Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

# Vision Assisted Biasing for Robot Manipulation Planning

Puang En Yen

| | |
|---|---|
| **Course of Study:** | Computer Science |
| **Examiner:** | Prof. Dr. rer. nat. Marc Toussaint |
| **Supervisor:** | PD Dr. habil. Rudolph Triebel |
| **Commenced:** | March 19, 2018 |
| **Completed:** | August 22, 2018 |

## Abstract

Sampling efficiency has been one of the major bottlenecks of sampling-based motion planner. Although being more reliable in complex environments, Rapidly-exploring Random Tree for example often requires longer planning time than its optimisation-based counterpart. Recent developments have introduced numerous methods to bias sampling in configuration-space. Gaussian mixture model, in particular, was proposed to estimate feasible regions in configuration-space for low-variance task. Unfortunately this method does not adapt its biases according to individual planning scene during inference. Therefore, this work proposes vision assisted biasing to adapt biases by changing the weights of Gaussian components upon query. It uses autoencoder to extract features directly from depth image, and the resulted latent code is then used for either nearest neighbours search or direct weights prediction. With a modified pipeline, these extensions show improvements on not only the sampling efficiency but also path optimality of simple motion planner.

# Contents

# List of Abbreviations

**AAE** Adversarial Autoencoder. 21

**AE** Autoencoder. 10

**AIMM** Autonomous Industrial Mobile Manipulator. 37

**AUC*f*** Area Under Cumulative Frequency. 45

**CNN** Convolutional Neural Network. 10

**DoFs** Degrees of Freedom. 17

**FGSM** Fast Gradient Sign Method. 21

**GAN** Generative Adversarial Network. 21

**GMM** Gaussian Mixture Model. 9

**KDE** Kernel Density Estimation. 12

**KL** Kullback–Leibler. 20

**KOMO** k-order Markov path Optimisation. 11

**OMP** Optimisation-based Motion Planner. 9

**PCA** Principle Component Analysis. 14

**PRM** Probabilistic RoadMaps. 11

**RGB** Red-Green-Blue. 15

**RRT** Rapidly-exploring Random Tree. 9

**S**$^3$ Semantic Scene Similarity. 41

**SLC** Small Load Carrier. 37

**SMP** Sampling-based Motion Planner. 9

**TCP** Tool Center Point. 17

**VAE** Variational Autoencoder. 13

**WNA** *Weight-Norm-Activate*. 22

# 1 Introduction

In the pursue of Industrie 4.0, industrial robots nowadays are embedded with more and more autonomy[1]. Modern industrial robots are able to not just carry out commands but also make decisions in less heavily engineered workspace. For manipulation task in "smart factory", robots are often equipped with vision system and motion planner for understanding the workspace and generating feasible plan in achieving its task. In this context, speed has become an important criteria in judging the feasibility of the system. Hence, efficient combination between perception and manipulation has become one of the major research directions in the field.

Sampling-based Motion Planner (SMP) is known for its extraordinary capacity in solving planning problem in complex environments. This is an advantage over Optimisation-based Motion Planner (OMP) which can get stuck in local minimals. Rapidly-exploring Random Tree (RRT), a type of SMP, however, still take considerable amount of time to find a solution as it searches the complete configuration space ($\mathbb{C}$-space), which is unnecessary in many robotic applications. Inefficient random sampling especially in high dimensional $\mathbb{C}$-space is a very common problem faced by almost all SMP, and it becomes more pronounced when motion constraints are involved.

Among the efforts in improving sampling efficiency of SMP, repetition sampling[2] proposed a 2-stages $\mathbb{C}$-space biasing method. With bounded variance, it first collects a pool of paths for a particular task or a set of similar tasks. Gaussian Mixture Model (GMM) is then used to estimate the configuration density across $\mathbb{C}$-space from the collection. Each Gaussian components represent a task relevant region in $\mathbb{C}$-space. The first stage of biasing involves sampling from within these regions, while the second stage involves the ranking or weighting among Gaussian components. When queried, a sample is drawn by first sample one Gaussian component according to their weights before having a Gaussian sampling within the chosen Gaussian component.

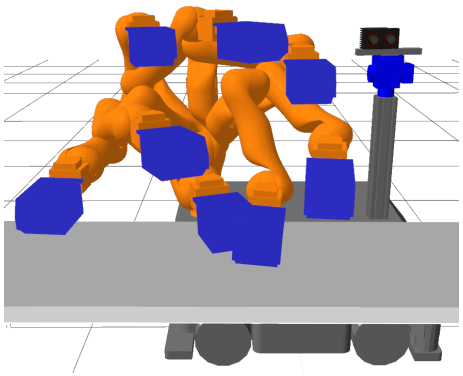


**Figure 1.1:** Joints configurations at the mean of Gaussian components for table-top pick-place task.

Repetition sampling offers a task-oriented, and one-size-fits-all $\mathbb{C}$-space biasing method. It does not update or adapt its bias according to situation of query, hence involve no vision system that aims to improve the performance of motion planner. In this work, a combination

of vision system and motion planner is proposed specifically to improve the applicability of repetition sampling. The objective is to further improve the sampling efficiency of repetition sampling by introducing a third stage of biasing using vision that adapts the bias according to planning scene, enhancing the overall performance.

The vision system involves an Autoencoder (AE) that is trained to understand depth image directly. Its encoder transforms a depth image from a depth sensor into latent code, a lower dimensional representation of the planning scene. With this latent code, efficient search can be carried out in database. The solutions of the similar looking scenes retrieved from the search are then used to estimate a new set of weights for all Gaussian components, re-rank them so that they fit to the queried scene.

Nonetheless, vanilla AE has limited capacity in the task, and original repetition sampling is inefficient in the fitting of GMM. Below are the main contributions of this work in providing a better SMP for industrial robot manipulations:

1. A new way of building database and fitting GMM is proposed to reduce redundancies in collected paths and configurations, hence improving the optimality of solution.
2. To improve the accuracy of nearest neighbour retrieval, several techniques have been proposed to improve the quality of the information encoded in latent code.
3. With the improved repetition sampling and nearest neighbours search, an aggregation method is proposed to output a new set of Gaussian components' weight.
4. A simplified alternative is proposed to speed up the process by predicting the weights distribution directly, hence skipping the database search.

Chapter 2 continues by discussing the literature reviews for both motion planning and Convolutional Neural Network (CNN). Chapter 3 discusses the basic of motion planning and CNN, while Chapter 4 discusses the methodology used in biasing a sampling. Lastly, Chapter 5 demonstrates the experiments' results and discusses their implications.

# 2 Related Work

## 2.1 Motion Planning

Robot motion planning can be formulated as a search problem for a sequence of configurations that fulfil multiple constraints. These constraints include target pose difference, obstacle clearance, kinematic and dynamic limits etc. OMP handles these by explicitly formulating these constraints into cost functions and solves it iteratively until convergence. OMP local method, although finds sub-optimal solution, is able to do so efficiently even in higher dimensionality. Differential Dynamic Programming[3], iterated Linear Quadratic Gaussian[4] and Approximate Inference COntrol[5] are examples that use dynamic programming and Bayesian inference method. Popular OMP library for robotics includes TrajOpt[6] that uses sequential convex optimisation procedure, and k-order Markov path Optimisation (KOMO)[7] that uses classical methods such as Gauss-Newton and augmented Lagrangian.

Although being known as fast and less susceptible from the curse of dimensionality, local methods can get stuck in local optimal, and unable to work in complex situations. As an alternative for cost gradient information, SMPs were introduced to solve complex problem using random samplings. Probabilistic RoadMaps (PRM)[8] and RRT[9] are examples of SMP that plan by building traversable graph using random sampling in $\mathbb{C}$-space. Despite being robust in complex situation, the major disadvantages of SMP are the extra time needed for graph/tree building, and the post-processing needed for local optimisation.

## 2.2 Biased Sampling

To reduce the planning time of SMP, RRT-connect[10] was proposed to reduce tree building time by having multiple trees growing toward each other at the same time. In reducing time for post-processing, RRT*[11] was proposed to improve the optimality of path by rewiring the nodes and edges of the tree while it is growing. [12] proposed extension in expanding tree by reducing Voronoi region of existing vertices in order to achieve better exploration. However, these improvements do not really solve the major bottleneck i.e. sampling efficiently in $\mathbb{C}$-space.

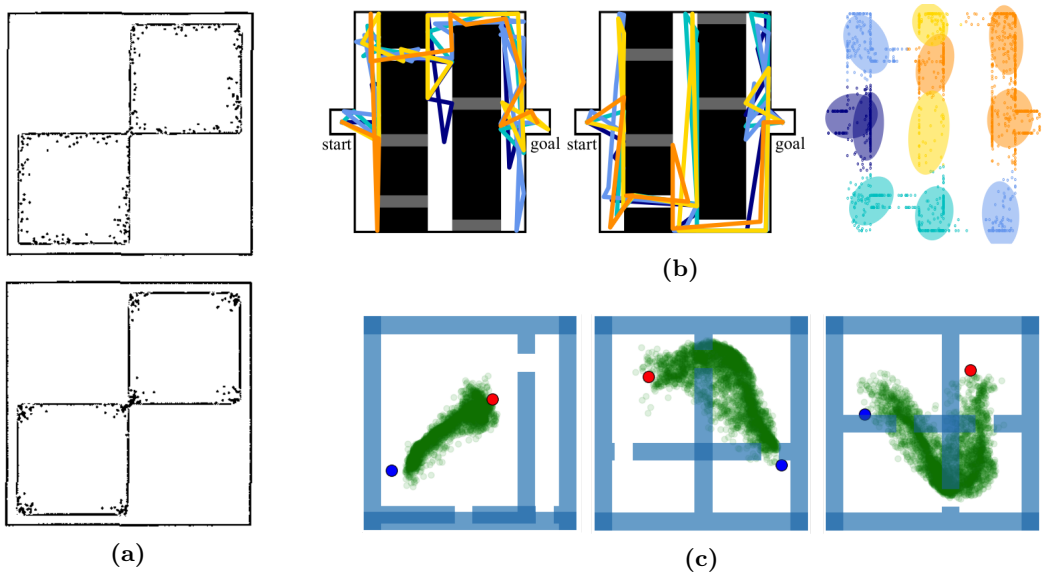



**Figure 2.1:** Examples of 2D $\mathbb{C}$-space biasing. Sampling are biased toward (a)boundary or obstacle [13, 14]©2003 IEEE, (b) task-space heat-map[2]©2017 IEEE, and (c) generated samples by sampling in latent code of VAE[16]©2017 IEEE.

### 2.2.1 $\mathbb{C}$-space Oriented Bias

[13, 14] proposed to bias sampling by only accepting samples that are close to obstacle or boundary. These samplers are more informative compared to those located at wide empty spaces, and therefore improve sampling density for difficult narrow passages while maintaining a smaller graph. [15] proposed to solve the same low sampling density problem using quite the opposite. By approximating the medial axis of $\mathbb{C}$-space, samples drawn around it are away from obstacle and boundary, at the same time sparse at open free space. These methods are easy to implement in 2D-space but not anymore straight forward in high dimensional $\mathbb{C}$-space as the definition of closeness becomes more complex.

### 2.2.2 Fixed Task-space Oriented Bias

[2, 17] proposed to compute the feasible region in $\mathbb{C}$-space for a repetitive and low-variance task. It is done by first collecting a pool of demonstrated paths and/or previous planner's solutions, and then, based on the collection, compute the $\mathbb{C}$-space heat-map using GMM and Kernel Density Estimation (KDE) respectively. The advantage of using GMM over KDE is that during inference GMM need not to select a kernel for drawing samples, which is an important hyperparameter for KDE-based biases.

[18, 19] proposed biased raodmap planner by introducing cost on every edges in the graph that was built using the same path collection. A discrete planner A*[20] is then used to compute the optimal path with minimal cost within the graph when queried. Reinforcement learning has also been introduced in sampling biasing. [21] proposed to first extract feature

from discretised workspace using discrete planner Dijkstra's algorithm[22]. Then Policy Gradient is used to updates a set of weights that make up the workspace bias distributions, using shorter planning time as the reward.

The clear disadvantage among these methods is the descretisation of $\mathbb{C}$-space. In contrast [2, 17] run in continuous $\mathbb{C}$-space and learn the bias distribution directly from data without using workspace feature. Our work build on top of [2], and give it the ability to adapt bias according to situations to enhance the effect.

### 2.2.3 Adaptive Task-space Oriented Bias

Instead of having medial axis that stay away from obstacles, [23] proposed to compute an auxiliary path based on Euclidean distance to goal on discretised $\mathbb{C}$-space, using discrete planner similar to [21]. Then the sampling are biased toward the auxiliary path for that query. Contrary to our work biases are suggested in the form of mixture of Guassians in continuous $\mathbb{C}$-space without additional planner.

[16] proposed a deep learning solution by making use of both the collection of paths, as well as the current query into biasing. Conditional Variational Autoencoder (VAE)[24] is used to encode solution path into latent code with defined distribution, conditioned on the planning scene. During inference, new sampling is generated simply by sampling on latent code distribution and pass it down to the decoder, which was trained to reconstruct solution given latent code and a planning scene. This new method offer promising algorithm but did not address cases in which the planning scenes are in high dimension i.e. image.

To make full use of previously collected or generated solutions for new queries, [19, 25] proposed frameworks to interact with database for efficient planning, retrieval and repair of path. While the Planning-From-Scratch module runs in parallel for exploration, its Retrieve-Repair module searches for suitable paths in the database based on task similarity (proximity of start and end points) and adaptability (amount of collision and constraint violation) in query scene. These works offered good algorithms that addressed the problem well but lacking completeness in practicality of data retrieval. Our work improves the database search with more efficient and informative scene description which will then improve the efficiency of Repair.

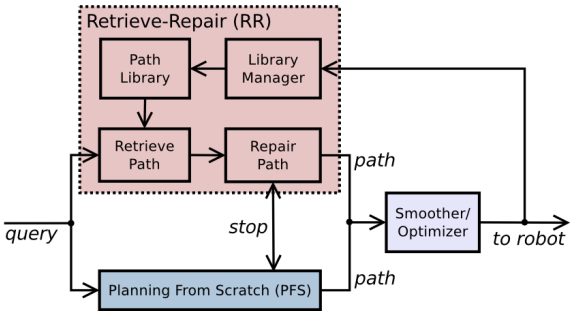



**Figure 2.2:** Lighting framework[25]©2012 IEEE.

Repair and adapt for retrieved path is the second major part of using database. [26] proposed to blend several segments of path retrieved from database into a smooth one. [27] proposed to penalise a squared difference term in its inverse kinematic in order to stay within the proximity of retrieved path. This can be seen as optimising the null-space of

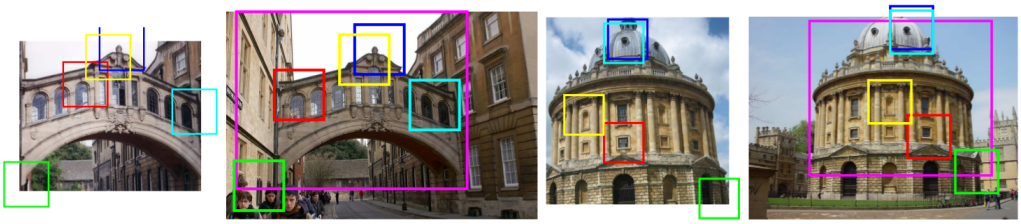


**Figure 2.3:** Localised feature extraction and matching[31].

inverse kinematic with preferred configurations. These methods have similar approach with the one proposed in our work, the differences lay in the final path optimisation used for smoothing.

## 2.3 Similarity Search

The key of using path database is the ability to retrieve relevant data when queried. Simply comparing the differences between start and goal points is insufficient due to the possible appearance and disappearance of obstacle in queried planning scene. [28] proposed to represent the 3D workspace with voxel grid, and then compress it using Principle Component Analysis (PCA) to lower dimensional descriptor. Similarity score is computed with a similarity function whose weight are optimised by minimising the effort required to adapt retrieved path to query scene. Our work does not involve hand-crafting feature for training descriptor in representing planning scene.

Classical hand-crafted feature descriptor for 2D image or 3D point cloud include Scale-Invariant Feature Transform(SIFT), Speeded-Up Robust Feature(SURF), Histogram of Oriented Gradient(HOG) and Fast Point Feature Histograms(FPFH), and they are often used together with Support Vector Machine(SVM), Bag-of-Words(BoW) and Fisher Vector for predictions. These methods make-up low-level representations but lack the ability to represent high-level semantic concept. For better robustness and computational efficiency, the alternative for scene descriptor and similarity measure is deep learning method. [29, 30] show the competitiveness of properly trained CNN compared to classical methods in image classification, object detection and visual retrieval tasks.

### 2.3.1 Feature Extraction

Pre-trained networks trained on large dataset i.e. [32] are popular when it come to feature extraction. Many use it for extracting low and intermediate level features for high level tasks that are different from what the network was originally trained for. According to applications, features can be extracted from different layers with different sizes. Weighted Sum-pooling[30] and spatial Max-activation of Convolutions[31] are some examples of direct features usage. [30, 33] proposed to further compress extracted features with PCA. To acquire detection and localisation ability, features can also be extracted from patches

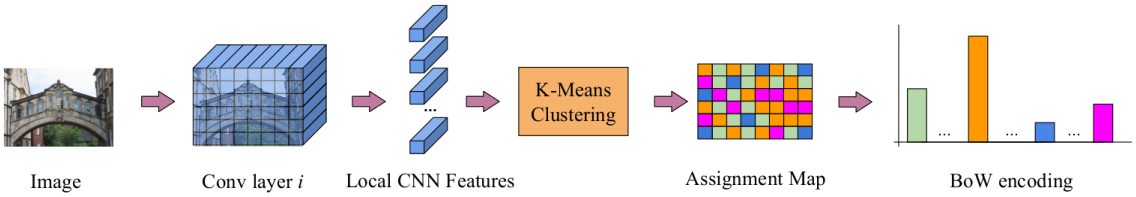



**Figure 2.4:** Bag of Local Convolutional Features pipeline[44]©2016 ACM.

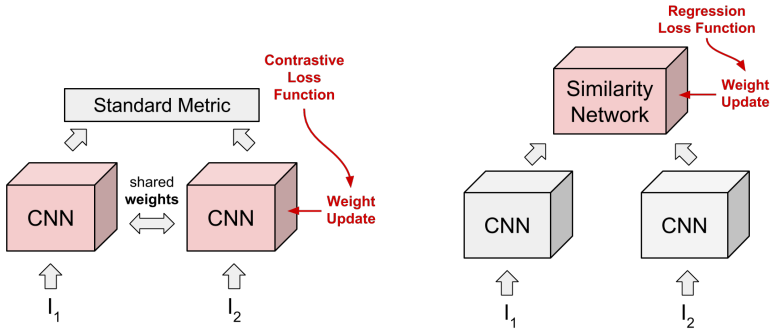
of single input image[31, 34]. In this case detailed features at specified locations can be extracted more accurately. To improve the efficiency in searching in large database, [35, 36] proposed to fine-tune features and project them into binary form. The main drawback of using pre-trained network is that the exact network must be reimplemented and they are usually very large and offer limited architecture flexibility. Our method trains a relatively smaller network from scratch with full control in its architectural design.

AE is another popular way of extracting features. It is done by using latent code as low dimensional representation of the input. To improve robustness, heavy data augmentations are normally used to keep the AE invariant to certain influences such as background, lighting and occlusions. [37, 38, 39] are examples that proposed to encode image patches for object pose estimation. Using the information aggregated at its bottleneck, [40] proposed an encoder-decoder architecture for scene understanding by doing semantic segmentation with decoder's output. [41] proposed an affordance detection for objects in tabletop scenario. It uses an Encoder-Decoder architecture, using Red-Green-Blue (RGB) and HHA[42] encoding for depth data as inputs, and VGG16-based network[43] to produce softmax of affordance types pixel-wise. The same idea of aggregating information in latent code is adopted in our method, but applied without HHA depth image encoding.

[34, 44] proposed Bag of Deep Local Convolutional Features for visual retrieval task. After gathering features extracted from images and patches of images, K-Means clustering is used to quantise features in continuous space into distinct clusters, represented by respective cluster center. Feature of an input is then the histogram that contains the frequencies of occurrence of each clusters. Figure 2.4 depicts the feature extraction pipeline. These methods have good accuracy in image retrieval task but lack the real-time performance required in robotic application. Our method uses entirely CNN that compute latent code in a single forward-pass.

### 2.3.2 Similarity Learning

Cosine similarity is a method for measuring the directional similarity between vectors. Due to the normalisation by vectors' magnitude, it is usually more preferable than L2-Norm to be used in comparing high dimensional deep feature vectors or histograms. To improvement retrieval performance, [31, 44] proposed a 2 stages ranking refinement that uses cosine similarity in first retrieving nearest images, and then re-rank them according to regional similarities. For fast retrieval, our method does not refine the ranking because regional details are not very important in our application.




**(a)** Siamese network **(b)** Similarity network

**Figure 2.5:** Differences between 2 architectures used in image similarity prediction task[45]©2017 IEEE.

The limitation of using standard similarity measures is that it lacks the ability to model non-linear dependencies, hence unable to map high-level visual appearances with human-level semantic concepts. One way to close this *semantic-gap* is to project deep feature into linear-space where standard metric can be useful in calculating linear distance. Siamese[46] was proposed to fine-tune the feature extraction network so that the similarity energy/L1-loss is lower for features came the same class. Based on the same idea, Triplet architecture[47] proposed to maintain a margin in squared Euclidean distance between features of positive-positive pair and positive-negative pair. These methods require paired training data with defined relationship which is not applicable in general similarity search.

[48, 49] proposed an end-to-end methods that map paired-input images into probability of them being in the same class. It is done by simply having an binary/softmax layer concatenated to the 2 feature extraction in the form of Siamese architecture. For image retrieval task, [45] proposed to have a continuous output that predict the cosine similarity between input pair. Besides the requirement of paired inputs, these methods do not transform input into smaller representations and hence, in our application, require run through the entire database for each query.

# 3 Background

## 3.1 Sampling-based Motion Planner

Configuration-space, or $\mathbb{C}$-space, is defined by the Degrees of Freedom (DoFs) of robot. One configuration point $\mathbf{q}$ in $\mathbb{C}$-space, in this case, has $J$ dimensions, representing $J$ different joints in the robot with respective joint limits. A path $\mathcal{T}$ with $T$ steps is a sequence of $T$ configuration points.

$$\begin{aligned} \mathbf{q} &= \left[q^1, \cdots, q^J\right]^{\mathrm{T}} && \in \mathbb{R}^J \\ \mathcal{T} &= \left[\mathbf{q}_0, \cdots, \mathbf{q}_T\right]^{\mathrm{T}} && \in \mathbb{R}^{T \times J} \end{aligned} \tag{3.1}$$

A valid path is a path that is able to link its initial configuration $\mathbf{q}_0$ with its target, while respecting constraints $\mathcal{C}$ in kinematics and dynamics. Target $\mathrm{T}_{\mathfrak{q}} \leftarrow \mathbf{q}_T$ can be defined in many forms, in this case it is defined as the 6D pose of end-effector/Tool Center Point (TCP). Figuring out the $\mathbf{q}$s in $\mathcal{T}$ require a planner that searches through $\mathbb{C}_{free}$, which is region in $\mathbb{C}$ obtained after excluding collision $\mathbb{C}_{obs}$ between robot $\mathcal{R}$ and obstacles $\mathcal{O}$, as well as fulfilling all constraint $\mathbb{C}_{con}$.

$$\begin{aligned} \mathbb{C}_{obs} &= \left\{ \mathbf{q} \in \mathbb{C} \,\middle|\, \mathcal{R}(\mathbf{q}) \cap \mathcal{O} \neq \emptyset \right\} \\ \mathbb{C}_{con} &= \left\{ \mathbf{q} \in \mathbb{C} \,\middle|\, \mathcal{R}(\mathbf{q}) \in \mathcal{C} \right\} \\ \mathbb{C}_{free} &= (\mathbb{C} \setminus \mathbb{C}_{obs}) \setminus \mathbb{C}_{con} \end{aligned} \tag{3.2}$$

SMP have shown promising performances in searching in complex environments. In contrast to OMP which has explicit modelling of $\mathbb{C}_{free}$, SMP performs collision check and constraint projection only upon sampling. RRT in particular is a type of incremental SMP in which $\mathbf{q}$s or vertexes are incrementally added into a search tree while the planner exploring $\mathbb{C}$-space. It is made out of several components:

- `RAND_CONF`($env$, $\mathcal{C}$) performs collision in $env$ and random sampling in $\mathbb{C}$-space uniformly across all dimension, followed by $\mathcal{C}$ constraint projection.
- `NEAREST`($\mathcal{G}$, $\mathbf{q}$) search for the nearest vertex in tree $\mathcal{G}$ to point $\mathbf{q}$, using algorithms such as kd-tree with suitable distance metric.
- `LOCAL`($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$) is a local planner that perform simple interpolation in $\mathbb{C}_{free}$ between 2 points.
- `ARRIVED`($\mathcal{G}$, $\mathrm{T}_{\mathfrak{q}}$) is a distance metric that measures the proximity between vertexes of tree $\mathcal{G}$ and target pose $\mathrm{T}_{\mathfrak{q}}$.

- `CONNECT`($\mathbf{q}_0$, $\mathbf{q}_T$) traces all edges that link $\mathbf{q}_0$ and $\mathbf{q}_T$, then return the respective list of vertexes.
- `RAND_SHORTCUT`($\mathcal{T}$) looks for new edges by applying `LOCAL` randomly among $\mathbf{q} \in \mathcal{T}$ to shorten $\mathcal{T}$
- `SMOOTHING`($\mathcal{T}$) smooths $\mathcal{T}$ with joint-space interpolation.

RMPL[1] is used here as the RRT library : $env \times task \rightarrow \mathcal{T}$, where *env* is robot workspace/obstacle definition. In this work, *env* contains the 3D model and 6D pose of all objects, hence the planner has full knowledge of the workspace geometry. Whereas *task* consists of the initial joint values $\mathbf{q}_0$, TCP target pose $\mathrm{T}_\mathfrak{q}$, kinematic constraint $\mathcal{C}$ and allocated time for planning $t_{plan}$ and optimisation $t_{opt}$. Algorithm 3.1 depicts the algorithm of RRT[1].

**Algorithm 3.1** Rapidly-Exploring Random Tree (RRT) motion planner[1]. *env* is robot workspace definition. $\mathbf{q}_0$ and $\mathrm{T}_\mathfrak{q}$ are initial joint values and requested/query TCP target pose. $\mathcal{C}$ is kinematic constraint on robot. $\mathcal{G}$ is the search tree consists of vertexes and edges. $t_{plan}$ and $t_{opt}$ are the time allocated for path planning and optimisation.

**procedure** RRT($env$, $\mathbf{q}_0$, $\mathrm{T}_\mathfrak{q}$, $\mathcal{C}$, $t_{plan}$, $t_{opt}$)
  $\mathcal{G}$.init($\mathbf{q}_0$)
  **repeat**
    **if** time $> t_{plan}$ **then**
      **return** $\emptyset$
    **end if**
    $\mathbf{q}_{rand} \leftarrow$ `RAND_CONF`($env$, $\mathcal{C}$)
    $\mathbf{q}_{near} \leftarrow$ `NEAREST`($\mathcal{G}$, $\mathbf{q}_{rand}$)
    $\mathbf{q}_{new} \leftarrow$ `LOCAL`($\mathbf{q}_{near}$, $\mathbf{q}_{rand}$)
    $\mathcal{G}$.`add_vertex`($\mathbf{q}_{new}$)
    $\mathcal{G}$.`add_edge`($\mathbf{q}_{near}$, $\mathbf{q}_{new}$)
  **until** `ARRIVED`($\mathcal{G}$, $\mathrm{T}_\mathfrak{q}$)
  $\mathbf{q}_T \leftarrow$`NEAREST`($\mathcal{G}$, $\mathrm{T}_\mathfrak{q}$)
  $\mathcal{T} \leftarrow$ `CONNECT`($\mathbf{q}_0$, $\mathbf{q}_T$)
  **while** time $< t_{opt}$ **do**
    $\mathcal{T} \leftarrow$ `RAND_SHORTCUT`($\mathcal{T}$)
    $\mathcal{T} \leftarrow$ `SMOOTHING`($\mathcal{T}$)
  **end while**
  **return** $\mathcal{T}$
**end procedure**

[1] Robot Motion Planning Library (RMPL) is a software used within DLR Robotics and Mechatronics Center: https://wiki.robotic.dlr.de/Rmpl

## 3.2 ℂ-Space Biasing

[2, 17] have proposed methods to bias the sampling in ℂ-space to improving sampling efficiency and the overall execution time. Both methods start by building a database $\mathcal{D}$ with paths generated for a particular repetitive task domain. Key-configuration points $Q$s are then extracted from the database $\mathcal{D}$ for building biasing mechanism.

$$\begin{aligned} \mathcal{D} &= \left[\mathcal{T}_0, \cdots, \mathcal{T}_m\right]^{\mathrm{T}} && \in \mathbb{R}^{m \times T \times J} \\ \mathbf{Q} &= \left\{ \Psi\left(\mathcal{T}_i\right) \,\middle|\, i \in m \right\} && \in \mathbb{R}^{\sum_i^m |\Psi(\mathcal{T}_i)| \times J} \end{aligned} \tag{3.3}$$

**Kernel Density Estimation** In [17], all $\mathcal{T}$ in $\mathcal{D}$ are first optimised using $\Psi[\cdot]$ which interpolates $\mathcal{T}$ into denser $Q$s and adds all of them into $\mathbf{Q}$. KDE $\mathcal{K}$ is used to determine regions of workspace where feasible paths are more likely to happen, according to the distribution in $\mathbf{Q}$. The kernel used is a multivariate Gaussian $\mathcal{N}$, with $\mathbf{0}$ mean $\mu$ and uncorrelated covariance $\Sigma = \mathbf{I} \cdot \sigma^2$ where $\sigma^2$ is the variance on dispersity of $\mathbf{Q}$.

$$\begin{aligned} \mathcal{N}(Q \,|\, \mu, \Sigma) &= \frac{1}{\sqrt{(2\pi)^J \|\Sigma\|}} \exp\left( -\frac{(Q-\mu)^{\mathrm{T}} \Sigma^{-1} (Q-\mu)}{2} \right) \\ \mathcal{K}(Q) &= \frac{1}{M} \sum_{i=1}^{M} \mathcal{N}\left( \frac{Q - Q_i}{h} \right) \end{aligned} \tag{3.4}$$

**Gaussian Mixture Model** In repetition sampling[2], $\mathcal{T}$ are not smoothed and $\Psi[\cdot]$ merely takes the raw vertexes $Q$ and adds then into $\mathbf{Q}$. With the similar concept of estimating regions with higher feasibility, GMM is used to approximate density distribution in $\mathbf{Q}$. It uses Expectation-Maximisation algorithm to estimate the mean $\mu_k$, covariance $\Sigma_k$ and weight $w_k$ of each Gaussian components $g \in G$ where $G$ is the number of components.

Expectation:

$$p_{i,g} = \frac{w_g \, \mathcal{N}(Q_i \,|\, \mu_g, \Sigma_g)}{\sum_j^G w_j \, \mathcal{N}(Q_i \,|\, \mu_j, \Sigma_j)} \tag{3.5a}$$

Maximisation:

$$\mu_g = \frac{1}{n_g} \sum_i^{|\mathbf{Q}|} p_{i,g} \, Q_i \qquad\qquad w_g = \frac{n_g}{|\mathbf{Q}|} \tag{3.5b}$$

$$\Sigma_g = \frac{1}{n_g} \sum_i^{|\mathbf{Q}|} p_{i,g} \, (Q_i - \mu_g)(Q_i - \mu_g)^T \qquad\qquad n_g = \sum_i^{|\mathbf{Q}|} p_{i,g} \tag{3.5c}$$

Both methods, during inference, select a Gaussian kernel/components to draw samples from. Moreover, repetition sampling utilises the weights of each components in GMM as its second biasing mechanism. The chances of a component being selected for sampling is proportional to its weight $w_g$, therefore providing a feasibility ranking in ℂ-space.

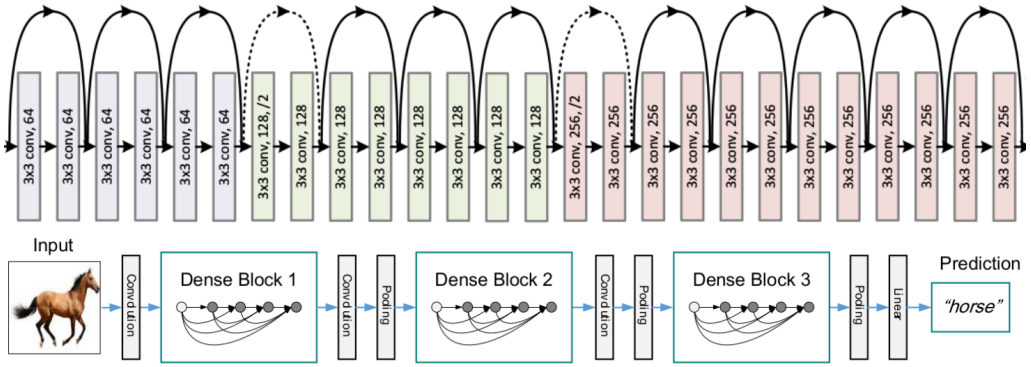



**Figure 3.1:** Truncated ResNet-34 and DenseNet[53, 54]©2016 IEEE. The major different between these two architectures is that Resnet's skipped connections ended with addition to the new channels; DenseNet's skipped connections ended with cascade to the new channels and therefore the number of output channel increases without explicitly increasing the number of parameter.

## 3.3 Convolutional Neural Network

Over the years, there have been many modifications made since LeNet[50], an application of CNN, was applied to recognise digits. AlexNet[51] revolutionised computer vision community by bringing Multi Layer Perceptron back to life. GoogLeNet[52] improved efficiency by using inception structure and bottleneck. ResNet[53] mitigated vanishing gradient and proved that number of layers is somehow proportional to performance. These are examples of contributions that have made machine surpassed human level in ImageNet[32]. To further improve efficiency, DenseNet[54] proposed feature reuse by having densely connected CNN and have shown to achieved the state of the art with lesser parameters. Figure 3.1 depicts a comparison between ResNet and DenseNet.

**Convolutional Autoencoder** AE was first introduced by [55] and it was used for transforming high dimensional input into low dimensional latent code representation. Convolutional AE combined CNN and AE for transforming images. Modifications made on top of AE are mainly to acquire various invariance properties and avoid over-fitting. Regularized AE proposed to impose weight decay which favours small weights to avoid over-fitting. Denoising AE [56] proposed to add random noise to the input image while maintaining a clean reconstruction target, so is to train an encoder that is robust to wider range of input. Contractive AE [57] on the other hand explicitly minimise the gradient of latent code with respect to input to achieve the same robustness.

VAE [24] proposed to impose prior distribution on latent code by minimising Kullback–Leibler (KL) divergence between latent code and a prior distribution of choice. To be able to back-propagate the KL divergence loss, the exact functional form of the prior distribution need to be accessible. Therefore, simple prior distribution e.g. unit Gaussian is always preferred than complex distribution e.g. Swiss roll.

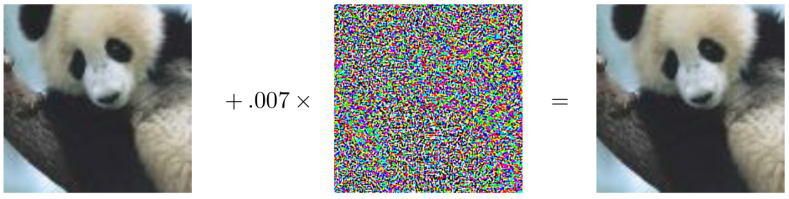



**Figure 3.2:** Before the attack the classifier was correct with 57%. After adding some very subtle noise (adversarial attack) that is imperceptible by human, the classifier changed it prediction to a wrong class with very high confidence[58].

**Generative Adversarial Network** Generative Adversarial Network (GAN) [59] was first introduced to train generative model in unsupervised fashion e.g. [60]. Traditional generative model such as directed or undirected graphical models often involve intractable computation and require Markov chains. GAN is computationally more efficient by just training a generator-discriminator pair. Moreover, the model design is also simplified since GAN takes in theoretically any differentiable function. Its discriminator is trained to differentiate between real and generated data point, whereas its generator is trained to generate realistic data and confuses the discriminator.

**Adversarial Autoencoder** Adversarial Autoencoder (AAE) [61] proposed to combine AE and GAN. It has a discriminators, a decoder and an encoder which also acts as a generator. The generator-discriminator pair impose prior distribution to latent code, while the encoder-decoder pair does dimension reduction and input reconstruction. Exact functional form of prior distribution is no longer needed for back-propagation because the discriminator acquires positive and negative samples directly without penalising KL divergence. Dimensionality handled by the generator-discriminator pair is also smaller e.g. $\mathbb{R}^{128}$ latent code compare to images e.g. $\mathbb{R}^{128\times128}$, which makes the training of AAE easier than image generating GAN.

**Adversarial Examples** Adversarial examples was first introduced as a reliability and security threats for models trained with machine learning techniques. [62] and many other have proved that neural networks and other techniques are highly vulnerable to adversarial attacks. Adversarial attacks refers to techniques that generate slightly perturbed input which then causes huge and destructive changes on the output. Figure 3.2 depicts an example of the effect of adversarial attack on an image classifier.

[58] proposed Fast Gradient Sign Method (FGSM) as a way to generate adversarial examples. [63] proposed an algorithm to train robust model against adversarial examples with variant of attacks such as iterative and least-likely FGSM. [64] proposed to add noise to input data before performing FGSM as to better estimate the local gradient of a data point.

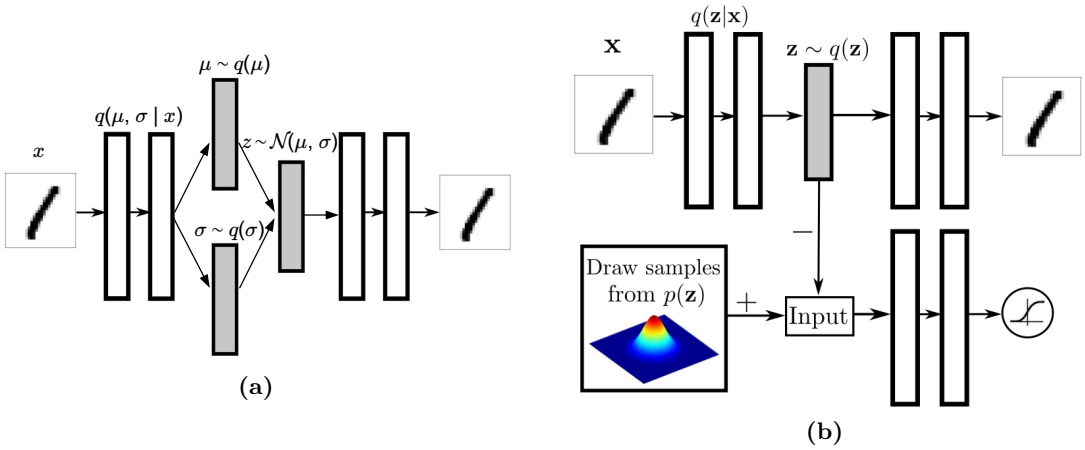



**Figure 3.3:** (a) VAE and (b) AAE architecture [61].

## 3.4 Autoencoder Dimension Reduction

AE consists of an encoder and a decoder. The encoder Enc: $x \to z$ transforms input $x \in \mathbb{R}^I$ into latent code $z \in \mathbb{R}^L$, where $I \gg L$. Whereas the decoder Dec: $z \to x$ does the inverse by transforming latent code back to input. This configuration requires the bottleneck latent code in much lower dimensions to encapsulate all essential information needed to reconstruct the input. Convolutional AE normally consists of multiple combinations of *Weight-Norm-Activate* (WNA). *Weight* is a layer of trainable parameters such as Convolutional *Conv* and Fully Connected layer $fc$. *Norm* is Batch Normalisation function. *Activate* is non-linear activation function such as Rectified Linear Unit and Sigmoid.

Classical AE that relies only on minimising reconstruction loss $\|x - \text{Dec}(\text{Enc}(x))\|$ often maps inputs into distinct locations in latent space. This rather random latent code scattering causes 'holes' in latent space where decoder has never been trained. As the result, latent codes become meaningless outside a particular encoder-decoder pair, and sampling in latent space becomes harder[65].

[24] proposed to regularise the distribution of latent space into Unit-Gaussian by first transforming input into latent mean and variance $\text{Enc}(\mu_z, \sigma_z \,|\, x)$, then enforces Unit-Gaussian distribution using KL divergence. [61] enforces latent space distribution with adversarial discriminator $D$. This additional binary classifier $D : z \to [0, 1]$ is trained to distinguish between samples from target distribution and latent code generated from encoder. Figure 3.3 depicts the architecture of VAE and AAE. [66] proposed to use maximum mean discrepancy with radial-basis-function and inverse multi-quadratics kernel to regularise latent code's mean within each mini-batch.

# 4 Methodology

The processes pipeline consists of 3 major segments: recording, training and inference, as depicted in Figure 4.1. On top of the original pipeline from repetition sampling in [2], vision modules are added in training and inference segments.

**Recording** Using a simulated environment, thousands of simulated planning scenes from the targeted workspace scenario are created. The respective RGB and depth images are paired with low variance paths generated by $\overline{\text{RRT}}$ for multiple pick and place tasks, and stored in the database.

**Training** Key-configurations are extracted from the paths database, and they are then used for fitting a GMM. Through the process, assignment probabilities among Gaussian components are computer for each key-configurations. They are then stored in the database together with the GMM definitions.

The AE that is designed to retrieve image from database AE-WA is then trained with only images. While the AE that is designed to predict GMMs' weights AE-WP is trained with both images and assignments of key-configurations.

**inference** Upon query, the encoder from AE-WA is used to encode query scene and the generated latent code is then used to search for $\mathbb{K}$ nearest neighbours in the database. Those assignments from the nearest neighbours are aggregated to produce a new set of weights for GMM, used by repetition sampling in the RRT. The other option is to use the encoder and predictor from AE-WP to predict GMM's weights directly.

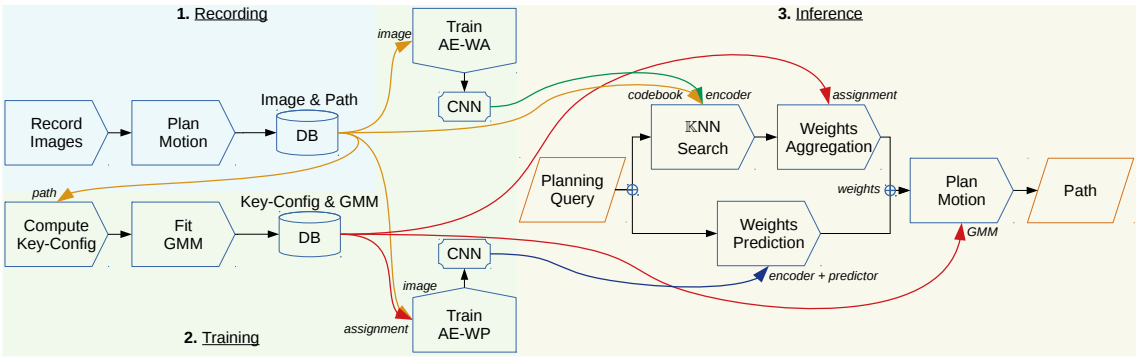



**Figure 4.1:** Processes pipeline.

## 4.1 Database

Database is a collection of scenes. Each scene contains information regarding the workspace, task and planned paths. It provides all information needed for the subsequence processes. Diversity of this database, which is built entirely in simulated world, is designed to reflect scenarios of the targeted real applications e.g. pick and place task for mobile robot.

### 4.1.1 Path Collection

Given a pair of scene and task, a path $\mathcal{T}$ with $T$ step is generated using the RRT planner:

$$\mathcal{T} \leftarrow \mathrm{RRT}(env,\, task) \tag{4.1a}$$

Since RRT planner has no guarantee in its optimality, due to a trade-off between quality and speed, an evaluation metric is introduced to ensure the quality of generated paths collected in the database. Through experiments, it is observed that the biggest sub-optimality came from path variance in Cartesian-space. Reducing variance is especially important for SMP in improving motion predictability and operational safety. The evaluation ,therefore, projects $\mathcal{T}$ and determines whether it can be fitted by a simple straight line on the $xy$ Cartesian-plane. This left TCP's Cartesian z-axis and rotation the only DoFs for obstacle avoidance, which is acceptable in certain type of scene i.e. tabletop.

At first the path time-steps are converted into polynomial features with degree 1, whereas joint values are converted into TCP $xy$ Cartesian-coordinate using forward kinematic $\phi$. Then a ridge regression is formulated and the straight line parameters $\beta$ are solved using pseudo inverse.

$$X = \begin{bmatrix} 1 & t_0 \\ 1 & t_1 \\ \vdots & \vdots \\ 1 & t_T \end{bmatrix} \in \mathbb{R}^{T\times 2} \qquad Y = \begin{bmatrix} \phi(\mathbf{q}_0)^x & \phi(\mathbf{q}_0)^y \\ \phi(\mathbf{q}_1)^x & \phi(\mathbf{q}_1)^y \\ \vdots & \vdots \\ \phi(\mathbf{q}_T)^x & \phi(\mathbf{q}_T)^y \end{bmatrix} \in \mathbb{R}^{T\times 2} \tag{4.1b}$$

$$\min_{\beta} \left\| X\beta - Y \right\|_2^2 + \lambda \left\| \beta \right\|_2^2 \quad \longrightarrow \quad \beta^* = \left( X^T X + \lambda I \right)^{-1} X^T Y \tag{4.1c}$$

$R^2$ is a measure of goodness-of-Fit between data and prediction, range (-$\infty$, 1]. It is used to evaluate path's simplicity by comparing it to the regressed straight line. $\overline{\mathrm{RRT}}$, hence, generates simple paths by rejecting those with $R^2 < \alpha_{\mathcal{T}}$ where $\alpha_{\mathcal{T}} = 0.9$. Figure 4.2 depicts paths with different $R^2$.

$$\overline{\mathrm{RRT}}(env,\, task) = \begin{cases} \mathrm{RRT}(env,\, task) & if \quad R^2 \geq \alpha_{\mathcal{T}} \\ \text{repeat} & otherwise \end{cases} \tag{4.1d}$$

$$R^2 = 1 - \frac{\left\| Y - X\beta \right\|_2^2}{\left\| Y - \mathbb{E}[Y] \right\|_2^2} \tag{4.1e}$$

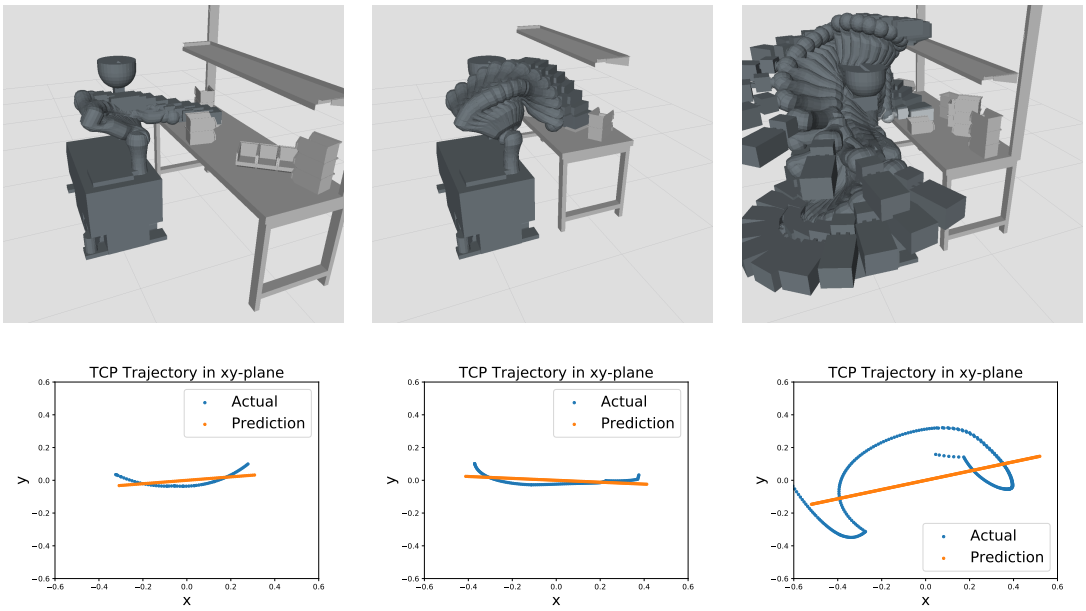


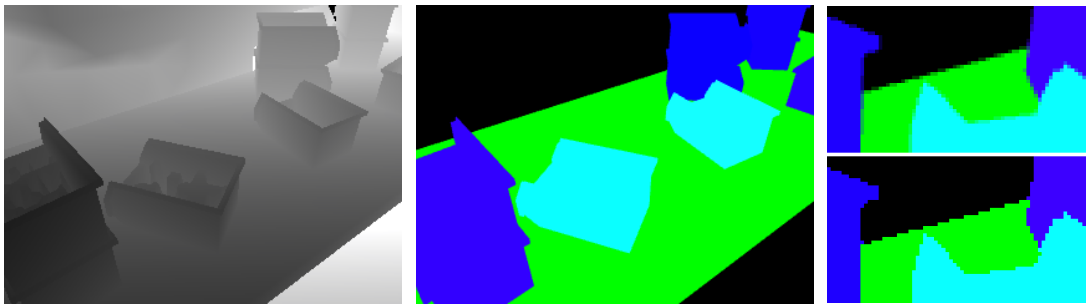

**Figure 4.2:** Comparisons between the simplicities of generated paths for 3 different scenes. Upper row are the paths in Cartesian-space, lower row are the respective comparison between its xy-plane projection and the regressed straight line. From left to right the $R^2$ scores are: 0.96, 0.98 and 0.64.

To further ensure paths collected in database are free from redundancy as much as possible while using the simple RRT, the base axis is frozen at first until the planner consecutively fails for several attempts. This is to make sure that the target is out of reach or block from initial location, and moving the base is the only option to fulfil the task.

### 4.1.2 Scene Collection

Planning scene is perceived from robot's perspective through 2 types of data: RGB and depth images. Both data are produced from a simulated sensor tuned to match with the hardware used in the targeted application. In addition, RGB images are used as free semantic segmentation labelling for every pixel in every scene, which is something depth data alone cannot provide easily. To do so, all objects appear in scene are colour-coded based on their role: green for table, blue for obstacle, aqua for pick-target and black for everything else. Hence, the semantic class size $\rho \leftarrow 4$. To provide more information regarding the identity of individual object, each of them has small colour difference even among the same class.

In order to avoid ambiguity in semantic labelling, RGB images are post-processed by re-assigning distorted pixels, especially those at image edges, to its nearest non-distorted neighbour. Figure 4.3 depicts the raw sensor data and the effect of RGB refinement.



**(a)** Depth image **(b)** RGB image **(c)** Pixel refinement

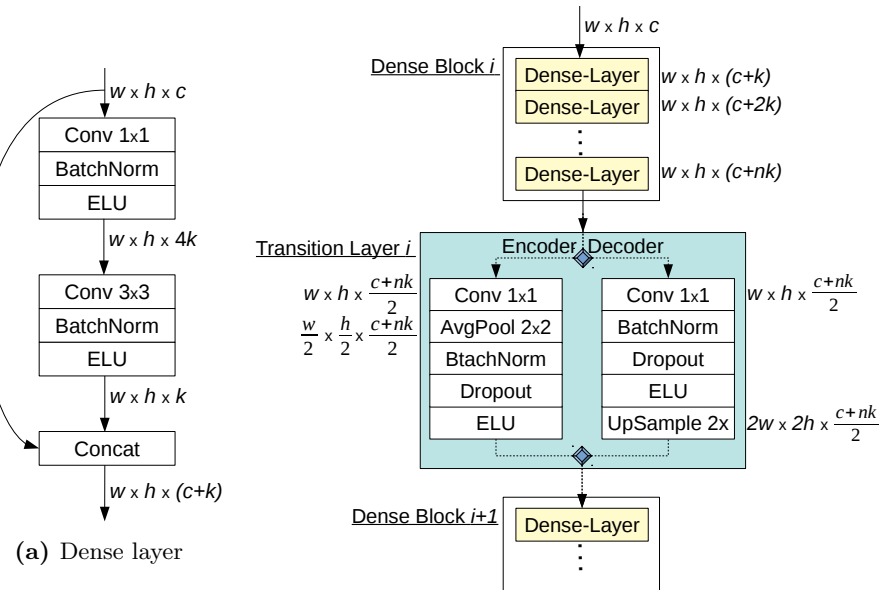
**Figure 4.3:** (a) and (b) are the images captured from robot's perspective, (c) depicts the effect of refinement on RGB image for tabletop scene.

## 4.2 Nearest Neighbours Retrieval

The purpose of having a database is to improve planning time by simply reusing a ready solution when queried. However, the degree of improvement rely heavily on whether the right data is fetched at the right time. AE is proposed for this task, by reducing the dimension of scene in the form of 2D depth image $x \in \mathbb{R}^{w \times h}$, into latent code $z \in \mathbb{R}^{L}$ with length $L$, where $w \times h \gg L$. AE constructed with neural network requires no hand-crafted feature and yet learn to represent high dimensional input using low dimensional representation. Through supervised training, the resulted latent code is a useful descriptor in database. Its small dimension makes efficient database search. Together with fast network forward-pass, the overall planning time of a SMP is expected to improve.

### 4.2.1 Network Architecture

The encoder $\text{Enc}(z \mid x)$ is responsible to, given an input $x$, output a set of continuous latent code $z \in \mathbb{R}^{L}$. Input $x$ refers to depth image that has been scaled to $\mathbb{R}^{64 \times 64}$. With latent code $z$ as input, the decoder $\text{Dec}(\hat{x}^c, \hat{x}^s, \hat{x}^b \mid z)$ produces a clean reconstruction $\hat{x}^c$, semantic segmentation $\hat{x}^s$ and object boundary prediction $\hat{x}^b$ of the input $x$. Since the objective is to have high quality latent code that encode the spatial relation of objects in a scene, there is no skip connections linking between encoder and decoder. All information must go through the only latent code bottleneck. DenseNet[54] architecture is the backbone of this AE and it consists of mainly dense-layers and transition-layers. A dense-layer is a set of WNA combination. This proposed combination is different from the original NAW combination adopted from ResNet[67]. Due to the difference in the construction of skip connection, i.e. addition vs concatenation, the original combination concatenates output of *Weight* and therefore, required to repeatedly computes the same *Norm-Activate* in every subsequence dense-layer. Since *Norm-Activate* make no difference when placed before or after feature concatenation, our proposed combination prevents unnecessary computations by concatenates activated feature directly.




**(a)** Dense layer

**(b)** DenseNet configuration

**Figure 4.4:** (a) is the configuration of Dense layer with bottleneck. (b) depicts the formation of Dense block and the configurations in Transition layer for Encoder and Decoder separately.

A dense-layer produces $k = 40$ new channels, known as growth rate. When the number of channel $c$ of input feature map is large, namely $c \geq 5k$, dense-layer deploys a bottleneck between two WNA combinations. The first WNA's $Weight$ has $1 \times 1$ kernel and serves mainly as dimension reduction by reducing the number of channel to $4k$. While the second WNA generates new $k$ new channels with input channel $c < 5k$. This is different from [54] where bottleneck WNA always exists regardless of the size of input channels. Figure 4.4(a) depicts a dense-layer with bottleneck WNA.

Dense-block is an aggregation of dense-layers whose output are concatenated. To improve perception range, dilated convolution[68] is adopted in every dense-layer in a dense-block. The amount of dilation in each $Conv$ of dense-layer grows from 1 to 3 and repeat until the end of dense-block. This configuration provides a wider range of feature scale dense-block can learn from as it gets deeper.

While dense-layers increase the number of channel in constant spatial size, transition-layer changes the spatial size between dense-blocks. Pooling and up-sampling are used in transition-layers in encoder and decoder to up and down-scale feature map respectively. Besides, transition-layer also does bottleneck dimension reduction with $1 \times 1$ kernel $Conv$ to reduce the number of channel accumulated from the preceding dense-block. Dropout[69] is also performed here, instead of in dense-layers, with drop probability 0.2. Figure 4.4(b) depicts the spatial configuration between dense-layer, dense-block and transition-layer.

Both encoder and decoder has 4 dense-blocks, each contains 2/3/4/5 dense-layers. Decoder is a mirrored version of encoder in most of its configuration. After the $4^{th}$ dense-block in

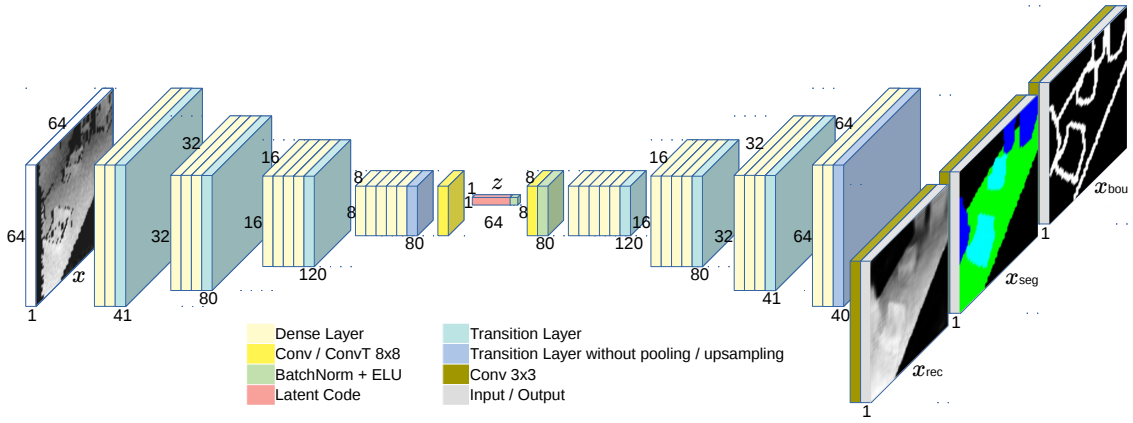


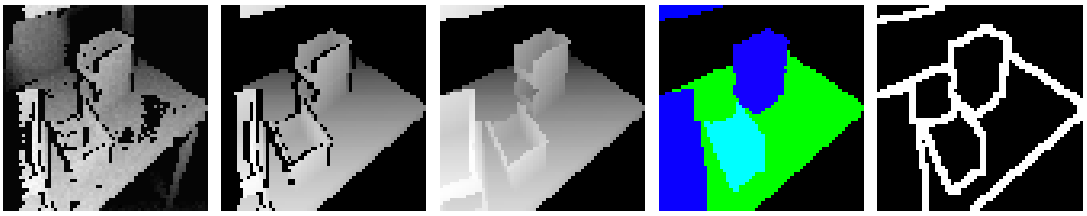

**Figure 4.5:** AE architecture.

encoder, latent code $z$ is produced by a $Conv$ with $8 \times 8$ kernel. The spatial size becomes $1 \times 1$ and the number of channel $c = L = 64$ where $L$ is the length of latent code $z \in \mathbb{R}^L$. Latent code is then fed into decoder following the WNA sequence. The end of decoder consists of 3 output heads with $3 \times 3$ kernel $Conv$ respectively.

### 4.2.2 Data and Augmentations

There are in total 5 types of data used in the training of this network:

- $\boldsymbol{x}$ is depth image the network is expected to receive as the only input. It should be either come directly (after scaling) from depth sensor of real robot during inference, or realistically rendered in simulation during training.
- $\boldsymbol{x^e}$ is embedding input used for latent code embedding in Equation (4.4) during training. It has less data augmentation than $x$, but maintain a sense of realism in its rendering. The discrepancies between $x$ and $x^e$ tell the network which features it should stay invariant of. It is used as input during codebook writing in Section 4.2.5.
- $\boldsymbol{x^c}$ is clean input or reconstruction target used for reconstruction loss in Equation (4.2). It has no data augmentation and contains only related pixels. The discrepancies between $x$ and $x^c$ tell the network which 'objects' it should be paying attention on.
- $\boldsymbol{x^s}$ is semantic segmentation label used for cross-entropy loss in Equation (4.2). It contains $\rho = 4$ semantic classes: tabletop(green), obstacle(blue), target object(aqua) and background/anything-else(black).
- $\boldsymbol{x^b}$ is object boundary label in binary form used for binary-cross-entropy in Equation (4.2). While semantic segmentation provides spatial arrangement among different classes, boundary prediction provides additional distinctions among adjacent objects with the same semantic class.



**(a)** $x$ **(b)** $x^e$ **(c)** $x^c$ **(d)** $x^s$ **(e)** $x^b$

**Figure 4.6:** Types of data used in training.

Since the entire training involves no real data (from real sensor), realism and diversity of synthetic training data is curial for generalisation in real-world scenario. Training data generation should takes into account the hardware properties of the targeted application. Here, for example, the properties of depth image acquired by passive stereo camera is used as realism standard and artificially added as mush as possible into synthetic data.

- **Baseline Shift**. The first property is the blind region at image horizontal boundary caused by cameras baseline shift. This blind region grows as object's distance shorten. To implement this synthetically, left most columns are nullified according to the depth values of its' left most pixel in respective rows.
- **Depth Shadow**. Occlusion is another property that defines stereo vision. To implement it, depth shadow is added right next to edges in depth image. Pixels are nullified in the opposite direction of horizontal image gradient with amount proportional to gradient magnitude. Since the main camera is the one on the left, shadows can only be chased to the left where right camera cannot see.
- **Homogeneous Dropout**. One of the major obstacle for passive stereo camera is homogeneous region. Features cannot be extracted from texture-less region which causes feature-matching to fail. To simulate failure in stereo-matching, pixels are randomly nullified according to their distance to nearest image edge with gradient larger than certain threshold. Perlin[70] noise is used to mask the homogeneous region so that the nullification looks more realistic. Figure 4.7 depicts each step of augmentation for depth image.

Techniques such as flip, in-plane rotation and affine-transformation are not suitable as they change the spatial labelling of training data in this case. Below are some other image augmentation techniques used for depth image.

- **Crop & Inverse**. Distance cropping is used to ensure that only relevant values are retained and distanced value are discarded to avoid confusion. Inverting distance values has the effect of providing higher weighting for closer object and lower weighting for distanced object.
- **Gaussian Noise**. Gaussian noise is added to all valid pixels to simulate sensor's accuracy and repeatability. Besides, noise with higher magnitude also motivate denoising ability in AE as proposed in [56].

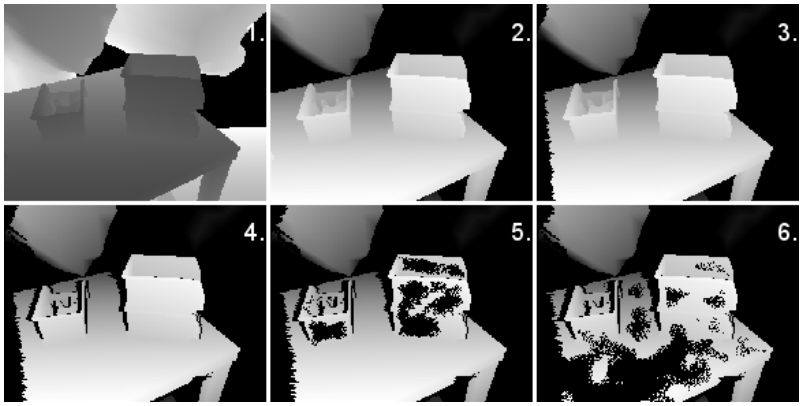



**Figure 4.7:** Data augmentations on depth image. The original depth image is first inverted, followed by baseline shift, depth shadow and homogeneous dropout on objects and table.

- **Perturbed Normalisation**. Data normalisation is a common pre-processing step for neural network. By randomly perturbing the normally fixed mean and standard deviation values, normalisation not only can whiten input data but also augment scale and contrast of depth value.

All augmentations above are applied on noisy input $x$ during training as pre-processing. Embedding input $x^e$ only has baseline shift, depth shadow, removal of all background pixel and fixed normalisation as pre-processing. Clean input $x^c$ on the other hand only has removal of all background pixel and fixed normalisation as pre-processing.

### 4.2.3 Adversarial Augmentation

Adversarial examples is used here to improve the robustness of encoder by further avoiding synthetic feature over-fitting. Hand-crafted data augmentations are useful up to point because they offer rather predictable modifications. Adversarial examples on the other hand provide a dynamic and adaptive approach in generating modifications on input data. It automatically discovers the weakness of the network with respect to a loss function, and then modify the input data accordingly to exploit the weakness. The aim here, however, is to widen the synthetic training data domain and hope that it increases its intersection with real data domain.

Algorithm 4.1 depicts the proposed algorithm. It is a randomised combination of multiple type of attacks: FGSM, iterative FGSM and least-likely FGSM. The idea of FGSM is to compute the gradient of loss with respect to input $x$, and then modify $x$ with this gradient. Least-likely FGSM changes the training targets and modification operation i.e. $+$ to $-$. Whereas iterative FGSM repeats the process with bounded augmentation strength.

Most adversarial attacks have been applied on classification tasks, hence the major difference here is the replacement of cross-entropy loss with multiple type of loss functions. Unlike the algorithm proposed in [63], the entire batch of input data is involved in the augmentation

due to the randomisation and the different in its purpose. The only drawback here is the additional computations which involve multiple forward and backward-passes.

**Algorithm 4.1** Randomised Adversarial Examples as Data Augmentation. $\mathcal{U}_{b,i,f}$ are uniform samplings for boolean, integer and float. $n$ is the batch size of input data $x = \{x_d\}_{d=1}^{n}$. $\rho$ is the number of semantic classes. Inputs are noisy depth image $x$, clean depth image $x^c$, semantic segmentation $x^s$ and object boundary $x^b$ of the respective depth image. Hyperparameter $\varepsilon$ controls the augmentation strengths. The output is a batch of augmented training data.

**procedure** AdversarialAugmentation($x$, $x^c$, $x^s$, $x^b$, $\varepsilon$)
  $l \leftarrow \mathcal{U}_b(0, 1)$
  **if** $l$ **then**
    $\bar{x}^c \leftarrow x^c$
    $\bar{x}^s \leftarrow x^s$
    $\bar{x}^b \leftarrow x^b$
  **else**
    $\varrho \in \mathbb{R}^n \leftarrow \mathcal{U}_i(1, \rho - 1)$
    $\bar{x}^c \leftarrow -x^c$
    $\bar{x}^s \leftarrow (x^s + \varrho) \bmod \rho$
    $\bar{x}^b \leftarrow (x^b + 1) \bmod 2$
  **end if**
  $k \leftarrow \mathcal{U}_i(1, 4)$
  $\omega \in \mathbb{R}^n \leftarrow \frac{\varepsilon}{k} \cdot \mathcal{U}_f(0, 1)$
  $x_0^{adv} \leftarrow x$
  **for** $i \in [0, ..., k-1]$ **do**
    $\hat{x}_i^c, \hat{x}_i^s, \hat{x}_i^b \leftarrow \mathrm{Dec}\Big(\mathrm{Enc}(x_i^{adv})\Big)$
    $\nabla\mathcal{L} \leftarrow \nabla_{x_i^{adv}}\mathcal{L}_{rec}\Big(\hat{x}_i^c, \bar{x}^c\Big) + \nabla_{x_i^{adv}}\mathcal{L}_{seg}\Big(\hat{x}_i^s, \bar{x}^s\Big) + \nabla_{x_i^{adv}}\mathcal{L}_{bou}\Big(\hat{x}_i^b, \bar{x}^b\Big)$
    **if** $l$ **then**
      $x_{i+1,\,j}^{adv} \leftarrow x_{i,\,j}^{adv} + \omega_j \cdot sign\nabla\mathcal{L}_j, \quad \forall j \in n$
    **else**
      $x_{i+1,\,j}^{adv} \leftarrow x_{i,\,j}^{adv} - \omega_j \cdot sign\nabla\mathcal{L}_j, \quad \forall j \in n$
    **end if**
  **end for**
  **return** $x_k^{adv}$
**end procedure**

### 4.2.4 Loss Functions

There are 3 losses for 3 decoder outputs $\mathrm{Dec}(\hat{x}^c, \hat{x}^s, \hat{x}^b \,|\, z)$ respectively. The first loss function is designed for decoder to reconstruct a clean version of depth image. Since all inputs are normalised, the reconstruction loss $\mathcal{L}_{rec}$ function is a *Huber* or *Smooth L1* loss which is less sensitive to outliers. The second and third loss functions are cross-entropy

$\mathcal{L}_{seg}$ for multi-class classification in semantic segmentation, and binary cross-entropy $\mathcal{L}_{bou}$ for binary classification in object boundary prediction.

$$\begin{aligned}
\mathcal{L}_{rec}\left(\hat{x}^c, x^c\right) &= \frac{1}{n \cdot \mathbb{K}} \sum_{i}^{n} \sum^{\mathbb{K}} \left[ \begin{cases} \frac{1}{2}\left(\hat{x}_i^c - x_i^c\right)^2 & if \quad \left|\hat{x}_i^c - x_i^c\right| < 1 \\ \left|\hat{x}_i^c - x_i^c\right| - 0.5 & otherwise \end{cases} \right]_{\mathbb{K}} \\
\mathcal{L}_{seg}\left(\hat{x}^s, x^s\right) &= \frac{-1}{n \cdot \mathbb{K}} \sum_{i}^{n} \sum^{\mathbb{K}} \left[ \log\left( \frac{\exp\left(\hat{x}^s_{i,\,x_i^s}\right)}{\sum_j^{\rho} \exp\left(\hat{x}^s_{i,\,j}\right)} \right) \right]_{\mathbb{K}} \\
\mathcal{L}_{bou}\left(\hat{x}^b, x^b\right) &= \frac{-1}{n \cdot \mathbb{K}} \sum_{i}^{n} \sum^{\mathbb{K}} \left[ x_i^b \ \log\left(\sigma\left(\hat{x}_i^b\right)\right) + \left(1 - x_i^b\right) \log\left(1 - \sigma\left(\hat{x}_i^b\right)\right) \right]_{\mathbb{K}}
\end{aligned} \tag{4.2}$$

[71] proposed online bootstrapping in the context of semantic segmentation, a method in which only the top $\mathbb{K}$ pixels with largest loss are selected for gradient computation and update during back-propagation. It neglects easy pixels and focuses the updates on solving difficult regions. It is implemented in all 3 decoder losses, $[\,\cdot\,]_{\mathbb{K}}$ above depict the set of top $\mathbb{K}$ pixels with largest loss within a spatial size $\mathbb{R}^{w \times h}$ of each output image.

In order to avoid training GAN with discriminators as in [61], or having to predict mean and standard deviation as in [24], distribution properties of latent code generated by encoder are directly evaluated in the loss function. $\mathcal{L}_{sty}$ simply enforces properties of unit Gaussian on latent code by penalising mean and standard deviation with $L2$ loss at 0 and 1, which is different from directly cultivating latent code in true Gaussian distribution.

$$\mathcal{L}_{sty}\left(z\right) = \frac{1}{L} \sum_{i}^{L} mean\left(\left[z_i\right]_n\right)^2 + \left(1 - std\left(\left[z_i\right]_n\right)\right)^2 \tag{4.3}$$

where *mean* and *std* are mean and standard deviation operation, whereas $[z_i]_n$ is the set of $i^{th}$ latent code across entire mini-batch.

Embedding loss $L_{emb} \in \mathbb{R}^n$ penalises deviation of latent codes between noisy input $x$ and cleaner embedding input $x^e$. The aim is to reduce variance in the AE bottleneck so that the generated latent code contain only essential information, regardless of the augmentations applied on $x$. Cosine similarity is used to compute similarity between 2 latent codes.

$$\mathcal{L}_{emb}\left(z, z^e\right) = 1 - \frac{z \cdot z^e}{\left\|z\right\|_2 \left\|z^e\right\|_2} \tag{4.4}$$

$\mathcal{L}_{emb}$ is constructed with a modified focal loss[72] scaling which further diminishes contributions of nicely predicted inputs and focus more on input it still cannot predict properly. $\alpha_f$ is a hyper-parameter, 0 means no scaling. The way it differ from online bootstrapping used in Equation (4.2) is that focal loss includes all instance of losses, which is more suitable in the domain of $\mathbb{R}^n$ compared to $\mathbb{R}^{w \times h}$ where $n \ll (w \times h)$. The final AE loss function is the combination of all losses with scaling hyper-parameters $\alpha_s$ and $\alpha_e$ that control the contribution of $\mathcal{L}_{sty}$ and $\mathcal{L}_{emb}$.

$$\mathcal{L}_{AE} = \mathcal{L}_{rec} + \mathcal{L}_{seg} + \mathcal{L}_{bou} + \alpha_s \mathcal{L}_{sty} + \alpha_e \frac{1}{n} \sum_{i}^{n} \mathcal{L}_{emb,\,i} \exp(-\mathcal{L}_{emb,\,i})^{\alpha_f} \tag{4.5}$$

### 4.2.5 Database Search

After AE training has completed, all scenes from database in the form of depth images are transformed into latent code. This is done by using the embedding input $x^e$ as input, forward-pass it through encoder. The generated latent code(codebook) are then appended back into database accordingly.

Retrieving nearest neighbours of an query scene is performed in 2 steps. First, depth image of the query scene is forward-passes into encoder to get its latent code $z_{\mathfrak{q}}$. Second, cosine similarity $\measuredangle$ is computed between query latent code $z_{\mathfrak{q}}$ and all latent code recorded in database. The $\mathbb{K}$ nearest neighbours $\mathcal{S}$ of the query scene is then the top $\mathbb{K}$ $[\cdot]_{\mathbb{K}}$ scenes in the database with the highest similarity score.

$$\begin{aligned} \mathcal{S} &= \left[\left\{ \measuredangle\left(z_{\mathfrak{q}},\, z_i\right) \;\middle|\; i \in \mathcal{D} \right\}\right]_{\mathbb{K}} \\ \measuredangle\left(z_{\mathfrak{q}},\, z_i\right) &= \frac{z^{\mathfrak{q}} \cdot z^i}{\left\|z^{\mathfrak{q}}\right\|_2 \left\|z^i\right\|_2} \end{aligned} \tag{4.6}$$

## 4.3 Biased GMM Sampling

This work proposes the third biasing mechanism for GMM-based sampling, on top of Gaussian partitioning in $\mathbb{C}$-space and importance weighting among Gaussian components. The main idea is to change the, otherwise fixed, weights of Gaussian components according to information deduced from query scene. This allow the fixed task-space oriented biasing in [2] to have the ability to adapt according to situation posted in every queries.

### 4.3.1 GMM Fitting

In improving the quality of $\mathbb{C}$-space density estimation by GMM, a new key-configuration extraction technique is proposed to first improve the quality of data used in fitting GMM. Given a low variance path from $\overline{\text{RRT}}$, a group of corresponding straight line paths $\overline{\mathcal{T}}$ are constructed with linear equation $a = bm + c$, where $b$ is path time-steps, $c$ is the $a$-intercept, and $m$ is the different between start and end point in the dimension of $a$. These straight line paths are constructed with TCP's translation and rotation in Cartesian-space using forward kinematic $\phi$.

$$\mathcal{T} \leftarrow \overline{\text{RRT}}(env,\, task) \tag{4.7a}$$

$$\overline{\mathcal{T}}^j = \left[\left(\phi(\mathcal{T}_T)^j - \phi(\mathcal{T}_0)^j\right) i + \phi(\mathcal{T}_0)^j\right]_{i=0}^{T} \tag{4.7b}$$

Discrepancies between actual and straight line path in Cartesian-space indicate the likelihood of informative configuration. The farther the actual path deviates from the optimal path, the more informative its configurations become. These deviations are, therefore, used as a measure to extract key configurations $Q$ from a path.

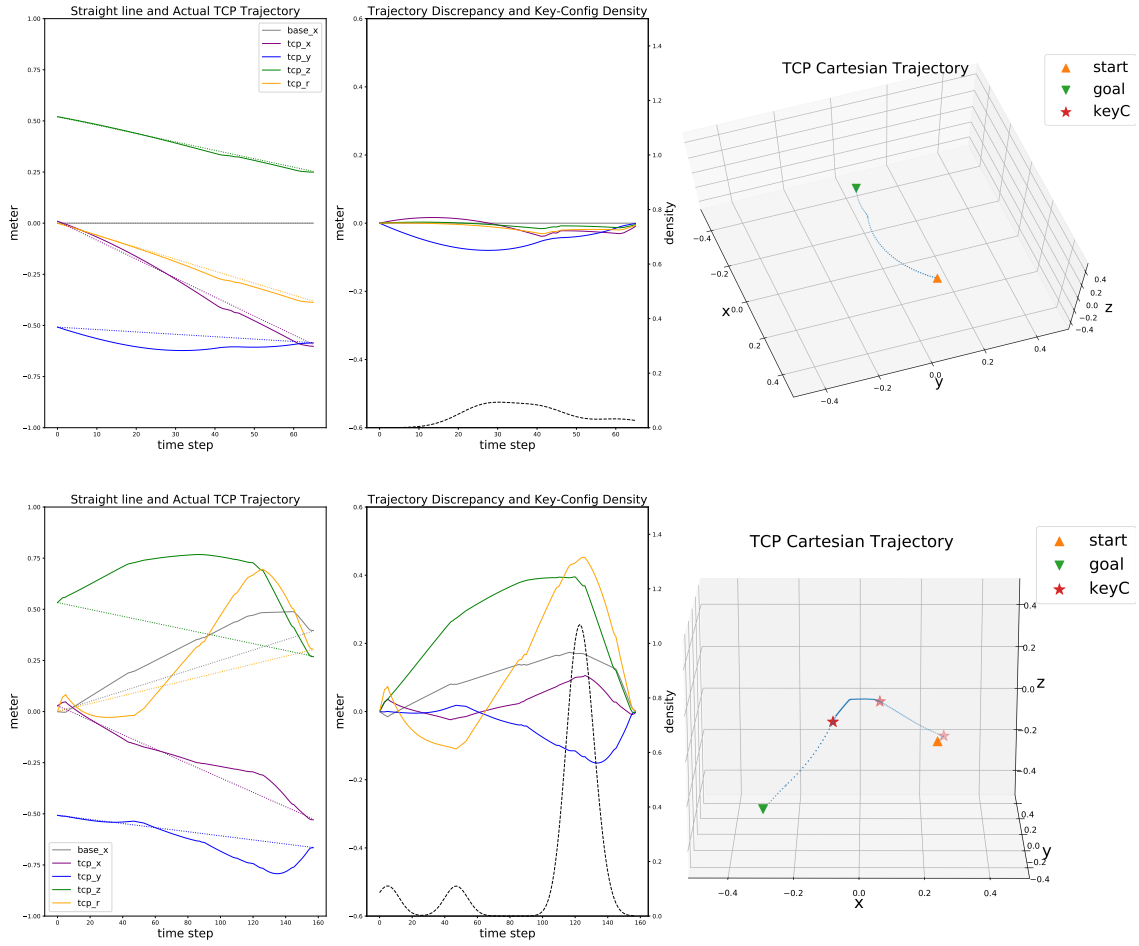



**Figure 4.8:** 2 paths in the process of key-configuration extraction.

Local extremal detector $\mathcal{X} : \{\delta\} \to \{e\}$, where $e$ is a local extrema. $\mathbf{e}$ has a collection of extrema with magnitude at least the lower extrema threshold $\alpha_{xl}$. The density distribution of these somehow noisy key configuration candidates are then estimated with KDE $\mathcal{K}$. The final key configurations are the local maxima of the density function with magnitude at least the upper extrema threshold $\alpha_{xu}$.

$$\mathbf{e} = \left\{ \mathcal{X}\Big(\phi\left(\mathcal{T}\right)^j - \overline{\mathcal{T}^j}\Big) \;\Big|\; |e| \geq \alpha_{xl},\, j \in \Big\{ \text{base}_x,\, \text{tcp}_{x,y,z,r} \Big\} \right\} \tag{4.7c}$$

$$\Psi(\mathcal{T}) = \left\{ \mathcal{X}\Big(\mathcal{K}(\mathbf{e})\Big) \;\Big|\; e \geq \alpha_{xu} \right\} \tag{4.7d}$$

Besides 2 magnitude thresholds, there are another 3 bandwidth parameters for 2 extrema detectors and 1 KDE. The extracted key configuration $Q$s in joint space are then stored in $\mathbf{Q}$ for building GMM as [2] in Equation (3.5). Figure 4.8 depicts the process of this extraction in plots and their implications in Cartesian-space.

### 4.3.2 Vision Biasing

**Weights Aggregation** Besides scene similarity score $\mathcal{S}_i$ that ranges $[-1, 1]$ computed with cosine similarity between latent codes, target pose proximity $\psi$ that ranges $[0, 1]$ is also applied on $x$, $y$, and $r$ of TCP in Cartesian-space. $\psi$ is an additional weightings used to scale the contribution of each paths contained in all $\mathbb{K}$ nearest neighbours' scenes, based on the differences between query $xyr_{\mathfrak{q}}$ and recorded $xyr_j$ target poses.

$$\mathcal{S}' = \left\{ \mathcal{S}_i \cdot \psi(xy_j,\, xy_{\mathfrak{q}} \,|\, \alpha_{xy}) \cdot \psi(r_j,\, r_{\mathfrak{q}} \,|\, \alpha_r) \,\middle|\, j \in |\mathcal{T}|_i,\, i \in \mathbb{K} \right\} \tag{4.8a}$$

$$\psi(a,\, b \,|\, \alpha) = \frac{1}{2}\cos\Big(\min(\pi,\, \alpha \left\|a - b\right\|_2)\Big) + \frac{1}{2} \tag{4.8b}$$

$\mathcal{S}'$ is the new set of similarity score that has taken both scene and task similarities into account, for all paths in $\mathbb{K}$ nearest neighbours. The proposed new weights for each Gaussian components $w'_g$ is a weighted average of assignment probabilities $p_{i,g}$ of all key-configurations $Q$ contained in all paths in $\mathcal{S}'$, according to Equation (3.5).

$$w'_g = \frac{1}{W} \sum_{j}^{|\mathcal{S}'|} \sum_{i}^{|Q|_j} \mathcal{S}'_j \cdot p_{i,g} \qquad\qquad W = \sum_{g}^{G} w'_g \tag{4.8c}$$

$$w^*_g = \alpha_w \sup \mathcal{S}' w'_g + (1 - \alpha_w \sup \mathcal{S}')\, w_g \tag{4.8d}$$

In calculating the final weight $w^*_g$, the supremum of similarity score $\sup \mathcal{S}'$ is used to adaptively scale the update factor $\alpha_w$, reducing update when there is no good match found. High $\alpha_w$ allows vision biasing to takeover based on the current scene and ignores the statical ranking of components derived from database, and vice-versa. Therefore, this method has $\alpha_w$ and $\alpha_{xy,\,r}$ as experimental hyper-parameters.

**Weights Prediction** In addition to the original AE, a weight predictor $\mathrm{W}(w' \,|\, z,\, task)$ is built to predict the weights of Gaussian components directly from latent code $z$ and task description $task$. W is constructed as an output head branches out at AE bottleneck with 2 additional fully-connected layers, and produces $w' \in [0, 1]^G$ after a softmax layer. Figure 4.9 depicts the new AE architecture.

$$\begin{aligned} \mathcal{L}_{\mathrm{W}} &= D_{\mathrm{KL}}(w' \,|\, w)_{\mathbb{K}} \\ &= \frac{1}{n} \sum_{i}^{n \cdot \mathbb{K}} \left[ w_i \log \frac{w_i}{w'_i} \right]_{\mathbb{K}} \\ \mathcal{L}_{AE} &\leftarrow \mathcal{L}_{AE} + \alpha_{\mathrm{W}} \mathcal{L}_{\mathrm{W}} \end{aligned} \tag{4.9}$$

Weights predictor loss $\mathcal{L}_{\mathrm{W}}$ is a KL divergence loss that minimises the differences in probability distribution between predicted weights $w'$ and actual weight $w$. The top $\mathbb{K}$ operator $[\cdot]_{\mathbb{K}}$ bootstraps/includes only the top $\mathbb{K}$ components' weight out of $G$ in computing losses and gradient updates for every data in a mini-batch of size $n$. The target distribution $w_g$ came from the assignment probability $p_{i,g}$ of key-configuration $\{Q\} \leftarrow \Psi(\mathcal{T})$ from a

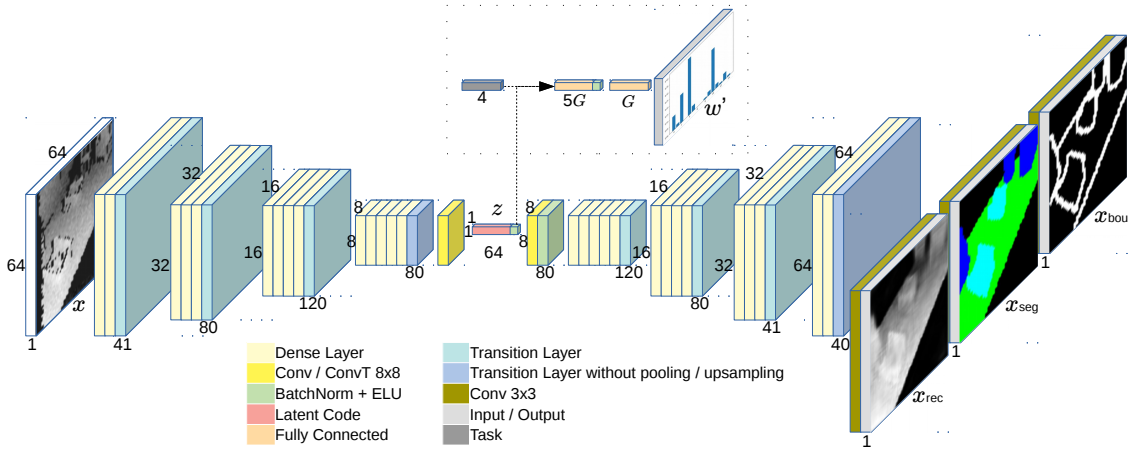



**Figure 4.9:** Network architecture of AE with direct components' weight prediction.

path $\mathcal{T}$ randomly chosen in scene. $p_{i,g}$ are summed component-wise and re-normalised if there exists more than one $Q$ in the selected path. The corresponding task description $task \in \mathbb{R}^4$ comprises: **1.** task type: either pick or place, represented in -1 and 1; **2.** target pose in x, y, and in-plane rotation of TCP in Cartesian-space.

Due to the very peaked weight distributions $p_{i,g}$, the top $\mathbb{K}$ of KL divergence across $G$ components normally only consist of those undershoot $\frac{w_g}{w'_g} > 1$ and a lot of those with $w_g = 0$. Those overshoot $\frac{w_g}{w'_g} < 1$ are then indirectly excluded. Since the network has the tendency to bias itself into learning only the empirical weight distribution of the entire dataset $\mathcal{D}$, penalising only the undershoots has the implication of learning scene specific knowledge while preserving the background information.

$$w^*_g = \alpha_w W(z_{\mathfrak{q}}, task_{\mathfrak{q}})_g + (1 - \alpha_w)\, w_g \tag{4.10}$$

During inference, weight predictor W is supplied with latent code $z_{\mathfrak{q}} \leftarrow \text{Enc}(x_{\mathfrak{q}})$ and queried target pose $xyr_q$, and produces new weight $w'$. With the update factor $\alpha_w$, the final weight $w^*$ is computed similar to Equation (4.8) but without adaptive scaling.

# 5 Evaluation

## 5.1 Database Creation

The selected application scenario for evaluation is a tabletop pick and place scene with Small Load Carrier (SLC) as object and Autonomous Industrial Mobile Manipulator (AIMM)[73] as manipulator. Table, the platform that serves all objects in a scene, is placed at a fixed position with a variable height vertical extension. Whereas SLC, the only type of object, can appear in 3 forms: obstacle, pick or place-target. An obstacle is a constellation of SLCs in either horizontal, vertical or both directions, arranged in shoulder to shoulder manner. Whereas pick-target is an individual isolated SLC, oriented in the way where it can be pick-up directly. Place-target, in contrast, is just a vacant place on the table where new SLC can be placed on.

AIMM as depicted in Figure 5.1, is a mobile platform (3 DoFs) with a robotic arm (7 DoFs) and a gripper (2 DoFs). It is equipped with one stereo camera mounted on a pan-tilt unit (2 DoFs) elevated from its base. This sensor produces depth image in, here defined as, the robot's perspective, used as the main sensor in perceiving robot workspace.

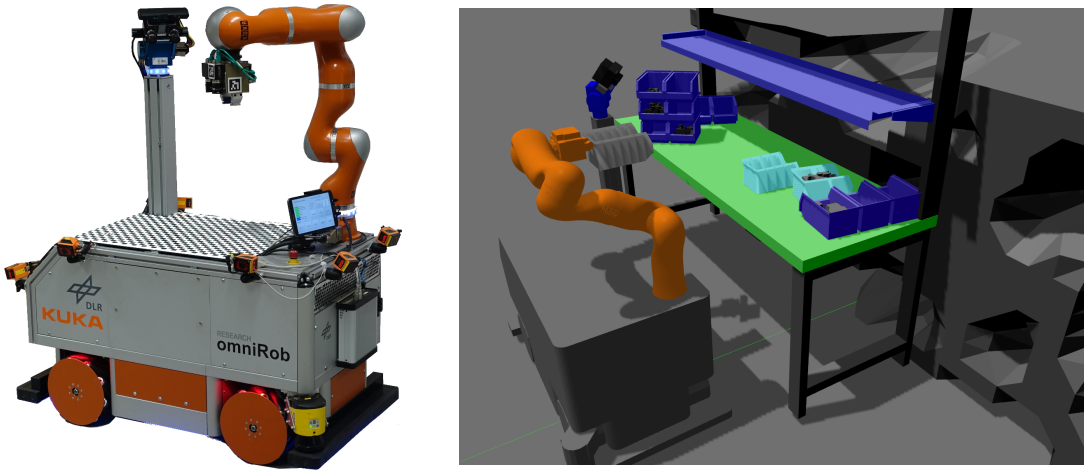



**Figure 5.1:** AIMM[1]in simulated robot workspace using Gazebo and ROS[2].

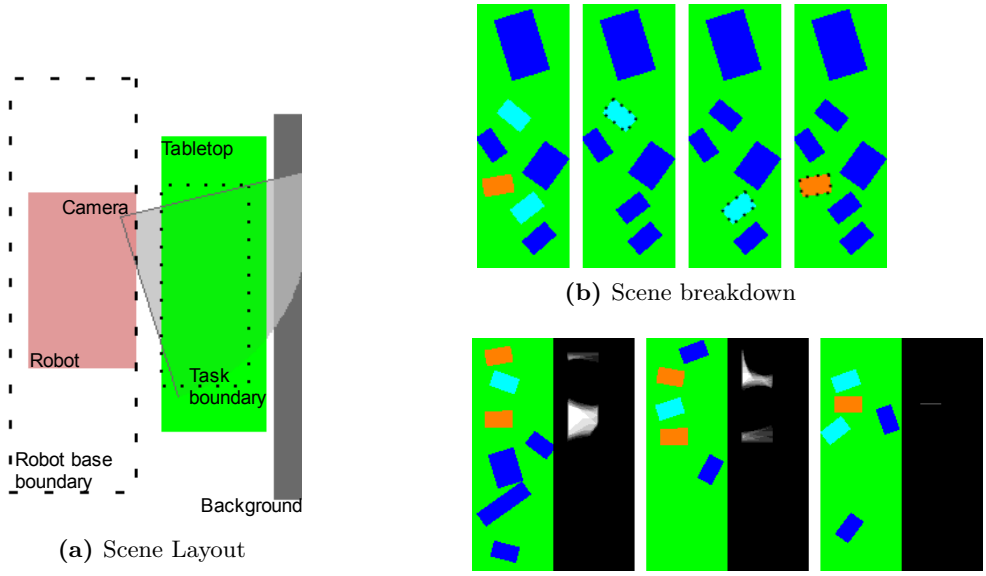
[1]DLR AIMM: www.dlr.de/rm/en/desktopdefault.aspx/tabid-11409

### 5.1.1 Scene Construction

There are 7 steps in constructing a scene for the targeted planning scenario in simulated workspace[2]. $\mathcal{U}(lb, ub)$ or $\mathcal{U}(\pm b)$ depicts uniform random sampling between lower and upper bound. Procedure:

1. Robot base joints are randomly sampled, around a fixed relative distance with table ($\delta x_{table}$, $\delta y_{table}$).
   - base x-axis, $base_x \leftarrow \mathcal{U}(\pm 0.8m) + \delta x_{table}$
   - base y-axis, $base_y \leftarrow \mathcal{U}(\pm 0.05m) + \delta y_{table}$
   - base in-plane rotation, $base_r \leftarrow \mathcal{U}(\pm 0.1rad)$
2. Camera pan-tilt joints are randomly sampled around a default viewing direction ($cam_{p0}, cam_{t0}$).
   - camera pan, $cam_p \leftarrow \mathcal{U}(\pm 0.05rad) + cam_{p0}$
   - camera tilt, $cam_t \leftarrow \mathcal{U}(\pm 0.05rad) + cam_{t0}$
3. Table height changes while $xy$ position remain constant.
   - table height, $table_z \leftarrow \mathcal{U}(\pm 0.05m)$
4. Scene background is randomly generated with extruded perlin noise.
5. Pick-target placement.
   - Number of pick-target $\leftarrow \mathcal{U}(1, 2)$
   - Pose on tabletop relative to robot base
     - $slc_x \leftarrow \mathcal{U}(\pm 0.35m) + \delta x_{base}$
     - $slc_y \leftarrow \mathcal{U}(\pm 0.12m) + \delta y_{base}$
     - $slc_r \leftarrow \mathcal{U}(\pm 0.7rad)$
   - Check table occupancy before inserting
   - Add load in SLC with extruded perlin noise
6. Obstacle placement.
   - Number of obstacle $\leftarrow \mathcal{U}(2, 7)$
   - Type of obstacle $\leftarrow \mathcal{U}(1, 5)$
   - Layer of obstacle $\leftarrow \mathcal{U}(1, 3)$
   - Pose on tabletop.

[2] Gazebo+ROS: gazebosim.org/tutorials?tut=ros_overview




**(a)** Scene Layout

**(b)** Scene breakdown

**(c)** Tabletop occupancies

**Figure 5.2:** (a) is the robot workspace configuration. (b) breakdowns each task comprised in ($1^{st}$). When one pick-target (upper aqua) is active, the second pick-target is treated as obstacle and vise-versa($2^{nd}$, $3^{rd}$). Place-target (orange) does not play any obstacle role($4^{th}$). (c) is the free space detection for place-targets.

  - $slc_x \leftarrow \mathcal{U}(length_{table}\, m)$
  - $slc_y \leftarrow \mathcal{U}(width_{table}\, m)$
  - $slc_r \leftarrow \mathcal{U}(2\pi\, rad)$
- Check table occupancy before inserting
- Add load in SLC with extruded perlin noise

7. Place-target placement.
   - Search for free space
   - Insert one at the centroid of each blob.

Perlin noise[70] is used in generating uneven surfaces for background and SLC load. To do that, a 2D perlin noise image is first generated, followed by an upward extrusion(multiplication) and conversion into mesh surface. Figure 5.1(b) depicts perlin loads and background with different texture.

There are 2 types of manipulation targets for robot TCP pose: pick and place. Both targets are located within the task boundary, which roughly represents the robot perception range at its initial location. This initial robot base location is randomly assigned within the robot base boundary as depicted in Figure 5.2(a). Moreover, SLCs' roles switch according to the current planning task. Figure 5.2(b) depicts the breakdown of an occupancy map.

Free space detection is done by convolving/correlating the tabletop occupancy map with kernels the size of a SLC, gripper and arm footprints, rotated in multiple orientations. Full kernel return indicates collision-free SLC placement at that specific orientation. Each blob of free-space is assigned one place-target at the centroid. Figure 5.2(c) depicts the free-space detections and occupancy maps of the tabletop scene at different scenarios.

RGB and depth images are produced from a simulated depth sensor, designed according to the specifications of *rc-visard*[3]. It is a stereo camera that is installed on the pan-tilt unit of the real AIMM. With 65mm baseline, it is suitable for tabletop scenario. There are in total 14000 training data generated using these settings. The first half of them contain all training labels i.e. image-path pairs, and the second half contain only images. With full information, the former form the database $\mathcal{D}$ with 7000 scenes, and it is able to train the all networks in the pipeline. Whereas the latter, which were acquired in relatively easier and faster pace, is excluded from updating weights prediction head in Equation (4.9).

## 5.2 Scene Retrieval Accuracy

### 5.2.1 Experiment Condition

Every data needs to go through several augmentations before being used in training. As mentioned in Section 4.2.2, the intensities of each augmentations are depicted in Table 5.1. Distance crop is applied pixel-wise. Nullified pixels are then excluded from noise addition. Adversarial augmentation in Algorithm 4.1 is applied at the end on normalised values.

| Augmentation | Intensity |
|---|---|
| Distance Crop | $x \leftarrow \begin{cases} 0 & if\ x < 200mm \\ 0 & if\ x > 1200mm \\ x & otherwise \end{cases}$ |
| Gaussian Noise | $x \leftarrow x + \mathcal{N}(0, \mathcal{U}_f(0,\ 0.02))$ |
| Perturbed Norm | $\mu_x \leftarrow \mu_x + \mathcal{U}(\pm 0.05)$ |
| | $\sigma_x \leftarrow \sigma_x + \mathcal{U}(\pm 0.03)$ |
| Adversarial Example | $\varepsilon = 0.3$ |

**Table 5.1:** Data augmentations.

| Parameter | | Value |
|---|---|---|
| $\mathcal{L}_{rec}$ : | $\mathbb{K}$ | $\frac{w \times h}{10}$ |
| $\mathcal{L}_{seg}$ : | $\mathbb{K}$ | $\frac{w \times h}{10}$ |
| $\mathcal{L}_{bou}$ : | $\mathbb{K}$ | $\frac{w \times h}{10}$ |
| $\mathcal{L}_{sty}$ : | $\alpha_s$ | 0.005 |
| $\mathcal{L}_{emb}$ : | $\alpha_e$ | 0.01 |
| | $\alpha_f$ | 2 |

**Table 5.2:** Hyper-parameters for networks' training.

The original image has spatial size $240 \times 320$. After data augmentations, it is scaled to $w \times h = 64 \times 64$ and form a training mini-batch with batch size $n = 128$. Training is then ran for 210 epochs. Weight initialisations for the network is done with [74]. AMSGrad[75], the improved Adam[76] is used for gradient descent during back-propagation. The first and second moment parameter $\beta_1$, $\beta_2$ is fixed at 0.9 and 0.999 with no weight decay. Cosine learning annealing with warm restart [77] is used to schedule the learning rate for every

[3]roboception rc_visard 3D Sensor: roboception.com/de/rc_visard

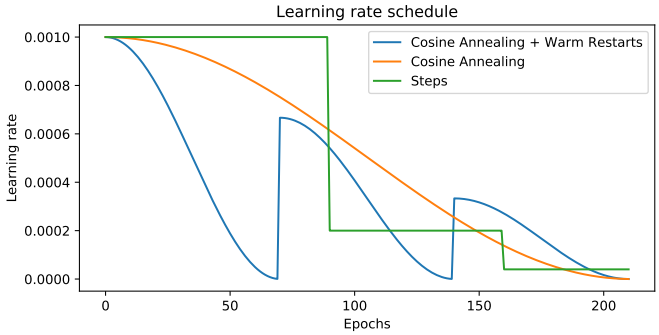



**Figure 5.3:** Comparison of learning rate annealing methods.

epoch during training. It has initial learning rate $10^{-3}$ and 3 warm restarts throughout the training epochs. Figure 5.3 depicts the comparison between multiple learning rate scheduling schemes. The number of trainable network parameters is 1.9M, which split equally (0.95M) among encoder and decoder.

### 5.2.2 Semantic Scene Similarity

**Performance Metric** There are 3 losses used in evaluating the performance of decoder in Equation (4.2), but none of them are suitable as the performance metric in the context of nearest neighbours retrieval. To evaluate the similarity between query $x_{\mathbf{q}}$ scene and $\mathbb{K}$ retrieved scenes $\mathcal{S}$ from Equation (4.6), their respective semantic segmentation labels $x^s$ are used for direct comparison.

$$\begin{aligned} \text{sim}(a^s,\, b^s) &= \sum_{k \in w \times h} \begin{cases} 1 & if \quad a_k^s \wedge b_k^s \\ 0 & otherwise \end{cases} \\ \text{S}^3 &= \frac{1}{w \times h} \frac{1}{\mathfrak{t}} \frac{1}{\mathbb{K}} \sum_{i=1}^{\mathfrak{t}} \sum_{j=1}^{\mathbb{K}} \text{sim}(x^s_{\mathbf{q}^i},\, x^s_{\mathcal{S}^j}) \end{aligned} \tag{5.1}$$

Semantic Scene Similarity ($\text{S}^3$) is computed across the entire test set $\mathfrak{t}$ with 500 test scenes, averaging the averaged similarity of semantic label between queries and their respective $\mathbb{K}$ nearest neighbours, where $\mathbb{K} = 5$. It is then normalised with total number of pixel $w \times h$ per scene to get a scalar output ranging [0, 1]. This score represents the performance of nearest neighbour retrieval, but it depends very much on the size of the database. Larger database will always have higher chances in getting more similar matches. To be consistent, all $\text{S}^3$ stated below are measured using the same database with 7000 complete scenes.

**Multi-Tasked Network** This experiment is designed to investigate the effects of multi-tasking a single network. Specifically, the effects of each decoder's losses in Equation (4.2) and their combinations are investigated below by judging on network abilities in nearest neighbours retrieval using $\text{S}^3$. ✓represents the present of the loss function during training.

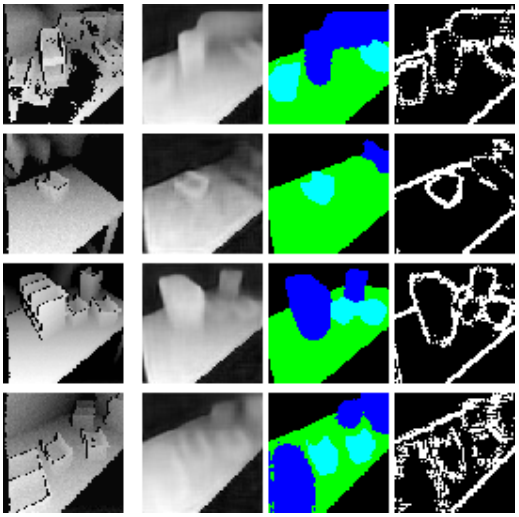


**Figure 5.4:** Encoder's inputs (first column) and three types of decoder's outputs.

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

**Table 5.3:** Four combinations of decoder's losses and their corresponding $S^3$.

This experiment shown that the network is capable of multi-tasking, and because of multi-tasking by including all reconstruction ***rec***, semantic segmentation ***seg*** and boundary prediction ***bou*** it out-performed those did not. Since the training objective is to improve $S^3$, their hyper-parameters are not tuned in perfecting decoder's outputs. Figure 5.4 depicts rather poor decoder outputs by the best performing combination.

**Latent Space Regularisation** This experiment is designed to investigate the effects of regularising latent code. Specifically, the effect of each AE bottle-neck's regularisations in Equation (4.5) and their combinations are investigated below by judging on network abilities in nearest neighbours retrieval using $S^3$. ✓represents the present of the regularisation loss function during training.

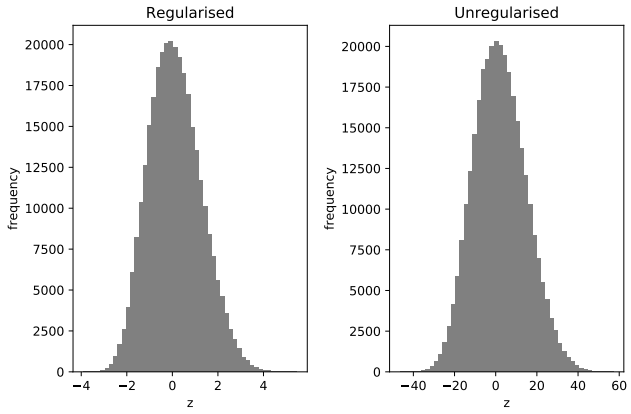



**Figure 5.5:** Regularised and unregularised latent code distribution.

| *sty* | *emb* | $S^3$ |
|---|---|---|
| ✓ | ✓ | 0.7906 |
| ✓ | | 0.7886 |
| | ✓ | 0.7914 |
| | | **0.7929** |

**Table 5.4:** Four combinations of latent code regularisations and their corresponding $S^3$.

This experiment shown that regularising latent code by penalising mean and standard deviation ***sty***, and clean input embedding ***emb*** do not help in improving $S^3$. The effect of regularising latent code, as shown in Figure 5.5, is obvious in restricting the variance of latent code distribution but hindered the quality of encoding and eventually worsen $S^3$.

**Adversarial Augmentation** This experiment is designed to investigate the effects of augmenting training data using adversarial example. Specifically, the effect of different augmentation strength $\varepsilon$ in Algorithm 4.1 are investigated below by judging on network abilities in nearest neighbours retrieval using $S^3$.

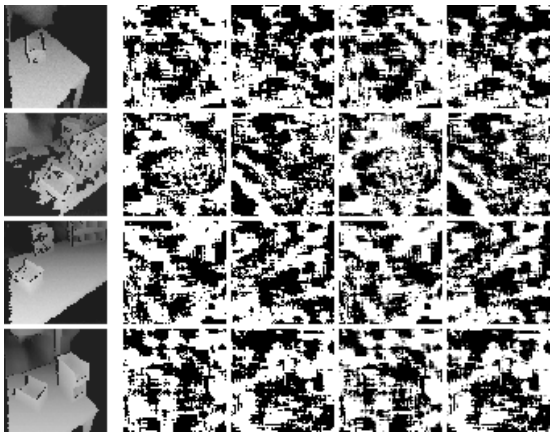


**Figure 5.6:** Inputs (first column) and their respective scaled augmentations.

| $\varepsilon$ | $S^3$ |
|---|---|
| 0 | 0.7891 |
| 0.1 | 0.7918 |
| 0.3 | **0.7929** |
| 0.5 | 0.7914 |

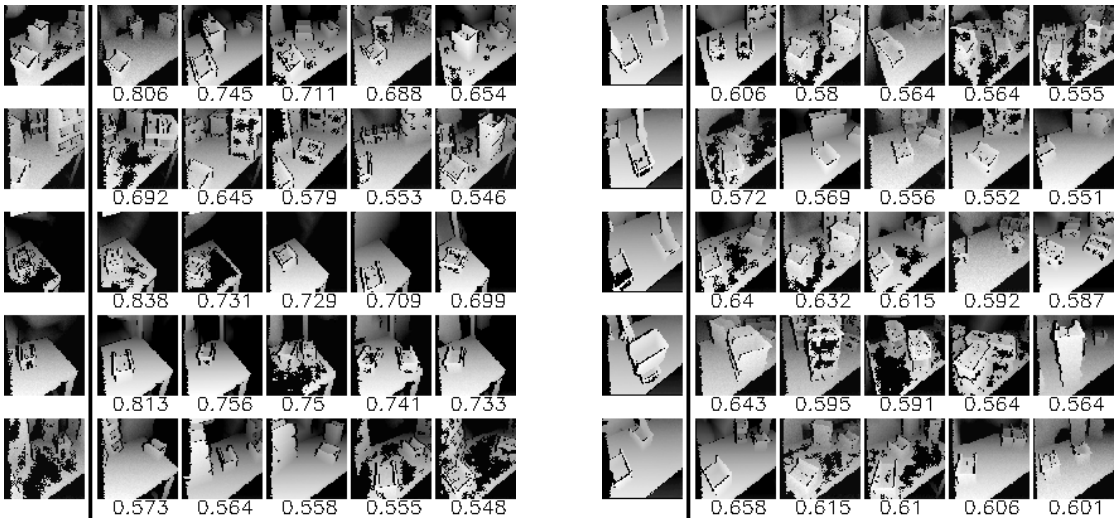
**Table 5.5:** Four different augmentation strength and their corresponding $S^3$.

This experiment shown that adversarial augmentation for this application indeed improved the performance, and tuning is required to maximise the improvement. Figure 5.6 depicts the actual augmentation applied on 4 example scenes, the first column is the normalised input image $x$ and the subsequence columns are $x_k^{adv} - x$ where $[l = 0, k = 1]$, $[l = 1, k = 1]$, $[l = 0, k = 3]$, $[l = 1, k = 3]$. For visualisation, $\omega \leftarrow \frac{\varepsilon_s}{2}$ and the augmentations are then scaled from $[-\omega, \omega]$ to [0, 255].

**inference** This experiment demonstrates properties of the network during inference. Specifically, time required for each steps in the inference pipeline is determined as depicted in Table 5.6. During the experiment, real depth images with size 640 × 480 are used as inputs. Data pre-processing includes image scaling, distance crop, invert and normalisation. The total time required for transforming raw sensor data to nearest neighbours is $\sim 9.2ms$.

| | |
|---|---|
| Pre-processing | 0.5 |
| Encoder | 8.0 |
| $\mathbb{K}$NN Search | 0.7 |
| Total | $9.2ms$ |

**Table 5.6:** Time required in each step of the inference pipeline.




**(a)** Simulated queries **(b)** Real queries

**Figure 5.7:** The first columns are the queried scenes; the subsequence columns are the retrieved neighbours in decreasing similarity score, labelled at the bottom.

The ability of applying knowledge learned in simulated environment onto real world is also investigated. Figure 5.7 depicts the examples of nearest neighbours retrieval in simulated and real planning scenes. The latter were captured with some offset in robot and camera pose compared to those in database, resulted in lower overall similarity scores. Despite the differences, this network is still able to function accurately for the nearer half of the scene.

## 5.3 Biased Sampling Efficiency

### 5.3.1 Experiment Condition

Out of all 14 DoFs, only 8 of them are included in motion planning. Robot is allowed to move its base along the x-axis (1 DoFs) and arm (7 DoFs) while reaching for target. The rest are set during scene construction and stay fixed during motion planning. There is one kinematic constraint that require the z-axis of TCP to align and pointing upward all the time. Moreover, TCP's $x, y, z$ translations and rotation around z-axis and base's x-axis are used in Section 4.3.1 for key-configuration extraction.

All experiments were conducted on a same test-set which were generated from the same scenario with training data. This test-set, which is not part of the database, consists of 200 scenes, each of them contain 1 to 2 pick-tasks and 0 to 3 place-tasks. In total it contains 295 and 233 tasks respectively. The values that are being evaluated $t_{search}$ is the time needed by planners in constructing its search tree $\mathcal{G}$. In another words, $t_{search}$ is the time needed for planner to `ARRIVED`, upper bounded by $t_{plan}$ as depicted in Algorithm 3.1.

For final path optimisation, a constant time $t_{opt} \leftarrow 0.3s$ is allocated for `RAND_SHORTCUT` and `SMOOTHING`. The total planning time should include both $t_{search}$, $t_{opt}$ and inference time for vision modules, but the latter are excluded when comparing among SMPs.

**Performance Metric** Area Under Cumulative Frequency (AUC$f$) is proposed to be the main metric in comparing performances among SMPs. It represents the area under curve of cumulative frequency of planning time histogram. At first, a histogram is constructed with $|\mathfrak{t}| = 528$ instances of planning time took to solve the test-set. Trapezoidal rule is then used to integrate the cumulative frequency C$f$ spanning from $t = \delta t$ where $\delta t \leftarrow 0.1s$ is the step for histogram, to $t = \overline{T}$ where $\overline{T} \leftarrow 5s$ is the threshold for failed attempt.

$$\begin{aligned} \mathrm{C}f(t) &= \sum_{i}^{|\mathfrak{t}|} \begin{cases} 1 & if \quad t_i < t \\ 0 & otherwise \end{cases} \\ \mathrm{AUC}f &= \frac{1}{|\mathfrak{t}| \left(\frac{\overline{T}}{\delta t} - 1\right)} \sum_{j=1}^{\frac{\overline{T}}{\delta t}-1} \frac{\mathrm{C}f(j\delta t) + \mathrm{C}f((j+1)\delta t)}{2} \delta t \end{aligned} \tag{5.2}$$

The advantage of AUC$f$ is that it is less susceptible to outlier compared to other performance metrics such as mean, standard deviation and extrema used in [2]. Normalisation in AUC$f$ provides stabilities over changes of $\overline{T}$ and $|\mathfrak{t}|$, which differ according to applications. The output value ranges from 0 meaning the planner failed the entire test-set, to 1 meaning the planner solved each and every query in test-set with $t_{search} < \delta t$.

**Baseline Performance** This experiment defined the baseline for all subsequence experiments by determining the performances of RRT that uses uniform sampling $\mathcal{U}$. Besides AUC$f$, other planner's properties are also determined: "$R^2$" represents the averaged $R^2$ scores of optimised paths; "Failure" represents the number of query planner failed to solve within $\overline{T}$; "Median" and "Min" are the statistical median and minimum planning time (in second) for solving the test-set. Moreover, performances of repetition sampling with improved pipeline are evaluated through 3 planners with different number Gaussian component $G$. Table 5.7 depicts the results of these 4 baseline planners.

| | $\mathcal{U}$ | GMM | | |
|---|---|---|---|---|
| | | $15G$ | $30G$ | $45G$ |
| $R^2$ | 0.921 | 0.964 | 0.964 | 0.964 |
| Failure | 26 | 6 | 4 | 5 |
| Median | 0.057 | 0.056 | 0.057 | 0.059 |
| Min | 0.039 | 0.038 | 0.039 | 0.041 |
| AUC$f$ | 0.906 | 0.966 | 0.967 | 0.970 |
| $\Delta$AUC$f$ (%) | | 6.0 | 6.1 | **6.4** |

**Table 5.7:** Baseline performances from RRT planners using uniform $\mathcal{U}$ and GMM sampling with 15, 30 and 45 Gaussian Components.

The results reinforced the claims made in [2] that $\mathbb{C}$-space biasing using GMM improve sampling efficiency, in term of AUC$f$ and number of failure. On top of that, the improvements on $R^2$ show that the proposed redundancy reduction measures helped not only in collecting training data, but also improved the path optimality during inference.

### 5.3.2 Weights Aggregation

This experiment is designed to investigate the effects of vision assisted biasing by weights aggregation of retrieved nearest neighbours. Specifically, the effects of proximity $\alpha_{xy,r}$ and weight update $\alpha_w$ parameters as depicted in Equation (4.8) are investigated below by judging on sampling efficiency using AUC$f$. The performance of uniform sampling $\mathcal{U}$ in Table 5.7 is used as the baseline in calculating the difference $\Delta$AUC$f$.

| $\alpha_{xy}$ | $\alpha_r$ | $\alpha_w$ | $\Delta$AUC$f$ (%) | | |
|---|---|---|---|---|---|
| | | | 15$G$ | 30$G$ | 45$G$ |
| 5.0 | 2.0 | 1.00 | 6.8 | 6.4 | 4.5 |
| 7.0 | 2.5 | 1.00 | **7.0** | 7.0 | 4.7 |
| 12.0 | 3.3 | 1.00 | 6.6 | 6.7 | 4.2 |

**Table 5.8:** Search in proximity parameters.

| $\alpha_{xy}$ | $\alpha_r$ | $\alpha_w$ | $\Delta$AUC$f$ (%) | | |
|---|---|---|---|---|---|
| | | | 15$G$ | 30$G$ | 45$G$ |
| 7.0 | 2.5 | 0.25 | 6.8 | 6.3 | 4.7 |
| | | 0.50 | 6.8 | 6.7 | 4.7 |
| | | 0.75 | 6.8 | **7.1** | **4.8** |

**Table 5.9:** Search in update parameter.

The best performing parameters combinations for each GMM are depicted in Table 5.8 and 5.9. All GMMs have improved results compared to $\mathcal{U}$ and their respective baselines, except GMM with 45 Gaussian components which has worse performance with vision assisted biasing using weight aggregation.

### 5.3.3 Weights Prediction

This experiment is designed to investigate the effects of vision assisted biasing by direct weights prediction. Specifically, the effects of loss $\alpha_W$ and weight $\alpha_w$ update parameters as depicted in Equation (4.9) are investigated below by judging on sampling efficiency using AUC$f$. The performance of uniform sampling $\mathcal{U}$ in Table 5.7 is used as the baseline in calculating the difference $\Delta$AUC$f$.

| $\alpha_W$ | $\alpha_w$ | $\Delta$AUC$f$ (%) | | |
|---|---|---|---|---|
| | | 15$G$ | 30$G$ | 45$G$ |
| 1.0 | 1.0 | 6.7 | 7.4 | 7.7 |
| 1.5 | 1.0 | 6.5 | 7.3 | **8.1** |
| 2.0 | 1.0 | 7.1 | 7.3 | 7.8 |
| 3.0 | 1.0 | **7.5** | 7.4 | 7.5 |
| 4.0 | 1.0 | 7.2 | **7.4** | 7.2 |

**Table 5.10:** Search in loss parameter.

| $\alpha_W$ | $\alpha_w$ | $\Delta$AUC$f$ (%) | | |
|---|---|---|---|---|
| | | 15$G$ | 30$G$ | 45$G$ |
| 1.5 | 0.6 | 6.6 | 7.2 | 7.4 |
| | 0.8 | 7.0 | 7.1 | 7.2 |
| 3.0 | 0.6 | 6.6 | 6.9 | 7.6 |
| | 0.8 | 6.7 | 7.1 | 7.8 |

**Table 5.11:** Search in update parameter.

The best performing parameters combinations for each GMM are depicted in Table 5.10. All GMMs have improved results compared to $\mathcal{U}$ and their respective baselines. Moreover, they also have the best performances over vanilla GMM and weight aggregation method. Using weights prediction, GMM with 45 Gaussian components turned out to be the best performing one, contradicting the results in weights aggregation.

### 5.3.4 Benchmark

When comparing overall planning efficiency between SMP and OMP, all aspects of the planning pipeline are included. Table 5.12 depicts the time for path post optimisation and vision modules needed by SMPs. Given the same planning conditions OMP, in contrast, requires none of these extra steps.

| | Pre Processes | Encoder | Weights Predict | $\mathbb{K}$NN Search | Weights Aggregate | Post Optim | Total $(ms)$ |
|---|---|---|---|---|---|---|---|
| $\mathcal{U}$ | - | - | - | - | - | 300.0 | 300.0 |
| GMM | - | - | - | - | - | 300.0 | 300.0 |
| WA | 0.5 | 8.0 | - | 0.7 | 0.5 | 300.0 | 309.7 |
| WP | 0.5 | 8.0 | 1.0 | - | - | 300.0 | 309.5 |

**Table 5.12:** Extra time needed by SMPs.

KOMO[7][4] is the OMP selected for this benchmarking due to the user-friendly interface and minimal setup dependencies. With augmented Lagrangian, this method solves the planning problem by minimising 2 type of costs: transitional cost penalises squared accelerations to maintain path's smoothness; task related costs penalises sum of squared errors from TCP target 6D pose and motion alignment. Additional inequality constraints are used for collision and joint limits. All trajectories has 1 phase and 20 time slices in 5 seconds.

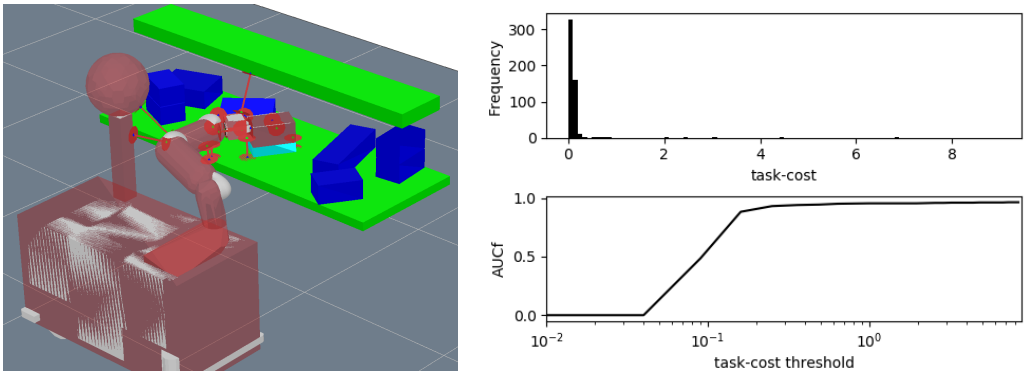



**Figure 5.8:** KOMO planner, Task-cost histogram and AUC$f$-threshold plot.

[4]KOMO source code: github.com/MarcToussaint/KOMO

In order to use the same AUC$f$ metric on KOMO, additional parameters are required to define planning failure by thresholding the resulted costs. First, collision constraint is limited at 0.01, anything more than that indicates a failure by collisions. Task-cost threshold, on the other hand, represents the tolerance for not fulfilling the requested tasks. It is therefore set to 1 where the corresponding AUC$f$ in test-set starts to saturate as depicted Figure 5.8. All path with costs within the thresholds are regarded as valid solution, the new performance metric "task-cost" is then derived from the averaged task-cost of all valid solutions.

| | $\mathcal{U}$ | GMM | | | KOMO |
|---|---|---|---|---|---|
| | | $45G$ | WA | WP | |
| $R^2$ | 0.921 | 0.964 | **0.965** | 0.964 | - |
| task-cost | - | - | - | - | 0.119 |
| Failure | 26 | 5 | 4 | **1** | 26 |
| Median | 0.357 | 0.359 | 0.365 | 0.365 | **0.140** |
| Min | 0.339 | 0.341 | 0.348 | 0.348 | **0.080** |
| AUC$f$ | 0.856 | 0.917 | 0.924 | **0.934** | 0.933 |

**Table 5.13:** Motion planners benchmark.

Table 5.13 depicts the performances comparison between RRT with uniform sampling $\mathcal{U}$, repetition sampling with 45 Gaussian components $45G$, best performing weights aggregation WA, best performing weights prediction WP and KOMO with the defined thresholds.

By observing "Median" and "Min", it is clear that KOMO is able to plan faster than all SMPs. However, KOMO has higher number of failure, yielding lower success rate ∼0.951% as compared to ∼0.998% from WP. As the result, KOMO and WP achieved comparable AUC$f$, one based-on faster optimisation cycles, the other one based-on higher reliability.

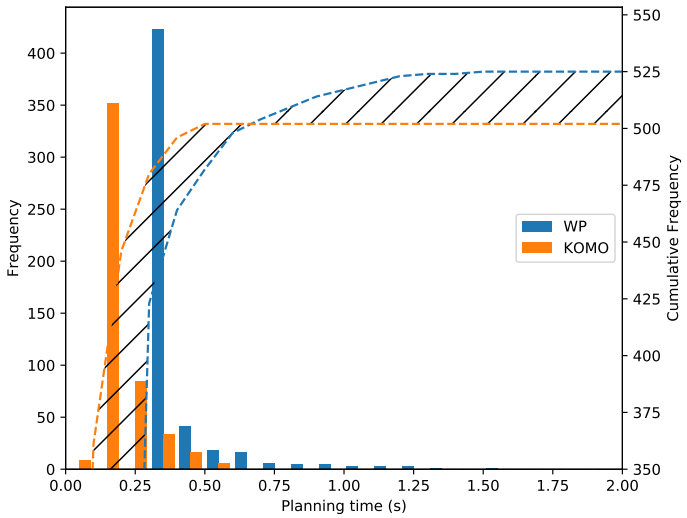



**Figure 5.9:** Comparison between WP and KOMO in AUC$f$.

# 6 Discussion and Conclusion

## 6.1 Results

### 6.1.1 Improved Repetition Sampling

While [2] only improves sampling efficiency, our proposal for repeating vanilla RRT based-on $R^2$ score during training has yielded not only higher success rate, but also improvements in solution's optimality. This nice property was inherited from training data to the fitted GMMs that are used during inference. It reduced RRT's dependency on path post-optimisation and eventually improved overall speed.

Moreover, performances remained competitive among GMMs with huge difference in number of components i.e. $\Delta 30G$ indicates that the redundancies of collected paths are at minimal. Together with the proposed key-configuration extraction technique, GMM with large number of Gaussian components is able to discover accurate and detailed density distributions in $\mathbb{C}$-space instead of over-fitted to noise.

### 6.1.2 Image Retrieval

The proposed multi-tasked decoder has proven to be better in image retrieval task compared to vanilla AE that only reconstruct its inputs. The additional semantic segmentation and boundary prediction tasks required the latent code to be more informative without increasing its size. This is beneficial because smaller latent code size $L = 64$ not only have higher efficiency in similarity search but also more robust to noise in latent-space.

Regularising latent code have turned out to be impairing for image retrieval task in the proposed architecture. Cosine similarity and BatchNorm layer have elevated the regulated magnitude of latent code from being the reasons of the deteriorations. Therefore, the negative effects of enforcing unit-Gaussian should be the results of bad implementation of regulariser loss function. Since the decoder was never treated as generative model, a well defined latent code distribution seem unnecessary anyway.

On the other hand, the attempt to reduce noise in latent-space by imposing embedding loss did not yield positive results. Although being less destructive that unit-Gaussian regularisation, this form of regularisation has the similar effects in diverging the learning.

When applied on real data, the proposed network that was trained entirely with synthetic data works reasonably well. This is due to the realistic depth image degradation and

input data augmentations. Moreover, the proposed adversarial example generation further improved the network's robustness on input variations, hence bridged the *reality-gap*.

Unfortunately, network's robustness does not directly extend to outside of what it was trained for. The network was able to handle unexpected variations in camera pose and unseen objects up to certain degree. Depending on the definition of individual applications and workspace scenarios, training data can always be customised accordingly. It is, however, always desirable to have better transferability without re-training.

### 6.1.3 Vision Assisted Biasing

With good image retrieval from database, weights aggregation method is able to collect related information from previous solutions and bias GMM adaptively. Compared to vanilla repetition sampling, it achieved better performance on GMMs with smaller number of Gaussian components. The deterioration in performance of GMM with large number of Gaussian components seem to be caused by the inability of representing highly non-linear similarity score by the simple cosine similarity and target proximity function.

Weights prediction method circumvented the similarity problems by output directly the weights for each Gaussian components. Built on top of the same network but without the hand-crafted similarity score, this method yielded better results than the previous. Besides, it utilised the detailed density estimation provided by GMM with larger number of Gaussian components and produced the best performance boost so far.

Despite the drawbacks, weights aggregation method is able to generate path with lower variance. By utilising existing solutions in database as lower bound, the resulted $R^2$ score is slightly better than weights prediction method that has no *safety-net* for worst performance.

Altering the upper bound for planning time $\overline{T}$, as well as task-cost threshold do changes the resulted AUC$f$ landscape when comparing between SMP and OMP. The purpose, however, is to show their respective strengths and weaknesses. The proposed weights prediction repetition sampling in RRT has the highest success rate and maintained competitive overall planning efficiency.

## 6.2 Insights

*Divide and conquer* was the original idea for this work. By *divide* it means dividing the planning scenes into multiple clusters base on scene similarity; by *conquer* it means fitting one GMM for each clusters. During inference, the cluster the queried scene belong to is determined before its respective GMM is deployed in the repetition sampling planner. While scene clustering can be done by e.g. K-Mean on latent code, or using AAE with class discriminator. Unfortunately, there is no direct correlation between scene and path similarity. 2 identical scenes can have 2 different manipulation tasks, which yielded 2

different paths. GMM fitted using a subset of dataset can contain the same amount of variance compared to GMM fitted using the entire dataset, hence provides no benefits.

The direct alternative was to cluster planning scenes base on path similarity. This can be done by having a classification output head at the end of encoder, right next to latent code. The classification uses target labels from parametrising each joint path with curve fitting method e.g. ridge regression, and then perform e.g. Agglomerative clustering base on those parameters. For this method to work, 2 conditions need to be fulfilled: there exists only one task per scene, and the task has only one solution. However, both condition cannot be fulfilled due to the nature of targeted robot workspace and stochasticity in SMP. Relaxing the second condition by reducing the number of parameters only lead to clustering that is based only on path's final pose and missed out all the important detours needed to get there.

It is possible to cluster planning scenes using all scene, task and path similarities, but number of clusters becomes an additional hyper-parameter to tune in this complex relationship. In contrast methods proposed in this work are more straight forward and easy to implement. In providing precise density estimation, large number of Gaussian components are used to fit one GMM using the entire dataset. Adaptively changing their weights resembled the effect of having multiple GMMs deployed at different time. The key-configurations used to fit GMM are extracted based-on the amount of detour contained in path, whose optimality is guarded by the $R^2$ score.

## 6.3 Alternatives

To further improve information encoding/transformation from image into latent code, object detection task can be used to multi-task the encoder. Additional prediction heads at multiple levels, as described in Single-shot multibox detector[78], can be added to encoder to improve scene understanding by increasing its sensitivity to localised features. Without extra overhead during runtime, this could be a faster alternative to [31, 34] which requires additional matchings for image patches when used for latent code generation.

In simplifying the entire pipeline, it will be interesting to adopt [16] by encoding all planning scene, task and path/key-configurations directly into latent codes. The conditional generative decoder plays the roles of: **1.** a density estimator in solution-space and **2.** solution sampler at the same time. Eventually it will eliminate the needs for GMM and repetition sampling.

Optimality of path generated from $\overline{\text{RRT}}$ can be further improved by replacing the `SMOOTHING` routine which perform joint-space interpolation with OMP local-methods[27]. Besides path optimisation, local-methods could also instil additional dynamic properties desirable according to applications. This upgrade could improve the overall optimality in the database, which will then benefit all subsequence processes in the pipeline.

From the perspective of industrial practicality, this work still lacks the *last-mile* in manipulation task. Target-poses in pick-tasks described in this work are actually just the pre-grasp

TCP pose. Object pose estimation[39] and grasp planning are additional modules needed to complete the manipulation task.

## 6.4 Conclusion

This work introduces vision assisted biasing for adaptive $\mathbb{C}$-space biased sampling in SMP. It is an extension for repetition sampling[2], which uses GMM to partition $\mathbb{C}$-space into feasible regions and offers fixed task oriented biases for a particular low-variance robot manipulation task. Our proposed methods use autoencoder to transform high dimensional planning scene in the form of depth image into low dimensional latent code. Weights Aggregation method then uses this latent code to search for nearest neighbours in the database and aggregates their solutions for a new set of GMM's weights. Weights prediction method, on the other hand, predicts the weights distribution directly at the bottleneck. Together with an improved pipeline for collecting training data and fitting GMM, this work not only enhances the performance of the vanilla repetition sampling, but also improved the solution's optimality without introducing significant overhead during inference. The results show a clear improvements from its predecessor and the competitiveness with its optimisation-based counterpart KOMO.

# Bibliography


[1] M. Hermann, T. Pentek, B. Otto. "Design principles for industrie 4.0 scenarios." In: *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. IEEE. 2016, pp. 3928–3937 (cit. on p. 9).

[2] P. Lehner, A. Albu-Schäffer. "Repetition sampling for efficiently planning similar constrained manipulation tasks." In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE. 2017, pp. 2851–2856 (cit. on pp. 9, 12, 13, 19, 23, 33, 34, 45, 46, 49, 52).

[3] D. H. Jacobson. "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach." In: *Journal of Optimization Theory and Applications* 2.6 (1968), pp. 411–440 (cit. on p. 11).

[4] E. Todorov, W. Li. "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems." In: *American Control Conference, 2005. Proceedings of the 2005*. IEEE. 2005, pp. 300–306 (cit. on p. 11).

[5] M. Toussaint. "Robot trajectory optimization using approximate inference." In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pp. 1049–1056 (cit. on p. 11).

[6] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, P. Abbeel. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization." In: *Robotics: science and systems*. Vol. 9. 1. Citeseer. 2013, pp. 1–10 (cit. on p. 11).

[7] M. Toussaint. "Newton methods for k-order Markov constrained motion problems." In: *arXiv preprint arXiv:1407.0414* (2014) (cit. on pp. 11, 47).

[8] L. Kavraki, P. Svestka, M. H. Overmars. *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*. Vol. 1994. Unknown Publisher, 1994 (cit. on p. 11).

[9] S. M. LaValle. "Rapidly-exploring random trees: A new tool for path planning." In: (1998) (cit. on p. 11).

[10] J. J. Kuffner, S. M. LaValle. "RRT-connect: An efficient approach to single-query path planning." In: *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. Vol. 2. IEEE. 2000, pp. 995–1001 (cit. on p. 11).

[11] S. Karaman, E. Frazzoli. "Sampling-based algorithms for optimal motion planning." In: *The international journal of robotics research* 30.7 (2011), pp. 846–894 (cit. on p. 11).

[12] C. Urmson, R. Simmons. "Approaches for heuristically biasing RRT growth." In: *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. Vol. 2. IEEE. 2003, pp. 1178–1183 (cit. on p. 11).

[13] V. Boor, M. H. Overmars, A. F. Van Der Stappen. "The Gaussian sampling strategy for probabilistic roadmap planners." In: *Robotics and automation, 1999. proceedings. 1999 ieee international conference on*. Vol. 2. IEEE. 1999, pp. 1018–1023 (cit. on p. 12).

[14] D. Hsu, T. Jiang, J. Reif, Z. Sun. "The bridge test for sampling narrow passages with probabilistic roadmap planners." In: *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 3. IEEE. 2003, pp. 4420–4426 (cit. on p. 12).

[15] Y. Yang, O. Brock. "Adapting the sampling distribution in PRM planners based on an approximated medial axis." In: *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. Vol. 5. IEEE. 2004, pp. 4405–4410 (cit. on p. 12).

[16] B. Ichter, J. Harrison, M. Pavone. "Learning sampling distributions for robot motion planning." In: *arXiv preprint arXiv:1709.05448* (2017) (cit. on pp. 12, 13, 51).

[17] T. F. Iversen, L.-P. Ellekilde. "Kernel density estimation based self-learning sampling strategy for motion planning of repetitive tasks." In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 1380–1387 (cit. on pp. 12, 13, 19).

[18] M. Phillips, B. J. Cohen, S. Chitta, M. Likhachev. "E-Graphs: Bootstrapping Planning with Experience Graphs." In: *Robotics: Science and Systems*. Vol. 5. 1. 2012 (cit. on p. 12).

[19] D. Coleman, I. A. Şucan, M. Moll, K. Okada, N. Correll. "Experience-based planning with sparse roadmap spanners." In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 900–905 (cit. on pp. 12, 13).

[20] M. Likhachev, G. J. Gordon, S. Thrun. "ARA*: Anytime A* with provable bounds on sub-optimality." In: *Advances in neural information processing systems*. 2004, pp. 767–774 (cit. on p. 12).

[21] M. Zucker, J. Kuffner, J. A. Bagnell. "Adaptive workspace biasing for sampling-based planners." In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 3757–3762 (cit. on pp. 12, 13).

[22] E. W. Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische mathematik* 1.1 (1959), pp. 269–271 (cit. on p. 13).

[23] L. Krammer, W. Granzer, W. Kastner. "A new approach for robot motion planning using rapidly-exploring randomized trees." In: *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*. IEEE. 2011, pp. 263–268 (cit. on p. 13).

[24] D. P. Kingma, M. Welling. "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on pp. 13, 20, 22, 32).

[25] D. Berenson, P. Abbeel, K. Goldberg. "A robot path planning framework that learns from experience." In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE. 2012, pp. 3671–3678 (cit. on p. 13).

[26] L.-P. Ellekilde, H. G. Petersen. "Motion planning efficient trajectories for industrial bin-picking." In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 991–1004 (cit. on p. 13).

[27] N. Jetchev, M. Toussaint. "Fast motion planning from experience: trajectory prediction for speeding up movement generation." In: *Autonomous Robots* 34.1-2 (2013), pp. 111–127 (cit. on pp. 13, 51).

[28] N. Jetchev, M. Toussaint. "Trajectory prediction in cluttered voxel environments." In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 2523–2528 (cit. on p. 14).

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

[30] A. Babenko, V. Lempitsky. "Aggregating local deep features for image retrieval." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1269–1277 (cit. on p. 14).

[31] G. Tolias, R. Sicre, H. Jégou. "Particular object retrieval with integral max-pooling of CNN activations." In: *arXiv preprint arXiv:1511.05879* (2015) (cit. on pp. 14, 15, 51).

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge." In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (cit. on pp. 14, 20).

[33] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky. "Neural codes for image retrieval." In: *European conference on computer vision*. Springer. 2014, pp. 584–599 (cit. on p. 14).

[34] Y. Liu, Y. Guo, S. Wu, M. S. Lew. "Deepindex for accurate and efficient image retrieval." In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ACM. 2015, pp. 43–50 (cit. on pp. 15, 51).

[35] V. Erin Liong, J. Lu, G. Wang, P. Moulin, J. Zhou. "Deep hashing for compact binary codes learning." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2475–2483 (cit. on p. 15).

[36] K. Lin, H.-F. Yang, J.-H. Hsiao, C.-S. Chen. "Deep learning of binary hash codes for fast image retrieval." In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2015, pp. 27–35 (cit. on p. 15).

[37] W. Kehl, F. Milletari, F. Tombari, S. Ilic, N. Navab. "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation." In: *European Conference on Computer Vision*. Springer. 2016, pp. 205–220 (cit. on p. 15).

[38] H. Zhang, Q. Cao. “Combined Holistic and Local Patches for Recovering 6D Object Pose.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2219–2227 (cit. on p. 15).

[39] M. Sundermeyer, E. Y. Puang, Z.-C. Marton, M. Durner, R. Triebel. “Learning Implicit Representations of 3D Object Orientations from RGB.” In: *Proceedings of the IEEE conference on robotics and Automation workshops*. Best Demo Award. 2018 (cit. on pp. 15, 52).

[40] A. Kendall, V. Badrinarayanan, R. Cipolla. “Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding.” In: *arXiv preprint arXiv:1511.02680* (2015) (cit. on p. 15).

[41] A. Nguyen, D. Kanoulas, D. G. Caldwell, N. G. Tsagarakis. “Detecting object affordances with convolutional neural networks.” In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 2765–2770 (cit. on p. 15).

[42] S. Gupta, R. Girshick, P. Arbeláez, J. Malik. “Learning rich features from RGB-D images for object detection and segmentation.” In: *European Conference on Computer Vision*. Springer. 2014, pp. 345–360 (cit. on p. 15).

[43] K. Simonyan, A. Zisserman. “Very deep convolutional networks for large-scale image recognition.” In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 15).

[44] E. Mohedano, K. McGuinness, N. E. O’Connor, A. Salvador, F. Marqués, X. Giro-i-Nieto. “Bags of local convolutional features for scalable instance search.” In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ACM. 2016, pp. 327–331 (cit. on p. 15).

[45] N. Garcia, G. Vogiatzis. “Learning Non-Metric Visual Similarity for Image Retrieval.” In: *arXiv preprint arXiv:1709.01353* (2017) (cit. on p. 16).

[46] S. Chopra, R. Hadsell, Y. LeCun. “Learning a similarity metric discriminatively, with application to face verification.” In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE. 2005, pp. 539–546 (cit. on p. 16).

[47] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, Y. Wu. “Learning fine-grained image similarity with deep ranking.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1386–1393 (cit. on p. 16).

[48] W. Li, R. Zhao, T. Xiao, X. Wang. “Deepreid: Deep filter pairing neural network for person re-identification.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 152–159 (cit. on p. 16).

[49] X. Han, T. Leung, Y. Jia, R. Sukthankar, A. C. Berg. “Matchnet: Unifying feature and metric learning for patch-based matching.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3279–3286 (cit. on p. 16).

[50] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 20).

[51] A. Krizhevsky, I. Sutskever, G. E. Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on p. 20).

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. "Going deeper with convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9 (cit. on p. 20).

[53] K. He, X. Zhang, S. Ren, J. Sun. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on p. 20).

[54] G. Huang, Z. Liu, K. Q. Weinberger, L. van der Maaten. "Densely connected convolutional networks." In: *arXiv preprint arXiv:1608.06993* (2016) (cit. on pp. 20, 26, 27).

[55] D. E. Rumelhart, G. E. Hinton, R. J. Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (cit. on p. 20).

[56] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of Machine Learning Research* 11.Dec (2010), pp. 3371–3408 (cit. on pp. 20, 29).

[57] S. Rifai, P. Vincent, X. Muller, X. Glorot, 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 (cit. on p. 20).

[58] I. J. Goodfellow, J. Shlens, C. Szegedy. "Explaining and harnessing adversarial examples." In: *arXiv preprint arXiv:1412.6572* (2014) (cit. on p. 21).

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on p. 21).

[60] A. Radford, L. Metz, S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." In: *arXiv preprint arXiv:1511.06434* (2015) (cit. on p. 21).

[61] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey. "Adversarial autoencoders." In: *arXiv preprint arXiv:1511.05644* (2015) (cit. on pp. 21, 22, 32).

[62] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, F. Roli. "Evasion attacks against machine learning at test time." In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 387–402 (cit. on p. 21).

[63] A. Kurakin, I. Goodfellow, S. Bengio. “Adversarial machine learning at scale.” In: *arXiv preprint arXiv:1611.01236* (2016) (cit. on pp. 21, 30).

[64] F. Tramèr, A. Kurakin, N. Papernot, D. Boneh, P. McDaniel. “Ensemble Adversarial Training: Attacks and Defenses.” In: *arXiv preprint arXiv:1705.07204* (2017) (cit. on p. 21).

[65] Y. Bengio, A. Courville, P. Vincent. “Representation learning: A review and new perspectives.” In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828 (cit. on p. 22).

[66] I. Tolstikhin, O. Bousquet, S. Gelly, B. Schoelkopf. “Wasserstein Auto-Encoders.” In: *arXiv preprint arXiv:1711.01558* (2017) (cit. on p. 22).

[67] K. He, X. Zhang, S. Ren, J. Sun. “Identity mappings in deep residual networks.” In: *European Conference on Computer Vision*. Springer. 2016, pp. 630–645 (cit. on p. 26).

[68] F. Yu, V. Koltun. “Multi-scale context aggregation by dilated convolutions.” In: *arXiv preprint arXiv:1511.07122* (2015) (cit. on p. 27).

[69] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. “Dropout: A simple way to prevent neural networks from overfitting.” In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on p. 27).

[70] K. Perlin. “Improving noise.” In: *ACM Transactions on Graphics (TOG)*. Vol. 21. 3. ACM. 2002, pp. 681–682 (cit. on pp. 29, 39).

[71] Z. Wu, C. Shen, A. v. d. Hengel. “Bridging category-level and instance-level semantic image segmentation.” In: *arXiv preprint arXiv:1605.06885* (2016) (cit. on p. 32).

[72] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár. “Focal loss for dense object detection.” In: *arXiv preprint arXiv:1708.02002* (2017) (cit. on p. 32).

[73] A. Dömel, S. Kriegel, M. Kaßecker, M. Brucker, T. Bodenmüller, M. Suppa. “Toward fully autonomous mobile manipulation for industrial environments.” In: *International Journal of Advanced Robotic Systems* 14.4 (2017), p. 1729881417718588 (cit. on p. 37).

[74] K. He, X. Zhang, S. Ren, J. Sun. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.” In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034 (cit. on p. 40).

[75] S. J. Reddi, S. Kale, S. Kumar. “On the convergence of adam and beyond.” In: (2018) (cit. on p. 40).

[76] D. Kingma, J. Ba. “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 40).

[77] I. Loshchilov, F. Hutter. “Sgdr: Stochastic gradient descent with warm restarts.” In: *arXiv preprint arXiv:1608.03983* (2016) (cit. on p. 40).

[78] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg. “Ssd: Single shot multibox detector.” In: *European conference on computer vision*. Springer. 2016, pp. 21–37 (cit. on p. 51).

# Appendix

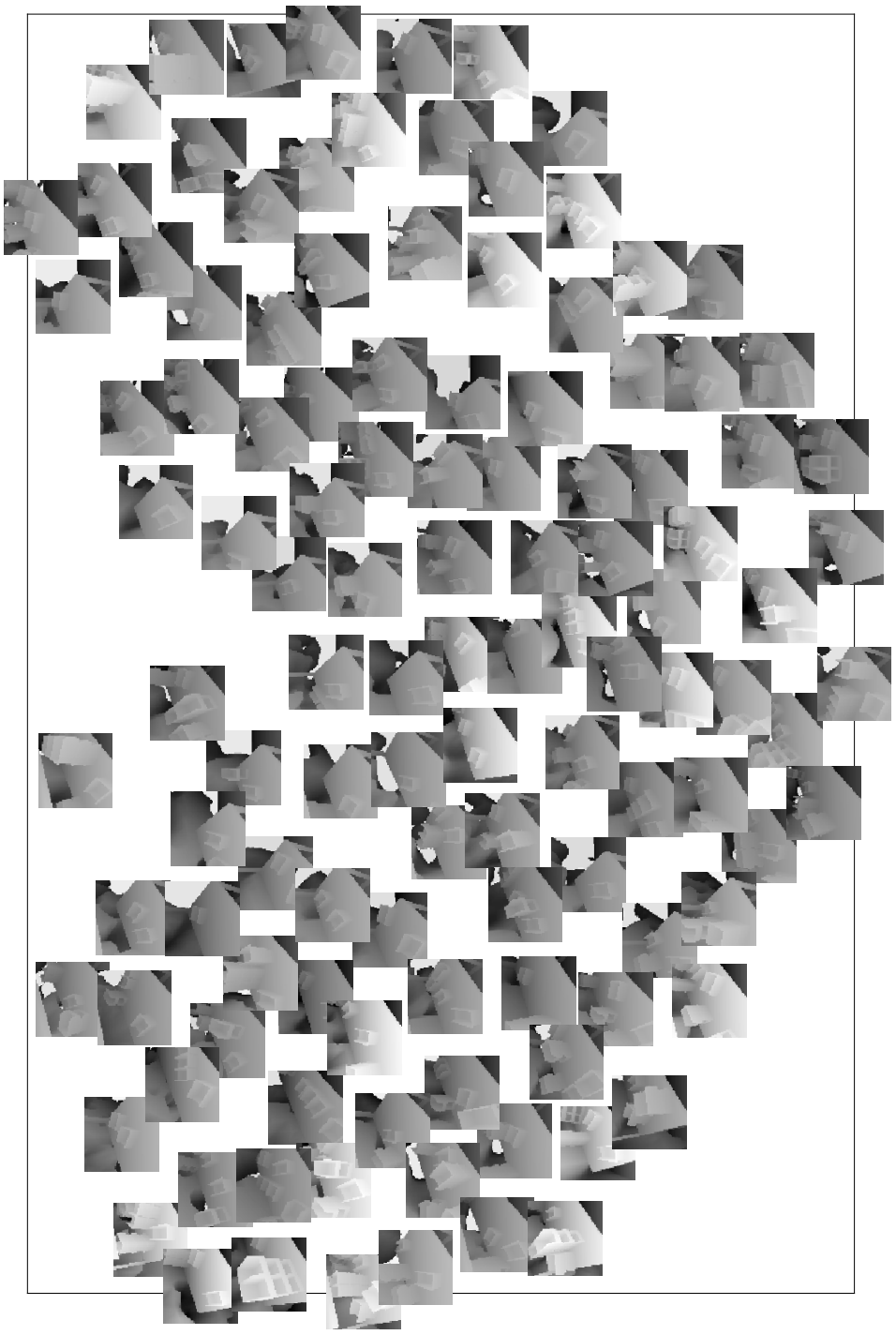


**Figure .1:** Database latent code 2D t-SNE plot.

Test-set results when failure are excluded.

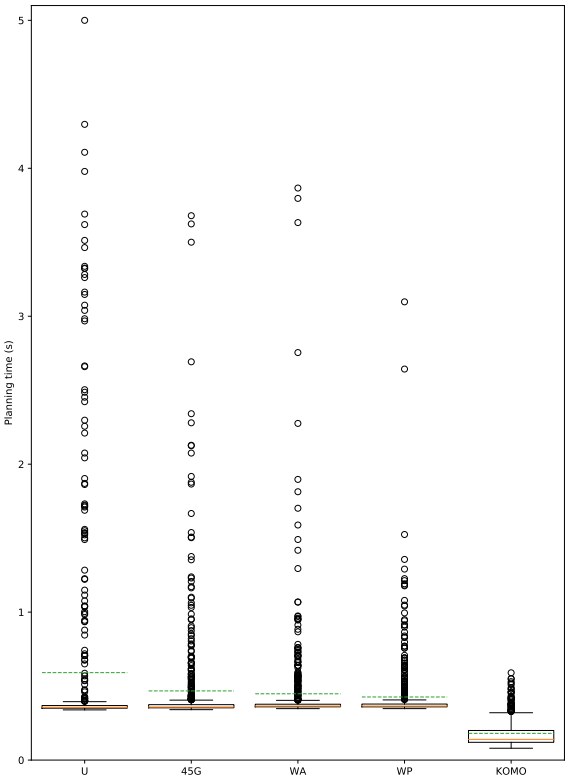



**Figure .2:** Box plot.

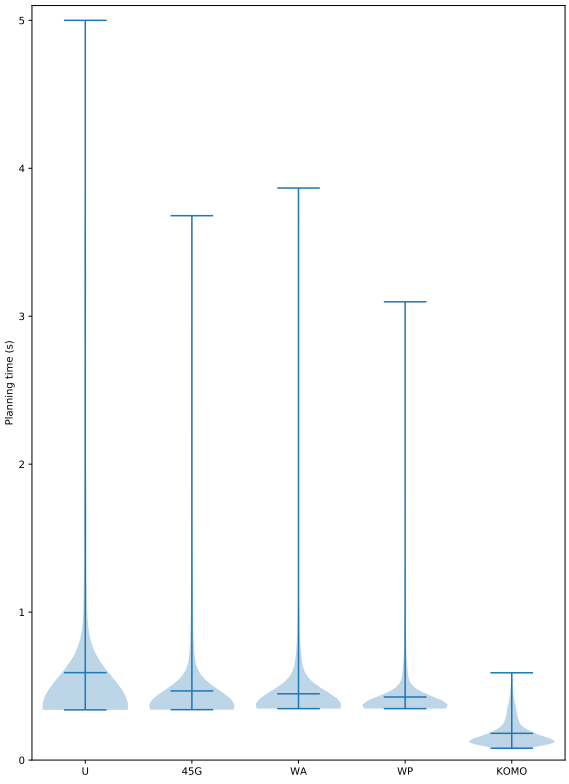



**Figure .3:** Violin plot.

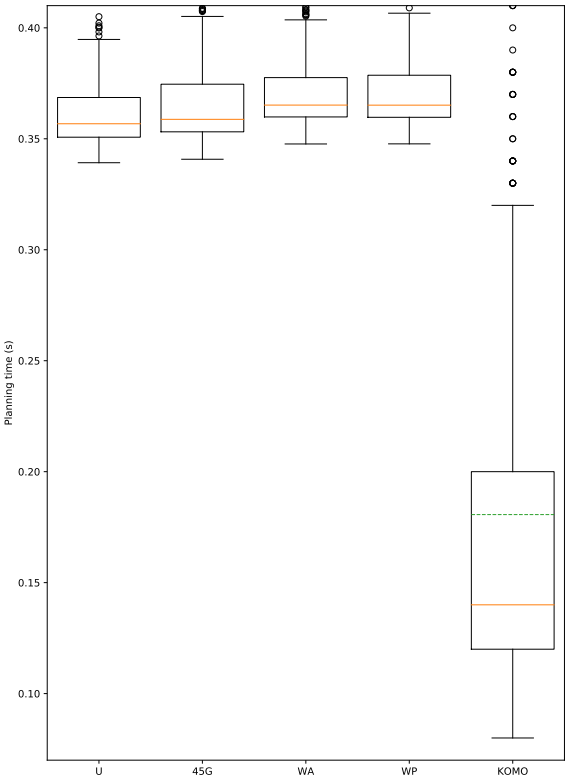



**Figure .4:** Zoomed box plot.

| | mean | std | max |
|---|---|---|---|
| $\mathcal{U}$ | 0.591 | 0.700 | 5.217 |
| $45G$ | 0.467 | 0.377 | 3.680 |
| WA | 0.448 | 0.335 | 3.866 |
| WP | 0.426 | 0.216 | 3.097 |
| KOMO | 0.181 | 0.092 | 0.590 |

**Table .1:** Planning time(s) statistic.

Test-set results when failure are included with planning-time $\leftarrow 5s$.

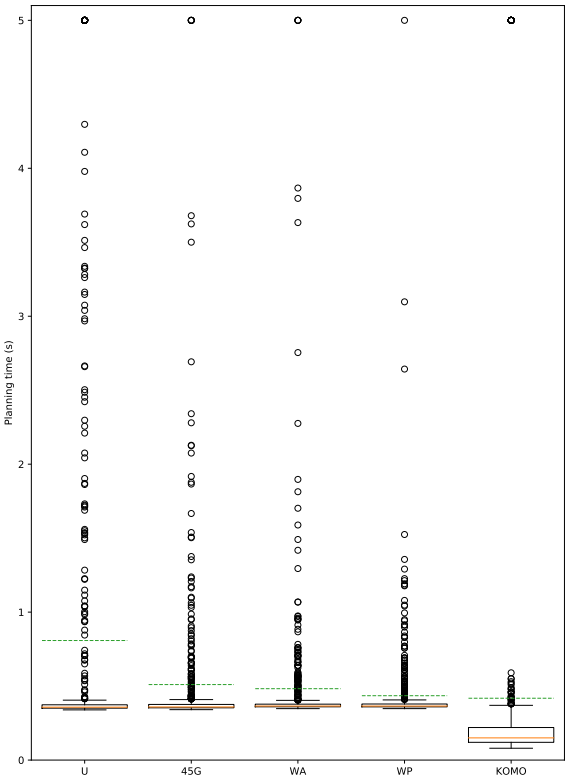



**Figure .5:** Box plot.

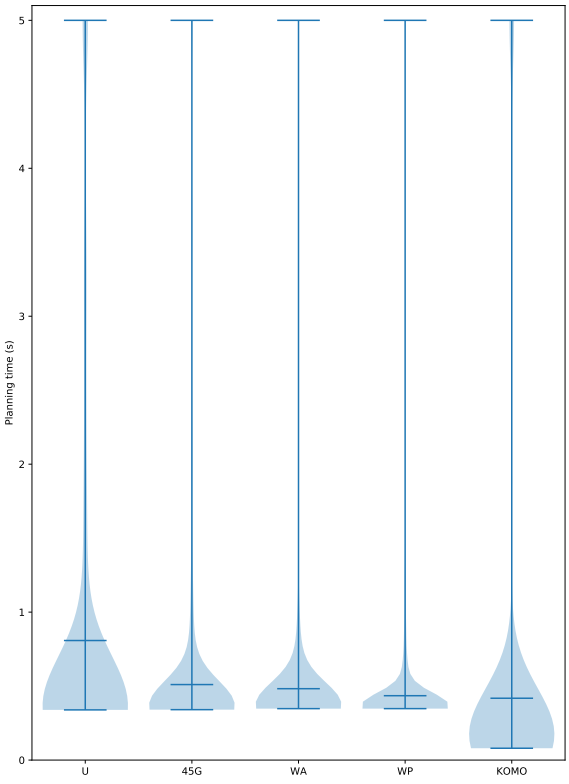



**Figure .6:** Violin plot.

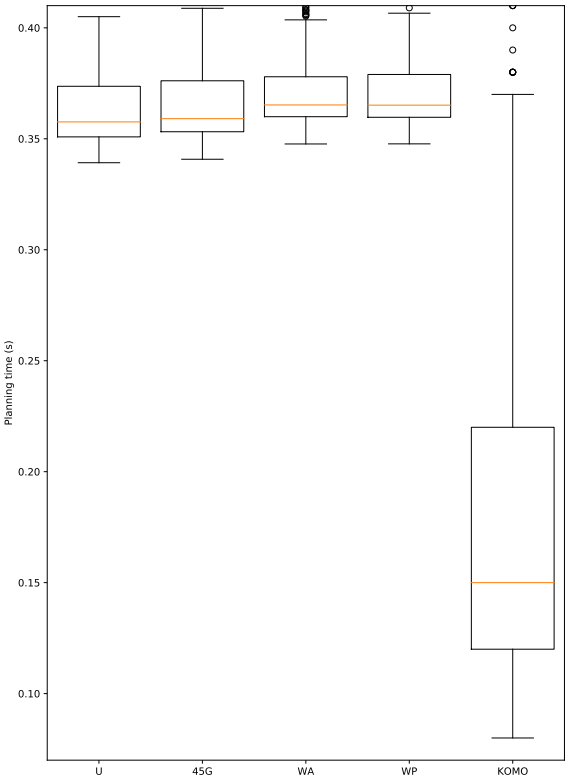



**Figure .7:** Zoomed box plot.

| | mean | std | max |
|---|---|---|---|
| $\mathcal{U}$ | 0.808 | 1.173 | 5.000 |
| $45G$ | 0.510 | 0.577 | 5.000 |
| WA | 0.483 | 0.517 | 5.000 |
| WP | 0.435 | 0.294 | 5.000 |
| KOMO | 0.418 | 1.047 | 5.000 |

**Table .2:** Planning time(s) statistic.

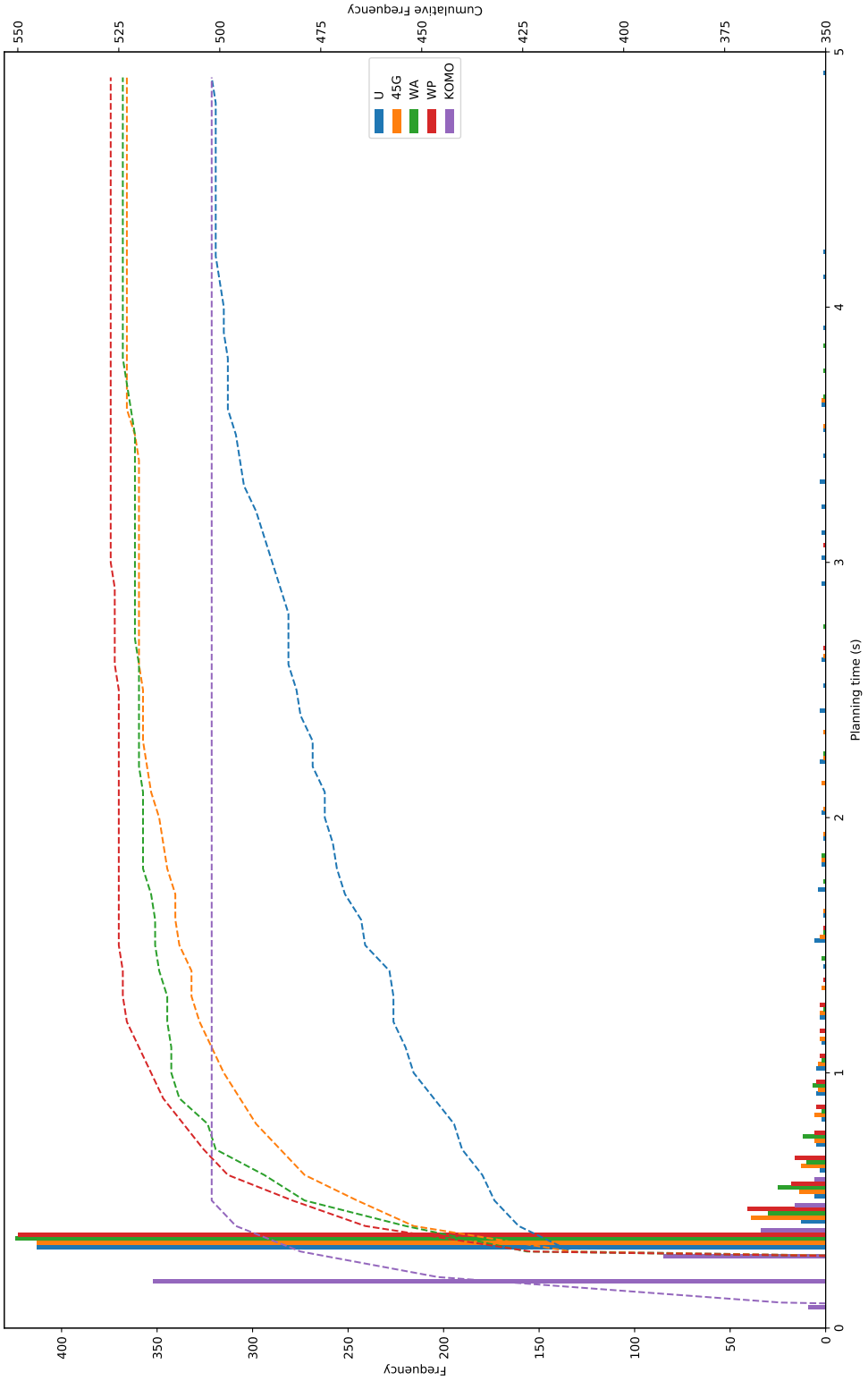
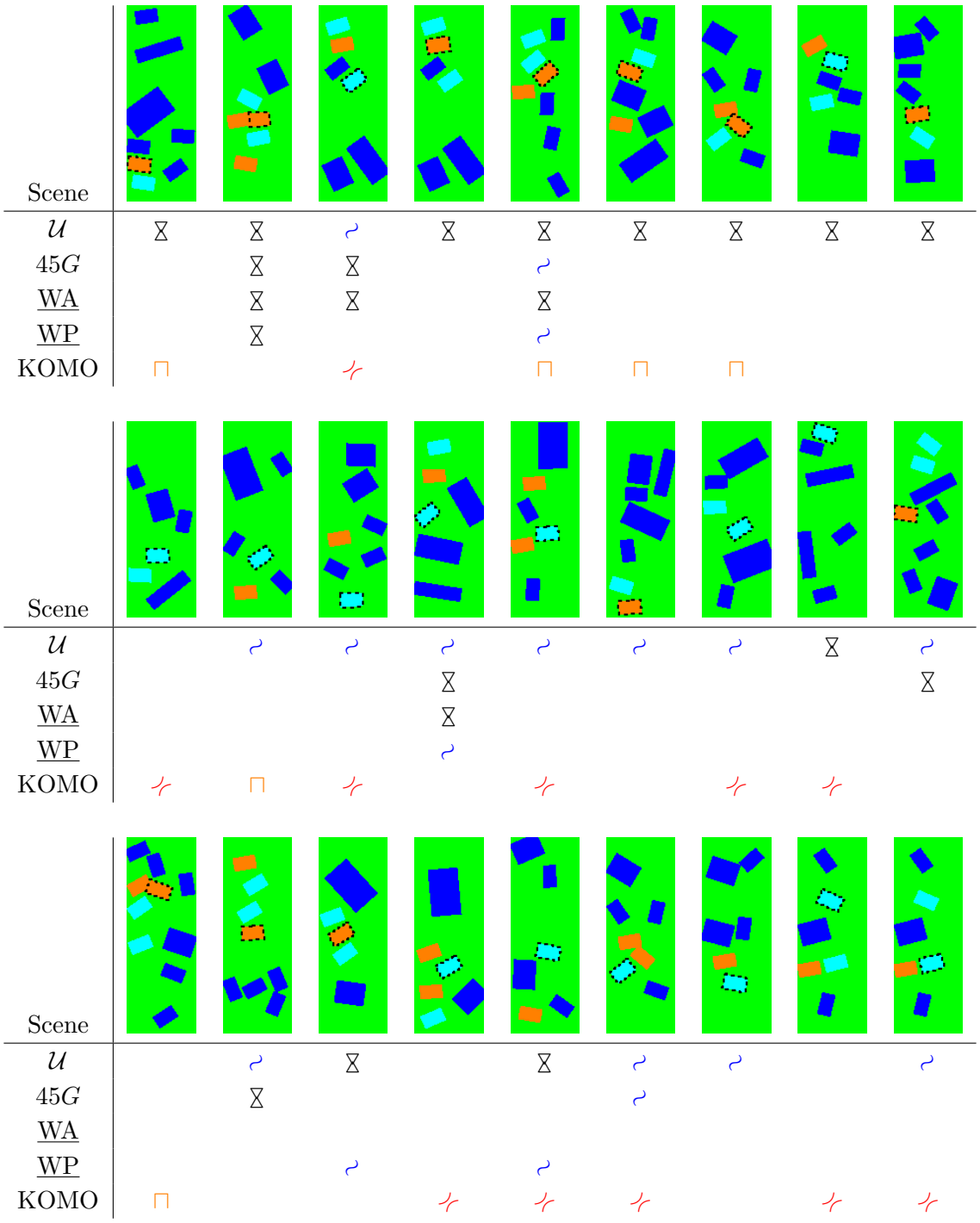



**Figure .8:** Planning time histograms and corresponding cumulative frequencies.




| | | | |
|---|---|---|---|
| $t_{search}$ | $>$ | 5s | ⧖ |
| $R^2$ | $<$ | 0.9 | ⤳ |
| task-cost | $>$ | 1 | ⊓ |
| collision-constraint | $>$ | 0.01 | ↯ |

**Table .3:** Test-set failure analysis. For detailed scene breakdown refer Figure 5.2.

## Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Datum und Unterschrift:

## Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Date and Signature: