Design of Modular Fault Tolerant Manipulators

Christiaan J. J. Paredis, *Carnegie Mellon University, Pittsburgh, PA, USA*
Pradeep K. Khosla, *Carnegie Mellon University, Pittsburgh, PA, USA*

In this paper, we deal with two important issues in relation to modular reconfigurable manipulators, namely, the determination of the modular assembly configuration optimally suited to perform a specific task and the synthesis of fault tolerant systems. We present a numerical approach yielding an assembly configuration that satisfies four kinematic task requirements: reachability, joint limits, obstacle avoidance and measure of isotropy. Further, because critical missions may involve high costs if the mission were to fail due to a failure in the manipulator system, we address the property of fault tolerance in more detail. We prove the existence of fault tolerant manipulators and develop an analysis tool to determine the fault tolerant work space. We also derive design templates for spatial fault tolerant manipulators. For general purpose manipulators two redundant degrees-of-freedom are needed for every order of fault tolerance. However, we show that only one degree of redundancy is sufficient for task specific fault tolerance.

1 Introduction

Conventional (serial or parallel link) manipulators are often considered to be general-purpose and flexible systems. Unfortunately, these systems are not general purpose. In order to understand this, consider a computer which is a general purpose computing engine if it can compute a computable function. Following a similar logic, a manipulator will be general purpose if it could do a 'doable' task. In defining a general purpose manipulator, one has, of course, to define a 'doable' task first. For the time being, let us avoid this open issue and consider two tasks that two different manipulators can perform, but that cannot be performed by either manipulator separately. If this is the case, then one may conclude that none of the above two manipulators are general purpose. So if one has to define a general purpose manipulator, then one has first to define a criterion for 'doable' tasks (like 'doability'). Such a definition may lead to the development of models of 'doability' (like computability) and maybe to Turing-like machine models of manipulators. While such a development would certainly do a lot for advancing the state-of-the-art in manipulators, it is not our intention to address this general problem.

In order to make the problem tractable, let us define a set of tasks that we would like to perform with a manipulator. Let us also define a set of basic modules (consisting of joints and links) that we may combine to create various manipulators. Finally, let us assume the existence of a methodology that will accept a task (in the form of a program or as a set of requirements) as input and find a manipulator that can be created from the given set of modules to perform the task. This scenario is described in Fig. 1, and it allows us to put forth one possible definition of a general-purpose manipulator.

General-purpose Manipulator: If for every task in the set of tasks, it is possible to find a manipulator that can be created from the given set of modules to do the task, then we define the system of modules (or the system of all possible manipulators) as general purpose with respect to the set of tasks. We will call such a system a Reconfigurable Modular Manipulator System (RMMS).

Note that the above definition does not require us to define a set of all possible 'doable' tasks nor does it require us to define the criteria for determining 'doability' even though that is the ultimate goal of our research.

Our past work has addressed the development of the modules and the technology for the RMMS [21]. The RMMS has many potential applications in both hazardous and industrial environments. It puts forth the idea of designing a specific manipulator for a task and

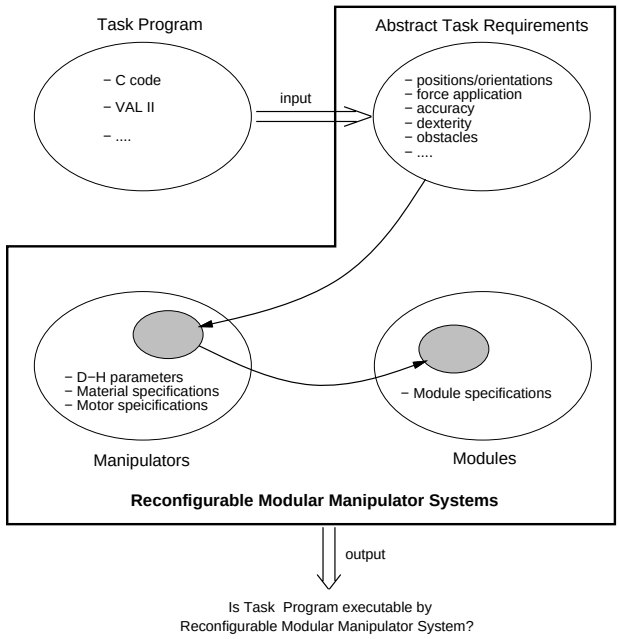

Figure 1: *Definition of a general purpose manipulator.*

also the notion of the user writing device (or manipulator) independent code. The RMMS raises several theoretical issues and it is our aim to address one of these in this paper. Specifically, we describe a design methodology that accepts a task specification as its input, determines a kinematic configuration of the desired manipulator and selects the modules to create this manipulator.

In order to support the current practice of picking the best configuration amongst available robots, several expert systems have been built to aid the user or the applications development engineer [15]. A straightforward extension of this selection process has been the inclusion of the design of new manipulators, optimally suited for a specific application [1, 16]. A totally different approach to the robot design problem finds its roots in simulation. A variety of commercial robot simulation packages are currently available [5, 22], providing designers with convenient tools to quickly check the implications of different design decisions. In general, however, these simulation packages still require a human to make the design decisions. Finally, a third way of dealing with the problem of robot design, has grown out of the field of mechanism design [13, 19]. Unlike the rule based expert systems, these programs are algorithmic in nature. Commonly, the design process is subdivided in two stages: form synthesis and dimensional synthesis. The first stage is usually performed by searching over the set of feasible mechanism types, while the second stage consists of optimizing the set of dimensional parameters.

The approach we propose in this paper differs from the methods listed above, because we are specifically interested in modular manipulators. The interest in modular manipulators has grown steadily over the last decade [6, 24], and several related research issues have been addressed [2, 3, 7, 10, 14, 18]. However, the problem of determining the modular configuration optimally suited for one specific task, has never been addressed before to the best of our knowledge. In this paper, we investigate modular design from kinematic task requirements. These requirements affect only the kinematic structure of the manipulator, while dynamic requirements affect both its kinematic and dynamic structure. Examples of kinematic requirements are work space volume, maximum reach, and maximum positional error. Examples of dynamic requirements are maximum pay-load, maximum joint velocities, and maximum joint accelerations. Just as task requirements can be classified as kinematic or dynamic requirements, the design procedure can also be split into two parts: kinematic design and dynamic design [10]. Kinematic design determines the kinematic structure of the manipulator, while dynamic design determines the dynamic configuration. However, the dynamic design may require a change in kinematic structure, and thus a few iterations may be necessary to find a manipulator that satisfies both kinematic and dynamic requirements.

In the first part of this paper, we only consider reachability, joint limit, obstacle avoidance, and measure of isotropy requirements. A numerical procedure is proposed which determines a modular assembly configuration that meets all the requirements. In the second part, we focus our attention on one additional requirement, namely, fault tolerance. Recently, fault tolerant (or failure tolerant) robotics has been the subject of several publications [11, 23], in which different aspects of the problem are addressed. Visinsky et al. [23] propose a framework to include failure detection in fault tolerant robot systems. Lewis and Maciejewski [11], on the other hand, discuss the importance of the controller and the redundancy resolution. In this paper,

we focus our attention on the design of fault tolerant manipulators. We define fault tolerance as the ability to continue the performance of a task even after immobilization of a joint due to failure. Several properties of fault tolerant manipulators are discussed and are illustrated with examples.

2 Kinematic Design: Preliminary Results

2.1 Problem Statement

The problem solved in this section is the determination of a modular assembly configuration, that satisfies all the kinematic task requirements. These requirements are that the manipulator must be able to reach a specified set of positions/orientations, $\mathbf{p}_j$, (reachability requirement), without violating the motion constraints of the joint modules (joint limit requirement), and without colliding with any parallelepiped-shaped obstacles in the work space (obstacle avoidance requirement). Moreover, at the positions/orientations, $\mathbf{p}_j$, the measure of isotropy, must be larger than a user specified minimum (measure of isotropy requirement).

In Section 2.2 and 2.3, we develop a numerical procedure to solve this design problem. To facilitate the implementation of our approach, we consider six types of modules, as shown in Fig. 2. The choice of these specific modules guarantees a simple conversion from the module dimensions and orientations into the Denavit-Hartenberg (D-H) parameters of the resulting manipulator (a set of 3 D-H parameters per degree-of-freedom, determines unambiguously the kinematic structure of any serial link manipulator). It has been shown by Kelmar and Khosla [7] that this conversion can be achieved for modules of arbitrary geometry. The actual number of different modules considered for the design can be far larger than six, due to variations in the parameterized dimensions, and our design method is general enough to allow for this.

We also require that the robot base be fixed and known, that the first joint module be of type 0 or 1 (i.e. the first joint axis is vertical), and that the last module be a wrist with three axes intersecting at a point. These restrictions result from our implementation of the inverse kinematics and can be relaxed by using iterative solutions to the inverse kinematics problem, as proposed in [2]. Also, the requirement that the robot

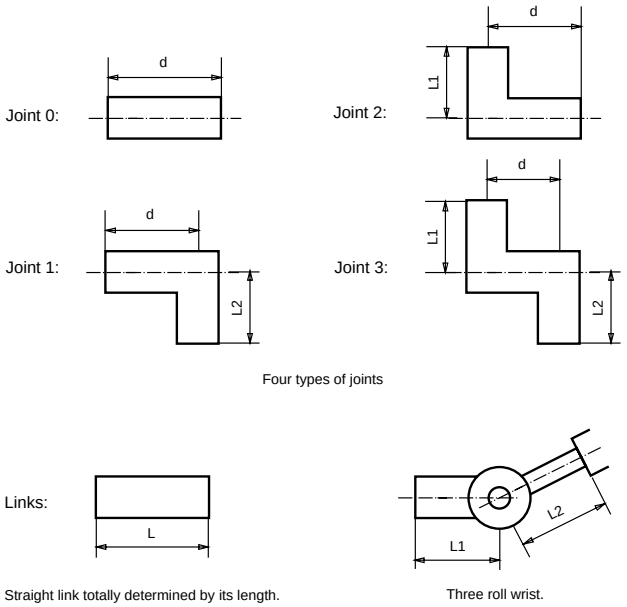

Figure 2: *The types of manipulator modules that are considered in the design procedure.*

base be fixed and known, can be relaxed as was shown by Kim [8], who addressed the problem of kinematic synthesis and base position synthesis simultaneously.

Finally, we would like to point out that this design problem can possibly have more than one solution. Consider the design of a 2-DOF planar manipulator, with link lengths L_1 and L_2, satisfying the task requirement that the manipulator should be able to reach a point located behind an obstacle without violating the joint limits, as is illustrated in Fig. 3. The region of the (L_1, L_2)-plane containing the solutions is bounded by the curves labeled c, d and e. All the manipulators inside this region satisfy all the design requirements and, therefore, are all equally good with respect to these requirements.

2.2 Solution Approach

In this section, we evaluate different approaches to the problem of determining the modular configuration, given some kinematic task specifications. The problem can be interpreted as a mapping from task specifications into constraints in the modular configuration space, as is shown in Fig. 1. This mapping is nontrivial due to the highly nonlinear character of the kinematic

Figure 3: *A two-DOF planar design example with solution.*

relations and due to the complexity of the task specifications. Krishnan [10], therefore, suggested to solve the inverse problem first, namely, to analyze which task requirements are satisfied by a given modular configuration. This information is stored in lookup tables, which can then be used in a search procedure. One obvious disadvantage to this approach is the combinatorial explosion in the number of different configurations. Let the number of different modules available be N, and let R be the number of relative orientations in which one module can be mounted on the previous module. The total number of configurations that can be obtained from this set of modules is:

$$Num = \sum_{i=1}^{N} R^{(i-1)} \frac{N!}{(N-i)!} \tag{1}$$

This approach would therefore require a very large amount of memory storage for the lookup tables. Also, adding a new module requires that all the lookup tables be updated.

A different approach is to first design a manipulator defined by a set of continuously varying D-H parameters, as proposed in [17], and then transform this design into a modular configuration. The main problem here is the discretization of the continuous solution. As is known from integer programming, simply taking the discrete configuration nearest to the continuous solution might result in an infeasible solution. Therefore, we suggest working directly in the modular configuration space. Of course, an exhaustive search in this space suffers from combinatorial explosion in much the same way the look up table approach does. However, the efficiency of the search procedure can be improved drastically by 'guiding' the search to the most promising regions of the search space. Instead of answering the question whether a certain modular design meets all the task requirements with a simple 'yes' or 'no', we estimate the 'goodness' or 'badness' of the design, i.e., "How far are we away from a solution?" Guiding the search then means focusing the search effort on directions of decreasing 'badness'. This approach is usually referred to as a heuristic search technique [20], because in general, it is impossible to compute the 'badness', or the distance to the nearest solution, exactly. The heuristic function only estimates this distance so that it is possible that, locally, the heuristic decreases even though the actual distance to a solution increases. This corresponds to a local minimum in optimization terminology. To overcome this inadequacy, we have to employ a search method, such as simulated annealing, that allows for local hill climbing.

Simulated annealing was first proposed by Kirkpatrick [9] as a combinatorial optimization algorithm. The method is a random iterative improvement algorithm with the modification that, under certain conditions, an increase in the heuristic function is accepted (In order to be compatible with the standard terminology in discussions of simulated annealing, we use the term objective function instead of heuristic function, henceforth). A new trial configuration is gener-

ated randomly in the neighborhood of the current configuration. The condition for acceptance of this trial configuration is:

$$\begin{cases} \Delta F_{obj} \leq 0 & \Rightarrow \text{accept} \\ \exp(-\Delta F_{obj}/T) > \text{random}[0,1) & \Rightarrow \text{accept} \end{cases} \quad (2)$$

which depends on a control variable, T, the temperature. The algorithm is started at a high temperature for which most new configurations are accepted. After each iteration the temperature is decreased until no new acceptable configuration can be found. The search is then frozen. We adapted this basic algorithm to include the special properties of our objective function. In particular, the algorithm is stopped when a new trial configuration has an objective function value equal to zero, even if the search is not yet frozen. We know that a configuration with a 'badness' of zero satisfies all the design requirements.

2.3 Computation of the Objective Function

The goal in this section is to find an objective function which is zero when all the design specifications are satisfied and which is otherwise proportional to the amount of violation of these specifications. Minimizing the objective function by simulated annealing corresponds then to a search, guided towards the most promising regions of the search space. For a given modular configuration, the corresponding objective function can also be interpreted as a penalty for violating certain task specifications. The goal of the search is then to find a configuration with zero penalty.

We now propose a methodology for constructing a penalty function. Let us first define some terminology. A configuration is the set of D-H parameters which determines unambiguously the kinematic structure of a modular manipulator configuration. A posture is the position of a manipulator corresponding to a specific set of joint angles. A task point is a specified position/orientation of the end effector that the manipulator must be able to reach without violating the other task requirements.

By taking a closer look at the task requirements, one notices that all the requirements are defined for a specific configuration in a specific posture reaching for a specific task point. The penalty for such a posture should be defined such that, if any single requirement is not satisfied, the penalty for the posture is positive. This can be achieved by defining a nonnegative penalty for each task requirement, as described in [17], and summing these penalties for a posture:

$$P_{post} = \sum_{requirements} P_{req} \quad (3)$$

The task penalty is now defined as the minimum over all the posture penalties, so that it is zero when all the task requirements are satisfied for at least one posture:

$$P_{task} = \min_{post} P_{post} = \min_{post}(\sum_{req} P_{req}) \quad (4)$$

Finally, the total penalty of a manipulator configuration is given by the sum of all the task penalties:

$$P_{conf} = \sum_{tasks} P_{task} = \sum_{tasks} (\min_{post}(\sum_{req} P_{req})) \quad (5)$$

2.4 Example

The example solved in this section is the design of a seven degree-of-freedom manipulator that is able to reach eight different task points, located in an environment which includes five obstacles. The joint modules have a limited motion range and the task points must be reached in a posture with a measure of isotropy of at least 0.6. It is also required that the manipulator consists of a subset of the twenty different modules (4 joints, 1 wrist, 16 links: we also consider a link of length zero). It is assumed, at this point, that an unlimited number of each type of module is available, so that a design which includes the same module type several times is acceptable. All the task requirements are summarized in the input file in Fig. 4.

A quick calculation gives us an idea of the extent of the search space. A 7-DOF manipulator consists of five links, four joints and one wrist, and is further determined by five angles, specifying the relative orientations of the joint modules. Taking into account the restrictions, that the first joint module must be of type zero or one and the last module must be a wrist, the number of configurations in the search space equals:

$$\begin{aligned} &(\#\text{links})^5(\#\text{joints})^3(\#\text{joints of type 0 or 1}) \\ &(\#\text{wrists})(\#\text{relative orientations})^5 \\ &= 16^5 \cdot 4^3 \cdot 2 \cdot 1 \cdot 8^5 \approx 4.4 \times 10^{12} \end{aligned} \quad (6)$$

```
7                               # number of degrees of freedom
3                               # number of dimensions

8                               # number of relative orientations
16                              # number of link modules
#number        |length
#---------------------
0               0.0
1               0.1
2               0.2
3               0.3
4               0.4
5               0.5
6               0.6
7               0.7
8               0.8
9               0.9
10              1.0
11              1.1
12              1.2
13              1.3
14              1.4
15              1.5

4                               # number of joint modules
#number|type   |l1     |d      |l2     |th_min |th_max
#-------------------------------------------------
0       0       0.0     0.1     0.0     -150.0  150.0
1       1       0.1     0.1     0.0     -150.0  150.0
2       2       0.1     0.1     0.1     -150.0  150.0
3       3       0.1     0.1     0.1     -150.0  150.0

1                               # number of wrist modules
#number|l1     |l2     |min1   |max1   |min2   |max2   |min3   |max3
#-------------------------------------------------------------------
0       0.1     0.05    -100.0  100.0   -150.0  150.0   -266.0  266.0


0.6                             # mi_min: min measure of isotropy.
8                               # num_points: number of points
#xpos           |ypos           |zpos   |Xrot   |Yrot   |Zrot
#-----------------------------------------------------------
0.5             0.5             1.0     0.      0.      90.
0.5             0.0             0.5     0.      90.     0.
0.5             0.0             1.5     0.      -90.    0.
0.5             -0.5            1.0     0.      0.      -90.
1.5             0.5             1.0     0.      0.      90.
1.5             0.0             0.5     0.      90.     0.
1.5             0.0             1.5     0.      -90.    0.
1.5             -0.5            1.0     0.      0.      -90.

5                               # num_obst: number of obstacles.
#xpos  |ypos   |zpos   |Xrot   |Yrot   |Zrot   |xdim   |ydim   |zdim
#-------------------------------------------------------------------
2.      -2.     0.425   0.      0.      0.      3.7     3.7     0.85
2.      2.      0.425   0.      0.      0.      3.7     3.7     0.85
2.      -2.     1.425   0.      0.      0.      3.7     3.7     0.85
2.      2.      1.425   0.      0.      0.      3.7     3.7     0.85
-1.     0.      1.      0.      0.      0.      1.9     4.      2.
```

Figure 4: *The input file of the 7-DOF example.*

	link#	angle#	joint#
1	10	4	1
2	14	4	3
3	3	3	0
4	12	6	3
5	14	0	—

Table 1: *Module number of 7-DOF design.*

Starting from a random initial guess, the simulated annealing algorithm evaluated on the average only about 2700 configurations before finding a solution. One of these solutions is tabulated in Table 1 and Table 2. It is a SCARA-like manipulator with a nearly spherical joint at the end of the second link. The offset

DOF	d_i	a_i	α_i
1	1.1	1.6	180^o
2	0.1	0.0	90^o
3	1.8	0.0	-90^o
4	0.1	0.0	90^o
5	1.6	0.0	90^o
6	0.0	0.0	-90^o
7	0.05	0.0	—

Table 2: *D-H parameters of 7-DOF design.*

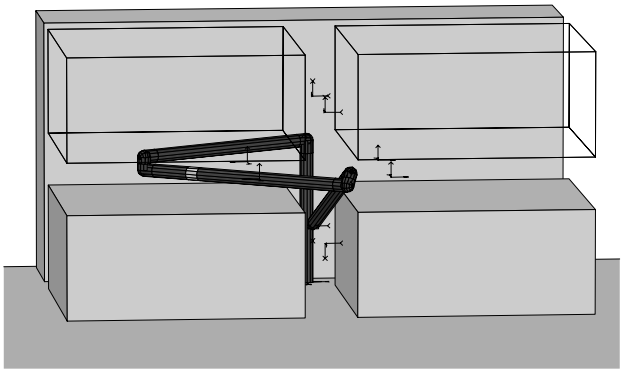

Figure 5: *The manipulator reaching point two while avoiding all the obstacles.*

along the first axis is 1 meter and the first twist angle is 180^o, so that the first and second link move in a horizontal plane exactly between the four obstacles. Because of the spherical joint, link 3 can move either in a horizontal or a vertical plane, so that all the task points can be reached without hitting any obstacles, as shown in Fig. 5.

3 General Purpose Fault Tolerant Manipulators

In the rest of this paper, we focus our attention on one additional task requirement, namely, fault tolerance. To set the stage for our development, we define the following properties of fault tolerant manipulators [2]:

- **General Purpose Fault Tolerant Manipulator:** An n-DOF manipulator that will still be able to meet the task specifications, even if any one or more of its joints fail and are frozen at any arbitrary joint angles.

- **k-Reduced Order Derivative (k-ROD):** When k joints of an n-DOF manipulator fail, the effective number of joints is $(n-k)$. The resulting faulty manipulator is called a k-reduced order derivative.
- **Order of Fault Tolerance:** An n-DOF manipulator is fault tolerant of the k-th order, if and only if all k-reduced order derivatives can still perform the specified task. We call the manipulator k-fault-tolerant.
- **Fault Tolerant Work Space (FTWS):** The fault tolerant work space of a k-fault tolerant manipulator is the set of points reachable by all possible k-reduced order derivatives.

These definitions differ from the concept of fault tolerance as proposed by Maciejewski [12]. Instead of attributing the property of fault tolerance to a *manipulator*, he quantifies a measure of fault tolerance for a manipulator *posture* and describes a technique to determine the optimal fault tolerant posture, based on the singular value decomposition of the Jacobian matrix. If a joint fails in this optimal posture, the resulting reduced order derivative will have maximum possible dexterity. However, a failure at a different angle may make the execution of the task impossible.

In the rest of this section, if no specific task is mentioned, it is assumed that the task consists of reaching a nonzero volume of points in the task space, i.e., an m-dimensional manifold in the m-dimensional task space. A manipulator that can only reach a manifold of dimension lower than m in a fault tolerant way, is considered not to be fault tolerant.

4 Properties of General Purpose Fault Tolerant Manipulators

4.1 Existence

A general purpose manipulator has six DOFs which allow it to position its end effector in an arbitrary position and orientation anywhere in its work space. An obvious way to make this manipulator fault tolerant is to design every joint with a redundant actuator. If one of the actuators of the resulting $2n$-DOF fault tolerant manipulator were to fail, the redundant actuator could take over and the manipulator would still be functional. Similarly, a k-fault tolerant manipulator can be constructed by duplicating every DOF k times, resulting in a $(k+1)n$-DOF manipulator.

4.2 Boundary of the Fault Tolerant Work Space

In this section, we show that a boundary point of the FTWS is a critical value. Consider a k-fault tolerant planar manipulator, $\mathcal{M}$. A boundary point, $\mathbf{p}_b$, of the FTWS has to be an element of the boundary of the work space of at least one ROD, $\mathcal{M}^*$, obtained by freezing k joints of $\mathcal{M}$. Indeed, if $\mathbf{p}_b$ were an interior point of the work spaces of all RODs, then it would by definition be an interior point of the FTWS and not a boundary point. The Jacobian of $\mathcal{M}^*$, J_{M^*}, can be obtained from the Jacobian of $\mathcal{M}$, J_M, by deleting the columns corresponding to the frozen DOFs. Because $\mathbf{p}_b$ is a boundary point of the work space of $\mathcal{M}^*$, the Jacobian of $\mathcal{M}^*$ at $\mathbf{p}_b$ is singular. We prove now that J_M is singular too. Suppose that J_M were non-singular, then at least one of the columns corresponding to a frozen DOF would be outside the column space of the singular matrix, J_{M^*}. Physically this means that a small change in the angle of that frozen DOF would cause the end effector of $\mathcal{M}$ to move in a direction with a component perpendicular to the boundary of the work space of the ROD, $\mathcal{M}^*$, as illustrated in Fig. 6. The ROD with this new frozen angle would be unable to reach the point, $\mathbf{p}_b$. As a result, $\mathbf{p}_b$ would be outside the FTWS, contradicting the fact that $\mathbf{p}_b$ is a boundary point of the FTWS. Thus, J_M is singular and $\mathbf{p}_b$ is a critical value.

Consequently, the FTWS is bounded by critical value manifolds. For planar positional manipulators, the critical value manifolds are concentric circles, and the FTWS is an annulus with inner radius R_{min}^{FTWS} and outer radius R_{max}^{FTWS}.

4.3 Required Degree of Redundancy

In Section 4.1, it is shown that, in general, kn redundant DOFs—i.e. $(k+1)n$ DOFs in total—are sufficient to achieve k-th order fault tolerance. For planar positional manipulators, however, we prove that $2k$ DOFs are also necessary for k-th order fault tolerance.

The proof shows that $(2k+1)$ DOFs (or $2k-1$ redundant DOFs) are insufficient, by finding a lower bound for R_{min}^{FTWS} and an upper bound for R_{max}^{FTWS} that are

Figure 7: *An upper bound for R_{max}^{FTWS} and a lower bound for R_{min}^{FTWS}.*

From Equation (7) and Equation (9), it follows that at best

$$R_{max}^{FTWS} = R_{min}^{FTWS} \tag{10}$$

resulting in a one-dimensional FTWS. Therefore, a $(2k+1)$-DOF manipulator cannot be fault tolerant ■

4.4 Including Orientation

Thus far, we have only considered planar positional manipulators. The results for positional manipulators can be easily extended to the case in which orientation is considered also, by converting the orientational problem into an equivalent positional problem:

> An n-DOF manipulator, $\mathcal{M}$, is k-fault tolerant with respect to a set of points, $W = \{(x_i, y_i, \varphi_i)\}$, if and only if:
>
> 1. the positional manipulator, $\mathcal{M}'$, obtained from $\mathcal{M}$ by deleting its last link, l_n , is k-th order fault tolerant with respect to the set of points $W' = \{(x_i - l_n \cos\varphi_i, y_i - l_n \sin\varphi_i)\}$

DOF	d_i	a_i	α_i
1	0	1	90^o
2	a	1	0^o
3	−a	1	90^o
4	b	1	0^o
5	−b	1	—

Table 3: *D-H parameters of a 5-DOF first order fault tolerant spatial manipulator without orientation*

2. $\mathcal{M}'$ is $(k-1)$-fault tolerant while reaching the points in W' in any direction.

The positional manipulator, $\mathcal{M}'$, needs at least $(2k+2)$ DOFs to be k-fault tolerant with respect to W'; therefore, the manipulator $\mathcal{M}$ needs at least $(2k+3)$ DOFs. Now, consider a $(2k+3)$-DOF manipulator with the first links having length, l, and the last link having length zero. It is easy to verify that this manipulator's k-th order FTWS is:

$$W = \{(x, y, \varphi) \mid \sqrt{x^2 + y^2} \leq 2l \text{ and } \varphi \in [0, 2\pi)\} \quad (11)$$

Thus, $(2k+3)$ DOFs are necessary and sufficient for k-th order fault tolerance of planar manipulators when orientation is included

This result and the result obtained in Section 4.3 can be summarized in the following theorem:

Theorem:
For planar manipulators, $2k$ redundant DOFs are necessary and sufficient for k-th order fault tolerance.

4.5 Spatial Fault Tolerant Manipulators

For planar fault tolerant manipulators, we were able to prove that $2k$ is the required degree of redundancy. The proof was based on geometric work space analysis. However, the geometric analysis becomes too complex for spatial manipulators, especially since we are dealing with redundant manipulators. Therefore, we will demonstrate some properties of spatial fault tolerant manipulators using two examples.

As a first example, consider a 5-DOF spatial positional manipulator. Its D-H parameters are listed in Table 3. This manipulator is first order fault tolerant, and because of its simple kinematic structure, an analytic expression for the boundary of the FTWS can be

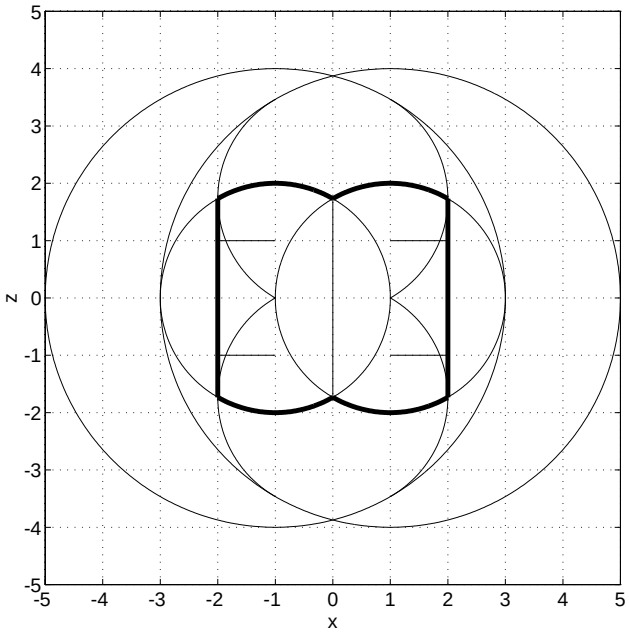

Figure 8: *A cross section of the boundary of the FTWS of a 5-DOF spatial manipulator (bold) as part of its critical value manifolds.*

derived. The FTWS is symmetric with respect to the first axis. A cross section (the X-Z plane), as shown in Fig. 8, can be described by two segments of a circle with radius 2 and center at $(x = 1, z = 0)$, and a straight line from $(x = 2, z = \sqrt{3})$ to $(x = 2, z = -\sqrt{3})$. An important property of this FTWS is that it does not have any holes or a central void, so that the FTWS of the same manipulator scaled by any factor, $\lambda > 1$, contains the original FTWS. As a result, this fault tolerant manipulator can be used as a design template. Any specified set of points can be reached in a first order fault tolerant way by a scaled version of the template.

In Section 4.2, it is shown that the boundary of the FTWS of a planar manipulator coincides with its critical value manifolds. Fig. 8 demonstrates that this property also holds for the 5-DOF spatial manipulator considered in this example. The critical value manifolds are computed using the algorithm described in [4] and are depicted in a solid line. The bold part of the critical value manifolds is the boundary of the FTWS.

As a second example, consider an 8-DOF manipulator, with D-H parameters listed in Table 4. It is

DOF	d_i	a_i	α_i
1	0	1	90^o
2	a	1	0^o
3	−a	1	90^o
4	b	1	0^o
5	−b	0	90^o
6	1	0	90^o
7	0	0	90^o
8	0	0	—

Table 4: *D-H parameters of an 8-DOF first order fault tolerant spatial manipulator with orientation*

the same manipulator as in example one, with a zero-length 3-roll-wrist added at the end. Using a Monte-Carlo method, it has been determined that this manipulator is first order fault tolerant while reaching all the points in the FTWS of example one, *in any direction*. This property can be demonstrated with the following arguments. When one of the first five DOFs fails, the manipulator can still reach any position in the FTWS (because the 5-DOF positional manipulator is fault tolerant) and can take any orientation at this position using the intact 3-roll-wrist. When one of the DOFs in the wrist fails, we are left with a 7-DOF manipulator which has enough orientational capabilities to reach any point in the FTWS in any orientation. Consequently, one could call this the *dextrous* FTWS. Since there are again no holes or voids in the FTWS, this manipulator can also be used as a design template.

Finally, one should notice that both examples have only two redundant DOFs, which seems to indicate that the theorem in Section 4.4 is extendible to spatial manipulators.

5 Task Specific Fault Tolerant Manipulators

In the previous section, we considered the design of fault tolerant manipulators for general use. We proved that two redundant DOFs are necessary for first order fault tolerance. However, as we will show in this section, a simpler kinematic structure is often sufficient when one specific task is considered. This implies, of course, that a different kinematic structure might be needed for every task—a disadvantage that can be alleviated by the use of a reconfigurable modular manipulator system.

We modify the definition of fault tolerance to include task specificity:

- **Task Specific Fault Tolerant Manipulator**: A manipulator is 1-fault-tolerant with respect to the task of following the Cartesian trajectory, $\mathbf{p}(t)$, if there exists a fault tolerant trajectory in joint space, $\theta(t)$, that maps into $\mathbf{p}(t)$, and which is such that when an arbitrary joint, j, were frozen at an instant, ${}^f t$, an alternate trajectory, $\theta(t, j, {}^f t)$, could be followed to complete the task.

The difference between this definition and the one for general purpose fault tolerance, is that we no longer require that a point be reachable when a joint fails at an arbitrary angle, but only at an angle that occurred previously in the fault tolerant trajectory. Under this assumption, k-fault tolerance can be achieved with only k redundant DOFs.

Consider the task of reaching all the points in an ϵ-neighborhood, $B(\mathbf{p}, \epsilon)$, of the point $\mathbf{p} \in \Re^m$. Suppose that $\mathbf{p}$ can be reached by an n-DOF manipulator in a posture $\theta \in T^n$. If the posture, θ, is non-singular, then there exists an $\epsilon > 0$, such that the manipulator can reach any point in $B(\mathbf{p}, \epsilon)$. However, for k-fault-tolerance, any point in $B(\mathbf{p}, \epsilon)$ needs to be reachable even when k of the joints of the manipulator are frozen. This is possible if and only if the Jacobians of all k-ROD in the posture θ have at least rank m. We call such a posture, θ, locally fault tolerant. The Jacobian of a k-ROD can be obtained by deleting the columns of the fault-free Jacobian corresponding to the frozen DOFs; its dimensions are $m \times (n - k)$. In order for the rank to be at least m, n has to be larger than or equal to $(m+k)$, i.e., the manipulator needs to have at least k redundant DOFs. When the rank of a k-ROD Jacobian is less than m, the robot is in an internal singularity; otherwise, it is in a locally fault tolerant posture. The locus of internal singularities is a set of $(m + k - 1)$-dimensional surfaces in T^n; or $(n - 1)$-dimensional surfaces, when $n = m + k$. Thus, nearly all postures of a manipulator with k redundant DOFs are locally k-fault tolerant.

We now extend this result to larger trajectories for which a global condition has to be satisfied. This can best be illustrated with an example. Consider a 3-DOF planar manipulator with normalized link lengths

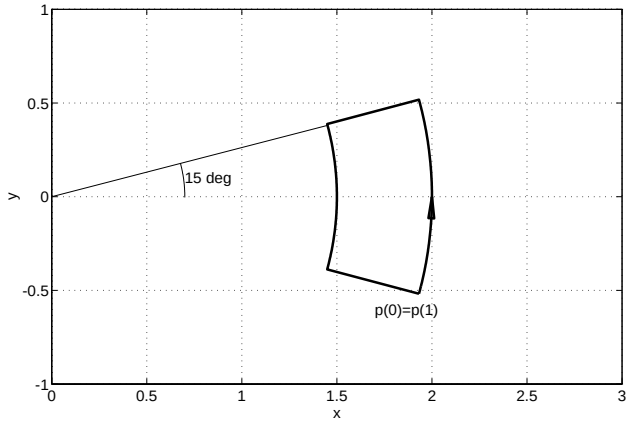

Figure 9: *The trajectory of the example of task specific fault tolerance.*

of 1. We want to determine whether this manipulator is able to execute the task of following the trajectory shown in Fig. 9, in a 1-fault tolerant way. The trajectory can be parameterized as $\mathbf{p}(\alpha)$ with $0 \leq \alpha \leq 1$. We assume for this example that $\mathbf{p}(0) = \mathbf{p}(1)$, and that the task is repeated from the beginning as soon as the end is reached. For every α, one can compute the preimage of $\mathbf{p}(\alpha)$. Since the manipulator has one degree of redundancy, the preimage of every $\mathbf{p}(\alpha)$ is a one-dimensional subset of T^n, and can be parameterized as $\theta = f(\mathbf{p}(\alpha), \beta)$ with $\beta \in T^1$. The continuous function, f, describes a 2-dimensional surface in T^3, as is shown in Fig. 10. Any joint trajectory that follows the specified Cartesian trajectory, $\mathbf{p}(\alpha)$, can be formulated as $\beta(\alpha)$, or $\theta(\alpha) = f(\mathbf{p}(\alpha), \beta(\alpha)$. According to the definition of task specific fault tolerance, the manipulator is fault tolerant if and only if a fault tolerant trajectory, $\theta(\alpha)$, can be found. It is clear that every posture of a fault tolerant trajectory, $\theta(\alpha)$, has to be locally fault tolerant. However, this requirement is not sufficient because a fault at a point, $\mathbf{p}(\alpha_1)$, might make another point, $\mathbf{p}(\alpha_2)$, unreachable, even when the posture $\theta(\alpha_1)$ is locally fault tolerant. Therefore, one should exclude as possible postures for a fault tolerant trajectory not only internally singular postures, but also postures that, in the case of failure, would cause an internal singularity elsewhere along the trajectory. For our example, the set of acceptable postures is shown in Fig. 11.

A fault tolerant trajectory exists when a continuous function, $\beta(\alpha)$ with $0 \leq \alpha \leq 1$, can be found for

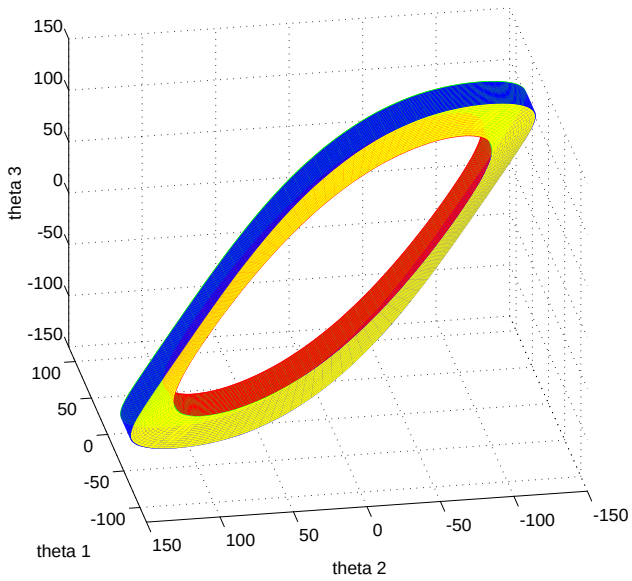

Figure 10: *The preimage of a trajectory.*

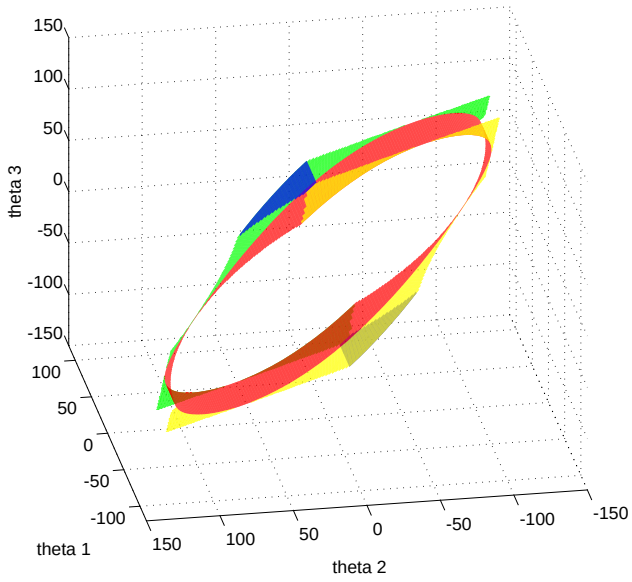

Figure 11: *The set of acceptable points for a fault tolerant trajectory.*

which all postures, $\theta = f(\mathbf{p}(\alpha), \beta(\alpha))$, are acceptable, i.e., satisfy the global fault tolerance condition. That such a trajectory exists for our example can be concluded from Fig. 11. The same conclusion follows more clearly from Fig. 12, in which only the postures of the 2-dimensional preimage of the trajectory are represented.

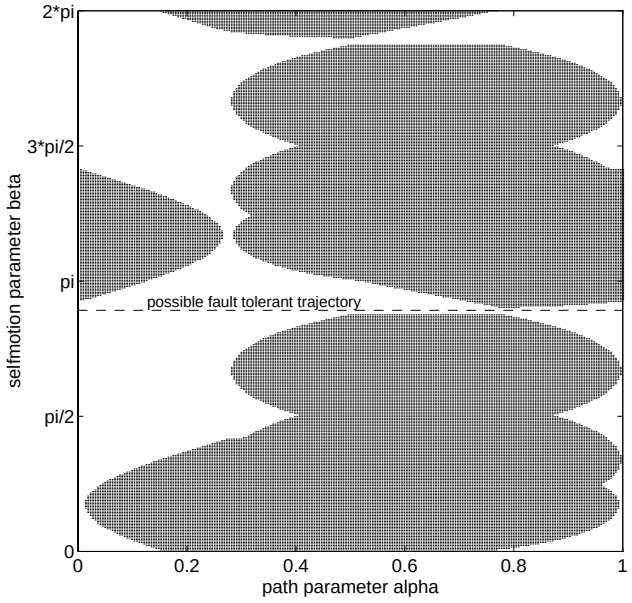

Figure 12: *A possible fault tolerant trajectory. Regions of unacceptable postures, (α, β), are marked in gray.*

The gray area is the set of postures that, in the case of a fault, would cause an internal singularity somewhere along the trajectory; these are the unacceptable postures. The manipulator is fault tolerant if a continuous trajectory, $\beta(\alpha)$, can be found that does not pass through any gray areas. One possible trajectory is shown in dashed line.

The conclusion of this section is that, if a fault-free trajectory is chosen carefully, one can possibly achieve first order fault tolerance with only one redundant DOF.

Our current research deals with the problem of translating the results of the analysis of task specific fault tolerance into specific design rules for kinematic structures of manipulators.

6 Summary

In this paper, we developed an approach for determining a configuration for a reconfigurable modular manipulator able to fulfill a specific task. We considered tasks that included four kinematic requirements: reachability, joint limits, obstacle avoidance and measure of isotropy. The attribution of a penalty to each manipulator configuration, enabled us to reduce the search effort drastically, by guiding the search to the most promising regions of the assembly configuration space. Local minima in the penalty were avoided by using simulated annealing as a search algorithm. We also defined a property of a small class of redundant manipulators, called fault tolerance. We proved the existence of general purpose fault tolerant manipulators, obtained through joint duplication. When no joint limits are considered, we proved analytically that, $2k$ redundant DOFs are necessary and sufficient for general purpose fault tolerance of planar manipulators, and that the boundary of the FTWS consists of critical values. Also, 8- and 5-DOF design templates were introduced, for spatial general purpose fault tolerant manipulators with and without orientational capabilities, respectively. Finally, we demonstrated that a task specific 1-fault tolerant manipulator possibly only needs one degree of redundancy, versus the two needed for general purpose manipulators. This simplification of the kinematic structure can be achieved at the cost of having to reconfigure the manipulator for every task.

Acknowledgment

This research was funded in part by DOE under grant DE-F902-89ER14042, by Sandia under contract AC-3752-A, by the Department of Electrical and Computer Engineering, and by The Robotics Institute, Carnegie Mellon University.

References

[1] V. P. Agrawal, V. Kohli, and S. Gupta. Computer aided robot selection: the 'multiple attribute decision making' approach. *International Journal of Production Research*, 29(8):1629–1644, 1991.

[2] W. K. F. Au, C. J. J. Paredis, and P. K. Khosla. Kinematic design of fault tolerant manipulators. In *Proceedings of the Allerton Conference*, Urbana-Champagne, Illinois, October 2 1992.

[3] B. Benhabib, G. Zak, and M. G. Lipton. A generalized kinematic modeling method for modular robots. *Journal of Robotic Systems*, 6(5):545–571, 1989.

[4] J. W. Burdick. Kinematic analysis and design of redundant robot manipulators. Stanford Computer Science Report STAN-CS-88-1207, Stanford University, 1989.

[5] P. Fanghella, C. Gellatti, and E. Giannotti. Computer-aided modeling and simulation of mechanisms and manipulators. *Computer Aided Design*, 21(9):577–583, 1989.

[6] T. Fukuda, G. Xue, F. Arai, H. Asama, H. Omori, I. Endo, and H. Kaetsu. A study on dynamically reconfigurable robotic systems. assembling, disassembling and reconfiguration of cellular manipulator by cooperation of two robot manipulators. In *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS '91)*, pages 1184–1189, Osaka, Japan, November 3-5, 1991.

[7] L. Kelmar and Pradeep K. Khosla. Automatic generation of forward and inverse kinematics for a reconfigurable modular manipulator system. *Journal of Robotic Systems*, 7(4):599–619, 1990.

[8] J.-O. Kim. *Task Based Kinematic Design of Robot Manipulators*. PhD thesis, Carnegie Mellon University, The Robotics Institute, Pittsburgh, PA, August 1992.

[9] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[10] A. Krishnan and P. K. Khosla. A methodology for determining the dynamic configuration of a reconfigurable manipulator system. In *Proceedings of the 5th Annual Aerospace Applications of AI Conference*, Dayton, Ohio, October 23-27, 1989.

[11] C. L. Lewis and A. A. Maciejewski. Dexterity optimization of kinematically redundant manipulators in the presence of joint failures. *Computers and Electrical Engineering*, 20(3):273–288, 1994.

[12] A. A. Maciejewski. Fault tolerant properties of kinematically redundant manipulators. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 638–642, Cincinnati, Ohio, May 1990.

[13] S. Manoochehri and A. A. Seireg. A computer-based methodology for the form synthesis and optimal design of robot manipulators. *Journal of Mechanical Design*, 112:501–508, December 1990.

[14] S. Murthy, P. K. Khosla, and S. Talukdar. Designing manipulators from task requirements: An asynchronous team approach. In *Proceedings of the 1st WWW Workshop on Multiple Distributed Robotic Systems*, Nagoya, Japan, July 1993.

[15] O. F. Offodile, B. K. Lambert, and R. A. Dudek. Development of a computer aided robot selection procedure (carsp). *International Journal of Production Research*, 25:1109–1121, 1987.

[16] O. F. Offodile, W. M. Marcy, and S. L. Johnson. Knowledge base design for flexible assembly robots. *International Journal of Production Research*, 29(2):317–328, 1991.

[17] C. J. J. Paredis. An approach for mapping kinematic task specifications into a manipulator design. Master's thesis, Carnegie Mellon University, Electrical and Computer Engineering Department, Pittsburgh, PA, September 1990.

[18] C. J. J. Paredis and P. K. Khosla. Kinematic design of serial link manipulators from task specifications. *The International Journal of Robotics Research*, 12(3):274–287, June 1993.

[19] V. Potkonjak and M. Vukobratovic. Computer-aided design of manipulation robots via multiparameter optimization. *Mechanism and Machine Theory*, 18(6):431–438, 1983.

[20] E. Rich and K. Knight. *Artificial Intelligence*. series in artificial intelligence. Mc Graw-Hill Inc., New York, second edition edition, 1989.

[21] D. E. Schmitz, P. K. Khosla, and T. Kanade. The CMU reconfigurable modular manipulator system. In *Proceedings of the 19-th International Symposium and Exposition on Robots (ISIR)*, Australia, 1988.

[22] A. A. Tseng. Software for robotic simulation. *Advances in Engineering Software*, 11(1):26–36, January 1989.

[23] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Layered dynamic fault detection and tolerance for robots. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pages 180–187, Atlanta, GA, May 1993.

[24] R. H. Weston, R. Harrison, A. H. Booth, and P. R. Moore. Universal machine control system primitives for modular distributed manipulator systems. *International Journal of Production Research*, 27(3):395–410, 1989.