Optimal Path Placement for Kinematically Redundant Manipulators

James S. Hemmerle
EDRC
Carnegie Mellon University
Pittsburgh, PA 15213

Fritz B. Prinz
EDRC
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

The problem of path placement for redundant manipulators is discussed in this paper. This includes the problem of where to place the part and other components (such as other robots, tables or machining stations) relative to each other and how to resolve the redundancies of the system and process in an optimal fashion. The method developed involves assigning a variable to each degree of freedom of the transforms which define the system (i.e., to the system redundancies). These variables are then used to incorporate the path constraints of the robot into the cost function to be minimized. The resulting unconstrained minimization problem is numerically solved. This method is applied to a system of three cooperating robots and results will be presented.

1 Introduction

Redundancies in mechanisms have historically been resolved through pseudo-inverse (Jacobian) methods. But, these suffer from the shortcomings of the local nature of their planning strategy. Any decisions are based only on the path information in the immediate neighborhood. In any unstructured motion planning environment, this is necessary. However, in structured environments (such as manufacturing) one may take advantage of the global knowledge about the path and environment. Motion can be optimized by looking at the entire environment, simultaneously planning component relationships and the trajectories of the manipulator. The following paper describes such a strategy. Although this work was motivated by manufacturing tasks (and the vocabulary in the paper is geared towards this end), the strategies developed are domain independent.

1.1 Path Placement

Path placement denotes the problem of the locating of paths (which are attached to the part to be manufactured) relative to the robot. Some higher level decisions are obvious, such as, the paths must be placed within the robot envelope (so the end-effector (or tool) can follow the programmed paths). But this still leaves an infinite number of possibilities for path placement, with the quality of the finished part depending considerably on the one chosen. (For instance, some placements would cause the robot to move close to a singularity.) The problem becomes one of automatically selecting among these possibilities (optimizing) relative to a useful criterion.

1.2 Trajectory Planning

The robot's trajectory is a time history of its joint values as it moves. In the case of a non-redundant system, the trajectory is fixed when the part is placed (thereby locating the paths). Complications arise when the process, in conjunction with the robotic system, contains a redundancy, that is, when the number of independent variables (the robots joint angles) is larger than the constraints applied by the process (such as having the tool tip at a certain location with a certain orientation). This complication can occur from a variety of causes. For example, if the end-effector has a symmetric axis, such as in welding, there is a redundancy associated with that axis. The robot (if it has six degrees of freedom) can rotate the tool about that axis without affecting the quantities that the process dictates (the position of the torch tip and its pointing direction). But the angle the tool is rotated greatly affects the posture of the robot as it does its work and how its joint trajectories evolve along the path. The redundancies, by adding a multitude of possible solutions, can make otherwise impossible tasks possible. But again, the problem of how to automatically choose from among the possibilities arises.

This paper introduces a method capable of simul-
taneously placing the equipment and the part and resolving the redundancies of the system.

2 Literature Survey

Work in the area of optimal trajectory planning for redundant manipulators can be broken down into two main categories, local and global optimizations. The former uses only local information about the path (current joint positions and the direction to move), while the latter uses information gleaned from the entire path to be followed.

2.1 Local Optimizations

Local optimization strategies can be broken into two classes, one operating on the velocity level (Jacobian methods) and the other operating at the joint level (inverse kinematic methods). The Jacobian methods, which operate on a differential level, calculate the optimal trajectories via joint velocities. By prescribing the joint velocities along the path, one prescribes the joint trajectories (given an initial configuration). The inverse kinematic methods try for a direct conversion from workspace constraints to joints values, (analogous to the reverse solution for a non-redundant manipulator).

2.1.1 Jacobian Methods

Jacobian methods are based on the use of the matrix of the derivatives of the robot task constraints. If we let \( x = f(\theta) \) be the \( m \) constraints defined as functions of the \( n \) joint values, then the change in the system's joint values \( (\Delta \theta) \) are found by solving

\[
\dot{x} = J(\theta) \dot{\theta}
\]

where \( \dot{x} \) are the known constraint value changes and \( J \) is the \( m \times n \) matrix \( \left[ \frac{\partial f}{\partial \theta} \right] \) (the Jacobian).

If \( m = n \), the matrix \( J \) can be inverted and \( \dot{\theta} \) easily calculated. The problem with a redundant manipulator is that \( n > m \) and \( J \) is not square.

In the case of a redundant system, Whitney [1] proposed the use of a pseudo-inverse, which is a generalized inverse for a rectangular matrix. He suggested minimizing the joint velocities as a method of resolving the redundancy. This leads to an equation of

\[
\dot{\theta} = J^+ \dot{x} = J^T (J J^T)^{-1} \dot{x}
\]

(1)

Others have built on this solution in order to minimize other criteria \([2,3,4,5]\), giving rise to

\[
\dot{\theta} = J^+ \dot{x} - (I - J^+ J) \nabla H
\]

where \( H \) is the cost function to be minimized. The quantity \((I - J^+ J)\) projects \( VH \) into the Null space of the Jacobian. This causes the joints to move in such a way as to decrease \( H \) without changing the constraint values \( x \). In this case, you give up minimum joint velocity in order to fulfill some other goal, such as staying away from the joint limits or avoiding obstacles.

Hollerbach and Suh \([6,7]\) took the derivative of Equation 1 and solved for the acceleration giving

\[
\ddot{\theta} = J^+(\ddot{x} - J \ddot{\theta})
\]

They combined this with the robot's dynamic equations as a method to minimize the torque input to the robot. Although they found some improvements, they also noted stability problems with this method.

Recent work has aimed at lowering the computation time of pseudo-inverse methods \([8,9,10]\). These authors partition the Jacobian into invertible and rectangular sections and find the inverse and pseudo-inverse of these smaller matrices. In \([10]\) an analytic solution was found for a seven degree-of-freedom manipulator with the minimum joint velocity criterion.

An offshoot of the pseudo-inverse methods is the extended Jacobian methods. In these approaches, the Jacobian is augmented with rows that belong to the Null space of the Jacobian so that this extended Jacobian matrix is now invertible (or at least less rectangular). This is equivalent to adding equations of the form \( H(\theta) = 0 \) to the original problem statement \([11,12,13,14,15]\).

There are a number of problems associated with the Jacobian methods. Although the pseudo-inverse will give the minimum norm solutions for the joint velocity, it can still pass arbitrarily close to singularities giving arbitrarily high joint velocities \([12]\). Cyclic paths in the workspace do not lead to cyclic paths in the joint trajectories. This could cause the manipulator to drift into an undesirable configuration. Also, since the initial position is given and the path specified, these methods do not lend themselves well to the problem of part placement.

2.1.2 Inverse Kinematic Methods

Inverse kinematic methods operate under a different philosophy. These methods give a direct mapping from workspace coordinates to joints. The redundancies are removed by adding constraints that in effect make the system non-redundant. But instead of operating at the velocity level, as with extended Jacobian methods, and integrating to get the joint values, the joint values are calculated directly. Engeland [16]
used this method for the control of a robot on a positioning table. The positioning table accomplished the gross motion (which was commanded by a high level planner) while the robot accomplished the fine motion given the positioning motion. Others [17,18,19] added constraints on the end effector (such as the shape of the manipulability [20] envelope) and solved the non-redundant system of equations. Although closed formed solutions can be found to simple systems (such as three DOF manipulators in the plane [17]), larger systems would most likely have to be solved numerically, leading to something closely related to the extended Jacobian methods. Most numerical schemes would require the derivatives of the constraints leading to a matrix identical to that used in the extended Jacobian method.

A disadvantage of the inverse kinematic functions is that they give rise to a unique mapping between workspace variables and joint variables. It seems likely that one may want the configuration of the robot to depend not only on the current workspace variables, but also on their previous and future values. Because of the one-to-one mapping, large motions may be required.

2.1.3 Other Local Methods

A few other researchers have approached the problem of joint limit and obstacle avoidance through the use of feedback control. In these cases, the error in world space coordinates is driven to zero with the use of the Jacobian (as opposed to an "inverse" of the Jacobian) of the manipulator. Khatib [21] attempts to avoid obstacles by placing potential fields around the obstacles, which introduces a torque disturbance to drive the manipulator away from the obstacles. Others [22] append equations to the robot system matrix (in the manner of the extended Jacobian method) which constrain the robot from approaching obstacles (the offending link moves tangent to an envelope around the object) or its joint limits (the link locks in position).

2.1.4 Problem With Local Methods

The problem with the methods examined to this point is that they can only rely on local information. They are not capable of sacrificing local interests so that the trajectories are globally satisfactory. It is not possible to spread a locally difficult motion over the path because that motion is only seen locally. For example, consider the motion of a robot welding around a corner. Under local control, there would probably be a large joint motion required to go around the corner, but little motion before and after the corner. With a global optimization, some of that motion can be distributed before and after the corner, giving a globally smoother trajectory.

2.2 Global Optimizations

The price one must pay to use a global optimization scheme is the increase in computation time needed. Because of this, the global methods can not be used in real time applications. This is not a severe handicap in the manufacturing (structured) environment, since the trajectory planning needs to be done only once during the setup of the system. Instead of resolving the redundancies one point at a time, they must be resolved for each point on the path simultaneously. The cost function to be minimized becomes

$$\int C \, dx$$

where the integral is over the length of the path and $C$ is a measure of the cost of the motion such as the joint velocities, the time of the motion, the motor torque, or energy used.

There is a wide range of work in this area. The problem is usually set up as an optimal control problem or a calculus of variations problem. Either way, the solution gives rise to a non-linear two-point boundary value problem. Some authors numerically solved the resulting equations, while others parameterized the possible solutions and then found the optimal parameters.

The complete path of the tool was specified by those authors who solved the differential equations. Vukobratovic and Kircanski [23] and Bobrow, Dubowsky and Gibson [24] studied non-redundant manipulators. Since the joints values were calculated from the end effector's given path, these authors found the optimal joint velocities to minimize their cost functions. Suh and Hollerbach [7] studied a redundant manipulator with the condition that the initial configuration of the robot be given. Nakamura and Hanafusa [25] and Kazeronian and Wang [26] were more general in that they also solved for the optimal initial condition. Nakamura and Hanafusa did a one-dimensional grid search to find their needed initial condition, while Kazeronian and Wang used Powell's optimization [27] to find their solution.

Uchiyama, Shimizu and Hakomar [28] also worked with redundant manipulators in cases where the end effector path was given. Instead of solving the differential equations, they represented the unknown components of the trajectory as sixth order polynomials.
and then solved for the coefficients using a random walk method.

Other authors attempted to find the optimal path given desired starting and ending points of the end effector of the manipulator. Gilbert and Johnson [29] consider a non-redundant robot (initial and final configurations are therefore known) in their work. They represent the trajectories with fourth order polynomials and use a numerical scheme to solve for the coefficients. Chung, Dessa and DeSilva [30] also use polynomials in their work. They study redundant manipulators in which the initial configuration is given and represent the redundancies with polynomials. Nagurka and Yen's [31] approach allows unspecified initial and final configurations. They approximate the joint trajectories by a summation of polynomials (to satisfy the boundary conditions if given) and a finite Fourier-type series. They then optimize the unused polynomial coefficients and the coefficients in the series.

None of the above works considered the problem of placing the path and resolving the redundancy simultaneously.

However, because of the advantages global optimizations offers over local optimizations (albeit at a computation cost), a global type optimization will be used in this paper.

2.3 Path Placement

One article was found on path placement [32]. The system described takes points for a assembly task and an initial guess as to where to place it in the robot's workspace. The system then generates a search grid (within user specified limits) and rates each placement relative to their cost function, choosing the best as the final position. The system does not consider redundant manipulators and relies on the user's initial guess and a primitive search strategy.

3 Semi-Global Algorithm

The algorithm presented in this paper is a semi-global positional algorithm that resolves the path placement and redundancies by optimizing a cost function. By semi-global, we mean that points along the entire path are used, but not the whole path itself. In terms of Figure 1, this means that the cost function considers the node points, but not the path in between them. Of course, it is assumed that the nodes characterize the path well enough for the robot to perform its task. This approach is taken to decrease the computational effort while including the benefits of a global approach. In effect, we are discretizing the path into segments corresponding to the nodes. The equality constraints (such as position and orientation of the tool relative to the part) introduced by the path are satisfied at these nodes. Any inequality constraints are incorporated into the cost function.

The algorithm is positional in that it considers only the positional variable in the cost function and not their instantaneous changes. The cost functions are assumed to be functions of the robot joint values at the node points only. This allows a finite difference type approach to the joint velocities (and accelerations) if desired, but does not use the instantaneous joint velocities at the nodes.

The cost function is a measure of the "goodness" of the placement and trajectory. It is written as a function of the joint values, but through the forward transformation of the robot, it allows cost to be placed on kinematic quantities (such as distances from obstacles) in world space. As mentioned above, inequality constraints, such as joint or variable limits, are included here in terms of a penalty function.

The example system described in the next section will be used to explain the methodology used. Other systems and a more thorough development can be found in [34].

3.1 Example System

Consider first the part to be worked on (Figure 1). On this part, there are paths (numbering from 1 to \( n_p \)) to be followed by the robot's tool. These paths are described relative to a coordinate frame mounted on the part - the part frame. As the part is moved relative to the robot, this frame moves along with it so that the description of the path in that frame does not change.

Each path consists of a string of nodes (\( n_I \) nodes on
the $i^{th}$ path), which are the points that will be fed to the robot controller. Each of these nodes have a number of constraints (relative to some frame) associated with them. For example, in a welding operation with a symmetric tool, the constraints may be that the torch tip’s position and direction must be some value in the part frame and its direction must be vertical in the world frame.

Consider the system shown in Figures 2 and 3. The system consists of three six-degree of freedom robots, two robots which rigidly hold the part and a third robot which performs a welding operation. A schematic of this system can be seen in Figure 3. In this schematic, the ellipses represent relevant frames, while the connecting lines represent the transformations between these frames. The transforms contain knowledge of the degrees of freedom in the system, as well as the constraints. Some of the transforms are the same for every path point’s kinematic chain (global), while others vary along the path (local). In this system, transforms such as $T_{RB2}$ and $T_{PP}$ (the location of robot 2 and the location of the part frame relative to its faceplate) are global while transforms such as $F_{RB2}$ and $T_{P}$ (the forward solution of robot 2 and the path point itself) vary as the robot moves along the path. The overall objective is to keep the kinematic chain closed (satisfying the constraints) for each path point. One must choose the values for the transforms which have degrees of freedom associated with them, so they, along with the transforms associated with the constraints, form a closed chain. In effect, we desire a reverse solution for the system. Since there are redundancies in the system, and a unique reverse solution does not exist, an optimization will be performed to choose among the possible solutions.

The problem is a constrained nonlinear algebraic optimization problem of

$$\Theta_{ij}, T_G, T_{L_{ij}} \quad \text{min} \quad C(\Theta_{ij})$$

such that

$$X_{ij} = F(\Theta_{ij}, T_G, T_{L_{ij}}) \quad 1 \leq i \leq n_p, 1 \leq j \leq n_p$$

where $\Theta_{ij}$ are the joint variables for node $j$ on path $i$, $T_G$ are the global transforms, $T_{L_{ij}}$ are each path point’s local transforms, $X_{ij}$ are the constraints for each path point $F$ is the problem’s kinematic model (chain).

In order to solve this problem, the constraints are incorporated into the cost function. (The classical method of introducing Lagrange multipliers is not used because it leads to a much larger system of equations to be solved.) To achieve this, redundant variables are introduced. The redundant variables correspond to the excess degrees of freedom in each of the transforms in the problem. Let $t_i$ correspond to the $l$ degrees of freedom in the various global transforms and $\alpha_{ijk}$ correspond to the $k$ excess degrees of freedom for node $j$ on path $i$. Once these variables have given values, the various transforms are fixed and the values of the joint variables can be calculated.

So, Equations 2 and 3 can be combined into the equation:

$$\min_{t_i, \alpha_{ijk}} C(\Theta_{ij}(X_{ij}, t_i, \alpha_{ijk}))$$

With this reformulation, we now have an unconstrained nonlinear algebraic optimization over the redundant variables. To solve this problem, the derivatives with respect to these variables are set equal to zero,

$$\frac{\partial C}{\partial t_i} = \frac{\partial C}{\partial \alpha_{ijk}} = 0$$

which by the chain rule implies

$$\frac{dC(\Theta_{ij})}{d\Theta_{ij}} \left. \frac{\partial \Theta_{ij}}{\partial \alpha_{ijk}} \right|_{X_{ij}} = 0$$
The first term in the equations above is relatively straight forward. The second term is a bit more complicated. It gives the amount the system's joints must change in order to effect the changes in the variables \(a_{ijk}\) and \(t_i\), while still satisfying the constraints imposed. To find the value of the second term, one first transforms the motions caused by changing the variable values into motions of the robot's face plate frame. Then, one uses the Jacobian of the non-redundant robot to translate this face plate motion into the necessary joint motion. The exact procedure for this depends on what redundancies are present in the system (i.e. what excess variables are chosen).

### 3.2 Choosing Variables

Although the excess variables depend on the application, the methodology can be applied to a wide range of problems. The method applies to any system made up of kinematic chain components shown in Figure 4. The essential component for each loop in the chain is a non-redundant robot capable of moving in the task space. For general three dimensional tasks, this means having a six-degree of freedom robot. For a planar task, a three-degree of freedom robot is sufficient. The robot must be capable of making any differential movement \(dF_R\) necessary to compensate for a change in the redundant variables thus keeping the kinematic chain closed. (A side benefit of this choice is that the robot's analytic reverse solution, if it has one, may be used in the algorithm.)

The first three optional components correspond to the sources of the redundant variables. The global variables arise from redundancies associated with global transforms (e.g. table location, part location, or tool frame). A change in these variables causes a change of the global transform by an amount \(dT_G\), which shifts all path points (that is affects all the kinematic chains). Other kinematic linkages (a table, another robot, or extra joints added to a non-redundant robot) and the path point transforms are the source of local variables (redundancies that affect only one kinematic chain). The joint variables of the other kinematic linkages are designated as local variables. Changing the joints variables of the linkages causes a change, \(dF_L\), in the forward solution that can be found from the Jacobian of that particular linkage. The number of variables associated with a transform (either global or local) equals the degrees of freedom associated with that transform (or, six minus the number of constraints associated with it). For example, if the transform is used to constrain the position of a frame, there would be three variables corresponding to the three directional components associated with it. In addition, the kinematic chain may also have a number of fixed transforms.

The general procedure for solving a problem starts with classifying the various transforms in the kinematic chain for the system and then assigning variables to the redundancies. Once this is done, the derivatives needed in equations 6 and 7 are found by transforming the motions caused by the variables into a corresponding motion of the non-redundant robot.

Let's return to the system introduced at the beginning of this section. Suppose that robots holding the part may be moved along the floor relative to robot doing the welding. Assume that the welding torch is axisymmetric but is constrained to be vertical in the world frame (see Figure 5).

For this case we have global transforms of \(T_{RB2}\) and \(T_{RB3}\), which give the nominal locations of the robots. Since each of the robots is allowed to be translated along the floor (in the \(x, y\) directions) and rotated on the floor (about the \(z\) axis), the system has six global variables. There are also five local variables for each path point corresponding to the rotation of the part about the torch's axis and the \(x, y, z\) position and \(z\) rotation of the torch in world space (since it is constrained to be vertical). The other labeled transforms are fixed. So for this case we have

\[
\begin{align*}
\frac{dC(\Theta_{ij})}{d\Theta_{ij}} & = \frac{\partial \Theta_{ij}}{\partial t_i} \bigg|_{X_{ij}} = 0 \\
\frac{dT_{RB2}}{dT_{RB2}} & = \text{Trans}(t_1, t_2, 0, 0) \cdot \text{Rot}(Z, t_2) \\
\frac{dT_{RB3}}{dT_{RB3}} & = \text{Trans}(t_1, t_3, 0, 0) \cdot \text{Rot}(Z, t_2) \\
\frac{dT_{Rot}}{dT_{Rot}} & = \text{Rot}(Z, a_{ij3}) \\
\frac{dT_{TLW}}{dT_{TLW}} & = \text{Trans}(a_{ij2}, a_{ij3}, a_{ij4}) \cdot \text{Rot}(Z, a_{ij5})
\end{align*}
\]

Figure 4: Components of Applicable Systems
A few items should be noted. First, each loop contains a six-degree of freedom robot, insuring that the loops can be closed as any of the chosen variables are changed. Secondly, the above assumes that each of the nominal transforms satisfy the constraints. Also, once values for the variables are fixed, the faceplate of each robot is fixed relative to its base and therefore, the joint values of the robots can be found through their reverse solutions. Because of the method of choosing variables, the function $O_{ij}(X_{ij}, t_1, \alpha_{ij})$ in equation 4, is more or less the reverse solutions of the robots. If other linkages were present (for example a table), those joints would have been chosen as variables, so that again, the only unknown joints would be the robots joints which could be found in the same manner as above.

This example leads to $5 \times n_p + 6$ variables to be solved for using the $5 \times n_p + 6$ equations from Equation 6 and the 6 equations of Equation 7.

The last changing transform shown in the figure is $dF_R$, which corresponds to the robot Jacobian. As stated above, the general approach for calculating the terms in Equations 6 and 7 is to find the robot joint changes needed (find $dF_R$), which moves the face plate frame ($F_{FP}$) an amount equivalent to the movement caused by a change in the variables (which keeps the kinematic chain closed).

Consider calculating the $\frac{\partial \Theta_{ij}}{\partial \alpha_{ij}}|_{X_p}$ term for one of the path points. For the example, this is an eighteen (joints) by five (local variables per node) matrix. The first step is to calculate the Jacobian matrices for the robots, which will be denoted $J_{R1}$, $J_{R2}$, and $J_{R3}$. Consider an incremental change in the rotation about the torch. This causes an incremental change in the transform $dT_{\text{ROT}}$, which will be denoted $\Delta^{\text{ROT}} = \text{Rot}(Z, \alpha_{ij})$. In order to keep the chain closed, this gives rise to a $dF_{R2}$ and a $dF_{R3}$. (There is no $dF_{R1}$ because this is a partial derivative and the welding torch frame is held fixed since the other variables are fixed.) This equivalent motion for robot 2 is calculated by

$$dF_{R2} = -T^{-1} \Delta^{\text{ROT}} T$$

(12)

where

$$T = T_{FP}^{-1} T_{P2}^{-1}$$

(13)

In general, the $T$ used in the scheme above is the transform from the transform which represents the changes due to the variable to the robot's face plate. The minus sign is needed since the changes are both counterclockwise in the diagram.

The needed robot joint motion can then be found (if the robot is not in a singular position) from

$$\delta\Theta_{R2} = J_{R2}^T dF_{R2}$$

(14)

Likewise, the necessary motion of robot 3 can be found, along with robot 1's joint changes if robot 1 had to have moved. A similar equation can be written for each of the local variables. There is a slight change for global variables, since a change in these variables cause a change in the robot joint values for all the nodes on all the paths.

### 3.3 Solving the Equations

Equations 6 and 7 are solved numerically using the Broyden-Fletcher-Goldfarb-Shanno algorithm [27,35,36]. This is a first-order update scheme which successively approximates the Hessian matrix (the matrix of second derivatives), which is stored in $LDL^T$ form to save on space. One advantage of this scheme is that it requires only the first derivatives of the function being minimized. Also the convergence is superlinear. Initially it starts as a steepest descent algorithm, which has a linear convergence. But as it iterates, it builds an approximation of the second derivatives and converges to a quadratic convergence rate. The method also possesses very good stability and tolerates inexact line searches well. A disadvantage with this method is that the storage for the matrix may get large for systems and parts with a large number of excess variables.

### 3.4 Cost Functions

The cost function to be used depends on the process. In all operations, it is important to keep inside...
the joint limits of the robot, which should be taken into account within the cost function. One would also prefer low joint speeds, since robots tend to be inaccurate at high joint speeds.

With this in mind, the cost function that has been used most frequently in this work is

\[
C(\Theta_{ij}) = \sum_{i=1}^{n_x} \left( \sum_{j=1}^{n_j} C_1 + \sum_{j=1}^{n_j} C_2 \right)
\]

where

\[
C_1 = \sum_{k=1}^{n_j} \frac{w_k}{d_{ij(k+1)}} (\Theta_{ij(k+1)} - \Theta_{ijk})^2
\]

and

\[
C_2 = \sum_{k=1}^{n_j} \omega_k
\]

using (a) if \( \theta_{ijk} < \theta_{ij}^{\min} + \epsilon \); (b) if \( \theta_{ij}^{\min} + \epsilon < \theta_{ijk} < \theta_{ij}^{\max} - \epsilon \); (c) if \( \theta_{ijk} > \theta_{ij}^{\max} - \epsilon \).

The first part, \( C_1 \), is used to minimize the motion of the robot joints. (It is the discrete version of a path integral of the joint velocities squared.) The terms \( w_k \) and \( d_{ij+1} \) denote weighting functions and the path length between the node point \( j \) and \( j+1 \), respectively. This second term is included to take into account the possibility of uneven spacing between the node points. The weight for each joint's motion has been set to be equal to the inverse of the square of the robot's joint speed limits, which in effect, normalizes the motion relative to the joint's speed capability. In this way, motion of the slower joints can be counted more heavily. The term is, in effect, a sum of the average normalized joint velocities between the nodes.

The second part of the cost function, \( C_2 \), attempts to steer the robot away from its joint limits. In this part, \( \omega_k \) are set proportional to the square of the motion range of each joint. (This normalizes the distance from the limits, \( \theta_{ij}^{\min} \) and \( \theta_{ij}^{\max} \), relative to the size of that joint's envelope.) The term \( \epsilon \) represents the buffer zone set around the joint limits so that the cost function does not go to infinity at the limits. Avoidance of singularities can also be included in this part of the cost function. Since singularities usually occur when one joint has a certain value (as with the second angle being zero in an Euler wrist), this condition can be viewed as a joint limit and substituted for either \( \theta_{ij}^{\min} \) or \( \theta_{ij}^{\max} \), depending on which side of the singularity you are working. If it is not possible to accomplish the motion without crossing the singularity, this barrier can be removed.

Other cost functions are possible (for example using manipulability [20] to steer away from singularities), as long as they depend only on the joint values, not on their velocities. By using a finite difference approach to calculate velocities and accelerations, functions depending on these values may be minimized. (In the applications we are considering, the path speed is dictated by the process and is not great enough to cause dynamic problems. Therefore the cost function above serves our needs.) It is possible to minimize the motor torques, if one models the dynamics of the robot and includes enough node points to make the finite difference values representative of the path segment on which they are calculated. The large number of nodes that may be required would also increase the computational time required for the optimization. Although an offline algorithm, one prefers results in a reasonable time so that they can be evaluated in a reasonable time.

3.5 Singularities

The method presented in this paper fails if any of the path points are at a singularity of the robot. If this occurs, the Jacobian of the robot is not invertible, and thus, \( dF \) is not sufficient to close the kinematic chain. The method does not abort if a singularity occurs on the path between the path points, since the cost (and the kinematic chains) are considered only at the path points. Of course, the results from the optimization may not be very good in this situation since there is an assumption that nothing spectacular occurs between the path points specified. We expect that the jump, which might occur at a singularity along the path, will be reduced because the cost function would try to minimize the change in joint values from the path point before the singularity, to the path point after the singularity.

4 Results

Four scenarios were examined for the system described in the last section. In the first case, the robot was constrained to weld downhand and the robots' positions were given. In case two, the positions of the robots were also part of the optimization. In the third and fourth case, the downhand welding constraint was eliminated, giving \( T_{11, W} \) six degrees of freedom instead of four. The robots' positions were fixed in the case three and free in case four.
The robot used in the simulation was a GMF-700. This robot has a singularity when its fifth joint value equals zero. In all cases to be viewed, the robot was restricted to remain in one configuration (i.e. on one side of the singularity) by setting the joint limits on the fifth axes appropriately. This was possible since an initial path entirely on one side of the singularity could be found for all the paths presented. Because that was possible, the maximum or minimum joint limit (depending on which side of the singularity the path was on) for joint 5 could be set to zero. The task was to weld around a tube which had the cross section represented by 22 nodes (see Figure 6).

![Figure 6: Alcoa Part (Path1)](image)

The initial paths for all the runs had the welding robot stationary with the other robots doing all required motion. This path had a cost of 212. The maximum path speed of the torch allowed before one of the robots reached its joint speed limits (which will be referred to as the critical path speed) was 197 in/min. A summary of the results using the various optimizations can be found in table 1.

<table>
<thead>
<tr>
<th>Case</th>
<th>Gib. Vars</th>
<th>Loc Vars</th>
<th>Final Cost</th>
<th>Critical Speed</th>
<th>Iters</th>
<th>Time on Titan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>110</td>
<td>104</td>
<td>315</td>
<td>574</td>
<td>280</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>110</td>
<td>94</td>
<td>306</td>
<td>650</td>
<td>395</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>154</td>
<td>50</td>
<td>648</td>
<td>712</td>
<td>380</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>154</td>
<td>44</td>
<td>645</td>
<td>929</td>
<td>508</td>
</tr>
</tbody>
</table>

Table 1: Results of Optimization

The cost of the paths with the downhand constraint was approximately 100 and their critical path speeds about 300 in/min. A twofold improvement is made when the downhand constraint is relaxed.

It should be noted that while the cost decreased when the robot locations were included in the optimizations, the critical path speeds increased slightly. This is possible since the sum, not the maximum, of the joint velocities were minimized. What occurred was relatively large improvements in one of the robots, with slight deterioration of another robot.

The joint trajectories for the robots for case 1 and case 4 can be found in Figures 7 and 8.

5 Conclusion

During both the initial layout and the operation of a robotic manufacturing cell, decisions must be made on where to locate the equipment and part and on the coordination of the motion of the various subsystems. These decisions have a great influence on the quality (and in some cases the possibility) of the manufactured part. To help the manufacturing engineer in this complex task, a semi-global positional optimization algorithm has been developed for the optimal path placement problem for kinematically redundant manipulators. The algorithm is capable of simultaneously placing equipment and resolving the redundancies introduced through extra joints or process related constraint relaxation. This is accomplished through the minimization of a cost function of the joint values at the path points (which is assumed to characterize the complete path). The cost function used penalized joint motion (a finite difference velocity) and closeness to the joint limits.

The developed method introduces variables to represent the excess degrees of freedom in both global (placement) transforms and local (extra joints or process related relaxation of constraints) transforms in order to convert the problem into an unconstrained algebraic optimization problem. Derivative information used in the numerical solution of the problem is obtained from relating motions caused by changes in the variables to motions of the robot faceplate necessary for the process constraints to be met at each path point.

The method is applied to a system containing three cooperating robots and its results have been presented. As expected, results improved as the number of optimization variables increased, although not as much for the location variables as for the downhand variables. Results from other systems can be found in [34]. Future work includes automatic generation of initial values for optimization variables, alternate cost functions, and a classification of parts and robot abilities.
Acknowledgements

This work was supported under grants from the Welding Consortium of the Robotics Institute, Babcock & Wilcox and the National Science Foundation's Engineering Design Research Center at Carnegie Mellon University.

References


