Journal of Korea Robotics Society (2024) 19(3):266-273
https://doi.org/10.7746/jkros.2024.19.3.266 ISSN: 1975-6291 / eISSN: 2287-3961

관절 정보를 이용한 토크 추정 방식의 트랜스포머 기반 로봇 충돌 검출 방법

Transformer based Collision Detection Approach by Torque Estimation using Joint Information

박 지 원[1] · 임 대 규[2] · 박 수 민[2] · 박 현 준†
Jiwon Park[1], Daegyu Lim[2], Sumin Park[2], Hyeonjun Park†

Abstract: With the rising interaction between robots and humans, detecting collisions has become increasingly vital for ensuring safety. In this paper, we propose a novel approach for detecting collisions without using force torque sensors or tactile sensors, utilizing a Transformer-based neural network architecture. The proposed collision detection approach comprises a torque estimator network that predicts the joint torque in a free-motion state using Synchronous time-step encoding, and a collision discriminator network that predicts collisions by leveraging the difference between estimated and actual torques. The collision discriminator finally creates a binary tensor that predicts collisions frame by frame. In simulations, the proposed network exhibited enhanced collision detection performance relative to the other kinds of networks both in terms of prediction speed and accuracy. This underscores the benefits of using Transformer networks for collision detection tasks, where rapid decision-making is essential.

Keywords: Collision Detection, Transformer, Torque Controlled Robot

1. Introduction

As robots increasingly collaborate with humans in dynamic and unstructured environments, the importance of precise and rapid collision detection has been more critical[1]. The integration of robots into shared spaces with humans requires solutions that enables the detection of collisions, thereby facilitating the planning of subsequent actions to mitigate potential injuries.

Traditionally, collision detection strategies have relied on the use of external sensors[2,3]. In this conventional setup, tactile arrays and Force/Torque sensors are integral in providing the sensory feedback necessary for robots to interpret interactions with their environment. However, the implementation of such external sensors entails considerable challenges[4]. Implementing skin sensors of force sensors on robot surfaces or joints increases the complexity and cost, while also reducing the robot's operational flexibility. The vision sensor-based approach requires high computing power due to the large amount of data to be processed and has limitations in recognizing collisions occurring in blind spots[5]. For these reasons, research has been conducted to detect unexpected collisions using only internal sensor such as joint encoder, without relying on external sensor[6-12].

A classical method for detecting external collisions without using sensors involves observing changes in the angular velocities of each joint[6,7]. This principle detects collisions by modeling the dynamics of the robot manipulator and designing angular velocity observers. However, this approach has a limitation in that it requires accurate information about the model.

To overcome the limitations of these model-based methods,

Received : Mar. 29. 2024; Revised : May. 15. 2024; Accepted : May. 22. 2024
※ This research was supported by Deep-Tech Tips funded by Ministry of SMEs and Startups.
1. Undergraduate Student, Dept. of Electronic Engineering, Kyung Hee University, Yongin, Korea (overflow21@khu.ac.kr)
2. Principal Researcher, Robros Inc., Seoul, Korea (dglim, sumin @robros.co.kr)
† Chief Technology Officer, Corresponding author: Robros Inc., Seoul, Korea (hpark@robros.co.kr)

collision detection algorithms utilizing neural networks have recently been actively researched[8-12]. These methods estimate disturbances by using neural networks to learn uncertain dynamics models. They also detect collisions by estimating difficult-to-model friction or model errors through learning based on momentum observers. Unlike fully connected layer networks that estimate simple models, Recurrent Neural Network (RNN), which are suitable for handling time-series data, are more appropriate for observing the dynamics variations of robots. Specifically, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), by overcoming the gradient vanishing problem of traditional RNNs, are useful for processing long sequence data. However, LSTM and RNN suffer from the disadvantage of being difficult to parallelize due to their sequential input processing. Transformer, on the other hand, overcomes this limitation by processing sequences all at once, enabling parallel processing[13]. While Transformers are predominantly utilized in large language models, this paper presents the potential applicability of Transformers in the field of robotics control by applying them to collision detection algorithms for robots.

To detect external collisions of robots without sensors using Transformers, consideration of the robot dynamics system is prerequisite. Based on a torque input-based controller and the structure of the robot system, a network utilizing Transformers is constructed to predict control inputs using information from each joint. Subsequently, a collision discriminator is designed using Transformers to compare the control input torque with the torque predicted by the network in order to determine collisions.

The presented Transformer-based collision detection algorithm, in this paper, introduces two potential contributions. Firstly, it showcases the utilization of Transformers to estimate the dynamics of robots and validates its efficacy through simulation. Secondly, it directly estimates the control inputs of robots solely relying on internal sensor information. This approach, being independent of external sensors and information about observers, controllers, and robot models, underscores the algorithm's versatility across diverse scenarios.

This paper is structured as follows: In Chapter 2, torque estimator and collision discriminator using Transformers are introduced. The torque estimator is designed to directly estimate control inputs without any modeling, and the collision discriminator is designed to detect collisions using both recent torque data. Chapter 3 demonstrates the effectiveness of the proposed methods in a simulation environment, and Chapter 4 presents conclusions and discussions on the proposed methods and results.

2. Methodology

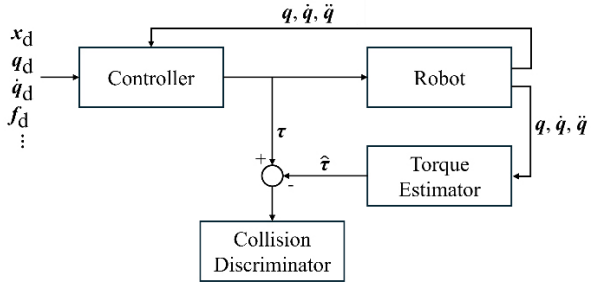
2.1 Problem formulation

The motion of a robot manipulator is elicited in response to the torque applied to each joint. When control input torque and externally applied torque are exerted on the manipulator, the dynamics of the robot are expressed by as

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}) = \boldsymbol{\tau}_{\mathrm{u}} + \boldsymbol{\tau}_{\mathrm{ext}}. \tag{1}$$

Where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}} \in \mathbb{R}^{n\times 1}$ denote the joint angle vector, angular velocity vector, and angular acceleration vector, respectively. $\mathbf{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times 1}$ represents the centrifugal and Coriolis forces, $\mathbf{g}(\boldsymbol{q}) \in \mathbb{R}^{n\times 1}$ denotes the gravity force, $\boldsymbol{\tau}_{\mathrm{u}}$ and $\boldsymbol{\tau}_{\mathrm{ext}} \in \mathbb{R}^{n\times 1}$ denotes the control input torque and external torque, respectively. The $\boldsymbol{\tau}_{\mathrm{ext}}$ encompasses disturbances such as friction within its scope, but in this paper, it is delimited to torque resulting from external impacts. Computed torque control (CTC) is a method for controlling robots by applying torque to each joint, considering dynamics, to enable the robot to achieve motion as desired. CTC estimates the dynamics of the robot system using dynamical models, thereby inversely calculating the angles and angular velocities of each joint to control or generate the desired force using Jacobian Transpose. In the situation of controlling the robot using CTC, if disturbances forces apply to the robot, its motion deviates from the intended motion represented by the control torque. In this paper, we present a collision detection approach using a Transformer network utilizing input torque and joint information in the absence of external forces. Subsequently, we identify disparities arising from the influence of external forces to ascertain collisions.

[Fig. 1] illustrates the structure of the proposed torque estimator network and collision discriminator network operating within the framework of a general robot control system. The controller receives joint angle information from the robot systems and computes torque input values for various robot control objectives, such as the desired position of the end-effector $\boldsymbol{x}_{\mathrm{d}} \in \mathbb{R}^{3\times 1}$, desired joint angle $\boldsymbol{q}_{\mathrm{d}}$, angular velocity $\dot{\boldsymbol{q}}_{\mathrm{d}}$, and desired force $\boldsymbol{f}_{\mathrm{d}} \in \mathbb{R}^{3\times 1}$, to be transmitted to the robot.



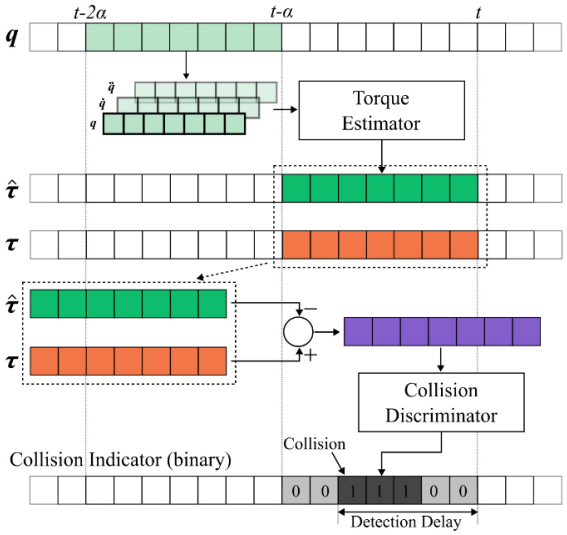
[Fig. 1] Overview of collision detection approach

The torque estimator shown in [Fig. 1] estimates the control input torque for the next step using received joint information from the robot. In this paper, to estimate the control input torque using only the joint information of the manipulator, we use a Transformer architecture based on the RNN to construct the network. The collision discriminator discerns disturbances of the robot by contrasting the control input torque values imposed by the controller onto the robot with the torque values estimated by the torque estimator. While the collision discriminator proposed in this paper solely conducts basic collision detection, for future research on external force estimation through external torque estimation, the collision discriminator is also constructed with a network architecture using a Transformer.

Ultimately, the proposed method consists of a torque estimation network and a collision discriminator network utilizing Transformer architecture. The torque estimator estimates the joint-based input torque $\hat{\boldsymbol{\tau}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$ in scenarios without the influence of external forces ($\boldsymbol{\tau}_{\text{ext}} = 0$). The collision discriminator detects situations where disturbances unexpectedly occur ($\boldsymbol{\tau}_{\text{ext}} \neq 0$) by comparing the estimated input torque with the actual control input torque.

2.2 System overview

[Fig. 2] illustrates the system overview of a data processing pipeline for detecting collisions using a robot joint information dataset. At each control step, joint angle data are collected and used to estimate torque for the torque estimator network. The torque estimator estimates the torque between $(t-\alpha)$ and t intervals by utilizing data from the $(t-2\alpha)$ to $(t-\alpha)$ interval. This estimated torque is then compared with the actual torque calculated from robot controller. The collision detection model performs inference at time step t, using a data window from $(t-\alpha)$ to t. The parameter α serves as a constant determining



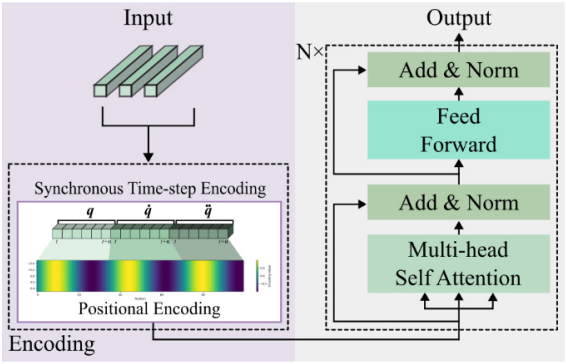
[Fig. 2] Timeline of the data pipeline for the collision discriminator

the amount of data utilized for collision detection.

The torque estimator network utilizes this window to estimate the torque values based on the joint angle, angular velocity, and angular acceleration. These estimated torque values are then subtracted from the actual torque values to determine discrepancies indicative of abnormal behavior, such as collisions. The collision discriminator module takes the discrepancy between the estimated and actual torques and determines whether a collision has occurred. The collision events are encoded as a binary collision indicator, with a positive indication signaling a collision at the respective time frame. The collision indicator is represented as a binary sequence for each control step, where 0 denotes collision-free instances and 1 indicates collision events. This system design enables real-time monitoring of robotic operations to identify collision events, enhancing the safety and reliability of automated processes.

2.3 Network architecture

In our proposed architecture, we use the Synchronous Time-step Encoding (STE) for positional encoding, tailored for handling time-series data involving joint angle, angular velocities, and angular accelerations as shown in [Fig. 3]. This technique addresses the challenge of preserving temporal coherence when these sequences are concatenated, ensuring accurate temporal interpretation of each data point. STE continuously discriminates neighboring time-steps by applying temporal identifiers from a single sinusoid, thereby maintaining the temporal relationship



[Fig. 3] Diagram of Synchronous Time-step Encoding with Transformer, tailored for time-series data of joint positions, angular velocities, and angular accelerations, ensuring temporal coherence

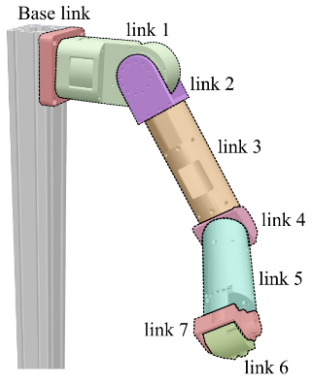
and sequence of movements, even when multiple features are combined into a single vector. This aspect is crucial for tasks like collision detection that demand an intricate understanding of sequential dynamics. To ensure temporal consistency, STE applies identical positional encoding at regular intervals within the concatenated sequences. This modification of the traditional sinusoidal embedding formula allows the model to understand that data points from $\boldsymbol{q}, \dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ at the same positions pertain to synchronous time steps. Such an embedding strategy enhances the model's capability to interpret and predict complex time-series patterns by leveraging the temporal coherence of sequential data, thereby improving its performance in tasks requiring precise analysis of dynamic movements.

3. Simulative Verification

To verify the feasibility of the collision detection approach proposed in this paper, simulations were conducted. The simulation environment was constructed using the MuJoCo[14] library, chosen for its capability to consider collision stability between the robot and its environment. The robot model utilized was a 7-degree-of-freedom manipulator developed by our research group as shown in [Fig. 4].

3.1 Computed torque controller

Since the proposed method outlined in Chapter 2 focuses on controllers managing the robot's torque, the controller designed for the data collection process, utilizing Jacobian transpose



[Fig. 4] Manipulator model utilized in simulation

control, is structured as

$$\boldsymbol{\tau}_{\mathrm{u}} = \boldsymbol{J}^{\mathrm{T}}\boldsymbol{f} + \boldsymbol{\tau}_{\mathrm{fric}} + \boldsymbol{\tau}_{\mathrm{g}}, \tag{2}$$

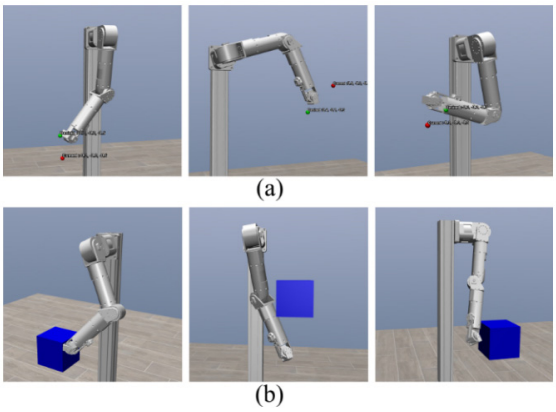
$$\boldsymbol{f} = \begin{bmatrix} \boldsymbol{K}_{\mathrm{p}}(\boldsymbol{x}_{\mathrm{d}} - \boldsymbol{x}_{\mathrm{c}}) - \boldsymbol{K}_{\mathrm{d}}\dot{\boldsymbol{x}}_{\mathrm{d}} \\ \boldsymbol{\delta\Phi} \end{bmatrix}, \ \boldsymbol{\tau}_{\mathrm{fric}} = \boldsymbol{K}_{\mathrm{c}} \tanh(k\dot{\boldsymbol{q}}). \tag{3}$$

Where $\boldsymbol{J} \in \mathbb{R}^{6\times n}$ represents the basic Jacobian matrix for an n-DOF manipulator, $\boldsymbol{\tau}_{\mathrm{g}}$ denotes gravity compensation torque vector. $\boldsymbol{K}_{\mathrm{p}} \in \mathbb{R}^{3\times 3}$ and $\boldsymbol{K}_{\mathrm{d}} \in \mathbb{R}^{3\times 3}$ respectively denote the virtual spring gain matrix and damping matrix of the end-effector position in Cartesian space. $\boldsymbol{x}_{\mathrm{d}}$, $\boldsymbol{x}_{\mathrm{c}} \in \mathbb{R}^{3\times 1}$, and $\boldsymbol{\delta\Phi} \in \mathbb{R}^{3\times 1}$ represent the desired position, current position, and orientation error of end-effector, respectively. The joint friction compensation torque vector $\boldsymbol{\tau}_{\mathrm{fric}}$ serves to compensate for joint gear friction by adding a constant torque of predetermined slope k and magnitude $\boldsymbol{K}_{\mathrm{c}} \in \mathbb{R}^{n\times n}$ for each joint rotation direction. Since the proposed collision detection method is independent of the controller's structure, intentional complexity is introduced by designing the controller to deliberately incorporate trigonometric functions and derivatives such as the hyperbolic tangent function and Jacobian.

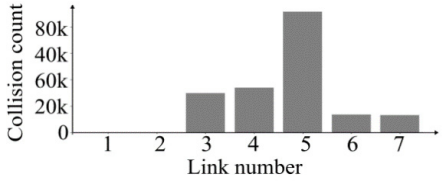
3.2 Data collection

Simulation was divided into two scenarios involving free motion and collision situations, with details and the operational range for both scenarios depicted in [Fig. 5]. For free motion, we selected a workspace within the robot's operational range where the robot commonly operates. Desired points were randomly designated within this workspace, and the robot was program-

med to repetitively move to these points. In collision scenarios, the robot moved within specific areas, but rectangular obstacles were randomly placed to induce collisions. These obstacles were essential in creating collision events, offering crucial opportunities for the model to detect and learn from the variations in torque patterns resulting from unforeseen physical contacts. Time-series data for each joint, including joint positions and computed torques, were recorded at a frequency of 500 *Hz*. For collisions, occurrences were logged in a binary format, frame by frame at the same frequency. Angular velocities were derived by differentiating joint positions, and angular accelerations were obtained by further differentiation. All collected data were min-max normalized before being inputted into the model. The dataset for these experiments consisted of a total length of 11,585,405 data points per each joint. The size of recent data α as presented in [Fig. 2] was set to 1,000. [Fig. 6] shows the distribution of collision occurrences per link during the data collection process. For the following reasons, biased distributions appeared: Links 1 and 2, which rotate only and are closer to the base, experienced minimal collisions, while Link 5, characterized by its longer length, exhibited relatively higher collision rates. Links 6 and 7 exhibited relatively fewer occurrences of collisions due to their smaller size and volume compared to links 3, 4, and 5. [Table 1] lists the accuracy of collision detection based on the collision link. It confirms that biased collision data does not affect the collision detection algorithm.



[Fig. 5] Snapshots of the data collection process in the simulation environment: (a) Free random motion scenario, (b) Collision scenario caused by a random box



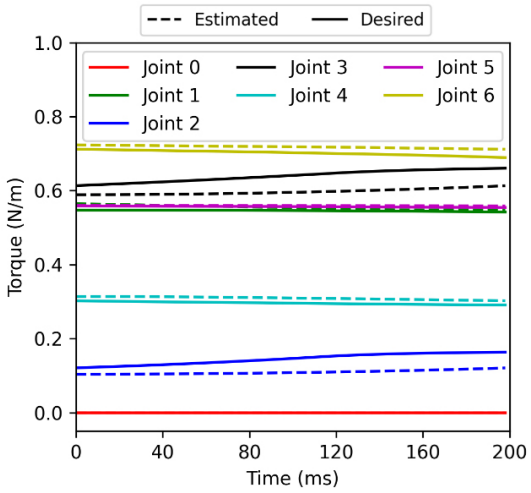
[Fig. 6] Distribution of collisions about each link

3.3 Performance of torque estimator network

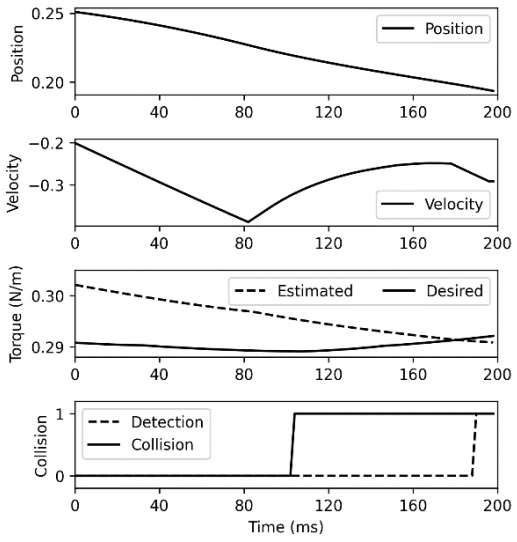
[Fig. 7] depicts the actual input torque and the torque estimated by the network for each joint in free motion scenarios.

[Table 1] Accuracy of collision detection by link

Link	1	2	3	4	5	6	7
Accuracy	-	-	86.4	96.5	92.3	89	95.1



[Fig. 7] Graphs of estimated and control torque for each joint in free motion



[Fig. 8] Graphs of joint angle, angular velocity, both torque, and collision detection during collision scenarios

[Fig. 8] illustrates the joint information, specifically for the 5th link, during collision situations. Due to the collision, there are changes in the trends of the desired torque and angular velocity. The discrepancy between the torque estimated by the network and the trend seems to serve as the basis for collision detection judgments.

To evaluate the performance of network architectures for the collision detection problem, we first examined the performance of the torque estimator network. KL divergence is used to measure how closely the probability distribution of the predictions made by our model matches the actual distribution observed in the training data. After training for 100 epochs, the estimated torque values on the validation set were compared to the target torque values using KL-divergence loss as

$$L_{KL} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i}\boldsymbol{\tau}_{n,i}\cdot(\log\boldsymbol{\tau}_{n,i}-\log\hat{\boldsymbol{\tau}}_{n,i}) . \quad (4)$$

Where N represents the batch size. i denotes the class index.

[Table 2] lists the results using RNN architecture, while [Table 3] lists the outcomes with the Transformer architecture. Despite having a generally higher number of network parameters, the Transformer architecture offers shorter inference times than RNNs.

This efficiency stems primarily from parallel processing capabilities, which are made possible by its self-attention mechanism that allows for the simultaneous processing of all input data elements. In contrast, RNNs process data sequentially, necessitating the completion of one element's before proceeding to the next. In the context of online collision detection, where rapid detection and response to collisions are crucial, the shorted inference time of the Transformer represents the advantage. This analysis underscores the Transformer's efficacy in tasks requiring quick and efficient processing, marking it as a superior choice for real-time collision detection applications.

[Table 2] Performance of RNN-based torque estimator

Batch size	Network information			Inference time (ms)	L_{KL} (1.0E-7)
	Layers	Hidden size	Parameters		
64	20	50	49907	54.48	40.1
128	20	50	49907	55.31	4.972
128	20	100	100907	107.23	2.25
128	40	100	800807	474.15	3.991

[Table 3] Performance of Transformer-based torque estimator

Network information				Inference time (ms)	L_{KL} (1.0E-7)
Heads	Encoder layers	Hidden size	Parameters		
4	6	256	977863	7.65	4.762
4	6	512	21489159	7.67	3.585
8	6	512	21489159	9.09	3.16
8	12	512	43890293	12.39	4.093

3.4 Performance of collision discriminator network

For the collision discriminator network, which predicts a binary tensor indicating the presence of a collision on a frame-by-frame basis, we conducted the following experiments to compare the performance across different architectures. Collisions are generated randomly in the workspace and the collision information is recorded for each joint. To determine whether the Transformer architecture is advantageous for the problem of collision detection, comparisons were made among a Multi-layer Perceptron (MLP), an RNN network, and a Transformer network. All three networks were trained for 100 epochs with a batch size of 16. The MLP consisted of three fully connected layers, the RNN had a hidden size of 100, and the Transformer used a hidden size of 100, with 8 heads and 6 encoder layers. Considering the potential imbalance in the class distribution between collision and no-collision instances, the F1 score is selected for its ability to balance both precision and recall of the model,

[Table 4] Performance of collision discriminator according to network architectures

	Inference Time (ms)	Accuracy	Detection Delay (ms)	F1 score
Multi Layer Perceptron	0.16	0.563	-	0.512
RNN	59.83	0.967	300	0.750
Transformer	10.96	0.999	98	0.812

[Table 5] Ablation study on torque estimator

Ablation settings		L_{KL} of Torque Estimator (1.0E-7)
STE	Decoder	
O	O	0.992
X	O	1.043
O	X	3.585
X	X	4.629

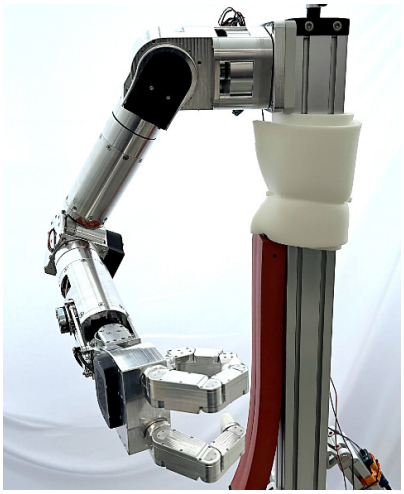
offering a more nuanced evaluation of its performance in effectively handling both positive and negative classes. The proportion of collision in the total data is 21%. As listed in [Table 4], the Transformer demonstrates higher accuracy compared to other architectures and, when compared with the similarly performing RNNs, it exhibits faster inference time.

3.5 Ablation studies

To enhance the learning performance of the Transformer network, STE and decoder are naturally utilized; however, an ablation study was conducted to assess their respective impacts. Simulations were carried out to examine the effects of applying STE and attaching a decoder to the Transformer-based torque estimator network. This study aimed to explore two main aspects. The first aspect was the application of STE to enhance the network's capability in handling time-series data with inherent temporal relationships. The second aspect involved attaching a decoder to the network with the hypothesis that it would improve the network's ability to reconstruct and predict torque values more accurately. The findings from these experiments are systematically presented in [Table 5]. The results demonstrate that both the application of STE and the attachment of a decoder to the Transformer-based torque estimator network contributed to its improved performance.

4. Conclusion

In this study, we present a Transformer-based approach for detecting collisions in a robot manipulator. This approach identifies external collisions of a robot using only the joint values measured by internal encoder sensors, without the need for external sensors like force-torque sensors. This system comprises a torque estimator network, which estimates torque in a free motion state using joint positions and a computed torque, and a collision discriminator, which recognizes collisions based on the difference between estimated and actual torque. The effectiveness of this approach has been demonstrated through simulations. For both networks, the performance of the Transformer-based network was superior, particularly in terms of inference time, demonstrating the Transformer's advantage. This suggests that the Transformer is well-suited for the task of the collision discriminator, where quick decision-making is crucial. The collision detection method proposed in this paper was designed without considering the controller and motion planner, aiming to induce generality in the control system. However, the proposed approach is deemed to have limitations in its validity as it has been verified through simulation for a single control system. In the future, we aim to improve the collision detector by verifying it with various controllers, dymanic motion, and diverse motion planners using an actual robot manipulator in [Fig. 9]. Furthermore, we aim to enhance the algorithm not only for simple collision detection but also for estimating the direction of force or inferring collision links.



[Fig. 9] Robros manipulator for future works

References

[1] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, "Working together: A review on safe human-robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26754-26773, Nov., 2017, DOI: 10.1109/ACCESS.2017.2773127.

[2] J. Pan, I. A. Şucan, S. Chitta, and D. Manocha, "Real-time collision detection and distance computation on point cloud sensor data," *IEEE Int. Conf. Robotics and Automation*, Karlsruhe, Germany, pp. 3593-3599, 2013, DOI: 10.1109/ICRA.2013.6631081.

[3] W. Li, Y. Han, J. Wu, and Z. Xiong, "Collision detection of robots based on a force/torque sensor at the bedplate," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 5, pp. 2565-2573, Oct., 2020, DOI: 10.1109/TMECH.2020.2995904.

[4] X. Wenzhong, X. Xi, and J. Xinjian, "Sensorless robot collision detection based on optimized velocity deviation," *2017 Chinese Automation Congress (CAC)*, Jinan, China, pp. 6200-6204, 2017, DOI: 10.1109/CAC.2017.8243894.

[5] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," *2012 IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, pp. 338-345, 2012, DOI: 10.1109/ICRA.2012.6225245.

[6] C.-N. Cho, S.-D. Lee, and J.-B. Song, "Collision Detection Algorithm based on Velocity Error," *Journal of Korea Robotics Society*, vol. 9, no. 2, pp. 111-116, May, 2014, DOI: 10.7746/jkros.2014.9.2.111.

[7] B.-J. Jung, T.-K. Kim, G. Won, D. S. Kim, and J. Hwang, "Development of Joint Controller and Collision Detection Methods for Series Elastic Manipulator of Relief Robot," *Journal of Korea Robotics Society*, vol. 13, no. 3, pp. 157-163, Aug., 2018, DOI: 10.7746/jkros.2018.13.3.157.

[8] A. D. Libera, E. Tosello, G. Pillonetto, S. Ghidoni, and R. Carli, "Proprioceptive Robot Collision Detection through Gaussian Process Regression," *2019 American Control Conference (ACC)*, Philadelphia, PA, USA, pp. 19-24, 2019, DOI: 10.23919/ACC.2019.8814361.

[9] D. Lim, M.-J. Kim, J. Cha, D. Kim, and J. Park, "Proprioceptive External Torque Learning for Floating Base Robot and its Applications to Humanoid Locomotion," *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8510-8517, 2023, DOI: 10.1109/IROS55552.2023.10342530.

[10] D. Lim, D. Kim, and J. Park, "Momentum observer-based collision detection using lstm for model uncertainty learning," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, pp. 4516-4522, 2021, DOI: 10.1109/ICRA48506.2021.9561667.

[11] D. Kim, D. Lim, and J. Park, "Transferable Collision Detection Learning for Collaborative Manipulator Using Versatile Modularized Neural Network," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2426-2445, Aug., 2021, DOI: 10.1109/TRO.2021.3129630.

[12] K. M. Park, Y. Park, S. Yoon, and F. C. Park, "Collision detection for robot manipulators using unsupervised anomaly detection algorithms," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 5, pp. 2841-2851, Oct., 2022, DOI: 10.1109/TMECH.2021.3119057.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv:1706.03762*, 2017, DOI: 10.48550/arXiv.1706.03762.

[14] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, pp. 5026-5033, 2012, DOI: 10.1109/IROS.2012.6386109.

박 지 원

2019~현재 경희대학교 전자공학과 학부생

2023.12~2024.02 (주)로브로스 방문연구원

관심분야: AI & Robotics, 3D Vision

임 대 규

2015 서울대학교 기계항공공학부(공학사)

2024 서울대학교 융합과학기술대학원 지능형융합시스템전공(공학박사)

2024~현재 (주)로브로스 책임연구원

관심분야: 휴머노이드, 모방학습, 강화학습, 딥러닝

박 수 민

2010 서울과학기술대학교 기계설계자동화공학부(공학사)

2012 서울대학교 융합과학기술대학원 지능형융합시스템학과(공학석사)

2020 서울대학교 융합과학기술대학원 지능형융합시스템전공(공학박사)

2020~2021 한국생산기술연구원 박사후연구원

2021~현재 (주)로브로스 책임연구원

관심분야: 대규모언어모델, 휴머노이드 로봇 보행, 생체역학

박 현 준

2011 광운대학교 제어공학과(공학사)

2014 서울대학교 융합과학기술대학원 지능형융합시스템학과(공학석사)

2020 서울대학교 융합과학기술대학원 지능형융합시스템전공(공학박사)

2020~2021 한국로봇융합연구원 선임연구원

2021~현재 (주)로브로스 이사

관심분야: 휴머노이드, 컴플라이언스 제어, 로봇 핸드, 물체 조작