Master's Thesis

Vision Based Robotic Manipulation using Adaptive Trajectory search in Latent Space

Rohith Kalathur

Submitted to Universität Siegen,
Department of Mechatronics
in partial fullfilment of the requirements for the degree
of Master of Science in Mechatronics

Supervised by

Prof.Dr.Michael Möller
Maximilian Durner

December 2019

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Rohith Kalathur

Abstract

This thesis aims at developing algorithms for vision based robotic manipulation. Given a robot arm with RGB cameras mounted on the TCP(Tool Centre Point) of the robot arm, we aim to find a relative pose between the TCP and a predefined view of the object of interest. An encoder is used to iteratively refine the cosine similarity between the latent encoding of the image from the RGB camera and the target image.The set up was simulated in Gazebo with the robot arm and the camera mounted on the TCP, together with the object placed on a table. Initially a random search algorithm was implemented followed by a CMA-ES(Covariance Matrix Adaptation -Evolutionary Strategy) algorithm. Both random search and CMAE-ES converge to the the threshold similarity within a range of 40-60 iterations with a success rate of 95 percent.

During the trajectory search, the robot arm has to avoid the unreachable poses of the TCP of the robot arm. CMAE-ES, although avoids unreachable poses with the evolution of generations, it doesn't converge to the threshold cosine similarity. Hence a SAC(Soft Actor Critic) reinforcement learning algorithm was implemented. The SAC algorithm successfully avoids the unreachable poses of the TCP of the robot arm during the trajectory search and also reaches the threshold cosine similarity with only 4-6 iterations on average with 100 percent success rate and hence outperforms CMA-ES and random search algorithms.

Acknowledgements

I convey my thanks to Prof.Dr.Michael Möller for supervising my thesis.I also convey my thanks to Maximilian Durner, my supervisor at DLR for his constant support and guidance.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motivation

Robotic manipulation is one of the deeply researched fields in recent years. Robotic manipulation refers to when a robot with its end effectors can successfully interact with its environment to accomplish a particular task. Robots capable of successfully manipulating the environment have been employed in a multitude of areas such as factories, for assembling of parts, restaurants, for catering to the public, logistics and healthcare to name a few. As a result, robotic manipulators are required to operate in dynamic environments and with multitude of task agnostic or task specific skills. This demands the robotic manipulators to become increasingly autonomous to adapt to varying tasks or dynamically changing environment. The ability of the robotic system to perceive their environment is the key element for achieving highly autonomous behaviour.

However, when it comes to manipulation, the robotic manipulators are required to function in dynamic environments. The robotic manipulators need to adapt to the changes in the environment like change of work space, inaccuracies in the control of the robotic arm or perturbations of the robotic arm itself. These requirements prompt for articulate sensory information of the surroundings and also learning task specific or task agnostic skills to be able to successfully adapt to a dynamic environment.

With the emergence of advanced sensors like Red- Green-Blue (RGB) and RGB-D

cameras, vision based manipulation has become increasingly ubiquitous. With the integration of various sensors together with the learning based algorithms, robotic arms are moving towards becoming more autonomous. With the application of deep learning algorithms in vision related tasks like object detection, object recognition and pose estimation, the perception system required for various robotic manipulation tasks like grasping, pick and place have been enhanced. This development in the perception of robotic manipulator systems has paved the way for various learning based methods like Imitation learning, Reinforcement learning etc. In traditional methods, the robotics manipulators use hand crafted algorithms for motion planning, grasping and manipulation as whole. By leveraging the knowledge of the environment, robotic manipulators are able to learn task specific behaviours for tasks such as pick and place in contrast to the traditional methods.

1.2 Problem Statement

This thesis aims at developing methods to leverage the sensory information in the form of RGB images to successfully guide the robotic arm from an initial view of the object to a target view of the image of the object required for manipulation.

To this end, the problem statement can be summarised as: given a robot arm with RGB-cameras mounted on the on the TCP (Tool Centre Point) of the robot arm, aim to find a relative target pose between the object of interest and the robot arm itself, utilising the latent embedding of the RGB image generated by an encoder to iteratively find a predefined view of the object.

1.3 Organization of the Report

The following chapters of this work are structured as follows:

- **Chapter 2** includes a brief background about the methods in robotic grasping with emphasis on vision based robotic manipulation techniques. Furthermore, related work to this thesis in discussed.

- **Chapter 3** describes the methodology of the simulation for the robotic arm, mainly focusing on the design of software architecture and modules. The metrics for the experiments are also presented.

- **Chapter 4** describes the algorithms in detail which are employed for the problem. Together with the algorithms the implementation and adaptation of these to our problem is also discussed.
- **Chapter 5** discusses the various experiments carried out for each of the algorithms. Further, the algorithms are compared according to the metrics and their suitability is also presented.

2

Background

2.1 Robotic Grasping

Although in recent years there has been a great progress in the field of robotics, robust robotic grasping remains to be one of the main challenges. As identified in [21],[8], the classical robotic grasping pipeline is essentially composed of the following sub-modules.

- **Object localisation module and Grasp Detection module :** To detect the potential grasp candidates from the perception of the environment from the available sensor data and to localise them with respect to a frame of reference.Then the task is to detect suitable grasping points for the localised objects.

- **Grasp Planning module and Execution module :** Once the potential grasp candidates have been identified, its necessary to plan an optimal trajectory to reach the grasp location identified in the grasp detection module.

- **Grasp Trajectory Execution module :** To execute the trajectory planned from the planning module using various controller modules, which delves into the filed of classical control systems theory.

Sahbani et al.[32] identifies stability, task compatibility and adaptability to previously unseen objects as three basic criteria for a good grasping strategy. The available

methods for each of the above mentioned modules can be broadly categorized into analytical methods and empirical methods. Analytical methods determine good grasps based on the kinematics and dynamics modelling of good contact locations for a particular object or a task. However, empirical methods rely more on perception and cognition of the target object or a grasping task[28]. In the following subsections, existing methodologies, analytical and empirical for each of the above stated modules in the grasping pipeline is discussed briefly. Additionally, as reviewed in [8], the grasp detection system can be categorized as object localisation task, 6D pose estimation of the object of interest and consequently suitable grasp point detection for the localised object. Also, the three tasks mentioned above can be accomplished independently or in an end to end manner or in combinations[8]. With the integration of high performance optical sensors such as RGB and RGB-D cameras with the present generation robots, vision based robotic grasping is ubiquitous. Hence, in the subsequent sections each module in the grasp detection pipeline, with focus on vision based grasping is presented briefly. More emphasis on the planning module is given, since the present work concerns with this part of the above stated grasping pipeline.

2.1.1 Grasp Detection

Grasp detection also known as Grasp synthesis refers to determining stable grasp configurations for a target object. Redmon and Angelova[28] provides a detailed explanation of what constitutes a stable grasp in terms of *force closure* and *form closure* properties. Essentially, force closure refers to the condition when all the tangential/normal forces and the moment/torsional forces at the contact points of target object are in equilibrium[28], whereas *form closure* is more geometrical in notion and is achieved when a grasp achieves force closure, when the contact points on the target object are friction less[28]. Therefore, the problem of grasp synthesis translates into generating the contact points depending on the fingers on the gripper, which satisfies *force closure* and/or *form closure* properties.

In analytical methods for grasp synthesis Liu and Wang[24] model, the problem as a Linear Programming problem to generate grasps with form closure such that "the origin of the wrench space lies within the convex hull of the primitive contact wrenches[24]" Ponce et al.[27] addressing the problem of generating grasps with force

closure, utilizing the parametric representation of a point on a plane face by two linear parameters, formulated necessary linear conditions for three contact point and four contact point grasp. Thereby reducing the set of constraints in the contact positions to a set of linear inequalities.

Empirical approaches can be two faceted, as identified in [32]. One category of methods try to mimic the human behaviour, related to grasping. Fischer et al.[10] developed a setup to control the robot hand by telemanipulation to realize grasping, Fischer et al. achieves this by employing a neural network to learn nonlinear calibration of the dataglove [10] for mapping of human and artificial hand work-spaces.

2.1.2 Grasp Planning or Motion Planning

The key task of this module is to plan a path from the robot hand to the object to be grasped at the grasping points and are traditionally implemented using open loop methods. There exists numerous paths or trajectories from the robot hand to the grasping point. The key task of this module is to find an efficient or optimised trajectory among many. The motion planning module thus also has to incorporate the environment while planning, such that the robotic hand avoids any obstacles while reaching the grasping points for the object.

The traditional method for motion planning are methods to generate motion primitives. One such method is DMP(Dynamic Motion Primitives), which was first introduced by Ijspeert et al.[20]. As non-linear systems such as the control motors of the robot often have emergent dynamical behaviour, DMP deals with goal oriented behaviour of nonlinear dynamical systems. Since non-linear systems are parameter sensitive predicting their long term behaviour with respect to the parameters is difficult and quite expensive in computation. Hence, DMP model the non-linear dynamical systems by transforming them into a weakly non-linear system governed by the attractor dynamics by introducing a forcing term that drives the non-linear system into a required goal oriented behaviour. The transient behaviour of the non-linear system is shaped by the forcing term of the attractor and hence, they encode kinematic control policies as control primitives for the non-linear differential equations of the dynamical system.

Other learning based methods include Imitation learning or Reinforcement learn-

ing for motion planning. Imitation learning also known as learning by demonstration is a method where the robot learns to grasp by learning the mapping of successful human reach and grasp from numerous demonstration. If the target object to be grasped is already available in the database the grasp are directly executed from the previously stored demonstration and if the target object is similar to a trained object, the object is compared to the objects in the database and closely matching grasp is selected. On the other hand, Reinforcement learning is a method of learning technique, where the algorithm is learned by interacting with the environment. Reinforcement learning techniques have been widely used in recent years to learn task specific and task agnostic robot skills.

2.2 Vision-based Robotic grasping

With the advancement of Computer vision, specifically due to the influence of powerful Artificial Neural Networks based Deep-Learning algorithms the perception of the robotic vision has been enhanced. Consequently, vision based robotic grasping pipeline broadly consists of object localisation, 6D pose estimation of the target object, grasp detection and grasp planning [32]. From the raw sensor data of the RGB or RGB-D cameras, object localisation is to locate the object of interest in the image. 6D pose estimation deals with obtaining the rotation as well as the translation of the object of interest in a specific frame of reference. Grasp detection estimates the configuration of the gripper with respect to the object of interest. Grasp planning is then to estimate an optimal path to grasp the object of interest. Furthermore, many approaches blur the hard distinction between the traditional pipeline of robotic grasping by combining one or more tasks as in [40] which constitute end to end grasping techniques.

2.2.1 Object Localisation

Object localisation deals with locating the object of interest for grasping with respect a frame of reference. In computer vision, it is normally referred to as object detection. In some cases along with the detecting the object of interest, image segmentation techniques are employed for localising objects in cluttered environments. The task of object detection translates into predicting bounding boxes in the image

encompassing the object of interest. In recent years with emergence of powerful Deep Learning algorithms Convolutional Neural Networks(CNNs) are ubiquitous for object detection and classification.

Girshick et al. [14] proposed an algorithm which they call R-CNN, where instead of running convolutions on huge number of regions as in a traditional Convolutional Neural Network, R-CNN generates candidate regions and combining them by using a greedy algorithm, such as selective search before passing them to a CNN based feature extractor. Hence the name Region-Proposal CNNs. Improving upon R-CNN the same author Girshick2015 [13] proposed an algorithm called Fast R-CNN, where regions of proposal are generated on the convolutional feature map produced by running CNN on the image and then pooling them by a layer called ROI(Region of interest pooling), before passing them onto a softmax layer for classification and prediction of the bounding boxes. Some of the other variants of object detectors include Mask R-CNN by He et al. [18], YOLO(You Only Look Once) by Redmon et al.[29].

Segmentation refers to classifying each pixel in an image as belonging to a particular class. Long et al.[25] achieved semantic segmentation of images by employing "fully convolutional networks"[25] trained in an end to end, pixel to pixel fashion which takes an image of arbitrary size and recreates the image with pixel to pixel labels. Following this, there have been many variants, some of which include Segnet by Badrinarayanan et al. [2] which employ an Encoder-Decoder architecture, U-net by Ronneberger et al.[30] and Deep-Lab by Chen et al.[6].

2.2.2 6-D Pose Estimation

6-D pose of an object refers to the location and orientation of an object in space. The knowledge of a 6-D pose of an object is especially crucial for robotic manipulation and grasping, as the robot needs to find the object and grasp it in a suitable orientation for stable grasps. Several methods are proposed to address the problem of pose estimation. While some of the methods only estimate the 6-D pose, other methods also achieve object detection and classification along with pose estimation. As identified in [8], 6-D pose estimation methods can be Correspondence based methods, Template- based methods, Voting-based methods and methods which

combine object detection and pose estimation.

Correspondence based methods match 2-D feature points on the images rendered by projecting a 3-D model of the object from various angles, with the image that is observed [36]. This method is suitable for objects with rich textures as the feature points are better detected and are not so reliable for texture-less objects. Some methods find correspondences between the observed 3D point cloud of the object and the 3D point cloud of the full object. Wong et al. proposed an integrated approach, which do object recognition and pose estimation by their method which they call "SegICP"[37]. In this approach, semantic segmentation of the scene is overlaid on the depth image of the scene and 3-D correspondence is used to estimate the 6-D pose of the object.

Hinterstoißer et al.[19] proposed a template based approach where a 3-D object is represented by a limited set of templates from different views, which are built from densely sampled image gradients and surface normals from the depth map. On finding one of the templates it provides object localisation as well as 6-D pose of the object.

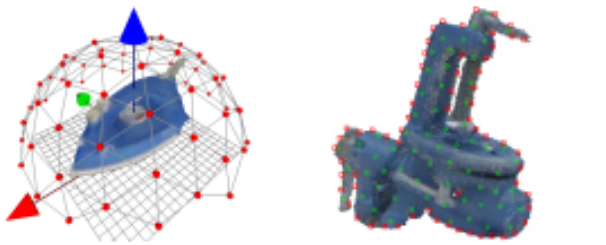


Figure 2.1: **Left:** Sampling of view points from upper hemisphere to generate templates. Camera centers are represented by the red vertices. **Right:** Colour gradients are represented in red and surface normals in green which are the selected features for template matching[19]

.

Brachmann et al.[4] proposed a voting based approach where an intermediate representation of the object is learned in the form of object co-ordinates of a dense 3-D object paired with a dense class labelling. The correspondence between the object and 3-D model and the image is established by each co-ordinate prediction of the RGB-D image. The proposed method works on texture-less objects and also under different lighting conditions.

Lot of efforts have been made in recent years for estimating pose combined with object detection in a single stage pipeline. These methods are normally referred to as regression based methods. Xiang et al. introduces a convolutional neural network called "POSE-CNN"[38] for 6-D pose estimation combined object detection. The trained network estimates the 3D translation by locating the object's center in the image and predicting the distance of the center from the camera. The 3D rotation of the object is estimated by regressing the quaternion representation of the 3D rotation of the object. They also introduced a novel loss function to handle object symmetries. Do et al. introduced a network based on MASK-RCNN[18] named "Deep-6D Pose" to jointly detect, segment and estimate 6D poses of objects from an RGB image. They extended the Mask-RCNN with a novel pose estimation branch to directly regress the 6D poses of the objects. The network is trained jointly by incorporating the loss from all the detection, segmentation as well as the pose estimation branch.

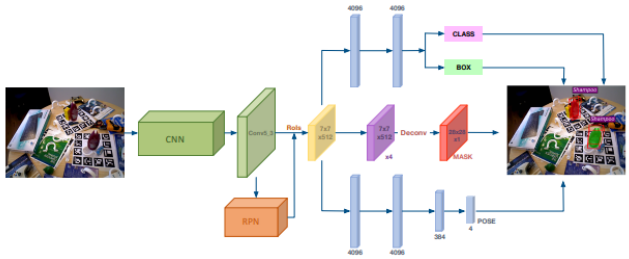


Figure 2.2: Network architecture for "Deep-6D Pose"[7]

.

2.2.3 Motion Planning

Once the 6D pose of the object is estimated and the grasping configuration is estimated, motion planning is to plan a suitable trajectory for the robotic arm to reach the desired point for grasping. Although multitude of trajectories could exist to reach to a desired point, the robotic arm is constrained by unreachable configuration limited by the joint angles of the robot. Hence, intelligent planning to reach to the target point is essential. There are several traditional methods for motion planning. Learning based methods employ techniques like Imitation learning and Reinforcement

learning for the same.

Ben Amor et al.[3] addressed the problem by applying imitation learning techniques, where a low dimensional latent space encoding is created for different human grasps types, the learned contact points for an object are warped onto a novel object by their algorithm called Contact Warping[3] and by using DMP a whole end to end grasp is synthesised.

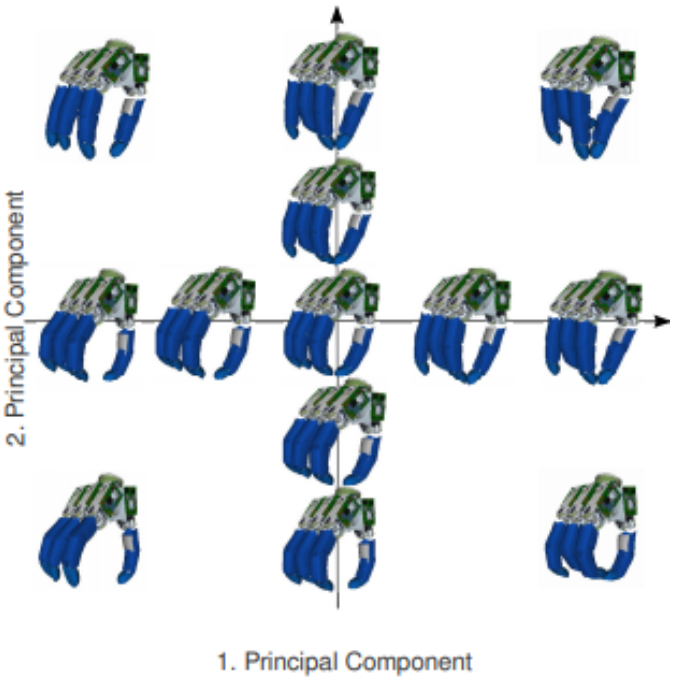


Figure 2.3: Principal Components of the latent space encoding for different human type grasps[3]

.

Kwiatkowski and Lipsonc[22] proposed an algorithm for self modelling of a robot in a task agnostic manner using reinforcement learning. Due to the physical constraints of the robot arm, self collision and other factors the robot may not be able to always execute the joint angles commanded by the motors.In this approach, the robot learns from the action-sensation pairs where action refers to the motor angle commands and sensations refer to the absolute co-ordinate of the end effector by moving through many trajectories. The robot learns feasibility of execution of motor commands to reach a particular end effector positions and the model can be

adapted to any particular task.

2.3 Related Work

Considerable efforts have been made in recent years for developing end-to-end mechanisms for learning robotic tasks as discussed in the above sections. With the advancement of the sensors and the amount of sensory perceptual data available, many methods map the sensory data to the direct task related commands for a robot.

In the approach introduced by Yun Jiang et al.[40], predict the 7 dimensional grasping configuration i.e., the location of the target object, orientation of the target object and the gripper opening width which they call a "grasping rectangle"[40]. Using the disparity maps from the depth camera along with the RGB image as features used for inferring the grasping rectangle using a SVM.

In a similar approach Redmon and Angelova[28] predict a 5 dimensional parametric rectangle from the RGB image with the centre, orientation of the rectangle relative to the horizontal axis, height and width of the rectangle as the grasp parameters. Redmon and Angelova use a convolutional neural network as a feature extractor from the RGB image with last layer predicting the 5 parameters of the rectangle.

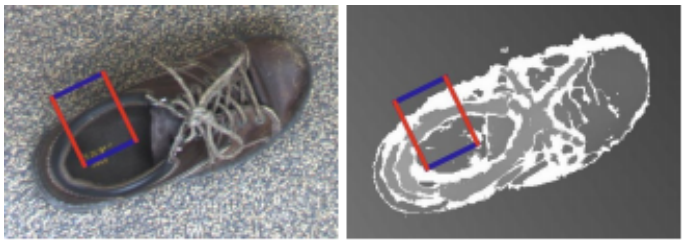


Figure 2.4: "Grasping Rectangle" prediction by combining RGB image with depth image[40]

Levine et al.[23] proposed a method for guided policy search for robotic manipulation tasks, where the policy or manipulation is learnt by mapping raw images from the cameras on the robot to the direct motor torque command at the robots joints. For this they use, a CNN to extract features from the images obtained from the camera to compute the expected position of the object. In the next stage, the robot configuration is concatenated and passed on to three fully connected layers to produce motor torques of the robot directly.

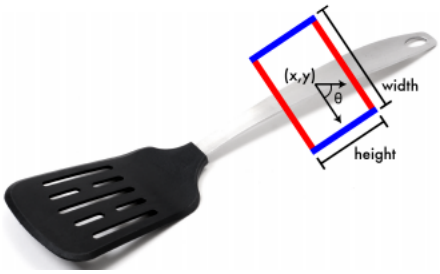


Figure 2.5: 5D grasping rectangle prediction from the RGB image[28]

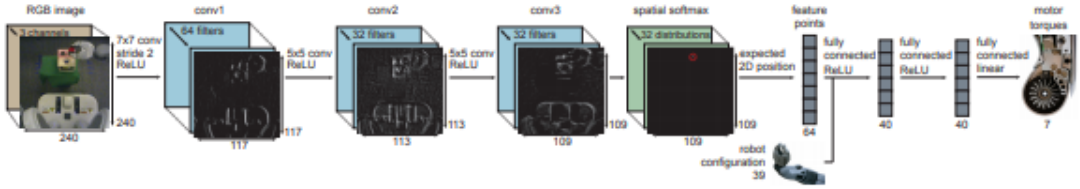


Figure 2.6: The network architecture for predicting motor torques.[23].The RGB image is passed through three convolutional layers, followed by a spatial softmax layer and a expected position layer to extract feature points from pixel-wise features to compute the expected 2D position of the object. Finally the feature vector is concatenated with robot configuration and passed on to three fully connected layers to predict the motor torques.

In the next stage, the final policy is learnt with guided policy search using reinforcement learning where the agent learns a policy in a trajectory centric manner, with observations being the initial joint angles of the robot and the actions being the desired motor torques to execute the required trajectory. With this two components, they learn a visuomotor policy to guide the robot for a particular task.

In a similar approach Yu et al.[39] use model free reinforcement learning technique using DPN to learn the policy for visuomotor control in an unsupervised manner. They try to reach the goal image by generating an embedding from the autoencoder and learn policy to reach the goal image by traversing in the latent space of the embedding from the autoencoder. The reinforcement learning agent is trained to learn a policy, in order to minimise the distance between the input image embedding

and the goal image embedding.

Piergiovanni et al.[26] proposed a model which they call "dreaming model"[26] to learn to control robots based on images. Their model learns visuomotor policies by interacting with the dreaming model instead of the real world model. The dreaming model consists of synthesised sequence of CNN representations for training derived from the sequence of image frames in a video from the actual environment. Hence, the dreamed model generates dreamed trajectories consisting of state-action pairs from the sequence of CNN representation.

For this approach, the dreaming model is a combination of convolutional autoencoder and an action-conditioned future representation regressor. The autoencoder encodes the current image and the future regressor which is conditioned on the robots actions predicts the image given the action taken by the robot. Thus, they learn the transition model for a model based reinforcement learning agent. Since the model is trained only on the initial video sequence consisting of a few frames, it eliminates the drawback of traditional reinforcement learning approaches, where the agent requires millions of samples over multiple training iterations to converge to a an optimal policy.

Also in their approach, since the dreaming model learns to encode each frame in the video sequence in the latent space, the model can thus have any of the frames as the initial starting point. In summary, the learnt dreaming model can be treated as a learning a simulator based on the examples from the real world model, which can be successfully applied to learn realistic policies.

Ghadirzadeh et al.[12] proposed a deep reinforcement learning based approach to map raw image observations into a sequence of motor activations, which they demonstrate on a PR2 robot for a ball throwing task and a object grasping task. In order to achieve such task dependent behaviour from visual sensory data, they learn a deep policy in a data efficient manner by learning the state representation in a low dimensional latent space by a layer, which they call "perception layer" and action representations by a layer called "behaviour layer".

The perception super layer is trained using a spatial autoencoder, which is a concatenation of four convolutional layers followed by a spatial soft arg-max layer to predict the positions or the image co-ordinates of the task related objects. The decoder for this layer reconstructs the gray-scaled version of the input image consisting

of only the task related object. To predict the spatial position of only the task related object, they train the network with images where the object is placed at different locations in the image, also overlaying a number of other objects at different positions as distractors. The decoder reconstructs the gray-scale version of the image with only the task-relevant object and hence the perception layer is trained to disregard the distractors in the image.

The behaviour super layer is a variational-autoencoder decoder architecture. The encoder maps input trajectories into a normally distributed 5D latent space. The decoder samples a point in the 5D latent space to a complete trajectory. During deployment the decoder consists of three hidden layers which is in the reverse order of the encoder i.e. the decoder maps 5D latent point which represents the input action encoding from the policy layer to a complete trajectory.

The policy layer is trained in a reinforcement learning framework. For each episode the state is derived from the perception layer which encodes the image into a state manifold, and the policy layer generates a distibution over the action manifold from which a sample is drawn by the behaviour layer to produce the trajectories. The trajectory reward a continuous or discrete value at the end of the trajectory in each episode.

Singh et al. [33] propose an approach where the robot learns from a fair amount of examples which have successful results, which eliminates the need for specifying the rewards using manual engineering. This is followed by active queries where the robot asks the user to label if the particular state indicates success or not.

Their state space and action space representations consists of observations from image and with the desired motion of the end effector. The reinforcement learning framework employed is the maximum entropy RL, where they use off-policy soft actor-critic (SAC) algorithm which gives stable and the robust policies and also can be integrated directly with their method used. They use the classifier based rewards, in which the user before training the policy gives the dataset and also a binary classifier predicts if the state is successful or not.

For the reinforcement learning with active queries the authors train classifiers to differentiate between goal and non-goal observations which is used to compute rewards. In order to eliminate the difficulty in manually labelling all the states requested by the robot, they propose an effective method to label the states where

they select the states which have highest probability of success based on classifier and the states which are previously-unlabeled. In-order to achieve reinforcement learning they combine the framework with active queries and the classical based rewards.

The framework of reinforcement learning with active queries(RAQ) does not make use of the entire data which is collected during learning. In-order to use the entire data, they employ an algorithm called VICE proposed by fu et al. [11] which is a framework to specify classifier-based rewards uses on policy Reinforcement learning. In general it requires examples indicating positive outcomes in large numbers.

In the later sections they describe the extension of VICE algorithm and complete VICE-RAQ algorithm which indicates combination with active queries. The authors extend VICE algorithm by employing soft actor-critic algorithm [15] which is an off policy learning algorithm.

They perform experiments in simulation with a 7 DOF arm which is modeled on the Rethink Sawyer for three different tasks which are pushing task, door opening task and the visual picker task. They compare between RAQ, VICE-RAQ, off policy VICE and classifier-based rewards. For the pushing task the off-policy VICE and VICE-RAQ give good performance. For the door opening task the VICE-RAQ performs better than all the methods.

The major limitations of the authors approach include requesting labels from the user which acts as and additional assumption. It also requires a large number of queries which can be minimized with an much more intelligent criterion for querying.

The authors in Duan et al. [9] propose a meta learning framework for the robots to be able to learn from less demonstrations of any given task, and also immediately generalizing to new scenarios of the same task which they term it as one shot imitation learning.

They present the task setting to be having an infinite set of tasks where each task has many instantiations. They use the stacking of blocks tasks as example where the tasks indicate stacking all the blocks on a table, and the other tasks as indicated by the user.

For training their neural network they make use of imitation learning algorithms such as behavioral cloning and DAGGER [31]. The algorithm is presented with a pair of demonstrations. For each sampled task a demonstration is sampled as a pair.

The architecture proposed by the authors consists of different modules namely the demonstration network, the manipulation network, and the context network. The input to the neural network is first demonstration and a sampled state from the second demonstration, thus predicting action for the sampled state. The experiments performed indicate that using the sift attention helps in generalizing the model.

3

Methodology

3.1 Setup

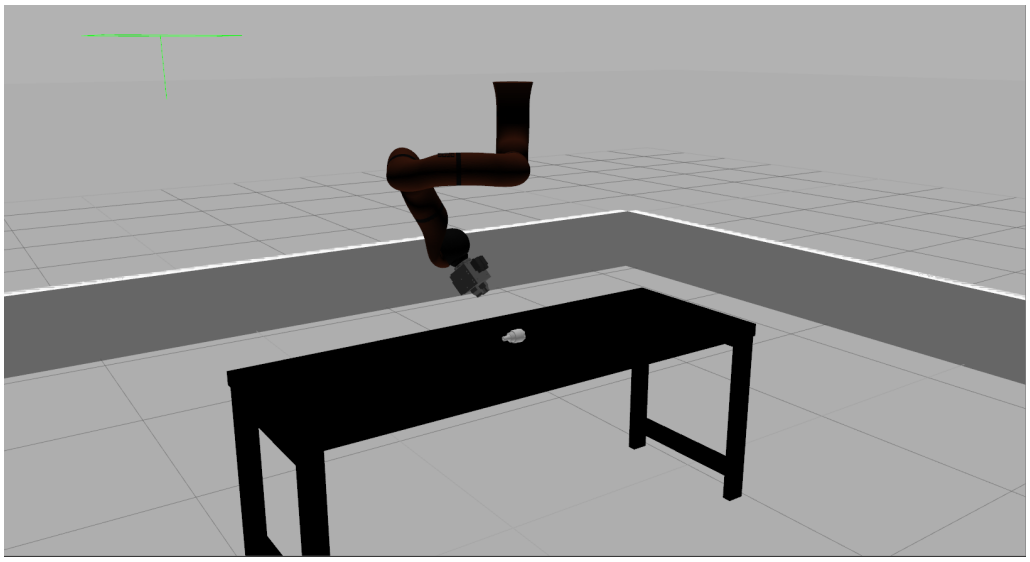


Figure 3.1: The setup for the gazebo simulation with the robot arm hanging on top of the object on the table.

The robot is a 6-axis KUKA-LWR (light weight robot) with a camera mounted on the TCP (tool centre position). The simulation platform is Gazebo with a table and the object placed on top of it. The robot is hung directly above the object in order to maximize the reachable poses for the TCP of the robot. The TCP of the

robot is constrained to move around the object in a sphere with the centre being at the co-ordinates of the object and a radius of 35cm. Hence the camera mounted on the TCP always moves in a sphere around the object always looking at the object. Since the TCP moves in a sphere the co-ordinates for the TCP shall be measured in the polar co-ordinate system i.e. the Cartesian co-ordinates of the TCP shall be expressed with azimuth and elevation. Since the TCP shall not collide with table the robot safety distance from the table is defined to be 1cm from the table. The joint angle limits of the robot impose reachability constraints on the TCP. With this configuration the reachability of the robot is 40-70 degrees of elevation and 0-360 degrees in azimuth.

3.2 Software Architecture

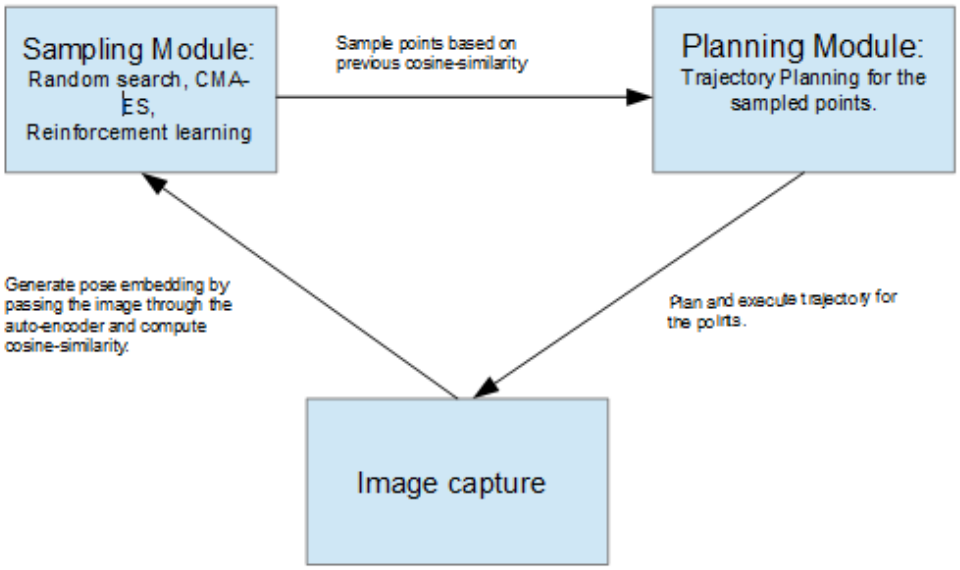


Figure 3.2: Schematic of the software architecture.

The software architecture on the modular level can be categorized in to three modules which are the Sampling Module, the Planning module and the Image capture module. During a single run of the robot to go from an initial pose to the target pose, an iteration can be defined as follows. The sampling module proposes the points

to be sampled for the iteration which depend on the algorithm embedded inside the module and the the cosine-similarity of the image captured by the camera from the previous iteration . The planning module generates and executes trajectories for the sampled points in a collision free manner to reach the pose proposed by the sampling module. Once the TCP reaches the proposed pose by the sampling module the image capture module captures the image from the camera and generates an encoding representative of the pose relative pose of the object as seen by the camera for the current pose reached by the TCP. In the following section each of the modules shall be described in detail.

3.3 Image Capture Module

3.3.1 Autoencoder

The image capture module consists of an autoencoder which generates an embedding in the latent space for the image of the object captured by the camera descriptive of the relative pose of the object with respect to the camera. Autoencoders are neural networks that are used primarily for application in representation learning. Autoencoders compress the high dimensional input in into a a relatively low dimensional latent space and are trained by reconstructing the input from the low dimensional latent space which acts as a bottleneck in the neural network. Thus autoencoder learns in an unsupervised manner by compressing the input $x \in R^n$ by the *Encoder* f into a latent embedding $z \in R^d$ and reconstructing the input through the *Decoder* g to produce $y \in R^n$ where,

The network is trained by minimizing the reconstruction loss $L(x, y)$ which is a measure of the distance between the original input and reconstructed input. The bottleneck acts as a gate limiting the information passing through the network and hence encodes a compression of the input.

3.3.2 Autoencoder for pose estimation

The autoencoder used in this thesis is an improved version of the autoencoder developed by Sundermeyer et al.[35]. The Augumented Autoencoder (AAE)[35]

learns view based representation of the relative pose by encoding the SO(3) views of the objects into the latent vector $zinR^128$. Since the AAE is trained on the synthetic object views, Domain Randomization technique is applied in order to alleviate the need to imitate every specific detail of the RGB image from a real sensor. This is achieved by applying random augmentations to the input training images such as random light positions, randomised diffuse and specular reflections, by inserting random background images, by varying image contrast, brightness, colour distortions and so on[35].

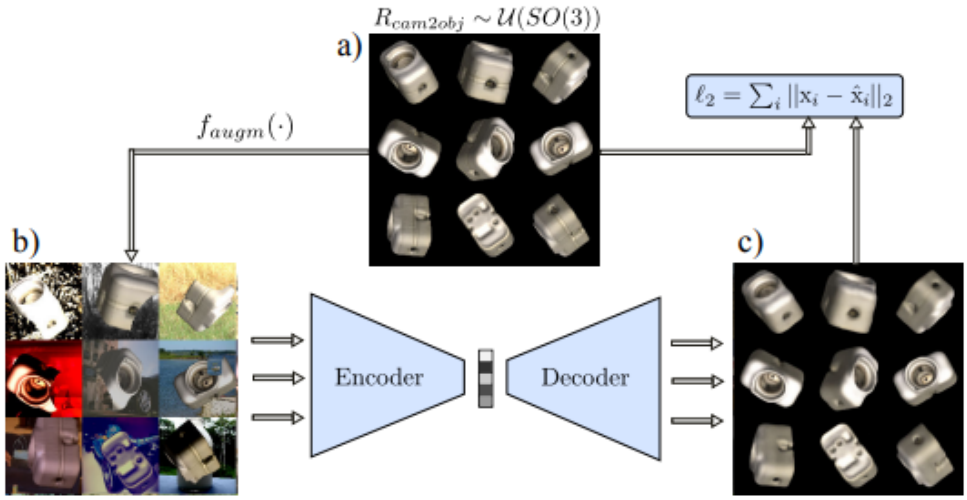


Figure 3.3: Training scheme of the AAE[35]

.

However the Autoencoder used in this thesis is an improved version of the AAE in [35] which is trained on a technique called "Multipath Learning"[34]. Unlike in the AAE in [35], the AAE in [34] is trained on simulated RGB views of multiple objects together where the encoder in the network is shared by all the objects, but the decoder each of the multiple decoders reconstructs views of a single objects. Using this technique Sundermeyer et al. show that the autoencoder learn to encode views of different object instances with shared common features and hence need not be separated for every instance as in [35]. They also show that the autoencoder by sharing an instance agnostic latent space for the encoding also adapts well to the novel object instances.

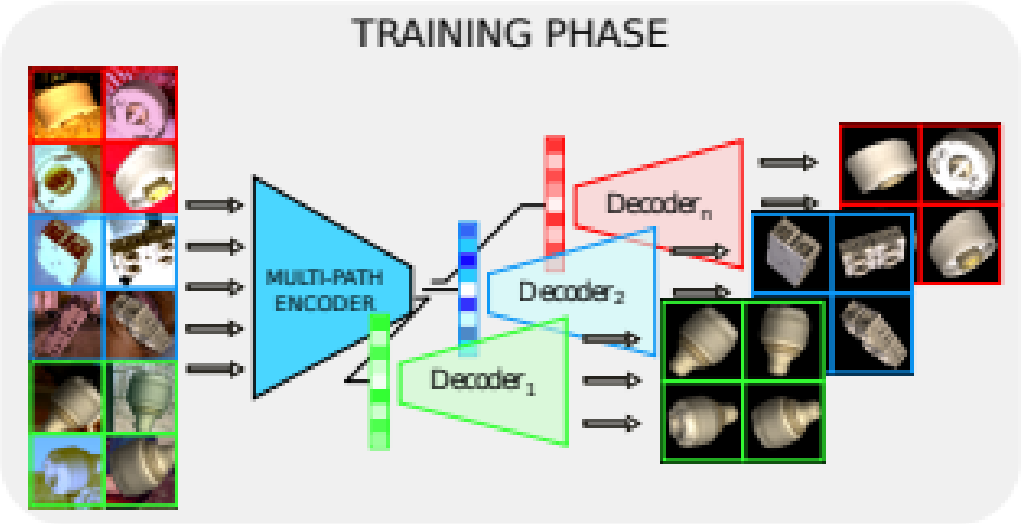


Figure 3.4: Training scheme of the Autoencoder by "Multipath learning"[34]

.

Hence the image module employs the encoder in [34] as relative pose estimator. As required in [34] the RGB image from the camera is quadratically cropped and resized to $128 \times 128 \times 3$ before passing it through the encoder. From the input RGB image the autoencoder outputs a latent vector z $\in R^{128}$ which is then compared to the target latent vector $z_{target} \in R^{128}$ by employing cosine similarity as the distance metric. The cosine similarity thus obtained serves as as indicator of how far in the latent space the current object view is from the target object view to be reached.

However during the experiments it was found that the encoder is susceptible to lighting conditions. When the robot rm reaches the same TCP pose with different configurations of the joint angle, it results in different shadow effects on the object. Due to this for the same view of the image the encoder gives 4 percent error of the cosine similarity which results in a rotational pose error of 3 deg. Hence to mitigate this problem the lighting in the gazebo simulation was set such that it casts minimum shadow on the object and the threshold cosine similarity for all the experiments was set to 0.98.

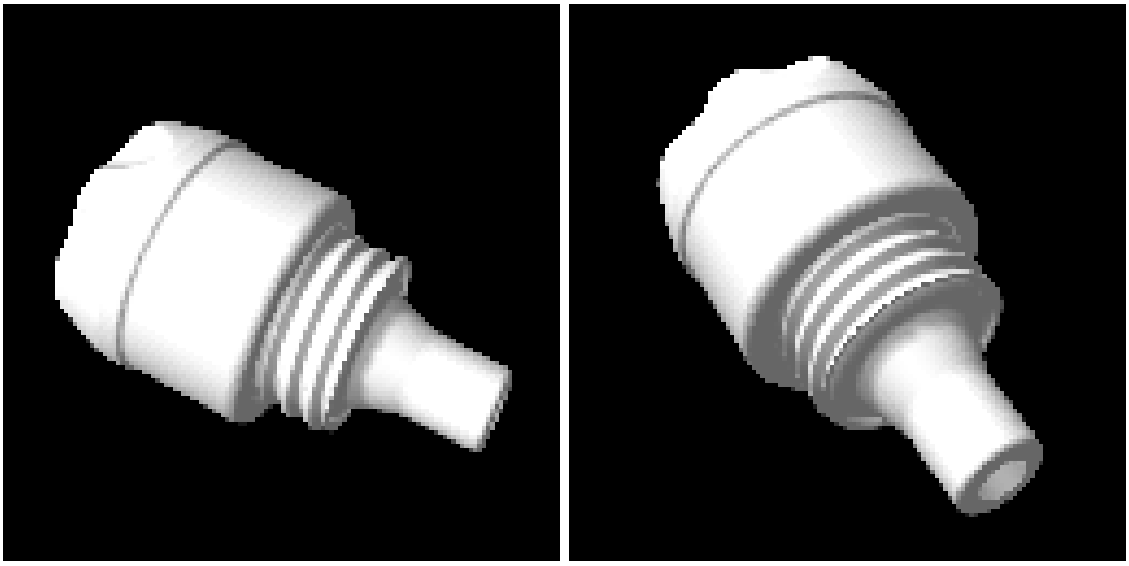


Figure 3.5: Quadratically cropped RGB image from the camera

3.4 Sampling Module

The sampling module is the algorithm we currently employ to search the latent space and move closer to the target embedding or the target object view. The sampling module has the cosine similarity obtained from the Image Capture Module as input and outputs the next batch of points to be sampled in the SO(3) depending on the cosine similarity of the latent embedding from the currently obtained image from the camera and the target image to be reached. Hence the Sampling module takes in the currently obtained cosine similarity and outputs points to be sampled where in each point is a pair of azimuth and elevation to be reached by the TCP as mentioned in 3.1. The algorithms employed for the sampling and their specific implementation shall be described in detail in Chapter 4.

3.5 Planning Module

The task of the planning module is to solve for the joint angles of the robot so that the TCP of the robot reached the point proposed in the Image Capture Module. Hence the planning module generate the trajectories for the robot by avoiding the collisions with environment. TO achieve this the planning module is configured with a configuration file which is a description of the environment the robot is currently

operating in. Primarily in our setting the planner needs to plan trajectories by avoiding any collisons with table and also account for the robot safety distance from the table. The planning module uses a uniform Rapidly exploring Random Trees(RRT)[10] planner in order to generate collision free smooth trajectories for the robot.

3.6 Metrics

3.6.1 Cosine Similarity

Given the target embedding of the RGB image to be reached z_{target}, the cosine similarity of an arbitrary embedding z of the RGB image obtained from the camera mounted on the robot arm is

$$cosinesimilarity = \frac{\vec{z_{target}}.\vec{z}}{\|\vec{z_{target}}\|.\|\vec{z}\|} \tag{3.1}$$

Hence after each run where the robot arm reaches the final position the final cosine similarity between the target latent embedding and the latent embedding of the image from the camera mounted on the robot is computed which gives a measure of how close the robot arm is to target image view.

3.6.2 Rotational Pose Error

During the experiments we know the ground truth 3D-rotational pose R_{gt} of the TCP of the robot arm for the target image view. If the estimated 3-D rotational pose of the TCP of the robot arm be R_{est}, then the 3-D rotational pose error R_{pe} is

$$R_{pe} = \arccos \frac{\vec{R_{gt}}.\vec{R_{est}}}{\|\vec{R_{gt}}\|.\|\vec{R_{est}}\|} \tag{3.2}$$

3.6.3 Translational Pose Error

During the experiments we know the ground truth 3D-translational pose T_{gt} of the TCP of the robot arm for the target image view. If the estimated 3-D rotational

pose of the TCP of the robot arm be T_{est}, then the 3-D rotational pose error T_{pe} is

$$T_{pe} = \|\vec{T_{gt}}\| - \|\vec{T_{est}}\| \tag{3.3}$$

4

Solution

4.1 Random Search

Initially the experiments are carried out using random search algorithm. The input to the algorithm is an initial image $x_{initial}$ and a target image x_{target}. Both the images are passed through the Encoder Φ to obtain an embedding in the latent space $z_{initial}$ corresponding to $x_{initial}$ and z_{target} corresponding to x_{target}, where $z \, \epsilon \, R^{128}$:

$$z = \Phi(x) \tag{4.1}$$

Let $c_{initial}$ be the cosine similarity between $z_{initial}$ and z_{target} and $c_{threshold}$ be the minimum threshold cosine similarity to be reached as the output of the algorithm.

The complete algorithm is formulated as:

Algorithm 1: Random Search

Input : $x_{initial}$, x_{target} , $z_{initial}$, z_{target} , $\phi_{initial}$, $\theta_{initial}$, n, k, e_{max}

Output : ϕ_{target}, θ_{target}, $c_{threshold}$

begin

Let $\{\phi_{initial}$, $\theta_{initial}\}$ = p = m

while $\mathbf{e} \neq \mathbf{e}_{max}$ **and** $\mathbf{c} \leq \mathbf{c}_{threshold}$ **do**

$\{\mathrm{p}_1, p_2, \ldots, p_n\} = \mathcal{N}(m, \mathbf{C})$

$\{\mathrm{z}_1, z_2, \ldots, z_n\} = \Phi(x)$

$\{\mathrm{c}_1, c_2, \ldots, c_n\} = cosinesimilarity(\mathrm{Z},\ \mathrm{z}_{target})$

$\mathrm{m} = \underset{p}{\operatorname{argmax}}\, \mathcal{C}$

end

$\{\phi_{target}, \theta_{target}\} = \mathrm{m}$

end

4.2 CMA-ES

Co-variance Matrix Adaptation evolutionary strategy(CMA-ES) is a class of Evolutionary algorithms which was first described in [17]. Evolutionary algorithms are a way of digital computation inspired by the natural biological evolution. Initially a set of candidate solutions are sampled form a random distribution which constitute the initial population. The population either evolves as a whole during the evolution or the individuals which constitute the population evolve individually. The set of individuals within a population at every step in the evolution is a generation.

The performance of the individuals in each generation is measured by the metric known as the fitness function. The fitness evaluation is problem dependent and is carefully tailored to measure the how far each solution is from the target, for instance the fitness may measure the performance of the robot for the proposed solution from the population at the current generation to complete a particular task.

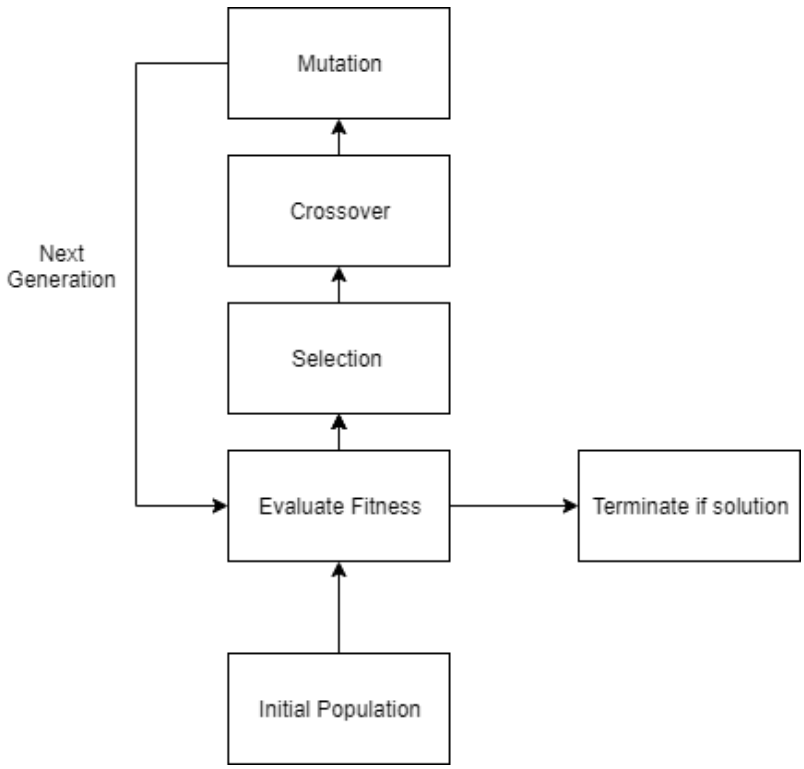


Figure 4.1: Schematic of Evolutionary Algorithm.

Once the fitness is evaluated the candidate solutions are ranked by strength and chosen as parents for the next generation. This constitutes the selection stage in the algorithm. To maintain diversity in the population the selected parents undergo crossover and mutation which are methods of efficiently combining the parents to populate the next generation of the candidate solutions. The implementation of crossover and mutation vary for different evolutionary strategies.

4.2.1 Sampling in CMA-ES

The problem definition for the CMA-ES is essentially a blackbox optimisation problem. Let f be the fitness function and $x \epsilon R^n$ be the candidate solution such that,

$$f \colon R^n \to R \tag{4.2}$$

$$x \to f(x) \tag{4.3}$$

The objective is to find x such that,

$$x = \min_x f(x) \tag{4.4}$$

The only accessible information on the function f is the function evaluations evaluated at the search points.

The sample solutions or the individuals in a generation are sampled from the multivariate normal distribution in CMA-ES. The equation for sampling the individuals for the generation [16]:

$$x_k^{(g+1)} \sim m^{(g)} + \sigma \mathcal{N}(0, \mathbf{C}^g) \qquad for\, k = 1, 2, ..., \lambda \tag{4.5}$$

where,
$\sim$ implies that the left hand distribution is same as the right hand distribution,
$x_k^{(g+1)} \epsilon R^n$ is the k-th offspring at generation g+1,
$m^{(g)} \epsilon R^n$ is the mean of the distribution at generation g,
$\sigma \epsilon R_{>}0$ is the overall standard deviation or step size at generation g,
$\mathcal{N}(0, \mathbf{C^g})$ is a multivariate normal distribution with mean 0 and covariance $\mathbf{C^g}$,

$\mathbf{C^g}_\epsilon R^{n\times n}$ is the covariance matrix at generation g,
$\lambda \geq 2$ is the number of individuals in the population at generation g or the population size at generation g.

At each iteration the new mean of the search distribution is the calculated as the weighted average of μ selected individuals from the sample $x_1^{(g+1)}, ..., x_\mu^{(g+1)}$. The mean update equation is [16]:

$$m^{(g+1)} = m^{(g)} + c_m \sum_{i=1}^{\mu} w_i (x_{i:\lambda}^{g+1} - m^{(g)}) \tag{4.6}$$

where c_m is a learning rate and is usually set to 1 [16]. When $c_m \sum_{i=1}^{\mu} w_i = 1$, the mean update simply is [16]:

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{g+1} \tag{4.7}$$

where $\sum_{i=1}^{\mu} w_i = 1 \qquad w_1 \geq w_2 \geq .... \geq w_\mu$, $\mu \leq \lambda$ is the number of selected points as parents for the next generation, $w_{i=1,..,\mu} \in R_>0$, are the recombination weights, $x_{i:\lambda}^{g+1}$, i-th best individual ranked from the individuals in the population such that $f(x_{1:\lambda}^{g+1}) \leq f(x_{2:\lambda}^{g+1}) \leq .. \leq f(x_{\lambda:\lambda}^{g+1})$ where f is the fitness function to be minimized.

4.2.2 Co-variance Matrix Adaptation

Th best adaptation of the co-variance matrix from the ranked individuals of the population $x_{i:\lambda}^{g+1}$ by incorporating the weighted selection mechanism [16] is :

$$C_\mu^{(g+1)} = \sum_{i=1}^{\mu} w_i (x_{i:\lambda}^{g+1} - m^{(g)})(x_{i:\lambda}^{g+1} - m^{(g)})^{\mathbf{T}} \tag{4.8}$$

As per [16] ${C_\mu}^{(g+1)}$ is an estimator of the co-variance matrix for the distribution of the selected steps and tends to reproduce successful steps and hence is an appropriate estimator.

4.2.3 Rank-μ Update

A reliable estimator of the co-variance matrix when the population size is small in order to facilitate fast search, Rank-μ Update of the co-variance matrix is applied. The equation for the Rank-μ Update of the co-variance matrix is [16]:

$$C^{(g+1)} = 1/(g+1) \sum_{i=0}^{g} 1/{\sigma^{(i)}}^2 C_{\mu}^{(i+1)} \tag{4.9}$$

This simply means that after a certain number of generations, the co-variance matrix for the subsequent generation shall be calculated as the mean of the co-variance matrices of the generation seen up until now. If we want to do a weighted mean of the co-variance matrices of the generations up until now thereby assigning more weight to the recent generations, an exponential smoothing is introduced and the equation is [16]:

$$C^{(g+1)} = 1 - c_{\mu} C^{(g)} + c_{\mu} \sum_{i=0}^{\mu} w_i y_{i:\lambda}^{(g+1)} {y_{i:\lambda}^{(g+1)}}^{\mathbf{T}} \tag{4.10}$$

where,
$y_{i:\lambda}^{(g+1)} = (x_{i:\lambda}^{g+1} - m^{(g)})/\sigma^{(g)}$

4.2.4 Rank-one Update

Instead of considering μ individuals from the generation to update the co-variance matrix as in the previous section, we can update the co-variance matrix in the generation sequence using a single individual. This method constitutes the Rank-one update of the co-variance matrix. To realize this an evolution path or cumulation is utilized. The evolution path is described by the sequence of successive steps that the strategy takes over a number of generations [16]. The distribution mean m with evolution path for three steps can be calculated as[16]:

$$\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} + \frac{m^{(g)} - m^{(g-1)}}{\sigma^{(g-1)}} + \frac{m^{(g-1)} - m^{(g-2)}}{\sigma^{(g-2)}} \tag{4.11}$$

During implementation the evolution path $p_c \in R^n$ with exponential smoothing is calculated as [16]:

$$p_c^{(g+1)} = (1 - c_c)p_c^{(g)} + \sqrt{c_c(2 - c_c\mu_{eff})}\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \tag{4.12}$$

$c_c \leq 1$ is called the backward time horizon and $\sqrt{c_c(2 - c_c\mu_{eff})}$ is a normalisation constant. Finally combining Rank-μ update and the cumulation the co-variance matrix update equation reads[16]:

$$C^{(g+1)} = (1 - c_1 - c_\mu \sum w_j)C^{(g)} + c_1 p_c^{(g+1)} p_c^{(g+1)^{\mathbf{T}}} + c_\mu \sum_{i=0}^{\lambda} w_i y_{i:\lambda}^{(g+1)} y_{i:\lambda}^{(g+1)^{\mathbf{T}}} \tag{4.13}$$

where $c_1 \approx 2/n^2$, $c_\mu \approx min(\mu_{eff}/n^2, 1 - c1)$ and $\sum w_j = \sum_{i=1}^{\lambda} w_i \approx -c_1/c_\mu$

4.2.5 Implementation

The input to the algorithm is the target embedding z_{target}. The population is defined by the individuals where each individual $i \in R^2$ is the a pair of azimuth ϕ and elevation θ of the TCP of the robot arm. For one trial starting from the initial position the TCP, the algorithm samples the individuals for the population of that generation. Then from the obtained latent embedding for the images for each individual in the population the fitness is evaluated and the mean and the co variance for the next generation is updated according to the update rule described in 4.2.4. The fitness function for the algorithm is given by $1 - cosinesimilarity$. For the experiments by constraining the solutions to lie within the reachable poses and additional fitness function of -5 is introduced.

4.3 Reinforcement Learning

Reinforcement learning is a class of machine learning algorithms where the learning agent is trained autonomously in an unsupervised manner by interacting with the environment to learn optimal behaviour in the given environment. The agent learns and improves over time through trial and error.

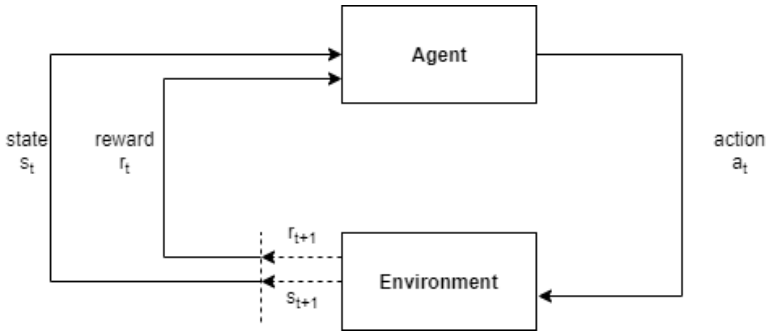


Figure 4.2: Schematic of Reinforcement learning Algorithm.

The general feed back loop in the reinforcement learning is depicted in the Figure 4.2. An autonomous agent embedded with a machine learning algorithm observes a state s_t of the environment at timestep t and the reward r_t the agent received at timestep t. Based on the observation of the state s_t the agent interacts with the environment by taking an action a_t, to change the state of the environment from s_t to s_{t+1}. With the interaction of the agent with environment by the action a_t, the agent receives a reward r_{t+1}. With the sequence of state, action and reward the agent learns a policy π that maximizes the expected reward from the interaction with environment. Reinforcement learning can be formalised by a MDP(Markov Decision Process) consisting of a set of states S , a set of actions A, an instantaneous reward function $r(s_t, a_t, s_{t+1})$ and state transition dynamics $\tau(s_{t+1}|s_t, a_t)$ which is a mapping from state-action pair at timestep t onto a distribution of state s_{t+1}.

The policy π is a mapping from states to a probability distribution over actions $:\pi : S \rightarrow p(A = a|S)$. A sequence of states, actions and rewards is known as trajectory and if the length of the trajectory is finite then each trajectory is called an *episode*. In an *episodic* also known as finite time horizon MDP the length of the trajectory is denoted by $\mathbf{T}$. For an episode of length T *Cumulative reward* R is defined as the discounted sum of instantaneous reward at each step:

$$R = \sum_{t=0}^{T} \gamma^t r_{t+1} \tag{4.14}$$

where $\gamma \in [0, 1]$ known as the discount factor signifies how much to take into

account the immediate reward as compared to a long term reward down the trajectory. The goal of reinforcement learning is to arrive at an optimal policy π^* which maximizes the expected return or rewards over all states:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[R|\pi] \tag{4.15}$$

In the reinforcement learning paradigm there are two main ways in which the reinforcement learning problem could be solved. One of the methods relies on finding an optimal *Value Function* and the other methods relies on finding an optimal *Policy Function* through *Policy Search* methods. The Value function and Policy function are described as follows.

1. **Value Function**: Generally Value function is the expected return of the agent being in a given state. The value function V^π , also termed as the state-value function is the expected return of being in state s and then following the policy π:

   $$V^\pi(s) = \mathbb{E}[R|s, \pi] \tag{4.16}$$

   The optimal value function V^* is then defined as,

   $$V^* = \max_\pi V^\pi(s) \qquad \forall s \in S \tag{4.17}$$

   Thus bu utilizing V^* an optimal policy π^*could be obtained by choosing action a from the set of actions at s_t that maximizes the expectation of the optimal value function $V^*(s_{t+1})$.

2. **Q-value function**: Q-value function computes the expected return of taking an action a in state s following the policy π. Thus the Q-function maps the state-actions pairs to the expected return:

   $$Q^\pi(s, a) = \mathbb{E}[R|s, a, \pi] \tag{4.18}$$

   and the Value function under a policy following a policy π can be defined as,

   $$V^\pi(s) = \max_a Q^\pi(s, a) \tag{4.19}$$

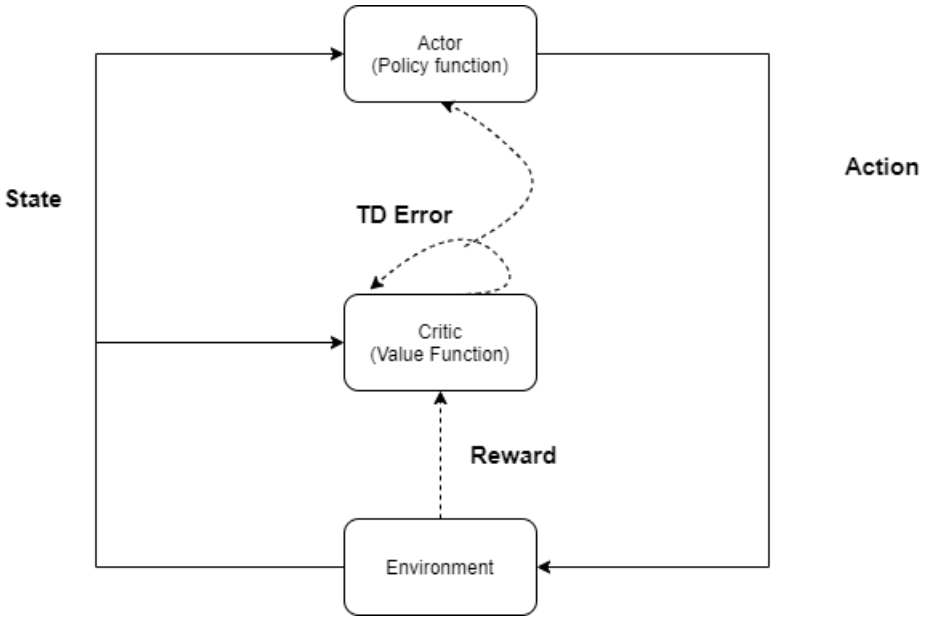


Figure 4.3: Schematic of Soft Actor Critic Algorithm

In Policy search methods instead of finding a policy that optimises the Value function, a direct policy parameterized as π_θ is chosen and then optimised to arrive at an optimal policy π^* by updating the parameters that maximize the expected return $\mathbb{E}[R|\theta]$. Thus the parameters can be optimized by using either gradient based methods or gradient free methods and an extensive listing can be found in [1].

4.3.1 Soft Actor Critic Algorithm (SAC)

With the advancement of Deep Learning, Deep Neural Network techniques have been employed to solve the reinforcement learning problem. In this framework known as Deep Reinforcement Learning the Value functions and Policy functions are represented by neural networks. The Soft Actor Critic Algorithm was introduced by Haarnoja et al.[15]. The SAC uses three neural networks as function approximators: State Value function $V_\psi(s_t)$ parameterized by ψ, State-Action Value function or the Q-function $Q_\theta(s_t, a_t)$ parametrized by θ and the policy function π_ϕ parametrized by ϕ[15]. While the State Value functions and the Q-function are neural networks directly computing the respective values, the policy is modelled to predict the mean and co-variance of a Gaussian distribution.

4.3.2 Entropy Regularisation

One of the key features of SAC in solving the RL problem is the Entropy regularisation of the loss functions. For a random variable v with probability density function given by $\mathcal{D}$, the entropy $\mathcal{H}$ is defined as,

$$\mathcal{H}(\mathcal{D}) = E[-log\mathcal{D}(v)] \tag{4.20}$$

In this setting the agent gets an additional reward proportional to the entropy of the policy at each timestep. With this the maximum entropy objective with the expected entropy of the policy over $\rho_\pi(s_t)$[15] :

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\tilde{\rho_\pi}}[r(s_t, a_t) + \alpha\mathcal{H}(\pi(.|s_t))] \tag{4.21}$$

where α known as the temperature term is the weighting between the reward and the entropy of the policy and hence is responsible for controlling the stochasticity of the optimal policy.

4.3.3 Loss Functions and Update

The SAC can be trained in an on-policy fashion or off-policy fashion. When training on off-policy a replay buffer $\mathcal{D}$ stores the state-action-reward transitions from which a a batch of transitions is randomly sampled during the training. The soft value function is trained by minimizing the squared residual error objective[15]

$$J_v(\psi) = \mathbb{E}_{s_t\sim\mathcal{D}}[\frac{1}{2}(V_\psi(s_t) - \mathbb{E}_{a_t\sim\pi_\phi}[Q_\theta(s_t, a_t) - \log\pi_\phi(a_t|s_t)]^2] \tag{4.22}$$

and the gradient is given by

$$\hat{\nabla}_\psi J_v(\psi) = \nabla_\psi V_\psi(s_t)(V_\psi(s_t) - \hat{Q}_\theta(s_t, a_t) + \log\pi_\phi(a_t|s_t)) \tag{4.23}$$

The soft Q-function parameters are trained by minimizing the soft Bellman residual[15]

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left((Q_\theta(s_t,a_t) - \hat{Q}_\theta(s_t,a_t)\right)^2\right] \tag{4.24}$$

The gradient is computed as

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(s_t,a_t)\left(Q_\theta(s_t,a_t) - r(s_t,a_t) - \gamma V_{\overline{\psi}}(s_{t+1})\right) \tag{4.25}$$

where the soft Q-function

$$\hat{Q}_\theta(s_t,a_t) = r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}[V_{\overline{\psi}}(s_{t+1})] \tag{4.26}$$

The loss functions are trained using stochastic gradient descent and $V_{\overline{\psi}}(s_{t+1})$ is output from the target value network, where the network weights $\overline{\psi}$ are computed as the exponentially moving average. The policy loss function is trained by minimizing the expected KL-divergence objective as[15]

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[D_{KL}\left(\pi_\phi(.|s_t)\middle\|\frac{\exp(Q_\theta(s_t,.))}{Z_\theta(s_t)}\right)\right] \tag{4.27}$$

To minimize the objective function the policy is reparameterized by the neural network transformation by with noise vector ϵ_t[15]

$$a_t = f_\phi(\epsilon_t; s_t) \tag{4.28}$$

With the reparameterized objective the gradient is thus computed as[15]

$$J_\pi(\phi)\mathbb{E}_{s_t\sim D,\epsilon_t\sim\mathcal{N}}[\log\pi(\phi)(f_\phi(\epsilon_t;s_t)|s_t) - Q_\theta(s_t, f_\phi(\epsilon_t;s_t))] \tag{4.29}$$

$$\hat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log\pi_\phi(a_t|s_t) + ((\nabla_{a_t}\log\pi_\phi(a_t|s_t) - \nabla_{a_t}Q(s_t,a_t))\nabla_\phi, f_\phi(\epsilon_t;s_t) \tag{4.30}$$

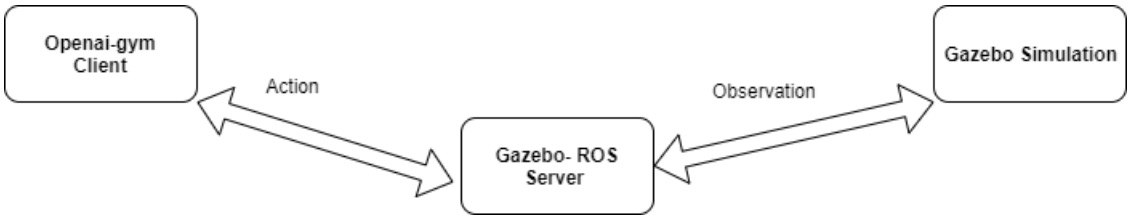


Figure 4.4: Client Server based architecture for Reinforcement learning simulation

4.3.4 Implementation

The toolkit used for the implementation of the environment is OpenAI-Gym [5]. OpenAI-Gym is a ubiquitous toolkit for modelling the environment for the reinforcement learning agent to act upon and hence train on. The environment in our case is the gazebo simulation environment with the robot. The utility of the Gym environment is that it provides interface functions which makes it easier to define the RL problem. The architecture for the reinforcement learning experiment is shown in The Gym environment is implemented as a client which sends requests to the Gazebo-ROS server. The Server is in turn linked to the Gazebo simulation which sends the response for the request from the client. The Gym environment client is configured by the following:

1. Observation Space: Observation is analogous to the state definition in the RL problem. The observation space here is defined as a Box which refers to the continuous class of observation spaces in the Gym environment definitions. For any position of the TCP of the robot arm the state $s \in R^{265}$ is the concatenation of the latent embedding $z \in R^{128}$ of the image currently being captured by the camera, $z_{target} \in R^{128}$ the latent embedding of the target image to be reached and $j \in R^9$ the vector consisting of the 9 joint angles for the current configuration of the robot arm.

2. Action Space: Action space defines the space for the action vector in the Gym environment. In our case the action space $a \in R^2$ consists of two scalars $< a_{azimuth}, a_{elevation} > \in [-1, 1]$. Before applying the action on the environment the action vector $\mathbf{a}$ is scaled by a scaling factor δ of 30deg such that the action, $a : [-1, 1] \mapsto [-30, 30]$. So when the agent takes an action the azimuth ϕ and

elevation θ co-ordinates of the TCP of the robot arm are updated as

$$\phi \leftarrow \phi + \delta * a_{azimuth} \tag{4.31}$$

$$\theta \leftarrow \theta + \delta * a_{elevation} \tag{4.32}$$

3. Reward: In the RL setting when the agent takes an action it receives a reward indicating how good the action was. In our environment the reward r is

$$\mathrm{r} = \begin{cases} cosinesimilarity(z, z_{target}) - 1 & if \;\; \phi, \theta \;\; are \;\; reachable \;\; configurations \\ -2(penalty) & if \;\; \phi, \theta \;\; are \;\; not \;\; reachable \;\; configurations \\ +2(bonus) & if \;\; \phi, \theta \;\; reach \;\; the \;\; threshold \;\; cosinesimilarity \end{cases} \tag{4.33}$$

4. Step: The Step is a function interface which implements the action definition in the Gym environment.

5. Reset: Starting from an initial TCP position of the robot the TCP moves such that at each step it get closer to the the target position which is defined as an episode. After each episode a boolean tag *done* is updated to True upon which the Reset function is called which resets the TCP of the robot to a new initial position for a new episode.

With the above configuration of the environment the SAC algorithm for training

the agent

Algorithm 2: SAC Algorithm [5]

Input : initial policy network parameters θ, Q-function parameters ϕ_1, ϕ_2, V-function parameters ψ and empty replay buffer **D**

Set target parameters equal to main parameters $\psi_{target} \leftarrow \psi$

repeat

Acquire initial state s_t *and select action* $a \sim \pi_\theta(.|s)$

Execute action a in the environment

Acquire next state s_{t+1}, *reward* r *and done flag* d

Store $(s_t, a_t, r, s_{t+1}, d)$ *in the replay buffe*$\mathring{D}$

If s_{t+1} *is the terminal state reset the environment*

if *update* **then**

for k *in range no of updates* **do**

Randomly sample a batch of $\mathbf{B} = (s_t, a_t, r, s_{t+1}, d)$ from **D**

Compute target Q and V function values:

$y_q = r + \gamma(1-d)V_{\psi_{target}}(s_{t+1})$

$y_q = Q_\phi(s, a_t) - \alpha \log \pi_\theta(a_t|s)$

update Q-function by one step of gradient descent using,

$\nabla_\phi \frac{1}{|B|} \sum (Q_\phi(s, a_t) - y_q)^2$

Update the V-function by one step of gradient descent using ,

$\nabla_\psi \frac{1}{|B|} \sum (V_\psi(s) - y_v)^2$

Update the policy by one step of gradient descent using,

$\nabla_\theta \frac{1}{|B|} \sum (Q_\phi(s, a_\theta(s)) - \alpha \log \pi_\theta(a_\theta(s)|s_t))$

Update target Value network with,

$\psi_{target} \leftarrow \rho\psi_{target} + (1-\rho)\psi$

end

end

until *convergence*;

5

Results

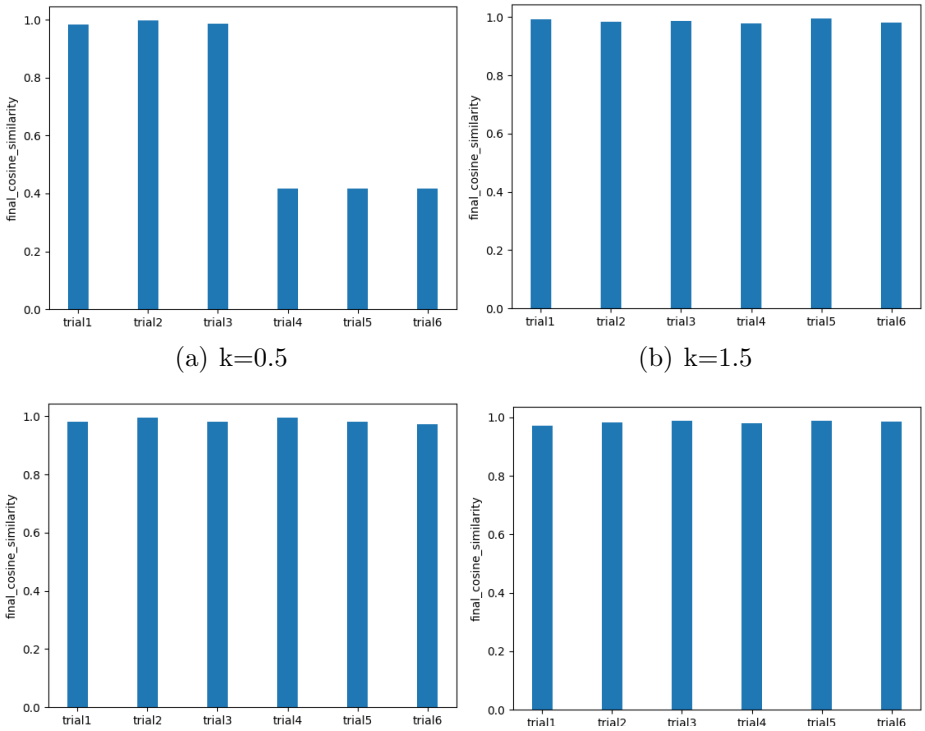
In this chapter the results for the experiments carried out for evaluating the random search, CMA-ES and Reinforcement learning are presented. Further, comparison of the algorithms is presented using the metrics described in Chapter3.

5.1 Random Search

The experiments with random search algorithm were carried out as described by the algorithm in the Section 4.1. Initially the trials were carried out with varying values for parameter k with six initial poses for the TCP of the robot. Initially the mean is set to the initial pose and in every *epoch* the algorithm samples 10 points normally distributed around the current mean and computes cosine similarity of the latent embedding with the target latent embedding. Then the mean is updated as the point for which the cosine similarity is the highest among the sampled points, which is the mean for the next epoch. The plots in the Figure 5.1, Figure 5.2 and Figure 5.3 show the final cosine similarity, rotational pose error and transnational pose error for four different k values respectively. The plots can be summarised as in the Table 5.1.

From the table it can be noted that the metrics for $k = 1.5$ performs better compared to others on average. Hence, the parameter k for the random search algorithm from the above mentioned tuning is set to 1.5.

After setting the value of the k to 1.5 further experiments were carried out with increased number of initial poses to determine the repeat-ability of the random search



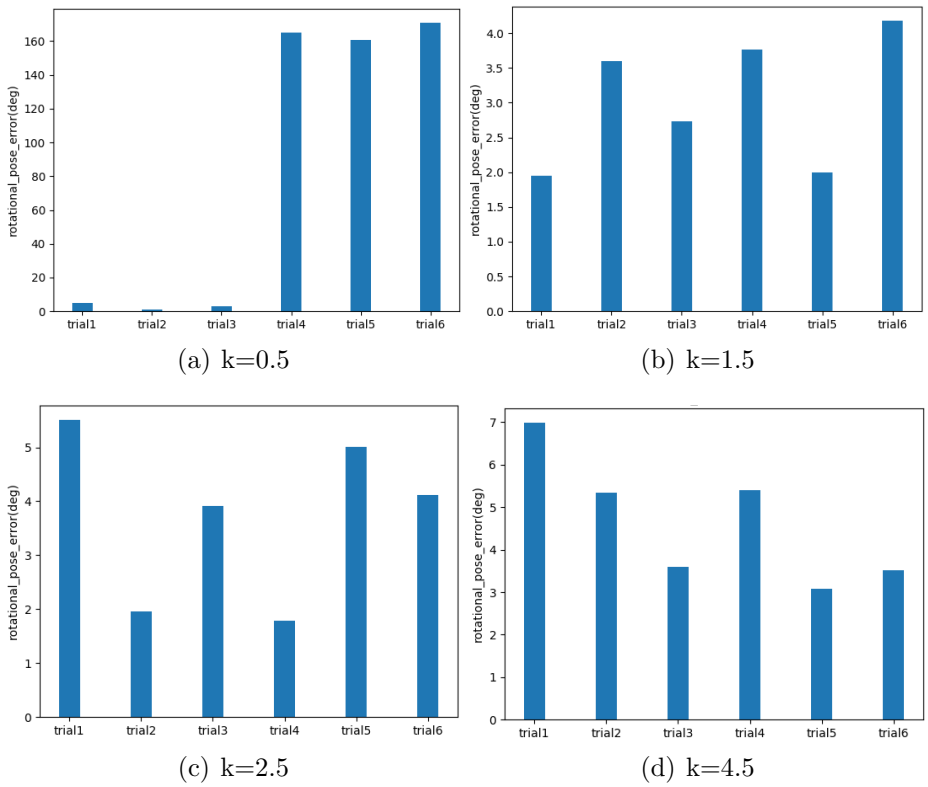
(a) k=0.5 (b) k=1.5

(c) k=2.5 (d) k=4.5

Figure 5.1: Random Search: Final Cosine similarity

k	mean final cosine similarity	mean rotational pose error	mean transnational pose error
0.5	0.702	84deg	28cm
1.5	0.987	3deg	1.6cm
2.5	0.983	3.7deg	2cm
4.5	0.982	4.6deg	2.6cm

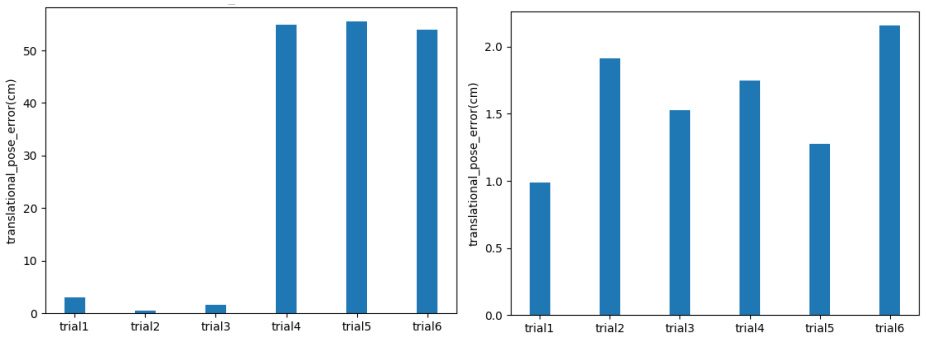
Table 5.1: Random Search: Comparison of metrics



(a) k=0.5 (b) k=1.5

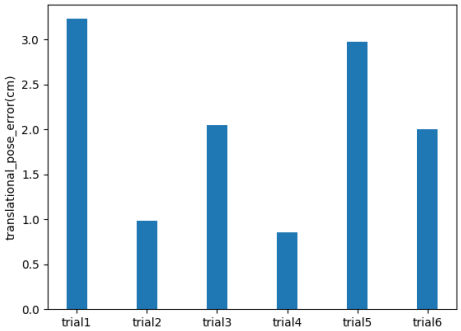
(c) k=2.5 (d) k=4.5

Figure 5.2: Random Search: Rotational pose error

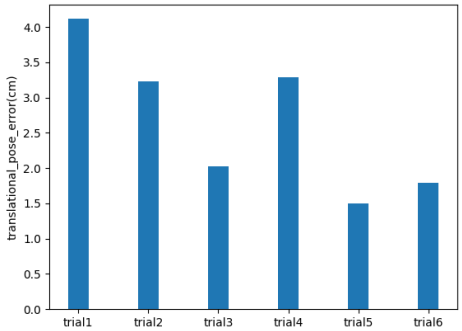


(a) k=0.5

(b) k=1.5



(c) k=2.5



(d) k=4.5

Figure 5.3: Random Search: Transnational pose error

algorithm. 10 runs are carried out with 12 trials(initial poses) each, 3 from each of the four quadrants from the sphere. The final cosine similarity, rotational pose error and the transnational pose error is shown in the Figure 5.4, Figure 5.5 and Figure 5.6 respectively. It can be seen all the trials converge to the cosine similarity greater or equal to the threshold in run 3, run 4, run 6 and run 8 and one trial each in run 1, run 2, run 5, run 7, run 9 and run 10 do not converge which can also be seen from the high rotational and translation pose error for these trials. Hence out of total of 120 trials across 10 runs 6 trials do not converge and therefore the success rate is 95 percent for random search.

(a) run=1 (b) run=2

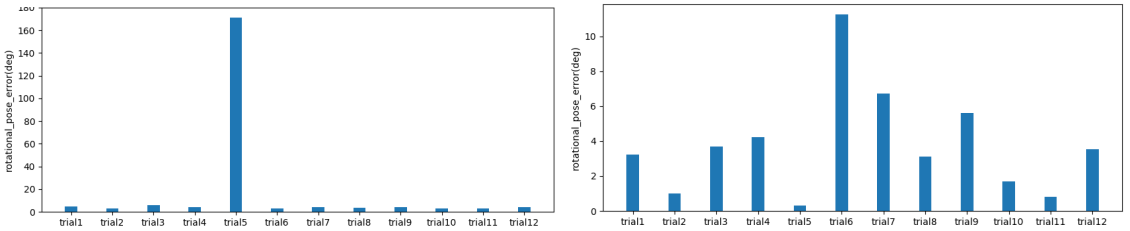
(c) run=3 (d) run=4

(e) run=5 (f) run=6

(g) run=7 (h) run=8

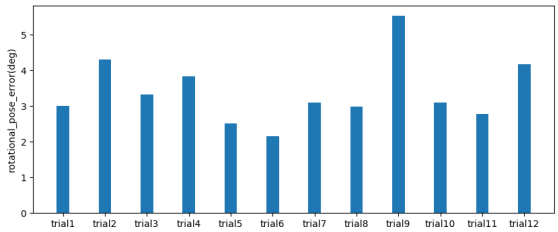
(i) run=9 (j) run=10

Figure 5.4: Random Search: Final cosine similarity

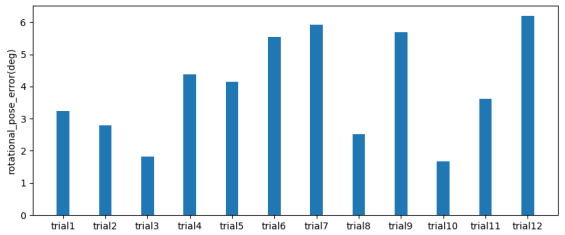


(a) run=1

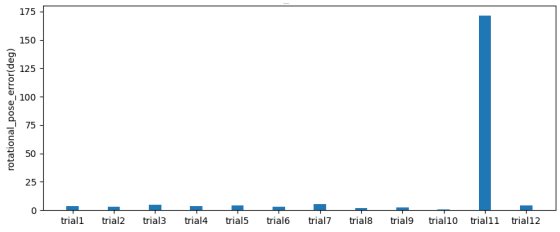
(b) run=2



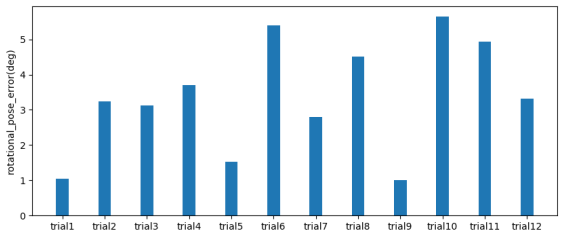
(c) run=3



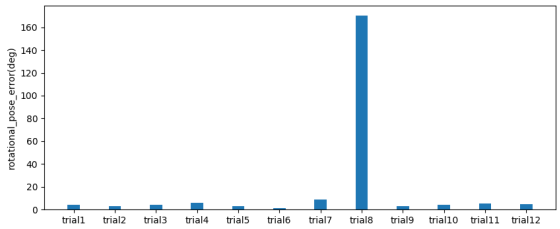
(d) run=4



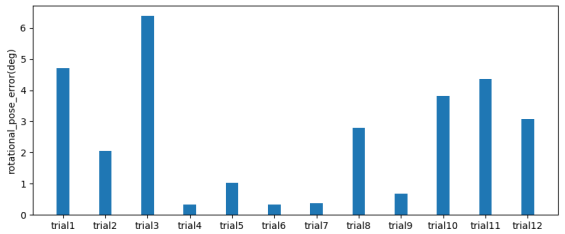
(e) run=5



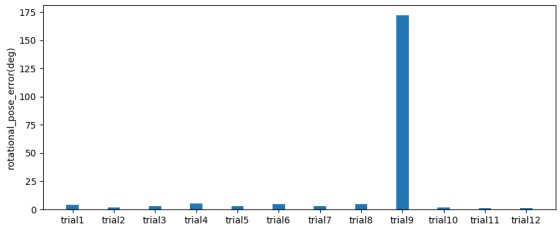
(f) run=6



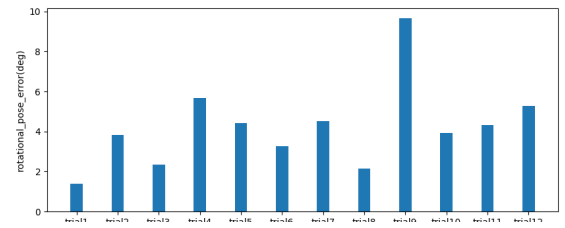
(g) run=7



(h) run=8

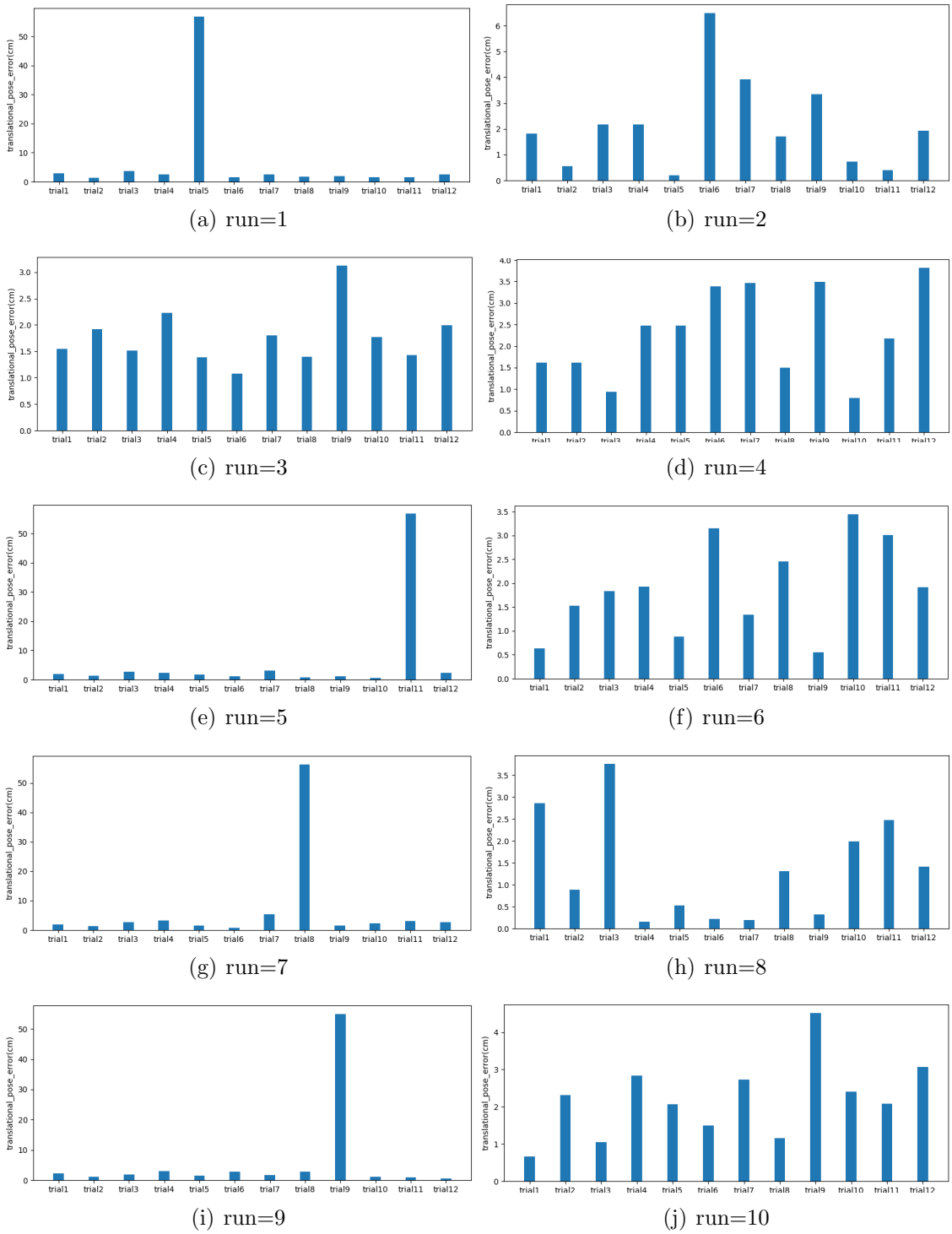


(i) run=9



(j) run=10

Figure 5.5: Random Search:Rotational pose error



(a) run=1 (b) run=2

(c) run=3 (d) run=4

(e) run=5 (f) run=6

(g) run=7 (h) run=8

(i) run=9 (j) run=10

Figure 5.6: Random Search:Transnational pose error

5.2 CMAE-ES

The experiment for CMA-ES was carried out with the algorithm that is described in Section 4.2. The experiments were carried out with a population size of 10. Similar to the experiments for random search, 10 runs are carried out with 12 trials (initial poses) each, three from each of the four quadrants of the sphere. Figure 5.7, Figure 5.8 and Figure 5.9 show the plot for final cosine similarity, transnational pose error and rotational pose error for the 10 runs respectively. From the plots,it is clear that all the trials converge to the threshold cosine similarity in run 1, run 4, run 6, run 8 and run 10 while one trial each in run 2, run 3, run 5, run 7 and run 9 converge below the threshold cosine similarity.

Hence out of total of 120 trials across 10 runs 5 trials do not converge and therefore the success rate is (120-5)*100/120=96% for CMA-ES.

5.3 Comparison

Figure 5.10 shows the mean and variance plots for the final cosine similarity, rotational pose error and transnational pose error for each of the 12 trials for 10runs for random search and CMA-ES. The mean cosine similarity for both random search and CMA-ES are in the range of 0.985 and 0.99, rotational pose error in the range of 4-5 deg and transnational pose error in the range of 1 to 2 cm when the trials converge beyond the threshold cosine similarity. For both random search and CMA-ES the mean of number of evaluations to reach the threshold are in the range of 40 to 60 evaluations.

However, for the trials which are the outliers that do not converge, the final cosine similarity, rotational pose error and translational pose error for random search are in the range 0.4-0.45, 170-175 deg and 50-55 cm respectively. Whereas for CMA-ES for the outliers, the final cosine similarity, rotational pose error and transnational pose error for random search are in the range 0.6-0.7, 20-25 deg and 10-15 cm respectively.

Therefore, CMA-ES perform better with lower values for the metrics when the trials do not converge as compared to the random search.

(a) run=1 (b) run=2

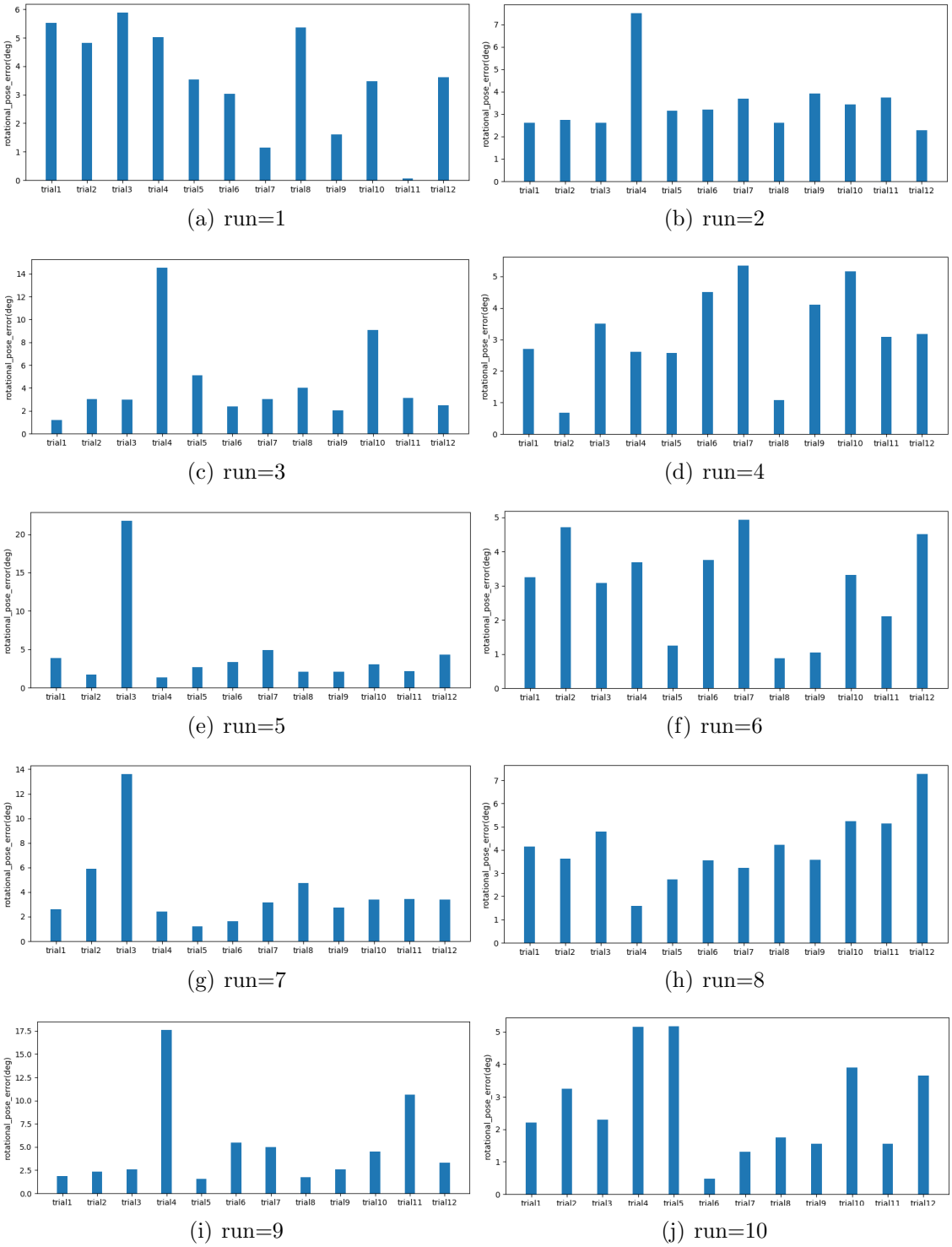
(c) run=3 (d) run=4

(e) run=5 (f) run=6

(g) run=7 (h) run=8

(i) run=9 (j) run=10

Figure 5.7: CMAE-ES: Final cosine similarity



(a) run=1 (b) run=2

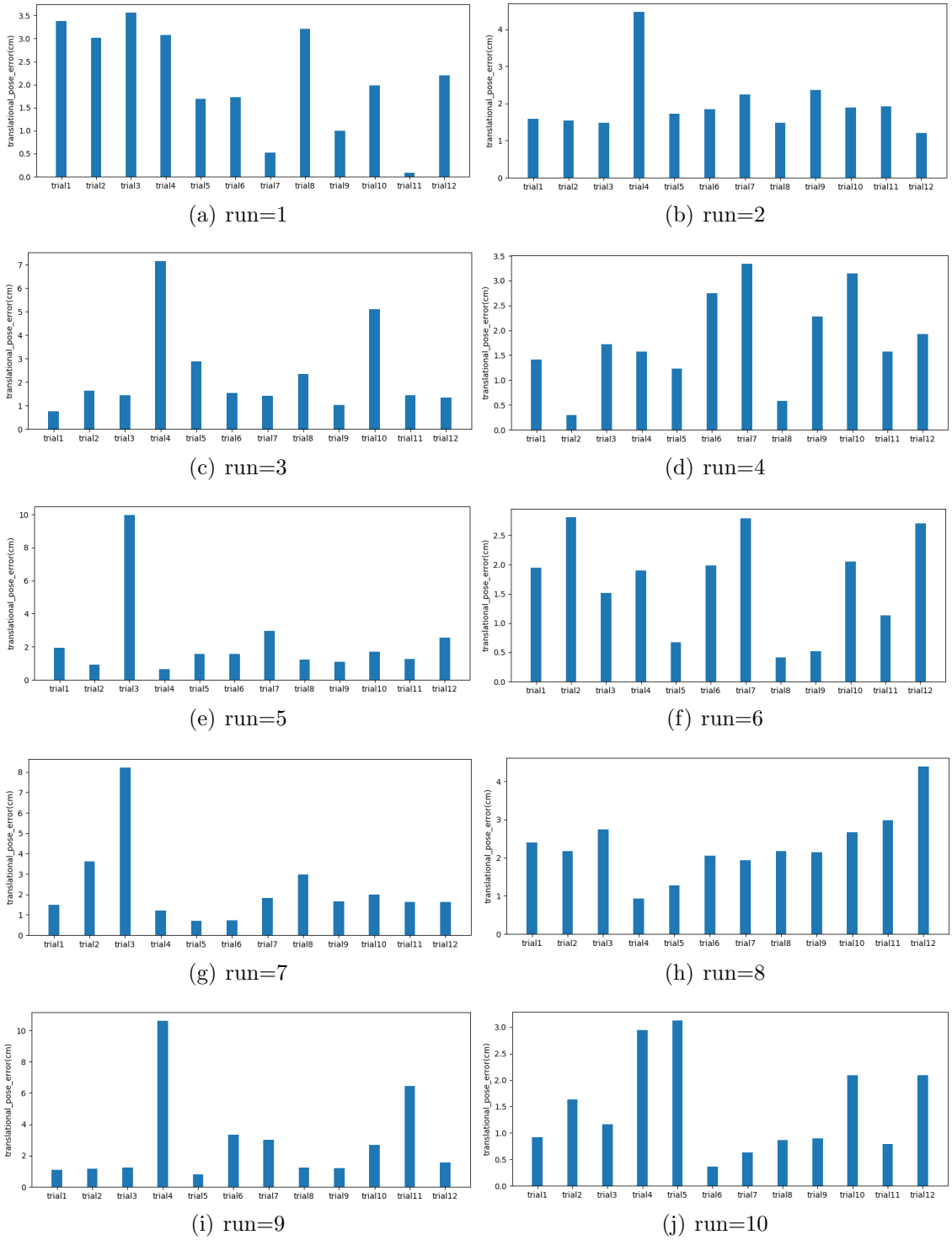
(c) run=3 (d) run=4

(e) run=5 (f) run=6

(g) run=7 (h) run=8

(i) run=9 (j) run=10

Figure 5.8: CAM-ES:Rotational pose error



(a) run=1 (b) run=2

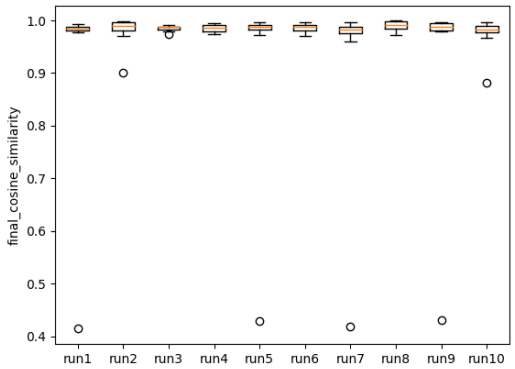
(c) run=3 (d) run=4

(e) run=5 (f) run=6

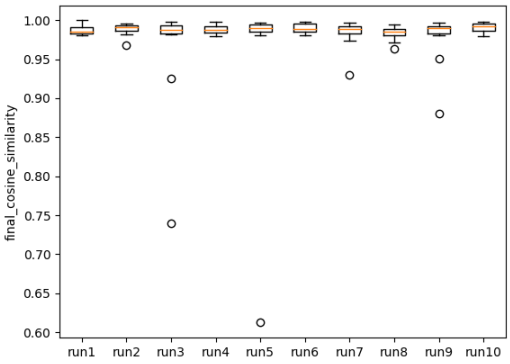
(g) run=7 (h) run=8

(i) run=9 (j) run=10

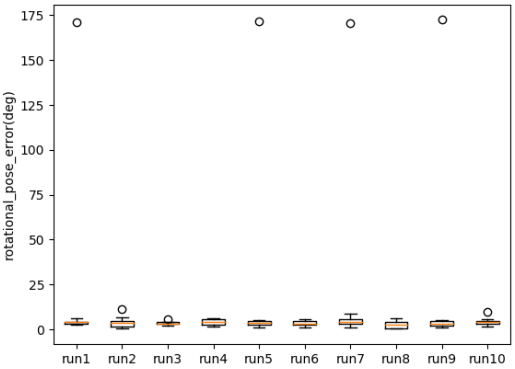
Figure 5.9: CMA-ES: Translational pose error



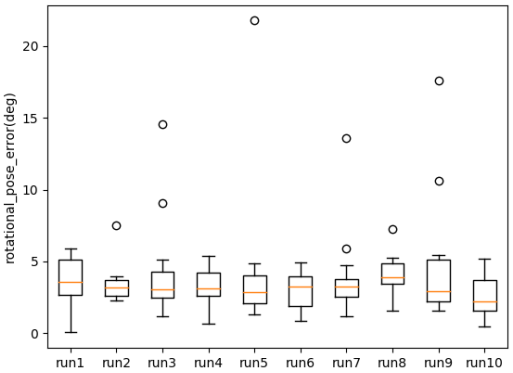
(a) random search: final cosine similarity



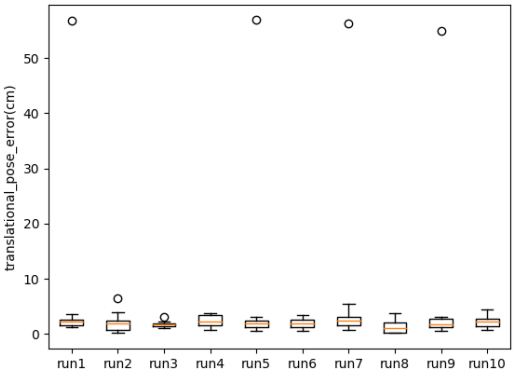
(b) CMA-ES : final cosine similarity



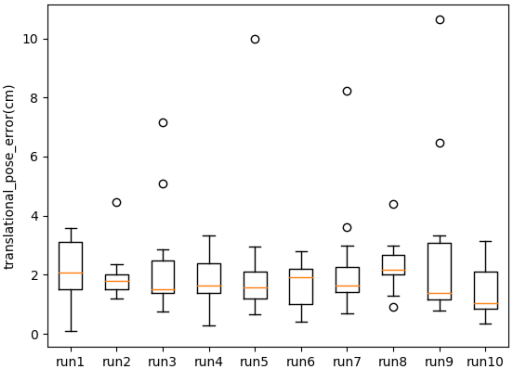
(c) random search: rotational pose error



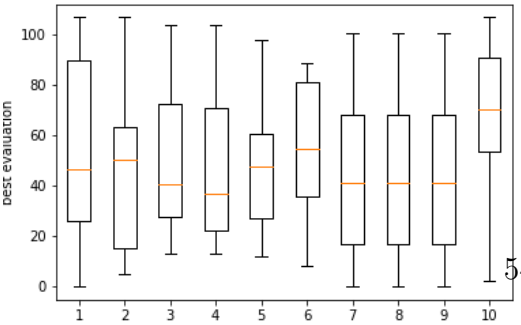
(d) CMA-ES : rotational pose error



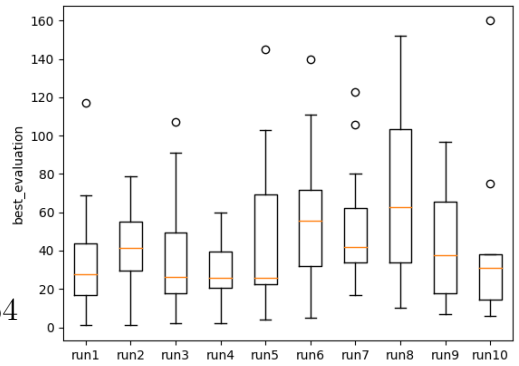
(e) random search: translational pose error



(f) CMA-ES: translational pose error



(g) random search: evaluations to reach threshold



(h) CMA-ES: evaluations to reach threshold

Figure 5.10: Comparison of random search and CMA-ES

5.4 Unreachable poses

The experiments were carried out with the CMA-ES algorithm without any constraints for the individuals in the population to be in the range of reachable poses of the TCP of the robot arm. In this setting, the fitness function was modified as to penalise the algorithm when the individuals in the population are out of the reachable poses of the TCP of the robot arm.

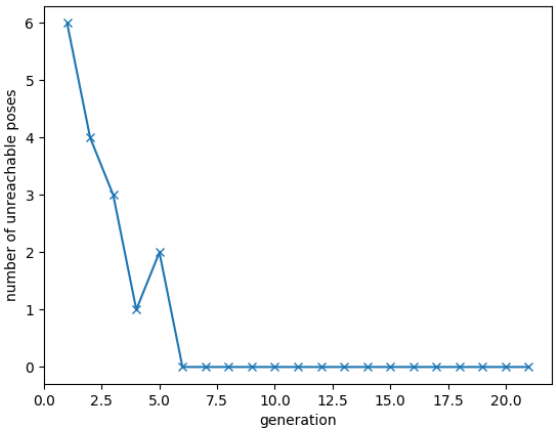
Three trials were carried out for three initial poses according to the Table 5.2. For each trial the number of individuals in the population in each generation which are outside the reachable poses for the TCP are plotted in the Figure 5.11.

It is clear from the plots that the number of solutions in the population which are out of the reachable poses of the TCP decreases with each generation. However, the final cosine similarity does not converge to the threshold even after running each trial for more than 25 generations as summarized in the Table 5.2.

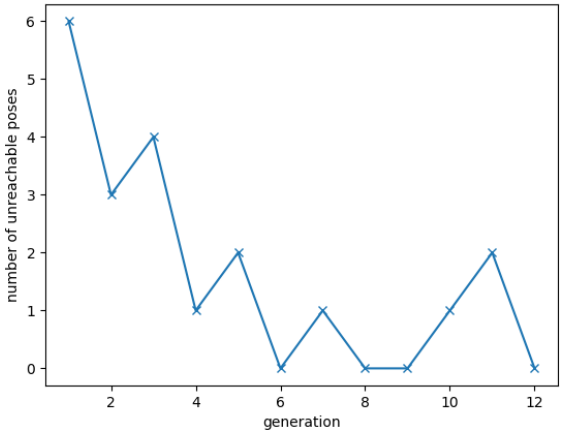
CMA-ES evolves very slowly under the constraint of unreachability and this prompts us towards learning the unreachbility constraint explicitly by a learning based method.

Trial	Initial cosine similarity	Initial rotational pose error(deg)	Initial transnational pose error(cm)	Final Cosine similarity
1	0.2	130	44	0.06
2	0.1	140	46	0.02
3	0.45	170	53	0.42

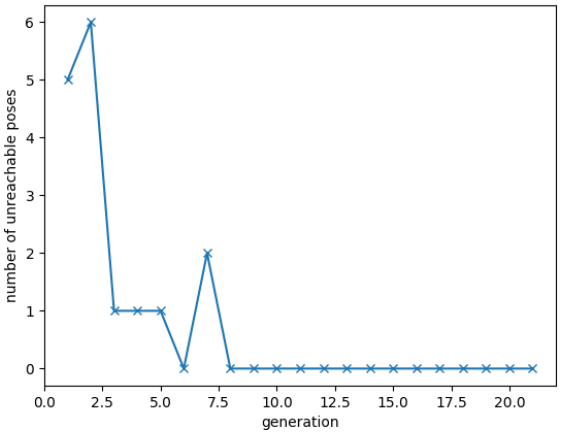
Table 5.2: CMA-ES: Results for optimisation with unreachability constraint



(a) CMA-ES: Trial 1



(b) CMA-ES: Trial 2



(c) CMA-ES: Trial 3

Figure 5.11: Unreachable poses

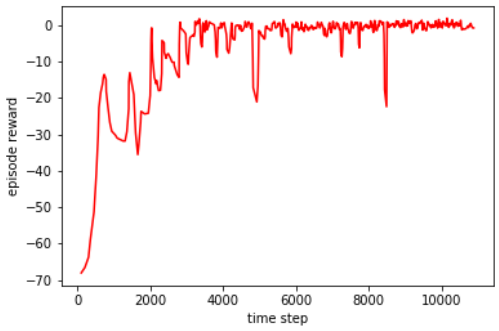
5.5 Reinforcement Learning

In this section the training and evaluation of the SAC reinforcement algorithm will be discussed. The algorithm and implementation details is described in Section 4.3. The robot arm starts from an initial pose of the TCP and tries to reach the target pose which is an *episode*. The observation space or the state space, the action space and the rewards are described in the Section 4.3.4.

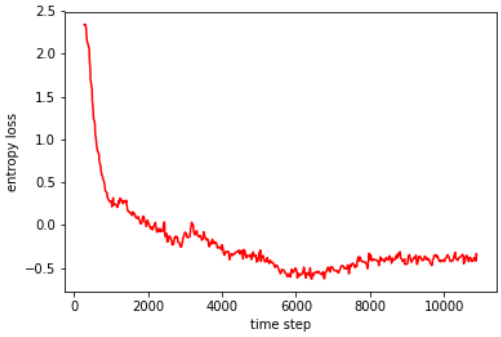
The training is carried out for 11,000 time steps and the loss functions are shown in the Figure 5.12. The evolution of the reward is shown in the Figure 5.12(a). At the beginning of the training the the agent receives a lot of negative rewards

since the agent predicts a lot of unreachable poses. As the training progresses the reward converges to 0 since which since the agent has successfully learnt to avoid the unreachable poses. Hence the agent successfully learns to navigate to the target pose by avoiding unreachable poses of the TCP.

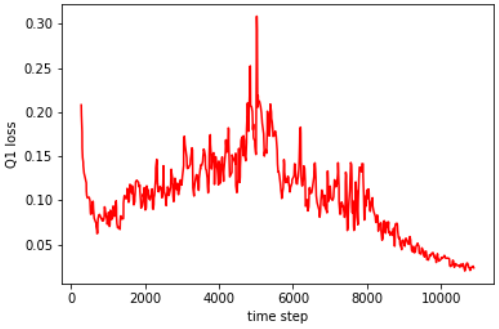
The trained agent is evaluated by carrying out 100 episodes. The results of the evaluation are shown in the Figure 5.13. The SAC algorithm converges to a cosine similarity of 0.99 on average with a mean episode length of 4. The mean rotational and translation pose error are 3deg and 1.3 cm respectively.



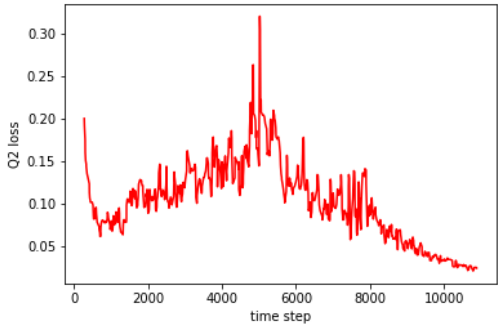
(a) SAC: episode reward



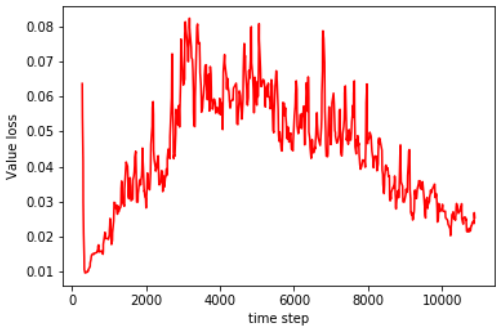
(b) SAC:loss entropy



(c) SAC: Q1 loss

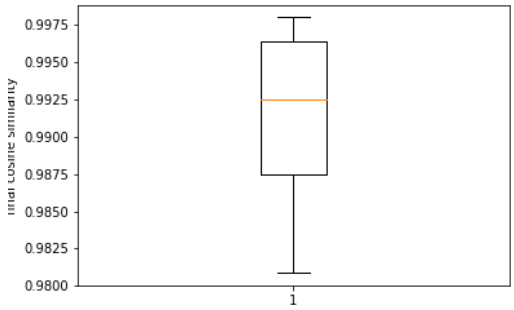


(d) SAC: Q2 loss

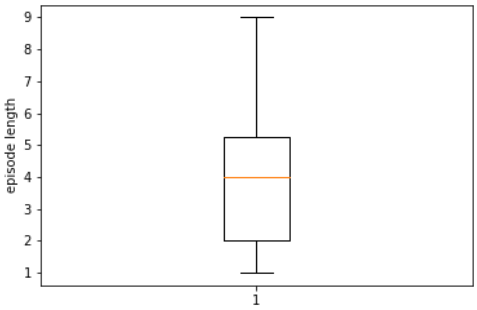


(e) SAC: value loss

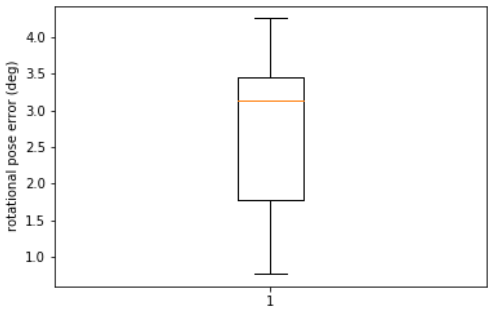
Figure 5.12: Reinforcement Learning



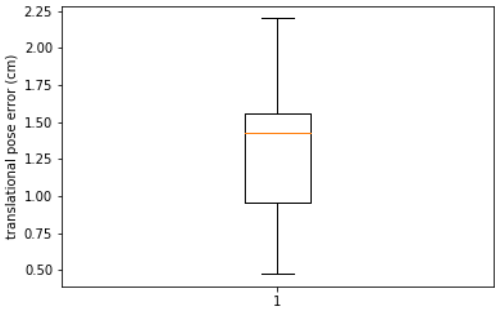
(a) Final cosine similarity



(b) episode length



(c) rotational pose error



(d) translational pose error

Figure 5.13: Reinforcement Learning

6

Conclusions

This thesis aimed at developing algorithms to successfully guide the robot arm to find a relative target pose between the object of interest and the robot arm itself suitable for robotic manipulation tasks. A RGB-camera was mounted on the TCP of the robot arm. An encoder was used to obtain the latent embedding of the RGB image captured by the TCP mounted camera. A random search algorithm was implemented followed by CMA-ES algorithm and the algorithms were compared with respect to the final cosine similarity, rotational pose error and the transnational pose error that they can achieve.

Although both random search and CMAE-ES achieve comparably similar success rates of convergence of 95 percent, it was found that CMA-ES converges to better cosine similarity and less rotational and transnational pose error when the algorithms do not reach the threshold cosine similarity.

Further experiments were carried out by modifying the fitness function of the CMA-ES algorithm so as to penalise the solutions resulting in the unreachable poses for the TCP of the robot arm. However, it was found that although the number of solutions resulting in the unreachable poses for the TCP of the robot arm decreases with evolving generations, the algorithm fails to reach the threshold cosine similarity even after many generations.

Further,the problem of unreachable poses of the TCP was successfully addressed by the reinforcement algorithm. After training, the reinforcement learning outperforms the random search and CMA-ES not only by successfully avoiding the unreachable

poses but also reaches the threshold cosine similarity in the range of 4-6 iterations on average. It was also found that the encoder is susceptible to the lighting conditions which occurs due to the shadow caused by the robot arm. Hence, it is required to train the encoder again in order to mitigate this effect.

6.1 Future work

Since the robot arm has to always avoid the unreachable poses, reinforcement learning is a good direction in this step. The method employed in this thesis requires the agent to be trained again for a previously unseen object. The agent can be made to generalise well to a new object by employing techniques of Meta-learning, which is a promising direction for the future work on this thesis.

A

Parameters

The default parameters for the CMAE-ES algorithm as according to [16]

$$\lambda = 4 + \lfloor 3lnn \rfloor \; can \;\; be \;\; increased \tag{A.1}$$

$$w_i{}' = ln\tfrac{\lambda+1}{2} - ln\, i \quad for \;\; i = 1, ..., \lambda \;\; preliminary \;\; convex \;\; shape \tag{A.2}$$

$$\alpha_\mu^- = 1 + \frac{c_1}{c_\mu} \quad let \;\; c_1 + c_\mu \sum w_i = c_1 + c_\mu - c_\mu \sum |w_i|^- \quad be \;\; 0 \tag{A.3}$$

$$\alpha_{\mu_{eff}}^- = 1 + \frac{2\mu_{eff}^-}{\mu_{eff} + 2} \quad bound \;\; \sum |w_i|^- \;\; to \;\; be \;\; compliant \;\; with \;\; c_\mu(\mu_{eff}) \tag{A.4}$$

$$\alpha_{pos\ def}^- = \frac{1 - c_1 - c_\mu}{nc_\mu} \quad \sum |w_i|^- \;\; to \;\; guaranty \;\; positive \;\; definiteness \tag{A.5}$$

$$w = \begin{cases} \frac{1}{\sum |w_j'|^+} w_i' & if \;\; w_i' \geq 0 \;\; positive \;\; weights \;\; sum \;\; to \;\; one \\ \frac{min(\alpha_\mu^-, \alpha_{\mu_{eff}}^-, \alpha_{pos\ def}^-)}{\sum |w_j'|^-} w_i' & if \;\; w_i' < 0 \;\; negative \;\; weights \;\; usually \;\; sum \;\; to \;\; -\alpha_\mu^- \end{cases} \tag{A.6}$$

$$c_m = 1 \tag{A.7}$$

Step-size control:

$$c_\sigma = \frac{\mu_{eff} + 2}{n + \mu_{eff} + 5} \tag{A.8}$$

$$d_\sigma = 1 + 2 \;\; max(0, \sqrt{\frac{\mu_{eff} - 1}{n+1}} - 1) + c_\sigma \tag{A.9}$$

Covariance matrix adapttion:

$$c_c = \frac{4 + \frac{\mu_{eff}}{n}}{n + 4 + 2\frac{\mu_{eff}}{n}} \tag{A.10}$$

$$c_1 = \frac{\alpha_{cov}}{(n+1.3)^2 + \mu_{eff}} \quad with \;\; \alpha_{cov} = 2 \tag{A.11}$$

$$c_\mu = min(1 - c_1, \alpha_{cov} \frac{\mu_{eff} - 2 + \frac{1}{\mu_{eff}}}{(n+2)^2 + \alpha_{cov} \frac{\mu_{eff}}{2}}) \quad with \;\; \alpha_{cov} = 2 \tag{A.12}$$

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):2638, Nov 2017. ISSN 1053-5888. doi: 10.1109/msp.2017.2743240. URL http://dx.doi.org/10.1109/MSP.2017.2743240.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2015.

[3] H. Ben Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters. Generalization of human grasping for multi-fingered robot hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2043–2050, 2012.

[4] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham, 2014. Springer International Publishing.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2016.

[7] Thanh-Toan Do, Ming Cai, Trung Pham, and Ian Reid. Deep-6dpose: Recovering 6d object pose from a single rgb image, 2018.

[8] Guoguang Du, Kai Wang, and Shiguo Lian. Vision-based robotic grasping from object localization, pose estimation, grasp detection to motion planning: A review. 2019.

[9] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017.

[10] Max Fischer, Patrick van der Smagt, and G. Hirzinger. Learning techniques in a dataglove based telemanipulation system for the dlr hand. pages 1603 – 1608 vol.2, 06 1998. ISBN 0-7803-4300-X. doi: 10.1109/ROBOT.1998.677377.

[11] Justin fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. 05 2018.

[12] Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mrten Bjrkman. Deep predictive policy training using reinforcement learning, 2017.

[13] Ross Girshick. Fast r-cnn, 2015.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[16] Nikolaus Hansen. The cma evolution strategy: A tutorial, 2016.

[17] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollr, and Ross Girshick. Mask r-cnn, 2017.

[19] Stefan Hinterstoißer, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Kurt Konolige, Gary R. Bradski, and Nassir Navab. Technical demonstration on model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ECCV Workshops*, 2012.

[20] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, Feb 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00393.

[21] Sulabh Kumra and Christopher Kanan. Robotic grasp detection using deep convolutional neural networks. 2016.

[22] Robert Kwiatkowski and Hod Lipson. Task-agnostic self-modeling machines. *Science Robotics*, 4:eaau9354, 01 2019. doi: 10.1126/scirobotics.aau9354.

[23] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies, 2015.

[24] Yunhui Liu and Mei Wang. Qualitative test and force optimization of 3d frictional force-closure grasps using linear programming. *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, 4: 3335–3340 vol.4, 1998.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2014.

[26] AJ Piergiovanni, Alan Wu, and Michael S. Ryoo. Learning real-world robot policies by dreaming, 2018.

[27] J. Ponce, S. Sullivan, J. . Boissonnat, and J. . Merlet. On characterizing and computing three- and four-finger force-closure grasps of polyhedral objects. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 821–827 vol.2, May 1993. doi: 10.1109/ROBOT.1993.291933.

[28] Joseph Redmon and Anelia Angelova. Real-time grasp detection using convolutional neural networks. In *International Conference on Robotics and*

Automation (ICRA), 2015. URL http://www.vision.caltech.edu/anelia/publications/RedmonAngelova15GraspDetection.pdf.

[29] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[31] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[32] Anis Sahbani, Sahar El-Khoury, and Philippe Bidaud. An Overview of 3D Object Grasp Synthesis Algorithms. *Robotics and Autonomous Systems*, 60(3): 326–336, 2012. URL https://hal.archives-ouvertes.fr/hal-00731127.

[33] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[34] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, and Rudolph Triebel. Multi-path learning for object pose estimation across domains, 2019.

[35] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images, 2019.

[36] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1385–1391, 2004.

[37] Jay Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton, Rahul Chipalkatty, Mitchell Hebert, David Johnson, Jimmy Wu, Bolei Zhou, and Antonio Torralba. Segicp: Integrated

deep semantic segmentation and pose estimation. pages 5784–5789, 09 2017. doi: 10.1109/IROS.2017.8206470.

[38] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2017.

[39] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. Unsupervised visuomotor control through distributional planning networks, 2019.

[40] Yun Jiang, S. Moseson, and A. Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *2011 IEEE International Conference on Robotics and Automation*, pages 3304–3311, May 2011. doi: 10.1109/ICRA.2011.5980145.