Use of Grid Method for Efficient Global Optimum Design of Robot Kinematics

Joon-Young Park*, Pyung-Hun Chang**, and Jin-Oh Kim***

* Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea (Tel: 82-42-869-3266; Fax: 82-42-869-5226; E-mail: asura@mecha.kaist.ac.kr)
** Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea (Tel: 82-42-869-3266; Fax: 82-42-869-5226; E-mail: phchang@mecha.kaist.ac.kr)
*** Department of Information and Control Engineering, Kwangwoon University, Seoul, Republic of Korea (Tel: 82-02-940-5158; Fax: 82-02-909-3255; E-mail: jokim@daisy.kw.ac.kr)

Abstract

We have presented Grid Method, a systematic and efficient problem formulation method for task-oriented design of robot kinematics and confirmed its efficiency through various design examples. Despite of its efficient problem formulation, however, our previous research has the following limitations: (i) it gives a local optimum only due to its use of a local optimization technique (ii) it uses constant weights for a cost function chosen by a manual weights tuning algorithm, thereby showing low efficiency in finding an optimum solution. To solve these problems, therefore, this paper presents systematic, efficient algorithms of Grid Method with a global optimization technique and an adaptive weights tuning algorithm.

1. Introduction

Task oriented design (TOD) for robots is expected to attract increasing attentions in step with the growth of service robotics. Its design procedure for task oriented robots can be divided into two parts: kinematic and dynamic design [1]. And, a small mistake in the kinematic design could easily make efforts in subsequent stages almost useless. In this paper, therefore, we focus on optimum kinematic design in the context of TOD.

The previous researches on problem formulation of kinematic design [2, 3, 4, 5] has a problem that the number of design variables increases in proportion to the number of degrees of freedom (DOF) and task points, increasing complexities of cost functions and constraints. In addition, these approaches become more inefficient as the number of DOF and task points increases, considering that the search space in general increases exponentially as the number of design variables increases [5]. To solve this problem, we have proposed Grid Method [6], based on a widely used method in Finite Difference Method (FDM). Our approach formulates optimum design problem by using constant number of design variables regardless of the number of DOF and task points. Despite of its efficient problem formulation, however, our previous research has the following limitations: (i) it gives a local optimum only due to its use of a local optimization technique, Generalized Reduced Gradient Method (ii) it uses constant weights for a cost function chosen by a manual weights tuning algorithm, thereby making a selection of the weights difficult and showing low efficiency in finding a optimum solution. To this end, therefore, this paper presents systematic, efficient algorithms of Grid Method with VFSA, a global optimization technique, and an adaptive weights tuning algorithm.

2. Grid Method

In this chapter, we briefly introduce a general concept of Grid Method. Please refer to [6] if you want the details of Grid Method.

The grid method is widely known through its application to FDM. It is Grid Method in the design area of robot kinematics that applies the concept of Grid Method to optimum design of robot kinematics that should meet given kinematic requirements. Let us consider the kinematic design of an n DOF robot under m task points given as kinematic requirements to be satisfied, as shown in Fig. 1a. If we consider every joints and task points as grid nodes, the robots of Fig. 1a can be represented as virtual grid space consisting of the grids as shown in Fig. 1b. The basic concept of Grid Method about these grid nodes is as follows: Just like the heat transfer problem, the base and task points are given as boundary conditions, and then joint positions and orientations within are calculated through
successive grid operations on the unit grids shown in Fig. 1b. The joint positions and orientations in the unit grid including boundaries are determined to satisfy the positions and orientations of base and task points given as boundary conditions using the grid operation. And for the unit grid not including boundaries, the grid operation is performed on the basis of the current joint positions and orientations. These current joint positions and orientations of course are temporary values to be updated. As grid operations continue consecutively for all the unit grids, all the joint positions and orientations are supposed to converge to specific locations.

3. Very Fast Simulated Annealing as Optimization Technique for Grid Method

This chapter introduces the optimization technique used for solving the optimum design problem of robot kinematics by Grid Method.

3.1. Very Fast Simulated Annealing

Simulated Annealing (SA) [7, 8] has been widely used, together with Genetic Algorithm, as a global optimization technique due to its effectiveness in finding a global minimum from local minima and its simple, compact algorithm itself. We choose Very Fast Simulated Annealing (VFSA) as a global optimization technique for our research. The basic algorithm of Very Fast Simulated Annealing can be summarized as follows [7]:

(Step 1: Initialization) Start with a high starting temperature $T$ and a random starting point $x$ where $x = \{x_i; x_i \in [A, B], i = 1...D\}$.

$T \leftarrow T_0, x \leftarrow x_0$  
(1)

(Step 2: Initial Evaluation) Find the function value of the starting point.

$E \leftarrow f(x)$  
(2)

(Step 3: Random Generation) Generate a new point $x'$ using the random variable, $y_i \in [-1,1]$, i.e., $y_i$'s are repeatedly generated until a valid set $x'$ are found $x' \leftarrow x + y_i (B - A), x' \in [A, B]$.

The VFSA generating distribution is defined as

$g_x(y) = \prod_{i=1}^{D} \frac{1}{2(||y_i||+T) \ln(1+1/T)}$  
(4)

To generate new points according to this distribution, new values of $y_i$ are generated from a $u_i$ from the uniform distribution $u_i \in U[0,1]$, by

$y_i = \text{sgn} \left( u_i - \frac{1}{2} \right) \left( 1 + \frac{1}{T} \right)^{-y_i} - 1$  
(5)

(Step 4: Function Evaluation) Calculate the function value of $x'$.

$E' \leftarrow f(x')$  
(6)

(Step 5: Metropolis Criterion) Accept or reject new point $x'$ using Metropolis criterion.

If $\Delta E < 0$ where $\Delta E = E' - E$, accept new point. If $\Delta E > 0$, accept new point with probability density function defined as
\[ p(x') = \frac{1}{1 + \exp(\Delta E / T)} \]  

(Step 6: Temperature Annealing) Reduce the temperature \( T \) by annealing schedule.

\[ T' \leftarrow T(0) \exp(-c_k^{1/\alpha}) \]  

(Step 7: Convergence Criterion) Return \( x \) and \( E \) as the optimal point and the optimum function value. If convergence criterion is not satisfied, go to step 3. Otherwise, stop executing the algorithm.

3.2. Recursive Evolution Technique

Jung and Cho proposed Recursive Evolution Technique to accelerate the convergence characteristics of the Metropolis algorithm [9]. In this paper, we briefly introduce its basic concept only as follows: If \( f^{mn} \) is not improved during given trial number, the recursive evolution of solutions is actuated from \( x^{mn} \).

\[ E \leftarrow f^{mn}, \quad x \leftarrow x^{mn} \]  

3.3. Overall Algorithm of Optimum Design using VFSA and Grid Method

In this paper, we propose a whole design algorithm for global optimization of robot kinematics such as Fig. 2. This algorithm is to combine the techniques introduced in this section with problem formulation using Grid Method.

4. Adaptive Weights Tuning Algorithm

We newly propose the normalization of the cost function and an adaptive weights tuning algorithm. To begin with, we state the normalization of the cost function.

4.1. Normalization of Cost Function

The constraint functions for the cost function can be classified into the following: functions of length (distance) variables, functions of angle variables, functions of both variables. Here, there happens big difference between the variation ranges of the variables due to non-homogeneity of physical units of the variables. Such imbalance makes it difficult for each term of the cost function to equally participate in the optimization of the cost function. To solve these problems, we normalized all the variables. Here, we state the normalization procedure of LC (Limit Constraint) only as such an example. LC before normalization is as follows:

\[ \text{Figure 2. Flow chart for VFSA with Grid Method} \]
Developing the case of \( \phi < \phi_{mn} \) in the same way, we can get the normalized function of Eq. (10) as follows:

\[
w'_c \cdot f'_c = \begin{cases} 
    w'_c \cdot (\phi' - 1)^2 & \text{if } \phi' > 1 \\
    w'_c \cdot (\phi') & \text{if } \phi' < 0 \\
    0 & \text{otherwise}
\end{cases}
\]  

(13)

In the same manner, all the constraint functions except DM (Dexterity Measure) can be normalized. As a normalized DM for our research, we chose a transform of the Layout Conditioning \( \kappa (T) \) [10], which has zero as its optimum and varies between 0 and 1.

### 4.2. Development of Adaptive Weights Tuning Algorithm

Despite the normalization, however, the actual variation ranges of the normalized functions of constraint functions can be different each other. For each term to equally participate in the optimization of the cost function, therefore, an adaptive weights tuning algorithm is necessary to adaptively keep their balance according to current condition of each term every an iteration. We propose an adaptive weights tuning algorithm as follows:

(1) Initially, all the weights are set to 1.

(2) The grid operations are consecutively executed on all the unit grids by using the determined weights.

(3) We should first know current magnitude of each constraint function to determine the weights for the balance between the terms.

(4) DM is a measure that should be minimized as far as possible, while other constraint functions are sure to be very close to 0. In like manner of manual weights tuning algorithm, therefore, after determining the weights of other terms except DM, we should determine the weight of DM to balance with other terms. The detail methods of weighting DM will be explained later.

As mentioned before, the weight of DM is determined in a different way compared with other constraint functions. In this step, therefore, we obtain a maximum \( f'_{cm} \) among constraint functions except \( f'_{cm} \). This value is used in \( f'_{cm} / f'_{cm} < \alpha_{mn} \), a standard of step 5 to select a priority group, a set of the constraint functions that have priority to participate in the optimization over other ones.

(5) To exclude the constraint functions that have relatively small values owing to enough optimization from a priority group, we find a priority group by using \( f'_{cm} / f'_{cm} < \alpha_{mn} \), obtaining a minimum \( f'_{cm} \) among the priority group.

(6) The priority group from the steps above, that is, the constraint functions that have its value between \( f'_{cm} \) and \( f'_{cm} \) should take precedence in the minimization of total cost function. Our basic weighting method for this purpose is as follows:

\[
w'_{cm} = \frac{f'_{cm}}{f'_{cm}}
\]  

(14)

In addition to this algorithm, we introduce the constraint priority between each constraint functions into the basic weighting method as follows:

\[
w'_{cm} = \left( \frac{f'_{cm}}{f'_{cm}} \right)^{\alpha_{mn}}
\]  

(15)

Table 1 shows the priority number, index \( N_{cm} \), and its corresponding constraint functions from our experience. However, the classification on priority of each constraint function depends upon the characteristics of the given design problem. Therefore, a user should determine it according to the characteristics of each design problem.

(7) Finally, we state weighting algorithm of DM. As mentioned before, DM cannot easily become close to zero of the optimum unlike other constraint functions, and mostly has a relatively large value compared with other ones. The weighting method of DM in this paper is in consideration of such characteristics is to determine the weight of DM in order to balance the magnitude of DM term with those of other terms chosen as a priority group for optimization every an iteration. To this end, we determined the weight of DM to be the average level of all the terms except DM through the following weighting:

\[
w'_{cm} = K_{mn} \sum_{i=1}^{m} f'_{cm} \frac{1}{m}
\]

\[
: w'_{cm} = K_{mn} \frac{\sum_{i=1}^{m} f'_{cm}}{m}
\]  

(16)

(8) If all the weights are determined through the procedures above, return to the step 2 and repeatedly execute the procedures above.
<table>
<thead>
<tr>
<th>Priority No</th>
<th>$N_{ei}$</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>LC</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>EC (Equalization Constraint) for twist angle, DOC (Desired Orientation)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>EC for link length and offset, DC (Dimension Constraint), JAC (Joint Angle Change)</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>OA (Obstacle Avoidance)</td>
</tr>
</tbody>
</table>

Figure 3. Initial postures and design results

5. Application Example

A 6 DOF spatial manipulator having a spherical wrist has been designed for seven task points. The twist angles of the robot are given as $\alpha = [-90^\circ, 0^\circ, -90^\circ, 90^\circ, -90^\circ, 0^\circ]$. The robot is to avoid two sphere-type obstacles, which have radius of 3 and safety factor of 2. Kinematic requirements for this design are as follows: EC, DOC, LC of $0 \leq l_i \leq 30, -30 \leq d_i \leq 30$ ($i = 1,2,\ldots, 6$), DC, OA, DM, and JAC of $\Delta \theta_{ei} = 60^\circ$ ($i = 1,2,\ldots, 6$).

5.1. Verification of the Efficiency of Adaptive Weights Tuning Algorithm

Applying Grid Method on initial postures of Fig. 3a, Fig. 3b and 3c are the design results obtained by the
manual weights tuning algorithm and the adaptive weights tuning algorithm, respectively. In the case of the manual weights tuning algorithm, it took 10 trials to choose the weight of DM through best tuning and 2763 iterations to get an optimum with the chosen weight; In the case of the adaptive weights tuning algorithm, 4 trials and 1531 iterations. Here, note that the adaptive weights tuning algorithm requires fewer trials in choosing the weight of DM. The results above show that the adaptive weights tuning algorithm is more efficient than the manual weights tuning algorithm.

5.2. Design Results for Global Optimum Kinematics

Fig. 3d shows the design results using the whole design algorithm by combining Grid Method with VFSA and the adaptive weights tuning algorithm. As seen from the results, initial link lengths and offsets are not identical, because a user arbitrarily gives initial joint positions. Through grid method, however, we can get same link lengths and offsets for every task points, that is, identical links for every task points. Careful observation in Fig. 3 leads you to find the dotted and solid lines representing the desired orientation and the orientation of the end-effector, respectively. From this figure, we can see that in the design results each end-effector exactly indicates the desired orientation, while each end-effector in initial postures does not. Also, we achieve obstacle avoidance from initial postures that collide with two obstacles. In addition, these results are optimized from the viewpoint of DM, as seen from \( f'_{DM_{\text{final}}} = 6.6662 \Rightarrow 5.0500 \), which is more optimized than the result obtained using GRG, \( f'_{DM_{\text{final}}} = 5.2968 \).

6. Conclusion

To solve the problems of Grid Method in our previous research, we proposed the whole design procedure by combining Very Fast Simulated Annealing with the problem formulation using Grid Method. This result showed that a global optimization technique could be applicable to the formulated problem using Grid Method as well as a local optimization technique. And we proposed the adaptive weights tuning algorithm that automatically determined the weights in consideration of dimensional non-homogeneity between the metrics and deviation of each metric from the optimum, and confirmed its effectiveness.

The results above present a simple, systematic, effective design approach for robot optimum kinematics: a problem formulation using Grid Method, a global optimization technique and an adaptive weights tuning algorithm. Hence, if we input initial postures for every task points to the computerized program of these algorithms, it automatically produces a global optimum of robot kinematics. By virtue of this, even a novice designer can easily design robot kinematics, and the development period of robot is expected to shorten.

Acknowledgement

The authors would like to express their gratitude for the KOSEF (Korea Science and Engineering Foundation), for supporting this research under their HWRS-ERC project.

References