Failure-Tolerant Path Planning for Kinematically Redundant Manipulators Anticipating Locked-Joint Failures

Rodrigo S. Jamisola, Jr., *Student Member, IEEE*, Anthony A. Maciejewski, *Fellow, IEEE*, and Rodney G. Roberts, *Senior Member, IEEE*

***Abstract*—This work considers kinematic failure tolerance when obstacles are present in the environment. It addresses the issue of finding a collision-free path such that a redundant robot can successfully move from a start to a goal position and/or orientation in the workspace despite any single locked-joint failure at any time. An algorithm is presented that searches for a simply-connected, obstacle-free surface with no internal local minimum or maximum in the configuration space that guarantees the existence of a solution. The method discussed is based on the following assumptions: a robot is redundant relative to its task, only a single locked-joint failure occurs at any given time, the robot is capable of detecting a joint failure and immediately locks the failed joint, and the environment is static and known. The technique is illustrated on a seven degree-of-freedom commercially available redundant robot. Although developed and illustrated for a single degree of redundancy, it is possible to extend the algorithm to higher degrees of redundancy.**

***Index Terms*—Kinematic failure tolerance, locked-joint failures, path planning, redundant manipulators, self-motion manifolds.**

I. Introduction

FAILURE-TOLERANT path planning is a motion planning strategy that gives a robot the ability to gracefully accommodate joint failures. This ability enhances safety and reliability for robots carrying out tasks in remote or hazardous environments, such as in space exploration [1], [2], underwater exploration [3], and nuclear waste disposal [4]–[6]. It allows a robot to immediately complete the task at hand without unnecessary delays due to robot repair. It also avoids the potentially significant danger associated with a robot failure during task execution amongst hazardous materials.

A number of studies have been dedicated to the assessment [6], [7] and analysis [5], [8]–[12] of robot safety and reliability, including robots designed primarily for this purpose [13]–[15]. The earliest work on kinematic failure tolerance [16] used the minimum singular value of the manipulator Jacobian matrix as a local worst-case measure of a robot's tolerance to a joint failure. The nature of joint failures that have been studied include locked-joint [17]–[19] and free-swinging joint failures [20]. A real-time implementation of local kinematic failure tolerance has been demonstrated in [19]. Other studies related to enhancing a robot's tolerance to failure include work on failure detection [21]–[23], low-level failure avoidance and recoverability [24], [25], layered failure tolerance control [26]–[28], failure tolerance by trajectory planning [29], and kinematic failure recovery [30]. In all of these previous studies on kinematic failure tolerance cited above, workspace obstacles were not considered.

Despite the lack of attention that it has received, the presence of obstacles in the environment greatly affects a kinematic failure tolerance algorithm. To our knowledge, the only work to consider obstacles along with failure tolerance was [31]. Their approach allows one to guarantee that a manipulator can avoid obstacles while the end-effector follows a desired workspace path, even in the presence of an arbitrary single locked-joint failure that occurs at any point along the trajectory. Our work is similar in that it allows a manipulator to avoid obstacles in the presence of joint failures, however, it differs in that the task definition is specified as a desired start and goal location in the workspace, as is typical of pick-and-place tasks. This is significant because constraining the end-effector to a specific path will greatly reduce the likelihood that an obstacle-free, failure-tolerant trajectory exists.

The remainder of this paper is organized as follows. An overview of the proposed approach is presented in Section II. Section III states the conditions that guarantee the existence of a solution to the failure-tolerant path-planning problem and then uses these conditions to form the basis of an algorithm. Section IV illustrates the efficacy of the algorithm on a three degree-of-freedom planar robot. A seven degree-of-freedom example is presented in Section V using the Mitsubishi PA-10 robot. Finally, the summary and conclusions of this work are given in Section VI.

II. Overview of the Proposed Approach

Consider a kinematically redundant manipulator, operating in a failure-prone environment, that must complete a pick-and-place type task in a workspace filled with obstacles. The goal of our work is to generate a family of joint trajectories, each of which would allow the manipulator to avoid obstacles and tolerate any single locked-joint failure as its end effector travels from the given start workspace location to the desired goal location. To achieve this goal, two fundamental issues need to be

Manuscript received December 13, 2005. This paper was recommended for publication by Associate Editor I. Bonev and Editor K. Lynch upon evaluation of the reviewers' comments.

R. S. Jamisola, Jr., and A. A. Maciejewski are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523-1373 USA (e-mail: r.jamisola@colostate.edu; aam@colostate.edu).

R. G. Roberts is with the Department of Electrical and Computer Engineering, Florida A&M—Florida State University, Tallahassee, FL 32310-6046 USA (e-mail: rroberts@eng.fsu.edu).

Digital Object Identifier 10.1109/TRO.2006.878959

resolved. First, the manipulator must be restricted to a region of the configuration space (C-space) that is obstacle free. Second, while in this region, one must guarantee that the goal workspace location is reachable even after any joint failure. Our approach is to precompute a surface in the C-space that is obstacle free and guarantees that the goal remains reachable. The characterization of the properties of such a surface is a key contribution of this work.

The requirement of failure tolerance dictates the boundaries of where the surface must be contained so that the goal remains reachable. This is equivalent to determining the range of the joint values for being at the goal workspace location. Thus, the first step is to make sure that the starting joint configuration falls within this range. This is the same type of test used in [31] and has also been applied to determine a manipulator's failure-tolerant workspace [18].

Staying within these boundaries is not sufficient for guaranteeing that the workspace goal is reachable. However, it is possible to guarantee reachability if the surface has certain properties, namely that it contains no internal local minimum or maximum in terms of the individual joint variables. If a surface with these properties is also obstacle free, then operating on this surface guarantees failure tolerance. Thus, the crux of our approach is to identify such a surface.

The remainder of this section will define the terms that are used in this work. Let n denote the number of degrees of freedom (DOFs) of a robot and let m denote the number of DOFs of a robot's workspace. For a kinematically redundant robot $(n > m)$, the degree of redundancy is $r = n - m$. Its given end-effector position and/or orientation, denoted $\mathbf{x}$, generally corresponds to an infinite number of configurations in the C-space. The set of configurations in C-space that result in the same end-effector workspace location $\mathbf{x}$ is called the pre-image of $\mathbf{x}$. The pre-image can be written as a union of disjoint connected sets

$$\mathbf{f}^{-1}(\mathbf{x}) = \bigcup_{i=1}^{n_m} \boldsymbol{\mathcal{M}}_i \tag{1}$$

where $\boldsymbol{\mathcal{M}}_i$ is the ith r-dimensional self-motion manifold in the inverse kinematic pre-image such that $\boldsymbol{\mathcal{M}}_i \cap \boldsymbol{\mathcal{M}}_j = \emptyset$ when $i \neq j$, and n_m is the number of self-motion manifolds provided that the pre-image contains no singular configurations [32]. When the pre-image of $\mathbf{x}$ contains both singular and non-singular configurations it results in manifolds that meet at a singular configuration [33].

A start self-motion manifold, denoted $\boldsymbol{\mathcal{M}}_s$, corresponds to a start position and/or orientation, denoted $\mathbf{x}_s$, while a goal self-motion manifold, denoted $\boldsymbol{\mathcal{M}}_g$, corresponds to a goal position and/or orientation, denoted $\mathbf{x}_g$. Fig. 1 shows a pair of single dimensional start and goal self-motion manifolds for a robot with $r = 1$. The dark portions of the self-motion manifolds denote configurations of the robot that are outside the joint limits or are in contact with obstacles. A continuous, obstacle-free portion of the goal self-motion manifold is denoted $\boldsymbol{\gamma}_g$, while an obstacle-free start configuration is denoted $\boldsymbol{\theta}_s$. A point on $\boldsymbol{\gamma}_g$ is denoted $\boldsymbol{\theta}_g$.

When a robot suffers from a locked-joint failure, its feasible motion in the C-space is reduced. In particular, a single locked-joint failure restricts the reachable robot configurations in the C-space to an $(n-1)$-dimensional hyperplane. For example, if a failure occurs in the ith joint at the configuration $\boldsymbol{\theta}_s$, its ith component remains fixed so that the resulting reduced C-space becomes

$$\mathbf{H}_i(\boldsymbol{\theta}_s) = \{\boldsymbol{\theta} | \theta_i = \theta_{si}\} \tag{2}$$

where θ_i denotes the ith component of $\boldsymbol{\theta}$ and θ_{si} denotes the ith component of $\boldsymbol{\theta}_s$. We will call $\mathbf{H}_i(\boldsymbol{\theta}_s)$ the *failure hyperplane* associated with $\boldsymbol{\theta}_s$. Fig. 1 shows $\boldsymbol{\theta}_s$ with its corresponding failure planes $\mathbf{H}_1(\boldsymbol{\theta}_s)$, $\mathbf{H}_2(\boldsymbol{\theta}_s)$, and $\mathbf{H}_3(\boldsymbol{\theta}_s)$.

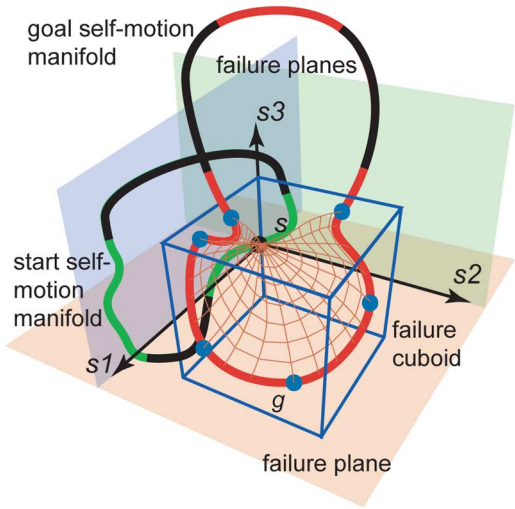


Fig. 1. Configuration space for a single degree-of-redundancy robot shown with a start and a goal self-motion manifold. All the failure planes corresponding to an obstacle-free start configuration, $\boldsymbol{\theta}_s$, intersect a continuous, obstacle-free portion of the goal self-motion manifold, γ_g. The failure cuboid contains $\boldsymbol{\theta}_s$ and $\boldsymbol{\gamma}_g(v)$. The failure surface corresponding to $\boldsymbol{\theta}_s$, shown as a web-like network of paths, is identified by connecting $\boldsymbol{\theta}_s$ to points on $\boldsymbol{\gamma}_g(v)$ via monotonic paths within the failure cuboid. Each node along $\boldsymbol{\gamma}_g(v)$ defines an intersection with a face of the failure cuboid. (Color version available online at: http://ieeexplore.ieee.org.)

Given a desired workspace goal $\mathbf{x}_g$, a necessary condition will be developed to determine if that goal can be reached from $\boldsymbol{\theta}_s$ in spite of any locked-joint failure. This condition will identify a portion of the $\boldsymbol{\gamma}_g$ curve, parameterized here by the function $\boldsymbol{\gamma}_g(v) : [0,1] \rightarrow \mathbb{R}^n$. The extremal points of $\boldsymbol{\gamma}_g(v)$ define the bounds of a hypervolume, $\mathbf{V}$, in C-space called a *failure hypercuboid*. This failure hypercuboid has the form

$$\mathbf{V} = \left\{ \boldsymbol{\theta} \middle| \min_{0 \leq v \leq 1} \gamma_{gi}(v) \leq \theta_i \leq \max_{0 \leq v \leq 1} \gamma_{gi}(v), i = 1, \ldots, n \right\}. \tag{3}$$

This determines a failure-tolerant workspace corresponding to $\boldsymbol{\theta}_s$ and $\boldsymbol{\gamma}_g(v)$. In this case $\boldsymbol{\theta}_s$ is called a feasible start configuration. The failure-tolerant workspace discussed in this work is a subset of the one discussed in [18], which is a function of $\mathbf{x}_s$ and $\mathbf{x}_g$.

Because there are obstacles in the workspace, there is a need to identify a suitable region in the failure hypercuboid that would allow the robot to reach the goal while avoiding obstacles. Furthermore, this region must have the property that the robot can still reach its goal in spite of a single locked-joint

failure. This will be accomplished by choosing a surface connecting a suitable $\boldsymbol{\theta}_s$ to a $\boldsymbol{\gamma}_g$. This surface is chosen so that the manipulator will not encounter obstacles and can remain on the surface even if a joint is locked. The conditions for the existence of such a surface, called a *failure surface*, $\boldsymbol{\mathcal{S}}$, will be derived in the following section.

III. Guaranteeing a Failure-Tolerant Path

A. A Necessary Condition and a Sufficient Condition

In the previous section, we described an approach to guarantee failure tolerance in the presence of obstacles using a surface. In this section we will derive conditions concerning the existence of such a surface. The first condition is a necessary condition to determine the suitability of a start configuration, $\boldsymbol{\theta}_s$, based on the ability of the manipulator to reach the $\boldsymbol{\gamma}_g$ curve with any single joint locked. While $\boldsymbol{\theta}_s$ and $\boldsymbol{\gamma}_g$ are obstacle free, this condition does not take into account the possibility of encountering obstacles on the way, and is thus only a necessary condition. We will see later that this tends to be a strong indicator of the likelihood that such a surface exists. Fortunately it turns out that this is a relatively fast calculation that can easily eliminate cases where no surface exists. The second condition is used to determine the existence of the solution once a feasible $\boldsymbol{\theta}_s$ is identified. The proof of this result will motivate an algorithm for calculating a suitable surface. Such a surface cannot have a local minimum or maximum in any of its θ_i components.

The necessary condition is given in terms of the failure hyperplanes associated with $\boldsymbol{\theta}_s$. Given $\mathbf{x}_s$ and $\mathbf{x}_g$, the manipulator can reach $\boldsymbol{\gamma}_g$ when joint i is locked at $\boldsymbol{\theta}_s$ provided that the corresponding failure hyperplane of $\boldsymbol{\theta}_s$ intersects $\boldsymbol{\gamma}_g$, that is, $\mathbf{H}_i(\boldsymbol{\theta}_s) \cap \boldsymbol{\gamma}_g \neq \emptyset$. Physically this means that the resulting failure-induced C-space due to a locked joint i failure at $\boldsymbol{\theta}_s$ contains at least a point of $\boldsymbol{\gamma}_g$. This indicates the possibility of reaching $\mathbf{x}_g$ despite a locked-joint failure at $\boldsymbol{\theta}_s$. Thus, to guarantee that the robot will reach $\boldsymbol{\gamma}_g$ from $\boldsymbol{\theta}_s$ for any locked-joint failure, it is necessary that each of the n failure hyperplanes, $\mathbf{H}_i(\boldsymbol{\theta}_s)$, intersects the $\boldsymbol{\gamma}_g$ curve. This motivates the definition of a feasible start configuration $\boldsymbol{\theta}_s$ when all of its corresponding failure hyperplanes intersect $\boldsymbol{\gamma}_g$. This proposition is formally stated as a necessary condition.

Proposition 1 (Necessary Condition): A necessary condition for a given obstacle-free start configuration, $\boldsymbol{\theta}_s$, to be a feasible start configuration is that all of the corresponding failure hyperplanes of $\boldsymbol{\theta}_s$ intersect a continuous, obstacle-free portion of the goal self-motion manifold, $\boldsymbol{\gamma}_g$, that is

$$\mathbf{H}_i(\boldsymbol{\theta}_s) \cap \boldsymbol{\gamma}_g \neq \emptyset \quad \text{for all } i = 1, \ldots, n. \tag{4}$$

After a feasible $\boldsymbol{\theta}_s$ has been identified, its corresponding $\boldsymbol{\gamma}_g(v)$ defines the bounds of a failure hypercuboid, $\mathbf{V}$. If there were no obstacles in the workspace, then the identification of $\mathbf{V}$ is sufficient to guarantee that a given robot can successfully reach $\mathbf{x}_g$ from $\boldsymbol{\theta}_s$ for any single locked-joint failure at any time by keeping its configurations within $\mathbf{V}$ as the given robot approaches $\mathbf{x}_g$. When obstacles are present in the workspace, some regions of $\mathbf{V}$ may no longer be available. The problem now becomes that of searching for a solution set within $\mathbf{V}$ that can guarantee successful completion. This will be accomplished by generating an obstacle-free surface connecting $\boldsymbol{\theta}_s$ to the $\boldsymbol{\gamma}_g$ curve in such a way that the manipulator can continue to move along the surface to $\boldsymbol{\gamma}_g$ in spite of any single locked-joint failure. When a joint is locked the manipulator is constrained to be on a corresponding hyperplane. Therefore, the intersection of that hyperplane with the surface needs to connect with $\boldsymbol{\gamma}_g$. The surface must satisfy a monotonicity property in the following sense. Consider locking joint i. As the value at which joint i is locked varies, this determines a contour-like plot in which θ_i is analogous to the height. The different contours represent the manipulator's motion along the surface with its ith joint locked at different fixed values. In order for a manipulator to be able to reach its goal for any locked-joint failure, no closed contours can exist. Therefore, the surface cannot have any local internal minimum or maximum with respect to any θ_i. This monotonicity condition is captured by the following theorem.

Theorem 1 (Sufficient Condition): Let $\boldsymbol{\mathcal{S}}$ be an obstacle-free, two-dimensional, simply-connected surface in $\mathbb{R}^n$. Suppose $\boldsymbol{\mathcal{S}} = \{\mathbf{S}(u,v) | u, v \in [0,1]\}$ is the image of a continuous function $\mathbf{S} : [0,1] \times [0,1] \to \mathbb{R}^n$ that is continuously differentiable in its interior and that $\mathbf{S}(u,v)$ has the following properties:

i) $\mathbf{S}(0,v) = \boldsymbol{\theta}_s$ for all $v \in [0,1]$;
ii) $\mathbf{S}(1,v) = \boldsymbol{\gamma}_g(v)$ for all $v \in [0,1]$ where $\boldsymbol{\gamma}_g(v)$ is a smooth connected curve;
iii) For a fixed $v \in [0,1]$ and a fixed integer $i = 1, 2, \ldots, n$, the ith component $S_i(u,v)$ of $\mathbf{S}(u,v)$ is either a strictly monotonic function of u such that $\partial S_i / \partial u \neq 0$ for $0 < u < 1$ or, in the case when $\gamma_{gi}(v) = \theta_{si}$, a constant with respect to u.

Furthermore, assume that any point $\boldsymbol{\theta}$ on $\boldsymbol{\mathcal{S}}$ satisfies

$$\min_{0 \leq v \leq 1} \gamma_{gi}(v) \leq \theta_i \leq \max_{0 \leq v \leq 1} \gamma_{gi}(v), \quad \text{for } i = 1, 2, \ldots, n.$$

Then given any point $\boldsymbol{\theta}_0$ on $\boldsymbol{\mathcal{S}}$ and any integer $i = 1, 2, \ldots, n$, there is a curve $\boldsymbol{\theta} : [0,1] \to \mathbb{R}^n$ lying entirely on $\boldsymbol{\mathcal{S}}$ that connects $\boldsymbol{\theta}_0$ to a point $\boldsymbol{\theta}_1$ on the curve $\boldsymbol{\gamma}_g(v)$ such that $\theta_i(t) = \theta_{0i}$ for all $t \in [0,1]$.

Proof: The result will be proven by constructing a suitable curve. Fix $i \in \{1, 2, \ldots, n\}$. Given $\boldsymbol{\theta}_0 \in \boldsymbol{\mathcal{S}}$, there is a pair (u_0, v_0) such that $\mathbf{S}(u_0, v_0) = \boldsymbol{\theta}_0$. By assumption (iv), $\gamma_{gi}(v_1) = \theta_{0i}$ for some $v_1 \in [0,1]$ where we assume v_1 is the closest such value to v_0. We need to show that there is a curve $\boldsymbol{\theta} : [0,1] \to \mathbb{R}^n$ such that $\boldsymbol{\theta}(0) = \boldsymbol{\theta}_0$, $\boldsymbol{\theta}(1) = \mathbf{S}(1, v_1)$, and $\theta_i(t) = \theta_{0i}$ for all $t \in [0,1]$.

The problem is trivial if $\boldsymbol{\theta}_0$ is on the $\boldsymbol{\gamma}_g$ curve. Assuming that it is not, we consider three cases. First, suppose that $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_s$. Then $\gamma_{gi}(v_1) = \theta_{si}$ and by assumption (iii) the curve $\boldsymbol{\theta}(t) = \mathbf{S}(t, v_1)$ is a suitable curve. Next, consider the case when $\boldsymbol{\theta}_0 \neq \boldsymbol{\theta}_s$ but $\theta_{0i} = \theta_{si}$. Then $u_0 \neq 0$ but $S_i(0, v_0) = S_i(u_0, v_0)$ so that $S_i(u, v_0)$ is not strictly monotonic in u. But, by assumption (iii), this implies that $S_i(u, v_0) = \theta_{si}$ for $0 \leq u \leq 1$. This gives a suitable curve $\boldsymbol{\theta}(t) = \mathbf{S}(u_0(1-t) + t, v_0)$.

Lastly, consider the case when $\theta_{0i} \neq \theta_{si}$. The goal is to determine a trajectory $\mathbf{w}(t) = [u(t)\ v(t)]^T$, $0 \leq t \leq 1$, going from $\mathbf{w}_0 = [u_0\ v_0]^T$ to $\mathbf{w}_1 = [1\ v_1]^T$ while keeping $S_i(u(t), v(t)) = \theta_{0i}$ constant. Since the domain is compact, this

can be done by applying an inverse velocity kinematics type algorithm to the system

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \mathbf{g}(\mathbf{w}) = \begin{bmatrix} v \\ S_i(u,v) \end{bmatrix} \tag{5}$$

to find a suitable trajectory $(u(t), v(t))$ provided, of course, that the Jacobian of $\mathbf{g}$

$$\mathbf{G} = \frac{\partial \mathbf{g}}{\partial \mathbf{w}} = \begin{bmatrix} 0 & 1 \\ \frac{\partial S_i}{\partial u} & \frac{\partial S_i}{\partial v} \end{bmatrix} \tag{6}$$

does not become singular along the way. However, by assumption (iii), the determinant of $\mathbf{G}$, which is equal to $-\partial S_i/\partial u$, is zero only when $S_i(u,v) = \theta_{si}$. This never occurs along the trajectory as $S_i(u(t), v(t)) \neq \theta_{si}$ is fixed. We can therefore generate a trajectory $(u(t), v(t))$ such that

$$\mathbf{z}(t) = \begin{bmatrix} v_0(1-t) + v_1 t \\ \theta_{0i} \end{bmatrix}, \quad 0 \leq t \leq 1. \tag{7}$$

To finish off the proof, we need to show that $u(t)$ stays on the interval [0,1] throughout the trajectory and ends at 1. We do this by first showing that $u(t)$ cannot be 1 or 0 for $0 \leq t < 1$ and then using an argument concerning the monotonicity of S_i with respect to u to show that $u(1)$ must be 1. The case $u(t) = 0$ for some $0 \leq t < 1$ violates the assumption that $\theta_{0i} \neq \theta_{si}$ while the case $u(t) = 1$ for some $0 \leq t < 1$ violates the assumption that v_1 is the closest v-value to v_0 such that $\gamma_{gi} = \theta_{0i}$. Since $S_i(u(t), v(t)) = \theta_{0i}$ on the trajectory, we have that $S_i(u(1), v_1) = \theta_{0i} = \gamma_{gi}(v_1) = S_i(1, v_1)$ so that by assumption (iii), $u(1) = 1$. We thus have a suitable curve $\boldsymbol{\theta}(t) = \mathbf{S}(u(t), v(t))$. ∎

Thus, the monotonicity of paths from $\boldsymbol{\theta}_s$ to $\boldsymbol{\gamma}_g(v)$ guarantees that after a single locked-joint failure, the given robot will not get stuck in the vicinity of an internal local minimum or maximum and will reach a point on $\boldsymbol{\gamma}_g(v)$. By specifying only the start and goal locations for pick and place tasks in the presence of obstacles, the solution being presented provides some flexibility in choosing the obstacle-free, failure-tolerant path towards the goal based on some desired optimization criteria prior to a failure. After a failure, there would exist a finite number of feasible paths. When there is more than one feasible path the shortest path towards the goal, for example, could be chosen.

Theorem 1 motivates the method for identifying a failure surface in the algorithm by generating monotonic paths from $\boldsymbol{\theta}_s$ to $\boldsymbol{\gamma}_g(v)$. Note that this theorem is only a sufficient condition and can only guarantee task completion when the failure surface exists. When the failure surface identification is unsuccessful, a solution may still be present.

B. An Algorithm

The procedure for implementing failure-tolerant path planning with obstacle avoidance is enumerated in the following.

1) Determine the start self-motion manifold, $\boldsymbol{\mathcal{M}}_s$, and goal self-motion manifold, $\boldsymbol{\mathcal{M}}_g$, from the given start, $\mathbf{x}_s$, and goal, $\mathbf{x}_g$, workspace position and/or orientation, respectively.
2) Identify an obstacle-free start configuration, $\boldsymbol{\theta}_s$, and an obstacle-free portion of the goal self-motion manifold, $\boldsymbol{\gamma}_g$. Those portions of the self-motion manifold that are outside the joint limits are treated in the same manner as configurations corresponding to collisions with obstacles. Thus, by obstacle-free configurations we mean those configurations that are not in collision with obstacles and are within the joint limits.
3) Check for intersections of the failure hyperplane $\mathbf{H}_i(\boldsymbol{\theta}_s)$ with $\boldsymbol{\gamma}_g$ for $i = 1, \ldots, n$. (Note that this step uses the necessary condition in Section III-A.) This check, although in a different form, was also performed in [18] and [31].
4) Check for the existence of a failure surface, $\boldsymbol{\mathcal{S}}$. This is performed by generating monotonic paths from the feasible $\boldsymbol{\theta}_s$ to points on $\boldsymbol{\gamma}_g(v)$ and checking for intersections with obstacles.[1] (This step utilizes the sufficient condition in Section III-A.)

The computational complexity of the proposed algorithm is highly dependent on the method used for computing the start and goal self-motion manifolds, and the method used for collision detection. For $r = 1$, the computational complexity is $O(mn^2) + O(mnp)$ where p is the number of obstacles in the workspace. The first term corresponds to the computation of the self-motion manifolds, while the second term is due to collision detection.

For higher degrees of redundancy, a $\boldsymbol{\gamma}_g$ curve would need to be identified in the corresponding higher dimensional goal self-motion manifold. This is done by first identifying the obstacle-free intersections of the manifold with all of the failure hyperplanes $\mathbf{H}_i(\boldsymbol{\theta}_s)$ for $i = 1, \ldots, n$. If the goal self-motion manifold does not intersect each $\mathbf{H}_i$, then no $\boldsymbol{\gamma}_g$ can exist. The next step is to connect one point from each failure hyperplane intersection. Two failure hyperplane points are connected by generating a path along the self-motion manifold, e.g., by using the null vectors from the Jacobian, along with a term to avoid obstacles. If all intersection points can be connected with such obstacle-free paths that lie in the goal self-motion manifold, then the resulting connected curves represent a suitable $\boldsymbol{\gamma}_g$.

C. Generating Monotonic Surfaces

1) Monotonic Curves: Parametric monotonic quadratic polynomials $\mathbf{p}(u)$ are used to generate a family of paths from $\boldsymbol{\theta}_s$ to $\boldsymbol{\gamma}_g$. These paths are chosen to form an obstacle-free, continuously differentiable surface $\boldsymbol{\mathcal{S}}$ satisfying the conditions of Theorem 1.

It is easy to see that any quadratic polynomial $p_i(u)$ satisfying the constraints $p_i(0) = \theta_{si}$ and $p_i(1) = \theta_{gi}$ has the form

$$p_i(u) = \theta_{si}(1-u) + \theta_{gi}u + \alpha_i u(1-u) \tag{8}$$

where the parameter α_i can be any real number and θ_{gi} denotes the ith component of $\boldsymbol{\theta}_g$. Although α_i does not affect the values at the end-points, it does completely characterize whether the polynomial is monotonic or not. To see this, note that a polynomial is monotonic on the closed interval [0,1] if and only if its derivative does not change sign on the open interval (0,1). Since the derivative of a quadratic polynomial represents the equation of a line, it follows that one only needs to check the endpoints of the interval [0,1] to determine the set of α_i's that

[1]Note that the obstacles must be grown in order to guarantee that no collision can exist between curves.

makes (8) monotonic on [0,1]. Therefore, for (8) to be monotonic on [0,1], $p_i'(0) = \theta_{gi} - \theta_{si} + \alpha_i$ and $p_i'(1) = \theta_{gi} - \theta_{si} - \alpha_i$ should not have opposite signs. This is clearly true if and only if $|\alpha_i| \leq |\theta_{gi} - \theta_{si}|$.

2) Monotonic Surfaces: Equation (8) can be used to determine a failure surface $\boldsymbol{\mathcal{S}}$ described by

$$\mathbf{S}(u,v) = \boldsymbol{\theta}_s(1-u) + \boldsymbol{\theta}_g u + \boldsymbol{\alpha}(v)u(1-u). \tag{9}$$

Suitable values for $\alpha_i(v)$ are determined for discrete values of $v = k\delta v$ along the $\boldsymbol{\gamma}_g(v)$ curve. These values are then connected so that $\alpha_i(v)$ has first-order continuity. The region between the monotonic paths determined by $\alpha_i(k\delta v)$ can be guaranteed to be collision free if the obstacles are suitably enlarged prior to computing the monotonic quadratic curves. One must also check that the monotonicity condition $|\alpha_i| \leq |\theta_{gi} - \theta_{si}|$ continues to hold in between the discrete values $v = k\delta v$. With this choice of $\alpha_i(v)$, the conditions of Theorem 1 are satisfied so that $\boldsymbol{\mathcal{S}}$ is a suitable failure surface.

3) Trajectory Generation: Once a monotonic surface is precomputed, trajectory generation for the robot is very computationally efficient. Specifically, the robot starts from configuration $\boldsymbol{\theta}_s$ and its trajectory is generated by following a desired path on the surface given by a fixed value of the parameter v by evaluating the quadratic $\mathbf{S}(u, v = v_d)$ for successively increasing values of u. If a failure occurs in joint f at $\mathbf{S}(u_0, v_0)$ while the robot is moving, then, in general, it will no longer be possible to follow the same curve because S_f must remain constant. The trajectory to reach the desired goal workspace location is now computed by solving the quadratic $S_f(u_0, v_0) = S_f(u_0 + \delta u, v)$ for the value of v that will keep the trajectory on the surface. The value of δu is incremented until $u_0 + \delta u = 1$ at which point the robot will be at the desired workspace location.

IV. 3-DOF Planar Robot Example

A 3-DOF planar robot with equal link lengths of 100 units was first used to explore the feasibility of the proposed approach. The workspace contained a number of circular obstacles, each of a diameter of 40 units, where the number of obstacles was varied from zero to twenty in two-obstacle increments. For each number of obstacles, experiments were performed for 1000 randomly generated scenarios, where a scenario consists of a given start workspace location, a given goal workspace location, and the specified locations of the obstacles. In each scenario, the locations for the corresponding number of obstacles are randomly selected from a uniform distribution throughout the entire robot workspace. The start workspace location, $\mathbf{x}_s$, is randomly selected from a uniform distribution of [100,200] along the x-axis, that is, at a distance that corresponds to between one and two-link lengths away from the robot base. The goal workspace location, $\mathbf{x}_g$, is randomly selected to be within a range of [0,200] units from the range of start workspace locations (while restricting the goal to be within the reachable workspace of the manipulator). Fig. 2 presents an example of the start and goal locations generated for one thousand scenarios.

Five examples from the set of 11 000 scenarios are presented in Fig. 3 where the number of obstacles is varied from 20 to 12 in two-obstacle decrements. In all cases shown, the failure-tolerant, path-planning algorithm was able to identify a failure surface, $\boldsymbol{\mathcal{S}}$. The manipulator configuration shown in each scenario is the feasible starting configuration, $\boldsymbol{\theta}_s$, that was determined by the algorithm. To illustrate the difficulty of these problem scenarios, the shaded regions identify locations in the workspace where even a single obstacle will prevent the existence of a set of collision-free monotonic curves.

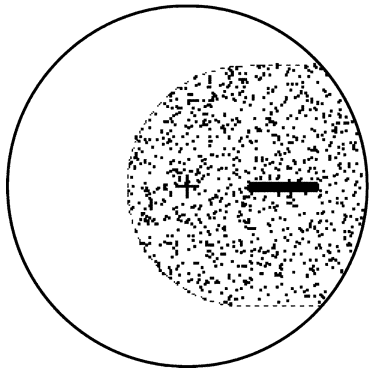


Fig. 2. Workspace of a 3-DOF planar manipulator used for the simulation experiments (all three link lengths are 100 units long). The 1000 start locations, $\mathbf{x}_s$, are randomly generated within the range [100,200] units along the x-axis (shown as a thick bold line). The 1000 goal workspace locations, $\mathbf{x}_g$, (shown as dots) are randomly generated to be within the range [0,200] units away from the range of start locations (but inside the reachable workspace). The center of the workspace is marked with a bold cross, the workspace boundary with a solid line, and the boundary of the goal locations with a dashed line.

Table I shows the values of the start configuration $\boldsymbol{\theta}_s = [\theta_{s1}, \theta_{s2}, \theta_{s3}]^T$ corresponding to the scenarios in Fig. 3. The failure plane $\mathbf{H}_i(\boldsymbol{\theta}_s)$ intersects the continuous, obstacle-free goal self-motion manifold $\boldsymbol{\gamma}_g$ at $\boldsymbol{\theta}_{Hi} = [\theta_{Hi1}, \theta_{Hi2}, \theta_{Hi3}]^T$ where $i = 1, 2, 3$. Fig. 4 shows projections of the self-motion manifolds in C-space for the scenarios of Fig. 3(a)–(c) where a failure surface exists. The cross on a self-motion manifold denotes the feasible start configuration, $\boldsymbol{\theta}_s$, for the corresponding scenario. The $\boldsymbol{\gamma}_g(v)$ that satisfied the necessary condition is the curve on the self-motion manifold between the points labeled "0" and "1". The web-like network of paths represents the failure surface for each of the corresponding scenarios. In the following subsections, data gathered from these 11 000 simulation experiments will be presented according to the order in which these steps are performed in the algorithm described in Section III-B. To illustrate the effects of joint limits, an additional 11 000 scenarios were performed with $-135° \leq \theta_1 \leq 135°$, $-90° \leq \theta_2 \leq 90°$, and $-135° \leq \theta_3 \leq 135°$. In all cases the algorithm was executed on a computer with dual Intel Xeon processors running at 2.4 GHz.

A. Computation of Self-Motion Manifolds

The first step in the failure-tolerant, path-planning algorithm is to compute the self-motion manifold(s) for both the start and goal workspace locations. These are computed by identifying a single configuration on each disjoint manifold and then stepping along the manifold (in two degree increments) by integrating the null vector of the manipulator Jacobian matrix (which corresponds to the tangent of the self-motion manifold). The average lengths for the sum of all manifolds corresponding to a

TABLE I
SET OF FEASIBLE, OBSTACLE-FREE START CONFIGURATIONS, $\boldsymbol{\theta}_s$'s, SHOWN IN FIG. 3 WITH THE CORRESPONDING FAILURE PLANES $\mathbf{H}_i(\boldsymbol{\theta}_s)$ INTERSECTING A CONTINUOUS, OBSTACLE-FREE PORTION OF THE GOAL SELF-MOTION MANIFOLD, $\boldsymbol{\gamma}_g(v)$, SHOWN IN UNITS OF DEGREES

Fig. 3	Feasible Start Conf., $\boldsymbol{\theta}_s$			Failure Plane $\mathbf{H}_i(\boldsymbol{\theta}_s)$ Intersecting $\boldsymbol{\gamma}_g(v)$								
	θ_{s1}	θ_{s2}	θ_{s3}	θ_{H11}	θ_{H12}	θ_{H13}	θ_{H21}	θ_{H22}	θ_{H23}	θ_{H31}	θ_{H32}	θ_{H33}
(a)	-7.3	123.2	-166.5	-6.3	173.8	-128.9	39.3	123.1	-174.2	32.1	155.5	-168.3
(b)	38.0	-126.9	111.5	36.3	-155.6	163.0	26.6	-126.8	171.4	-43.6	62.4	110.3
(c)	-64.7	61.8	75.9	-65.8	60.9	140.7	-71.6	63.6	136.0	102.2	164.4	77.1
(d)	4.5	-90.0	152.3	4.0	66.1	104.1	73.6	-88.3	170.2	47.5	8.5	150.5
(e)	-71.6	75.3	58.5	-69.9	160.5	-2.7	-7.2	76.9	96.0	-40.8	114.9	57.1

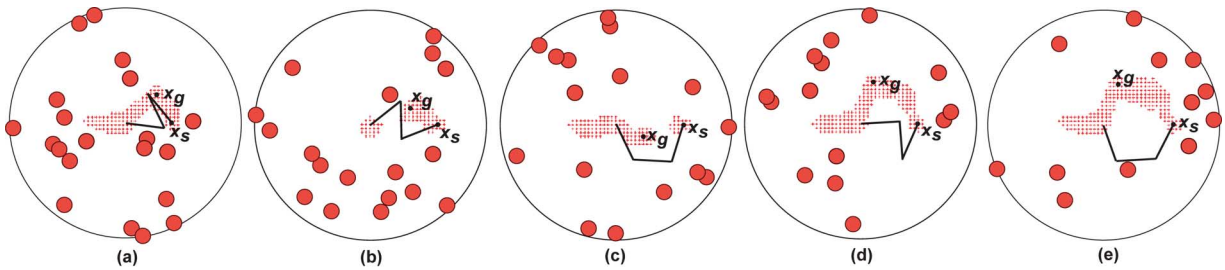


Fig. 3. Five examples selected from the 11 000 scenarios where the failure-tolerant, path-planning algorithm was applied to a 3-DOF planar manipulator. The scenarios correspond to workspaces containing from 20 to 12 obstacles where both the necessary and the sufficient conditions were satisfied. The shaded region in each scenario represents an area where a single obstacle will eliminate any possible monotonic curve connecting a $\boldsymbol{\theta}_s$ to its corresponding $\boldsymbol{\theta}_g$. (Color version available online at: http://ieeexplore.ieee.org.)

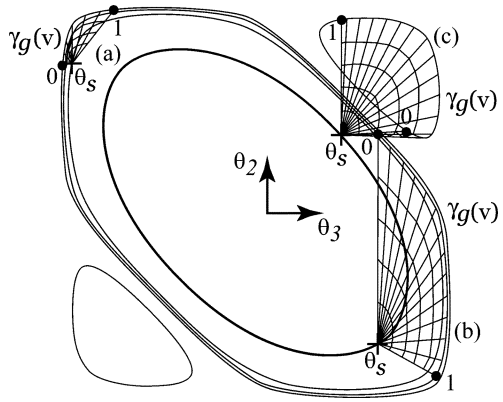


Fig. 4. Projections of the self-motion manifolds on the $\theta_2\theta_3$-plane for the scenarios of Fig. 3(a)–(c). The failure surface, $\boldsymbol{\mathcal{S}}$, is shown as a web-like network of paths. The feasible $\boldsymbol{\theta}_s$ is shown as a cross on the self-motion manifold while the $\boldsymbol{\gamma}_g(v)$ is the curve on the self-motion manifold between the points labeled "0" and "1". For the example shown in Fig. 3(c), the goal manifold consists of two disjoint manifolds in the C-space because $\mathbf{x}_g$ is less than one link length away from the robot base.

TABLE II
COMPUTATION OF SELF-MOTION MANIFOLDS

	Ave.	Std. Dev.
Length of $\boldsymbol{\mathcal{M}}_s$ (deg)	1043.95	4.97
Length of $\boldsymbol{\mathcal{M}}_g$ (deg)	883.71	8.52
Time to compute $\boldsymbol{\mathcal{M}}_s$, $\boldsymbol{\mathcal{M}}_g$ (ms)	41.90	5.88

workspace location are given in Table II. The length for the start manifolds are larger than those for the goal manifold because the start workspace locations were intentionally restricted to a region of the workspace where these manifolds are larger, and thus, the locations are more failure tolerant [18]. (A distance of one link length from the base is optimally failure tolerant, i.e., all joints span their entire range of motion, while the locations on the workspace boundary are most failure intolerant, i.e., their self-motion manifolds consist of a single point.) The average computation time over all manifolds was 41.9 ms.

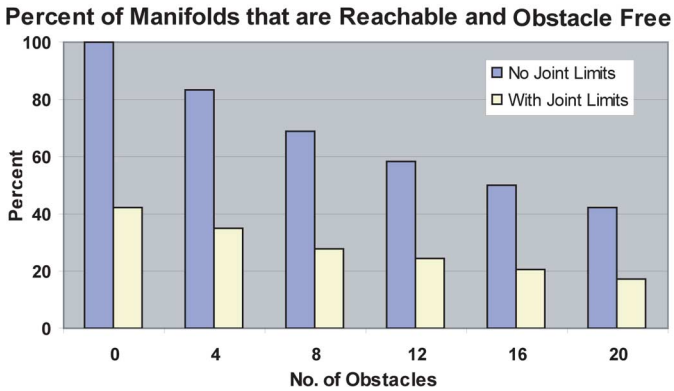


Fig. 5. Percent of the self-motion manifolds that are reachable and obstacle free for a 3-DOF manipulator with and without joint limits. (Color version available online at: http://ieeexplore.ieee.org.)

B. Identifying Obstacle-Free Portions of $\boldsymbol{\mathcal{M}}_s$ and $\boldsymbol{\mathcal{M}}_g$

The next step is to determine the obstacle-free portions of the manifolds. This identifies candidate feasible, obstacle-free start configurations, $\boldsymbol{\theta}_s$, and continuous, obstacle-free portions of the goal self-motion manifold, $\boldsymbol{\gamma}_g$, that can possibly satisfy the necessary condition.

Fig. 5 shows the percentage of the start self-motion manifold, $\boldsymbol{\mathcal{M}}_s$, and goal self-motion manifold, $\boldsymbol{\mathcal{M}}_g$, that are reachable and obstacle free as a function of the number of obstacles in the workspace. As expected, the percentage of the obstacle-free self-motion manifold decreases as the number of workspace obstacles increases. The time required to determine which portions

TABLE III
COMPUTATION TO CHECK THE NECESSARY CONDITION (N.C.)

No. of Obs.	Time N.C. Sat. (ms)	Time N.C. Not Sat. (ms)	% Cases N.C. Sat.	% $\mathcal{M}_s$ N.C. Sat.
0	0.73	4.39	72.0	25.4
2	0.84	4.59	65.5	20.5
4	1.13	5.03	60.6	16.5
6	1.13	4.86	55.5	15.4
8	1.05	4.57	49.2	12.4
10	1.33	5.13	47.1	10.0
12	1.48	5.35	37.3	7.6
14	1.50	5.43	35.4	6.3
16	1.84	6.15	31.3	5.9
18	1.90	5.56	28.7	4.1
20	2.92	5.53	24.1	3.7

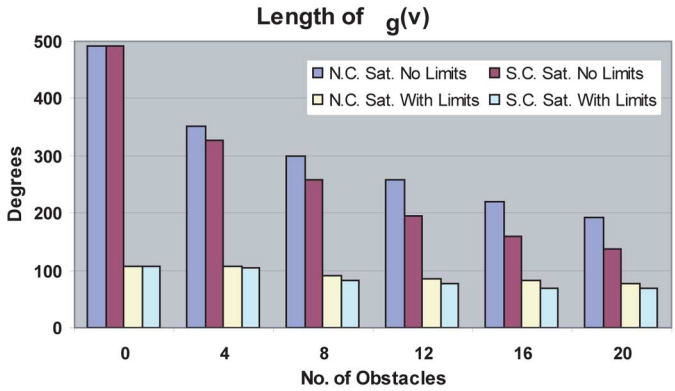


Fig. 6. Average length of a $\boldsymbol{\gamma}_g(v)$ that satisfies the necessary condition (N.C.) and that also satisfies the sufficient condition (S.C.) as a function of the number of obstacles in the workspace. The trends are the same for cases with and without joint limits. (Color version available online at: http://ieeexplore.ieee.org.)

of a manifold are obstacle free is proportional to the number of obstacles by a proportionality constant of 16.5 ms/obstacle.

In many cases, one could avoid most of the computation time associated with checking the entire manifold for collisions with obstacles by performing the necessary condition check first, and then verifying that the start configuration and the corresponding portion of the goal self-motion manifold are collision free.

C. Checking the Necessary Condition

Table III presents the computational data associated with checking the necessary condition. As the number of obstacles in the workspace increases, the percentage of cases that satisfy the necessary condition decreases, reaching a minimum of 24% for the case with twenty obstacles (10% if joint limits are included). For those cases where a $\boldsymbol{\theta}_s$ satisfied the necessary condition, we further processed the start manifold to see what percentage of the start manifold would be able to satisfy the necessary condition. (This is not required by the algorithm and the time required to perform this computation was not included in the overall execution time data presented.) Table III shows that this is also a monotonically decreasing function of the number of obstacles in the workspace. Thus, it becomes increasingly more time consuming to identify a $\boldsymbol{\theta}_s$ that satisfies the necessary condition as the number of obstacles in the workspace increases. This is illustrated by the fact that the computation time is a monotonically increasing function of the number of obstacles. This is true despite the fact that there is less and less of the start manifold that is obstacle free (see Fig. 5) because an increasingly smaller percentage of the obstacle-free manifold is able to satisfy the necessary condition. In contrast, the time to compute that the necessary condition is not satisfied is relatively independent of the number of obstacles. This at first appears anomalous because one would expect this to be a monotonically decreasing function due to a smaller percentage of the start manifold needing to be checked (because less of it is obstacle free). However, this is offset by the fact that larger and larger manifolds are now failing the necessary condition, thus keeping the computation time relatively constant.

Fig. 6 shows the average length of the $\boldsymbol{\gamma}_g(v)$ from a $\{\boldsymbol{\theta}_s, \boldsymbol{\gamma}_g\}$ pair that satisfies the necessary condition. This generally decreases as the number of obstacles increases due to the fact that it is more difficult to have a large $\boldsymbol{\gamma}_g(v)$ because they are required to be obstacle free. The size of a $\boldsymbol{\gamma}_g(v)$ that satisfies the necessary condition is correlated to the distance between the start and goal workspace locations so that, as expected, start and goal locations must be closer together as more and more obstacles are added to the workspace.

D. Computing a Failure Surface

The final step in the algorithm is to check the sufficient condition by attempting to compute a failure surface that guarantees the existence of a solution to the failure-tolerant, path-planning problem. Once a feasible $\boldsymbol{\theta}_s$ is found from the previous step, the search for a failure surface begins. A failure surface is identified by generating monotonic paths that connect a feasible $\boldsymbol{\theta}_s$ to points on its corresponding $\boldsymbol{\gamma}_g(v)$ that satisfies the necessary condition. (The $\boldsymbol{\gamma}_g(v)$ curve is discretized at a resolution of two degrees.) For each point on $\boldsymbol{\gamma}_g(v)$ the algorithm first attempts to use a straight-line path. If this path is not obstacle free, then it attempts to find a monotonic quadratic path that is obstacle free up to a desired resolution of α_i in (8). If no such path can be found then the algorithm discards this $\{\boldsymbol{\theta}_s, \boldsymbol{\gamma}_g\}$ pair and uses the next $\{\boldsymbol{\theta}_s, \boldsymbol{\gamma}_g\}$ pair that satisfies the necessary condition. If all such pairs are exhausted without completing a failure surface then the algorithm terminates with a message that it was unsuccessful.

It is interesting to note that once the necessary condition is satisfied, it is highly likely that a failure surface will be found, i.e., this occurs 84% of the time. In addition, this percentage is relatively independent of the number of obstacles that are in the workspace as illustrated in Fig. 7 (except, of course, for the case of no obstacles). This is fortuitous, because the construction of failure surfaces is by far the most time consuming portion of the algorithm. The average time for computing a failure surface as a function of the number of obstacles is given in the second column of Table IV. The overall average time was 1.5 s with the maximum time to compute a failure surface over all scenarios was 78.0 s. Similarly, it takes on average 1.8 s to exhaust all possible candidates in cases where a surface cannot be found, with a maximum time of 66.7 s. Thus, the time to evaluate surfaces is seldom wasted, with the majority of the cases where no solution exists being identified in a matter of milliseconds by the necessary condition test.

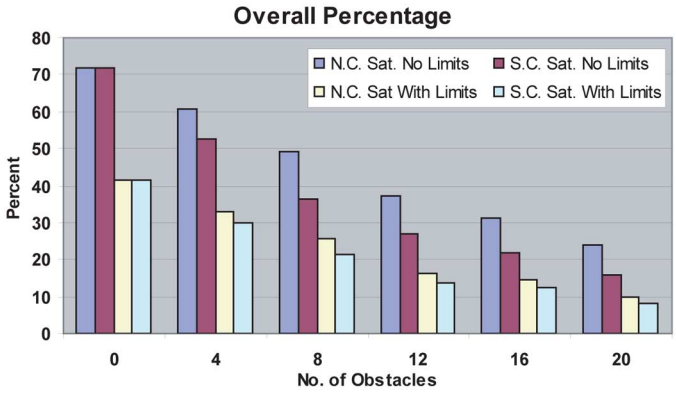


Fig. 7. Percent of total cases where the necessary condition (N.C.) is satisfied and percent of total cases where the sufficient condition (S.C.) is satisfied, i.e., where a failure surface $\mathcal{S}$ is found. The trends are the same for cases with and without joint limits. (Color version available online at: http://ieeexplore.ieee.org.)

TABLE IV
COMPUTATIONS WHEN $\mathcal{S}$ IS FOUND

No. of Obs.	Overall Time (s) L+Q	(L+Q)*	L	No. of Path Evaluations L+Q	(L+Q)*	L	Diff.
0	0.04	0.04	0.04	246	246	246	0
2	0.35	0.34	0.41	658	651	755	5
4	0.78	0.75	0.75	818	793	800	4
6	1.16	1.10	1.07	844	793	752	4
8	1.49	1.42	1.32	818	778	718	2
10	1.77	1.61	1.35	775	698	569	5
12	2.19	2.21	1.78	815	823	630	8
14	2.28	2.26	1.62	718	709	481	2
16	2.68	2.49	1.90	743	682	496	5
18	2.15	2.16	1.72	503	505	379	4
20	1.89	1.87	1.40	381	376	257	1

Because monotonic path generation represents the most time consuming portion of the algorithm, an additional analysis was performed to determine the number and computational cost of linear paths versus quadratic paths. The data from this analysis is presented in Table IV where "L" denotes that only linear paths were tried by the algorithm, "L+Q" denotes that both linear and quadratic paths were tried, and "(L+Q)*" represents the subset of "L+Q" scenarios that correspond to cases where the "L" algorithm was also able to compute a failure surface. It is interesting to note that nearly all (99.6%) of the paths in failure surfaces are linear. Furthermore, the number of additional failure surfaces that are identified when trying quadratic paths is minimal, i.e., always less than 1% (see the last column in Table IV). Given that the use of quadratic paths results in higher execution times (this is true for all cases except for four or fewer obstacles) and a minimal improvement in the number of failure surfaces being identified, the benefit of implementing higher order, monotonic paths is questionable.

E. Comparison to Previous Approaches

It is interesting to compare the results of our approach with that in [31]. Both approaches are similar in that the set of monotonic curves generated by the algorithm presented here plays the same role as the connectivity graph in [31]. However, any workspace trajectory generated by a path through the connectivity graph will result in the same end-effector trajectory because this is required by the problem definition in [31]. While this is appropriate for tasks that must have the end-effector follow a prescribed trajectory, it limits the number of possible choices for failure-tolerant trajectories if the robot is only required to perform a pick-and-place type operation. The method proposed here will produce an infinite number of possible obstacle-free, failure-tolerant paths from the start to the goal[2] because it does not constrain the end-effector trajectory.

Fig. 8 illustrates the broad range of end-effector trajectories that are failure tolerant if the task is only constrained to a desired start and goal location rather than a complete trajectory. Each of the subfigures (a)–(e) show the mapping of a set of failure-tolerant, monotonic curves in the configuration space to the workspace for the same start and goal location. Clearly, relaxing the constraint on the end-effector trajectory makes it much more likely that a collision-free, failure-tolerant path will exist.

V. SEVEN DOF REDUNDANT ROBOT EXAMPLE

The proposed failure tolerant path-planning algorithm was also implemented for the Mitsubishi PA-10 seven degree-of-freedom manipulator because it is the most common commercially available redundant robot. Unfortunately, the PA-10 is not a fully kinematically failure-tolerant robot because the null space component of joint four is identically zero, i.e.,

$$\hat{n}_{J4} = 0 \tag{10}$$

throughout the entire C-space. Physically, this is due to the fact that joint four is the only joint that can alter the distance between the wrist and the shoulder, which is why the PA-10 is intolerant to a failure in this joint.[3] Therefore, for this example we will assume that there will be no failure in joint four and consider planning a failure-tolerant path for the remaining six joints. For this example, we do not consider joint limits or self-collisions.

A total of 12 000 randomly generated scenarios were performed on the PA-10 where spherical obstacles with a diameter of 0.254 m (10 in) are randomly placed in the robot's workspace. The number of obstacles was varied from zero to ten such that 2000 experiments were performed for each number of obstacles. The range of desired workspace locations is constrained in x and y to [−0.8,0.8] m and in z to [0,0.8] m. It is further specified that the distance of the resulting position from the base of the robot be in the range of [0.3,0.8] m. Thus, the volume where the workspace positions are randomly picked consists of a hemisphere with an inner radius of 0.3 m and outer radius of 0.8 m. It is in this range of positions where the excursion of the self-motion manifolds for the PA-10 are relatively large. For simplicity, the end-effector is considered to be at the wrist so that no restrictions on orientation are required. The results of the experiments are shown in Fig. 9. These results are very similar to the 3-DOF

[2]The specific trajectory that the robot follows can then be selected based on some secondary criterion.

[3]There are a number of different measures that can be used to quantify global fault tolerance based on the maximum excursion of a joint as it spans the self-motion manifold. See [34] for several such measures.

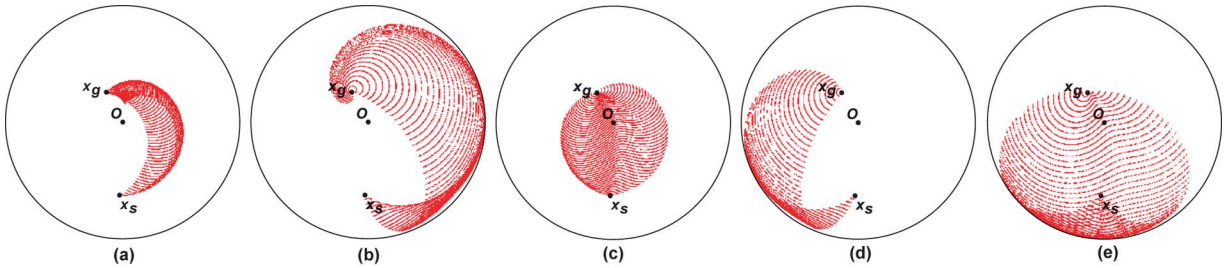


Fig. 8. Subfigures (a)–(e) have the same start and goal workspace locations where $\mathbf{x}_s = [-8.8, -188]^T$ and $\mathbf{x}_g = [-42.5, 76.8]^T$. The shaded regions represent the end-effector locations corresponding to different sets of monotonic curves that will take the end-effector from $\mathbf{x}_s$ to $\mathbf{x}_g$. (Color version available online at: http://ieeexplore.ieee.org.)

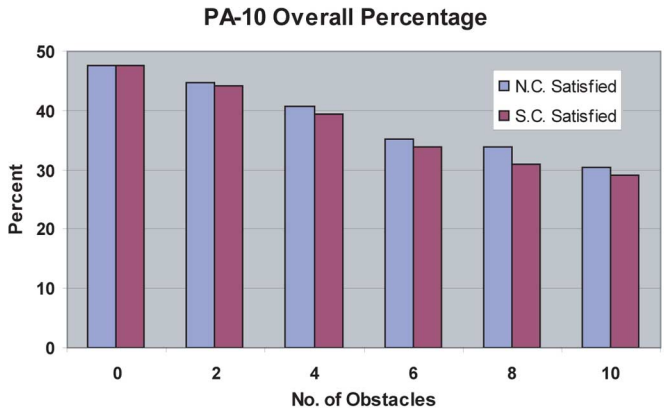


Fig. 9. Percent of 12 000 random scenarios for the PA-10 where the necessary condition (N.C.) is satisfied and where the sufficient condition (S.C.) is satisfied, i.e., where a failure surface, $\boldsymbol{\mathcal{S}}$, is found. (Color version available online at: http://ieeexplore.ieee.org.)

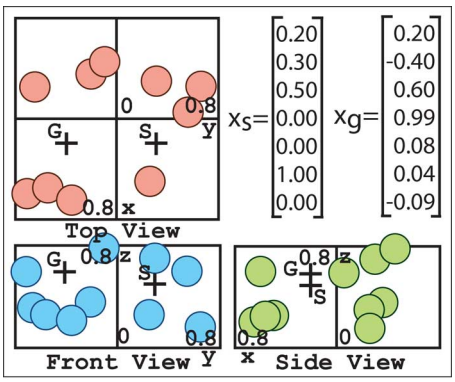


Fig. 10. A scenario within the specified range of workspace locations of the 12 000 experiments for the PA-10 robot with ten random obstacles in the workspace. Each of the obstacles has a diameter of 0.254 m (10 in). The corresponding $\mathbf{x}_s$ and $\mathbf{x}_g$ are shown. The first three vector components correspond to the desired position in units of meters. The last four vector components correspond to the desired orientations expressed as quaternions. (Color version available online at: http://ieeexplore.ieee.org.)

case, i.e., if the necessary condition is satisfied it is highly likely that the sufficient condition will also be satisfied.

A specific example from one of the 12 000 random experiments is shown in Fig. 10. The start configuration determined by the algorithm is

$$\boldsymbol{\theta}_s = [-11.5, 75.1, 43.8, -49.2, 46.8, -117.9, 27.4]^T$$

in degrees. Its corresponding failure hyperplane $\mathbf{H}_i(\boldsymbol{\theta}_s)$ intersects $\boldsymbol{\gamma}_g$ in the following order: $i = 7, 3, 5, 2, 6, 1$.[4] Fig. 11 shows the corresponding failure surface $\boldsymbol{\mathcal{S}}$ for the example in Fig. 10.

[4]The failure hyperplane $\mathbf{H}_4(\boldsymbol{\theta}_s)$ does not intersect $\boldsymbol{\gamma}_g$ because of the intolerance to a joint four failure.

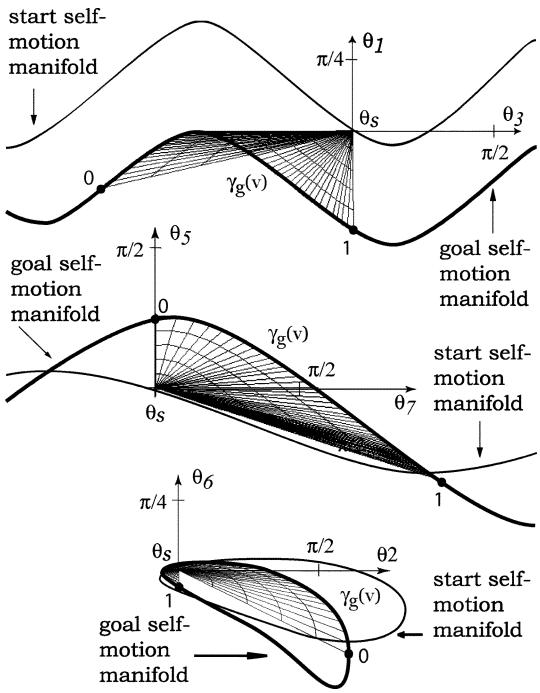


Fig. 11. Failure surface corresponding to the example in Fig. 10 shown as a web of paths in the configuration space, with projections from joint axes 1 and 3, 5 and 7, and 2 and 6. The projections are shown in the same scale with units of radians. The bold curves represent portions of $\boldsymbol{\mathcal{M}}_g$, while the less thick curves represent portions of $\boldsymbol{\mathcal{M}}_s$. The axes shown are translated from the origin to the feasible $\boldsymbol{\theta}_s$. Its corresponding $\boldsymbol{\gamma}_g(v)$ is the curve between the points labeled "0" and "1".

VI. Summary and Conclusion

This work considered the problem of guaranteeing failure tolerance when obstacles are present in the environment and the desired task is of the pick-and-place type. Conditions were formulated that guarantee the existence of a solution to this problem. An algorithm was presented that searches for a simply-connected, obstacle-free surface with no internal local minimum or maximum in the configuration space, called a failure surface, whose existence guarantees a solution. Numerous examples were presented using both a 3-DOF planar robot and a 7-DOF spatial robot to illustrate the efficacy of the algorithm.

References

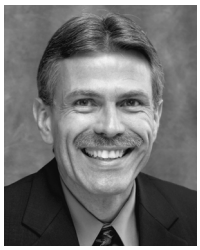
[1] E. C. Wu, J. C. Hwang, and J. T. Chladek, "Fault-tolerant joint development for the space shuttle remote manipulator system: analysis and experiment," *IEEE Trans. Robot. Automat.*, vol. 9, no. 5, pp. 675–684, Oct. 1993.

[2] G. Visentin and F. Didot, Testing Space Robotics on the Japanese ETS-VII Satellite, ESA Bulletin. European Space Agency, Paris, France, Sep. 1999, pp. 61–65.
[3] P. S. Babcock and J. J. Zinchuk, "Fault-tolerant design optimization: application to an autonomous underwater vehicle navigation system," in *Proc. Symp. Autonom. Underwater Vehicle Technol.*, Washington, D.C., Jun. 5-6, 1990, pp. 34–43.
[4] R. Colbaugh and M. Jamshidi, "Robot manipulator control for hazardous waste-handling applications," *J. Robot. Syst.*, vol. 9, no. 2, pp. 215–250, 1992.
[5] W. H. McCulloch, "Safety analysis requirements for robotic systems in DOE nuclear facilities," in *Proc. 2nd Specialty Conf. Robot. Challenging Environ.*, Albuquerque, NM, Jun. 1–6, 1996, pp. 235–240.
[6] M. L. Leuschen, I. D. Walker, and J. R. Cavallaro, "Investigation of reliability of hydraulic robots for hazardous environment using analytic redundancy," in *Proc. Annu. Rel. Maintain. Symp.*, Washington, D.C., Jan. 18–21, 1999, pp. 122–128.
[7] S. Tosunoglu and V. Monteverde, "Kinematic and structural design assessment of fault-tolerant manipulators," *Intell. Automat. Soft Comput.*, vol. 4, no. 3, pp. 261–268, 1998.
[8] D. L. Schneider, D. Tesar, and J. W. Barnes, "Development and testing of a reliability performance index for modular robotic systems," in *Proc. Annu. Rel. Maintain. Symp.*, Anaheim, CA, Jan. 24–27, 1994, pp. 263–271.
[9] B. S. Dhillon, *Robot Reliability and Safety*. New York: Springer-Verlag, 1991.
[10] K. Khodabandehloo, "Analysis of robot systems using fault and event trees: Case studies," *Rel. Eng. Syst. Safety*, vol. 53, no. 3, pp. 247–264, Sep. 1996.
[11] B. S. Dhillon and A. R. M. Fashandi, "Safety and reliability assessment techniques in robotics," *Robotica*, vol. 15, no. 6, pp. 701–708, Nov. –Dec. 1997.
[12] C. Carreras and I. D. Walker, "Interval methods for fault-tree analysis in robotics," *IEEE Trans. Robot. Automat.*, vol. 50, no. 1, pp. 3–11, Mar. 2001.
[13] W. S. Ng and C. K. Tan, "On safety enhancements for medical robots," *Rel. Eng. Syst. Safety*, vol. 54, no. 1, pp. 35–35, Oct. 1996.
[14] J. Trevelyan, "Simplifying robotics—a challenge for research," *Robot. Autonom. Syst.*, vol. 21, no. 3, pp. 207–220, Sep. 1997.
[15] I. D. Walker and J. R. Cavallaro, "The use of fault trees for the design of robots for hazardous environments," in *Proc. Annu. Rel. Maintain. Symp.*, Las Vegas, NV, Jan. 22–25, 1996, pp. 229–235.
[16] A. A. Maciejewski, "Fault tolerant properties of kinematically redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, Cincinnati, OH, May 13–18, 1990, pp. 638–642.
[17] R. G. Roberts and A. A. Maciejewski, "A local measure of fault tolerance for kinematically redundant manipulators," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 543–552, Aug. 1996.
[18] C. L. Lewis and A. A. Maciejewski, "Fault tolerant operation of kinematically redundant manipulators for locked joint failures," *IEEE Trans. Robot. Automat.*, vol. 13, no. 4, pp. 622–629, Aug. 1997.
[19] K. N. Groom, A. A. Maciejewski, and V. Balakrishnan, "Real-time failure-tolerant control of kinematically redundant manipulators," *IEEE Trans. Robot. Automat.*, vol. 15, no. 6, pp. 1109–1116, Dec. 1999.
[20] J. D. English and A. A. Maciejewski, "Fault tolerance for kinematically redundant manipulators: anticipating free-swinging joint failures," *IEEE Trans. Robot. Automat.*, vol. 14, no. 4, pp. 566–575, Aug. 1998.
[21] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro, "New dynamic model-based fault detection threshold for robot manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, San Diego, CA, May 8–13, 1994, pp. 1388–1395.
[22] L. S. Lopes and L. M. Camarinha-Matos, "A machine learning approach to error detection and recovery in assembly," in *Proc. Int. Conf. Intell. Robots Syst.*, Pittsburgh, PA, Aug. 5–9, 1995, pp. 197–203.
[23] H. Schneider and P. M. Frank, "Fuzzy logic based threshold adaption for fault detection in robots," in *Proc. 3rd IEEE Conf. Contr. Applicat.*, Glasgow, U.K., Aug. 24–26, 1994, pp. 1127–1132.
[24] T. S. Wikman, M. S. Branicky, and W. S. Newman, "Reflexive collision avoidance: a generalized approach," in *Proc. IEEE Int. Conf. Robot. Automat.*, Atlanta, GA, May 2–6, 1993, pp. 31–36.
[25] Y. Ting, S. Tosunoglu, and R. Freeman, "Actuator saturation avoidance for fault-tolerant robots," in *Proc. 32nd Conf. Decision Contr.*, San Antonio, TX, Dec. 1993, pp. 2125–2130.
[26] Y. Ting, S. Tosunoglu, and D. Tesar, "A control structure for fault-tolerant operation of robotic manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, Atlanta, GA, May 2–6, 1993, pp. 684–690.
[27] K. S. Tso, M. Hecht, and N. I. Marzwell, "Fault-tolerant robotic system for critical applications," in *Proc. IEEE Int. Conf. Robot. Automat.*, Atlanta, GA, May 2–6, 1993, pp. 691–696.
[28] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker, "A dynamic fault tolerance framework for remote robots," *IEEE Trans. Robot. Automat.*, vol. 11, no. 4, pp. 477–490, Aug. 1995.
[29] S. K. Ralph and D. K. Pai, "Computing fault tolerant motions for a robot manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, Detroit, MI, May 10–15, 1999, pp. 486–493.
[30] J. Park, W.-K. Chung, and Y. Youm, "Failure recovery by exploiting kinematic redundancy," in *Proc. 5th Int. Workshop Robot Human Commun.*, Tsukuba, Japan, Nov. 11–14, 1996, pp. 298–305.
[31] C. J. J. Paredis and P. K. Khosla, "Fault tolerant task execution through global trajectory planning," *Rel. Eng. Syst. Safety*, vol. 53, pp. 225–235, 1996.
[32] J. W. Burdick, "On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds," in *Proc. IEEE Int. Conf. Robot. Automat.*, Scottsdale, AZ, May 14–19, 1989, pp. 264–270.
[33] ——, "Kinematic analysis and design of redundant robot manipulators," Ph.D. dissertation, Stanford Univ., Stanford, CA, Mar. 1988.
[34] R. S. Jamisola, Jr., A. A. Maciejewski, and R. G. Roberts, "Failure-tolerant path planning for the PA-10 robot operating amongst obstacles," in *Proc. IEEE Int. Conf. Robot. Automat.*, New Orleans, LA, Apr. 26–May 1 2004, vol. 5, pp. 4995–5000.

Rodrigo S. Jamisola, Jr. (S'97) received the B.S. degree in mechanical engineering from the University of the Philippines, Diliman, in 1993. From 1997 to 1999, he worked toward the M.E. degree in mechanical engineering at the National University of Singapore and received the degree in 2001. He is currently working toward the Ph.D. degree in electrical and computer engineering at Colorado State University, Fort Collins.

He joined the industry in the Philippines and later in Singapore from 1994 to 1997. He was with the Singapore Institute of Manufacturing Technology from 1999 to 2001.

His research interests include force control, motion planning, and virtual environments.



Anthony A. Maciejewski (M'87–SM'00–F'05) received the B.S.E.E., M.S., and Ph.D. degrees from Ohio State University, Columbus, in 1982, 1984, and 1987, respectively.

From 1988 to 2001, he was a Professor of electrical and computer engineering at Purdue University, West Lafayette, IN. He is currently the Department Head of Electrical and Computer Engineering at Colorado State University, Fort Collins.

Rodney G. Roberts (M'92–SM'02) received the B.S. degrees in electrical engineering and mathematics from Rose-Hulman Institute of Technology, Terre Haute, IN, in 1987 and the M.S.E.E. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1988 and 1992, respectively.

From 1992 to 1994, he was a National Research Council Fellow at Wright Patterson Air Force Base, Dayton, OH. Since 1994, he has been with the Florida A&M University—Florida State University College of Engineering, Tallahassee, where he is currently a Professor of electrical and computer engineering. His research interests are in the areas of robotics and image processing.