths between the robot body and the obstacles.

B. Two-Stage Gradient Descent Update

We iteratively update the trajectory to minimize the cost of objective functions with three different properties. To minimize the cost, our update rule is based on the gradient descent technique adopted in many optimization methods. The functional gradient of the obstacle term, \( \nabla F_{\text{obs}} \), pushes the robot out of obstacles, and the functional gradient of the smooth term, \( \nabla F_{\text{smooth}} \), keeps the continuity between joint configurations.

Both \( \nabla F_{\text{obs}} \) and \( \nabla F_{\text{smooth}} \) are responsible for synthesizing a feasible trajectory, whereas the functional gradient of the pose term, \( \nabla F_{\text{pose}} \), is responsible for matching the end-effector poses \( \mathcal{X} \). Even though we can update the trajectory by considering the three functional gradients simultaneously, we found that it can lead to conflicts between each functional gradient.

To alleviate this problem, we present a two-stage gradient descent (TSGD) that consists of updating to make a feasible trajectory and updating the trajectory to match closer to \( \mathcal{X} \). The TSGD is repeated in which each odd iteration updates the trajectory using \( \nabla F_{\text{smooth}} \) and \( \nabla F_{\text{obs}} \) and in which even iteration updates the trajectory using \( \nabla F_{\text{pose}} \):

\[
\xi_{i+1} = \begin{cases} 
\xi_i - \eta_1 A^{-1}(\lambda_1 \nabla F_{\text{smooth}} + \lambda_2 \nabla F_{\text{obs}}), & \text{if } i \text{ is odd}, \\
\xi_i - \eta_2 \nabla F_{\text{pose}}, & \text{otherwise},
\end{cases}
\]

where \( \eta \) is a learning rate, and \( A \) is from an equally transformed representation of the smooth term, \( F_{\text{smooth}} = \frac{1}{2} \xi^T A \xi + \xi^T b + c; A^{-1} \) acts to retain smoothness and to accelerate the optimization by having a small amount of impact on the overall trajectory [19]. Since \( F_{\text{smooth}} \) and \( F_{\text{obs}} \) have a correlation between consecutive joint configurations, it is effective for updating the trajectory with covariant gradient descent using \( A^{-1} \). On the other hand, \( F_{\text{pose}} \) does not consider consecutive joint configurations, thus we do not use \( A^{-1} \).

Our problem is to avoid collisions and to follow the given end-effector path in the Cartesian workspace, while we have to solve the problem through a sequential movement of the joint configurations in the configuration space (C-space). Although \( F_{\text{pose}} \) and \( F_{\text{obs}} \) are computed in workspace using FK, \( \nabla F_{\text{pose}} \) and \( \nabla F_{\text{obs}} \) should be defined in C-space for calculating the change of joint configurations.

Since \( \left\| x_i - FK(q_i) \right\| \) of Eq. 2 can be represented as \( \frac{1}{2} (x_i - FK(q_i))^T (x_i - FK(q_i)) \), we can derive the functional gradient of the pose term, \( \nabla F_{\text{pose}} \), by the following:

\[
\nabla F_{\text{pose}}(q_i) = J^T (x_i - FK(q_i)),
\]
where \( J = \frac{\partial A}{\partial q} \in \mathbb{R}^{6 \times d} \) is the Jacobian matrix. Note that the Jacobian-based approach can lead to false-negative failures as explained in Sec. II-A. We, however, found that our Jacobian-based approach can lead to false-negative failures.

\[
\nabla F_{\text{obs}}(q_i) = \sum_{b} \left( ||x_i,b|| - c(x_i,b) \right),
\]

where \( x_i,b \) is equal to \( x(q_i,b) \), \( c_i,b \) to \( c(x_i,b) \), \( \dot{x}_i,b \) is the normalized velocity vector, and \( \kappa = ||x_i,b||^{-3} (I - \dot{x}_i,b^T \dot{x}_i,b) x_i \) is the curvature vector (see the details in [19]). Since \( F_{\text{smooth}} \) is already represented in C-space, \( \nabla F_{\text{smooth}} \) is represented simply as \( \nabla F_{\text{smooth}}(\xi) = A\xi + b \).

Once a start configuration \( q_0 \) is given, we iteratively update from its next configuration, \( q_1 \), to the goal configuration \( q_n \), based on our TSGD. In our problem, however, the start configuration may not be given, while an end-effector pose is given.

C. Finding an Initial Trajectory

We can optimize a trajectory based on our update rule, but it is not straightforward to get an initial trajectory for our task, because only the end-effector path is given. In this section, we introduce a simple, heuristic method of constructing an initial trajectory in a greedy manner, which is updated in our quantum annealing (Sec. III-D).

Constructing a good initial trajectory faces the main challenge, which we need to consider our objectives, smoothness, avoiding obstacles, and following a given path. Note that there can be multiple or even an infinite number of joint configurations at an end-effector pose. Therefore, simply obtaining joint configurations using IK at each end-effector pose and linearly interpolating, some of them is not a good approach for computing an initial trajectory.

As the first step, we simplify the given end-effector poses \( \mathcal{X} \) into simplified poses, \( S = \{ s_0, \ldots, s_n \} \), since the target end-effector path can be represented in a finely tessellated form with many poses, and working with more poses tends to increase the complexity of generating a reasonably initial trajectory. Specifically, we choose the Ramer-Douglas-Peucker algorithm [26]. Starting from the first simplified pose, we find suitable joint configurations at the pose, and we connect one of the configurations with another one in the next pose. For choosing the joint configuration, we greedily select one that minimizes our objective function. Lastly, we connect chosen joint configurations for computing the initial trajectory through linear interpolation.

D. Quantum Annealing

While we got an initial trajectory, there is a high probability that the trajectory is sub-optimal due to the local natures of our update method and initial trajectory construction method. Furthermore, there can be many surrounding local minima in terms of our optimization search space, since we incorporate three different properties of constraints into the objective functions.

To avoid getting stuck in local minima, many prior approaches adopt a kind of global optimization, i.e., simulated annealing [27] or stochastic methods [28]. In this work, we apply the quantum annealing (QA) that is more effective in getting out of many surrounding local minima, the main characteristics of our problem, than the simulated annealing [24].

QA is a kind of randomized algorithms, which prepares random parameters utilizing a quantum transition, and performs a local optimization. The quantum transition searches a lower cost by random explorations controlled by a parameter \( \gamma \). The parameter \( \gamma \) controls the intensity of randomness depending on the cost function.

Adopting QA, our algorithm iteratively performs two parts, as shown in Alg. 1: exploring a better trajectory (line 4-7 of Alg. 1) and updating the trajectory (line 3 of Alg. 1). Unlike the original QA, we use a sequential exploring (SE) that gives more weight on the exploration part by repeating the quantum transition \( \beta \) times. The SE is chosen for robustly identifying a high-quality solution given our randomized algorithm.

The exploration part uses the quantum transition to find a new trajectory, \( \xi_{\text{new}} \), and then refines the new trajectory \( \xi_{\text{new}} \) by two-stage gradient descent. In our case, the quantum transition explores a potentially good solution by estimating randomly constructed trajectories using our objective functions (line 5 of Alg. 1). The trajectories are computed by connecting random joint configurations obtained at simplified poses using IK.

When the new trajectory \( \xi_{\text{new}} \) taken during the exploration has a lower cost, we use it to the current trajectory (line 7 of Alg. 1). However, if the new trajectory \( \xi_{\text{new}} \) violates the constraints, we do not use it to the current trajectory (Sec. III- E). The update part locally refines the trajectory using the two-stage gradient descent (Sec. III-B). This process continues until satisfying a given condition, e.g., running time or cost.

E. Trajectory Constraints

We synthesize the desired trajectory by finding a low cost through our optimization process that incorporates our objective functions. Although the optimization process finds a low cost solution, it can violate several constraints.
example, a trajectory can have collisions with obstacles or self-collisions, even though the trajectory accurately follows the given end-effector pose. Hence, we check collisions every time we find a better trajectory during the quantum annealing process.

In addition to checking collision, a manipulator commonly has several constraints that are satisfying lower and upper limits of joints, velocity limits, and kinematic singularities for joints. In the case of lower and upper limits of joints, it is traditionally handled by performing $L_1$ projection that resets the violating joints value to its limit value. To retain smoothness, we use a smooth projection used by CHOMP [19] during the update process. The smooth projection uses the Riemannian metric $A^{-1}$, which can be written as $\tilde{\xi} = \xi + \alpha A^{-1} v$, where $v$ is the vector of joint values, and $\alpha$ is a scale constant. This process is repeated until there is no violation.

For other constraints like the velocity limit, we check them together while checking collisions. The velocity limit for joints is checked by computing the velocities of joints between $q_{t-1}$ and $q_t$ for a given time interval, $\Delta t$. Another constraint is the kinematic singularity that is a point where the robot is unstable, and it can occur when the Jacobian matrix loses full rank. To check kinematic singularities, we use the manipulability measure [29] used by Relaxed-IK [3]. At a high level, it avoids making the manipulability measure less than a certain value that is computed by random samples. Note that our method returns the lowest cost trajectory guaranteed through checking constraints for constructing a feasible trajectory.

IV. EXPERIMENTS AND ANALYSIS

We use the Fetch robot, whose manipulator arm has 7-DoF for various tests. We report the average performance by performing 20 tests with a machine that has 3.4 GHz Intel i7-6700 CPU and 16 GB RAM. Also, we set $\lambda_1 = \frac{5.0}{n+1}$, $\lambda_2 = 1.8$, $\eta_1 = 0.03$, and $\eta_2 = 1.0$.

In this setting, the two-stage gradient descent, the quantum transition, and checking constraints of our method take 61%, 37%, and 1% of the overall running time; the two-stage gradient descent is frequently iterated to refine the trajectory, as the main update operation.

![Fig. 3](image-url) This shows our problems that are to follow the given end-effector path (red lines) $X^*$ while avoiding obstacles for a sequence of joint configurations. The start configuration $q_0$ is fixed and located close to the obstacles. (a) is to trace the square, and (b) is to trace the “S”. Both (a) and (b) must avoid the table, and (b) additionally considers the blue box.

(a) Square tracing w/ obstacles.  
(b) “S” tracing w/ obstacles.

![Fig. 4](image-url) This shows the pose error (Eq. 2) over the planning time of our methods. We test our method w/o the two-stage gradient descent (TSGD), w/o the quantum annealing (QA), and w/o the sequential exploring (SE). We visualize graphs once an initial solution is computed.

| Free start configuration | Square tracing w/ obstacles | “S” tracing w/ obstacles | Ours w/o SE | Ours w/o QA | Ours w/o TSGD | Ours |
|--------------------------|-----------------------------|--------------------------|------------|------------|---------------|------|
|                          | Average | Min     | Max     | AJJ       | IST        | # of failures |
| Square tracing w/o SE    | 2.09e-2 | 5.19e-3 | 4.47e-2 | 0.73       | 1.81        | 0 |
| Square tracing w/o QA    | 3.99e-2 | 3.35e-5 | 9.85e-2 | 0.72       | 1.13        | 4 |
| Square tracing w/o SE    | 1.3e-4  | 3.4e-5  | 1.05e-3 | 0.67       | 2.32        | 0 |
| “S” tracing w/o SE       | 8.25e-5 | 2.56e-5 | 1.80e-4 | 0.65       | 1.89        | 0 |
| “S” tracing w/o QA       | 5.95e-3 | 1.20e-3 | 4.18e-2 | 0.35       | 4.04        | 0 |
| “S” tracing w/o SE       | 3.80e-2 | 1.65e-6 | 1.18e-4 | 0.38       | 1.61        | 3 |
| “S” tracing w/o SE       | 3.44e-5 | 1.72e-6 | 4.45e-4 | 0.34       | 3.71        | 0 |

Table II

| Free start configuration | Pose error | AJJ | IST | # of failures |
|--------------------------|------------|-----|-----|---------------|
| Square tracing w/o SE    | 5.19e-2    | 0.73| 1.81| 0 |
| Square tracing w/o QA    | 3.99e-2    | 0.72| 1.13| 4 |
| Square tracing w/o SE    | 1.3e-4     | 0.67| 2.32| 0 |
| “S” tracing w/o SE       | 8.25e-5    | 0.65| 1.89| 0 |
| “S” tracing w/o QA       | 5.95e-3    | 0.35| 4.04| 0 |
| “S” tracing w/o SE       | 3.80e-2    | 0.38| 1.61| 3 |
| “S” tracing w/o SE       | 3.44e-5    | 0.34| 3.71| 0 |

A. Evaluation w/ Obstacles

We first evaluate our method with obstacles. For this test, we set up two different benchmarks, i.e., the square tracing and “S” tracing problems with obstacles (Fig. 3). For the square tracing, it is necessary to avoid collisions with the table, and we have to consider the blue box on the table additionally in the “S” tracing problem.

We first evaluate whether our method accurately follows the given end-effector path while avoiding obstacles in two obstacle problems (Fig. 3). For this test, we prepare ablated methods by disabling the two-stage gradient descents (TSGD) and thus updating three functional gradients at once. We also test our methods without the quantum annealing (QA) and without the sequential exploring (SE) in QA. Excluding the SE of $\beta$ is set to 1, and we test our method by setting $\beta$ to 4. Fig. 4 and Table II show the results in two obstacle problems. Note that these results were extracted from feasible trajectories satisfying the given constraints (Sec. III-E).

The results of both problems show similar performance trend between tested methods. Excluding the TSGD has a higher pose error on average than our method and also has the highest min. pose error among 20 tests, compared to other methods. These results demonstrate that the different functional gradients compete with each other. Therefore, the TSGD prevents the competition of different functional gradients and is useful to get a highly-accurate solution.
In this paper, we have presented the trajectory optimization of a redundant manipulator (TORM) based on the two-stage gradient descent to follow a given end-effector path and to make a feasible trajectory in the environment with obstacles. We have also applied quantum annealing for our optimization to avoid local minima. We have shown benefits of our method over the state-of-the-art techniques in environments without obstacles where those prior methods can work while demonstrating our method working well even with obstacles. Further, we have verified the feasibility of our synthesized trajectory using the real, Fetch manipulator.

While our method finds an accurate solution reasonably fast, there is no theoretical analysis of the error reduction rate of our approach. While the theoretical analysis could be challenging, it would shed light on deeply understanding the accuracy of our method.
REFERENCES

[1] Kapil D Katyal, Christopher Y Brown, Steven A Hechtman, Matthew P Para, Timothy G McGee, Kevin C Wolfe, Ryan J Murphy, Michael DM Kutzer, Edward W Tunstel, Michael P McLoughlin, et al., “Approaches to robotic teleoperation in a disaster scenario: From supervised autonomy to direct control”, in IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014, pp. 1874–1881.

[2] Philip Long, Tarik Keleştemur, Aykut Özgün Önol, and Taşkin Padir, “optimization-based human-in-the-loop manipulation using joint space polytopes”, in IEEE International Conference on Robotics and Automation. IEEE, 2019, pp. 204–210.

[3] Daniel Rakita, Bilge Mutlu, and Michael Gleicher, “RelaxedIK: Real-time synthesis of accurate and feasible robot arm motion.”, in Robotics: Science and Systems. Citeseer, 2013, vol. 9, pp. 1–10.

[4] Daniel Rakita, Bilge Mutlu, and Michael Gleicher, “STAMPEDE: A discrete-optimization method for solving pathwise-inverse kinematics”, in IEEE International Conference on Robotics and Automation. IEEE, 2019, pp. 3507–3513.

[5] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel, “Finding locally optimal, collision-free trajectories with sequential convex optimization”, in Robotics: Science and Systems. Citeseer, 2013, vol. 9, pp. 1–10.

[6] Samuel R Buss, “Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods”, IEEE Journal of Robotics and Automation, vol. 17, no. 1-19, pp. 16, 2004.

[7] Rosen Diankov, “Automated construction of robotic manipulation programs”, 2010.

[8] Anirban Sinha and Nilanjan Chakraborty, “Geometric search-based inverse kinematics of 7-dof redundant manipulator with multiple joint offsets”, in IEEE International Conference on Robotics and Automation. IEEE, 2019, pp. 5592–5598.

[9] Pasquale Chiacchio, Stefano Chiaverini, Lorenzo Sciavicco, and Bruno Sicilliano, “Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy”, The International Journal of Robotics Research, vol. 10, no. 4, pp. 410–425, 1991.

[10] Stefano Chiaverini, “Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators”, IEEE Transactions on Robotics and Automation, vol. 13, no. 3, pp. 398–410, 1997.

[11] Oussama Kanoun, Florent Lamiraux, and Pierre-Brice Wieber, “Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task”, IEEE Transactions on Robotics, vol. 27, no. 4, pp. 785–792, 2011.

[12] Mike Stilman, “Task constrained motion planning in robot joint space”, in IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2007, pp. 3074–3081.

[13] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner, “Manipulation planning on constraint manifolds”, in IEEE International Conference on Robotics and Automation. IEEE, 2009, pp. 625–632.

[14] Nikolaus Vahrenkamp, Dmitry Berenson, Tanvir Asfour, James Kuffner, and Rudiger Dillmann, “Humanoid motion planning for dual-arm manipulation and re-grasping tasks”, in IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2009, pp. 2464–2470.

[15] Tobias Kunz and Mike Stilman, “Manipulation planning with soft task constraints”, in IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012, pp. 1937–1942.

[16] Patrick Beeson and Barrett Ames, “TRAC-IK: An open-source library for improved solving of generic inverse kinematics”, in IEEE International Conference on Humanoid Robots. IEEE, 2015, pp. 928–935.

[17] Jingru Luo and Kris Hauser, “Interactive generation of dynamically feasible robot trajectories from sketches using temporal mimicking”, in IEEE International Conference on Robotics and Automation. IEEE, 2012, pp. 3665–3670.

[18] Pragathi Praveena, Daniel Rakita, Bilge Mutlu, and Michael Gleicher, “User-guided offline synthesis of robot arm motion from 6-dof paths”, in IEEE International Conference on Robotics and Automation. IEEE, 2019, pp. 8825–8831.

[19] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellen, J Andrew Bagnell, and Siddhartha S Srinivasa, “CHOMP: Covariant hamiltonian optimization for motion planning”, The International Journal of Robotics Research, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[20] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal, “STOMP: Stochastic trajectory optimization for motion planning”, in IEEE International Conference on Robotics and Automation. IEEE, 2011, pp. 4569–4574.

[21] Anca D Dragan, Nathan D Ratliff, and Siddhartha S Srinivasa, “Manipulation planning with goal sets using constrained trajectory optimization”, in IEEE International Conference on Robotics and Automation. IEEE, 2011, pp. 4582–4588.

[22] Nikita Kitaev, Igor Mordatch, Sachin Patil, and Pieter Abbeel, “Physics-based trajectory optimization for grasping in cluttered environments”, in IEEE International Conference on Robotics and Automation. IEEE, 2015, pp. 3102–3109.

[23] Donghyuk Kim, Mincheul Kang, and Sung-eui Yoon, “Volumetric tree*: Adaptive sparse graph for effective exploration of homotopy classes”, in IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2019.

[24] Diego de Falco and Dario Tassacelli, “An introduction to quantum annealing”, RAIRO-Theoretical Informatics and Applications, vol. 45, no. 1, pp. 99–116, 2011.

[25] Tamar Flash and Renfrey B Potts, “Communication: Discrete trajectory planning”, The International Journal of Robotics Research, vol. 7, no. 5, pp. 48–57, 1988.

[26] David H Douglas and Thomas K Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature”, Cartographica: The International Journal for Geographic Information and Geovisualization, vol. 10, no. 2, pp. 112–122, 1973.

[27] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi, “Optimization by simulated annealing”, Science, vol. 220, no. 4598, pp. 671–680, 1983.

[28] Wolfgang Wenzel and Kay Hamacher, “Stochastic tunneling approach for global minimization of complex potential energy landscapes”, Physical Review Letters, vol. 82, no. 15, pp. 3003, 1999.

[29] Tsuneo Yoshikawa, “Manipulability of robotic mechanisms”, The International Journal of Robotics Research, vol. 4, no. 2, pp. 3–9, 1985.