1 Method

In this thesis

1.1 System Setup

The robot that was used throughout this thesis is the kuka iiwa fabricated by a company called .... To simulate said robot, the software vrept by ... was used

Algorithm 1 Base Line

```
1:  procedure MAIN
2:      EPISODES ← N
3:      ITERATIONS ← M
4:      Initialize simulation
5:
6:      for i ← 0, EPISODES do
7:
8:          Start Simulation
9:
10:         for j ← 1, ITERATIONS do
11:
12:             Sample new set of actions {a_{k,j}}_{k=1}^K with N(0, constraint/√2)
13:             Interpolate trajectory between {a_{k,j-1}}_{k=1}^K and {a_{k,j}}_{k=1}^K
14:             Add current state and action {s_k, a_{k,j}}_{k=1}^K to replay buffer R
15:             Stop simulation
16:
```

Figure 1: Code Base Line

1.2 Choosing new desired action

Three different ways of generating a new set of actions are presented here. The first method is to just simply sample a new action from a normal distribution, the second option is to calculate a new actions with the help of an autoencoder and the CMA-ES and the last option involves sampling through PCA.

1.2.1 Sampling from normal distribution

Sampling a new action from a normal distribution $X \sim \mathcal{N}(\mu, \sigma)$ is the simplest method computation wise and also the quickest. A random value is drawn from the normal distribution, where μ is equal to zero and σ is equal to $\frac{constraint}{\sqrt{2}}$ (See Figure 1, Line 12). In this thesis the constraint is referring to the angular speed of the joint and is set to 10 degrees. This process is repeated for every joint, seven in this thesis (See Figure 2).

```
# Get new end velocity and check whether it satisfies the constraints
for i in range(7):
    # Choose a random sample from a Gaussian normal Distribution
    vel_end[i] = rn.gauss(0,constraint/math.sqrt(2))
```

Figure 2: Code Base Line

After a action is sampled for each joint, it is checked whether the actions satisfy all the constraints. If even one of them is violated, the entire process is started anew.

1.2.2 Sampling through an Autoencoder and the CMA-ES

Another way of sampling a new set of actions is through the CMA-ES and an autoencoder. In chapter ... and .. it has been explained how they work and what they are used for individually. The process of how to sample a new set of actions is twofold.

The first part is to set up the autoencoder itself. In this thesis a two layer approach has been chosen. The first layer reduced the 21 dimensional input vector to a 16 dimensional vector. This vector is reduces further to five dimensions in the second layer (See Figure 3).

```
self.encoder = nn.Sequential(nn.Linear(21,16),nn.BatchNorm1d(16),nn.PReLU(),
                             nn.Linear(16,5),nn.BatchNorm1d(5),nn.PReLU())
self.decoder = nn.Sequential(nn.Linear(5,16),nn.BatchNorm1d(16),nn.PReLU(),nn.Linear(16,21))
```

Figure 3: Code Base Line

For the activation function PReLU was used. After the autoencoder has been setup, it is trained after each episode with all the to this point acquired data (See Figure 4, Line 18f).

The second step is the actual sampling of the actions with the help of the CMA-ES. In each iteration of the program the CMA-ES is initialized with the last action (the current velocity), a standard deviation of 0.25, a population size of 16 and a maximum of iterations of 50 (See Figure 5). That means that the CMA-ES runs at max 50 iterations and that each iteration 16 candidate solutions are being sampled by the algorithm.

```
# Iniitialize the cma-optimizer with the current actions, a popualtion of 32 and a std dev of 0.25
es = cma.CMAEvolutionStrategy(action,0.25,{'popsize':16,'maxiter':50})
```

Figure 4: Code Base Line

In each iteration of the CMA-ES, 16 sets of seven actions each are being

sampled. Each of those sets gets concatenated with the current state and propagated through the autoencoder to obtain the compressed representation. The compressed representation of the current state actions pairs are then evaluated on the compressed representation of all state actions pairs. The set of actions that yielded the best result (lowest density) is then chosen as the base for the next iteration of the CMA-ES. This is repeated until no better result can be obtained or 50 iterations are reached, The best overall result is the chosen as the new action pair.

Constraints

Algorithm 2 Autoencoder

```
1:  procedure MAIN
2:      EPISODES ← N
3:      ITERATIONS ← M
4:      Initialize simulation
5:      Initialize Autoencoder
6:
7:      for i ← 0, EPISODES do
8:
9:          Start Simulation
10:
11:         for j ← 1, ITERATIONS do
12:
13:             Calculate new set of desired actions {a_{k,j}}_{k=1}^K with CMA-ES
14:             Interpolate trajectory between {a_{k,j-1}}_{k=1}^K and {a_{k,j}}_{k=1}^K
15:             Add current state and action {s_k, a_{k,j}}_{k=1}^K to replay buffer R
16:             Stop simulation
17:
18:         Initialize Autoencoder with all the data in the replay buffer R
19:         Train Autoencoder
20:         Save loss and hidden layer data
```

Figure 5: Code Base Line