CS230

Learning the Inverse Kinematics of a 6-DOF Concentric Tube Robot through a Generative Adversarial Network

Soham Sinha
Stanford University
sohams@stanford.edu

Abstract

Concentric tube robots (CTR's) are a highly complex robotic system due to a combination of both joint and transitional spaces as well as having high degrees of freedom. Due to their advantages in having a small form factor, ability to navigate in constrained spaces, there is a great need for a robust control system. Analytical inverse kinematics (IK) offer one such approach, but suffer from numerical problems and instabilities in its Jacobian. Furthermore, redundancy is not addressed in such solutions. Neural Networks offer an opportunity to approximate the inverse kinematics, as well as offer advantages that numerical solution cannot offer. In particular, we approach inverse kinematics through a simulation intensive method by bootstrapping the Jacobian to generate data to simulate both the Jacobian and the forward kinematics with a shallow feed-forward neural network. We then perform a standard optimisation of the positional error through feedback to solve for IK. Furthermore, to tackle the issue of redundancy and generate starting configurations, we explore utilising an ensemble Generative Adversarial Network to generate a variety of robot configurations such that the end-effector is close to the target point. By comparing with standard inverse kinematic approaches, from a pure function approximation from target end poses to robot configuration spaces, to a data-driven lookup process, we show that our mixture of Jacobian Learning + GAN offer relatively high accuracy (2.5% of total robot length) as well as low computational time (1 - 1.5 s/point), a significant improvement over literature values of 10% of total robot length for neural network based implementations.

1 Introduction

Concentric tube robots (CTR's) are a class of continuum robotics that are well suited for access into occlusive spaces whether it be in a minimal surgery setting or manufacturing in a complex mold (Figure 1) [4, 1]. The robot is constructed from sets of pre-curved superelastic (nickel-titanium or Nitinol) alloy tubes and are concentrically aligned which allow for rendering of complex 3-dimensional curves with a small form factor [4]. Achieving precise, accurate, and fast control of such robots is an ongoing research problem, especially closed-loop control which requires inverse kinematics of the robot [11, 8, 7].

2 Related Work and Overview of Methodology

Inverse kinematics of robotics has been explored in various forms, from numerical estimation through IVP and BVP methods, [8, 11], data-driven approaches with interpolation [12], multilayer perceptron

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

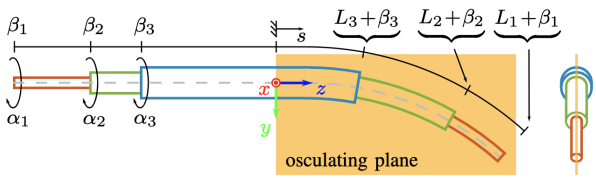

Figure 1: This figure shows the 3 possible translations $\beta_1, \beta_2, \beta_3$ and 3 rotations $\alpha 1, \alpha_2, \alpha_3$ of a 3-tube CTR for 6 degrees of freedom. In an application setting, the translations are achieved by stepper motors, and the rotations are by rotary encoders. Taken with no modifications from Grassman et. al [6]

deep learning with 2 hidden layers[6], numerical control and optimisation [8, 10, 2], to even rewriting of the parameter space in terms of differing coordinate systems from cylindrical to spherical and the configuration space in terms of quaternions to boost neural network performance; however, the results are limited by both accuracy ($\tilde{1}0\%$ of total length of the robot), redundancy issues (multiple configurations can lead to the same tip position), hysteresis, and sparse solution sets due to very weak pseudo-inverse of the Jacobian Matrix. In particular, simulation data has shown to be quite useful in determining performance of neural networks [3], as it allows for collection of sample points via Monte Carlo methods.

2.1 Optimisation via MLP Forward Kinematics Approximations

An alternate formulation of Inverse Kinematics is to think of it as an optimisation problem of reducing the error between the desired path position, and the end tip position of the robot.

In particular, given a set of robotic internal parameters $\{q_1, q_2, ...q_m\} \in \mathbb{R}^m$, and a set of target points $\{p_1, p_2, ...p_i, p_{i+1}, ...\} \in \mathbb{R}^n$, the problem can be reformulated as the following optimisation problem

$$\mathbf{q_i} = \text{argmin}_{\mathbf{q}} \mathcal{O}(\mathbf{p_i}, \mathbf{q}) = \text{argmin}_{\mathbf{q}} ||\mathbf{p_i} - \mathbf{p}(\mathbf{q})||_2^2 \tag{1}$$

We can solve the argmin by finding a suitable learning rate α and performing the following update

$$\mathbf{q_{i+1}} = \mathbf{q_i} - 2\alpha(\mathbf{p_i} - \mathbf{p}(\mathbf{q})) \cdot \frac{d\mathbf{p}}{d\mathbf{c}} \tag{2}$$

$$= \mathbf{q_i} - 2\alpha(\mathbf{p_i} - \mathbf{p}(\mathbf{q})) \cdot \mathbf{J}(\mathbf{q}) \tag{3}$$

with $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{n \times m}$ Guoxing Fang et. Al, in late 2020, detail a methodology of approximating the Jacobian as a neural network to ease up computation for accurate path planning of soft-robotics [5]. Furthermore, from previous results replacing the forward kinematics function $\mathbf{p}(\mathbf{q})$, with a neural network approximation significantly speeds up computation by a factor of 2-10 times [5, 3, 6].

2.2 Generative Adversarial Neural Network for Starting Configurations

However, a key issue still remains in the optimisation procedure outlined above - the generation of the first configuration to optimise from. Redundancy is a critical issue, as many configurations lead to the same points. A brute force search algorithm from simulation data can be used to generate a starting configuration for optimisation; however, it is computationally expensive. Teguh Lembono et. Al., in late 2020, detail a GAN Framework to generate valid robot configuration space parameters for a joint-space parametric robots (7-DOF Panda Manipulator and 28-DOF Humanoid Talos robots in Gym) [9]. A Generative Adversarial Network is a zero-sum game between two neural networks, a generator which generates 'false' data and tries to fool a discriminator which decides whether the generated data comes from the same distribution or not. Hence, a network's loss is another's gain. In particular, they used an ensemble of generators, each trained separately on the same dataset, to give the various possible configurations.

The difference between a standard GAN for images and this is that besides randomly sampled noise, a target position is given to the Generator to generate configurations that correspond to the target position, and some specialised loss functions are used, which are described in more detail in Section 3. The GAN framework is shown in Figure 2.

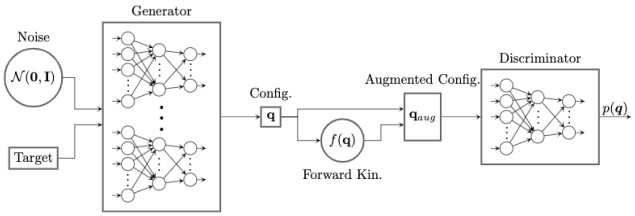

Figure 2: This figure shows GAN framework to generate valid CTR configurations for a specific target point. The generator consists of an ensemble of Neural networks, while the discriminator consists of a single neural network. Standard Gaussian noise is the input, as well as the target position. The output of the generator can be augmented with other constraints before given to the discriminator. Taken with no modifications from Lembono et. Al. [9].

3 Dataset and Features

A challenge of this project was generating data points from a CTR Robot. Due to some delays with the physical CTR robot being built in the Skylar-Scott Lab, simulation data approximating the real 3-tube 6-DOF robot was instead utilised. CTR Simulation data was generated using Cosserat Rod Theory, which has been the standard used in modelling CTR Robots [3, 8, 4]. Furthermore, due to amount of matrix multiplications involved in the system, and ease of vectorisation MATLAB R2020B was used to both generate and train neural networks. For more details on the theory, please see work by Burgner-Kahrs et. Al ([3]) and physical parameters of the tubes (Table 8.1), please see Appendix.

3.1 Concentric Tube Kinematic Constraints

The joint space of the CTR is defined by the following inequalities, with reference to symbols defined in Figure 1.

$$\alpha_i \in [-\pi, \pi) \tag{4}$$

$$\beta_i \in [-L_i, 0] \tag{5}$$

$$\beta_1 \leq ... \leq \beta_n \leq 0 \tag{6}$$

$$L_n + \beta_n \leq ... \leq L_1 + \beta_1 \tag{7}$$

$$\beta_1 \in [-L_1, 0], \beta_2 \in [-L_2, \beta_1], \beta_3 \in [-L_2, \beta_2] \tag{8}$$

where $i = 1$ denotes the innermost tube. A Monte-Carlo approach was taken to sample these parameters subject to the constraints, and 500,000 data points (s) corresponding to the end-effector poses of the CTR were generated (Figure 4). A 98-1-1 rule is followed for training, validation, and testing. The kinematics of the CTR is challenging because it has both a joint and a translational space which is bounded by inequalities. A picture of the configuration space of the robot and dataset image is given in the appendix.

3.2 Generation of Numerical Jacobian

Adapting Lembono et. Al [9], sample points to learn each of the columns of the Jacobian Neural network $\mathcal{N}_J$ Jacobian were numerically generated through the following equation.

$$\mathcal{N}_J^s \approx \frac{\partial \mathbf{p^s}(\mathbf{q})}{\partial q_k} = \frac{\mathbf{p}^s(..., q_k + \Delta q, ...) - \mathbf{p}^s(..., q_k - \Delta q, ...)}{2\Delta q} \tag{9}$$

There are 6 columns to the matrix, and a $\Delta q = 0.001$ was chosen. 50000 data points for each of the 6 parameters were generated with 98-1-1 split rule for training-testing-validation.

3.3 Loss Functions for Training

There are 4 sets of neural networks utilised in this project.

1. Forward Kinematics $\mathcal{N}_F$ - Mean squared Loss was used.

$$\ell_f = ||s - \mathcal{N}_F(q)||_2^2 \tag{10}$$

2. Jacobian Approximation $\mathcal{N}_J$ - Mean squared loss was used

$$\ell_J^s = ||\frac{\partial \mathbf{p^s(q)}}{\partial q_k} - \mathcal{N}_f(q)||_2^2 \tag{11}$$

3. GAN Approximation $\mathcal{N}_G, \mathcal{N}_D$ - Series of Standard and Non-Standard Losses

$$\ell_{SG} = -\frac{1}{n}\sum_{i=1}^{n}\log(p_{gi}) \qquad \text{Standard Generator Loss} \tag{12}$$

$$\ell_{SD} = -\frac{1}{n}\sum_{i=1}^{n}\log(p_{ri}) - \frac{1}{n}\sum_{i=1}^{n}\log(1-p_{gi}) \qquad \text{Standard Discriminator Loss} \tag{13}$$

$$\ell_{G1} = ||s - \mathcal{N}_G(I)||_2^2 \qquad \text{Target Loss based on Input I} \tag{14}$$

$$\tag{15}$$

Next, we have some costs associated with the kinematic constraints of the robot. For the rotational (α), and translational (β) parameters with lower l_b and upper limits l_u, we define a quadratic barrier function $f_b(x)$.

$$r_b(x) = \min(x - l_b, 0) \qquad \text{violation of lower bound} \tag{16}$$

$$r_u(x) = \max(x - l_u, 0) \qquad \text{violation of upper bound} \tag{17}$$

$$f_b(x, l_b, l_u) = 0.5(r_b(x)^2 + r_u(x)^2) \tag{18}$$

$$\tag{19}$$

4. Inverse Kinematics $\mathcal{N}_I$ Mean squared Loss was used as comparison.

$$\ell_f = ||q - \mathcal{N}_I(s)||_2^2 \tag{20}$$

4 Architecture, Algorithm, and Methods

Table 1: Architecture and Results

Neural Network	**Training with Adam Optimizer**				**Architeecture**				**Approximation Error**	
	N_{bs}	N_{es}	$\|S_{ts}\|$	λ	ψ	N_{ip}	N_h	N_{op}	MSE	Score
$\mathcal{N}_F$	128	200	490,000	0.0002	ReLU	6	(100,200)	3	0.2 mm	NA
$\mathcal{N}_J^s$	128	200	49,000	0.0002	ReLU	6	50	3	0.1 mm	NA
$\mathcal{N}_G$ (3)	20000	882	490000	0.002	ReLU	12 (9 noise + 3 target)	(500,250,100)	6	NA	0.23
$\mathcal{N}_D$	20000	882	490000	0.004	ReLU, Sigmoid (last)	6	(200), $dr(0.5)$, (100), $dr(0.5)$	1	NA	0.77
$\mathcal{N}_I$	128	300	490000	0.0002	ReLU	3	(500, 200, 100)	6	$e_\alpha = 3, e_\beta = 6$	NA

The architecture of the final neural networks are shown in Table 8.1 below and the batch size (N_{bs}), number of epochs (N_{es}), size of the training set ($|S_{ts}|$), learning rate λ with standard Adam parameters ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-8}$), activation function ($\psi$), number of output parameters (N_{op}), number of input parameters (N_{ip}), hidden layer architecture (N_h), all neuron count $(\cdot)$ correspond to fully connected layers, $dr(k)$ signifies dropout with probability k. All neural networks were trained with gradient descent, with Adam optimiser was used. Other activation functions were explored such as Tanh, Leaky ReLU, but ReLU gave the best results. Adam (Adaptive Moment Estimation) is particularly useful as it is adaptive - it stores an exponentially decaying gradient average, and is particularly well suited to the large amount of data being trained (500k). For the GAN, 3 Generators were trained, to create an ensemble generator, chosen to due to computational limitations of the author [9]. GAN network was trained until generator loss was below 10, and discriminator and generator had a score difference of > 0.5. The dimension of noise was chosen to be 9 (as it equals to 6 robot configurations + 3 end-effector positions), and it was fed to the Generator with the target position. We choose a higher learning rate for the discriminator as an experimental result for stability.

5 Experiments

The best way to evaluate the results is to see it in action in a selection of paths (n=3) in the CTR's configuration space and calculate the inverse kinematics for it - 3 methods are used - A straight Inverse Kinematic Approach using $\mathcal{N}_I$, a brute force data driven approach combined with optimisation using $\mathcal{N}_F, \mathcal{N}_J$, and the combined GAN Approach using optimisation $\mathcal{N}_F, \mathcal{N}_J, \mathcal{N}_G$. Path 3 is shown in Figure 3 for clarity, the remaining two are in the appendix. Experiment results are shown in Table 2.

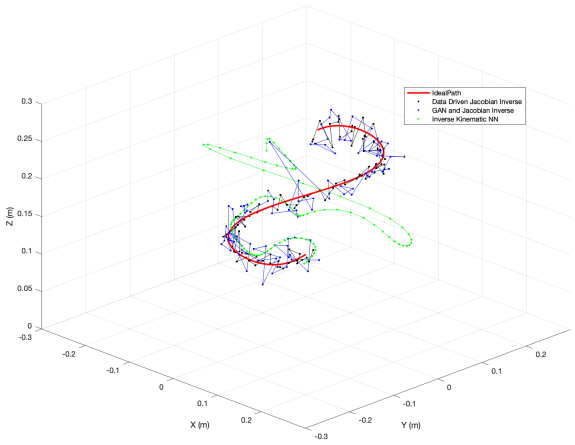

Figure 3: This figure shows the path following of the different models of inverse kinematics. 100 points along the path were considered and calculated via the three models. Red corresponds to the path. Green is the result of the inverse kinematic neural network, black is the data driven + Jacobian Learning, and blue is Jacobian + GAN approach.

Table 2: Model Results for 100 points along Desired Path

Model	**Path 1**		**Path 2**		**Path 3**	
	MSE (mm)	Time (s)	MSE (mm)	Time (s)	MSE(mm)	Time (s)
Inverse	9.87	30	14.41	30	15.02	30
Jacobian Learning + Data Driven	1.91	654	2.01	713	2.04	539
Jacobian Learning + GAN	4.93	128	5.37	142	5.55	137

6 Discussion and Future Work

From Table 2 and Figure 3, it is clear that the the solely the Inverse model is not sufficient, with high error even at low computation cost. The Data Driven model although accurate, is quite computationally expensive, taking close to 10 minutes to generate 100 points. The Jacobian Learning + GAN Approach although has some error, it is not as computationally expensive as the others. With respect to total % of entire robot body, the Jacobian Learning + GAN corresponds to 2.5%, which is significantly lower than results reported in literature, which is extremely promising for translation with a physical robot.

A main consideration of improvement would be the GAN, the authors utilised 5 generators in their ensemble; here, 3 was used due to computational limitations. Furthermore, the quadratic barrier functions for the loss may need to be rewritten to guarantee better results for the Generator. This leads to perhaps rewriting the parameter space of the robot as well. The difficulty of creating a convex, or semi-convex loss function for the inequalities is a significant barrier, and a large portion of future work will be devoted to exploring appropriate loss functions. In addition, hyperparameter tuning to improve performance is still relevant, as controlling the stability of a GAN is very difficult, with frequent mode collapse was observed. Furthermore, MATLAB is perhaps not the best environment to handle such a complex GAN,rewriting the implementation in tensorflow would probably be better as it is highly optimised [9].

7 Conclusion

Although GANs + Jacobian Learning offer a rather heavy neural network usage (10 in this report), it is evident that utilising a combination of standard optimisation techniques with GAN neural networks offer great promise in robot control. Particularly, redundancy, a critical barrier, can be potentially be reasonably handled using GANs. From the preliminary results in this report, there is potential for

GANs to generate robot configurations with more constrained parameter spaces. When translating simulation results with a physical robot, implementing a quick simulation-2-real lightweight network can potentially take care of domain shift issues of the transfer learning [5].

References

[1] Hessa Alfalahi, Federico Renda, and Cesare Stefanini. Concentric Tube Robots for Minimally Invasive Surgery: Current Applications and Future Opportunities. *IEEE Transactions on Medical Robotics and Bionics*, 2(3):410–424, 2020.

[2] Christos Bergeles and Pierre E. Dupont. Planning Stable Paths for Concentric Tube Robots. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3077–3082, 2013.

[3] Jessica Burgner-Kahrs, Hunter B. Gilbert, Josephine Granna, Philip J. Swaney, and Robert J. Webster. Workspace Characterization for Concentric Tube Continuum Robots. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1269–1275, 2014.

[4] Pierre E. Dupont, Jesse Lock, Brandon Itkowitz, and Evan Butler. Design and Control of Concentric-Tube Robots. *IEEE Transactions on Robotics*, 26(2):209–225, 2010.

[5] Guoxin Fang, Yingjun Tian, Zhi-Xin Yang, Jo M P Geraedts, and Charlie C L Wang. Jacobian-based learning for inverse kinematics of soft robots. *arXiv*, 2020.

[6] Reinhard Grassmann, Vincent Modes, and Jessica Burgner-Kahrs. Learning the Forward and Inverse Kinematics of a 6-DOF Concentric Tube Continuum Robot in SE(3). *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 00:5125–5132, 2018.

[7] Keshav Iyengar, George Dwyer, and Danail Stoyanov. Investigating exploration for deep reinforcement learning of concentric tube robot control. *International Journal of Computer Assisted Radiology and Surgery*, 15(7):1157–1165, 2020.

[8] Mohsen Khadem, John ONeill, Zisos Mitros, Lyndon da Cruz, and Christos Bergeles. Autonomous Steering of Concentric Tube Robots via Nonlinear Model Predictive Control. *IEEE Transactions on Robotics*, 36(5):1595–1602, 2019.

[9] Teguh Santoso Lembono, Emmanuel Pignat, Julius Jankowski, and Sylvain Calinon. Generative Adversarial Network to Learn Valid Distributions of Robot Configurations for Inverse Kinematics and Constrained Motion Planning. *arXiv*, 2020.

[10] Patrick Sears and Pierre E. Dupont. Inverse Kinematics of Concentric Tube Steerable Needles. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1887–1892, 2007.

[11] John Till, Vincent Aloi, Katherine E. Riojas, Patrick L. Anderson, Robert James Webster III, and Caleb Rucker. A Dynamic Model for Concentric Tube Robots. *IEEE Transactions on Robotics*, 36(6):1704–1718, 2019.

[12] Jie Wang, Di Zhang, Tao Ma, Shuang Song, Wei Liu, and Max Q.-H Meng. A New Solution for the Inverse Kinematics of Concentric-Tube Robots. *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, 00:234–239, 2018.

8 Appendix

8.1 Tube parameters

Table 1: Parameters for the Tubes used in Simulation Study

Parameter	Tube 1	Tube 2	Tube 3
Length (mm)	200	150	100
Curved Length (mm)	100	50	80
Inner Diameter (mm)	0.7366	1.0414	1.3335
Outer Diameter (mm)	0.8128	1.1176	2.0320
Stiffness (GPa)	50	50	50
Torsional Stiffness (GPa)	23	23	23
Curvature (1/m)	1/35	1/70	1/160

8.2 CTR Configuration Space and Dataset with Training Graphs

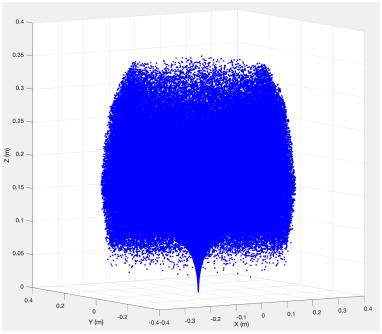

Figure 4: This figure shows the results of the Monte-Carlo sampling of the robot's configuration space with 500k samples (i.e dataset).

	x	y	z	t1	t2	t3	r1	r2	r3
2	-0.02537778	0.04283182	0.34691872	-0.48920134	-0.13505235	-0.04219040	5.84828175	3.44694373	3.34369367
3	0.12034270	0.14521268	0.38213310	-0.37709457	-0.00409680	-0.00329025	2.35968531	3.86895545	2.26276387
4	0.09419104	0.02557026	0.32909596	-0.49501447	-0.14196182	-0.02624431	2.76683803	5.79835374	1.28967778
5	0.08172623	0.18576694	0.39656752	-0.06983629	-0.01582022	-0.00189550	5.21509228	0.88783096	2.93550151
6	-0.18400892	-0.02701702	0.42570447	-0.07431969	-0.01915940	-0.00274528	5.09680908	2.59868552	4.97759789
7	-0.04548369	-0.20747069	0.39965540	-0.05581691	-0.02705638	-0.00052388	3.85432297	1.79933503	5.85252308
8	0.05981042	-0.17797291	0.45102494	-0.05780926	-0.02303002	-0.02007541	0.60859107	3.56536746	0.37093725
9	0.05226838	0.10011329	0.31776170	-0.28393735	-0.17103381	-0.00763622	3.38494284	5.56269620	2.77232598
10	0.09139019	0.17330758	0.36337052	-0.10347708	-0.07434684	-0.02162092	0.96029654	6.17314990	2.50470814
11	-0.10623818	0.07136923	0.27241382	-0.23029113	-0.17787562	-0.06277540	3.76696929	0.81010174	4.17274455

Figure 5: This figure shows a snippet of the dataset.

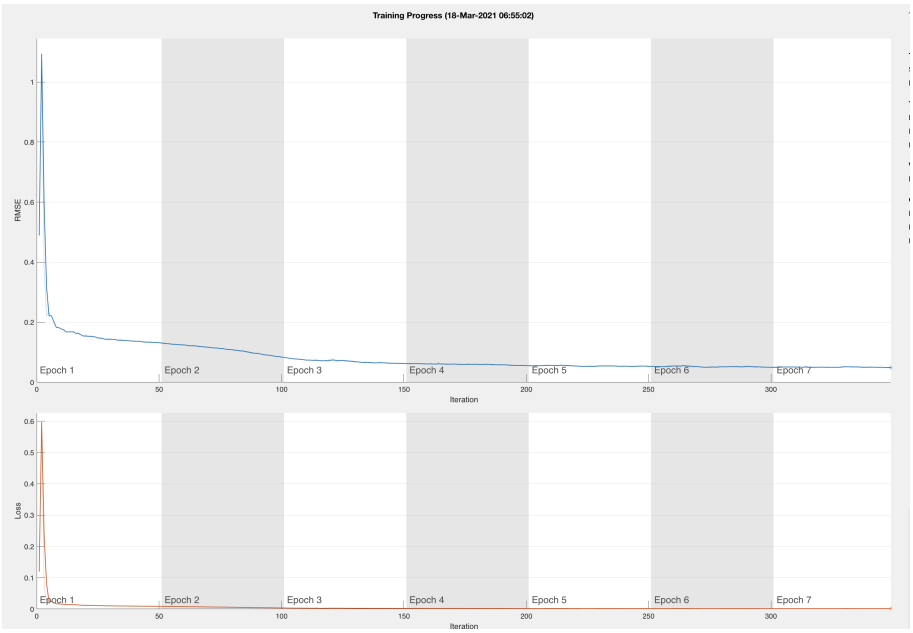

Figure 6: This figure shows training of a Jacobian Column Neural Network

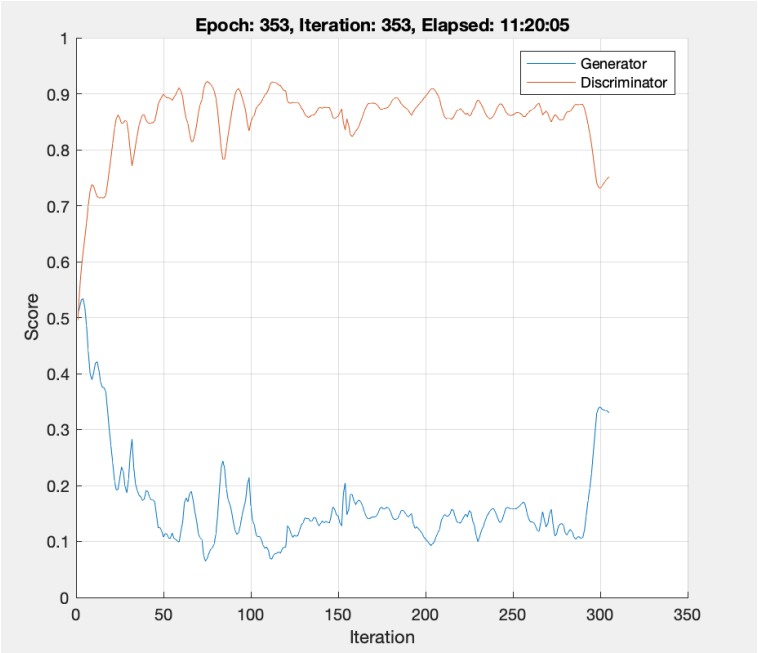

Figure 7: This figure shows a training session of a GAN with one generator.

8.3 Inverse Kinematic Algorithms

Algorithm 1: Inverse Kinematics: GAN

Result: Find the Robot Configuration Space Corresponding to Point using a GAN
INPUT: $p_{target}, \mathbf{maxiter}, \alpha$;
while $n <$ $\mathbf{maxiter}$ **do**
  $\mathbf{q_i} \leftarrow \mathrm{N}_G$;
  Evaluate $\mathcal{O}(\mathbf{p_{target}}, \mathbf{q_{gen}})$;
  $\mathbf{q_{i+1}} \leftarrow \mathbf{q_i} - 2\alpha(\mathbf{p_i} - \mathbf{p}(\mathbf{q})) \cdot \mathcal{N}_J$;
  $n = n + 1$;
end

Algorithm 2: Inverse Kinematics: DataSearch

Result: Find the Robot Configuration Space Corresponding to Point using a Greedy Data Search
INPUT: $p_{target}, \mathbf{maxiter}, \alpha$;
while $n <$ $\mathbf{maxiter}$ **do**
  $\mathbf{q_i} \leftarrow \mathbf{SearchSimulationData}(\mathbf{p_{target}})$;
  Evaluate $\mathcal{O}(\mathbf{p_{target}}, \mathbf{q_{gen}})$;
  $\mathbf{q_{i+1}} \leftarrow \mathbf{q_i} - 2\alpha(\mathbf{p_i} - \mathbf{p}(\mathbf{q})) \cdot \mathcal{N}_J$;
  $n = n + 1$;
end

8.4 Paths Studied

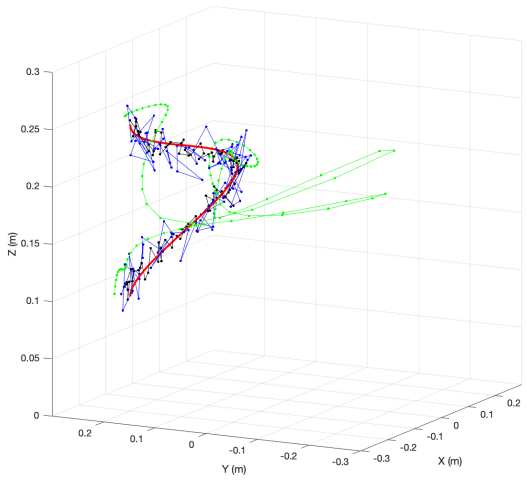

Figure 8: This figure shows the path following of Path 2 of the different models of inverse kinematics. 100 points along the path were considered and calculated via the three models.

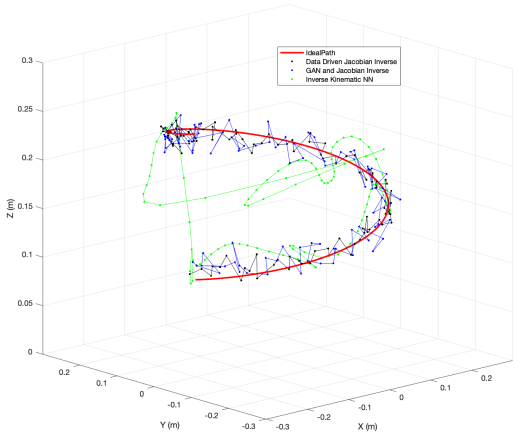

Figure 9: This figure shows the path following of Path 1 by the different models of inverse kinematics. 100 points along the path were considered and calculated via the three models.