

Fusion IK

Solving Inverse Kinematics with Evolution and Deep Learning

Steven Rice^a, Ahmed Azab^b, Sherif Saad^a

^aSchool of Computer Science

^bProduction & Operations Research Lab
University of Windsor

June 18, 2024



University
of Windsor

Table of Contents

- 1 Introduction
- 2 Related Works
- 3 Fusion IK Algorithm
- 4 Experimental Results
- 5 Conclusion

Industrial Robots

- Typically have six axes or joints.
 - Degrees of freedom (DoF).
- The end effector is the end of the arm we want to position.
- Need high repeatability and efficient movements.

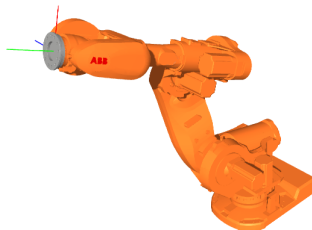


Figure: ABB IRB 7600 [ABB, nd]

Redundant or Extended Robots

- Beyond six degrees of freedom.
- Help overcome singularities and allow for greater flexibility and maneuvering around obstacles.
- Have recently been used in self-reconfiguring modular robots.
[Romanov et al., 2021, Pavliuk et al., 2020]

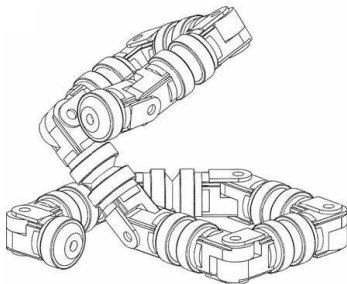


Figure: MARS Modular Robot [Pavliuk et al., 2020]

Robot Movement

- Forward Kinematics (FK)
 - Define joints values and then get the end effector transform.
 - “If joints are set to these values, where does the robot end up?”
- Inverse Kinematics (IK)
 - Define the desired end effector transform and then get the joint values required to do so
 - “If I want the robot to be here, what do I need to set the joints to?”
- Overall, inverse kinematics is more useful for motion planning.

The Problem

- Need to solve the inverse kinematics for robots.
- Requires either solving custom solutions, working within or communication with programs which have the required solutions, or using implementations that may not truly reflect the real robot.
 - Can impede development time.
 - Limits portability.
 - Can produce inaccurate results.

Analytic Solvers

- Solves the inverse kinematics similar to how a human would.
- Produce very efficient solutions with a high degree of accuracy.
- Do not work well beyond six or seven degrees of freedom.
- Often have complex setups compared to other methods.
 - Reduces iteration time and portability for practical use.
 - The results may not be optimal.
- IKFast. [Diankov, 2010]

Evolutionary Solvers

- Often based on Genetic Algorithms (GA) or related methods.
- Can solve kinematic chains of any length.
- Will generate different solutions every run.
 - Uncertain accuracy and low repeatability.
- Bio IK is one of the most successful evolutionary methods.
[Starke, 2020]
 - Evolves a random population seeded with the robot's initial joint values.
 - The seed is used to help keep result joint movements minimal.

Deep Learning

- Have shown acceptable accuracies but are not perfect. [Daya et al., 2009]
- Deep learning requires a data set for training. [Daya et al., 2009]
 - Past methods have been trained on data generated from analytic models. [Köker et al., 2004]
- A process using Generative Adversarial Networks (GANs) reduced the amount of analytic data required while still being successful. [Ren and Ben-Tzvi, 2019]
 - The analytic model was still present for some initial data generation. [Ren and Ben-Tzvi, 2019]

Next Steps

- Evolutionary models are great at reaching a solution.
 - May not produce optimal or consistent results.
- Deep learning models produce consistent, but not perfect, results.
 - The existence of an analytic model to train them makes deep learning solutions redundant for practical applications.
 - Why would you use a deep learning model that is likely not perfect when you already have a perfect analytical model?
- What if we could combine the accuracy of evolutionary models with the consistency of deep learning models?
- Use an evolutionary model to generate a data set for deep learning completely removing the need for an analytical model.

Process

① Data Set Generation

- Use the evolutionary algorithm Bio IK to generate a data set consisting of the inverse kinematics solutions for reaching multiple target transforms.

② Network Training

- Train a deep learning model on the generated data set.

③ Run Fusion IK

- Run inference on trained the network.
- Pass the results for the joints from the network as a seed for Bio IK to complete the inverse kinematics solution.

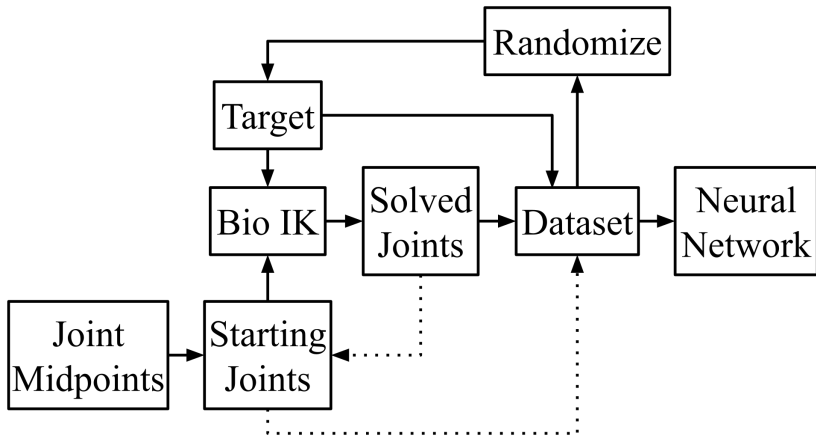
Data Set Generation

- 1 Get the robot's current joint values as the starting joint values.
- 2 Set the robot to a random pose and store the end effector transform as the target to reach.
- 3 Run Bio IK multiple times to reach the target, passing in the starting joint values as the seed.
- 4 Check the move time for each attempt.
- 5 Add the fastest result to the data set, discarding the attempt if no.
 - We want the deep learning to learn efficient moves so they are consistently produced.
- 6 Move to robot to the fastest result to be the starting pose for the next iteration.
- 7 Repeat for as many instances required for the data set.

Standard Fusion IK

- The network for a robot with N degrees of freedom will have $N+6$ inputs and N outputs.
 - One input for each of the N joints.
 - Three inputs for the Cartesian position of the target to reach.
 - Three inputs for the Euler angles orientation of the target to reach.
- The network is designed to be fast and thus only consists of a single layer of neurons.

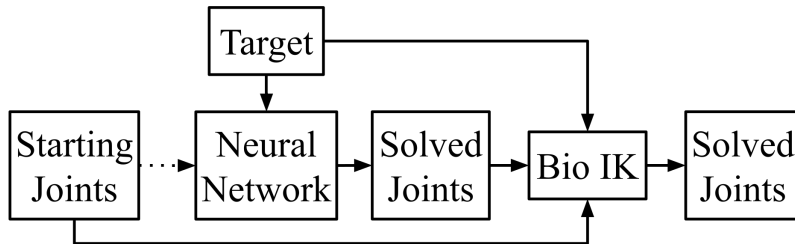
Fusion IK Algorithm



The dotted lines do not apply to the Minimal Fusion IK variation.

Figure: Fusion IK Training Process

Fusion IK Algorithm



The dotted lines do not apply to the Minimal Fusion IK variation.

Figure: Fusion IK Diagram

Minimal Fusion IK

- The network for the minimal variation only has six inputs.
 - Passing the N starting joint values may result in too much information for the network to fit.
 - Pass only the minimal information being the position and rotation to the target.
 - In theory offers higher accuracy at the cost of being less efficient.
- The training process is modified to always start Bio IK from the joint midpoints rather than the current pose of the robot.
 - Optimize around the average pose the robot will be in.

Exploitative Fusion IK

- Within the Bio IK process, during every generation of evolving the population, if the exploitation of an elite is unsuccessful and thus shows no improvement, it is replaced with a new random member. [Starke, 2020]
- The exploitative variation instead runs this member of the population through the network instead and uses that result as the new member.

Iterative Fusion IK

- The Bio IK process will be restarted after a given number of generations.
- The fittest members of the population will have their joint values run through the network.
- These re-seed the Bio IK process.

Exploitative Iterative Fusion IK

- Combines the exploitative and iterative methods.

Process

- Tested all Fusion IK variations against Bio IK on an ABB IRB 7600 and a simulated 20 degree of freedom robot.
- Data sets of 5,000 entries were generated for each robot.
- Testing then involved attempting to reach 5,000 random targets on each robot.
- A timeout of 1,000 milliseconds was given for all algorithms.
 - Algorithms would continue to search for better solutions until the time limit was reached.

Experimental Results

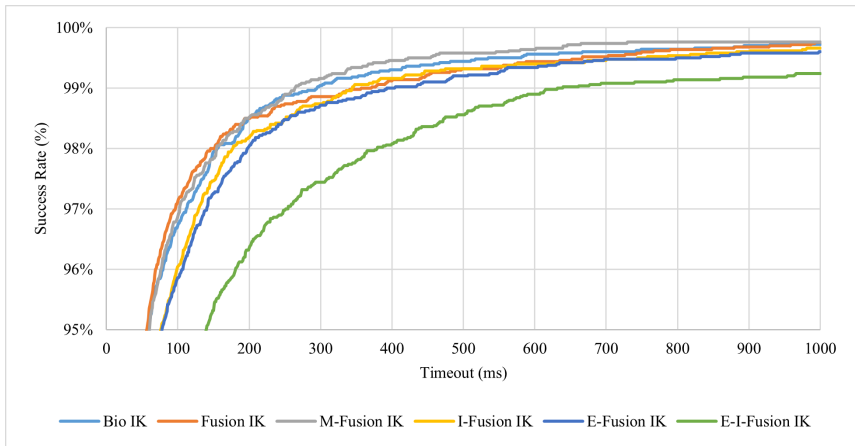


Figure: ABB IRB 7600 Success Rate over Timeout

Experimental Results

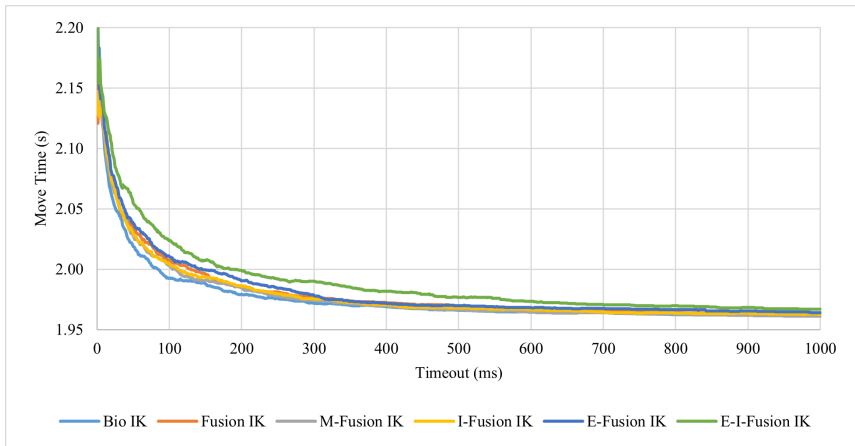


Figure: ABB IRB 7600 Move Time over Timeout on Successful Moves

Experimental Results

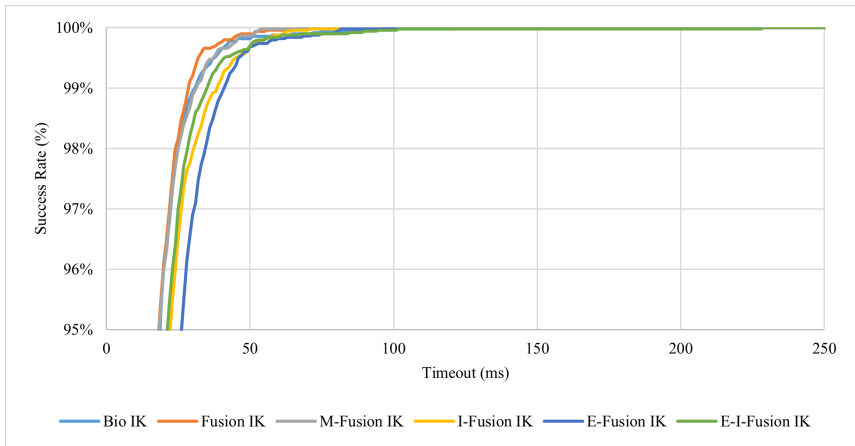


Figure: 20 DOF Success Rate over Timeout

Experimental Results

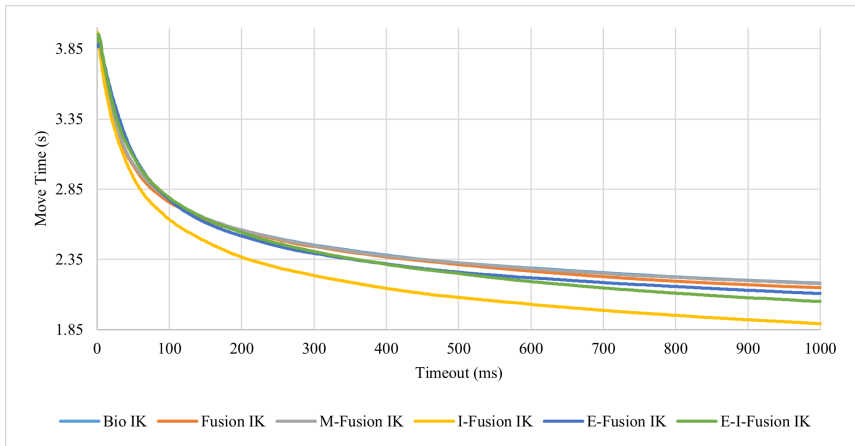


Figure: 20 DOF Move Time over Timeout on Successful Moves

Experimental Results




Algorithm	Success Rate (%)	Average Move Time (s) on Successful Moves
Bio IK	100%	2.179943703 s
Fusion IK	100%	2.149641041 s
M-Fusion IK	100%	2.177154016 s
I-Fusion IK	100%	1.893386547 s
E-Fusion IK	100%	2.10716028 s
E-I-Fusion IK	100%	2.050601622 s

Table: 20 DoF Results at 1,000 ms

Conclusion

- Exploitative Iterative Fusion IK performed noticeably worse than other variations.
- No Fusion IK variation exhibited any improvements over Bio IK on traditional industrial robots.
- Iterative Fusion IK showed a 14% improvement in move time over Bio IK on the 20 DOF robot after a 1,000 millisecond timeout.
- Fusion IK, especially Iterative Fusion IK, has shown itself to be a viable method for performing inverse kinematics on extended kinematic chains.
- With hyper-parameter tuning, further improvements could be exhibited for extended kinematic chains, but also potentially allow for Fusion IK to improve upon traditional industrial robots.

References I

-  ABB (n.d.).
IRB 7600.
-  Daya, B., Khawandi, S., and Akoum, M. (2009).
Applying Neural Network Architecture for Inverse Kinematics Problem
in Robotics.
Saida, Lebanon. Institute of Technology, Lebanese University.
-  Diankov, R. (2010).
Automated Construction of Robotic Manipulation Programs.
Pittsburgh, Pennsylvania, USA. The Robotics Institute Carnegie
Mellon University.

References II



Köker, R., Öz, C., Çakar, T., and Ekiz, H. (2004).

A study of neural network based inverse kinematics solution for a three-joint robot.

Sakarya, Turkey. Department of Computer Engineering, Sakarya University.



Pavliuk, N., Saveliev, A., Cherskikh, E., and Pykhov, D. (2020).

Formation of Modular Structures with Mobile Autonomous Reconfigurable System.

Proceedings of 14th International Conference on Electromechanics and Robotics Zavalishin's Readings.

References III



Ren, H. and Ben-Tzvi, P. (2019).

Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks.

Blacksburg, VA, USA. Department of Mechanical Engineering, Virginia Tech.



Romanov, A. M., Yashunskiy, V. D., and Chiu, W.-Y. (2021).

A Modular Reconfigurable Robot for Future Autonomous Extraterrestrial Missions.

IEEE Access.



Starke, S. (2020).

Bio IK: A Memetic Evolutionary Algorithm for Generic Multi-Objective Inverse Kinematics.

Hamburg, Germany. Universität Hamburg Faculty of Mathematics, Informatics and Natural Sciences Department of Informatics



University
of Windsor