NODEIK: Solving Inverse Kinematics with Neural Ordinary Differential Equations for Path Planning

Suhan Park1, Mathew Schwartz2, Jaeheung Park3*

1Department of Intelligence and Information, Seoul National University, Seoul, 08826, Korea (psh117@snu.ac.kr)
2College of Architecture and Design, NJIT, New Jersey, 07102, USA (cadop@njit.edu)
3Department of Intelligence and Information, ASRI, RICS, Seoul National University, Seoul, 08826, Korea, Advanced Institutes and Advanced Institutes of Convergence Technology (AICT), Suwon, 16229, Korea (park73@snu.ac.kr) * Corresponding author

Abstract: This paper proposes a novel inverse kinematics (IK) solver of articulated robotic systems for path planning. IK is a traditional but essential problem for robot manipulation. Recently, data-driven methods have been proposed to quickly solve IK for path planning. These machine learning-based models can handle a large amount of IK requests at once by leveraging the GPU. However, such methods suffer from reduced accuracy and considerable training time. We propose an IK solver that improves accuracy and memory efficiency with continuous normalizing flows by utilizing the continuous hidden dynamics of a Neural ODE network. The performance is compared using multiple robots, and our method is shown to be highly performant on complex (including dual end effector) manipulators.

Keywords: kinematics, neural networks, robotics, trajectory, path planning

1. INTRODUCTION

An Inverse kinematics (IK) problem is to find the joint space solution that satisfies the given Cartesian pose constraint. This process is important as most real-world tasks, such as furniture assembly [16] and robotic manufacturing [19], have Cartesian targets and require a meaningful method of determining a corresponding joint configuration to achieve the desired end-goal. By extending a single IK solution into a series of multiple IK solutions, one can construct a method for moving along arbitrary locations in Cartesian space – creating a path of motion for a robot. For this reason, IK has been a topic of robotics research for decades, and the improved computational speed, configuration options, and accuracy have been the subject of numerous open-source initiatives. These include e.g., an analytic inverse kinematics solver [6] for six degree-of-freedom (DoF) robots and a solver using numerical optimization [3]. Although the analytical IK solution is fast, it is limited to applications of only specific robots that satisfy a configuration and DOF requirement. In particular, the complexity of the IK solver drastically increases when redundant degrees of freedom are present, as weighting and other priorities must be implemented to reduce the solution space. On the other hand, numerical optimization methods can be used in all robots, but suffer from computational complexity. Therefore, while under certain circumstances these analytic and numerical methods may be suitable, the significant overhead in time for solving multiple IK requests (e.g., for a Cartesian space path) limits their use in many online systems.

Finding joint configurations for the given Cartesian space path is a constrained motion planning problem [14], which is a known complex problem due to the implicit constraint on the path. The end-effector pose constraint is nonlinear and implicit for most robots. Therefore, searching for a path w.r.t., the constraint has a considerable time cost. Prior studies have shown that the constrained motion planning problem can be solved faster using IK [4], [11]. Therefore, batch IK predictions can enhance searching for a path in the end-effector pose constrained space and provide various paths that can be used as a warm start (initial) path.

Recently, data-driven methods [7], [18] have been proposed to quickly find the multiple solutions required for path planning. By utilizing generative models, such as conditional variational autoencoder (CVAE) and conditional generative adversarial network (CGAN), [1] improved the sample drawing for constrained motion planning. However, CVAE and CGAN have known issues– manifold mismatch and mode collapse (CGAN)– which eventually reduce available space and diversity. Kim and Perez [13] introduced an autoencoder style inverse kinematics network and density estimator using normalizing flows (NF). Similar to our work, IKFlow [2] directly utilizes the conditional NF to improve the accuracy of the generated inverse kinematics. However, IKFlow needs extensive model parameters for highly accurate prediction due to the model complexity depending on the layer
width and depths.

On the other hand, continuous layer models are known for their capacity to represent complex distributions with a relatively simple model. The continuous model can be structured using an ordinary differential equation (ODE) with hidden neural dynamics (known as a Neural ODE (NODE)), and normalizing flows with NODE are called continuous normalizing flows (CNF) [5].

This paper proposes a novel IK method using a Neural ODE network for lightweight and accurate inverse kinematics generation. We refer to this process as NODEIK (Neural Ordinary Differential Equations for Inverse Kinematics). NODEIK is demonstrated on a redundant manipulator that has more than 6 DoF, as illustrated in Fig. 1, and dual-arm manipulators as shown in §4.2. The objective of NODEIK is to find multiple IK solutions quickly from given target poses and latent variables that can be sampled in tractable normal distribution as shown in Fig. 2. In addition, the proposed model deals with multi-target IK as well as single-target IK using the expressive NODE model. As an application of the batch-IK, we demonstrate efficiently (w.r.t., space and time) finding a joint path for the given Cartesian end-effector path.

The contribution of this paper is to reduce the number of model parameters while achieving better accuracy compared to state-of-the-art data-driven methods. In addition, multi-target IK has been demonstrated whose joint configuration comprises multi-branch kinematic chains (i.e., not just single chains). For example, a joint configuration for dual arm tasks of humanoid robots includes common joints, such as waist joints, connecting the pelvis frame to each end-effector frame. Finally, as the objective of the proposed method is a large number of IK solutions, we demonstrate path generation results of the Cartesian pose-constrained planning problem. Technically, integrating cutting-edge open source software, NODEIK is provided with GPU acceleration for both model training and forward kinematics. This enables online dataset generation in the training process and scalable GPU-accelerated solutions.

2. BACKGROUND

2.1 Inverse Kinematics

Inverse kinematics is, as the name implies, the inverse of forward kinematics defined by a function \( f : \mathbb{R}^n \rightarrow SE(3) \), mapping joint configurations \( q \in \mathbb{R}^n \) to an end-effector pose \( x \in SE(3) \), where \( n \) is the number of joints. Methods for solving the IK problem can be divided into analytic and numerical approaches. An analytic approach directly solves the joint configurations using geometric relationships. Unfortunately, a geometry analysis for inverse kinematics of a robot with more than 6 DoF is challenging and often not feasible. On the other hand, numerical methods can be trivially implemented and used for all single-chain robot configurations. The Newton-Rhapson method [9] and sequential quadratic programming [3] are

well-known algorithms for solving the inverse kinematics problem. However, these methods take a considerable time for a large number of solutions – a requirement for a smooth joint path through a Cartesian target path.

Critically important to the IK problem is the possibility of infinite solutions for a single target pose on a redundant robot configuration (one in which a typical arm has more than six DoF). Thus, introducing a latent variable \( z \in \mathbb{R}^n \), a unique solution to the IK problem can be determined as \( q = f^{-1}(x, z) \). Note that the latent variable has the same dimension as the configuration because Normalizing Flow (discussed in §2.2) require the same dimension for the invertibility of the function. In this paper, and prior work of [2], Normalizing Flow is used to transform between a latent variable \( z \) and joint configuration \( q \).

2.2 Normalizing Flows

Normalizing flows (NF) [17] are generative models with a tractable distribution for efficient density estimation and sampling in the complex distribution. Using a tractable probability density function, such as normal distribution, NF are used to estimate the complex target distribution. When a sample is drawn from a tractable distribution, for example, \( x_t \sim \mathcal{N}(0, I) \), a model with learnable parameters \( f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^D \) transforms the sample \( x_t \) to a desired state \( x_e \) in the complex distribution:

\[
x_e = f_\theta(x_t).
\]

(1)

Conversely, the training process proceeds through the inverse function of the model \( y_\theta = f_\theta^{-1} \), maximizing log-likelihood. Let \( x_t, x_e \) be the random variables in tractable and complex distribution and \( y_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^D \) be an invertible function where \( x_t = y_\theta(x_e) \). Then, \( p(x_t) \) is known by tractable distribution, and the change of variable theorem enables the computation of exact changes in probability:

\[
p(x_e) = p(x_t) \left| \det \frac{\partial f_\theta}{\partial x_t} \right|^{-1}.
\]

(2)

And the log probability is

\[
\log p(x_e) = \log p(x_t) - \log \left| \det \frac{\partial f_\theta}{\partial x_t} \right|.
\]

(3)
The model parameter $\theta$ is updated to maximize $\log p(x_c)$.

Prior work (IKFlow [2]) modeled the transform functions by a composition of the invertible functions:

$$f_\theta = f_1 \circ \cdots \circ f_{N-1} \circ f_N.$$  

(4)

Which requires the model depth $N$ to be large enough for an expressive model. However, this requirement to the number of parameters increases a model’s depth $N$ increases. In the following sections, we describe how to reduce such parameter requirements with an alternative form.

### 2.3 Neural ODE and Continuous Normalizing Flows

A neural network for hidden dynamics $h_\theta$ representing ODE:

$$\frac{dz(t)}{dt} = h_\theta(z(t), t),$$

(5)

referred to as a Neural ODE [5]—has an ability to estimate complex distribution using a relatively simple model. Starting from the initial value, which is used as an input layer, $z(t_0)$, the desired value $z(t_1)$ is the solution to the initial value problem at some time $t_1$:

$$z(t_1) = z(t_0) + \int_{t_0}^{t_1} h_\theta(z(t), t) dt.$$  

(6)

A black-box differential equation solver can compute the desired value, automatically evaluating the hidden dynamics $h_\theta$ as needed with the desired accuracy. This structure lets the hidden dynamics $h_\theta$ form continuous vector fields for transforming from the initial state to the desired state. The gradients for training are computed using the adjoint sensitivity method. Let $\mathcal{L}(z(t_1))$ be a loss function for the output of ODE solver:

$$\mathcal{L}(z(t_1)) = \mathcal{L}(z(t_0)) + \int_{t_0}^{t_1} h_\theta(z(t), t) dt.$$  

(7)

Utilizing the adjoint $a(t) = \partial \mathcal{L}/\partial z(t)$, the gradient with regard to $\theta$ for optimizing the hidden neural dynamics can be obtained by solving another differential equation:

$$\frac{d\mathcal{L}}{d\theta} = \int_{t_1}^{t_0} a(t)^T \frac{\partial h_\theta(z(t), t)}{\partial \theta} dt.$$  

(8)

This Neural ODE form can be used for NF structures, which is called continuous normalizing flows (CNF) [5]. Let $z(t_0)$, $z(t_1)$ be the random variables in tractable and complex distribution as in §2.2. Then Eq. (6) can be used as a transform function like Eq. (1). This form is easily invertible in reverting $t$. Thus, $h_\theta$ does not need to be bijective for NF. In addition, this structure can be seen as having infinitely deep recurrent layers, enabling the expressive model. Note that time $t$ does not indicate robot trajectory time but the degree of transformation from $z$ to $q$. We will describe $z(t = 0)$ as $z$ and $z(t = 1)$ as $q$ in the rest of the paper.

CNF do not require determinant computation of the Jacobian because the change in log-density follows the instantaneous change of variables formula [5]:

$$\frac{\partial \log p(z(t))}{\partial t} = -\text{Tr} \left( \frac{\partial h_\theta}{\partial z(t)} \right).$$  

(9)

Therefore, the target log-density is

$$\log p(z(t_1)) = \log p(z(t_0)) - \int_{t_0}^{t_1} \text{Tr} \left( \frac{\partial h_\theta}{\partial z(t)} \right) dt,$$

(10)

which is solved by ODE solver. To maximize the log-likelihood, the loss function is

$$\mathcal{L} = -\log p(z(t_1)).$$

(11)

Additionally, free-form Jacobian of reversible dynamics (FFJORD) [8] was proposed to reduce computation costs of the Jacobian trace of Eq. (9). This method utilizes Hutchinson’s trace estimator [10], which allows a scalable unbiased estimation of the log-density. While computing Tr costs $O(n^2)$, FFJORD can approximate the log-density at cost $O(n)$.

### 3. METHODS

#### 3.1 Overview

The proposed model exploits the aforementioned property of the NF and Neural ODE for lightweight and accurate IK prediction. The proposed method uses conditional CNF to learn the configuration distribution for a given Cartesian pose $x$. The proposed method is implemented on top of the SoftFlow [12] architecture for
conditioning and FFJORD [8] for fast training. NVIDIA Warp [15] is utilized to compute the forward kinematics using GPU resources.

3.2 Architecture

The proposed model utilizes conditional CNF using neural ODE:

$$\frac{dz(t)}{dt} = h_\theta(z(t), x, t).$$  \hspace{1cm} (12)

The hidden dynamics model are conditioned by $t$ and $x$ for the dynamics corresponding to the end-effector pose. During training, an online generated dataset is used. Joint configurations $q$ are sampled in uniform distribution within the joint range. And, the forward kinematics for the joints is computed online:

$$q \sim U(q_\text{min}, q_\text{max}),$$  \hspace{1cm} (13)

$$x = f(q),$$  \hspace{1cm} (14)

where $q$ and $q_\text{max}$ are the lower and upper bounds of the joints. Then, the trace of Jacobian Eq. (9) and the target value $z(0)$ of the current network model is computed as following relationship:

$$z(1) = q,$$  \hspace{1cm} (15)

$$z \triangleq z(0) = z(1) - \int_0^1 h_\theta(z(t), x, t) dt.$$  \hspace{1cm} (16)

The proposed model uses a normal distribution as a tractable base distribution: $z \sim N(0, I)$ to easily sample and calculate the log density $\log p(z(0))$. The model is directly updated by Eq. (8) with the loss function Eq. (11).

The structure of neural dynamics model $h_\theta$ is shown in Fig. 3. As CNF do not require the Jacobian determinant, the structure of hidden dynamics can be simply designed. Every hidden layer is conditioned in affine form.

3.3 Batch IK and Path generation

When acquiring the batch IK for $k$ targets, a set of latent variable $Z = [z_1, \ldots, z_{k-1}, z_k]^T \in \mathbb{R}^{n \times k}$ are randomly sampled in normal distribution and target poses $X = [x_1, \ldots, x_{k-1}, x_k]^T \in SE(3)^{k \times m}$ are set for the given path, where $m$ is the number of IK targets, e.g., dual end-effector targets for humanoid robots. The output of the NODEIK is the corresponding joint configurations $Q = [q_1, \ldots, q_{k-1}, q_k]^T \in \mathbb{R}^{n \times k}$. Fig. 2 illustrates this relationship. This process is a one-time calculation taking advantage of GPUs.

When generating IK solutions for the given Cartesian path, $z$ should change continuously. The same or continuous change of $z$ is required for the continuous joint solution path. However, because the NF can learn multi-modal distribution, the path can be discontinuous even with the continuous change of the input variables. Therefore, after batch generation, every path is checked whether the path is continuous.

4. RESULTS

4.1 Performance

The proposed method (NODEIK) and IKFlow were compared in accuracy, prediction time, and the number of model parameters. Accuracy was compared using a 7 DoF Franka Emika Panda robot. Model hyperparameters for IKFlow [2] were set to 1024 wide network width and 12 coupling layers (as this was mentioned in the original paper to have the best performance with this robot). The hidden dynamics of NODEIK consisted of 4 x 1024 width layers. The accuracy was evaluated from 1000 sampled Cartesian target poses $x$ and the predicted 250 joint solutions for each target pose as in the experiments of previous work [2]. By passing the joint solutions to the forward kinematics function, the end-effector pose of predicted joint solutions $\hat{x}$ was computed.

Our experiments showed NODEIK was able to learn faster and more accurately than IKFlow with the same number of data passed to the network during the training process. Fig. 4a illustrates the position error of the end-effector poses $||x_p - \hat{x}_p||_2$. The mean position error of NODEIK was approximately half compared to that of IKFlow. Fig. 4b shows the orientation errors, which is quaternion geodesic distance between two poses. In both images, the interquartile range of solution error is shown as shaded regions, with a dark line showing the mean of all solution errors. The orientation error of NODEIK was considerably lower than IKFlow. Outlier data with a large error existed among the generated solutions by IKFlow, which adversely affected the average value. Both plots show a tendency for NODEIK to find a majority of solutions within a distribution around the mean, while the large outliers of IKFlow suggest difficulty in producing
similar results at the same timestep.

A more significant difference in performance than just IK solution error was the difference in network parameter size. The number of model parameters of NODEIK was 3,316,320 for achieving the sub-centimeter error, while that of IKFlow was 50,934,364. NODEIK had 6.5% of the parameters created by IKFlow.

The limitation of NODEIK was in the prediction time, likely due to the Neural ODE process requiring multiple prediction steps of the hidden dynamics. The system setup for evaluation of the models was done on an Intel i7-10700K CPU and NVIDIA RTX 2080 Ti GPU. NODEIK and IKFlow took 85.93 ms and 14.97 ms, respectively, to find 1000 IK solutions. While NODEIK is slower for the inference step, these timing suggest it is still a very viable option for online tasks. In particular for the IK path-constrained problem which requires multiple IK solutions, NODEIK performed better than state-of-the-art numerical solvers such as TRAC-IK (C++) [3] and Klampt IK (Python) took 1 s and 2.6 s, respectively, for 1000 IK solutions.

4.2 IK for path generation

Path generation utilizing the advantages of batch IK was performed to evaluate the performance of the proposed model. Franka Emika Panda and TOCABI were used for the path generation. TOCABI is a 33 DoF humanoid robot, but, only upper body joints (arm and waist have eight and three joints, respectively) are used for IK. Target paths are given arbitrarily, which are circular paths. Orientation of the end-effector was fixed along the path. \( z \) was sampled in \( \mathcal{N}(0, I) \) once and all IK solutions were acquired using each target pose and the same \( z \). The discontinuous paths were rejected. Path errors are the average of position and orientation errors at each target pose. Notably, NODEIK was able to learn dual target IK for humanoid robots. 19 joints are used for dual target IK, which includes three shared waist joints.

| Robots | Position (mm) | Orientation (deg) |
|--------|---------------|-------------------|
|        | F | N | F | N |
| Panda  | 8.15 | 4.39 | 0.77 | 0.32 |
| TOCABI(S) | 27.11 | 14.31 | 6.64 | 0.82 |
| TOCABI(D) | - | 48.09 | - | 8.71 |

The average path errors are listed in Tab. 1. Overall, NODEIK showed less error than IKFlow. As the IK complexity increased, the errors increased. A path generated from NODEIK is shown in Fig. 5 as a qualitative result. The position and orientation errors differed according to the targets, but the relative change was small. Notably, the orientation error of NODEIK was similar on the Panda and TOCABI tests (in comparison to IKFlow). While the table shows a Dual Target task performance for NODEIK, we were not able to change the configuration of the repository offered by IKFlow to implement such experiments. This comparison may be interesting for future work.
5. CONCLUSION

This paper proposes a method of solving the Inverse Kinematics problem repetitively for path planning using a Neural ODE of a continuous normalizing flow. Experimental results show the NODEIK model to have 93% fewer parameters than the state-of-the-art method IKFlow, allowing the highly accurate learning-based IK for the low memory system. In the experiments using the same training dataset, NODEIK outperforms in position and orientation errors. Additionally, we demonstrated dual arm tasks for simultaneous manipulation using NODEIK. NODEIK showed a longer prediction time than the state-of-the-art method IKFlow, likely due to the ODE solver requiring multiple predictions of the hidden dynamics network. However, for the requirement of path planning to generate multiple IK solutions, NODEIK was significantly faster than numeric solvers. We expect that the performance of NODEIK can be improved with future work and contributions from the machine learning community by implementing optimized solvers.

Our NODEIK implementation can be found at: https://github.com/cadop/nodeik

REFERENCES

[1] Cihan Acar and Keng Peng Tee. Approximating constraint manifolds using generative models for sampling-based constrained motion planning. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 8451–8457, 2021.
[2] Barrett Ames and Jeremy Morgan. IKFlow: Generating diverse inverse kinematics solutions. IEEE Robotics and Automation Letters, 2022.
[3] Patrick Beeson and Barrett Ames. TRAC-IK: An open-source library for improved solving of generic inverse kinematics. In 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), pages 928–935. IEEE, 2015.
[4] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour. An integrated approach to inverse kinematics and path planning for redundant manipulators. In Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., pages 1874–1879, 2006.
[5] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. Advances in neural information processing systems, 31, 2018.
[6] Rosen Diankov. Automated construction of robotic manipulation programs. 2010.
[7] Aaron D’Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. In Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, Expanding the Societal Role of Robotics in the next Millennium (Cat. No. 01CH37180), volume 1, pages 298–303. IEEE, 2001.
[8] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In International Conference on Learning Representations, 2018.
[9] K Gupta and Kazem Kazerounian. Improved numerical solutions of inverse kinematics of robots. In Proceedings. 1985 IEEE International Conference on Robotics and Automation, volume 2, pages 743–748. IEEE, 1985.
[10] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. Communications in Statistics-Simulation and Computation, 18(3):1059–1076, 1989.
[11] Keunwoo Jung, Jiyeong Baek, Suhan Park, and Jaeheung Park. Motion planning for closed-chain constraints based on probabilistic roadmap with improved connectivity. IEEE/ASME Transactions on Mechatronics, pages 1–9, 2022.
[12] Hyeongju Kim, Hyonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. Softflow: Probabilistic framework for normalizing flow on manifolds. Advances in Neural Information Processing Systems, 33:16388–16397, 2020.
[13] Seungsu Kim and Julien Perez. Learning reachable manifold and inverse mapping for a redundant robot manipulator. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 4731–4737. IEEE, 2021.
[14] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Exploring implicit spaces for constrained sampling-based planning. The International Journal of Robotics Research, 38(10-11):1151–1178, 2019.
[15] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, March 2022. NVIDIA GPU Technology Conference (GTC).
[16] Suhan Park, Hae seong Lee, Seung yeon Kim, Jiyeong Baek, Keunwoo Jung, Hyoung Cheol Kim, Myeongsoo Kim, and Jaeheung Park. Robotic furniture assembly: task abstraction, motion planning, and control. Intelligent Service Robotics, pages 1–17, 2022.
[17] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In International conference on machine learning, pages 1530–1538. PMLR, 2015.
[18] Matthias Rolf, Jochen J Steil, and Michael Gien ger. Goal babbling permits direct learning of inverse kinematics. IEEE Transactions on Autonomous Mental Development, 2(3):216–229, 2010.
[19] Mathew Schwartz and Jason Prasad. Robosculp. In Rob—Arch 2012, pages 230–237. Springer, 2013.