Robust Motion Mapping Between Human and Humanoids Using CycleAutoencoder

Matthew Stanley, Lingfeng Tao, and Xiaoli Zhang

***Abstract*— Teleoperation needs accurate and robust motion mapping between human and humanoid motion to generate intuitive robot control with human-like motion. Data-driven methods are often deployed as it can result in intuitive, real time motion mapping. When using these methods, the common focus is on the accuracy of the motion mapping model. However, effort needs to be put into making the mapping model robust in face of noisy or incomplete dataset. In other words, the model needs to learn the generalizable mapping rules, not just be accurate in predicting the training data. To create a robust and accurate model for motion mapping, we developed the novel CycleAutoencoder method. This method simultaneously trains two autoencoders using traditional losses, mixed losses, and cycle losses. These losses allow the autoencoders to reconstruct the motion mutually between humans and humanoids. This allows the method to learn the mapping with improved accuracy and robustness compared to training a traditional autoencoder. The results of human subject involved experiments demonstrated that the CycleAutoencoder method can achieve both accuracy and robustness for the mapping compared with other autoencoder-based mapping methods.**

I. INTRODUCTION

Teleoperation allows an operator to control the robot without needing to be physically present in the task environment. This allows an operator to safely perform a task in dangerous environments, across large distances, and over different scales. Motion mapping techniques have been utilized for creating teleoperation models as it allows the operator to use their body to intuitively control the humanoid robot. Properly developing motion mapping models is essential to generating human-like and intricate robot movements [1], [2].

Many researchers have been interested in using data-driven methods for mapping human motion to robots instead of developing empirical models. Rather than laboriously creating mapping rules or generating inverse kinematics models for every human tracking system and robot structure, data-driven methods learn how to perform the motion mapping through collected training samples [1], [3]. These models can overcome the high computation associated with conventional inverse kinematics-based motion mapping equations. Data-driven models can also learn the mapping without needing the operator to convert the human and humanoid's coordinate systems into a common coordinate system.

The challenge of creating a data-driven mapping model is to achieve high accuracy and robustness while maintaining efficient training [4]–[9]. A model that is not accurate is unable to precisely predict the information provided. Robustness is defined as how sensitive a model is to perturbances in the training data [8]–[10]. This includes the ability of the model to understand the mapping even in locations where the training data is lacking. In other words, the model should be able to map motion rather than overfitting to the training data. Training efficiency is defined as how long it takes the model to converge to a valid mapping solution from the training data; ideally it should not take long for the model to converge. These objectives will often conflict when training a model, meaning that improving one of the criteria will decrease another. To improve these three criteria, a new model frame work will need to be designed.

This is the purpose for the design of the CycleAutoencoder model. We designed this model to use two autoencoders that can map information between each other. Autoencders are used since they are designed to reduce the dimensionality of mapping problems [11]. Reducing the dimensionality of the motion should result in key features arising that represents the motion that can be converted into human motion and robot motion. The dimension reduction should produce robust mapping quicker. By training these two autoencoders simultaneously, they should generalize the mapping rules between human motion and robot motion. This will produce more robust mapping compared to other models that prioritize one mapping direction. To train both autoencoders while achieving the desired accuracy and robustness, the CycleAutoencoder introduces three pairs of loss function structure to ensure accurate representation of the human and robot motion, to ensure accurate mapping between the two, and to improve robustness between the mapping.

In pursuit of a more accurate and robust data-driven method for teleoperation, we contributed the following:

1) Creating the CycleAutoencoder method to perform accurate and robust mapping teleoperation between a human and a humanoid robot.
2) Comparing the CycleAutoencoder method with other training methods for motion mapping autoencoders with human subject to robot experiments. The criteria used to compare these methods are:
    a) Accuracy: How well the model can predict the mapping.
    b) Robustness: How well the model can predict outside of the trained workspace.
    c) Efficiency: how long it takes to train the models for each method.

Matthew Stanley, Lingfeng Tao, and Xiaoli Zhang are with the Intelligent Robotics and Systems Lab, Colorado School of Mines, Golden, CO 80401, USA

II. Related Works

Data-driven methods have been widely accepted to perform motion mapping tasks. Many data-driven techniques have been developed to improve the performance for teleoperation mapping. Aleotti [12] proposes a system that uses feed-forward neural networks to generate a robot's arm joint angles from the detected motions with multiple demonstrations for the model training. In addition, Stanton [1] implements a system that can transfer the motions to a humanoid robot by using feed-forward neural networks to calculate all joint angles. Kim [13] used recurrent neural networks (RNN) to extract the features from the motions of a human and corresponding robot motions.

An issue with using data-driven methods for motion mapping is guaranteeing accurate, robust mapping. The quality of the data, data collection scenario, and learning policies affect how confident one can be in data-driven predictions [14]. This negatively impacts the accuracy and robustness of all data-driven methods, no matter how they are trained [9]. Another limitation of data-driven methods stems from discrepancy in robot structure from a human skeletal structure [4] and difference in the size of their manipulable workspace [2], [15]. The differences make creating an exact mapping dataset difficult, further diminishing the quality of the data. This furthers the importance of creating a data-driven model that is robust as well as accurate.

Autoencoders is promising to achieve the desired accuracy and robustness in the data-driven mapping method. The main benefit of an autoencoder is that it performs dimension reduction [11]. This allows the model simplify the complicated motions into a lower dimensional manifolds [9]. This is ideal for motion mapping, as mapping the common manifolds is simpler than mapping the full structure. Even autoencoders are subject to accuracy and robustness issues presented in the previous paragraph. While data quality is one feature that affects the accuracy and robustness, the autoencoder's structure and training method will also influence these criteria. In this paper, differences in training methods on the accuracy and robustness are investigated.

III. Methods

A. CycleAutoencoder

To increase the accuracy and reduce the computation complexities, a novel mapping technique called CycleAutoencoder is developed for motion mapping. This technique is inspired by the CycleGAN method, with the main difference being the use of regression losses instead of classification losses. Much like CycleGAN, the CycleAutoencoder is a method of training two autoencoders with cyclical loss functions. CycleGAN originated for recreating art in a different style [16]. Recently, this method has begun to appear in the field of robotics for associating gestures to expression internal and effective states [17] and creating reward images for human to robot reinforcement learning [18]. Since classification problems use logistic regression which is orthogonal to standard regression [19], it is possible to modify CycleGAN to perform regression mapping.

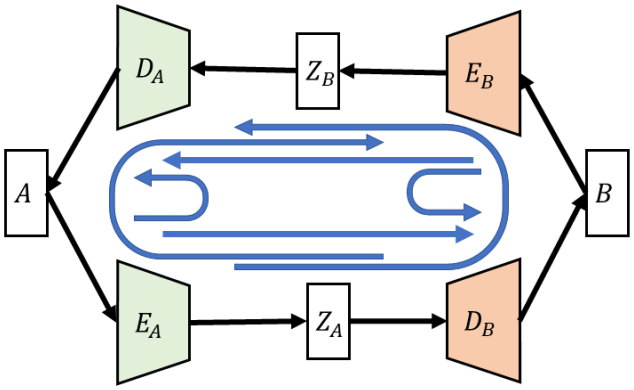


Fig. 1. The CycleAutoencoder method. This method trains two autoencoders with the same size latent features using six loss functions. The blue lines show the path of each loss functions. The inner two curves are traditional autoencoder losses to understand each agent. The middle straight lines are the mixed losses to understand the mapping between agents. The outer two curves are the cycle losses to enforce the mapping can be reversed.

Traditionally, the CycleGAN model uses two mappings separate autoencoders to map between two domains and two classifiers to identify if an image is in one of the specific domains. This involves having the model generate images for both outputs and using another model to classify them. If they cannot be accurately classified, the CycleGAN model updates its weights to improve the image generation. This structure works for image generation, since the images are the same size and contain similar subjects in different domains; however, it will not work for robot teleoperation. Robot design is very diverse, with ranging degrees of freedom, joint lengths, joint count, and control parameters. While the CycleGAN structure has shown to be effective at motion classification, it will not work for teleoperation. Therefore, the loss function for the CycleAutoencoder needs to be redesigned in order to work for robot control.

The CycleAutoencoder method combines encoders and decoders from two autoencoders to perform mapping between agents A and B. Both autoencoders are designed such that the latent features (Z_A and Z_B) have the same shape, which results in the motion from one agent to be encoded by its autoencoder then decoded by the other autoencoder. Instead of using discriminators, the CycleAutoencoder uses regression loss functions. Figure 1 demonstrates the six loss functions used to train this method. The loss functions belong to three categories. The first two losses are the traditional autoencoder losses, where each agent is encoded then decoded back to itself. For example, human motion should be encoded by the human encoder then decoded back by the human decoder to human motion. This ensures that each autoencoder knows the information it is trying to encode.

$$L_{AA} = ||A - D_A(E_A(A))||_1 \tag{1}$$

$$L_{BB} = ||B - D_B(E_B(B))||_1 \tag{2}$$

The second type of loss functions is the mixed losses. This loss function encodes an agent then decodes it into the

form of the other agent. For example, the human motion is mapped to the robot where it is compared to the expected robot motion. These loss functions ensure that information can be converted between agents.

$$L_{AB} = ||B - D_B(E_A(A))||_1 \quad (3)$$

$$L_{BA} = ||A - D_A(E_B(B))||_1 \quad (4)$$

The final type of loss functions is the cycle losses. For this type of loss, each agent is encoded, decoded into the form of the other agent, encoded again, and finally decoded back into its original form. For example, the human motion is encoded and decoded into robotic motion, then encoded and decoded back to human motion. These last loss functions are used to verify that information can be reconstructed correctly after being transferred between agents.

$$L_{ABA} = ||A - D_A(E_B(D_B(E_A(A))))||_1 \quad (5)$$

$$L_{BAB} = ||B - D_B(E_A(D_A(E_B(B))))||_1 \quad (6)$$

By enforcing all six loss functions, the model learns how to correctly map motion information between the two agents. These losses will balance trying to learn learning the features to map back to the original agent and the features to map to the other agent. The first loss guarantees accurate mapping from an agent back to itself while the second one ensures decoders will work with both encoders. Finally, the last lost ensures that motion mapped to the other agent will contain the same information when mapped back to the original agent. To evaluate its performance, the CycleAutoencoder model is compared against two other autoencoder training methods: a direct feedforward autoencoder method and using Recommendation via Dual-Autoencoder (ReDa) [20], [21].

B. Baseline Method 1 - Direct Autoencoder

The direct autoencoder method uses a single, feedforward autoencoder to encode the motion of agent A and decode it to the motion of agent B. Unlike the CycleAutoencoder, training a direct autoencoder model only produces one mapping; however, more separate direct autoencoder models can be created to create the other mappings such as mapping agent B to agent A. Only one loss function, shown in Figure 2 is used to train the direct autoencoder to ensure that the output of the model matches expectation.

$$L_{Direct} = ||B - D_B(E_A(A))||_1 \quad (7)$$

C. Baseline Method 2 - Recommendation via Dual-Autoencoder

The other mapping technique used for creating the motion mapping is the ReDa [20], [21]. Like the CycleGAN method, this method simultaneously trains two autoencoders to perform the mapping between agents A and B where both latent features have the same shape. The method achieves the motion mapping with three loss functions, shown in Figure 3. The first two losses are standard autoencoder losses to ensure the motion can be reconstructed.

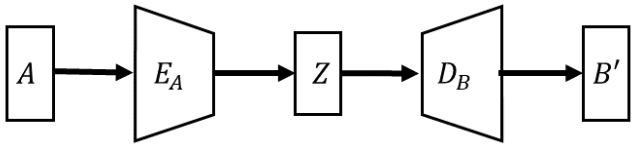


Fig. 2. The direct autoencoder method trains a single feedforward autoencoder to map agent A to agent B using a single loss function. The loss function the predicted motion of B against the expected motion of B for the input motion from A.

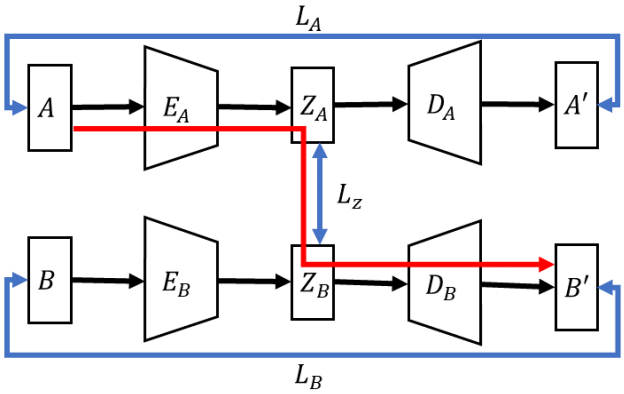


Fig. 3. The ReDa method trains two autoencoders with the same size latent features using three loss functions. The blue lines show the path of each loss functions. The L_A and L_B lines are traditional autoencoder losses to understand each agent. The L_Z line is the latent loss to enforce each autoencoder to generate the same latent features. The red line shows how the input can be transferred since the latent features should be the same.

$$L_A = ||A - D_A(E_A(A))||_1 \quad (8)$$

$$L_B = ||B - D_B(E_B(B))||_1 \quad (9)$$

The third loss function is to force the resulting latent features of both models to be equal. This loss is used to balance the importance of each autoencoder's latent features. This results in both autoencoders using the same latent features to encode the motion mapping information.

$$L_Z = ||E_A(A) - E_B(B)||_1 \quad (10)$$

IV. Experiments

A. Data Collection

To demonstrate the capability of these models for teleoperation, we collected motion data between a human and the humanoid robot, Pepper. Rather than using full body tracking, we collected data for the human and Pepper moving their arms. This was done to reduce the input size of the models while maintaining a complex relationship to demonstrate the capability of our CycleAutoencoder. The dataset is comprised of Pepper joint rotations and the skeletal structure of the human. Figure 4 demonstrates the data collection process used to generate synchronized motion pairs. Pepper is programmed with a predefined path to move its arms while the human is instructed to mimick Pepper. This ensures that

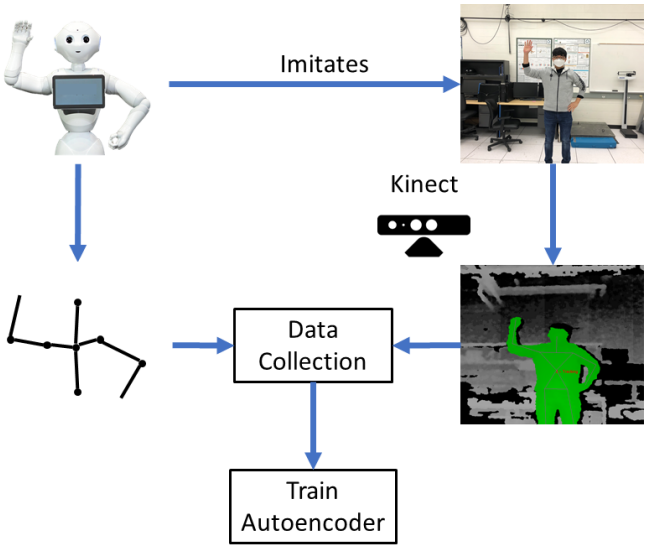
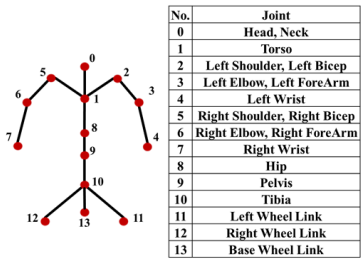
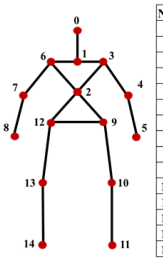


Fig. 4. Illustration of the data collection process. Pepper is programmed with an action and the joint angles are recorded. A human then imitates Pepper while a Kinect tracks the human's joint position. This results in paired data used to train the three autoencoder methods.



No.	Joint
0	Head, Neck
1	Torso
2	Left Shoulder, Left Bicep
3	Left Elbow, Left ForeArm
4	Left Wrist
5	Right Shoulder, Right Bicep
6	Right Elbow, Right ForeArm
7	Right Wrist
8	Hip
9	Pelvis
10	Tibia
11	Left Wheel Link
12	Right Wheel Link
13	Base Wheel Link



No.	Joint
0	Head
1	Neck
2	Torso
3	Left Shoulder
4	Left Elbow
5	Left Hand
6	Right Shoulder
7	Right Elbow
8	Right Hand
9	Left Hip
10	Left Knee
11	Left Foot
12	Right Hip
13	Right Knee
14	Right Foot

Fig. 5. The Pepper (a) and human (b) skeletal structures. The models are trained to match arm positions between the human (joints 3-8) and Pepper (joints 2-7) since Pepper's lower body has different manipulability than a human. Each of Pepper's joints has one degree of freedom. A human is more complicated with three degrees of freedom in the shoulders and two degrees of freedom in the elbow and hand.

the models will have a one-to-one mapping between the human arms and robot arms.

Pepper data consisted of 10 joints (numbers 2 through 7 in Figure 5a) and 10 degrees of freedom. The joint angles are recorded as a unit quaternion. The human data consists of 6 joints (numbers 3 through 8 in Figure 5b) and 14 degrees of freedom. The human joints are represented as a location relative to the previous joint in meters. The previous joint for the shoulder and the origin of the human skeleton is the torso. The skeleton data is captured using the ROS skeleton_markers package by Patrick Goebei with a reported accuracy for the Kinect being about 10 cm [22]. In total, 788 sample pairs were generated to evaluate the models.

Since Pepper's joint positions are normalized quaternions, the human's joint position needs to be normalized as well. The reason is to both reduce any error based on differences in the human's distance from the Kinect and to equalize the importance of the human joints and Pepper joints during training. Since the joints are represented using different coordinates units, the loss magnitudes will be different for each agent. To normalize human joint information, the unit vector in the direction from the torso to each joint is calculated. The unit vector is used since only the angle of each joint changes, the length is a human arm is constant.

B. Model Construction

To demonstrate the effectiveness of the CycleAutoencoder model, we compare it's ability to map human to robot arm motion against a direct feedforward autoencoder and the ReDa method. To ensure fair training for the three model types, each model is constructed such that they will have a similar number of nodes to train. The CycleAutoencoder and ReDa methods both train two autoencoders, so the structure of their autoencoders are the exact same. However, the direct autoencoder methods only trains one feedforward autoencoder. Therefore, the direct autoencoder uses twice as many layers as the other two methods to produce a similar node count.

The CycleAutoencoder and ReDa models used four layers for encoding and four layers for decoding.The encoding layers for the human joint information goes from a node count of 18 to 16, 14, 12, then finally 10 latent features. The encoding layers for Pepper's joint information goes from a node count of 40 to 32, 24, 16, then finally to 10 latent features. The encoders for the direct autoencoder model still reduces the latent features to 10, but adds a new layer half way between the layers in the CycleAutoencoder and ReDa models. The decoders for each methods uses the same node counts as the encoders but in reverse order. The leaky rectified linear activation function with an alpha value of 0.01 is used between each layer of all the autoencoder methods, while the output layer uses a linear activation function.

C. Evalutation Metrics

When comparing three mapping methods, there are three criteria of interest. The first is the general accuracy of the model. To determine the accuracy, the dataset is split randomly between training and testing across the entire workspace. After the model is trained, the RMSE is calculated over every joint in the predictions of the testing set. After testing the methods 40 times, the mean and standard deviation is calculated for the RMSE accuracy.

The second criteria is the model's robustness. To perform this test, the dataset is split based on the distance the human arm travels from the torso joint. By using points closer to the torso for training, the RMSE can be calculated over every joint in the untrained region furthest from the human's torso. After testing the methods 40 times, the mean and standard deviation is calculated for the RMSE robustness.

The final criteria is to determine the efficiency of training each model. As the models are being trained, we use a validation dataset to compare loss function evaluations with that from the training data. For this, the root mean squared loss was plotted during the training of the models at each epoch. The speed of convergence for the model learning

TABLE I

THE RMSE ACCURACY OF EACH MODEL ACROSS DIFFERENT SIZES OF TRAINING AND TEST DATASETS. HUMAN TO PEPPER MAPPINGS ARE ERRORS IN UNIT QUATERNIONS WHILE PEPPER TO HUMAN MAPPINGS ARE ERRORS IN UNIT VECTORS. FOR A GIVEN TESTING SIZE, THE CYCLEAUTOENCODER HAS THE SMALLEST MEAN ERROR AND SMALLEST DEVIATION IN ERROR.

	CycleAutoencoder				Direct				ReDa			
	Human to Pepper		Pepper to Human		Human to Pepper		Pepper to Human		Human to Pepper		Pepper to Human	
% Test	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
10	0.0632	0.0089	0.0769	0.0061	0.0642	0.0106	/	/	0.1304	0.0232	0.1195	0.0223
15	0.0677	0.0077	0.0771	0.0046	0.0743	0.0128	/	/	0.1319	0.0191	0.1209	0.0209
20	0.0680	0.0074	0.0773	0.0047	0.0700	0.0113	/	/	0.1330	0.0152	0.1273	0.0228
25	0.0709	0.0085	0.0801	0.0040	0.0714	0.0107	/	/	0.1283	0.0162	0.1281	0.0235
30	0.0719	0.0079	0.0795	0.0038	0.0773	0.0130	/	/	0.1291	0.0129	0.1252	0.0242

TABLE II

THE RMSE ROBUSTNESS OF EACH MODEL ACROSS DIFFERENT SIZES OF TRAINING AND TEST DATASETS. HUMAN TO PEPPER MAPPINGS ARE ERRORS IN UNIT QUATERNIONS WHILE PEPPER TO HUMAN MAPPINGS ARE ERRORS IN UNIT VECTORS. FOR A GIVEN TESTING SIZE, THE CYCLEAUTOENCODER HAS THE SMALLEST MEAN ERROR AND SMALLEST DEVIATION IN ERROR.

	CycleAutoencoder				Direct				ReDa			
	Human to Pepper		Pepper to Human		Human to Pepper		Pepper to Human		Human to Pepper		Pepper to Human	
% Test	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
10	0.2094	0.0168	0.1810	0.0188	0.2175	0.0201	/	/	0.2383	0.0191	0.2297	0.0383
15	0.2069	0.0136	0.1722	0.0210	0.2190	0.0158	/	/	0.2322	0.0227	0.2236	0.0413
20	0.2071	0.0125	0.1771	0.0190	0.2169	0.0146	/	/	0.2493	0.0300	0.2360	0.0426
25	0.2094	0.0116	0.1756	0.0203	0.2212	0.0160	/	/	0.2394	0.0226	0.2380	0.0425
30	0.2083	0.0136	0.1772	0.0255	0.2161	0.0155	/	/	0.2338	0.0234	0.2220	0.0370

the random training set and the point of divergence of the validation samples were noted.

V. RESULTS

Table I shows the accuracy measurement of each model over different sized training data. Mapping from human to Pepper motion, the CycleAutoencoder has the lowest RMSE accuracy across all the trials and the Reda method had an error that is an order of magnitude greater than other two methods. As the testing set increase and training set decrease, the accuracy slowly decreases for the CycleAutoencoder and direct autoencoder methods. The CycleAutoencoder shows little variance as the most accurate model. Even when mapping from Pepper to human motion, the CycleAutoencoder method is consistently the most accurate model.

Table II shows the robustness measurement of each model over different sized training data. Again, the CycleAutoencoder method is the most robust method for mapping human to Pepper motion and Pepper to human motion. Unlike the accuracy results, there is not an apparent pattern with the RMSE robustness as a function of percent training size. This is due to the cycle loss which ensures information reconstruction across different manipulable regions. The 5% improvement clearly demonstrates the the cycle loss improves the robustness.

Between the results in Tables I and II, the CycleAutoencoder model is shown to be the most accurate and robust model. Over the 40 samples, the CycleAutoencoder also shows the lowest variance between the tests. Despite the improvement to accuracy and robustness, the CycleAutoencoder method does not have a good training efficiency. As seen in Figure 6, all the models were trained without the validation deviating from the training loss. The direct autoencoder and ReDa training methods learned the mapping process quickly and converged at a solution around epoch 25. The CycleAutoencoder model took more than 75 epochs to converge to a similar loss value as the other two models.

VI. DISCUSSION AND CONCLUSION

These results can mostly be explained by examining the loss functions of the three training methods. The direct autoencoder only has one loss function but is only concerned with the forwards mapping from human to robot mapping. It is the simplest model focused on a single task. The simplicity makes the model accurate and robust. The CycleAutoencoder method needs to train two autoencoders but it uses six loss functions to train the model. The mixed losses functions are used to achieve the same accuracy as the direct autoencoder, while the standard autoencoder and cycle losses further improve the accuracy and robustness. The training takes

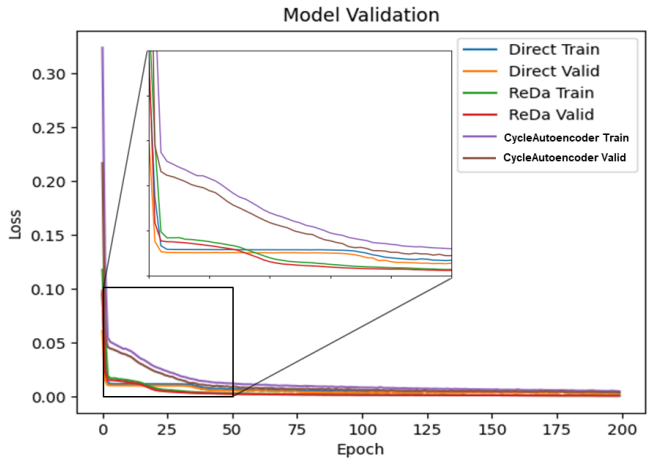


Fig. 6. The loss validation curve for training each model. None of the validation losses deviate much from the training losses, meaning they are being properly trained. The CycleAutoencoder method produces a large loss at the start due to having six loss functions. This also results in the method taking the longest time to converge. By the end of the training period, all three methods converge to a similar loss value.

longer as the model needs to fulfil the requirements of all the loss functions; however, this results in a more accurate and robust mapping between human and robot. The ReDa method can quickly determine the mapping as its three loss functions are unique. However, since the latent feature loss only ensure that the model is robust instead of making the ReDa's mapping more accurate.

The CycleAutoencoder method presented in this paper is proposed as a better alternative to traditional data-driven, motion mapping methods. In two of the three evaluation criteria, the CycleAutoencoder method was superior to the baseline methods. The CycleAutoencoder method was more accurate and robust, but at the cost taking more time to train. The method is efficient at converting human motion in relative joint positions to Pepper motion in unit quaternions. These results indicate that the CycleAutoencoder method is good for teleoperation to generate human-like motion.

ACKNOWLEDGMENT

This material is based on work supported by the US NSF under grant 1652454. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

REFERENCES

[1] C. Stanton, A. Bogdanovych, and E. Ratanasena, "Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning," in *Proc. Australasian Conference on Robotics and Automation*, vol. 8, 2012, p. 51.

[2] D. Matsui, T. Minato, K. F. MacDorman, and H. Ishiguro, "Generating natural motion in an android by mapping human motion," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3301–3308.

[3] R. M. Pierce and K. J. Kuchenbecker, "A data-driven method for determining natural human-robot motion mappings in teleoperation," in *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, 2012, pp. 169–176.

[4] J. Koenemann, F. Burget, and M. Bennewitz, "Real-time imitation of human whole-body motions by humanoids," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2806–2812.

[5] C. Ott, D. Lee, and Y. Nakamura, "Motion capture based human motion recognition and imitation by direct marker control," in *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 399–405.

[6] S.-Y. Chiang, S.-C. Kuo, J.-B. Lin, and C.-H. Chen, "Dynamic imitation of human motion for humanoid robot," in *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2017, pp. 1–4.

[7] X. Li, H. Cheng, G. Ji, and J. Chen, "Learning complex assembly skills from kinect based human robot interaction," in *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2017, pp. 2646–2651.

[8] J. M. Hernández-Lobato, "Balancing flexibility and robustness in machine learning: semi-parametric methods and sparse linear models," Ph.D. dissertation, Autonomous University of Madrid, 2010.

[9] D. Gopinath, G. Katz, C. S. Păsăreanu, and C. Barrett, "Deepsafe: A data-driven approach for assessing robustness of neural networks," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2018, pp. 3–19.

[10] M. A. Goodrich, J. W. Crandall, and E. Barakova, "Teleoperation and beyond for assistive humanoid robots," *Reviews of Human factors and ergonomics*, vol. 9, no. 1, pp. 175–226, 2013.

[11] Y. Li, Z. Wang, X. Yang, M. Wang, S. I. Poiana, E. Chaudhry, and J. Zhang, "Efficient convolutional hierarchical autoencoder for human motion prediction," *The Visual Computer*, vol. 35, no. 6-8, pp. 1143–1156, 2019.

[12] J. Aleotti, A. Skoglund, and T. Duckett, "Position teaching of a robot arm by demonstration with a wearable input device," in *International Conference on Intelligent Manipulation and Grasping (IMG04)*, 2004, pp. 459–464.

[13] M. Kim, S. Kim, and J. Park, "Human motion imitation for humanoid by recurrent neural network," in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2016, pp. 519–520.

[14] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, *et al.*, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[15] Y. Lin, H. Zhao, Y. Zhang, and H. Ding, "A data-driven motion mapping method for space teleoperation of kinematically dissimilar master/slave robots," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2018, pp. 862–867.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[17] M. Suguitan, M. Bretan, and G. Hoffman, "Affective robot movement generation using cyclegans," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 534–535.

[18] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 157–11 166.

[19] D. G. Denison, C. C. Holmes, B. K. Mallick, and A. F. Smith, *Bayesian methods for nonlinear classification and regression*. John Wiley & Sons, 2002, vol. 386.

[20] F. Zhuang, Z. Zhang, M. Qian, C. Shi, X. Xie, and Q. He, "Representation learning via dual-autoencoder for recommendation," *Neural Networks*, vol. 90, pp. 83–89, 2017.

[21] B. Dong, Y. Zhu, L. Li, and X. Wu, "Hybrid collaborative recommendation via dual-autoencoder," *IEEE Access*, vol. 8, pp. 46 030–46 040, 2020.

[22] Q. Wang, G. Kurillo, F. Ofli, and R. Bajcsy, "Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect," in *2015 international conference on healthcare informatics*. IEEE, 2015, pp. 380–389.