Visual Repetition Sampling for Robot Manipulation Planning

En Yen Puang[1], Peter Lehner[1], Zoltan-Csaba Marton[1], Maximilian Durner[1]
Rudolph Triebel[1,2], Alin Albu-Schäffer[1,2]

***Abstract*— One of the main challenges in sampling-based motion planners is to find an efficient sampling strategy. While methods such as Rapidly-exploring Random Tree (RRT) have shown to be more reliable in complex environments than optimization-based methods, they often require longer planning times, which reduces their usability for real-time applications. Recently, biased sampling methods have shown to remedy this issue. For example Gaussian Mixture Models (GMMs) have been used to sample more efficiently in feasible regions of the configuration space. Once the GMM is learned, however, this approach does not adapt its biases to individual planning scene during inference. Hence, we propose in this work a more efficient sampling strategy to further bias the GMM based on visual input upon query. We employ an autoencoder trained entirely in simulation to extract features from depth images and use the latent representation to adjust the weights of each Gaussian components in the GMM. We show empirically that this improves the sampling efficiency of an RRT motion planner in both real and simulated scenes.**

I. INTRODUCTION

Mobile robotic manipulation systems are becoming more and more important for industrial applications, because modern production lines are becoming more customized, and this requires flexible mobile robots that can adapt to new situations. Consequently, planning efficient and collision-free paths for the manipulator must be performed online and can not be pre-processed. However, the time constraints of most industrial applications do not allow any intensive search in Configuration-space ($\mathcal{C}$-space), which limits the usability of most of the current path planning methods. For example, RRT belong to a recent family of Sampling-based Motion Planner (SMP), but despite being relatively fast, in general they still exhibit an inefficient sampling strategy. This often leads to large areas in configuration space being explored and consequently to long planning times.

Recent work has focused on reducing the computation time by exploiting past experiences for efficient $\mathcal{C}$-space sampling. Repetition Sampling [1] in particular learns a probability distribution to bias the search of SMP into the task relevant regions of the $\mathcal{C}$-space. The algorithm learns the probability distribution by combining the key information of previous, similar queries in a GMM. The method shows promising results in computational efficiency, but is unaware if the current query actually reflects a previous experience.

In this work, we propose an efficient combination of visual scene recognition system and experience based motion planner to *recognize* and *exploit* past experiences in solving new motion planning queries efficiently, as shown in Fig. 1. We design and train an Autoencoder (AE) to extract key information about the planning scene directly from depth image. Based on the information encoded by the AE we then learn sampling distributions for similar planning queries with Repetition Sampling. During inference our methods further bias the sampling distribution of Repetition Sampling to reflect the task relevant configuration space. The main contributions here therefore involve two major areas:

- Improve efficiency of SMP for robot manipulation task with deep-learned scene-understanding vision module.
- Improve data efficiency of training AE with only simulated data and adaptation-randomization methods for robotics application.

We evaluate the complete pipeline on pick-and-place experiments in simulation as well as on our real mobile manipulator. The results show that visual workspace understanding of past experiences improve sampling efficiency and hence decrease computation time during execution.

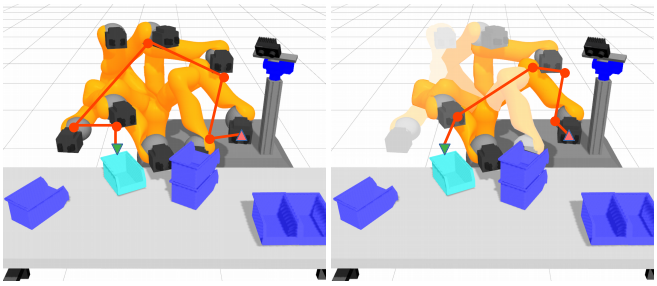

Fig. 1: Adapting GMM sampling biases (right) in RRT based-on visual information deduced from planning scene resulted in more efficient planning compared to vanilla Repetition Sampling (left).

II. RELATED WORK

Boor et al. [2] proposed to only accept samples that are close to obstacle while Yang et al. [3] proposed the opposite by biasing along the medial axis. The utility-guided approach [4], maximizes the information gain of the next sample by predicting the sample utility with a probability distribution. Informed path sampling [5] learns a probability distribution to generate paths which are robust to obstacle uncertainties. These methods bias the sampling distribution of SMP but are not designed to include information from previous experiences. Kernel Density Estimation (KDE)-based sampling [6] and Repetition Sampling [1] utilize previous

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Germany. en.puang@dlr.de
[2]Department of Computer Science, Technical Univ. of Munich, Germany.

experiences by estimating the probability distribution in $\mathcal{C}$-space based on solutions of a set of similar tasks using KDE and GMM respectively. During inference, however, GMM need not to select a kernel for samples drawing, which is a critical aspect for KDE-based biases.

Phillips et al. [7] proposed a biased roadmap planner by introducing cost on every edges in the graph. The graph is built using a solutions collection and a discrete planner is used to compute path with minimal cost when queried. Zucker et al. [8] extract features from a discretized workspace using a discrete planner and then apply reinforcement learning to update the workspace bias distributions using shorter planning time as reward. Berenson et al. [9] propose a framework that collects solutions in a sparse roadmap and exploits it via a retrieve-repair module. These methods discretize the $\mathcal{C}$-space in contrast to Repetition Sampling which learns a probability distribution in continuous $\mathcal{C}$-space without using workspace feature. Utilizing Repetition Sampling, our work appends the ability to adapt biases according to situations.

Ellekilde et al. [10] blend several segments of solution retrieved from database into a smooth one. Jetchev et al. [11] proposed to penalize differences from the retrieved solutions in its inverse kinematic. In contrast our methods do not use the retrieved solutions directly in generating new solution. In further work the same authors proposed a planning scene descriptor using voxel grid and principal component analysis [12]. Scene similarity computed with a function whose weights are optimized by minimizing the required effort to adapt retrieved solutions to query scene. Our work does not require hand-crafting features to represent the planning scene. Ichter et al. [13] proposed to reconstruct solutions using Variational AE, conditioned on planning scene new samples are generated by sampling on latent distribution and forwarded to the decoder. This method offers promising algorithm but did not address cases in which the planning scenes are in high dimension i.e. image.

Sharif et al. [14] shows the competitiveness of Convolutional Neural Network (CNN) compared to classical methods in multiple visual tasks. Recent work used pre-trained networks for feature extraction in image retrieval task [15], [16]. General purpose pre-trained networks are normally huge and offer limited architecture flexibility. Our method trains a significantly smaller network from scratch with reduced computation cost. Using variants of AE Sundermeyer et al. [17] proposed object pose estimation; Kendall et al. [18] proposed scene semantic segmentation; and Nguyen et al. [19] proposed tabletop affordance detection. Gupta et al. [20] proposed hand-crafted features for depth image (HHA). Our work uses a variant of AE, but without HHA encoding for faster data pre-processing.

Recent works fine-tune feature extraction networks end-to-end so that the resulted descriptors yield high similarity score for correctly paired input [21], [22]. These methods require paired data with a defined relationship which is not applicable in general database similarity search. Our methods employ self-supervise training to circumvent the need of paired images and hence achieve high data efficiency. Li et al. [23] proposed to predict similarity scores directly. Besides requiring paired data, this method does not transform images into smaller representations and hence requires repeated network's forward-passes for each query. Mohedano et al. [15] proposed a Bag-of-Words method for visual retrieval which involve CNN features, K-Means clustering and histogram descriptors. These method has good accuracy but lacks the runtime efficiency required in robotic applications. Our method uses entirely CNN that computes the latent representation in a single encoder's forward-pass.

III. Repetition Sampling

Repetition Sampling improves the efficiency of SMP by biasing the sampling towards specific regions in $\mathcal{C}$-space. It starts by building a database $\mathcal{D}$ consisting of n robot motion paths $\mathcal{P}$ that are generated in a particular task domain using standard RRT. Important *milestones* or key-configurations $\mathbf{q} \in \mathcal{C}$ are then extracted $\Psi : \mathcal{P} \rightarrow \{\mathbf{q}\}$ from each paths in the database and form $\mathcal{Q}$ with a total N key-configurations

$$\mathcal{D} = \left\{\mathcal{P}_i\right\}_{i=1}^{n} \tag{1}$$

$$\mathcal{Q} = \bigcup_{i=1}^{n} \Psi\left(\mathcal{P}_i\right). \tag{2}$$

Next, $\mathcal{Q}$ is clustered using GMM to obtain a parametric approximation of the density distribution in $\mathcal{C}$-space. Mean $\boldsymbol{\mu}_j$, covariance Σ_j and weight w_j of each mixture components j where $j = 1, \ldots, G$ and G is the total number of mixture components are then estimated using Expectation-Maximization (EM). Concretely, the E-step computes the assignment probabilities $\mathbf{p}_i \in [0,\, 1]^G$ s.t. $\sum_{j=1}^{G} p_{ij} = 1$ for each key-configurations in $\mathcal{Q}$ where $i = 1 \ldots N$

$$p_{ij} = \frac{w_j\, \mathcal{N}(\mathbf{q}_i \,|\, \boldsymbol{\mu}_j, \Sigma_j)}{\sum_{l=1}^{G} w_l\, \mathcal{N}(\mathbf{q}_i \,|\, \boldsymbol{\mu}_l, \Sigma_l)}. \tag{3a}$$

Thus, p_{ij} represents the probability that key-configuration $\mathbf{q}_i$ corresponds to mixture component j. Then, in the M-step the mixtures' parameters are updated

$$\boldsymbol{\mu}_j = \frac{1}{n_j} \sum_{i}^{N} p_{ij}\, \mathbf{q}_i \tag{3b}$$

$$\Sigma_j = \frac{1}{n_j} \sum_{i}^{N} p_{ij}\, (\mathbf{q}_i - \boldsymbol{\mu}_j)(\mathbf{q}_i - \boldsymbol{\mu}_j)^{\mathsf{T}} \tag{3c}$$

$$w_j = \frac{n_j}{N}\,, \qquad\qquad n_j = \sum_{i}^{N} p_{ij}. \tag{3d}$$

After convergence of EM, Repetition Sampling randomly selects a Gaussian component $(\boldsymbol{\mu}_j, \Sigma_j)$ proportional to its weight w_j, to generate samples used by RRT for expanding the search tree. However, as mentioned above, this approach cannot adapt to individual new query scene because it does not involve information obtained from perception. Therefore, we propose an alternative method in the following section.

IV. Biased Repetition Sampling

Our main idea is to modify the weights of mixture components according to the information deduced from individual query scene using a vision component. More concretely, we extend our model by incorporating a depth image of the initial planning scene into biasing the vanilla Repetition Sampling using two methods, namely Weight Aggregation (WA) and Weight Prediction (WP).

A. Weight Aggregation

As shown in Fig. 2 we first populate the database in simulation with paired depth image $\mathcal{I}$ of the initial planning scene and all queries Φ in the scene. Each query ϕ contains a target pose $\mathbf{t}_\phi$ (end-effector's 6 Degrees of Freedom (DoF) rigid transformation) and is solved as described in Sec. III

$$\mathcal{D} = \left\{ \left(\mathcal{I}_i,\, \Phi_i \right) \right\}_{i=1}^{n}. \tag{4}$$

To be able to *reuse* existing solution in the database during inference, we employ an AE to facilitate *retrieval* based-on the associated planning scenes. This AE learns to represent the planning scene in the form of depth image $\mathcal{I} \in \mathbb{R}^{w \times h}$ with a latent-encoding $\mathbf{z} \in \mathbb{R}^{\ell}$ where $\ell \ll w \times h$. Concretely, we treat the encoding as *key* and the queries associated to the scene as *value* for *key*$\|$*value* retrieval in the database

$$\mathrm{T}_i = \begin{bmatrix} \mathbf{t}_1^{\mathsf{T}} \\ \vdots \\ \mathbf{t}_{|\Phi_i|}^{\mathsf{T}} \end{bmatrix} \qquad \mathrm{P}_i = \begin{bmatrix} \{\mathbf{p} \in \Psi(\mathcal{P}_1)\} \\ \vdots \\ \{\mathbf{p} \in \Psi(\mathcal{P}_{|\Phi_i|})\} \end{bmatrix} \tag{5a}$$

$$\mathcal{D} = \left\{ \left(\mathbf{z}_i \,\|\, \mathrm{T}_i,\, \mathrm{P}_i \right) \right\}_{i=1}^{n}. \tag{5b}$$

When a real planning task $(\mathcal{I}_\mathrm{q}, \mathbf{t}_\mathrm{q})$ is queried during inference, retrieving the right information from the database requires a metric that reflects both the visual similarity between scenes and the geometric similarity between target poses. The first stage of the retrieval uses σ_s to retrieves the K Nearest Neighbors (KNN) from the database in term of the top K $[\cdot]_\mathrm{K}$ cosine similarities between queried encoding $\mathbf{z}_\mathrm{q}$ and all *key*s in the database

$$\sigma^s(\mathbf{z}_\mathrm{q}) = \left[\left\{ \frac{\mathbf{z}_\mathrm{q}^{\mathsf{T}} \mathbf{z}_i}{\|\mathbf{z}_\mathrm{q}\| \, \|\mathbf{z}_i\|} \,\middle|\, \mathbf{z}_i \in \mathcal{D} \right\} \right]_\mathrm{K}. \tag{6}$$

The second stage uses σ^t to compute task similarity between queried target pose $\mathbf{t}_\mathrm{q}$ and those in the KNN. L2-Norm is used to compute the differences between target poses and α_t is the hyper-parameter used to tune the sensitivity of σ^t

$$\sigma^t(\mathbf{t}_\mathrm{q},\, \mathbf{t}_i) = \frac{1}{2} \cos\left(\min\left(\pi,\, \alpha_t \left\| \mathbf{t}_\mathrm{q} - \mathbf{t}_i \right\|_2 \right) \right) + \frac{1}{2}. \tag{7}$$

The proposed new set of weights $\mathbf{w}'$ is then a weighted average of assignment probabilities of all key-configurations contained in the KNN. $\mathrm{P}_{k\phi lj} = p_{(k\phi l),j}$ is the assignment probability for j-th mixture component of l-th key-configuration in ϕ-th query in scene k in the database, and

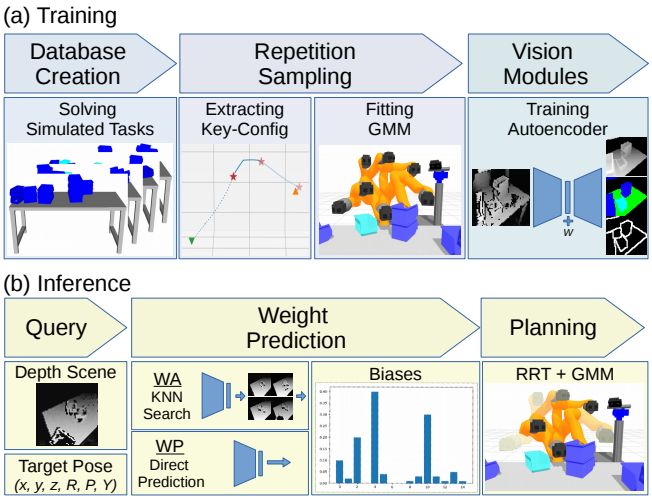

Fig. 2: Training phase first accumulates planning scenes paired with generated paths before fitting a GMM using extracted key-configurations. AEs are then trained for their respective biasing methods using data from the database. During inference both WA and WP produce a new set of weights to bias the GMM used in Repetition Sampling.

$w' = \sum_j^G w'_j$ is the normalizing factor for all j

$$w'_j = \frac{1}{w'} \sum_k^{\mathrm{K}} \sum_\phi^{|\Phi_k|} \sum_l^{|\mathrm{P}_{k\phi}|} \sigma^s(\mathbf{z}_\mathrm{q})_k \, \sigma^t(\mathbf{t}_\mathrm{q},\, \mathrm{T}_{k\phi}) \, \mathrm{P}_{k\phi lj} \tag{8}$$

$$w^*_j = \alpha_w \sup \mathcal{S} \, w'_j + (1 - \alpha_w \sup \mathcal{S}) \, w_j. \tag{9}$$

By calculating the final weight $\mathbf{w}^*$, the supremum of the similarity score $\sup \mathcal{S}$ is used to adaptively scale the update factor α_w, reducing updates when no good match can be found. A high update factor favor visual biasing over the statical estimations from the database, while the opposite attenuates adaptive biasing.

B. Weight Prediction

Using the same database structure in (5), a weights predictor $\mathrm{W}(\mathbf{w}' \,|\, \mathbf{z},\, \mathbf{t})$ is built on top of the same AE as an additional *softmax* output head to predict the weights of the mixture components $\mathbf{w}' \in [0,\, 1]^G$ directly using the encoding $\mathbf{z}$ and target pose $\mathbf{t}$.

$$\begin{aligned} \mathcal{L}_\mathrm{W} &= \alpha_W \left[D_\mathrm{KL}(\mathbf{w}' \,|\, \mathbf{w}) \right]_\mathrm{K} \\ &= \alpha_W \sum_j^{\mathrm{K}} w_j \log \frac{w_j}{w'_j} \end{aligned} \tag{10}$$

$\mathcal{L}_\mathrm{W}$ is a Kullback–Leibler (KL) divergence loss designed to minimize differences in probability distribution between prediction $\mathbf{w}'$ and target $\mathbf{w}$. This target distribution is the assignment probability $\mathbf{p} \in \mathrm{P}_{i\phi}$, paired with $\mathbf{z}_i$ and its corresponding $\mathbf{t}_\phi \in \mathrm{T}_i$. $\mathbf{p}$ are summed component-wise and re-normalized if $|\mathrm{P}_{i\phi}| > 1$.

Operator $[\cdot]_\mathrm{K}$ includes only the K largest components' losses out of G. Due to the peaked distributions in $\mathbf{p}$, the top K of KL divergence consist of those that undershoot

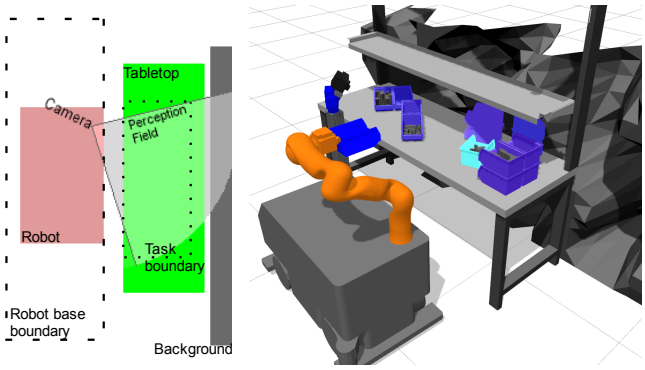

Fig. 3: Simulated workspace. Perlin noise [25] is used in generating uneven surfaces for background and SLC load.

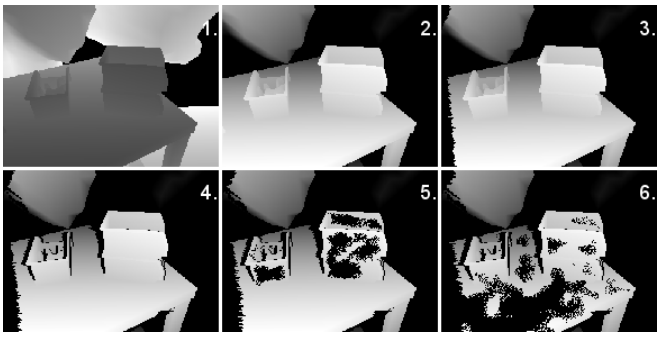

Fig. 4: Synthetic depth image augmentations involve distance crop, normalization, invert, baseline shift, depth shadow and random homogeneous dropout on objects and table.

$w_j/w'_j > 1$ and a lot of those with $w_j = 0$. Those that overshoot $w_j/w'_j < 1$ are then indirectly excluded. Since the network has the tendency to bias itself into learning the empirical weight distribution of the entire dataset, penalizing only the undershoots has the implication of learning scene specific knowledge while preserving the background information. We therefore set K to be half of G for $\mathcal{L}_{\mathrm{W}}$.

$$w_j^* = \alpha_w \mathrm{W}(\mathbf{z}_\mathrm{q}, \mathbf{t}_\mathrm{q})_j + (1 - \alpha_w)\, w_j \tag{11}$$

During inference the weights predictor is fed with a paired latent code $\mathbf{z}_\mathrm{q}$ and target pose $\mathbf{t}_\mathrm{q}$ of a query scene. With the update factor α_w, the final weight $\mathbf{w}^*$ is computed without adaptive scaling as in (9).

V. Implementation

A. Database Creation

For evaluation we use tabletop pick-and-place scenes which involve a robot with a manipulator on a mobile platform [24] manipulating Small Load Carrier (SLC). The robot perceives the workspace with a stereo camera system mounted on a vision mast. The robot is able to move its base along the x-axis (1 DoF) and arm (7 DoF) while reaching for empty nullspace pre-grasp/release target pose. The arm has a fixed start configuration while the base is randomly placed along the table edge at the beginning shown in Fig. 3. The only kinematic constraint requires the end effector to be upright all the time. Each scene contains 1-2 pick-tasks and 0-3 place-tasks depending on the random obstacle arrangement on the tabletop.

In Cartesian-space, we specify the DoF of the end effector for obstacle avoidance and require the rest of DoF to be simple if not fixed by kinematic constraints. For the tabletop scenario the z-axis of the end effector is assigned for obstacle avoidance. Straight line curve fittings are therefore applied on the xy Cartesian paths of the end effector. By thresholding the goodness-of-fit between actual and regressed straight line we populate the database only with low variance solutions.

For the extraction of key-configurations, we use the discrepancies between actual and straight line path to indicate the likelihood of useful configuration. Given a low variance path, a group of corresponding straight lines is constructed by using the DoF of the end effector in Cartesian-space. $\Psi(\mathcal{P})$ uses local extremal detectors to determine the time-steps where the amount of deviations exceed certain threshold.

B. Autoencoder

DenseNet [26] with *Weight-BatchNorm-Activation* architecture is the backbone of this AE. It is different from the original *BatchNorm-Activation-Weight* combination. Since *BatchNorm-Activation* make no difference when placed before or after a feature concatenation, our proposed combination prevents unnecessary computations by concatenating activated features directly. Dilated convolution [27] is applied in every dense-layer and the amount of dilation goes in the cycle of 1 to 3 until the end of each dense-block.

Given a depth image, the decoder uses the latent code generated by the encoder to predict a clean input reconstruction x_{rec}, semantic segmentation x_{sec} and object boundary x_{bou} shown in Fig. 5. The loss functions are smooth L1, multi-class and binary cross-entropy respectively. On-line bootstrapping [28] is implemented in all 3 losses, and only pixels with the highest loss per image are back-propagated.

C. Data Augmentation

To bridge the *reality-gap*, our applied synthetic training data generation takes the hardware properties of targeted application into account. Hence, artifacts in the depth images acquired by a passive stereo camera system are artificially introduced (see Fig. 4):

- **Baseline Shift** induced blind region at vertical image boundary is implemented by nullifying left-most pixels.
- **Depth Shadow** caused by occlusion is added according to the magnitude of horizontal image gradient.
- **Homogeneous Dropout** simulates failed depth estimation using distance to nearest image edge and [25] mask.

Unlike [29] we use the entire mini-batch, randomized augmentation strength and adversarial example generation (Fast Gradient Sign Method (FGSM), iterative FGSM and least-likely FGSM) for the regression and classification tasks to further avoid synthetic features from over-fitting.

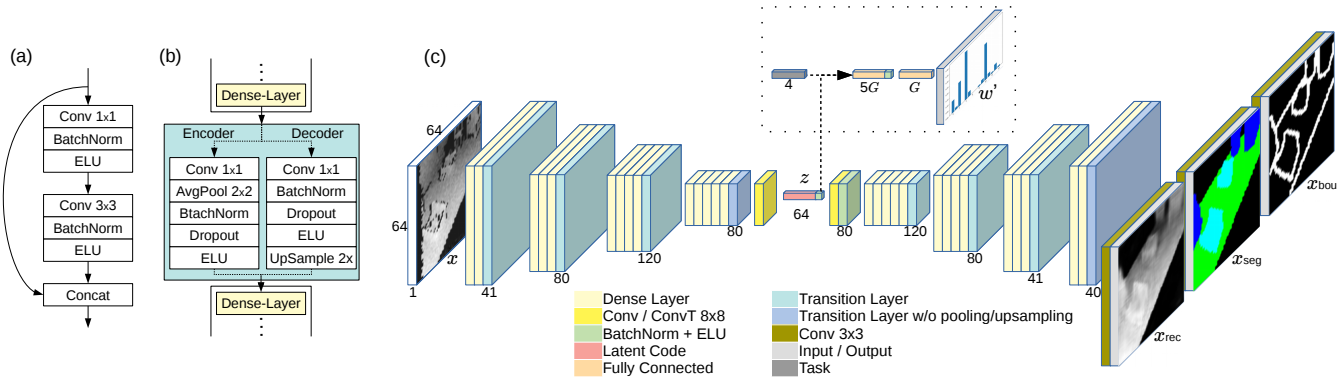

Fig. 5: (a) Dense-layer with bottleneck; (b) Transition-layer in encoder and decoder. (c) Autoencoder with weight prediction head. The number of trainable network parameters is 1.9M, which split equally (0.95M) among encoder and decoder.

TABLE I: Combinations of multi-tasked decoders (left) and randomized adversarial augmentation strengths ϵ (right).

rec	***seg***	***bou***	S^3
✓			0.7540
✓		✓	0.7533
✓	✓		0.7900
✓	✓	✓	**0.7929**

ε	S^3
0	0.7891
0.1	0.7918
0.3	**0.7929**
0.5	0.7914

TABLE II: Number of nearest neighbors required to retrieve the correct scene in real test-set. $K \leftarrow 5$ in all other WA.

KNN	1	2	3	5	10	20
Retrieval (%)	62	83	88	91	97	100

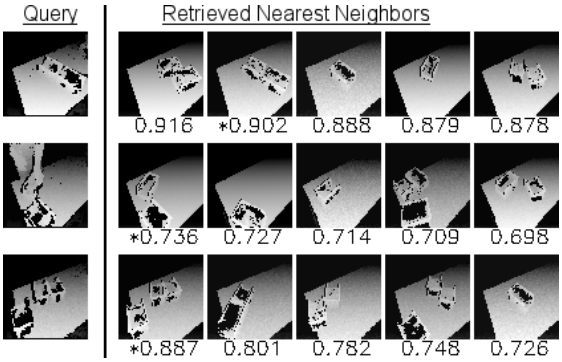

Fig. 6: Examples of simulated scene retrieval for real queries. $*$ indicates the corresponding identically rendered scene.

VI. Evaluation

A. Scene Retrieval Accuracy

In total 14000 scenes are generated. The first half of them contains all training labels i.e. image-path pairs; The second half contains only images. With the full information, the former half forms the database $\mathcal{D}$ with 7000 scenes, and is able to train the entire network. Whereas the latter, which was acquired relatively easier and with faster pace, is excluded from updating weights prediction head.

To evaluate the image retrieval, semantic segmentation labels of query $\mathfrak{q}$ and K retrieved scenes $\mathcal{S}$ are used for direct comparison in Semantic Scene Similarity (S^3). $x^s_{i,j}$ is the semantic label of the i^{th} image at j^{th} pixel location.

$$S^3 = \frac{1}{|\mathfrak{t}|}\frac{1}{K}\frac{1}{w \times h}\sum_{i=1}^{|\mathfrak{t}|}\sum_{k=1}^{K}\sum_{\ell=1}^{w\times h}\begin{cases}1 & if\ x^s_{\mathfrak{q}^i,\ell} \wedge x^s_{\mathcal{S}^k,\ell} \\ 0 & otherwise\end{cases} \quad (12)$$

is averaged across $|\mathfrak{t}| = 500$ test scenes, their respective $K = 5$ nearest neighbors and the number of pixels per output.

As shown in Table I, we evaluate the effects of multi-tasking the decoder on S^3 by using various combinations of input reconstruction ***rec***, semantic segmentation ***seg*** and boundary prediction ***bou***. We furthermore show the effects of augmenting training data on S^3 by applying the randomized adversarial example with varying strength ε.

The additional semantic segmentation and boundary prediction tasks force the latent code to be more informative without increasing the code length. This resulted in a more efficient database search as well as better robustness to noise.

In order to draw conclusions about the generalizability of our method, we investigate the ability of applying knowledge learned in simulated environment onto the real world. Therefore, 100 real planning scenes with ground truth poses were collected and their identically rendered scene in simulation is added to the existing database. Table II shows the amount of nearest neighbors required to retrieve the correct rendered scene given the corresponding real query scene. Fig. 6 depicts some examples of planning scenes retrieval.

B. Biased Sampling Efficiency

The test-set $\mathfrak{t}$ for this evaluation consists of 200 scenes with a total of 528 pick-and-place tasks. For the final path optimization RRTs are given $300ms$. The vision modules in WA and WP take about $10ms$ for image encoding and weights generation.

We propose performance metric AUCf which represents the normalized area under curve of cumulative frequency of planning time histogram with $100ms$ resolution over the range of $5s$, which is the threshold for failed attempt by timeout. Normalized by timeout threshold and the size of the

TABLE III: Left: Hyper-parameters search. α_t involves parameters for xy translation and z euler angle of end effector. Right: Motion planner benchmark with 528 planning instances in pick-and-place scenes.

	α_t	α_W	α_w	ΔAUCf (%) 15G	30G	45G
WA	(7.0, 2.5)	-	0.50	6.5	6.4	**4.5**
			0.75	6.5	**6.8**	4.5
			1.00	**6.7**	6.7	4.5
WP	-	1.5	1.00	6.1	7.0	**7.8**
		3.0		**7.2**	7.0	7.2
		4.0		6.9	**7.0**	6.9

	RRT $\mathcal{U}$	RRT GMM 15G	RRT GMM 30G	RRT GMM 45G	RRT GMM WA	RRT GMM WP	KOMO
Failure	26	6	4	5	4	**1**	26
Median (s)	0.357	0.356	0.357	0.359	0.365	0.365	**0.140**
Min (s)	0.339	0.338	0.339	0.341	0.348	0.348	**0.080**
AUCf	0.856	0.914	0.915	0.917	0.924	**0.934**	0.933

test-set AUCf's output ranges from 0 meaning the planner failed the entire test-set, to 1 meaning the planner solved every query within $100ms$.

We apply both methods with RRT planner and evaluate the effects of hyper-parameters by comparing their AUCf with uniform samplings. The best performing combinations are then used in the benchmarking among vanilla repetition sampling and KOMO [30] the optimization-based planner. KOMO has squared accelerations and sum of squared pose errors as costs, and inequality constraints for collision and joint limits. Its trajectories have 1 phase and 20 time slices in $5s$. Additional criteria for failed attempts involve limiting collision constraint and task cost at 0.01 and 1.0 respectively.

The deterioration of WA's performance with large number of components as depicted in Table III (left) seems to be the result of the inability to represent highly non-linear similarity scores by the cosine similarity and task proximity function. WP in contrast is able to utilize the detailed density estimation provided by the larger number of components. Table III (right) shows where WP achieves the highest success rate and maintained high overall planning efficiency.

Altering failure definitions do change the landscape of comparison between sampling-based and optimization-based planner. The point here is to show their pros and cons as they cover different area in AUCf depicted in Fig. 8.

Using the same 100 real and identically rendered planning scenes in the full pipeline evaluation, we show that our methods achieve high data efficiency by only using synthetic data in the entire training and yet are able to perform in both real and simulated input domain. Fig. 9 depicts the negligible differences in AUCf using the best performing WP.

VII. CONCLUSION

This work introduces vision assisted biasing for adaptive $\mathcal{C}$-space biased sampling in RRT, an extension for Repetition Sampling that only offers fixed biases in GMM. The end-to-end method (WP) that predicts weights directly performs better than the modular method (WA) which contains similarity measures and solutions aggregation modules. The modularity however allows the method to work even with other sets of GMMs (with on-line assignment probability adaptation), which would requires re-training for end-to-end method. Both methods nevertheless enhance the performance of vanilla Repetition Sampling and show competitiveness with optimization-based counterpart without introducing significant overhead during inference. Furthermore, a successful and data-efficient *sim2real* is achieved by using multiple adaptation-randomization methods when training the AE.

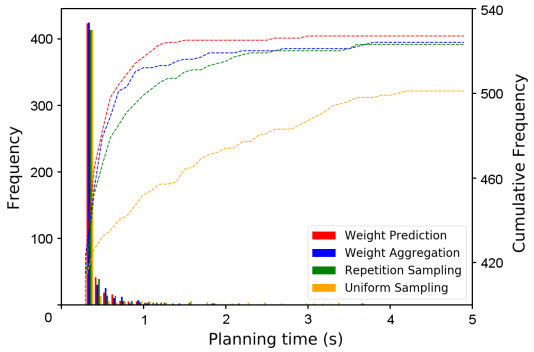

Fig. 7: Comparison between difference sampling strategies.

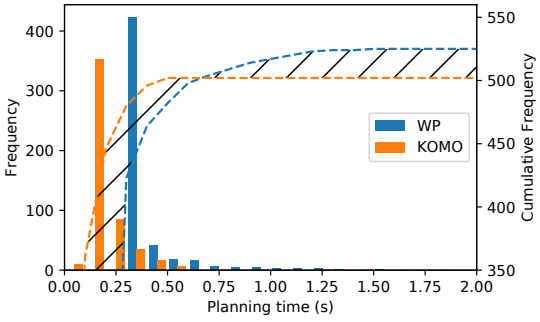

Fig. 8: Comparison between WP and KOMO in AUCf.

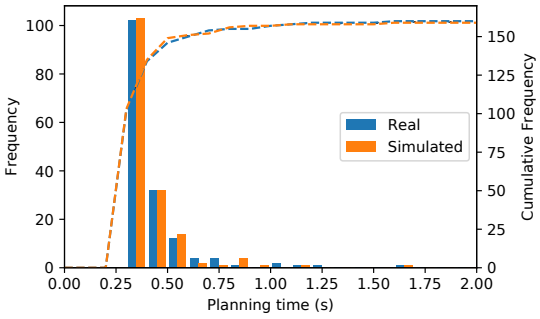

Fig. 9: Comparison between real and simulated scene using WP. Both yielded ~0.93 AUCf in the real test-set.

REFERENCES

[1] P. Lehner and A. Albu-Schäffer, "Repetition sampling for efficiently planning similar constrained manipulation tasks," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 2851–2856.

[2] V. Boor, M. H. Overmars, and A. F. Van Der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Proc. IEEE International Conference on Robotics and Automation*, 1999, pp. 1018–1023.

[3] Y. Yang and O. Brock, "Adapting the sampling distribution in PRM planners based on an approximated medial axis," in *Proc. IEEE International Conference on Robotics and Automation*, 2004, pp. 4405–4410.

[4] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Proc. Robotics: Science and Systems*, 2005, pp. 105–112.

[5] R. A. Knepper and M. T. Mason, "Real-time informed path sampling for motion planning search," *The International Journal of Robotics Research*, 2012.

[6] T. F. Iversen and L.-P. Ellekilde, "Kernel density estimation based self-learning sampling strategy for motion planning of repetitive tasks," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 1380–1387.

[7] M. Phillips, B. J. Cohen, S. Chitta, and M. Likhachev, "E-graphs: Bootstrapping planning with experience graphs," in *Proc. Robotics: Science and Systems*, 2012.

[8] M. Zucker, J. Kuffner, and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners," in *Proc. IEEE International Conference on Robotics and Automation*, 2008, pp. 3757–3762.

[9] D. Berenson, P. Abbeel, and K. Goldberg, "A robot path planning framework that learns from experience," in *Proc. IEEE International Conference on Robotics and Automation*, 2012, pp. 3671–3678.

[10] L.-P. Ellekilde and H. G. Petersen, "Motion planning efficient trajectories for industrial bin-picking," *The International Journal of Robotics Research*, 2013.

[11] N. Jetchev and M. Toussaint, "Fast motion planning from experience: Trajectory prediction for speeding up movement generation," *Autonomous Robots*, 2013.

[12] ——, "Trajectory prediction in cluttered voxel environments," in *Proc. IEEE International Conference on Robotics and Automation*, 2010, pp. 2523–2528.

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

[14] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proc. IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 806–813.

[15] E. Mohedano, K. McGuinness, N. E. O'Connor, A. Salvador, F. Marqués, and X. Giro-i Nieto, "Bags of local convolutional features for scalable instance search," in *Proc. International Conference on Multimedia Retrieval*. ACM, 2016, pp. 327–331.

[16] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," *Proc. International Conference on Learning Representations*, 2016.

[17] M. Sundermeyer, E. Y. Puang, Z.-C. Marton, M. Durner, and R. Triebel, "Learning implicit representations of 3d object orientations from rgb," in *Proc. IEEE conference on robotics and Automation workshops*, 2018.

[18] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," 2015.

[19] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 2765–2770.

[20] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *Proc. European Conference on Computer Vision*. Springer, 2014, pp. 345–360.

[21] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 539–546.

[22] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[23] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 152–159.

[24] A. Dömel, S. Kriegel, M. Kaßecker, M. Brucker, T. Bodenmüller, and M. Suppa, "Toward fully autonomous mobile manipulation for industrial environments," *International Journal of Advanced Robotic Systems*, 2017.

[25] K. Perlin, "Improving noise," in *Proc. Transactions on Graphics (TOG)*. ACM, 2002, pp. 681–682.

[26] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," 2016.

[27] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2016.

[28] Z. Wu, C. Shen, and A. v. d. Hengel, "Bridging category-level and instance-level semantic image segmentation," *arXiv preprint arXiv:1605.06885*, 2016.

[29] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2017.

[30] M. Toussaint, "Newton methods for k-order markov constrained motion problems," *arXiv preprint arXiv:1407.0414*, 2014.