INVERSE KINEMATICS OF ARBITRARY ROBOTIC MANIPULATORS USING GENETIC ALGORITHMS

A.A. KHWAJA, M.O. RAHMAN AND M.G. WAGNER
Arizona State University,
Department of Computer Science and Engineering,
Tempe, AZ 85287, USA,
email: `khwaja@asu.edu`, `obaid@asu.edu`,
`wagner@asu.edu`

Abstract: In this paper we present a unified approach for solving the inverse kinematics problem of an arbitrary robotic manipulator based on Genetic Algorithms. The fitness function we use in our algorithm does a multi-objective optimization satisfying both the position and orientation of the end-effector. As an example we show how this approach can be used for computing the motion of an n-R robotic manipulator following a specified end-effector path. To avoid unnecessary manipulator undulations in case of a redundant design, we thereby introduce a third objective to our fitness function minimizing the discrete joint velocities. Unlike Jacobian-based solutions our approach deals efficiently with redundant designs and singularities.

1. Introduction

There exists only a small number of attempts to use Genetic Algorithms (GAs) in the context of kinematics or Computer Animation. A reason being that GAs are themselves new with lots of accompanying problems which need to be resolved along with the real objectives. Most of the work, reports results in terms of planar manipulators and none of them made any mention of how to deal with singularities. Furthermore very few referenced the handling of redundant robots and all the problems that go along with it.

Davidor (1991) uses GAs to generate and optimize robot trajectories of a redundant robot in two dimensions. The robot mentioned in the paper has three links but the author claims that the technique can easily be extended for n-link structures. Ahuactzin et al. (1993) use genetic algorithms for planning the motion of a robot arm among moving obstacles. While the proposed method does not make any assumptions about the number of links and the authors only present a 2R robot for ease of graphical representation. Gritz and Hahn (1995) made use of genetic programming (an extension of genetic algorithms to programs) for 3D articulated figure motion.

In this paper we exhibit the potential of genetic algorithms by applying them to arbitrary 3D manipulators. Our experiments were conducted with linkages with a total number of links ranging from 4 to 15. The approach proved to handle singularities or redundancies in a very effective way. It is easily extensible to other problem areas such as inverse dynamics.

2. Genetic Algorithms

Genetic Algorithms are search algorithms for finding optimal or near optimal solutions. They can be considered a cross between gradient-based calculus methods and Artificial Intelligence (AI) algorithmic search methods.

Conventional AI search methods proceed by building a search tree along their way. The traversal is usually done by a fixed traversal scheme. These methods, in general, do not perform a directed search. Solutions that are obtained during the traversal are judged and discarded and new solutions are searched for until the optimal solution is found.

Gradient-based calculus methods on the other hand start with a initial solution and traverse the parameter space proceeding in the direction that reduces the error, hence obtaining increasingly better solutions. These methods, while being very directed, are highly localized and solution quality commonly depends on the initial solution. Because of this, they are highly susceptible to getting stuck in a local minimum in problems having a multimodal error surface.

GAs use a directed search like gradient-based calculus methods but do not rely on derivatives. Thus they do not require the parameter space to be continuous. Along with that, they are global search methods. While gradient-based methods pick one initial solution, GAs pick a whole population of randomly generated initial solutions so that the whole space is searched in parallel. This prevents GAs to get stuck in a local minimum with a certain probability depending on the size of the population. There is, however, the problem of so-called premature convergence which is closely reminiscent of the local minima problem. In a premature convergence situation the GA loses diversity in its population of solutions resulting in no improvement for a couple of generations until random mutation reintroduces some of the missing elements. While this impedes the performance of GA and the quality of solution significantly, it is not difficult to handle and various methods are proposed by researchers to get around it effectively (see e.g. Goldberg (1989)).

2.1. MAIN CONCEPTS

The key concept in Genetic Algorithms is that problems are solved at a level different than that at which they are created. There is always some kind of coding scheme involved with which the parameters of one particular solution are encoded into strings called *chromosomes.*

Genetic Algorithms usually work with fixed sized populations of chromosomes. An initial population is generated entirely randomly. The quality of a chromosome is measured by the fitness function which evaluates the closeness of the corresponding solution to the desired solution with a proper distance measure. Then three genetic operators are applied to produce the next generation of solutions (chromosomes). These operators are *selection/reproduction*, *crossover* and *mutation.*

Selection or reproduction is used to build the next generation using the fitness value of the chromosomes as their probability of survival. The more fit an individual is the more chance it has to be a part of the next generation. The crossover operator picks two parent chromosomes randomly from the new generation, chooses a random crossover point and swaps the chromosome strings after the crossover point. This way the operator produces two new child chromosomes. Then, a mutation operator is applied with a certain probability which changes one or more elements of the chromosomes randomly. Finally the fitness values of the next generation's chromosomes are re-evaluated. This process is repeated for a fixed number of generation after which the chromosome with the best fitness value is taken to be the solution. For an excellent introduction to GAs, we refer the reader to Goldberg (1989) and Mitchell (1996).

3. Genetic Approach to Inverse Kinematics

Since GAs are search algorithms, their application to inverse kinematics causes a radical change in the underlying model. As opposed to the velocity model approach, the GA-based approach does not require the computation of the Jacobian and thus works for every possible mechanism in the same way as long as one is able to solve the forward kinematics problem.

3.1. BUILDING CHROMOSOMES

As an example, we use an n-R robot such as shown in figure 1. There are no requirements on the design of the manipulator. The parameters we are going to encode are the joint angles θ_i of the n revolute joints. Without loss of generality we may assume that these angles hold

$$-\theta_{max} \leq \theta_i < \theta_{max}, \quad i = 0, \ldots, n-1, \tag{1}$$

with $\theta_{max} = \pi$. We use a binary coding of these state variables since GAs are known to work best with binary coded problems (see Goldberg (1989)). In order to achieve this we convert θ_i into an unsigned integer T_i according to

$$T_i = \mathrm{int}\left((2^k - 1)\,\frac{\theta_i + \theta_{max}}{2\,\theta_{max}}\right), \quad i = 0, \ldots, n-1, \tag{2}$$

where k is the number of chromosome bits per angle. Concatenating all binary strings together, the total chromosome size for an n-R robot is kn bits. For the decoding phase, this process is reversed.

3.2. FITNESS FUNCTION

In order to obtain the fitness of each chromosome one first has to solve the forward kinematics for each corresponding set of joint angles. The fitness value is then set to be equal to the distance of the resulting end-effector position to the desired end-effector position, measured with a proper distance measure. Tests showed that best results are achieved if the problem is decoupled into the translational and orientational part thus giving two independent fitness values. Note that this approach cannot be coordinate independent.

However, the problem with this approach is that one is now facing a multi-objective optimization where each individual objective is a constraint that the solution has to satisfy. This has been a source of complication in both gradient-based optimization and genetic algorithms and is referred by some researchers as the *curse of dimensionality.*

Conventionally, fitness functions in GAs return a single numeric quantity which when divided by the sum of all fitness values serves as the probability of selection or *survival* of that particular chromosome. This works without problems when there is only one criterion to optimize. But when there are more than one objectives to meet, combining them to obtain one unique quantity cannot be achieved in a straightforward manner. One possible way is to use a linear combination

$$f = \sum_{i=1}^{d} c_i f_i \tag{3}$$

of the subfitness values f_i from d different optimization criteria. The constant coefficients c_i serve as weights for this approach and can be used to adjust the influence of each objective to the total fitness. As it turns out, this method fails to provide an optimal solution to most problems because of the failure in providing a unique direction for the system to evolve in.

Gritz and Hahn (1995) used the above formulation in their genetic programming approach with the exception of making the weights adaptable. They start out with all weights zero except for the main objective and as generations proceed, increase the other weights slowly. Although they reported satisfactory results, the method, however, still doesn't guarantee a near-optimal solution.

Biological organisms usually also have many objectives and constraints to satisfy in order to survive and be a part of the next generation. Organisms that have higher fitness than others are more likely to escape attacks by predators thus having a higher survival probability. We therefore decided to take recourse in nature's approach.

3.3. PREDATOR FUNCTION

The two main objectives that our system has to meet are the position and orientation accuracy of the end-effector. Let f_t be the subfitness value associated with the position and f_r the subfitness value associated with the orientation. For instance, we choose f_t to be the Euclidean distance of the tool center point to its desired position, and f_r to be equal to the rotation angle necessary to achieve the desired orientation. We first convert these subfitness values into two individual survival probabilities p_t and p_r by

$$p_t = \frac{f_{tmax} - f_t}{f_{tmax}}, \quad p_r = \frac{f_{rmax} - f_r}{f_{rmax}}, \tag{4}$$

where f_{tmax} is the maximum distance and f_{rmax} the maximum angle possible. Without loss of generality we may always choose $f_{rmax} = \pi$ whereas f_{tmax} depends on the geometry of the robot. If for any reason f_t is larger than f_{tmax} we set p_t equal to zero. In order to improve results one might additionally use rescaling functions to rescale p_t and p_r such that they are properly distributed. These probabilities are calculated for each individual chromosome.

We then introduce a *predator function* which will keep the population size at a certain level by trying to terminate individuals. We therefore randomly choose an objective, a chromosome in the current population, and a *predator fitness* ranging between 0 and 1. If the predator fitness is higher than the chosen survival probability of the selected chromosome, the chromosome is deleted. The procedure works as a low level component of the tournament selection method described below. It is repeated until the desired population size is achieved.

3.4. SELECTION, CROSSOVER AND MUTATION

For selection, a method known as *tournament selection* is used (see Crossley (1994)) with a tournament size of 2. It works as follows: Two chromosomes are randomly selected without replacement from the current population and the best of the two is taken to be the first mate. This process is repeated and a second mate is obtained. These two parent chromosomes are then mated together to produce two new chromosomes for the next generation.

Tournament selection helps keeping the destructive effects of crossover minimal. High order and longer schemas are more likely to be destroyed by the clutches of the crossover operator. This effect, however, will be minimal if the two chromosomes, that are selected as mates, are close to each other in fitness value. While tournament selection does not guarantee as such a close selection, the probability of selection of two such chromosomes is more than that in a simple selection and this probability increases as the tournament size is increased. But increasing the tournament size too much shows up as a considerable loss in efficiency. The crossover and mutation operators we used are the standard ones as described in section 2.1.

3.5. FOLLOWING THE END-EFFECTOR PATH

The inverse kinematics problem is intrinsically non-unique in terms of its solutions. GAs search the whole solution space in parallel and for that reason can come up with any of the multiple satisfying solutions. One can, however, introduce additional objectives to the fitness function, which will guide the evolutionary process to a solution with special desired features. One of the cases where this is useful is when the manipulator is to follow a specified end-effector path. This can be implemented as follows.

Let us assume that we have already calculated the joint angles for the end-effector position at time instant t. Following the end-effector path we want to calculate the joint angles at $t' = t + \Delta t$ for a small Δt. Our goal is to keep the change in the joint angles small. In order to achieve this we decided to encode the changes $\Delta\theta_i = \theta'_i - \theta_i$ of the joint angles instead of the joint angles themselves into the chromosomes. Note that $\Delta\theta_i$ can be considered the *discrete joint velocity* of the ith joint. Furthermore, to force the GA to search for solutions only in a certain neighborhood of the previous solution, we set θ_{max} in (1), now describing the maximum discrete joint velocity, equal to $\theta_{max} = \frac{\pi}{6}$. Finally, we introduce a third objective to our fitness function by minimizing the sum

$$\sum_{i=0}^{n} \Delta\theta_i^2 \tag{5}$$

of the squares of the discrete joint velocities. Selection, crossover and mutation are implemented as described above.
Since the solution of a genetic algorithm is only a near-optimal solution, this technique is likely to produce small but still unwanted jumps in the discrete joint velocities as seen in figure 2. It is therefore useful to postprocess the joint angle data with an appropriate data smoothing technique, for instance by applying a discrete Weiner filter.

4. Examples and Conclusion

Figure 1 shows an example of an 8R robotic manipulator following a specified end-effector path while figure 2 is an example of how the algorithm is able to escape singularities. In both examples the motion of the end-effector is interpolated linearly between initial and final positions.

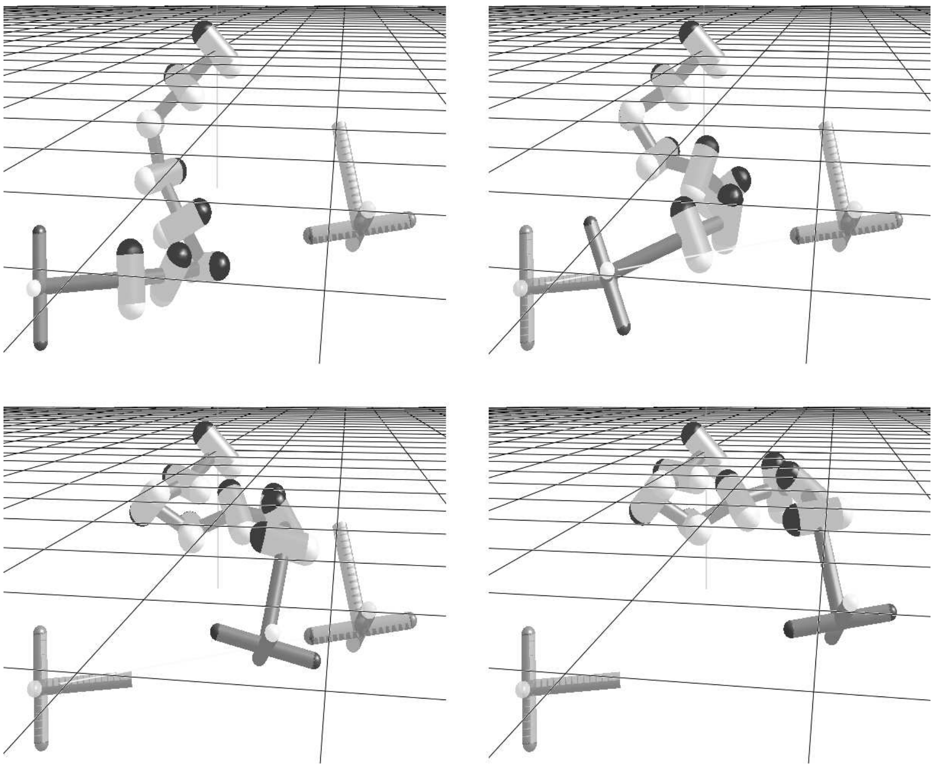

Figure 1. 8R manipulator following specified end-effector path.

We used a one point crossover operator with a crossover probability of 0.8. Mutation operator has not been considered very useful by GA researchers because of its purely random nature. We used the usual proposed mutation probability of 0.001. The number of generations and the population size were selected empirically to be 50 and 150, respectively.

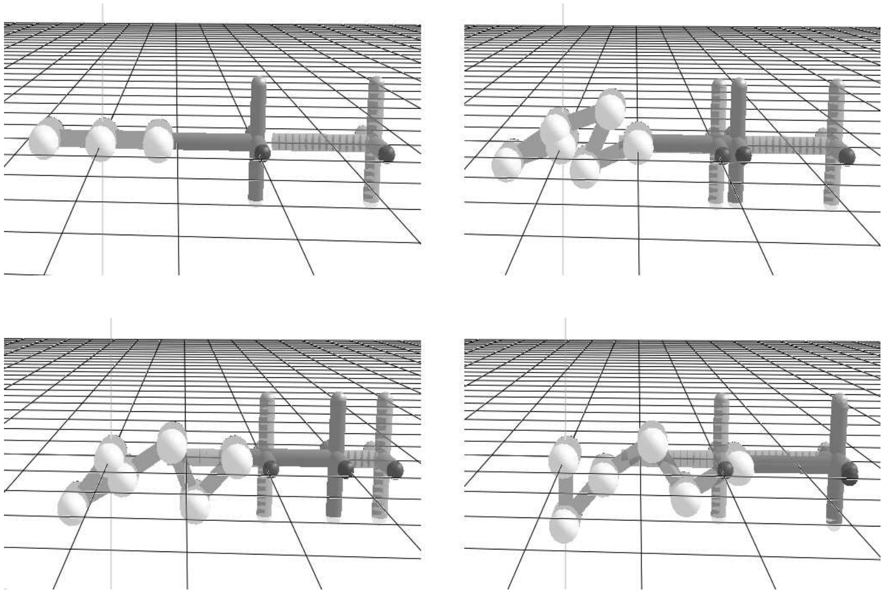

Figure 2. 6R manipulator escaping a singular position.

Although Genetic Algorithms provide a straightforward approach to inverse kinematics, there are still a lot of open questions which have to be answered. The above implementation did not address the problem of premature convergence, which leads to loss in efficiency and accuracy. In particular, we were not able to achieve real-time computation. In order to overcome these problems we are currently working on a hybridization of genetic algorithms which will include entropy optimization techniques.

References

Ahuactzin, J.M., Talbi, E.G., Bessiere, P., and Mazer, E., (1993), Using Genetic Algorithms for Robot Motion Planning, Geometric Reasoning for Perception and Action Workshop '93, pp. 84–93.

Crossley, W.A., Wells, V.L., and Laananen D.H., (1994), The Potential of Genetic Algorithms for Conceptual Design of Rotor Systems, Semiannual Report, Arizona State University.

Davidor, Y., A Genetic Algorithm Applied To Robot Trajectory Generation, in: Handbook of Genetic Algorithms, ed: Davis, L., Van Nostrand Reinhold 1991, pp. 144-165.

Goldberg, D.E., (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley.

Gritz, L., and Hahn, J.K., (1995), Genetic Programming for Articulated Figure Motion, Journal of Visualization and Computer Animation.

Mitchell, M., (1996), An Introduction to Genetic Algorithms, MIT Press.