Maximizing the Probability of Task Completion for Redundant Robots Experiencing Locked Joint Failures

Biyun Xie, Member, IEEE, and Anthony A. Maciejewski, Fellow, IEEE

Abstract—This article considers the problem of planning a trajectory that maximizes the probability that a robot will be able to complete a set of point-to-point tasks, after experiencing locked joint failures. The proposed approach first develops a method to calculate the probability of task failure for an arbitrary trajectory based on its failure scenarios, which are efficiently computed by identifying the ranges of task point self-motion manifolds. Then, a novel trajectory planning algorithm is proposed to find the optimal trajectory with maximum probability of task completion. The planning algorithm exploits the overlap of self-motion manifold bounding boxes, as opposed to always using the shortest distance, to determine an optimal trajectory. The proposed trajectory planning algorithm is demonstrated on planar positioning 3R, spatial positioning 4R, and spatial positioning/orienting 7R redundant robots, resulting in average improvement of 17%, 22%, and 30%, respectively, compared to the best shortest distance trajectory.

Index Terms—Fault tolerance, kinematics, motion and path planning, redundant robots.

I. INTRODUCTION

Fault tolerance is an essential property for robots operating in remote or hazardous environments [2]–[4], where it is difficult or dangerous to fix the robot after a failure. For example, space robot arms are commonly used to collect and transfer samples in robotic planetary surface exploration. The extremely low temperatures and high-radiation levels on some planets pose a significant challenge to the robustness and reliability of these robotic systems, which are required to continue operating without intervention in the presence of unpredictable failures [5]. Another example is rescue robots working in precarious environments due to natural or man-made disasters, where a robot should be able to complete its assigned tasks even after experiencing a failure [6].

Many failures can be represented by the inability to control one of the degrees of freedom (DOFs). In such cases, employing kinematically redundant robots is one way to realize fault tolerance, where the extra DOFs can be used to compensate for joint failures. Frequently, failures either result in a locked joint, or can be transformed into a locked joint by applying brakes [5], for example, with failures that cause free swinging joints [7].

One fundamental problem is how to take advantage of the extra DOFs of redundant robots to tolerate joint failures during motion, i.e., fault-tolerant motion planning. Fault-tolerant motion planning strategies can be applied to recover from joint failures [8], [9] or be incorporated in anticipation of failures. In the latter case, the goal is to keep the robot in a configuration that guarantees good performance after a potential failure. This guarantee takes different forms depending on the type of task being performed.

For point-to-point tasks, such as pick and place, the robot should be able to reach all task points even after an arbitrary joint failure. This can be guaranteed by making sure that none of the robot’s joints go outside of the ranges of the self-motion manifolds associated with every task point. For a single task point, this range is determined by the bounding box enclosing the self-motion manifolds of that task point. Similarly, for multiple task points, the robot configuration must be constrained to the intersection of all bounding boxes, which provides a set of software-imposed artificial joint limits to ensure fault tolerance [10]. This is more efficient than previous techniques that exhaustively check all configurations along a trajectory to see if the task point can be reached after a failure [11], [12]. In addition, limiting the joint motion prior to a failure is able to guarantee fault tolerance, as opposed to recovery strategies that only recompute the inverse kinematics after a failure occurs [13], [14].

For path tracking tasks, such as cutting, the robot should be able to complete the entire path after a failure. This can be guaranteed by making sure that the end effector path is within the “fault-tolerant workspace,” which is defined as the intersection of the prefailure and all postfailure workspaces [15], [16]. The size and shape of these workspaces are determined by
the artificial joint limits that are applied prior to a failure. In those cases where a relatively large prefailure workspace is required, it may not be possible to guarantee a sufficient fault-tolerant workspace. For such cases, the reliability map, which is defined based on the joint failure probabilities, can be used to maximize the probability of task completion [17].

Fault tolerance for the first type of task, i.e., point-to-point tasks, is studied in this article. Unfortunately, the abovementioned method based on artificial joint limits [10] only works for task points that satisfy a very restrictive set of conditions. Therefore, this article studies the fault-tolerant motion planning problem for point-to-point tasks from a probability point of view. However, unlike reliability maps [17] that assign probabilities to task points, probabilities are computed for entire joint-space trajectories. Specifically, given a starting configuration in the joint space and a set of ordered tasks points in the task space, one needs to plan a trajectory from the starting configuration to these tasks points, in the specified order, that maximizes the probability of reaching all the tasks when locked joint failures can occur. Because we are not concerned about the end-effector trajectory between task points the kinematics of the trajectory is determined in the joint space. After the robot reaches each task point, we assume that it only moves its joints within a small range to complete the assigned fine-manipulation task required at this point. The main contributions of this article are as follows: 1) A new method to compute the failure probability of an arbitrary joint trajectory; and 2) an algorithm for determining a trajectory with maximum probability of task completion.

The rest of this article is organized as follows. In the next section a new technique for computing the probability of task failure for an arbitrary joint trajectory is presented. The algorithm for determining the optimal trajectory with maximum probability of task completion is discussed in Section III and examples of applying this algorithm are given in Section IV. Finally, Section V concludes this article.

II. CALCULATING THE PROBABILITY OF TASK FAILURE FOR AN ARBITRARY JOINT TRAJECTORY

A. Background on Bounding Boxes of Self-Motion Manifolds

Given a set of joint angles, denoted $\theta$, the end-effector location of the robot, denoted $x$, can be easily obtained by the following forward kinematics equation:

$$x = f(\theta).$$  \hfill (1)\]

However, the inverse kinematics of a robot is much more complicated. In particular, for redundant robots, there typically exists an infinite number of joint angles for a given task point, i.e., end-effector location inside the workspace, due to the extra DOFs. The self-motion manifolds of a task point consist of all these joint angles, two examples of which are shown by the blue and green curves in Fig. 1. When the robot moves along these self-motion manifolds, the end-effector of the robot stays at the same location. It is important to note that the dimension of the self-motion manifold is equal to the degree of redundancy.

To guarantee that a joint trajectory is fault tolerant, i.e., can reach a given task point after a joint failure, the robot motion must be restricted to the bounding box enclosing the self-motion manifold associated with this task point, as shown by the blue and green boxes for tasks 1 and 2, respectively, in Fig. 1. For one degree of redundancy, arguably the most common way to calculate the self-motion manifolds of a task point is to move along the null vector of the Jacobian with a small step to reach a new configuration on the self-motion manifold, and repeat this operation. This works because the null vector is tangent to the self-motion manifold. For higher degrees of redundancy, one can project a unit vector along the ith joint axis onto the null space of the Jacobian, and iterate until the unit vector is orthogonal to the null space. This provides an estimate of the bounding box for joint i [18].

Restricting the robot motion to the bounding box can be done by imposing artificial joint limits, which are then released after a failure occurs. If there are multiple task points, then the robot must be restricted to the intersection of these bounding boxes, as shown by the shaded area in Fig. 1 [10]. Unfortunately, when there are many task points or task points are far away from each other, there may not be an intersection between the self-motion manifolds of all task points. Furthermore, even if there exists an intersection, the robot may not be able to reach all the task points when restricted to this intersection, as is illustrated in Fig. 1, where the robot will not be able to reach either task 1 or task 2, if it can only move inside the intersection of the bounding boxes. For these cases, where a fault-tolerant joint trajectory cannot be guaranteed, one may want to identify the trajectory that has the highest probability of task completion, which is the topic of this article.

B. Joint Space Division

To efficiently compute the failure probability of a trajectory, one can first divide the joint space up into regions of different “failure scenarios” that correspond to specific joints failing and whether that failure prevents the robot from reaching the following desired task points. The boundaries of these regions correspond to the boundaries of the intersection(s) of the bounding boxes enclosing the self-motion manifolds of the following desired task points. These boundaries can be easily identified because the corresponding element of the null vector is zero at the extremal values of the self-motion manifolds. It is important to point out that after a task point is reached, the intersection(s) of the bounding boxes enclosing the self-motion manifolds of

Fig. 1. Self-motion manifolds of tasks 1 and 2, along with the bounding boxes enclosing these manifolds, are shown. Note that if the robot is restricted to stay within the intersection of these bounding boxes, then neither task is reachable.
the remaining task points will change, so the joint space needs to be redivided accordingly.

A simple two-dimensional example with two task points is shown in Fig. 2 to illustrate the concept of regions. The blue and green curves are the self-motion manifolds of task points 1 and 2, respectively. In Fig. 2(a), the robot moves from the start configuration to task point 1, and one needs to determine whether the robot is still able to reach both task points 1 and 2 if an arbitrary failure occurs prior to reaching task point 1. The dashed lines are the $\theta_1$ limits on the intersection of the bounding boxes enclosing the self-motion manifolds of task points 1 and 2, and the dash-dotted lines are the $\theta_2$ limits. The entire joint space is divided into four regions by these limits. After task point 1 is reached, the robot moves toward task point 2, and so one only needs to determine the reachability of task point 2. Therefore, the joint space is redivided by the limits on the bounding box enclosing the self-motion manifold of task point 2, as shown in Fig. 2(b).

One now needs to identify the failure scenarios of these regions. For example, consider Fig. 2(a) where the reachability of both task points 1 and 2 needs to be considered. In region 1, if either joint 1 or joint 2 fails, task point 1 and task point 2 are no longer both reachable, because after a failure the robot can only move either vertically or horizontally in the configuration space. Thus for region 1 one can define two failure scenarios: A, which is when joint 1 fails in this region and B, which is when joint 2 fails in this region. In either scenario, it does not matter whether the other joint fails or not.

Now consider region 4 in Fig. 2(a), which is the intersection of the bounding boxes of the two self-motion manifolds. If joint 1 is locked in any position in this region, the robot is still able to reach both task points 1 and 2, unless joint 2 is also locked in the subsequent postfailure trajectory. This is also true for the analogous case, i.e., the robot is able to reach both task points 1 and 2 if joint 2 is locked in any position in this region, unless joint 1 is also locked in the subsequent postfailure trajectory. Therefore, the failure scenarios of region 4 in Fig. 2(a) are: C, which is when joint 1 fails in this region, and joint 2 also fails in the subsequent postfailure trajectory and D, which is when joint 2 fails in this region, and joint 1 also fails in the subsequent postfailure trajectory.

The failure scenarios of the regions in Fig. 2(b) can be identified in a similar way, where only the reachability of task point 2 needs to be considered. It is important to note that the failure scenarios will increase as the number of DOFs increases; however, the failure scenarios can be enumerated in an analogous manner.

Table I lists the failure scenarios of each region both before and after task point 1 has been reached. Obviously, in Fig. 2(a) region 4 has the lowest probability of task failure because it is the intersection of the bounding boxes of the two self-motion manifolds. This is also true in Fig. 2(b) because region 4′ is the bounding box of the self-motion manifold of task point 2. In contrast, the regions 1 and 1′, both before and after task point 1 has been reached, have the highest probability of task failure. Note that higher dimensional joint spaces can be divided and categorized in a similar manner, as long as the extremal values of the self-motion manifolds of each task point are calculated by checking if one of the elements of the null vector of the Jacobian is zero.

### C. Failure Probability Calculation

For an arbitrary joint trajectory, it may go through different regions with different failure scenarios. Therefore, to calculate its probability of task failure, the trajectory should be divided into pieces based on the regions it traverses. For example, the joint trajectory in Fig. 2(a) goes through regions 1 and 3 when the robot moves from the start configuration to task point 1. After task point 1 is reached, the joint space is redivided, and the joint trajectory in Fig. 2(b) goes through regions 3′ and 4′ when the robot moves toward task point 2. Therefore, the entire trajectory is divided into four pieces. Based on the failure scenarios of each region, the probability of task failure is calculated for each piece. The sum of the individual probabilities is the probability of task failure for the entire trajectory.

For the example shown in Table I, there are a total of four types of failure scenarios, and the probabilities of these four types are derived as follows. For failure scenario A, if joint one

<table>
<thead>
<tr>
<th>regions before task 1 reached</th>
<th>regions after task 1 reached</th>
<th>failure scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1^\text{R}$</td>
<td>A, B</td>
</tr>
<tr>
<td>2</td>
<td>$2^\text{R}$</td>
<td>B, C</td>
</tr>
<tr>
<td>3</td>
<td>$3^\text{R}$</td>
<td>A, D</td>
</tr>
<tr>
<td>4</td>
<td>$4^\text{R}$</td>
<td>C, D</td>
</tr>
</tbody>
</table>

Task failure scenarios:
A: Joint 1 fails.
B: Joint 2 fails.
C: Joint 1 fails and then joint 2 also fails.
D: Joint 2 fails and then joint 1 also fails.
fails in any position, then the task cannot be completed. Thus, the probability of task failure in scenario A, denoted $P_A$, is equal to the probability of joint 1 failing in this region, i.e.,

$$P_A = e^{-\lambda_1 t_0} - e^{-\lambda_1 (t_0 + \Delta t_r)}$$

(2)

where $\lambda_1$ is the failure rate\(^3\) of joint 1, $t_0$ is the time when the robot moves into this region, and $\Delta t_r$ is the time period spent in the region. Similarly, the probability of task failure in scenario B, denoted $P_B$, is

$$P_B = e^{-\lambda_2 t_0} - e^{-\lambda_2 (t_0 + \Delta t_r)}$$

(3)

where $\lambda_2$ is the failure rate of joint 2. For failure scenario C, the postfailure trajectory to reach the task points depends on the position where joint 1 is locked, i.e., the time period spent in the postfailure trajectory $\Delta t_p$ is a function of the time when joint 1 fails. Therefore, the probability of task failure in scenario C, denoted $P_C$, is

$$P_C = \int_{t_0}^{t_0+\Delta t_r} \lambda_1 e^{-\lambda_1 t} \times (e^{-\lambda_2 t} - e^{-\lambda_2 (t+\Delta t_p)}) \, dt$$

(4)

where the term $\lambda_1 e^{-\lambda_1 t}$ is the failure density function of joint 1 at time $t$, and the second factor is the failure probability of joint 2 failing in the postfailure trajectory when joint 1 fails at time $t$. Similarly, the probability of task failure in scenario D, denoted $P_D$, is

$$P_D = \int_{t_0}^{t_0+\Delta t_r} \lambda_2 e^{-\lambda_2 t} \times (e^{-\lambda_1 t} - e^{-\lambda_1 (t+\Delta t_p)}) \, dt$$

(5)

where $\lambda_1$ and $\lambda_2$ switch roles.

The probabilities of failure scenarios A and B can be directly calculated using (2) and (3) because the value of $\Delta t_r$ can be computed. However, the value of $\Delta t_p$ depends on when the failure occurs. As a result, to estimate the probability of task failure in scenarios C or D, the entire trajectory is divided into small segments where all postfailure trajectories that occur in a segment are considered approximately the same. If one denotes the probability of failure in the $i$th small segment of a trajectory in scenario C as $\Delta P_{Ci}$, then (4) becomes

$$P_C = \sum_{i=1}^{n} \Delta P_{Ci}$$

(6)

where $n$ is the number of segments in the trajectory through the region with failure scenario C. Similarly, (5) becomes

$$P_D = \sum_{i=1}^{n} \Delta P_{Di}.$$  

(7)

Fig. 3 shows examples of estimating the probabilities of $\Delta P_{Ci}$ and $\Delta P_{Di}$, when the failure occurs in a segment located at the tip of the solid arrow. In Fig. 3(a) and (b), which illustrate scenarios C and D, respectively, the failures occur before reaching task point 1, and the resulting postfailure trajectories are shown in dashed lines. Thus, $\Delta P_{Ci}$ is approximately equal to the probability of joint 1 failing in this segment times the probability of joint 2 failing in the postfailure trajectory, i.e.,

$$\Delta P_{Ci} = (e^{-\lambda_1 t_0} - e^{-\lambda_1 (t_0 + \Delta t_s)}) (e^{-\lambda_2 (t_0 + \Delta t_s)} - e^{-\lambda_2 (t_0 + \Delta t_s + \Delta t_p)})$$

(8)

where $t_0$ is the time when the robot moves into this segment, $\Delta t_s$ is the time period spent in this segment, and $\Delta t_p$ is the time period spent in the post failure trajectory. Similarly, $\Delta P_{Di}$ is approximately equal to the probability of joint 2 failing in this segment times the probability of joint 1 failing in the postfailure trajectory, i.e.,

$$\Delta P_{Di} = (e^{-\lambda_2 t_0} - e^{-\lambda_2 (t_0 + \Delta t_s)}) (e^{-\lambda_1 (t_0 + \Delta t_s)} - e^{-\lambda_1 (t_0 + \Delta t_s + \Delta t_p)}).$$

(9)

Fig. 3(c) shows the case when the failure occurs after reaching task point 1 so that the computation of $\Delta t_p$ only includes the time to reach task point 2, which is computed from $\Delta \theta_{2,post}$ or $\Delta \theta_{1,post}$, for scenarios C or D, respectively.

It is assumed that the maximum velocity of all joints, which is determined based on the task requirements of accuracy or safety, are approximately the same, denoted by $\dot{\theta}_{\text{max}}$, and each joint rotates at a constant velocity $\dot{\theta}_i \leq \dot{\theta}_{\text{max}}$ within a region. Therefore, considering that each joint moves independently, the shortest time needed to finish a piece of an arbitrary given joint trajectory is equal to the largest rotation angle among all the joints divided by the maximum velocity, i.e.,

$$\Delta t = \max(\Delta \theta_1, \Delta \theta_2, \ldots, \Delta \theta_n) \cdot \frac{1}{\dot{\theta}_{\text{max}}}.$$  

(10)

III. DETERMINING THE TRAJECTORY WITH MAXIMUM TASK COMPLETION PROBABILITY

A. Overview of the Trajectory Planning Algorithm

The above method for estimating the probability of task failure for an arbitrary joint trajectory is now used to develop a novel motion planning algorithm to determine a joint trajectory with maximum probability of task completion for a given set of task points. Note that this article focuses on the problem of joint trajectory planning for point-to-point tasks so that the exact end-effector trajectory is not important, and the robot only needs to reach the task points accurately. To simplify this problem, without loss of generality, one can neglect the probability of two (or more) joints failing simultaneously because it is significantly lower than the probability of only one joint failing. Based on this

---

\(^3\)Joint failure rates can be obtained by employing the reliability analysis described in [19] using the component reliability data available in [20].
approximation, when calculating the probability of task failure, the joint space division in Section II-B is still the same, but the failure probability of scenarios C and D are considered to be zero. This approximation does not appreciably change the optimal trajectory that is computed.

The algorithm starts by checking the two conditions that guarantee a fault tolerant trajectory for all task points: 1) Does there exist an intersection among the bounding boxes enclosing the self-motion manifolds for all task points, and 2) if so, does there also exist at least one configuration for the first task point within this intersection. The first condition ensures that as long as the robot moves inside the intersection, the robot can always reach all of the task points after a failure, regardless of which joint fails. The second condition ensures that the robot is able to reach the first task point while staying within the intersection. Therefore, if both conditions are satisfied for all the task points, then the problem can be simplified to planning an optimal trajectory from the start configuration to the closest configuration of the closest intersection of the bounding boxes. After reaching the intersection, the robot can choose an arbitrary trajectory inside the intersection to reach all the tasks with a smaller velocity for greater accuracy, because the task failure probability inside the intersection is zero.

If either of the conditions is false, the self-motion manifold associated with task point 1 is discretized, and then the optimal trajectories from the start configuration to each configuration along the self-motion manifold must be determined. After task point 1 is reached, it is deleted from the list of task points, and the above two conditions are reevaluated for the remaining task points. This procedure is repeated for every task point, until all tasks are completed. Note that for the last task point, the bounding box(es) enclosing its self-motion manifold(s) always satisfy the above two conditions. Finally, the failure probability of each potential trajectory is calculated by applying the method developed in Section II and the optimal trajectory can be obtained by choosing the trajectory with the smallest failure probability.

This process is illustrated in Fig. 4 for these two cases with two task points. In Fig. 4(a), the self-motion manifolds of task points 1 and 2 satisfy the two conditions, so the optimal trajectory is the optimal trajectory from the start configuration to the intersection of the bounding boxes. In Fig. 4(b), there is no intersection and so condition (a) is violated. Therefore, one must first determine the optimal trajectories from the start configuration to each discretized configuration of task 1’s self-motion manifold. After task 1 is reached, task point 2 satisfies both conditions so that one must only identify an optimal trajectory to the bounding box of the self-motion manifold of task 2. Two example potential trajectories are shown in red and purple. The failure probability of each potential trajectory can be calculated by the method developed in Section II to determine the optimal trajectory.

As shown in Section II, the failure probability of a trajectory depends on the failure scenarios of the regions that it goes through and the time spent in those regions. To minimize the failure probability, the optimal trajectory should try to minimize the following two criteria. The first criterion is minimizing the time needed to finish the trajectory, i.e., the joint with maximum \( \Delta \theta_i \) should always move at the maximum velocity based on (10). The other criterion is minimizing the failure scenarios associated with each piece of the trajectory. Therefore the robot should try to reach the boundaries of regions with fewer failure scenarios as soon as possible. In the following two subsections, an algorithm to plan the optimal trajectory from a configuration to an intersection of bounding boxes and an algorithm to plan the optimal trajectory from one configuration to another will be introduced based on these two criteria.

### B. Optimal Trajectory From a Configuration to an Intersection

Define \( [\bar{\theta}_i, \bar{\theta}_i] \) as a range of values of joint \( i \), denoted \( \theta_i \), for all \( \theta \) that are in the intersection of the self-motion manifold bounding boxes that is closest to the given configuration. To minimize the two criteria in the above subsection, every joint \( \theta_i \), whose value is outside the \( [\bar{\theta}_i, \bar{\theta}_i] \) range should move at the maximum velocity \( \dot{\theta}_i \) until it’s value is within its \( [\bar{\theta}_i, \bar{\theta}_i] \) range. Once \( \theta_i \) reaches its boundary, then it stops moving. Therefore, the velocity of each joint can be determined by

\[
\dot{\theta}_i = \begin{cases} 
0 & \theta_i \leq \bar{\theta}_i \\
\dot{\theta}_{\text{max}} & \text{otherwise.}
\end{cases}
\]

The above process stops when the robot configuration \( \theta \) is within the intersection of the bounding boxes, i.e., all \( \dot{\theta}_i = 0 \). Fig. 4(a) shows a simple example, where joints 1 and 2 are both outside of their ranges \( [\bar{\theta}_1, \bar{\theta}_1] \) and \( [\bar{\theta}_2, \bar{\theta}_2] \) at the start configuration, and they both move at \( \dot{\theta}_{\text{max}} \) toward the intersection of the bounding boxes. Joint 2 is the first to reach its boundary, where it stops, and so joint 1 moves along the \( \theta_2 \) boundary to reach the vertex of the intersection.

### C. Optimal Trajectory Between Two Configurations

In this subsection, the algorithm to plan the optimal trajectory between two configurations, denoted \( \theta \) and \( \theta' \), will be introduced. If it exists, let \( [\bar{\theta}_i, \bar{\theta}_i] \) be the range of values of joint \( i \) that overlap all the self-motion manifold bounding boxes for all the remaining task points. For example, in Fig. 4(b) \( [\bar{\theta}_2, \bar{\theta}_2] \) does not exist because there is no overlap between the bounding
The third case C is that there exists at least one \([\theta^i_{\text{b}}, \theta^i_{\text{o}}]\) between \(\theta_i\) and \(\theta'_i\), and one of them is inside one of the ranges, whereas the other is outside this range. The configuration outside the range, denoted by \(\theta'^i_{\text{b}}\), first moves to the range boundary \(\theta'^i_{\text{o}}\) at \(\theta^i_{\text{max}}\), which takes time \(t_1 = |\theta'^i_{\text{b}} - \theta'^i_{\text{o}}|/\theta^i_{\text{max}}\). Then, \(\theta_i\) and \(\theta'_i\) are connected by moving at a constant velocity, which is given by

\[
\dot{\theta}^C_i = \frac{|\theta'_i - \theta_i| - |\theta'^i_{\text{b}} - \theta'_i|}{t_T - t_1 - t_2}.
\]

This case is illustrated in Fig. 5, where the joint with maximum difference between configurations 3 and 4 is joint 1, so it always moves at \(\theta^i_{\text{max}}\). For joint 2, configuration 4 first moves at \(\theta^i_{\text{max}}\) to reach the range boundary as soon as possible, as shown by the red arrow. Then they are connected by moving at the constant velocity \(\dot{\theta}^C_i\) within this range, as shown by the orange line.

The last case D is that there exists at least one \([\theta^i_{\text{b}}, \theta^i_{\text{o}}]\) in joint \(i\), but none of these ranges are between \(\theta_i\) and \(\theta'_i\). In this case there are two potential trajectories that should be checked. One is connecting \(\theta_i\) and \(\theta'_i\) with a straight line to ensure the shortest distance, and the other is both \(\theta_i\) and \(\theta'_i\) move to the same closest range boundary \(\theta^i_{\text{b}}\) to reduce the failure scenarios. If \(\theta^i_{\text{b}} - \theta_i\) is less than \(\Delta\theta^i_{\text{max}}\), then the latter trajectory is optimal, where \(\theta_i\) and \(\theta'_i\) both move at \(\theta^i_{\text{max}}\) to reach \(\theta^i_{\text{b}}\), and then stop at this boundary, as shown by the trajectory connecting configurations 5 and 6 in Fig. 5. In practice, joint \(i\) should move a small amount further to account for any error in computing the bounding box, so that this joint is guaranteed to be inside the range \([\theta^i_{\text{b}}, \theta^i_{\text{o}}]\).

Otherwise, the straight line path is optimal, where \(\theta_i\) and \(\theta'_i\) are connected by moving at the constant velocity \(\dot{\theta}^A_i\), as shown by the trajectory connecting configurations 7 and 8 in Fig. 5. The pseudocode of this algorithm is given in Algorithm 1.

IV. RESULTS

The above algorithm for computing the trajectory with the maximum probability of task completion is illustrated for a planar 3R positioning robot with equal link lengths, an optimally fault tolerant spatial 4R positioning robot design from [21], and the Kinova Gen3 spatial 7R positioning and orienting robot. It is assumed that all joints have the same nonzero failure rate. The algorithm was evaluated on two hundred scenarios that consist of randomly selected start configurations, task point 1 locations, and task point 2 locations that are uniformly distributed inside the robot’s workspace. As a comparison, for each scenario the failure probabilities of all the shortest configuration space trajectories from the start configuration, going through a self-motion manifold configuration of task 1, and ending at the bounding box of task 2 are also computed, and the one with the maximum probability of task completion, computed by using the method in Section II, is selected. For simplicity we will refer to this comparison trajectory as the “shortest distance” trajectory. Based on the resulting failure probabilities, the increase in the task completion probability for each robot when using the proposed algorithm is shown in the violin plot in Fig. 6, where the colored horizontal line is the mean value, the white dot is the median value, and the gray vertical lines indicate the 25th and 75th percentiles.
Fig. 6. To compare the optimal trajectory with the shortest distance trajectory, two hundred sets of start configurations and task points are randomly sampled, and the increase in the task completion probability for different robots is shown.

Algorithm 1: Planning the Optimal Trajectory From One Configuration to Another.

1: for $i = 1 : n$ do
2:   if $\Delta\dot{\theta}_i = \Delta\dot{\theta}_{\text{max}}$ then
3:     $\dot{\theta}_i = \dot{\theta}_{\text{max}}$, $0 \leq t \leq t_T$
4:   else if $\theta_i$ and $\theta_i'$ belong to case A then
5:     $\dot{\theta}_i = \dot{\theta}_i^A$, $0 \leq t \leq t_T$
6:   else if $\theta_i$ and $\theta_i'$ belong to case B then
7:     $\dot{\theta}_i = \begin{cases} 
\dot{\theta}_{\text{max}} & 0 \leq t \leq t_1 \\
\dot{\theta}_i^B & t_1 \leq t \leq t_T - t_2 \\
\dot{\theta}_{\text{max}} & t_T - t_2 \leq t \leq t_T 
\end{cases}$
8:   else if $\theta_i$ and $\theta_i'$ belong to case C then
9:     $\dot{\theta}_i = \begin{cases} 
\dot{\theta}_{\text{max}} & 0 \leq t \leq t_1 \\
\dot{\theta}_i^C & t_1 \leq t \leq t_T 
\end{cases}$
10: else
11:   if $|\theta_i^p - \theta_i| + |\theta_i' - \theta_i'| < \Delta\theta_{\text{max}}$ then
12:     $\dot{\theta}_i = \begin{cases} 
\dot{\theta}_{\text{max}} & 0 \leq t \leq t_1 \\
0 & t_1 \leq t \leq t_T - t_2 \\
\dot{\theta}_{\text{max}} & t_T - t_2 \leq t \leq t_T 
\end{cases}$
13:   else
14:     $\dot{\theta}_i = \dot{\theta}_i^A$, $0 \leq t \leq t_T$
15: end if
16: end if
17: end for

Fig. 7. For a planar 3R robot, the optimal trajectory with maximum probability of task completion shown in red and the shortest distance trajectory shown in pink are projected into the $\theta_1$, $\theta_2$ plane in (a), $\theta_1$, $\theta_3$ plane in (b), and $\theta_2$, $\theta_3$ plane in (c). Their corresponding end-effector trajectories are shown in (d), where the blue and green dots are task points 1 and 2, respectively.
optimal trajectory always has one joint moving at maximum velocity to minimize the time needed to reach the task points. In addition, other joints try to reach the intersection of the bounding boxes of the self-motion manifolds, using the shortest distance trajectory, to minimize the failure scenarios that will prevent task completion. Because the goal is to maximize the task completion probability, the end-effector trajectory in the workspace can be quite unintuitive. Thus, it would be difficult for a human being to program this trajectory based purely on their intuition.

To further analyze the characteristics of the optimal trajectories generated by this algorithm, one can compute the probability of task completion for failures in each joint. Fig. 10 illustrates this for the Kinova Gen3 spatial 7R positioning and orienting robot, where the percentage of task completion is shown for each joint failure individually. These percentages were estimated by taking twenty equally timed samples along each trajectory and are shown in blue and orange bars for the optimal trajectory and the shortest distance trajectory, respectively. As can be seen in Fig. 9, joints 2, 5, and 6 move to the overlap of the self-motion manifold bounding boxes as soon as possible in the optimal trajectory, while these joints spend much more time outside the overlap of the bounding boxes in the shortest distance trajectory. Therefore, the percentage of task completion of these joints is much higher for the optimal trajectory as compared to the shortest distance trajectory. It is well known that robots that emulate the kinematic structure of the human arm are intolerant to failures in the elbow joint [22]. This is because it is the only joint that can change the distance from the shoulder to the wrist. Therefore, the percentage of task completion for a failure in joint 4 will be zero for any trajectory. This is because the probability of the start configuration, task 1, and task 2 being exactly at the same distance from the shoulder is zero. In contrast, human-like robot arms are completely tolerant to failures in joint 7 because its self-motion manifold range is $2\pi$, so that any trajectory will have 100% task completion probability for a failure in this joint.

To further analyze the difference between the trajectories computed by the proposed algorithm as compared to the shortest distance trajectories, one can compute the number of A, B, C, and D cases when determining the potential optimal trajectory from the start configuration to each discretized configuration along the self-motion manifolds of task point 1. Fig. 11 illustrates this for the Kinova Gen3 robot trajectory shown in Fig. 9, where the number of A, B, C, and D cases is shown for each joint individually. Note that joints 4 and 7 consist of only case A so that the optimal trajectory is the same as the shortest distance...
Fig. 11. Number of each case, A, B, C, and D, when computing the potential optimal trajectory from the start configuration to each discretized configuration along the self-motion manifolds of task point 1 are shown for the spatial 7R robot trajectory given in Fig. 9.

trajectory. For joint 4, this is because there is no bounding box and for joint 7 it is because the bounding box spans the entire joint range $[-\pi, \pi]$, as discussed above. In contrast, no case As occur for the other joints, which indicates that the probability of task completion was improved by applying the proposed method for these joints.

To illustrate the change in trajectories due to a joint failure, Fig. 12 shows two simulations, where joint 2 is locked at the same time (in terms of the percent of trajectory completed) for both the optimal trajectory [Fig. 12(a)] and the shortest distance trajectory [Fig. 12(b)]. As can be seen in Fig. 9(b), joint 2 is always inside the range of the bounding box enclosing the self-motion manifolds of task 1 for both trajectories, therefore task 1 is still reachable after joint 2 is locked, as shown in Fig. 12(c) and Fig. 12(d). However, the difference is in how the optimal trajectory has configured the robot by having joint 2 move to the overlap of the bounding boxes of task 1 and task 2 as soon as possible, so task 2 is still reachable after the failure, as shown in Fig. 12(e). In contrast, joint 2 keeps moving outside the range of the bounding boxes enclosing the self-motion manifold of task 2 in the shortest distance trajectory, so that the robot is not able to reach task 2 after the failure occurs, as shown in Fig. 12(f).

It can be clearly seen that the robot is at the boundary of its workspace, where the robot achieves the target orientation but it is no longer able to get closer to the target position.4

V. CONCLUSION

This work presents an algorithm to compute the trajectory with the maximum probability of completing a set of point-to-point tasks after a failure that results in a locked joint. It is shown how task failure probability can be computed by dividing the joint space into different regions based on the extremal points of the self-motion manifolds, and the failure scenarios in these regions. This method is then used to develop a novel trajectory planning algorithm to maximize the probability of task completion. The effectiveness of this algorithm, as compared to a shortest distance trajectory, is validated for planar positioning 3R, spatial positioning 4R, and spatial positioning and orienting 7R redundant robots. The results show that to maximize the probability of task completion, the robot must select trajectories that balance the tradeoff between trajectories with minimum time and those that contain a smaller number of failures scenarios. It is also shown that as the number of DOFs increases, the problem of identifying an optimal trajectory becomes more complicated and the benefits of the proposed algorithm, as opposed to a straightforward shortest distance trajectory, become more pronounced. Finally, because the computation of the proposed algorithm only grows linearly with the number of DOFs, determining optimally failure-tolerant trajectories for very high-DOF systems is computationally feasible.

However, there are definitely aspects of the algorithm that can be improved. In particular, one currently has to evaluate all of the configurations on the self-motion manifolds of each task point, except for the last one. If one could identify a condition for determining the optimal configuration on a manifold, this would greatly decrease the computational expense. In addition, this motion planning algorithm can be further extended to avoid obstacles. The current algorithm identifies a family of optimal trajectories in the configuration space. If workspace obstacles are mapped to the configuration space, then one can identify a specific trajectory from within this family to avoid the obstacle. This can be done by adjusting the joint velocities of all joints except the joint that must move at maximum velocity. We will explore these improvements in future work.

4The simulation videos are available on YouTube at https://youtu.be/5xQjwQyS_c
REFERENCES


Biyun Xie (Member, IEEE) received the B.S. and the Ph.D. degrees in mechanical engineering from the Beijing University of Technology, Beijing, China, in 2009 and 2015, respectively. She received the Ph.D. degree in electrical engineering from Colorado State University, Fort Collins, CO, USA, in 2019.

She is currently an Assistant Professor of Electrical and Computer Engineering with the University of Kentucky, Lexington, KY, USA. Her research interests include redundant robots and human–robot interaction.

Anthony A. Maciejewski (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Ohio State University, Columbus, OH, USA, in 1982, 1984, and 1987, respectively.

From 1988 to 2001, he was a Professor of Electrical and Computer Engineering with Purdue University, West Lafayette, IN, USA. He is currently a Professor and Department Head of Electrical and Computer Engineering with Colorado State University, Fort Collins, CO, USA. His research interests include robotics, high-performance computing, and STEM education.