A mobile manipulator generally possesses kinematic redundancy, which requires some method for selecting the configuration appropriate to the task. Evolutionary Programming (EP) has been used successfully to select an appropriate configuration using a large penalty on the end-point position to force the end-point to the goal position for the task. This paper demonstrates that using an equality constraint for the end-point position results in much faster convergence to approximately the same final configurations. The reasons for this increase in speed are examined using a simplified model to determine the relative size of their effects.
Configuration Optimization of Mobile Manipulators with Equality Constraints using Evolutionary Programming

Brian Andersen
John McDonnell
Ward Page

Naval Command, Control and Ocean Surveillance Center RDT&E Division

Abstract

A mobile manipulator generally possesses kinematic redundancy, which requires some method for selecting the configuration appropriate to the task. Evolutionary Programming (EP) has been used successfully to select an appropriate configuration using a large penalty on the end-point position to force the end-point to the goal position for the task. This paper demonstrates that using an equality constraint for the end-point position results in much faster convergence to approximately the same final configuration. The reasons for this increase in speed are examined using a simplified model to determine the relative size of their effects.

Introduction

The use of mobile manipulators is becoming more prevalent on the factory floor, undersea, and in space applications. Most of these systems are teleoperated or telepresence systems [1], relying upon the operator to eliminate any kinematic redundancies through direct isomorphic mappings. To make these systems more autonomous, the kinematic redundancies must be resolved without requiring operator intervention.

Various methods have been proposed to search for the best configuration of a mobile manipulator for a given task. These methods include gradient descent searches of the joint-space of the manipulator used by Pin and Culioli [2], and proposed by Burdick and Seraji [3], simulated annealing used by Carriker et al [4], and EP, an evolutionary search strategy, used by McDonnell et al [5]. The EP technique was found to be an effective search technique that avoided local optima.

The cost function for optimization used in [5] imposed a high cost on deviations of the end-point position from the goal position to force the end-point to be at the required position for the task. Unfortunately, this high cost relative to the remaining elements of the cost function results in a search surface that is difficult for the EP algorithm to traverse efficiently. This paper proposes constraining the end-point position to the goal position and searching the remaining smaller-dimensional space. In addition, this paper examines the effect of the ratio of weights for different elements of the cost function on the efficiency of the EP search algorithm.

Optimization Criteria

The ultimate configuration of the manipulator is a direct function of the choice of the optimization criteria. This criteria must take into account the end-point position, obstacles in the environment, static torques required at the joints due to external forces and the weights of the links, and the ability of the manipulator to perform tasks in a particular configuration, known as the manipulability.
This section summarizes the criteria developed in [2] and [5] to determine the cost associated with mobile manipulator configurations.

The end-point position of the manipulator is the most important factor in the cost of a configuration since the manipulator will be unable to perform the task if it is not at the goal position. The cost associated with the end-point position is

\[ C_{\text{goal}} = |X_d - X_e| \]

where
\[ X_d = \text{Desired Cartesian tip position} \]
\[ X_e = \text{Actual Cartesian tip position} \]

It is essential that the manipulator avoid any obstacles that may be present in its workspace. To do this, points of avoidance are specified along the links of the manipulator. The cost associated with obstacle avoidance is

\[ C_{\text{obs}} = \sum_{i=0}^{l} \sum_{j=0}^{m} \frac{1}{|X_{a,i} - X_{o,j}|^2} \]

where
\[ X_{a,i} = \text{Cartesian position of the } i^{th} \text{ point of avoidance} \]
\[ X_{o,j} = \text{Cartesian Position of the } j^{th} \text{ obstacle} \]
\[ l = \text{Number of points of avoidance} \]
\[ m = \text{Number of obstacles} \]

The static joint torques required in a given configuration are a function of the external forces applied and the weights of each link. Any force on the manipulator can be mapped to the static joint torques required for equilibrium using the Jacobian of the manipulator at the point where the force is applied. This mapping is given by

\[ \tau = J^T(\theta) F \]

where
\[ \tau_m = \text{Joint torque vector} \]
\[ J = \text{Jacobian of the manipulator} \]
\[ F = \text{Force applied to manipulator} \]

Therefore, the static joint torques required are given by

\[ \tau = \sum_{i=1}^{N} M_i F_i + \sum_{j=1}^{N} J_j F_j \]

where
\[ M_i = \text{Jacobian associated with the center of mass of the } i^{th} \text{ link} \]
\[ F_i = \text{Force of the weight of the } i^{th} \text{ link} \]
\[ J_j = \text{Jacobian associated with the location of the } j^{th} \text{ point load} \]
\[ \tau_j = j^{th} \text{ point load} \]
\[ N = \text{Number of links} \]
\[ n = \text{Number of point loads} \]

The cost associated with the static torques at the joints is given by

\[ C_{\text{torq}} = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{\tau_i}{\tau_{\text{max, } i}} \right)^2 \]

where
\[ \tau_i = \text{Torque on } i^{th} \text{ joint} \]
\[ \tau_{\text{max, } i} = \text{Maximum torque on } i^{th} \text{ joint} \]

The final element of the cost function is the manipulability, which is designed to avoid configurations near joint-space singularities where the manipulator is difficult to control. The cost associated with manipulability is given by

\[ C_{\text{man}} = \frac{1}{\det (J^T J)} \]

The complete cost function can be formulated as a weighted sum of the elements of the individual cost functions. This is given by
\[ \text{Cost} = \alpha C_{\text{goal}} + \beta C_{\text{obs}} + \gamma C_{\text{man}} + \epsilon C_{\text{corq}} \]

As Pin [2] has pointed out, this cost function can be trapped in local minima when using a gradient descent algorithm. Therefore, the EP algorithm is an attractive alternative.

**The EP Algorithm**

The evolutionary programming algorithm performs a search by methods analogous to those found in natural evolution. The algorithm involves generating an initial population, mutating this population, determining the fitness of each element of the population, competing elements of the population against each other, and selecting the most fit elements of the population to survive to serve as the parent vectors for the next generation.

EP can be described by the following seven-step procedure, similar to the one found in Fogel [6].

1. Generate an initial population of N configuration vectors.
2. Mutate each of the N parent configuration vectors to create a population of size 2N.
3. Determine the cost of each of the 2N configuration vectors.
4. Compete each member of the population against the same number of randomly-chosen opponents, assigning a win to the one with the lowest score.
5. Sort the population by decreasing number of wins.
6. Remove the N configurations with the lowest number of wins.
7. Continue with Step 2.

**The Mobile Manipulator**

The mobile manipulator used in these simulations is shown in Figure 1. It consists of a platform that is free to move in one dimension with a three-link planar manipulator mounted on it. The link lengths are 1.0, 1.0, and 0.5 units, respectively. Each link has a density of 10 units per unit length, and the center of gravity is at the midpoint of each link. The height of the first joint of the manipulator is 0.5 units. An external force of 0.5 units is applied acting downward at the tip of the manipulator.

The size of the mutation for each element of the population is a function of the generation number. This allows the search to cover a large range of possible configurations and still to refine the configuration vectors as the search progresses. The size of the mutations is given by:

\[ \delta = \mu e^{-0.001k} \]

where
- \( \delta \) = Size of the mutation
- \( \mu \) = N(0,1) random variable
- \( \tau \) = U(0,1) random variable
- \( k \) = Generation number

The goal position for the search was located at the Cartesian coordinates (5,1.5), as designated in Figure 1 by the filled rectangle. One obstacle was placed at the Cartesian coordinates (5,1.0), as designated in Figure 1 by the circle. One point of avoidance was chosen at the midpoint of the second link, as designated in Figure 1 by the cross. The weights in the cost function were chosen to be 100 on \( C_{\text{goal}} \), 10 on
Cobs, I on C_{torq} and 1 on C_{man} (\alpha = 100, \beta = 10, \gamma = 1, and e = 1).

Two different approaches were compared. The first approach used a configuration vector consisting of the cart position and the angle of each of the links relative to the horizontal, given by

\[ \mathbf{p}_1 = [x, \theta_1, \theta_2, \theta_3] \]

This representation required four elements in each configuration vector and relied on a large weight on C_{goal} to force the end-point to the goal position.

The second approach used a configuration vector that consists of only the angles of the outer two links of the manipulator, given by

\[ \mathbf{p}_2 = [\theta_2, \theta_3] \]

The cart position and inner joint angle were calculated so that the end-point was at the goal position. Therefore, instead of requiring four elements in the configuration vector, only two were required.

A complication of employing the second approach is the cart position and first angle are not uniquely determined by the outer two angles, but can take on two distinct sets of values. The values used in a given configuration were chosen randomly with equal probability to allow the search to cover as large a portion of the space as possible.

The best configuration obtained after 5000 iterations of the EP algorithm are shown in Figures 1 and 2 for the first and second approaches, respectively. The second approach results in a slightly better configuration than the first, although the configurations are approximately the same.

However, a dramatic difference between the two approaches can be seen in Figures 3 and 4, which show the mean and best costs for the two approaches, respectively. The difference in the horizontal scales of the two figures should be noted. The first approach took 4875 iterations to converge to the point where the value of the cost function of the best configuration was within 0.01 of its final value. The second approach, however, took only 15 iterations to reach that point. Clearly, searching for the optima using an equality constraint instead of putting a large penalty on the end-point position is by far the better approach for this problem.
ment of the cost function while the other
elements have much smaller weights
results in a search space that is difficult
to traverse. The effects of these two
factors are examined in the next two sec-
tions, using a simplified model.

Large Weights in the Cost Function

Placing a large weight on one element of
the cost function that represents a partic-
ularly important constraint on the
system can result in a search space that
is difficult to traverse. In this section, a
simplified model is presented to study
the effect this has on convergence.

The cost function chosen to study this
effect was

$$Cost=A(x+y-10)^2+(x'-y')^2$$

The two terms of this cost function are
minimized along different lines in the
space of x and y. The minima of this
function occurs where x = 5 and y = 5.
There are no local minima in the search
space.

The search space for this cost function is
shown in Figure 7 for the case when
A=100. With the large weight on the
first term in the cost function, the search
surface is a narrow, steep valley, where
the floor of the valley is nearly flat com-
pared to the steepness of the walls.
Once an evolutionary search has found
its way to the floor of the valley, it
would have to mutate the solution in
precisely the correct direction to find a
better solution than the one it has al-
ready found.

To illustrate this, let a parent vector be
x = 6 and y = 4, and let the mutation of

![Figure 3: Best and Mean Cost when Weighting Tip Position](image)
However, if the solution is assumed to satisfy the equality constraint,
\[ 10 = x + y \]

the search surface is reduced to the curve shown in Figure 8. This demonstrates the simplification to the search surface that results from using the equality constraint.

To study the effect of the ratio of the weights, the value of A was varied. For each value of A, ten evolutionary searches were conducted, each starting with a different parent population. Each search was continued until the best solution was within a radius of 0.01 units of the actual minima of the function. Figure 9 shows the average of the number of iterations to achieve this level of convergence as a function of the value of A. This indicates that the number of iterations required for convergence increases as the ratio of the weights (the steepness of the valley walls) increases. However, the number of iterations increases by only approximately a factor of seven as the weight ratio increases by seven orders of magnitude. For this system, the effect of the weight ratio does not increase dramatically until very large weight ratios are employed.

The difficulties of searching in these narrow, steep valleys with relatively flat floors might be overcome by augmenting the evolutionary search with a local search method. This approach has been suggested by Waagen et al [7], using the direction set algorithm as the local method.

**Number of Parameters in Search**

The other factor that increases the effi-
To study the effect of the number of parameters, the simplified model was modified. The cost function was modified to be a function of twelve variables, $x_1$ through $x_6$ and $y_1$ through $y_6$. The cost function was specified by the three equations

\[
x = \sum_{i=1}^{6} x_i
\]
\[
y = \sum_{i=1}^{6} y_i
\]
\[
\text{Cost} = 1000 (x+y-10)^2 + (x-y)^2 + \sum_{i=1}^{6} \left( x_i - \frac{5}{6} \right)^2 + \left( y_i - \frac{5}{6} \right)^2
\]

The minima of this cost function occurs when all the $x_i$ and $y_i$ are $5/6$.

First, the minima was found searching for all twelve variables. The search was repeated ten times, starting from different randomly-selected parent vectors, and the average number of iterations to converge to a solution within 0.01 units of the actual solution was determined. This process was repeated five additional times, each time constraining an additional pair of $x_i$ and $y_i$ to the values where the function was minimized. Figure 10 shows the average number of iterations for convergence as a function of the number of parameters which the search attempted to find. This number of iterations does not increase monotonically with the number of parameters in the search as was expected. While the general trend is upward, it is not clear why the number of iterations drop for increasing number of parameters.

It is interesting to note that the combined effect of the reduction in dimensionality and the simplification of the
the problem of finding the best configuration for a mobile manipulator with redundant degrees of freedom. The search was conducted both by placing a large penalty on the deviations of the end-point position from the goal position and by fixing the end-point at the goal position and only searching for two elements of the configuration vector. The search converged far faster when the end-point of the manipulator was constrained.

This increase in performance is due to two factors: the search surface is difficult to traverse when one factor of the cost function is weighted significantly more than the other; and the dimensionality of the search has been reduced. The results of an analysis of a simplified model indicate that these factors increase the convergence speed by approximately one order of magnitude, but the combined effect does not seem to actually account for the total effect seen in the simulation of the actual manipulator.

Conclusions

Evolutionary programming has been applied to the problem of finding the best configuration for a mobile manipulator with redundant degrees of freedom. The search was conducted both by placing a large penalty on the deviations of the end-point position from the goal position and by fixing the end-point at the goal position and only searching for two elements of the configuration vector. The search converged far faster when the end-point of the manipulator was constrained.

This increase in performance is due to two factors: the search surface is difficult to traverse when one factor of the cost function is weighted significantly more than the other; and the dimensionality of the search has been reduced. The results of an analysis of a simplified model indicate that these factors increase the convergence speed by approximately one order of magnitude, but the combined effect does not seem to actually account for the total effect seen in the simulation of the actual manipulator.

References


2. F.G. Pin, and J-C. Culioli, "Multi-Criteria Position and Configuration
Figure 10: The Number of Iterations for Convergence of the Search for Varying Numbers of Parameters in the Search

Optimization for Redundant Platform / Manipulator Systems", IEEE Int. Workshop on Intelligent Robots and Systems, IROS '90.


