A Forward Kinematics Data Structure for Efficient Evolutionary Inverse Kinematics

Sebastian Starke(✉), Norman Hendrich, and Jianwei Zhang

Department of Informatics, Group TAMS
(Technical Aspects of Multimodal Systems),
University of Hamburg, 22527 Hamburg, Germany
{starke,hendrich,zhang}@informatik.uni-hamburg.de

Abstract. Various approaches to solving inverse kinematics implicitly rely on computing forward kinematics in order to obtain an approximate solution. This work proposes an optimised data structure to efficiently compute these equations by avoiding redundant transformations and calculations. This is particulary relevant for highly articulated kinematic models and multiple end effectors with shared joints along their kinematic chains. By integrating the developed OFKT (Optimised Forward Kinematics Tree), less computation time within each iteration is required, which contributes to a significant speedup in convergence. Experiments were conducted using a novel evolutionary approach which was designed for handling complex kinematic geometries.

Keywords: Forward kinematics · Inverse kinematics · Data structures · Computational efficiency · Evolutionary optimisation · Robotics · Character animation

1 Introduction

A rigid body kinematic system can be described by a set of kinematic chains, each consisting of a consecutive set of segments and joints from the root to the end effectors. Each end effector results in a certain Cartesian configuration $\mathscr{X}$ given a specific joint variable configuration θ. Together, the motion axes of the joints define the DOF (Degree of Freedom) and thus the computational complexity of the whole kinematic system. [1]

While forward kinematics (FK) is straightforward to compute by a consecutive set of coordinate transformations, obtaining solutions for inverse kinematics (IK) in contrast is not as easy. For any given IK query, a varying or even infinite number of solutions can exist, and it is not generally clear which one to prefer. However, IK takes an important role in various applications such as robotics, including object manipulation and grasping with anthropomorphic hands, as well as for character animation in computer graphics. Since analytical approaches to this problem are not generally available as they must be derived individually for specific kinematic structures, numerical algorithms for obtaining approximate solutions have become more popular. In order to optimise an appropriate solution for IK, such methods rely on calculating the FK equations using the known

S. Zeghloul et al. (eds.), *Computational Kinematics*, Mechanisms and Machine Science 50,
DOI 10.1007/978-3-319-60867-9_64

kinematic structure. Then, sampling-based joint variable updates are generated using gradient-based or probabilistic techniques, and the Cartesian end effector configurations are calculated individually for each objective. Multiple end effector systems—such as the finger tips of an arm—usually contain many shared joints along their kinematic chains, and the FK equations then become partially equivalent. As a result, most computation time is typically required for repeated FK computation, and many transformations become redundant when only small joint variable changes are applied. This is especially the case for evolutionary approaches, for which the genetic operators—such as recombination and mutation—cause most joint variable configurations to be only slightly modified within each generation. Given the joint variables which correspond to the genes of an individual, the resulting end effector configurations $\mathscr{X}_{1,\ldots k}$ can be obtained by evaluating the FK function. Based on this, it is then possible to define the multi-objective fitness function Ω to be minimised as the root-mean-square error over all individually weighted objectives $\mathscr{L}_{1,\ldots k}$ with end effector targets $\mathscr{Y}_{1,\ldots k}$.

$$\phi = \Omega(x) = \sqrt{\frac{1}{k}\sum_{i=1}^{k} w_i \mathscr{L}_i^2(\mathscr{X}_i, \mathscr{Y}_i)} \tag{1}$$

Repeated evaluation of the fitness ϕ of each individual within the population then drives the evolutionary optimisation. Hence, efficient computation of the FK is essential for the overall performance and convergence of the IK algorithm. Figure 1 demonstrates solving articulated IK of the Kyle humanoid, with the OFKT (Optimised Forward Kinematics Tree) data structure integrated to achieve higher computational efficiency for repeated and only slightly varying joint variable queries.

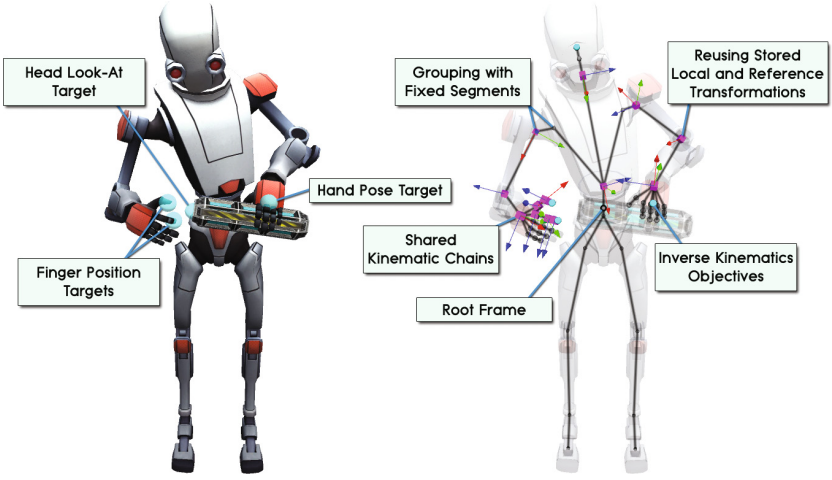

Fig. 1. Kinematic geometry from the pelvis to the head and finger tips of the Kyle humanoid (28 DOF). Inverse kinematics is solved by evolutionary computation while efficiently calculating forward kinematics using the OFKT data structure.

2 Background and Related Work

Solving IK is a fundamental problem which is relevant in very different fields of research, such as robotics, computer graphics or human-computer interaction. Typical scenarios include control of virtual characters and runtime manipulation of underlying animations, grasping with anthropomorphic hands, teleoperation tasks as well as motion planning and trajectory generation. Therefore, numerous sophisticated approaches have been developed that tackle the problem by the specific requirements of their applications, typically regarding computational efficiency, accuracy or flexibility. The numerical methods can be categorised into four groups: gradient-based, probabilistic, geometric or learning. In this work, we will primarily focus on the former two as they require generating FK samples for IK optimisation.

Considering the FK calculation for a given a kinematic structure, homogeneous coordinate transformations in robotics are commonly represented using Denavit-Hartenberg parameters, which can achieve a considerable reduction in the required amount of calculations [4,5]. Implementing rotations by quaternions rather than matrices is slightly more efficient from a computational perspective, and also offers a unique representation for the resulting orientations. Furthermore, information about axis alignments in serial or parallel mechanisms can be incorporated [2,3]. However, those computational optimisations are only appropriate for specific geometries and must be derived individually. It is also possible to learn the FK equations by neural networks which can be used to create a functional relation between joint and Cartesian space [6]. Nevertheless, this method introduces an additional inaccuracy for the numerical IK optimisation due to the inherent learning error.

Gradient-based Jacobian or SQP methods optimise an IK solution by slightly modifying each joint variable to obtain the gradient [7–9]. These methods are often applied in robotics as they can achieve a fast convergence, but can suffer from multiple local extrema in the search space. In this regard, genetic algorithms (GA) perform a more robust search space exploration by means of biologically inspired probabilistic optimisation, and offer better scalability for higher DOF [10,11]. The key idea is to encode joint variable configurations as individuals, and to iteratively evolve new solutions until convergence. The fitness is obtained by the resulting end effector errors using the FK equations. However, traditional methods require many parameters to be tuned, or the required computation time or attainable accuracy might remain insufficient. In our prior work, a novel evolutionary hybridisation of GA and swarm intelligence achieved promising results both in accuracy and computation time, with adaptive parameter control for varying dimensionality and kinematic geometries [12,13]. As each fitness evaluation for every individual requires a FK pass in order to obtain the resulting errors in position and orientation, it was observed that most of the computation time was due to the required coordinate transformations. More specifically, many of those were redundant as with increasing accuracy, fewer genetic mutations were applied, and partially shared kinematic chains for multiple end effectors were computed repeatedly.

3 Algorithmic Approach

The main purpose of our OFKT data structure is to avoid calculating repeating or redundant consecutive transformations. Given a single joint variable configuration $\theta = (\theta_1, ..., \theta_n)$ as input, the individual Cartesian transformations for end effectors $\mathscr{X}_{1,...,k}$ can be obtained as denoted by (2). Hence, calculating FK becomes processing a tree of single kinematic chains with either individual or partially shared joints.

$$f(\theta) = \mathscr{X}_{1,...,k} \tag{2}$$

For each kinematic chain (3), the end effector configuration is computed by consecutive transformations starting from its root.

$${}^{root}T_{ee} = {}^{root}T_1 \; {}^{1}T_2 \; ... \; {}^{n-1}T_{ee} \tag{3}$$

According to (4), these transformations between the single relevant segments can be grouped into *reference* and *local* transformations, R_i and L_i respectively.

$$R_i = R_{i-1} \; L_i \qquad\qquad L_i = S_i \; T(\theta_j) \tag{4}$$

However, not every segment of the kinematic chains is necessarily connected to a joint. Thus, the *static* transformation S_i denotes the constant transformation between the segment's preceding non-static segment to the segment's local transformation with $\theta_j = 0$. Note that S_i only needs to be computed once, and is then stored to avoid recalculating non-changing transformations. The OFKT itself is then represented by a linked list of segments, one for each moveable part of the kinematic structure. Within each node, R_i and L_i are individually computed and stored, together with the currently assigned corresponding joint variable θ_j. While the former depends on the preceding reference and the current local transformation, the latter is calculated using the segment's static transformation S_i modified by θ_j. Algorithm 1 summarises building the OFKT data structure which can then be used for efficiently processing multiple successive FK queries.

Algorithm 1. Building the Optimised Forward Kinematics Tree

```
   Input  : Geometry, Root, End Effectors
 1 OFKT = CreateLinkedList(Root, End Effectors);
 2 Chains = GetChains(Root, End Effectors);
 3 foreach Segment in Chains do
 4 |   if Segment.HasJoint() then
 5 |   |   Node = OFKT.Insert(Segment);
 6 |   |   Node.ComputeAndStoreStaticTransformation();
 7 |   |   Node.StoreJointVariable();
 8 |   |   Node.ComputeAndStoreLocalTransformation();
 9 |   |   Node.ComputeAndStoreReferenceTransformation();
10 |   end
11 end
```

When performing a FK query, a joint variable configuration is given as input to the OFKT. The goal is to perform kinematic queries efficiently by using the stored variables for the current transformation and the joint value within each node. As described by Algorithm 2, the update procedure is started at the root node of the linked list, and is recursively called for all childs. Also, a boolean parameter is recursively passed which initially assumes that no update would be required. The flag is set in case of joint variable changes, as otherwise the stored local transformation can be reused. As soon as one local update occured during the FK calculation, a relative update is also necessary for all subsequent nodes. Note that transformation updates in local and reference space are treated independently by propagating the boolean flag. After the tree traversal, the resulting end effector transformations can be returned in world space using (5), where the additional ${}^{world}T_{root}$ transformation is prepended to handle movement in world space. Thus, the OFKT keeps all transformations in reference to the kinematic model while representing only non-static connections.

$$W_i = {}^{world}T_{root}\ R_i \tag{5}$$

Algorithm 2. Querying the Optimised Forward Kinematics Tree

```
   Input  : Joint Variable Configuration
   Output: End Effector Transformations
 1 Function UpdateFK(Node, RequireUpdate):
 2     if HasJointVariableChanged() then
 3         Node.StoreJointVariable();
 4         Node.ComputeAndStoreLocalTransformation();
 5         RequireUpdate = true
 6     end
 7     if RequireUpdate then
 8         Node.ComputeAndStoreReferenceTransformation();
 9     end
10     foreach Child of Node do
11         UpdateFK(Child, RequireUpdate);
12     end
13     return ;
14 UpdateFK(OFKT.Root, false);
15 foreach End Effector Node do
16     return Node.ComputeWorldTransformation();
17 end
```

Intuitively, the OFKT data structure is optimised to efficiently process multiple FK queries by caching transformations from preceding calculations, assuming that only a few joint values will change between successive queries. Considering the genetic evolutionary IK algorithm which was our original motivation, only some genes (joint angles) of an individual are usually modified during one generation, and many reference as well as local transformations can be reused, resulting in a large performance increase.

4 Evaluation and Results

First, a theoretical evaluation for the OFKT data structure is done regarding the total required transformations in four scenarios, summarised in Table 1. Given a n-dimensional serial joint variable configuration, one FK pass requires calculating n local transformations which are then concatenated by n transformations, and 1 further from world to root — this will be used as baseline for performance comparison.

1. *FK computation by updating a (random) number of values along a serial kinematic chain:* This is the typical query after evolving the genes of an individual. In general, $2n + 1$ calculations would be needed for independent FK queries. Using the OFKT, previous results can be reused efficiently, and the required amount of local transformation updates becomes equivalent to the number of changed joint values $j < n$. Traversing the single segments then results in $n - i$ instead of n reference transformations, where i is the first modified index along the serial kinematic chain.
2. *Predicting the end effector world transformation by modifying exactly one joint value:* This is helpful for determining or estimating the error gradient. Only 1 local $L_i^{'}$ as well as 3 further transformations $R_{i-1}\ L_i^{'}\ R_i^{-1}\ R_{ee}$ are necessary for calculating the end effector transformation, followed by 1 additional world transformation. In particular, the required transformations of the single segments are already available, and enable to directly obtain the relative end effector change.
3. *Iteratively updating exactly one joint value while maintaining information about all segment transformations:* This is particularly important for enabling efficient further computation of relative transformations within the kinematic tree. n queries are performed iteratively, requiring $n(2n + 1)$ calculations. Using the OFKT, each of the n queries automatically avoids recalculating unchanged transformations, resulting in n local updated segments and a total of $\frac{n(n+1)}{2}$ calculations for the affected reference transformations.
4. *FK calculations on different chains with multiple end effectors of an anthropomorphic arm:* This is relevant in terms of scalability for complex geometries. A 27 DOF anthropomorphic geometry is considered, starting with a 7 DOF arm and splitting up into a hand with five 4 DOF fingers — giving rise to $k = 5$ chains with 11 DOF each. Hence, calculating all end effectors individually would require $k(2n+1)$ transformations, while the OFKT automatically avoids recalculating the shared $s = 7$ joints along the arm.

Table 1. Comparison of the required amount of transformations in different scenarios.

Scenario	Standard	OFKT
Random modifications	$2n + 1$	$n - i + j + 1$
End effector computation	$2n + 1$	5
Single iterative modifications	$2n^2 + n$	$\frac{n^2}{2} + \frac{5n}{2}$
Multiple end effectors	$k(2n + 1)$	$k(2n + 1) - 2(k - 1)s$

In order to put the previous FK evaluations into practical context, experiments were conducted in performing IK on an articulated 10 DOF kinematic model using the presented algorithm in [12,13]. It was observed that $\approx 10^3$ generations in average were required for solution convergence. In this regard, Fig. 2 demonstrates the computational improvement at each generation during one IK query (left) and in average for increasing DOF (right) when using the OFKT. In particular, note that less computation time per generation is required since fewer genetic changes are applied as the population scores progress over several generations. It can be observed that this computational improvement scales significantly for more complex geometries, reaching a cost reduction per generation by a factor of ≈ 8 for 30 DOF.

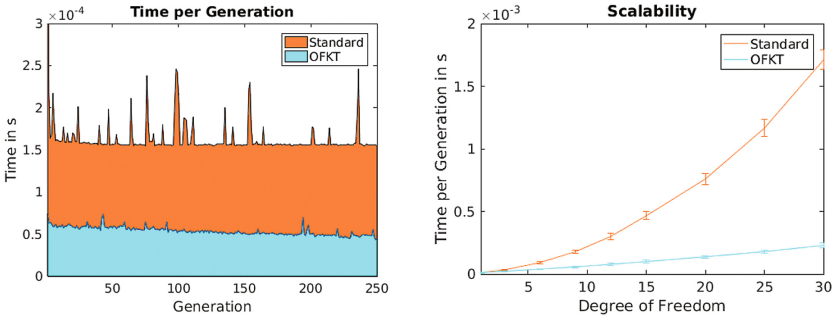

Fig. 2. Computational cost per generation for one IK query (left) and for increasing DOF (right).

Ultimately, we investigated the speedup in solving evolutionary IK on different robot geometries of 6 to 10 DOF. The results are shown in Fig. 3 and are averaged over 10.000 randomly generated samples using full pose objectives. In particular,

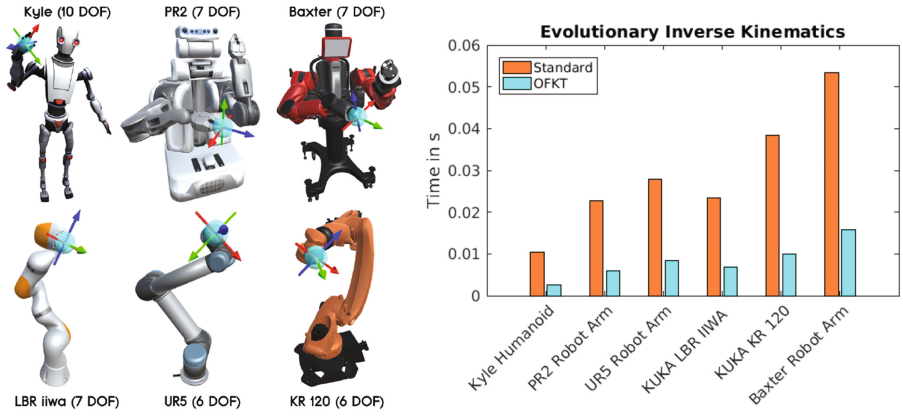

Fig. 3. Computational cost in solving random reachable IK queries on different robots.

a considerable computational improvement can be observed by requiring approximately one third of the original time for convergence on all tested kinematic models.

5 Conclusions

This work proposed a method to efficiently compute the FK equations for multiple consecutive queries with only slightly varying joint variable configurations. The developed OFKT data structure caches results that were computed in preceding calls, and only updates the transformations along the kinematically affected segments within the whole kinematic tree. Hence, the required amount of costly calculations for sampling-based IK optimisation on complex and multiple end effector geometries can be reduced. Integrating the OFKT into our evolutionary IK algorithm, one third of the original computation time for solving full poses on typical lower DOF industrial robots was required, with significantly larger improvements obtained for increasing DOF. This work will be further investigated applied to dexterous manipulation, humanoid robots and character animation.

Acknowledgements. This research was funded by the German Research Foundation (DFG) and the National Science Foundation of China (NSFC) in project Crossmodal Learning, TRR-169.

References

1. Kathib, O., Siciliano, B.: Handbook of Robotics. Springer, Heidelberg (2008)
2. Wu, P., Wu, C., Yu, L.: An method for forward kinematics of stewart parallel manipulators. In: Proceedings of the IEEE International Conference on Intelligent Robotics and Applications, 171–178 (2008)
3. Song, S., Kwon, D.: Efficient formulation approach for the forward kinematics of 3-6 parallel mechanisms. Adv. Robot. **16**(2), 191–215 (2002)
4. Denavit, J., Hartenberg, R.: A kinematic notation for lower-pair mechanisms based on matrices. Trans. ASME J. Appl. Mech. **23**, 215–221 (1955)
5. Denavit, J., Hartenberg, R.: Kinematic Synthesis of Linkages. McGraw-Hill Series in Mechanical Engineering, p. 435. McGraw-Hill, New York (1965)
6. Sadjadian, H., Taghirad, H.D., Fatehi, A.: Neural networks approaches for computing the forward kinematics of a redundant parallel manipulator. Int. J. Comput. Electr. Autom. Control Inf. Eng. **2**(5), 1664–1671 (2008)
7. Beeson, P., Ames, B.: TRAC-IK: an open-source library for improved solving of generic inverse kinematics. In: Proceedings of the IEEE RAS Humanoids Conference (2015)
8. Kim, I., Oh, J.: Inverse kinematic control of humanoids under joint constraints. Int. J. Adv. Robot. Syst. **10** (2012)
9. Lee, D., An, S.: Prioritized inverse kinematics with multiple task definitions. In: IEEE International Conference on Robotics and Automation (2015)
10. Tabandeh, S., Clark, C.M., Melek, W.W.: An adaptive niching genetic algorithm approach for generating multiple solutions of serial manipulator inverse kinematics with applications to modular robots. Robotica **28**, 493–507 (2010)

11. González Uzcátegui, C.E.: A Memetic Approach to the Inverse Kinematics Problem for Robotic Applications. Doctoral Thesis, Carlos III University of Madrid (2014)
12. Starke, S.: A Hybrid Genetic Swarm Algorithm for Interactive Inverse Kinematics. Master Thesis, University of Hamburg (2016)
13. Starke, S., Hendrich, N., Magg, S., Zhang, J.: An efficient hybridization of genetic algorithms and particle swarm optimization for inverse kinematics. In: Proceedings of the IEEE International Conference on Robotics and Biomimetics (2016)