2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636742

Identifying Valid Robot Configurations via a Deep Learning Approach

Tuan Tran and Chinwe Ekenna

***Abstract*— Many state-of-art robotics applications require fast and efficient motion planning algorithms. Existing motion planning methods become less effective as the dimensionality of the robot and its workspace increases, especially the computational cost of collision detection routines. In this work, we present a framework to address the cost of expensive primitive operations in sampling-based motion planning. This framework determines the validity of a sample robot configuration through a novel combination of a Contractive AutoEncoder (CAE), which captures an occupancy grids representation of the robot's workspace, and a Multilayer Perceptron (MLP), which efficiently predicts the collision state of the robot using the output from the CAE. We evaluate our framework on multiple planning problems with a variety of robots in 2D and 3D workspaces. The results show that (1) the framework is computationally efficient in all investigated problems, and (2) the framework generalizes well to new workspaces.**

I. Introduction

Recently, machine learning has been infused in many solutions to notoriously difficult problems in robotics. These strategies have seen a wide array of success in visual sensing [1], task learning [2], and human-robot cooperation [3], etc. In this work, we apply machine learning to predict the validity of robot configurations, a common procedure in sampling-based motion planners. Sampling-based approaches have seen broad applicability to many high-dimensional and complex motion planning problems in robotics [4], computer graphics [5], and computational biology [6]. Most of these approaches rely on the classification of robotic configurations into valid, e.g., collision-free, and invalid samples, while they work to construct graph approximations of a state space, called a roadmap. Thus, these approaches avoid full representation of the topology of the state space. However, as the dimensionality of the robot and the complexity of the workspace increase, the primitive operations of sampling-based methods become computationally obstructive [7].

Prior research has focused on improved geometric analysis techniques for collision checking [8], approximately modeling state space and obstacles to predict the collision status of robot configurations, and lazily invoking the configuration validation subroutine [4], among others. While these methods have shown success in many applications, none have fully and permanently bounded the computational cost of validating configurations in sampling-based motion planners.

We propose a novel framework to efficiently and effectively predict the validity of robot configurations in complex motion planning problems. The overview of our framework is shown in figure 1. A CAE embeds a representation of the workspace, corresponding to an occupancy grid, into a latent space. Then an MLP predicts the validity of a given input sample using the CAE encoding of the workspace. Thus, instead of using the information of the whole workspace to decide whether a sample's label is valid or invalid, only the limited information from the encoded latent space is used. We examine the potential impact of such an approach with a variety of robots in complex 2D and 3D workspaces. We focus on finding valid samples due to their importance in narrow and cluttered passages. This work presents a first step towards applying our unique autoencoder variant to Sampling-based Motion Planning. Our results indicate that: our framework is computationally efficient in most investigated problems, and our approach generalizes well to previously un-encoded workspaces. As such, our framework can be applied to many existing sampling-based routines to improve their computational efficiency.

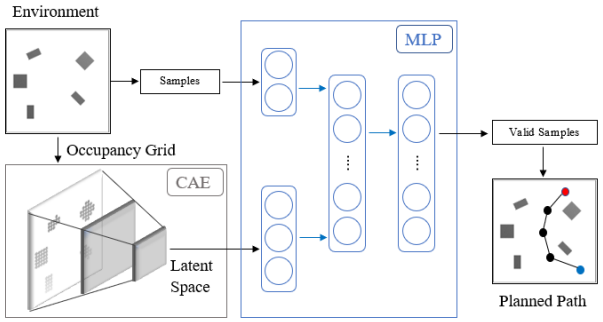


Fig. 1: Overview of the proposed framework.

II. Related Work

A. Sampling-based Motion Planning (SBMP)

A common solution to the motion planning problem is the utilization of sampling-based methods. Most often, sampling-based motion planners randomly select, progressively expand, and connect robot configurations to form approximate graph representations of X_{free}, called roadmaps. A roadmap encodes valid states as its nodes and transitions between them as its edges. To find a path, a starting state and goal region are connected to the roadmap, and a feasible path is extracted. These techniques employ the invocation of a "black box" collision detection module that classifies any state in either X_{free} or X_{obs} to avoid explicit construction of X_{obs}. This is used in the verification of both the nodes and the edges of any roadmap constructed using these methodologies.

The main paradigms of sampling-based motion planning are the Probabilistic RoadMap [9] and the Rapidly-exploring

*This work was not supported by any organization.

[1]Tuan Tran and Chinwe Ekenna are with the Department of Computer Science, University at Albany, SUNY, NY 12206, USA {ttran3, cekenna}@albany.edu.

Random Tree (RRT) algorithms [10]. Both PRM and RRT are known to be probabilistically complete, i.e., if a (robust) solution exists, then a solution will almost surely be found as the number of samples reaches infinity [7]. This work aims to improve the performance of sampling-based approaches specifically through reducing the cost of collision detection routines and subsequently the overall path planning time.

B. Neural Network in Motion Planning

Representation learning [11] mainly aims to extract features from unstructured data to either achieve a lower-dimensional representation (often referred to as encoding) or learn features for supervised learning or reinforcement learning. Recent work has utilized a learned low-dimensional latent model for motion planning. Ichter et al. [12] and Zhang et al. [13] used the Conditional Variational Autoencoders to learn a non-uniform sampling methodology of a sampling-based motion planning algorithm. Qureshi et al. [14] learned a low-dimensional representation of the obstacle space by using a Contractive Autoencoder to encode an occupancy grid of the obstacles into a latent space and then used a feed-forward neural network to predict the next step an optimal planner would take given a start and goal. Similarly, we aim to use a lower-dimensional representation to reduce the cost of collision detection for use in motion planning algorithms.

The neural network is becoming prominent in path planning for its outstanding non-linear mapping ability. Shi et al. [15] presented a motion planning system based on hybrid deep learning, which uses a convolutional neural network to reduce the dimension of the input image, a recurrent neural network to create a path tracking model, and a fully connected neural network to construct a control model. In this work, we take advantage of the multilayer perceptron to determine the validity of generated samples and improve path planning for a variety of problems.

III. RESEARCH FRAMEWORK

Our proposed framework uses two neural network models, a Contractive AutoEncoder and a Multilayer Perceptrons. The valid samples obtained from them are then used to generate a path for the robot as shown in Figure 1.

A. Contractive AutoEncoder (CAE)

A CAE is used to embed workspace occupancy grids X into an invariant and robust latent space $Z \subseteq \mathbb{R}^m$, where $m \in \mathbb{N}$ is the dimensionality of the latent space. The information of the obstacles in the workspaces is extracted from the occupancy grid and its size is compressed by the CAE. The overview of the CAE model is shown in figure 2.

The construction of a CAE consists of training neural networks for the encoder and the decoder. Let $f(x, W^e)$ be an encoding function with weight matrix W^e that encodes an input vector $x \in X$ into a vector in the latent space $z \in Z$. A decoding function $g(z, W^d)$, with weight matrix W^d, decodes a vector from the latent space $z \in Z$ back into a vector in the workspace $x \in X$. The objective function (loss function) for the CAE is:

$$L_{CAE} = \frac{1}{|D|} \sum_{x \in D} ||x - g(f(x, W^e), W^d)||^2 + \lambda \sum_{ij} (W^e_{ij})^2 \tag{1}$$

where λ is a penalizing coefficient and $D \subseteq X$ is the occupancy grid data for different workspaces. The penalizing term forces the latent space $z = f(x, W^e)$ to be contractive in the neighborhood of the training data which results in an invariant and robust feature learning [16]. Since we use a linear encoder, the loss function of CAE is similar to the loss function of a regularized autoencoder. Moreover, our training data mainly consists of an unlabeled representation of the workspace, we apply autoencoders to learn the low-dimensional representation of the training data instead of end-to-end architecture. By using the reconstruction error from the decoder, we are able to analyze a quantifiable metric used to decide which low-dimensional representation accurately replicates our workspace the best.

B. Multilayer Perceptrons (MLP)

We use a multilayer perceptron to perform sample validity. By sample validity, we mean samples that are collision-free. The overview of the MLP model is shown in figure 3. Given a workspace encoding $z = f(x, W^e) \in Z$ obtained from CAE, the samples' information s, MLP predicts its label θ whether a sample is valid or not. The encoded workspace information returned from the CAE is integrated with the sample information in the input layer of the MLP. Incorporating the information of both the obstacles and the workspace to the MLP makes it possible to determine the validity of samples for various workspaces. Without the information from CAE, the MLP only learns the statistics of the sample's validity without accurate information about the obstacles and the topology information of the workspace. The training objective for the MLP is to minimize the classification loss between the predicted sample's label and its actual label. Thus, we use the Cross-Entropy loss function since it's been shown to perform excellently for classification problems [17], [18].

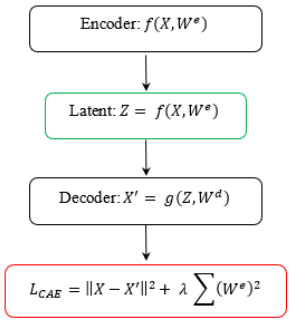


Fig. 2: Contractive AutoEncoder

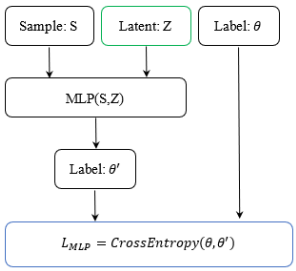


Fig. 3: Multilayer Perceptron

C. Sample Validation and Path Planning

Algorithm 1 presents the overall framework of our approach. It shows the stages between CAE and MLP and how

we obtain a roadmap for the motion planning problem.

Algorithm 1 SBMP Variant

Input. New workspace W (occupancy grid data), Query x_{init}, X_{goal}

1: CAE encodes W into the latent space Z.
2: Randomly sample a set of states, S.
3: Feed S and Z into MLP, which predicts the validity of each sample in S. After this step S' a set of most probably valid samples are retained.
4: Feed S' into a standard PRM approach to yield a valid roadmap R. If R is not found, create additional valid samples using auxiliary samplers.
5: Extract a path τ from R between x_{init} and X_{goal}.

1) Workspace Encoder: Using the trained CAE, the workspace W is encoded into a latent space Z. Then, the encoding function $f(x, W^e)$ is used to map the workspace occupancy grid $x \in X$ into a latent space $Z \subseteq \mathbb{R}^m$.

2) Sample Generation: A set of samples S is generated based on the specification of a robot R and its particular workspace W. We use the following auxiliary samplers to generate the initial sample set S from each workspace. Basic PRM (PRM) [9] creates samples uniformly and randomly. Obstacle Based PRM (OBPRM) [19] creates samples near the boundary of obstacles. Gaussian Sampler (Gauss) [20] also creates samples around obstacles using adaptive probability and collision data. Bridge Test (Bridge) [21] creates samples using a short segment with two configurations and their midpoint.

3) Sample's Validity: The trained MLP takes the sample set S and the encoded latent space Z as input to determine a set of probably valid samples $S' \subseteq S$. To introduce stochasticity into the MLP and to prevent over-fitting, dropout is applied layer-wise to a neural network and it drops each unit in the hidden layer with a probability $p : [0, 1] \in \mathbb{R}$.

4) Connection and Path Planning: The probably valid sample set S' is connected to create a roadmap R by a proximity search function. If R is not found, additional valid samples are generated using previously mentioned auxiliary samplers. Finally, from R, a local planning function would return a feasible path between a given start and goal. We use a straight-line local planner. Particularly a PRM approach is applied to S' to generate a roadmap R, and the path is extracted and verified.

IV. EXPERIMENT DETAILS

This section gives the experiment details for our framework. The proposed neural models, CAE and MLP, are implemented in PyTorch [22]. The experiments are conducted on a PC with a 3.6 GHz CPU, 16 GB memory, and NVIDIA GeForce RTX 2070 GPU.

A. Workspace Setup

We represent a workspace as an occupancy grid with values of -1 (free space) and 1 (obstacle). The input to the encoder is an occupancy grid of size $n \times m$ for 2D and $n \times m \times k$ for 3D, where n, m, and k are the number of points along each dimension of the workspace. We perform experiments in the following workspaces and generate different training and testing representations of these workspaces.

TABLE I: Workspace parameters.

	2D	Kuka	Cluttered-7	Cluttered-9
Dimension	31*31	41*41*6	11*11*11	
Number of Obstacle	3 to 5	25 to 30	110 to 125	
DoF of Robot	2	8	7	9
Size of Robot	0.5*0.5	1.5*1*1	0.1*0.1*0.1; 0.4*0.1*0.1	

- **Simple workspaces (2D)**: In 2D space shown in Figure 4, a point-mass robot with 2 degrees of freedom (DoF). The robot has to traverse through these obstacles successfully to reach its goal.
- **Kuka YouBot workspace (Kuka)**: An 8 DOF KuKa robot [23] in a workspace with four different rooms, in Figure 5. Its base has 5 DOFs that allow it to move forward, backward, and rotate (horizontally), and its arm has 3 DOFs. The robot moves through different rooms within narrow passages and performs an action (grasps or puts an object down) at its destination.
- **Cluttered workspace (Cluttered-7)**: In 3D space, obstacles are cluttered around the room as shown in Figure 6. The robot has to traverse through these obstacles successfully to reach its goal. We use rectangular robots with 7 DoF. The robot consists of 2 types of links: a small link of (0.1*0.1*0.1) and a big link of (0.4*0.1*0.1). The robot has 1 joint for 1 small link and 1 big link.
- **Cluttered workspace (Cluttered-9)**: This is the same workspace as Cluttered-7. However, we increase the DOF of the robot to 9 with 3 joints consisting of 2 small links and 2 big links.

Obstacles are randomly generated in the workspaces thus creating a diverse representation of the workspaces. For each random generation, we create a new workspace. The location, type (small, big, square, rectangular), and orientation are randomly generated. Table I gives a schematic for our workspace training and testing setup.

1) Contractive AutoEncoder: We use multiple workspace representations to train and test our CAE. The training data set of the 2D workspaces comprised of 30 workspaces, and for testing, two types of test data sets were created to evaluate the proposed method. The first test data set comprised of the 30 workspaces used in training (seen workspace), and the second test data set comprised of 10 previously unseen workspaces, i.e., workspaces not from the training set. The training data set of the Kuka workspaces comprised of 100 workspaces, and for testing 100 known workspaces and 20 previously unseen workspaces. The training data set of the Cluttered workspaces comprised of 50 workspaces, and for testing 30 known workspaces and 10 previously unseen workspaces.

2) Multi-layer Perceptrons: Utilizing the latent space information about the workspaces gotten from the CAE,

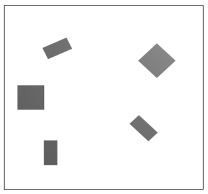


Fig. 4: Examples of 2D simple workspace

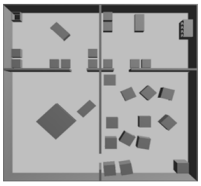


Fig. 5: Examples of Kuka workspace (8 DOF robot)

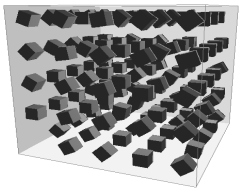


Fig. 6: Example of cluttered workspace (7 DOF and 9 DOF robot)

TABLE II: Statistic of our dataset

	CAE (workspaces)		MLP (samples/workspace)	
	Training	Testing	Training	Testing
2D	30	40	100	200
Kuka	100	120	200	2,000
Cluttered	50	40	200	400

TABLE III: Parameters of the CAE.

	2D	Kuka	Cluttered
Input dimension	31*31	41*41*6	11*11*11
Output dimension	12	50	50
Number of hidden layer	5	7	6
Activation function	Parametric Rectified Linear Unit [24]		
Loss function	Mean square error		
Learning rate adjustment	Adagrad [25]		
Dropout probability	0.5		
Penalizing term	0.001		

we generate 100 samples for each 2D workspace and 200 samples for each Kuka and Cluttered workspace for training and testing. Additionally, we don't want our samples to contain any information about the workspace, we choose to generate those samples using Basic PRM since it generates samples uniformly and randomly. Those samples are approximately distributed 50/50 between valid and invalid samples. Since our approach mainly focuses on verifying the sample's validity, we incorporate it into the standard PRM approaches using algorithm 1 to test its effectiveness. Table II gives information about our training and testing dataset.

B. Model Architecture

1) Contractive AutoEncoder: Since the structure of the decoders is the inverse of the encoder, we only describe the structure of the encoder. The parameters of the CAE are listed in table III.

For the 2D workspaces, the encoding function and decoding function consist of five linear layers and one output layer. Layers 1 to 5 transform the input vector to 512, 256, 128, 64, and 32 hidden units. For the Kuka workspaces, the encoding function and decoding function are the same as in a 2D simple workspace but consist of seven linear layers and one output layer. Layers one to seven transform the input vectors to 5043, 3125, 1600, 800, 400, 200, 100 hidden units. For the Cluttered workspaces, layers one to six transform the input vectors to 1000, 800, 600, 400, 200, 100 hidden units.

2) Multi-layer Perceptrons: The input is given by concatenating the encoded workspace's representation Z, and the DoF for each robot from a given state. Each of the layers is a combination of a linear layer, a Parametric Rectified Linear Unit (PReLU), and Dropout (p). For the 2D workspaces, MLP is a 4-layer neural network, in which the hidden layers have 6, 4 units. For both Kuka and Cluttered workspaces, MLP is a 7-layer neural network, in which the hidden layers have 50, 40, 30, 20, 10, 5 units. The parameters of the MLP is listed in Table IV.

TABLE IV: Parameters of the MLP.

	2D	Kuka	Cluttered-7	Cluttered-9
Input dimension	14	58	57	59
Output dimension	2			
Activation function	Parametric Rectified Linear Unit			
Loss function	Cross-Entropy			
Learning rate adjustment	Adagrad			
Dropout probability	0.5			
Penalizing term	0.001			

V. RESULTS

A. Performance of CAE and MLP

Table V shows the performance for the CAE for seen workspaces and unseen workspaces. For all the workspaces, the percentage accuracy for the seen workspace using our CAE was above 95% and the unseen workspace was above 93%. Thus we record excellent results.

An example of the 3D Kuka workspace and its occupancy grid reconstructed using CAE is shown in figure 7. The results of the reconstruction demonstrate that the CAE acquired the ability to extract workspace features. Overall, the accuracy of our CAE is excellent. Thus, we are able to learn the underlying structure of the workspaces quite well.

TABLE V: Performance of CAE

	Seen	Unseen
2DS	99 %	99 %
Kuka	96 %	93 %
Cluttered	97 %	94 %

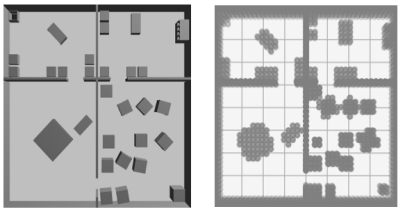


Fig. 7: Example of the workspace and its occupancy grid reconstructed using CAE.

Table VI shows the accuracy, true positive rate, and true negative rate for the MLP. The average accuracy of our MLP is from 87% to 90% for 2D workspaces, 73% to 76% for

Kuka workspace, and 70% to 76% for Cluttered workspace. Additionally, the true positive rates and true negative rates are very good for our MLP, thus an acceptable number of valid samples and invalid samples are successfully validated by MLP. Overall, the accuracy of our MLP affects the performance of our approach unnoticeably, shown in the next section. We believe that there are enough valid samples gotten from the MLP for the planner to find a feasible path. Even if there is a need for extra valid samples, those valid samples are generated using provided auxiliary samplers.

TABLE VI: Accuracy (Acc), True Positive Rate (TPR), and True Negative Rate (TNR) of MLP

	Seen (%)			Unseen (%)		
	Acc	TPR	TNR	Acc	TPR	TNR
2DS	90	95	88	87	90	64
Kuka	76	87	65	73	85	60
Cluttered-7	76	88	65	70	84	62
Cluttered-9	73	83	65	71	80	63

B. Roadmap generation using OBPRM, PRM, Gaussian and Bridge Planning Method:

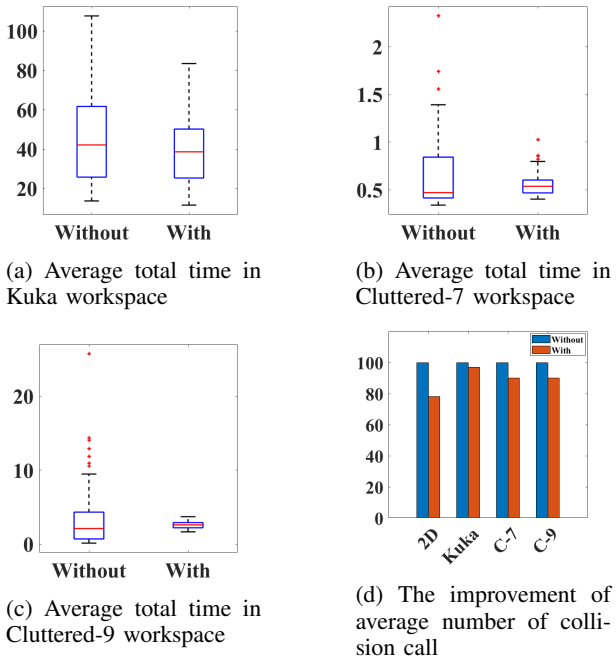
We report our improvements in generating a roadmap when the valid samples obtained from the MLP are used.

a) Results using OBPRM: The performance with and without applying our framework using OBPRM are shown in figure 8. OBPRM generates samples near the boundary of obstacles, the workspaces we studied have a large number of obstacles, and our CAE provides the representation of workspace accurately, therefore our MLP classified the sample's validity quite precisely. We see a reduction in the sampling time by approximately 50%, 10% reduction in the number of collision calls, and 13% reduction for the total time which includes sampling node connection and path generation. The standard deviations of total time when applying our framework decreased significantly. Thus, our framework augments the algorithms perfectly and makes them more stable in solving queries reliably.

b) Results using PRM, Gauss, and Bridge Test Planners: Table VII-IX show the results for all the planners in terms of the sampling time, the total time needed to build the roadmap, and the average number of collision calls returned.

In the 2D workspace, the PRM method with the inclusion of our framework shows a substantial improvement and outperforms other planners. By applying our framework, there is a good amount of reduction in the number of collision calls (13% to 22%). This is to be expected considering the uniform sampling approach of PRM and the 2D representation of the workspace with limited obstacle clutter. The variance for sampling time and total time for all of the experiments were between 0.002 and 0.03, and between 0.01 and 0.05 respectively. Other planners i.e., Gauss and Bridge Test also see an improvement when our framework was added to it.

The Gauss planner has a similar performance to OBPRM, it outperforms OBPRM only in the 2D workspace but with comparable performance in the other workspaces. Gauss planner generates samples based on a Gaussian distribution around the obstacles hence the similar performance, however,



(a) Average total time in Kuka workspace

(b) Average total time in Cluttered-7 workspace

(c) Average total time in Cluttered-9 workspace

(d) The improvement of average number of collision call

Fig. 8: The performance with and without applying our method using OBPRM

it produces slightly more collision calls for the cluttered-7 workspace but OBPRM produces less in all cases. Bridge test planner makes substantial improvement with our method but records more collision calls and average total time in the Kuka workspace which is to be expected considering the amount of non-symmetry representation of the obstacles.

Overall, there is a notable improvement in performance in all cases. Although we only test our approach using fixed-size workspaces, it is expected to be robust to varying sizes. By splitting the workspace into smaller parts with a predefined size, the proposed approach would still be able to validate samples because the CAE would be able to encode that smaller workspace and the MLP would be able to use that information for validating the samples. Moreover, the result in [26] shows that adding samples can only improve the solution. Thus, by adjusting the number of samples in the sampling validation phase and the number of samples generated by the auxiliary sampler, our method guarantees probabilistic completeness for any motion planning problem. Therefore, our approach should be applicable to other sampling-based planners.

VI. Conclusion

In this paper, we present a fast and efficient neural network framework for sampling-based motion planners. The framework consists of a Contractive AutoEncoder that encodes an occupancy grid representation of a robot's workspace into a latent feature space and a Multilayer Perceptrons that takes that encoded workspace and robot configuration details to predict the validity of that configuration. Since our method considers the workspace and robot information when performing collision checks, for future work, we plan to train them on sub-regions of the workspace. Thus, an SBMP

TABLE VII: Average sampling time (second) with and without applying our proposed framework.

	PRM	PRM-Our	OBPRM	OBPRM-Our	Gauss	Gauss-Our	Bridge	Bridge-Our
2D	0.004	**0.002**	0.05	**0.03**	0.04	**0.02**	0.26	**0.11**
Kuka	0.12	**0.07**	1.14	**0.58**	0.22	**0.13**	0.78	**0.34**
Cluttered-7	0.007	**0.004**	0.21	**0.10**	0.09	**0.04**	0.40	**0.19**
Cluttered-9	0.016	**0.011**	0.08	**0.04**	0.04	**0.02**	0.11	**0.05**

TABLE VIII: Average total time (second) with and without applying our proposed framework.

	PRM	PRM-Our	OBPRM	OBPRM-Our	Gauss	Gauss-Our	Bridge	Bridge-Our
2D	0.12	**0.10**	0.38	**0.32**	0.09	**0.08**	0.13	**0.11**
Kuka	**41.28**	41.47	43.84	**39.65**	42.02	**39.35**	**39.35**	44.85
Cluttered-7	**0.53**	0.55	0.64	**0.56**	0.61	**0.49**	0.97	**0.61**
Cluttered-9	0.74	**0.61**	3.30	**2.58**	3.86	**3.23**	6.38	**4.62**

TABLE IX: Average number of collision calls with and without applying our proposed framework.

	PRM	PRM-Our	OBPRM	OBPRM-Our	Gauss	Gauss-Our	Bridge	Bridge-Our
2D	31,783	**25,819**	156,566	**122,512**	8,503	**7,388**	42,384	**36,225**
Kuka	**436,965**	441,334	689,146	**668,361**	474,658	**460,441**	**404,023**	420,680
Cluttered-7	**11,164**	11,203	14,855	**13,404**	**12,764**	14,096	18,058	**14,450**
Cluttered-9	24,059	**21,895**	166,155	**149,182**	150,224	**141,017**	195,632	**160,144**

algorithm could use our method when it reaches certain portions of that workspace. Additionally, this approach can be applied to other aspects of SBMP, e.g., edge prediction, and path validation.

REFERENCES

[1] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *ieee Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[2] C. L. Nehaniv and K. E. Dautenhahn, *Imitation and social learning in robots, humans and animals: behavioural, social and communicative dimensions.* Cambridge University Press, 2007.

[3] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[4] J. Denny, K. Shi, and N. M. Amato, "Lazy toggle prm: A single-query approach to motion planning," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2407–2414.

[5] L. Kobbelt and M. Botsch, "A survey of point-based techniques in computer graphics," *Computers & Graphics*, vol. 28, no. 6, pp. 801–814, 2004.

[6] C. Ekenna, S. Thomas, and N. M. Amato, "Adaptive local learning in sampling based motion planning for protein folding," *BMC systems biology*, vol. 10, no. 2, p. 49, 2016.

[7] J. Bialkowski, S. Karaman, M. Otte, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 365–380.

[8] M. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *Proc. of IMA conference on mathematics of surfaces*, vol. 1, 1998, pp. 602–608.

[9] L. Kavraki, P. Svestka, and M. H. Overmars, *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. Unknown Publisher, 1994, vol. 1994.

[10] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[12] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[13] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3654–3661.

[14] A. H. Qureshi, M. J. Bency, and M. C. Yip, "Motion planning networks," *arXiv preprint arXiv:1806.05767*, 2018.

[15] C. Shi, X. Lan, and Y. Wang, "Motion planning for unmanned vehicle based on hybrid deep learning," in *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*. IEEE, 2017, pp. 473–478.

[16] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011, pp. 833–840.

[17] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*. ACM, 2016, pp. 191–198.

[18] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 4153–4156.

[19] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Obprm: An obstacle-based prm for 3d workspaces," 1998.

[20] V. Boor, M. H. Overmars, and A. F. Van Der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *ICRA*, 1999, pp. 1018–1023.

[21] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *2003 IEEE international conference on robotics and automation (cat. no. 03CH37422)*, vol. 3. IEEE, 2003, pp. 4420–4426.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[23] K. R. Corporation, "Kuka youbot," www.youbot-store.com, accessed: June 1, 2013.

[24] L. Trottier, P. Gigu, B. Chaib-draa *et al.*, "Parametric exponential linear unit for deep convolutional neural networks," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 207–214.

[25] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[26] L. Janson, B. Ichter, and M. Pavone, "Deterministic sampling-based motion planning: Optimality, complexity, and performance," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 46–61, 2018.