International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Consumption and Realization of Conditional Convolutional Variational Autoencoder for Robot Trajectory Learning

Midhun Muraleedharan Sylaja*[1], Ann Varghese[2], James Kurian[3]

Submitted: 29/01/2024 **Revised**: 08/03/2024 **Accepted**: 15/03/2024

Abstract: Learning-based motion planning methods have recently shown notable advantages in solving multiple planning challenges in high-dimensional spaces and challenging situations. The complex higher-dimensional trajectory with many constraints makes the robot task generation complicated. Furthermore, the availability of a sizeable robot open dataset for path learning is a significant challenge. In this work, a variant of Conditional Variational Autoencoder with Convolutional Neural network is utilised to capture each task's hidden probability distribution and generated back from the latent representation. The proposed approaches focuses on a generative model for offline generation of trajectory continuous task under static structured environment. The implemented model interfaced with the Robot Operating System (ROS) layer, which can directly feed into any ROS enabled robots. Reconstructed error, precision and accuracy were evaluated by the experiment of robot trajectory with reconstructed trajectory and yielded encouraging results.

Keywords: *Variational Autoencoder, Robotics, Deep Learning, Machine Learning, Trajectory Continuous Task*

1. Introduction

Robot trajectory planning is a fundamental aspect of robotics, involving the generation of precise paths and orientations that a robot follows during its motion. Humans face difficulty identifying the raw robot trajectory points as the robot moves in a very complex path and it is a challenging task to visualise the robot's movement. Various advanced algorithms, including Deep Learning [1], [2], Genetic algorithms [3], Model Predictive Control [4], Rapidly exploring Random Trees [5], Swarm optimisation [6] etc., were utilised to make collision-free smooth motion for various applications.

A robot task is a specific operation or action that a robot is programmed to perform to achieve a predefined objective. Robot tasks can be categorized into sequencing tasks and continuous tasks. The principal objective of sequencing tasks revolves around minimizing time and effort by determining the optimal sequence of actions [7], [8], [9]. In contrast, continuous tasks encompass both path continuous tasks and trajectory continuous tasks, where the robot's end effector must follow a seamless and uninterrupted motion within the task space [10], [11], [12]. Quang-Nam and Quang-Cuong discuss about the trajectory optimisation task in the context of 3D printing [5]. This entails the execution of a time-parameterized path while concurrently engaging in the printing process.

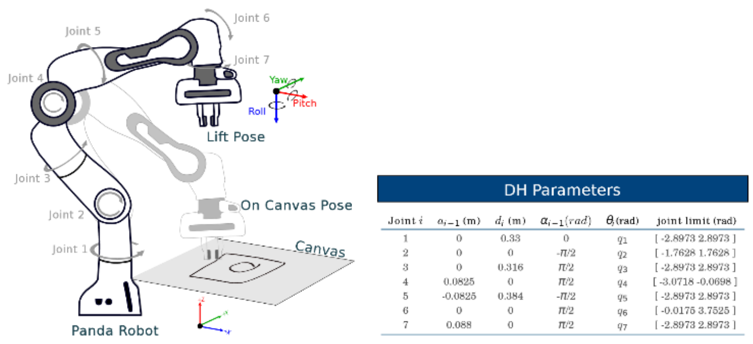
[1] *Department of Electronics, Cochin University of Science and Technology, Kalamassery, India*
ORCID ID : 0000-0003-3666-7520
[2] *Department of Electronics, Cochin University of Science and Technology, Kalamassery, India*
[3] *Department of Electronics, Cochin University of Science and Technology, Kalamassery, India*
* *Corresponding Author Email: midhunms@cusat.ac.in*

Recent advancements in the field of Generative modelling yielded mind-blowing outputs [13], [14], [15]. David Ha and Douglas Eck presented a sequence-to-sequence network for sketch generation [16]. Chao *et al.* put forth Chinese character drawing robots using recurrent neural networks, which is limited to the too-small number of strokes [17]. Shao *et al.* enhances the character generation using images and generate sequence by sequence, which is also limited to a small number of strokes and the network uses recurrent deep learning layers which causes the training process much slower [18].

Neural network layers encapsulate the hidden complex structure of the data distribution. Autoencoder (AE) is an unsupervised neural network architecture used for non-linear data-specific dimensionality reduction, where output data has same shape as input [19]. Variational Autoencoder (VAE) is a hybrid architecture of deep learning and variational inference which generates new data similar to the input. VAE is a stable and flexible generative model suitable for art generation [20]. A conditional variational autoencoder for robot path individually generates various paths.

Drawing a picture on canvas with pen lift between strokes is a subset of robot tasks, is taken for generating the dataset. The model can instruct to generate different robot sketches by giving task as draw an apple. The learned probability representation produces realistic generative sketches. Each path contains a higher-dimensional time-parameterized complex sequence of points with many constraints. Hard-coding of each set will be time consuming and hard to analyse and modify. Complex trajectory generation and modification are formidably tricky even for an expert. Conditional trajectory continuous tasks generation in robotics is a challenging task achieved by the proposed deep neural network architecture.

dataset. Quick-draw [16] is the google stroke dataset containing strokes for different images generated by people



DH Parameters

Joint i	a_{i-1} (m)	d_i (m)	$\alpha_{i-1}(rad)$	θ_i(rad)	joint limit (rad)
1	0	0.33	0	q_1	[-2.8973 2.8973]
2	0	0	-π/2	q_2	[-1.7628 1.7628]
3	0	0.316	π/2	q_3	[-2.8973 2.8973]
4	0.0825	0	π/2	q_4	[-3.0718 -0.0698]
5	-0.0825	0.384	-π/2	q_5	[-2.8973 2.8973]
6	0	0	π/2	q_6	[-0.0175 3.7525]
7	0.088	0	π/2	q_7	[-2.8973 2.8973]

Fig. 1. Panda robot drawing on canvas (left) with DH parameters and joint limits (right).

The work focus on inducing the prior knowledge into the model as data for reducing the exploration space and to make an offline generative model for a task structured static environment. In this paper, a Variational Autoencoder-based generative model is proposed, which can effectively generate trajectories and suppress noise. The model uses one-dimensional convolutional neural networks (CNN) for feature extraction and generates the paths' latent representation in a low-dimensional space [21]. The main contributions in this paper are summarized as

- A novel problem of conditional generation of continuous trajectory task.

- A method to generate robot datasets from stroke based datasets.

- A method to evaluate the performance of models based on ISO 9283 standard.

2. System Implementation

Robots are designed to use in many different tasks with various workspace conditions. The robot data sequence for each joint is recorded for a fixed length ($N_{max} = 512$ and feed as input data, represented by $x \in \mathbb{R}^D$ Where $\mathbb{R}$ is the set of real numbers, and D is the dimensionality. The label provided as one-hot encoded and the latent space vector z with dimensionality k represented as $z \in \mathbb{R}^k$.

In this work, a simulated version of serial manipulator Panda robot [22], developed by Franka Emika, is employed. Fig. 1 shows the Panda robot with the canvas and the two poses (pen-up and pen-down). The Denavit-Hartenberg parameters (DH parameter) [23], which describe the geometry and kinematics of a robot's manipulator are shown in Fig. 1 along with the joint limits.

2.1. Dataset

Developing a robot path dataset with better two-dimensional visualization is carried out by selecting the Quick-draw in different parts of the world. Fig. 2 contains the steps involved in generating the dataset. The procedure involves extracting the raw stroke values, as depicted in Fig. 2a, from the Quickdraw dataset. These values are then subjected to scaling to generate waypoints for the robot's path. During the execution of this path, the end-effector follows a trajectory that intersects with the canvas whenever a stroke from the dataset is encountered. Subsequently, it is raised to traverse to the next stroke in the sequence, a process that continues until the completion of the drawing's final stroke. This approach ensures that the robot's movement aligns with the strokes in the dataset, replicating the desired visual representation. Open Motion Planning Library (OMPL)[24] - an open-source probabilistic library for robot motion planners- is used for generating cartesian path in Robot Operating System(ROS) (Fig. 2d)[25], [26], [27], [28]. The robot trajectory contains a sequence of seven-dimensional vector stored into a Hierarchical Data Format version 5 (HDF5) dataset (Fig. 2e) to deal with a much larger dataset than memory and faster processing while training [29]. For fed into the model, the data is normalised by using each joints upper and lower limits, as shown in Fig. 1. Finally an inverse transform is applied for predicted/reconstructed data from the model.

The proposed work chooses eight different datasets as illustrated in Table. 1 from the quickdraw dataset. Each contains a unique id, strokes and dataset name. All datasets contain 70k+ samples. Lots of outliers are present in datasets since the data is generated from people from different parts of the world.

Robot data contains different joint variables in a higher-dimensional space. This work defined each task as a sketch drawing path on paper. For better visualization, trajectory is visualized by taking forward kinematics, then plotting the x and y on a canvas and adding the z component as the stroke colour.

Table 1. Task dataset description with a sample from each instance is plotted in the task space. Total instances from each dataset is unified to avoid biasing to class imbalance

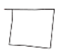
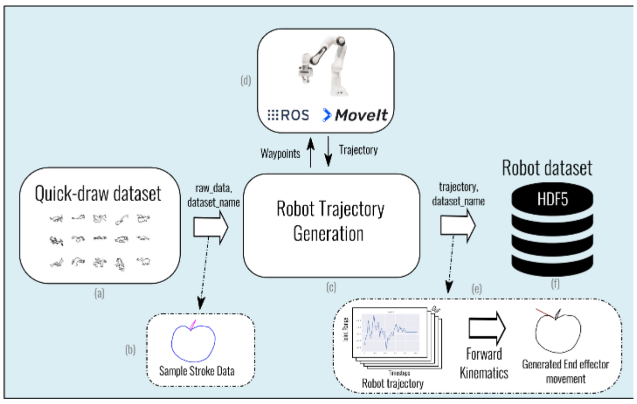
Task	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Example								
Total instances	50000	50000	50000	50000	50000	50000	50000	50000

Fig. 2. Dataset generation from Quickdraw dataset, a)



Raw data from quickdraw dataset is fed into Robot trajectory generation module. b) The Sample stroke data is visualised with each pen stroke with a different colour. c) ROS python module interfaced generates waypoints for cartesian path and pass it to Moveit module using ROS message services. d) Compute trajectory inside Moveit module is done using RRT Planner available in Open Motion Planning Library. Computed trajectory is converted into same scale using the joint limits of robot. e) Visualisation of generated trajectory in joint plots and 2D visualisation is done using forward kinematics. f) Save the generated trajectory into dataset for deep Learning.

2.2. Autoencoder Architecture

VAE learns the underlying probability distribution of paths in latent dimension, and Conditional Variational Autoencoder[30], [31], [32] is a VAE with a conditional generation of the path with x as inputs, y as labels, z as latent representation. $p_\theta(x|z,y)$ represents the prior, $p_\theta(z|x,y)$ represents the posterior distributions with normally distributed latent representation $p_\theta(z) = \mathcal{N}(z; 0, I_d)$ where $\mathcal{N}$ represents the normal distribution, d is the dimensionality, and I_d represents the d dimensional identity matrix.

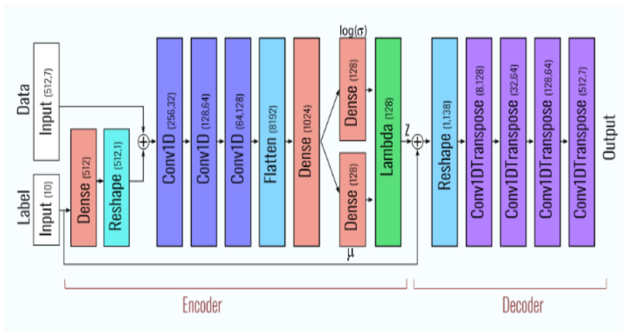


Fig. 3. Network architecture of Conditional Convolutional Variational Autoencoder model with latent dimension 128. Encoder network inputs Data and Label and produce a latent vector. Decoder reconstruct data using latent vector and label.

The proposed network architecture is portrayed in Fig. 3. CNN is a specialized layer used to capture the spatial and temporal reliance inside the data. One-dimensional CNN is used to better capture the features in the sequential robot data. Data and label concatenated and fed into a Fully connected strided convolution layer [33] and a non-linear activation function rectified linear unit (ReLU) is applied. The re-parametrisation trick $z = \mu + \sigma \odot \epsilon$ (where $\epsilon \in \mathcal{N}(0,1)$) for back-propagating in VAE is implemented using a non-parametric Lambda layer and fully connected dense layers for mu and sigma with linear activation. The decoder section consists of a series of strided convolution transpose layers with ReLU activation, and the output layer uses the sigmoid activation function. The autoencoder model is optimised with an RMSprop optimiser plus minibatch gradient descent.

Given attribute input path $x \in \mathbb{R}^{N_{max} \times DoF}$ abels $y \in \mathbb{R}^{labels}$ and latent variable $z \in \mathbb{R}^{latent\ dimension}$ forms a generator model $p_\theta(x|y,z)$ that generates realistic robot trajectories x conditions on y and z. The process is done in two steps, initially sample the latent dimension z randomly from the prior distribution and then compute the posterior distribution of generated path by conditioned on z and label.

Loss Function: A key challenge when computing the precision and accuracy is that even though the reconstruction loss is low, computing precision and accuracy may be higher in some cases. It is because all the joints are not considered equally in the training. End joints have a more negligible effect in end-effector position than the joints near the base link. Kullback-Leibler(KL) Divergence is a measure of how two probability distribution is different from one another [34]. Variational autoencoder model is trained using KL divergence loss L_{KL} and Mean Absolute Error (MAE) employed as the reconstruction loss L_R. as shown in equation. 1. KL divergence will measure the distribution mean and variance difference between normal distribution as shown in eq. 2. The model training is done by finding the parameter θ that minimises the mean absolute error.

$$L_R = \frac{1}{N} \sum \left| x_{predicted} - x_{actual} \right| \quad (1)$$

$$L_{KL} = f_{KL}(q_\phi(z|x,y)||p_\theta(z|x,y)) \quad (2)$$

$$L_{total} = L_R + L_{KL} \quad (3)$$

where f_{KL} denotes KL function and N is the total number of data pairs used. A set of four models with different latent dimensions as 16, 32, 64 and 128 are introduced and represented as Model 16, Model 32, Model 64 and Model 128 respectively.

3. Qualitative Analysis

Three qualitative analysis experiments were conducted in this study. The first experiment evaluates the visual similarity between the command path (input path) with the reconstructed path; the second experiment visualizes the effect of noise in the input, and the third one plots interpolation poses between two robot paths.

3.1. Conditional generation of path

Conditional generation of the robot path is achieved by supplying command pose and one-hot encoded label to the model, as presented in Fig. 4a. Fig. 5 shows a single command path(left) and nine generated paths(right) which shows the repeatability of the model. Conditional generation of a single task (command path) and corresponding joint values are plotted in Fig. 7.

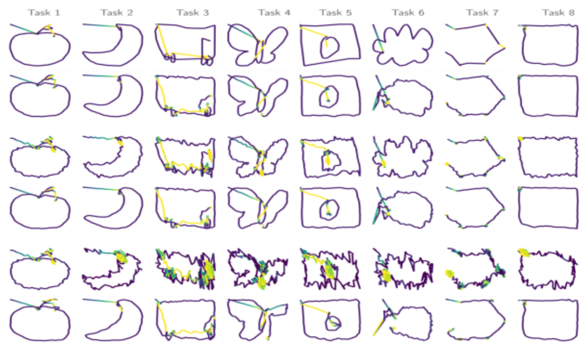


Fig. 4. Conditional reconstruction with command pose and the performance evaluation of the models a) test input from dataset b) test input with added gaussian noise(mean=0, variance=0.01) c) test input with added uniform noise(peak=0.01)

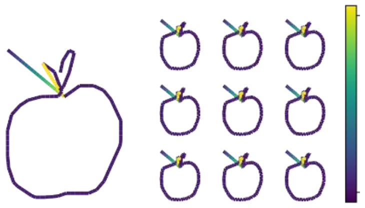


Fig. 5. Conditional generation of path from the dataset by supplying command pose and label. Input data (left) and generated nine samples (right) in the figure from generated Task 1 dataset.

3.2. De-noising of the path

De-noising of input is an added advantage of the proposed model. The noisy data input and the label encode into a lower-dimensional latent space point, and the decoder reconstructs it. Since the model is trained using data without any noise, any reconstruction from the latent dimension produces the sample from the training data's probability distribution. Performance evaluation is checked using two noise models - Uniform random noise and Gaussian noise. Gaussian noise is the normally distributed additive noise defined by variables noise mean and variance. Uniform random noise is an uncorrelated additive noise that follows the uniform distribution. Evaluated performance of the reconstruction by adding noise at the input is shown in Fig. 4b and 4c.

3.3. Latent Space Interpolation

Latent space is a compressed representation where similar data are placed closer to each other. Probabilistic latent space interpolation is performed using eq. 4, which morphs one sketch to another. Fig. 6 displays the image of a path in the dataset (Task 4) created by conditional generation.

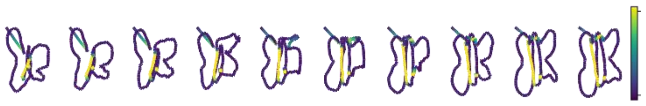
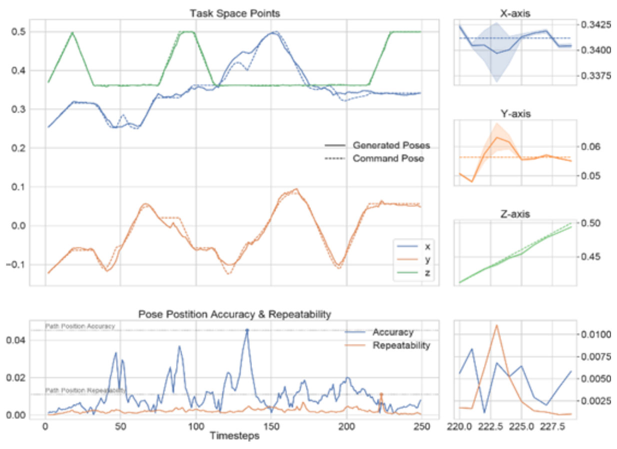


Fig. 6. Conditional one-dimensional linear interpolation of latent space for Task 4 dataset.

$$z_k = (z_j - z_i)\alpha + z_i \quad (4)$$

where $\alpha \in [0,1]$, $i, j \in N$, N is the set of natural numbers, z_i and z_j represent two different points on the latent space, and z_k represents the linear interpolated representation between z_i and z_j.



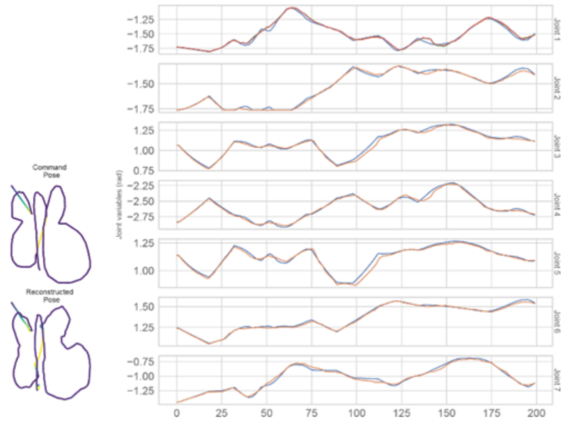


Fig. 7. (a) Image shows a portion of 200 time-steps of the commanded pose and reconstructed x, y pose (b) The image of the corresponding joint angles are shown in figure for all the joints (joints 1 to 7) commanded pose is represented in

4. Performance metrics

Performance metrics consist of accuracy and repeatability are the two prime quantitative analysis methods in robotics. Pose Position Accuracy, Pose Orientation Accuracy, Path Position accuracy and Path Orientation accuracy are the four metrics analysed in this work, as defined in ISO 9283[35].

4.1. Pose Accuracy

The test procedure starts from the start point in the robot workspace and moves till the end. Each point is visited in a unidirectional format. Each time the position of the robot end-effector is recorded, and then calculate the accuracy and precision. Accuracy is computed by generating thirty samples from each command pose of the test set.

Pose Position Accuracy Accuracy is the difference between the centroid of generated poses (called a barycenter) and the commanded position. The position accuracy is calculated using the standard equation.

$$AP_P = \sqrt{(\bar{x} - x_c)^2 + (\bar{y} - y_c)^2 + (\bar{z} - z_c)^2} \tag{5}$$

Where $\bar{x}, \bar{y}, \bar{z}$ are the barycenter of position x, y and z and represented as

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i, \quad \bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$$

and x_c, y_c, z_c represents command poses of x, y and z-axis and n represents the number of samples taken.

Pose Orientation Accuracy Fig. 1 shows the orientation frame of the end-effector, where the roll orientation is irrelevant while considering the drawing path. So the work considered only pitch and yaw axes for computing accuracy and repeatability. Pose orientation accuracy for pitch(β) and yaw(γ) is calculated as

$$AP_\beta = \overline{(\beta - \beta_c)} \tag{6}$$

$$AP_\gamma = \overline{(\gamma - \gamma_c)} \tag{7}$$

Where

$$\bar{\beta} = \frac{1}{n}\sum_{i=1}^{n} \beta_i, \quad \bar{\gamma} = \frac{1}{n}\sum_{i=1}^{n} \gamma_i,$$

Fig. 8. Pose Precision and Repeatability of Position of the robot end effector using Model 32

4.2. Pose Repeatability

Repeatability is a measure of the variation of the data around the calculated centroid. Repeatability RPL is calculated using

$$RP_L = \bar{I} + 3S_L \tag{8}$$

where $\bar{I}$ is the mean of the set and S_L is the standard deviation.

Pose Position Repeatability is calculated using eq. 8 where

$$\bar{I} = \frac{1}{N}\sum_{j=1}^{n} I_j, \qquad S_L = \sqrt{\frac{\sum_{j=1}^{n}(I_j - \bar{I})^2}{n-1}}$$

$$I_j = \sqrt{(x_j - \bar{x})^2 + (y_j - \bar{y})^2 + (z_j - \bar{z})^2}$$

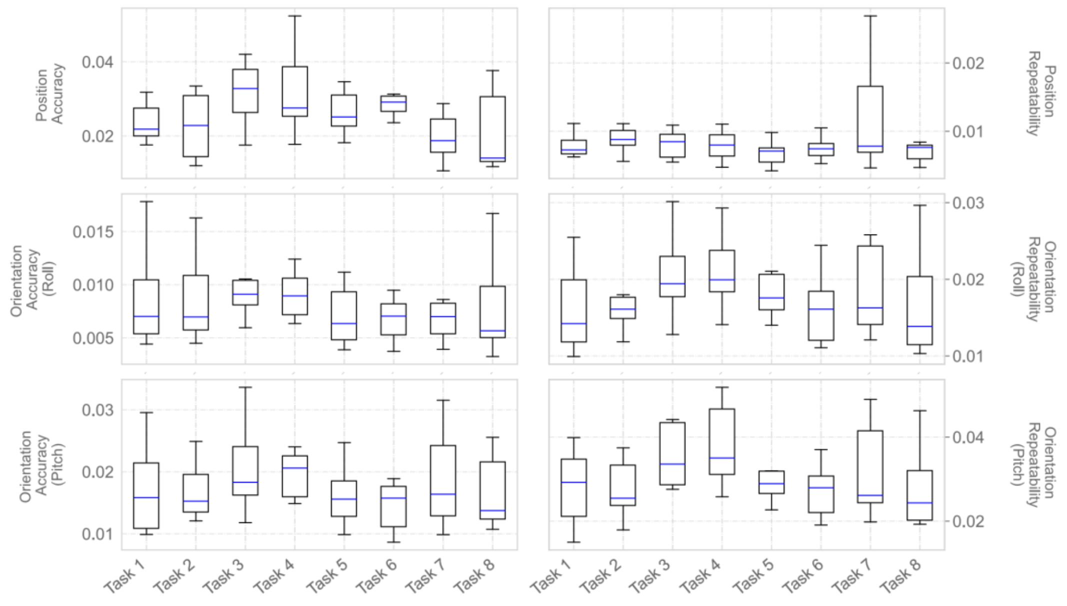


Fig. 9. Path Accuracy and Repeatability of Position and Roll and Pitch orientation of the robot end effector in Model 32

Pose Orienatation Repeatability is calculated for pitch(β) and yaw(γ) as

$$RP_\beta = \pm 3S_\beta = \pm 3\sqrt{\frac{\sum_{j=1}^{n}(\beta_j - \bar{\beta})^2}{n-1}}$$

$$RP_\gamma = \pm 3S_\gamma = \pm 3\sqrt{\frac{\sum_{j=1}^{n}(\gamma_j - \bar{\gamma})^2}{n-1}}$$

where S_α, S_β are standard deviations

Fig. 8(left) shows the pose position accuracy and repeatability of the robotic path in Fig. 7. The barycentre of x, y and z-axis are more visible in the highlighted version of path points shown in Fig. 8(right).

4.3. Path Accuracy and Repeatability

The ISO 9283 standard defines path accuracy as a metric for assessing the quality of robot end effector movement, where path position accuracy is determined by generating ten paths from the command path, calculating the barycenter, and estimating accuracy based on the maximum value of pose accuracy. Similarly, path repeatability is estimated as the maximum value of pose repeatability.

5. Evaluation and Applications

Autoencoder learns explicit probability distribution of the multi-dimensional gaussian normal latent distribution. First, demonstrate the repeatability of robotic path generation multiple times using the same command pose as instructed in ISO 9283 standards. Conditional generation of the path is achieved and is shown in Fig. 4 - 6. Command poses and reconstructed poses show a correlation. However, there are some minor differences in each generated sample since VAE acts like a generator model. Fig. 7 shows the joint values corresponding to a command-pose and generated poses. The correlation of plots shows the reconstruction quality. Since the Variational Autoencoder is based on probability distributed latent space, the model can predict intermediate data generation. Fig. 7 shows linear interpolation of generated sequences.

Table 2. Comparison of different latent space models Accuracy and Repeatability of different datasets (mean ± standard deviation, all values are in cm)

Task	Model 16	Model 32	Model 64	Model 128
		Accuracy		
Task 1	4.84 ± 3.22	3.14 ± 0.99	2.44 ± 0.70	2.68 ± 0.70
Task 2	3.15 ± 1.83	2.63 ± 0.62	2.15 ± 0.80	2.45 ± 0.55
Task 3	5.11 ± 1.69	4.44 ± 2.11	3.09 ± 0.79	3.49 ± 1.03
Task 4	6.38 ± 2.14	4.84 ± 1.73	3.50 ± 1.68	4.01 ± 1.63
Task 5	5.28 ± 2.89	3.87 ± 2.02	2.64 ± 0.57	3.87 ± 3.16
Task 6	4.24 ± 1.77	3.26 ± 0.67	2.85 ± 0.62	3.14 ± 0.91
Task 7	2.30 ± 1.45	2.21 ± 0.71	1.78 ± 0.53	2.13 ± 0.81
Task 8	2.61 ± 2.12	2.61 ± 1.89	2.08 ± 1.04	2.55 ± 1.54
		Repeatability		
Task 1	0.52 ± 0.15	0.80 ± 0.81	0.86 ± 0.34	1.12 ± 0.27
Task 2	0.48 ± 0.25	0.84 ± 0.34	0.89 ± 0.18	1.67 ± 0.84
Task 3	0.79 ± 0.35	0.83 ± 0.56	0.79 ± 0.19	1.21 ± 0.19

Task 4	0.80 ± 0.55	0.98 ± 0.41	1.98 ± 3.85	2.11 ± 2.53
Task 5	0.61 ± 0.27	0.75 ± 0.26	0.68 ± 0.16	3.32 ± 6.28
Task 6	0.59 ± 0.33	0.48 ± 0.17	0.75 ± 0.15	2.13 ± 3.18
Task 7	0.48 ± 0.27	0.63 ± 0.44	0.93 ± 0.49	1.30 ± 1.14
Task 8	0.42 ± 0.17	0.65 ± 0.47	0.75 ± 0.24	1.30 ± 0.57

Pose position accuracy and repeatability of a single path is shown in Fig. 8. It is pretty visible from the figure is that most of the points on the path achieve pose position accuracy much below 4cm and repeatability below 1cm. The barycenter of the data is too close to the commanded pose, showing better accuracy, and all generated images are close to each other, showing better repeatability.

Fig. 9 represents the path accuracy and repeatability of a single model (model 32) that runs on ten different commanded paths in each dataset and observes the results. Since the proposed dataset generation method uses data drawn by different people from various parts of the world, the dataset contains many outliers. While removing the outliers, the data shows that, for this particular path, all achieved position repeatability below 3cm and position accuracy below 5cm. Path Orientation accuracy is below 0.03 radians, and path orientation repeatability is below 0.05 radians.

Table. 2 shows the comparison of path accuracy between different models, among which Model 64 perform better. However, Model 16 and Model 32 perform better in path repeatability. While considering both path position accuracy and repeatability, Model 64 is selected, and if it requires more compressed latent dimension representation, Model 16 and 32 also be used. Even though as the latent space dimension increases in the Autoencoder model, the reconstruction loss generally decreases. However, the computed precision and accuracy may vary because they transform joint space data to task space points for computation. Training of each model is done using the early stopping method. The complexity of each dataset also varies. Each people will generate data differently depending on their imagination. Some common datasets like Task 1 perform well because the dataset is simple, and people from different parts of the world draw it based on some standard features. However, some datasets like Task 4 and Task 5 have lots of complex structures. There may not be a typical structure in all those drawings.

Autoencoder performs lossy compression and causes the reconstructed paths to drift in the output, reducing the path's accuracy and repeatability. Experts can use it to fine-tune the robotic paths by generating, interpolating and de-noising.

6. Conclusion

This work uses a 1D CNN-based conditional Variational Autoencoder for the robotic path generation task. All the models were trained using the generated dataset. Different qualitative plots were also used to evaluate the model performance along with quantitative measures like accuracy and repeatability. From path position accuracy, it has been found that all the datasets have a good accuracy value below 5cm and precision below 3cm, which is quite promising and confirms the reliability of the proposed model.

The current system uses generated dataset of robot tasks as there is no sizeable open dataset available. Actual robot task trajectory can be utilized for training the model in future. Based on the results obtained, the model will produce a better trained model using accurate data. Since the dataset and generated data follow similar distributions, the model can be employed for training any continuous robot trajectory under static structured environment with appropriate datasets.

Acknowledgements

The authors would like to acknowledge University Grants Commission (UGC), India under NFOBC scheme for providing financial assistance to carry out this research work and Cochin University of Science and Technology for the valuable support and infrastructure.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances in Robot Learning from Demonstration," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 3, no. 1, pp. 297–330, May 2020, doi: 10.1146/annurev-control-100819-063206.

[2] S. Cebollada, L. Payá, M. Flores, A. Peidró, and O. Reinoso, "A state-of-the-art review on mobile robotics tasks using artificial intelligence and visual data," *Expert Syst. Appl.*, vol. 167, p. 114195, Apr. 2021, doi: 10.1016/j.eswa.2020.114195.

[3] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger, "Positioning mobile manipulators to perform constrained linear trajectories," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 2578–2584. doi: 10.1109/IROS.2008.4650617.

[4] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constraint-based Model Predictive Control for holonomic mobile manipulators," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 1473–1479. doi: 10.1109/IROS.2015.7353562.

[5] Q.-N. Nguyen and Q.-C. Pham, “Planning Optimal Trajectories for Mobile Manipulators under End-effector Trajectory Continuity Constraint,” 2023, doi: 10.48550/ARXIV.2309.12251.

[6] R. Li, M. Liu, J. Teutsch, and D. Wollherr, “Constraint trajectory planning for redundant space robot,” *Neural Comput. Appl.*, vol. 35, no. 34, pp. 24243–24258, Dec. 2023, doi: 10.1007/s00521-023-08972-5.

[7] Q.-N. Nguyen, N. Adrian, and Q.-C. Pham, “Task-Space Clustering for Mobile Manipulator Task Sequencing,” 2023, doi: 10.48550/ARXIV.2305.17345.

[8] J. Xu, Y. Domae, T. Ueshiba, W. Wan, and K. Harada, “Planning a Minimum Sequence of Positions for Picking Parts From Multiple Trays Using a Mobile Manipulator,” *IEEE Access*, vol. 9, pp. 165526–165541, 2021, doi: 10.1109/ACCESS.2021.3135374.

[9] R. Malhan and S. K. Gupta, “Finding Optimal Sequence of Mobile Manipulator Placements for Automated Coverage Planning of Large Complex Parts,” in *Volume 2: 42nd Computers and Information in Engineering Conference (CIE)*, St. Louis, Missouri, USA: American Society of Mechanical Engineers, Aug. 2022, p. V002T02A006. doi: 10.1115/DETC2022-90105.

[10] L. Velazquez, G. Palardy, and C. Barbalata, “A robotic 3D printer for UV-curable thermosets: dimensionality prediction using a data-driven approach,” *Int. J. Comput. Integr. Manuf.*, pp. 1–18, Sep. 2023, doi: 10.1080/0951192X.2023.2257652.

[11] J. Sustarevas, D. Kanoulas, and S. Julier, “Task-Consistent Path Planning for Mobile 3D Printing,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 2143–2150. doi: 10.1109/IROS51168.2021.9635916.

[12] K. Nagatani, T. Hirayama, A. Gofuku, and Y. Tanaka, “Motion planning for mobile manipulator with keeping manipulability,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2002, pp. 1663–1668 vol.2. doi: 10.1109/IRDS.2002.1043994.

[13] H. Ha, S. Agrawal, and S. Song, “Fit2Form: 3D Generative Model for Robot Gripper Form Design,” 2020, doi: 10.48550/ARXIV.2011.06498.

[14] S. Kumra, S. Joshi, and F. Sahin, “Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network,” 2019, doi: 10.48550/ARXIV.1909.04810.

[15] J. J. Bird, A. Naser, and A. Lotfi, “Writer-independent signature verification; Evaluation of robotic and generative adversarial attacks,” *Inf. Sci. Int. J.*, vol. 633, no. C, pp. 170–181, Jul. 2023, doi: 10.1016/j.ins.2023.03.029.

[16] D. Ha and D. Eck, “A Neural Representation of Sketch Drawings,” 2017, doi: 10.48550/ARXIV.1704.03477.

[17] F. Chao *et al.*, “An LSTM Based Generative Adversarial Architecture for Robotic Calligraphy Learning System,” *Sustainability*, vol. 12, no. 21, p. 9092, Oct. 2020, doi: 10.3390/su12219092.

[18] Y. Shao and C.-L. Liu, “Teaching machines to write like humans using L-attributed grammar,” *Eng. Appl. Artif. Intell.*, vol. 90, p. 103489, Apr. 2020, doi: 10.1016/j.engappai.2020.103489.

[19] G. E. Hinton and R. R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006, doi: 10.1126/science.1127647.

[20] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” 2013, doi: 10.48550/ARXIV.1312.6114.

[21] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” 2015, doi: 10.48550/ARXIV.1511.08458.

[22] B. Zhang and P. Liu, “Control and benchmarking of a 7-DOF robotic arm using Gazebo and ROS,” *PeerJ Comput. Sci.*, vol. 7, p. e383, Mar. 2021, doi: 10.7717/peerj-cs.383.

[23] J. Denavit and R. S. Hartenberg, “A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices,” *J. Appl. Mech.*, vol. 22, no. 2, pp. 215–221, Jun. 1955, doi: 10.1115/1.4011045.

[24] I. A. Sucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012, doi: 10.1109/MRA.2012.2205651.

[25] S. Chitta, I. Sucan, and S. Cousins, “MoveIt! [ROS Topics],” *IEEE Robot. Autom. Mag.*, vol. 19, no. 1, pp. 18–19, Mar. 2012, doi: 10.1109/MRA.2011.2181749.

[26] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” in *ICRA workshop on open source software*, Kobe, Japan, 2009, p. 5.

[27] M. Cashmore *et al.*, “ROSPlan: Planning in the Robot Operating System,” *Proc. Int. Conf. Autom. Plan. Sched.*, vol. 25, pp. 333–341, Apr. 2015, doi: 10.1609/icaps.v25i1.13699.

[28] A. Koubaa, Ed., *Robot Operating System (ROS): The Complete Reference (Volume 6)*, vol. 962. in Studies in Computational Intelligence, vol. 962. Cham:

Springer International Publishing, 2021. doi: 10.1007/978-3-030-75472-3.

[29] A. Collette, *Python and HDF5*, First edition. Beijiing: O'Reilly, 2014.

[30] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2020, doi: 10.48550/ARXIV.2003.05991.

[31] D. E. Rumelhart and J. L. McClelland, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, MIT Press, 1987, pp. 318–362. Accessed: Feb. 19, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/6302929

[32] P. Baldi, "Autoencoders, unsupervised learning and deep architectures," in *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop - Volume 27*, in UTLW'11. Washington, USA: JMLR.org, Jul. 2011, pp. 37–50.

[33] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," 2015, doi: 10.48550/ARXIV.1511.06434.

[34] J. M. Joyce, "Kullback-Leibler Divergence," in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 720–722. doi: 10.1007/978-3-642-04898-2_327.

[35] ISO, "Manipulating industrial robots Performance criteria and related test methods." International Organization for Standardization, Geneva, CH, Apr. 1998. Accessed: Feb. 19, 2024. [Online]. Available: https://www.iso.org/standard/22244.html