Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

A Closed Form for Inverse Kinematics Approximation of General 6R Manipulators using Genetic Programming

F. Chapelle, P. Bidaud
Laboratoire de Robotique de Paris
Université Paris 6
10-12 avenue de l'Europe
78140 Vélizy-Villacoublay, FRANCE
Email: chapelle,philippe@robot.uvsq.fr

Abstract

This paper presents an original use of Evolutionary Algorithms in order to approximate by a closed form the inverse kinematic model (IKM) of analytical, non-analytical and general (i.e. with an arbitrary geometry) manipulators. The objective is to provide a fast and general solution to the inverse kinematic problem when it is extensively evaluated as in design processes of manipulators. A mathematical function is evolved through Genetic Programming according to the known direct kinematic model to determine an analytical expression which approximates the joint variable solution for a given end-effector configuration. As an illustration of this evolutionary symbolic regression process, the inverse kinematic models of the PUMA and the GMF Arc Mate are approximated before to apply the algorithm to general 6R manipulators.

1 Introduction

Basically, the inverse kinematic problem (IKP) consists in finding the set of joint variables to achieve a desired configuration of the tool frame. This problem usually involves a set of nonlinear, coupled, algebraic equations and there are no general algorithms which may be employed to solve it. Depending on the joint axis geometry of the system, closed form or numerical solutions can be determined. Closed form solutions exist for special manipulator geometries, for instance decoupled manipulators and more generally when the degree of the characteristic polynomial is less or equal to 4 [8]. Only numerical solutions relying on iterative procedures can be used for non-special architectures. Literature on robotics proposes number of numerical methods for solving the IKP starting from modified Newton-Raphson algorithms, to more recent ones such as those exploiting polynomial continuation [11], dyalitic elimination [10], genetic algorithms [9] or neural network [12]. Since numerical methods are generally time consuming, it can be interesting to have an approximation of the IKM under an analytical form. This is particularly true when the IKM is involved in a computer-aided-design process where it has to be solved for many different manipulator geometries and extensively evaluated [7].

In this context, the search for techniques to produce approximations of complex models can have an application to the inverse kinematics. This paper investigates the possibility to find the closed form solution of the IKP by the mean of evolutionary symbolic regression. We start with a description of the algorithm principles which rely on Genetic Programming. Examples of 6R analytical and general manipulators are used to demonstrate the proposed technique validity.

2 Evolutionary Symbolic Regression Algorithm

2.1 Principles

The algorithm we developed seeks programs which approximate the inverse of a given function by the use of evolutionary symbolic regression. These programs will be the closed forms of the IKM. The algorithm relies on Genetic Programming (GP) techniques which are global, semi-stochastic, optimization mechanisms with intrinsic parallelism. GP is particularly well adapted to symbolic regression since it works on programs with variable sizes and shapes.

The direct model is supposed to be known under the form of a mathematical function F. It can also

0-7803-6475-9/01/$10.00© 2001 IEEE

be any process getting as input the unknows, which can be considered as the design parameters (Y), and returns evaluation values (X) as output. In the case of the IKM, (Y) is the joint configuration ($Y \equiv \theta$) and (X) the configuration of the end-effector. (X) can also contain a set of already determined joint parameters (Y_i) and the geometric parameters of the manipulator.

The aim of the algorithm is to find a program-function (PF) which approximates the inverse of F:

$$F(Y) = (X) \quad (1)$$

$$F^{-1}(X) = (Y) \quad (2)$$

The term "program-function" comes from the fact that the algorithm works on programs, which represent here analytical expressions. The structure of the algorithm is defined in figure 1.

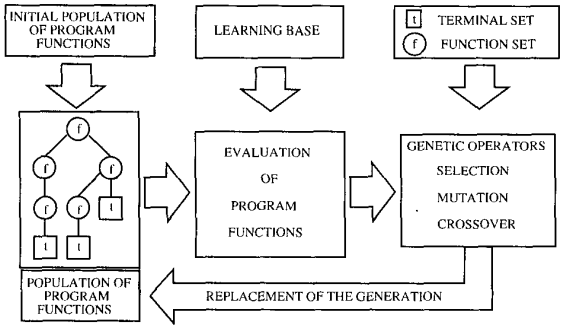

Figure 1: Evolutionary Symbolic Regression

The learning base $\mathcal{L}$ is composed of a set of characteristic points. Each point is made up of a set of design parameters (Y) and associated yielding values (which are the values of (X) that F returns). The symbolic regression is carried out by applying an evaluation function using the points of $\mathcal{L}$. The algorithm is run for each joint parameter Y_i.

The underlying principles are developed in the following sections.

2.2 Genetic Programming

GP belongs to the family of Evolutionary Algorithms (EA). They are semi-stochastic algorithms which simulate the evolution of a population. GP has been proposed in the early 90s by J.R. Koza [5] and has been since improved in its performances [6], [1]. An initial population of PF has to be generated. Then, genetic operators are applied to each generation (selection, crossover and mutation). Mutation can be removed in GP [5]. The best individual obtained versus the number of generations is chosen to be the solution returned by the algorithm. GP uses a particular kind of individual encoding which is computer program: solutions (individuals) are encoded as a tree-structure similar to LISP code. Nodes can be functions or terminals. It is important to notice that functions can require several arguments and that terminals can be numbers or variables. Then, two sets of available functions and terminals have to be defined precisely for each run and they are strongly dependent on the problem the GP has to solve. We use the most recent evolutions in the GP: Automatically Defined Function (ADF) [6] (part of PF which behaves within one individual like an encapsulated function), demetic grouping [4] or steady state [13].

2.3 Evaluation

The algorithm seeks a model that fits a given sample of data. Each run of the algorithm approximates one joint parameter θ_i. The evaluation of each individual (PF) is done by applying a function which assigns a value called fitness, wich has to be minimized:

$$\mathcal{E}_i = \sqrt{\sum_{j=1}^{nb-points} \left(\theta_i^j - \theta_i^{jl}\right)^2} + lp \quad (3)$$

where:

- nb-points is the number of characteristic points of the learning base $\mathcal{L}$ ($\equiv$ Card($\mathcal{L}$)),
- θ_i^j is the joint value θ_i computed by the PF for the characteristic point j,
- θ_i^{jl} is the joint value of this point in the learning base,
- lp is a penalty for the PF length (the number of nodes). This makes possible to control the average length (or size) of the individuals in the population: those which have a high size tend to be eliminated by the selection operator. Generally lp = length(PF)/x where x is the ideal length desired by the programmer.

The algorithm, is applied to approximate the inverse kinematic model for manipulators with six revolute joints.

3 Application and Results

A uniform discretisation of whole or part of the variables space is used to build a learning base $\mathcal{L}$. The

direct kinematics gives the corresponding configurations of the end-effector.

The number of characteristic points is arbitrary. A trade-off has been made after several tests such as this number is sufficiently large so that the algorithm can work well, and sufficiently weak so that a larger number would be useless and expensive in computing time.

For the nodes of the tree-encoded individuals, functions defined with particular conditions can be used in the function set (for instance the division by zero). If the conditions are not satisfied, they could lead to situations where the process is blocked. Thus, protected functions have to be defined, in such a way that the algorithm tends to eliminate them without being blocked: division, square root, arcsin(x) and arccos(x).

In each of the following cases, the algorithm has been run several times with different parameters (population size, number of generations, mean of creation of the initial population, selection type, crossover probability, demetic grouping, mutation, steady state, number of ADF, length penalty, add best to the new population?). Characteristic results for the best parameter set are given hereafter.

Notice that for a 2R manipulator which have simple analytical solutions, results converge towards the exact analytical expression for one joint, and approximated the second with an average error of about 10^{-6} radian on each point of the learning base [2].

3.1 Industrial Manipulators

A test base $\mathcal{T}$ of about several hundreds of characteristic points, different from those of $\mathcal{L}$, is built from the direct kinematics to check the obtained solutions for different configurations in the variables space covered by the learning base. For one joint parameter θ_i, we determine the average error $(\mathcal{ERR})_i$ on each characteristic point j of $\mathcal{T}$ by using the best PF.

$$(\mathcal{ERR})_i = \frac{\sum_{j=1}^{Card(\mathcal{T})} |\theta_i^j - \theta_i^{jt}|}{Card(\mathcal{T})} \qquad (4)$$

where:

- θ_i^j is obtained from the PF, taking its arguments from (X) which contains the configuration of the end-effector and possibly the other joint parameters (see 2.1),
- θ_i^{jt} is the joint value in the test base.

3.1.1 *PUMA 560*

The algorithm approximates the expressions of the first three joint values with an average error of about

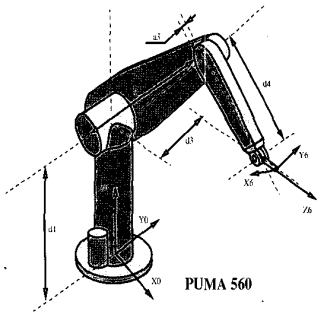

Figure 2: PUMA 560

10^{-4} radian on each characteristic point of the learning and the test bases. The figures 3 and 4 show examples of the evolution curves. Notice that the convergence is fast: approximately 10 generations.

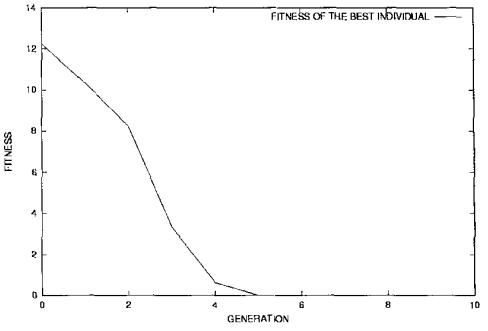

Figure 3: Best individual fitness for θ_1 versus the number of generations

The results allow to follow pre-defined trajectories with a precision not greater than 1 mm (simulated results). For the other joint values, errors between 10^{-1} and 10^{-2} radian are found. The less good results are explained by the complexity of the joint parameters solved. The length of the individuals cannot be restricted more than slightly. For example, after 20 generations a PF for θ_1 is:

```
atan2((pow(pow((y-160.0),2),2)*((pow(
pow((y-160.0),2),2)*sqrt(((y+x)-sqrt(
160.0))))*sqrt((160.0+y))-atan2((160.0+
y),y)))*sqrt((160.0+y)),((sqrt(sqrt(((x+y)
-(sqrt(sqrt(((y+x)-sqrt(160.0))))-atan2(
(160.0+y),y)))*sqrt((x+160.0)*(y-160.0))))
-sqrt(sqrt(160.0)))+(atan2(((x+y)-atan2(x,
y)),atan2((160.0+y),y))*sqrt((x+160.0)*
(y*y)))*(25600.0-y)))
```

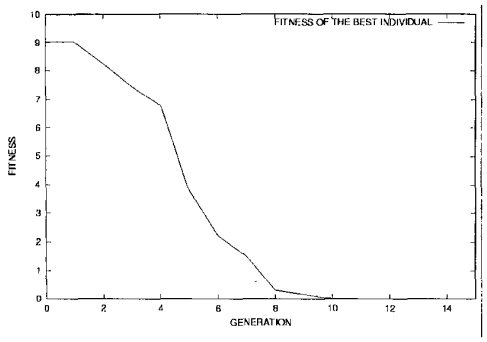

Figure 4: Best individual fitness for θ_2 versus the number of generations

The exact analytical expression is: $\theta_1 = Atan2(X,Y) - Atan2(d_3, \pm\sqrt{X^2+Y^2-d_3^2})$. The size of the obtained approximation remains reasonable. The function and terminal sets are described in the section 3.3. The most direct consequence of the non size restriction is to slow down the computation. It takes 50 generations and 30 minutes to a SiliconGraphics O_2 computer to determine one joint parameter.

3.1.2 *GMF Arc Mate*

There is a singularity missing between the GMF and the PUMA. The last three joint axis are not intersecting. Thus, there is no analytical solution to its inverse kinematics.

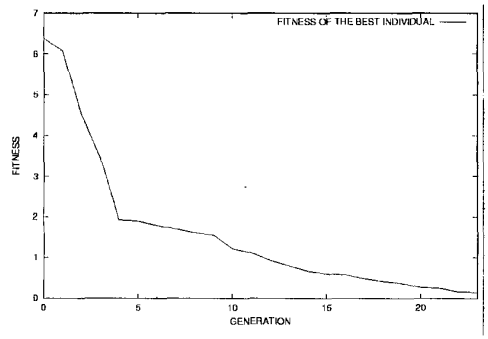

Figure 5: Best individual fitness for θ_1 versus the number of generations

The algorithm finds approximations of the joint values with an average error from 10^{-2} for θ_1 (figure 5) to 10^{-1} radian for θ_6. All the conclusions given for the PUMA are relevant to the GMF. The algorithm still gives good results for non analytical manipulators.

3.2 General Manipulators

"General" means that the topology is arbitrary. If the kinematic description uses the Denavit-Hartenberg (D.H.) notation [3], the aim is to find here an expression for each joint parameter, depending of the configuration of the end-effector, the D.H. structural parameters ($a_1 \ldots a_6$, $\alpha_1 \ldots \alpha_6$, $d_1 \ldots d_6$) and the joint variables previously determined.

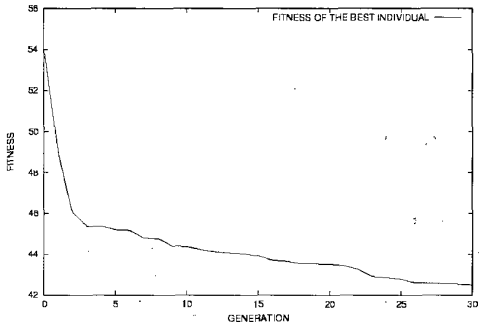

Figure 6: Best individual fitness for θ_1 versus the number of generations

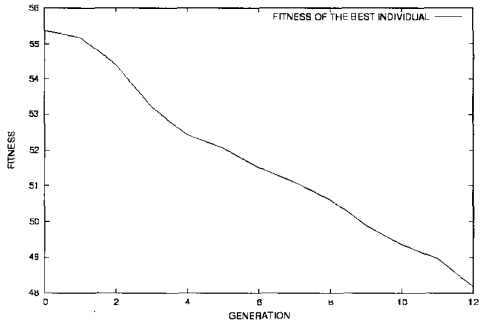

Figure 7: Best individual fitness for θ_6 versus the number of generations

The learning base $\mathcal{L}$ is then composed of 24 columns which are the variable terms of the problem (D.H. parameters and the configuration of the end-effector). $\mathcal{L}$ has been built by taking as input the D.H. parameters (the direct kinematics gives the coordinates of the cartesian configuration of the end-effector). If, for instance, three different values are chosen for each input, there are 3^{24} lines in $\mathcal{L}$. Considering this large size, we have to randomly eliminate the major part of them (more than 99%).

The results remain good: of the order of 10^{-1} radian for each joint parameter (see examples in figures 6 and 7). We can observe on these figures that the fitness does not converge towards 0. The reason is that it has to be divided by the square root of the total number of characteristic points (lines) in $\mathcal{L}$. An example of the obtained expression for θ_1 is:

```
sqrt(tan(pow(sqrt(pow(cos(sqrt(sqrt(sqrt((a2+
Y))))),2.0)),2.0)*(cos(sqrt(atan2(a2,Y)))*
(cos((sin((sin(cos(al2))+cos(al2)))+atan2(
cos(sqrt(sqrt(Y))),sqrt(atan2(a2,Y))))*
(sqrt(atan2(a2,Y))*cos(sin(cos(al2))))*cos(
sin(tan((sqrt(sqrt((a2+Y)))+pow(cos(sqrt(
atan2(a2,Y))),2.0))*(cos(cos(al2))*cos(sin(
cos(al2)))))))))
```

and for θ_6:

```
atan2(divide(atan2(divide(divide(cos(cos(
acos(a5))),cos(sin((cos(GAE)*tan(atan2(o4,
al4))-(GAE+2.0))))),cos((sin(pow(sin((atan2(
o5,a3)-(GAE+2.0))),2.0))+divide(tan(al1),tan(
2.0))))),atan2(sqrt(pow(a5,2.0)),divide(pow(
2.0,2.0),sin(cos(GAE))))),cos(sin((sin((al5-
a3))*sqrt(pow((sin(pow(sin(divide(asin(1.0),
sin(atan2(o5,a3)))),2.0))+pow(sin(atan2(o4,
al4)),2.0)),2.0))-(GAE+2.0))))),atan2(sqrt(
pow(a5,2.0)),divide(divide(pow(atan2(1.0,1.0)
,2.0),sin(cos(GAE))),cos(o3))))
```

Here, o1 ... o5 are the previously determined joint parameters. (ALE, BEE, GAE) are the Euler angles which define the orientation of the end-effector ($\equiv \alpha_{eff}$, β_{eff}, γ_{eff}); (X, Y, Z) its position. al_i is the D.H. parameter α_i.

The solutions returned by the algorithm are checked by applying them to the PUMA 560: A group of points is initially built by the kinematics model ([3]). An instance corresponding to the PUMA is then given to the D.H. parameters. The error between the joint values given by the obtained solutions and those of the set of points previously built is determined. The points are obviously chosen different from those of the Learning Base $\mathcal{L}$.

The results show an average error on each point comparable to the one which was obtained on $\mathcal{L}$ during the use of the algorithm. We can also notice that the elimination of a great number of points from the Learning Base does not introduce a fatal instability. The absolute average distance around this error yet increased. Its maximum value is 0.3 radian.

These results are optimal when the dimensions of the manipulator are included in the intervals between the values chosen for the D.H. parameters to generate the Learning Base.

3.3 Discussion

a) Determining Algorithm Parameters

Some parameters have a determining influence on the algorithm:

- Population Size P: the larger it is, the better the convergence is. There is also a minimum, which is around 10000, below which the algorithm does not find any good solutions. Populations are usually composed of thousands of individuals,
- Tournament Size: with these sizes of population, only the tournament selection makes the algorithm possible to converge towards good individuals. The size of the tournament is also determining (without seeming to follow precise rules). Beyond P=10000, a tournament of 200 individuals is needed,
- Length Penalty: in the determination of the PF fitness, a penalty for the length of the individuals can be included. It is problem dependant. If this penalty is too high, the algorithm will be too much limited in its search and there will be a premature convergence. If it is too weak, the expression of the individuals will tend to be too long, which can cause problems due to the limitations of the computer memory.

b) Learning Base

Since it is well-known that the IKP can have multiple solutions, it can be useful, for finding better results, to restrict the characteristic points in the learning base corresponding to only one of them. This can be obtained either by elimination of the points which tend to increase the fitness; and/or with an initial knowledge of the right points field given by the observation of the robotic system.

In general, the more the workspace covered by the learning base is restricted, the better the results are. This is the same if the discretisation intervals are small (but the computing time increases). In each case, the best trade-off has to be found.

For the PUMA and the GMF, learning bases of several hundreds of points were used; for general manipulators, several tens of thousands.

c) Function and Terminal Set

The choice of the elements of the function and terminal sets is very important. If there are non determining elements, in particular in the terminal set, the algorithm can be disturbed. If determining elements are missing, the algorithm will be limited in its search and will not find good solutions. As an illustration of

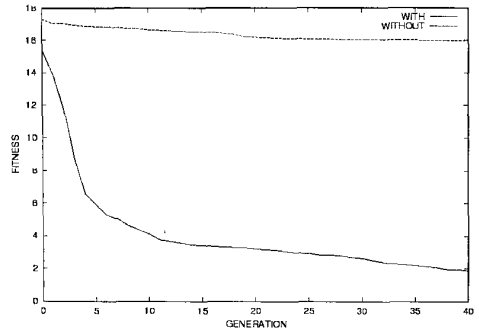

Figure 8: GMF Arc Mate - Best individual fitness for θ_3 with and without the coordinates of the desired orientation for the end-effector

this, for the search of the expression of θ_3 for the GMF Arc Mate, the convergence is faster and reaches lower fitness values if the coordinates of the desired orientation of the end-effector (α_{eff}, β_{eff}, γ_{eff}) are included in the terminals. We show this sensitivity in figure 8. The best results are found when only the necessary functions and terminals are used.

The elements of the function set are chosen among:

$$sqrt, square, sin, cos, tan, acos, asin, atan2, +, -, *, /,$$

Here acos, asin, /, sqrt (square root) are the protected function forms.

As explained in section 2.1, the terminal set is composed of the end-effector cartesian configuration, and the characteristic geometric dimensions of the manipulator (or D.H. parameters for general manipulators). It can also contain already determined joint parameters.

4 Conclusion

We have implemented an evolutionary symbolic regression algorithm in order to give models approximating the IKM of any general 6R manipulator by program-functions with variable forms and sizes.

From the author's knowledge, these approximating models are given here for the first time. They allow to produce a reliable and fast IKP solution for any manipulator geometry.

The main advantage of this method is that the online computation of the obtained models in different complex processes is of the order of the micro-second, for any configuration of the end-effector, and for analytical but also non analytical manipulators.

References

[1] W. Banzhaf, P. Nordin, R.E. Keller, and F.D. Francone. *Genetic Programming, An Introduction.* Morgan Kaufmann Publishers, Inc., 1998.

[2] F. Chapelle, O. Chocron, and P. Bidaud. Genetic programming for inverse kinematics approximation. In *Proc. I.S. on Robotics, Montreal*, 2000.

[3] John J. Craig. *Introduction to Robotics, Mechanics and Control.* Addison-Wesley, 1986.

[4] Adam P. Fraser. *Genetic Programming in C++.* Technical Report 040, University of Salford, Cybernetics Research Institute, 1994.

[5] John R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection.* MIT Press, 1992.

[6] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, 1994.

[7] C. Mavroidis, M. Alam, and E. Lee. Analytic geometric design of spatial r-r robot manipulators. In *Proceedings of the 2000 ASME Mechanisms and Robotics Conference*, 2000.

[8] C. Mavroidis, F.B. Ouezdou, and P. Bidaud. Inverse kinematics of six degree of freedom general and special manipulators using symbolic computation. *Robotica*, 1993.

[9] J.K. Parker, A.R. Khoogar, and D.E. Goldberg. Inverse kinematics of redundant robots using genetic algorithms. In *Proc. IEEE of ICRA*, 1989.

[10] M. Raghavan and B. Roth. Solving polynomial systems for the kinematics analysis and synthesis of mechanisms and robot manipulator. *Journal of Mechanical Design*, 117:71–79, 1995.

[11] L.W. Tsai and A. Morgan. Solving the kinematics of the most general six-and-five-degree-of-freedom manipulators by continuation methods. *Transactions of ASME, Journal of Mechanisms, Transmition and Automation in Design*, 107:189–200, 1985.

[12] Zhang Wei. The inverse kinematics for the orientation of a robot arm based on neural network. *Journal of Nanjing University of Aeronautics & Astronautics*, 29(1):46–50, 1997.

[13] Thomas Weinbrenner. *The Genetic Programming Kernel, Version 0.5.2.* University of Darmstadt, 1997.