Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Journal Track

IKBT: Solving Symbolic Inverse Kinematics with Behavior Tree (Extended Abstract) *

Dianmu Zhang[1,2] and **Blake Hannaford**[2]
[1]Applied Sciences Group (ASG), Microsoft
[2]University of Washington
{dianmuz, blake}@uw.edu

Abstract

Inverse kinematics solves the problem of how to control robot arm joints to achieve desired end effector positions, which is critical to any robot arm design and implementations of control algorithms. It is a common misunderstanding that closed-form inverse kinematics analysis is solved. Popular software and algorithms, such as gradient descent or any multi-variant equations solving algorithm, claims solving inverse kinematics but only on the numerical level. While the numerical inverse kinematics solutions are relatively straightforward to obtain, these methods often fail, even when the inverse kinematics solutions exist. Therefore, closed-form inverse kinematics analysis is superior, but there is no generalized automated algorithm. Up till now, the high-level logical reasoning involved in solving closed-form inverse kinematics made it hard to automate, so it's handled by human experts. We developed IKBT, a knowledge-based intelligent system that can mimic human experts' behaviors in solving closed-from inverse kinematics using Behavior Tree. Knowledge and rules used by engineers when solving closed-from inverse kinematics are encoded as actions in Behavior Tree. The order of applying these rules is governed by higher level composite nodes, which resembles the logical reasoning process of engineers. It is also the first time that the dependency of joint variables, an important issue in inverse kinematics analysis, is automatically tracked in graph form. Besides generating closed-form solutions, IKBT also explains its solving strategies in human (engineers) interpretable form. This is a proof-of-concept of using Behavior Trees to solve high-cognitive problems.

1 Introduction

Symbolic inverse kinematics analysis is a non-trivial task critical for operation and design of robot manipulators as well as animated characters. The common misconception in robotics community is that analytical inverse kinematics problem is solved. In reality the numerical solutions are often substituted for closed form symbolic solutions, where all elements in the desired end effector matrix are real numbers and solutions for joint variables are numerical values, using two major methods: A) gradient descent searches for a set of joint angles/length that minimize the cost function. Convergence of gradient descent can depend on the starting value and only generates one solution. Most existing software packages [Corke, 1996; Kelmar and Khosla, 1990] use a version of gradient descent as their core algorithm. The shared limitations include finding only one of the multiple solutions, convergence depending on the starting value, and problems with convergence near singular configurations. B) Solving multivariant polynomial equations [Manocha and Canny, 1994; Murray *et al.*, 1994]. Though this method generates multiple possible solutions, it fails when the augmented transformation matrix is ill-conditioned, which is unavoidable in practice. And this method is often DOF-specific.

Comparatively, closed-form inverse kinematics analysis overcomes all these shortcomings of the numerical methods, but it's difficult to automate conceptually, because of the high-level symbolic reasoning needed. Though many advances have been made in the field of AI (especially deep neural networks) through increased capability of handling large numerical computations, fewer was made on symbolic reasoning. Several groups have attempted to automate symbolic inverse kinematics analysis starting in the 1990's [Herrera-Bendezu *et al.*, 1988; Halperin, 1991], which laid the foundation for our work. However, up until now, there was no generalized, expandable, and human explainable AI agents capable of solving this problem.

In this work we develop an AI agent that solves closed-form inverse kinematics with the following goals: The agent

- Should solve closed-form inverse kinematics with a generalized algorithm applicable to most serial chain robot arms, without assumptions of configuration or degree-of-freedom
- Should explicitly use common knowledge that engineers use when solving the inverse kinematics problems, such as trig identities or the method of determinants, rather than relying on tricks for specific kinematic configura-

*The full version of this paper has been published as [Zhang and Hannaford, 2019], on *JAIR* link here https://doi.org/10.1613/jair.1.11592

tions

- Should have search and apply suitable knowledge or rules to equations containing unsolved joint variables as human experts do.
- Should be able to explain its solving strategy in an easy to interpret format
- Should be extensible and modifiable

To address these goals, we adapt Behavior Trees to construct an expert system, "IKBT", having the logical reasoning power to solve inverse kinematics symbolically without human supervision. A Behavior Tree - initially popular in video game AI, models intelligent agent behavior by incorporating specific tasks into action leaves [Lim *et al.*, 2010; Marzinotto *et al.*, 2014; Colledanchise *et al.*, 2017; Colledanchise *et al.*, 2016]. Behavior Trees have the advantages of composability and scalability compared to finite state machines.

The main contributions of this work [Zhang and Hannaford, 2019] are:

- We compactly encode the inverse kinematics logic and solution strategy in a Behavior Tree
- We code each knowledge-based solver into a modular leaf, forming a "tool box" which is organized and applied to equations and intermediate results by the Behavior Tree. The structure is readily extensible.
- IKBT generates a dependency *graph* of joint variables in the solutions, which specifies all possible poses. Tracking these dependencies facilitates grouping variables into distinct solutions, essential to downstream control softwares for robots.
- IKBT successfully solves complicated robots, such as the 6-DOF commercial robot manipulator PUMA 560 and successfully solved 18 out of 19 test robots, with recent expended support to solve the popular UR series robots.
- On average, IKBT generates symbolic solutions and source code in a few minutes on a normal PC. The same work often takes a human expert hours to complete.
- IKBT generates a report of its results and solution method in LaTeX, and generates code in Python and C++, creating functions which implement the derived solutions including domain (reachability) checking of numerical inputs.
- Inverse kinematics solutions from IKBT are verifiable with numerical computations (facilitated by the IKBT code generator).
- Implementation in a modern open-source, cross-platform, programming language (Python). IKBT requires few dependencies outside of the standard Python distribution (mainly the symbolic manipulation package `sympy` and the unit testing framework `unittest`).

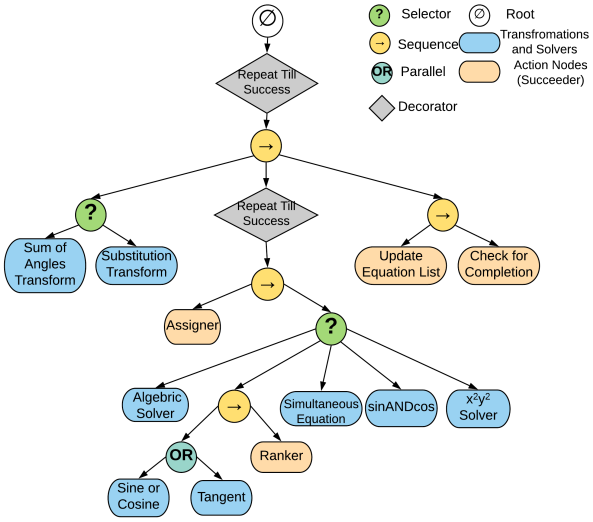

Figure 1: IKBT Structure. Node type explanation: Action nodes (leaves) carry out specific tasks, and returns SUCCESS or FAILURE. Succeeder is a special type of action nodes that only returns SUCCESS. Selector node ticks its children in turn, returns SUCCESS and stops if one of the children succeeds, otherwise returns FAILURE. Sequence node only returns SUCCESS if all its children succeed. Parallel node tries out all its children regardless of their return status, returns SUCCESS if any child succeeds.

2 Work Flow and Architecture

A Forward kinematics module computes symbolic kinematic equations to be solved ($T_d = T_s$) given the input DH parameters. Then the equations are evaluated for closed-form inverse kinematics solutions to each joint variable. Upon solving a robot, along with the solutions, a dependency graph, Latex report, and Python/C++ code are generated as convenience features.

The work reported here is the first to our knowledge to use Behavior Trees to encode algorithms for reasoning about and solving mathematical equations symbolically. When implementing intelligent behavior with Behavior Trees, the designer of a robotic control system breaks the task down into modules (Behavior Tree leaves) which return either "success" or "failure" when called by parent nodes. Higher level nodes define composition rules to combine the leaves including: Sequence, Selector, and Parallel node types which also return "success" or "failure". A Sequence node defines the order of execution of leaves and returns success if all leaves succeed in order. A Selector node (called "Priority" by some authors) tries leaf behaviors in a fixed order, returns success when a node succeeds, and returns failure if all leaves fail. We also implemented a "Parallel" node (represented as "OR" in Fig. 1), which executes all leaves regardless of their return status, and returns success if any one of the leaves succeeds. The IKBT structure used for our current results is shown in Fig.1.

For detailed explanations on rule-based solvers, see [Zhang and Hannaford, 2019].

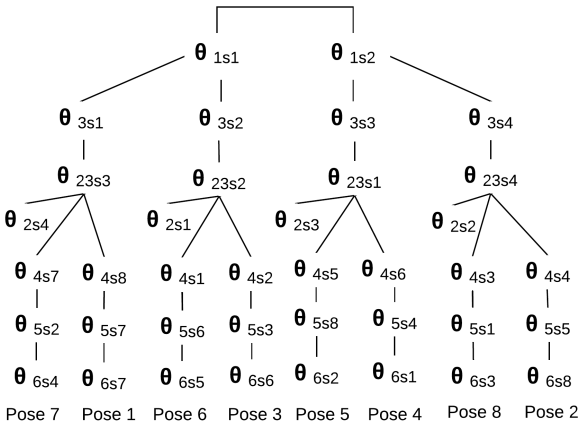

Figure 2: PUMA 560 Solution Graph.

number of DOF	Test examples	Solved
4	4	4
5	10	10
6	5	4

Table 1: IKBT test results

3 Solution Graph and Result Verification

IKBT provides solution graph to track the dependency among joint variables. IKBT traces these dependencies in a mathematical sense, based on their symbolic solution equations. Intuitively, in the physical world, it can be interpreted as when one joint is set to a new value, the dependent joints values changes with it, in order to achieve certain end effector position and orientation. The solutions produced by inverse kinematics are typically interdependent in that results obtained early in the process are used to solve later results. In principle, it is possible to substitute these dependencies until there are no joint variables on the right hand side, but this makes the solutions difficult to compare with previously published hand solutions. Thus the two sources of multiple solutions are: A) each joint variable may have multiple solutions due to its solver's characteristic; and B) dependence of the solution on other solved joint variables. On top of basic dependency tracking, IKBT also preforms redundancy elimination and automatically groups variables. These are important for outputting correct sets of joint variable solutions of all possible robot configurations. An example solution graph (for commercial robot Puma 560) is shown in 2. More examples and detailed explanations can be found in [Zhang and Hannaford, 2019].

After solving an inverse kinematics problem, a few simple steps can be taken to verify that the solutions are correct, as shown in 3.

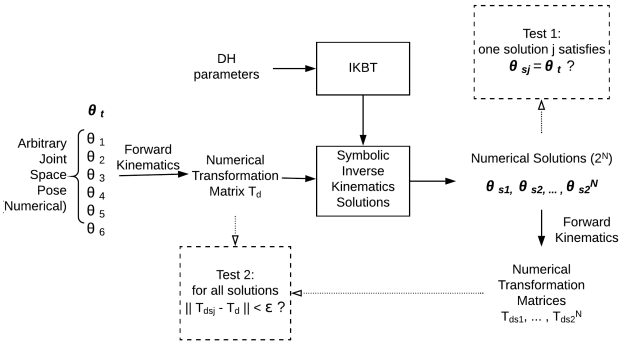

Figure 3: Result Verification. A numerical 4x4 homogeneous transformation matrix, T_d, is constructed from a reachable pose. Numerical joint space poses are computed from T_d using the closed-form solutions. For each solution pose, forward kinematics is calculated. The resulting transform matrices are compared against the original matrix, matching value is indicative of correct IKBT inverse kinematics analysis. 2^N indicates the number of solutions is always even.

4 Results and Discussion

We tested IKBT on many sets of DH parameters, representing serial arm robot designs (including commercial robots, and solved design examples from student homework), the successful solving rate is listed in Table 1. As the DOF number increases, the problem becomes more complex and the success rate decreases. In general it solves most of the robots, up to 6 DOF. Note that IKBT can solve robots regardless of their configurations, e.g. IKBT does not require robots having three intersecting axes. Detailed examples including analytical solutions with variable dependency graphs can be found in [Zhang and Hannaford, 2019]. Code is available at github.com/uw-biorobotics/IKBT.

The rule-based solvers included in IKBT's toolbox are commonly employed by human experts when solving inverse kinematics problems. IKBT's Behavior Tree represents an interpretable strategy - vital for judging many AI applications. This makes it easier to examine the correctness of the solution and the strategy formulating process.

IKBT constructs a generalized solving scheme applicable to an entire class of problems, using a small number of knowledge leaves. While most of current AI work focuses on recognizing and understanding scenarios, Behavior Trees emerge as a path to an equally vital component - combinatorial reasoning. Such logical reasoning enables AI agents to use limited knowledge base to solve problems of much larger magnitude. Intuitively, combinatorial reasoning is how human interact with the world: we decompose a unseen problem into solvable parts, then piece together modular knowledge to solve the larger problem. We don't get re-trained from scratch whenever we face new problems.

Leveraging combinatorial reasoning of behavior tree and the learning capabilities of modern machine learning techniques (especially unsupervised), we might be on our way to unlock the next level of autonomy.

Acknowledgements

We thank Dr. Steve Tanimoto for the invaluable advice on identifying the right audience and publication venue.

References

[Colledanchise *et al.*, 2016] Michele Colledanchise, Alejandro Marzinotto, Dimos V. Dimarogonas, and Petter Ogren. The advantages of using behavior trees in multi-robot systems. In *International Symposium on Robotics (ISR)*, June 2016.

[Colledanchise *et al.*, 2017] M. Colledanchise, R. Murray, and P. Ogren. Synthesis of Correct-by-Construction Behavior Trees. In *Intelligent Robots and Systems (IROS 2017), 2017 IEEE/RSJ International Conference on*, pages 1482–1488, Sept 2017.

[Corke, 1996] P. I. Corke. A robotics toolbox for matlab. *IEEE Robotics Automation Magazine*, 3(1):24–32, Mar 1996.

[Halperin, 1991] Dan Halperin. Automatic kinematic modelling of robot manipulators and symbolic generation of their inverse kinematics solutions (extended abstract), 1991.

[Herrera-Bendezu *et al.*, 1988] L. G. Herrera-Bendezu, E. Mu, and J. T. Cain. Symbolic computation of robot manipulator kinematics. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 993–998 vol.2, Apr 1988.

[Kelmar and Khosla, 1990] Laura Kelmar and Pradeep K. Khosla. Automatic generation of forward and inverse kinematics for a reconfigurable modular manipulator system. *Journal of Robotic Systems*, 7(4):599–619, 1990.

[Lim *et al.*, 2010] Chong-U Lim, Robin Baumgarten, and Simon Colton. Evolving behaviour trees for the commercial game defcon. In *European Conference on the Applications of Evolutionary Computation*, pages 100–110. Springer, 2010.

[Manocha and Canny, 1994] D. Manocha and J. F. Canny. Efficient inverse kinematics for general 6r manipulators. *IEEE Transactions on Robotics and Automation*, 10(5):648–657, Oct 1994.

[Marzinotto *et al.*, 2014] Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Petter Ogren. Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5420–5427. IEEE, 2014.

[Murray *et al.*, 1994] Richard M. Murray, S. Shankar Sastry, and Li Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.

[Zhang and Hannaford, 2019] Dianmu Zhang and Blake Hannaford. Ikbt: Solving symbolic inverse kinematics with behavior tree. *Journal of Artificial Intelligence Research*, 65:457–486, 07 2019.