creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학석사 학위논문

최적 근사 경로 생성을 위한 회귀 인공 신경망 모델 : 구조 로봇 적용 사례

Recurrent Neural Network Architectures for Sub-Optimal Motion Generation: A Recue Robot Case Study

2019년 2월

서울대학교 대학원

기계항공공학부

박 성 균

ABSTRACT

Recurrent Neural Network Architectures for Sub-Optimal Motion Generation: A Recue Robot Case Study

by

Seonggyun Park

Department of Mechanical and Aerospace Engineering
Seoul National University

Rescue robots typically consist of one or more manipulators mounted on a mobile platform,and must operate in complex and unstructured environments with limited power. As such, motion planning for such high-dimensional systems must be done in close to real-time, producing motions that are efficient and meet all

task requirements. As an alternative to dynamics-based optimal trajectory generation, which requires accurate dynamic models and significant computation, particularly for high-dimensional systems, we present in this thesis an imitation learning-based recurrent neural network architecture for rapidly generating near-optimal trajectories. Specifically, we examine the Long-Short Term Memory-Mixture Density Network (LSTM-MDN) as a possible network for generating efficient motions. We train this network with dynamically optimal motions provided by a dynamics-based motion optimization algorithm, and use the network to generate real-time, near-optimal motions for a prototype mobile dual-arm rescue robot.

Keywords: Imitation Learning, LSTM-MDN, Robot Optimization, Rescue Robot.

Student Number: 2017-24254

Contents

Abstract **i**

List of Tables **vi**

List of Figures **vii**

1 Introduction **1**

1.1 Previous Research 2

1.2 Contributions of This Thesis 4

1.3 Organization 7

2 Rescue Robot Platform and Objective **9**

2.1 Hurcules Rescue Robot 9

2.1.1 Hardware Architecture 10

2.1.2 Software Architecture 11

2.2 Robot Dynamics Optimization 12

2.2.1 Problem Definition 12

2.2.2 Objective Function 15

2.2.3 Optimization Algorithm 16
2.3 Target Motion Description 18
2.3.1 Single-Arm Grasping 18
2.3.2 Dual-Arm Cylinder Lifting 19
2.3.3 Dual-Arm Human Rescuing 19

3 Long Short Term Memory
- Mixture Density Network(LSTM-MDN) 22
3.1 Long Short-Term Memory(LSTM) 22
3.2 Mixture Density Network(MDN) 26
3.3 LSTM-MDN 27
3.4 Pre-Process of Target Data 32
3.4.1 Data Preprocessing 32
3.4.2 Frequency Sampling 33
3.5 Post-Process of Generated Data 34

4 Results and Analysis 36
4.1 Result of LSTM-MDN Trajectory 37
4.1.1 Trajectory Comparison 37
4.1.2 Result of Trajectory Generation by Mass 38
4.2 Result of Post-Processing 39
4.3 Result Analysis 39
4.4 Generate Untrained Trajectory 40

5 Conclusion 50

Bibliography 53

Abstract **56**

List of Tables

4.1 Comparison of single-arm grasping 40
4.2 Comparison of dual-arm cylinder lifting 41
4.3 Comparison of dual-arm human rescuing 41
4.4 Comparison of untrained trajectory 42

List of Figures

2.1 The conceptual working environment of Hurcules 10
2.2 Hardware architecture of the Hurcules 11
2.3 Software architecture of the Hurcules 12
2.4 Example of Single-arm grasping 19
2.5 Example of dual-arm cylinder lifting 20
2.6 Example of dual-arm human rescuing 21

3.1 The vanishing gradient problem. 23
3.2 The structure of LSTM cell 24
3.3 The concept of MDN 26
3.4 The result of L2 loss function 28
3.5 The result of MDN 29
3.6 LSTM-MDN training model architecture 31
3.7 LSTM-MDN prediction model architecture 31
3.8 Example of frequency sampling 33

4.1 Training loss 37
4.2 Result of single-arm grasping 43

4.3 Result of dual-arm cylinder lifting 44
4.4 Result of dual-arm human rescuing 45
4.5 Dual-arm lifting trajectory by object weight 46
4.6 Dual-arm rescuing trajectory by object weight 47
4.7 Result of post-processing 48
4.8 Result of perturbation 48
4.9 Generation of untrained trajectory 49

1

Introduction

A rescue robot is a robot has a purpose to help or rescue people in a disaster situation [1]. Their work environment is extremely different from industrial or living environment. Also, since the main task of robots is to rescue people, it is absolutely necessary to move heavy objects in response to these harsh conditions. Normally, these robots operate wirelessly, they have a limited energy source like a battery. Therefore, heavy object manipulation optimization is an important factor in limited energy source situations.

If accurate robot model exists, dynamics based trajectory optimization in near real time is now possible for a low-DoF system [2] [3]. However, optimal path generation in a high-dimensional model is complicated and consumes considerable time. Another situation is that it is difficult to optimize and converge even when the motion duration time is long. Lastly, the parametric relationship of the model is unclear in performing the optimization and thus the optimized trajectory may be sensitive to certain parameters.

The problems in the robot model, as well as the work environment, can not

be ignored. The working environment of the rescue robot is very unpredictable and time constrained. It is especially important to achieve the mission within the target time, especially for tasks such as the rescue of injured, the main goal of rescue robots.

Due to the problems described above, using the existing optimization algorithm takes long calculation time and does not converge frequently, which causes problems in mission performance. Therefore, it is necessary to approach other methods for near real time generation of high-quality motions for complex robots operating in an unstructured environment.

Recently, many researchers have been studying artificial neural networks. Particularly, it showed great progress in classification and regression problems. Reinforcement learning became a big issue through playing go with the top human player [4].

Efforts have been made to apply artificial neural networks not only to this but also to robot motion learning. Although the results of the research so far have taken a long time to train, it has been confirmed that the motion generation is performed in real time once it is properly trained [5].

Our approach is to apply the machine learning technique that can generate motion real-time to the optimal trajectory generation so that the rescue robot can satisfy the mission fulfillment condition.

1.1 Previous Research

The first machine learning method applied to robot motion generation is reinforcement learning. Reinforcement learning, which has shown remarkable progress in

virtual environments like Atari games and Go, is designed to have the highest reward value through the reward function. However, the robot performance was very different depending on how the reward function was designed and the training time was significantly long [1].

To ease the problem, imitation learning appeared. Imitation Learning is a type of machine learning which is a sequential task where the learner tries to mimic an expert's action in order to achieve the best performance [6]. Unlike Reinforcement Learning, learning from demonstration does not need to make a complicated reward function because it learns the desired action directly. Thus, it can shorten the learning time.

Robot actions along the trajectories are interdependent, as actions determine state transitions and thus states and actions down the road. Since this inter-dependency forms structure, many supervised learning models for policy have been proposed as a direct way.

Learning to search(L2S) algorithms such as Dataset AGGregation(DAgger) reduce this problem to learn a policy to adjust in a search space with minimum cost [7]. Still, L2S algorithms are problematic in terms of data sparsity when it comes to test data that was not previously experienced and make a poor prediction.

To solve this problem, a new model combining L2S search space and recurrent neural network(RNN) training method has emerged. Adopting the L2S search space mechanism, RNN can encode the action sequences for the training process. Furthermore, learning continuous space has made it possible to ease the problem of data sparsity and improve the performance of L2S [8].

Yet updating weight through back-propagation in RNN training process causes

the vanishing gradient in long time step. A number of studies have been developed to overcome this issue. One approach is combining Long-Short Term Memory(LSTM) with Mixture-Density Network(MDN). LSTM adopts gates to deal with exploding or vanishing gradient problem. Also, by predicting mixture weights over a Gaussian mixture model at the output with mean and variance for the mixture component, it is able to learn the diverse possible outcome [9].

Another important point is that the model extends the number of training data through frequency sampling and also helped to solve the data sparsity problem through data augmentation [5].

1.2 Contributions of This Thesis

The motivation of this study is as follows. The working environment of a rescue robot is various and resources are limited. The existing optimization methods show stable and good performance results, but it is inadequate to use because of the time and space limitations in the above environment. Therefore, we pay attention to the trajectory generation method through machine learning. However, the existing research has a problem to use immediately so that the motion is constituted only by the no-constraint in the simple and open-chain. Therefore, it is necessary to use a modification of the conventional machine learning model to suit the characteristics and work environment of the rescue robot.

Pre-processing for Whole Body Motion Generation with Neural Network

The researches that tried to control the robot through the above-mentioned machine learning have shown considerable improvement compared with the previous studies, but it is inappropriate to apply it directly to the rescue robot system

of this study. This is because the existing research is focused on simple operation with no constraints at low-dimension like single-arm pick and place.

As the DoFs of the robot increases, the number of possible actions from the initial posture increases exponentially, which causes significant time to learn and predict. Therefore, it is difficult to confirm the performance of the model with low complexity robot motion.

As will be explained in more detail later, rescue robots are articulated robots with 19 to 21 degrees of freedom. Applying the model of the previous research as it is to generate the motion of the robot causes a problem. Therefore, it is necessary to modify artificial neural network model that can generate training data suitable for current system and train this data.

In this study, we use high - dimensional whole body motion data to increase the complexity that artificial neural networks learn. We also pre-processed these data to reflect our desired behavior and constraints. Furthermore, we modified the LSTM-MDN model to train our pre-processed data.

Reduce Optimization Time and Computing Energy

The most important subject of this study is to use a machine learning model to generate optimal motion of a jointed-arm robot. Rescue robots operate as a battery due to the nature of their operating environment, which means that there are restrictions on the radius of action and the running time.

As mentioned earlier, the efficiency of the existing optimization algorithms varies greatly according to the time required, and it takes a relatively long time to generate a trajectory with proper efficiency. It takes a considerable amount of energy to maintain the robot combined with various systems such as the rescue robot, and the long calculation time itself has a fatal effect on the utilization of the rescue

robot.

Furthermore, the most important mission of the rescue robot, carrying a wounded, aimed at a seriously injured soldier who can not return to himself in a dangerous situation. In the worst case, if the time for optimization takes long while performing such a task, the injured may die.

Therefore, shortening the time required to generate motion is the most important factor directly related to the improvement of rescue robot performance. Although artificial neural networks consume much time to train, once they are fully trained, they can produce results much faster. Through this approach, it can overcome the above limitations and achieve real-time optimized motion planning.

Post-processing for Reliable Trajectory Generation

The biggest problem of robot control through machine learning is that it relies heavily on learning data. Existing control algorithms can generate paths considering the constraints of robots, but learning based algorithms are only trained using learning data that includes the above constraints and are not considered in path generation phase.

This feature may cause the following problem. When our model applied to an actual robot, it can damage the robot itself. Since only the trajectory with respect to time is stored in the learning data, the physical limitations of the robot are not considered so that the model may generate trajectory which damages robot joint. If the learning data includes these constraints, the data itself may become unnecessarily large.

The other is the learning model can not cope with imperfect forecasting at

the early stage of path generation. An incomplete prediction is due to the characteristics of the RNN that predict the value for the next time by applying back-propagation. At the beginning of the prediction, the time to perform back-propagation is small and therefore the value becomes incomplete. This is more complicated to solve because it is a problem that comes from the model itself, not from the data that comes from the learning data.

Therefore, it is necessary to examine the validity of the generated path even after the path generation proceeds through the model. In this study, we formulate the post-processing algorithm to test the two problems described above.

1.3 Organization

In Chapter 2, we briefly describe the platform used in this study. First, we explain the hardware and software of Hercules, a rescue robot designed by ADD. The following shows the motion generation for training data from problem definition and what algorithm we adopt for the trajectory generation Finally, we describe what target motions we decide to imitate and why theses motions are important.

Chapter 3 shows the underlying foundations of LSTM-MDN neural network and how we change the original network. First, we explain basic LSTM and MDN separately to explain the characteristic of each component. Then, we show how different models combine into one LSTM-MDN and the architecture of the model. Next, we describe how we preprocess target motion data as we define our training motions at Chapter 2 and what modification we make to train preprocess data. Finally, the post-process method will be explained to satisfy the constraints of the motion.

Chapter 4 shows the result of the explained methods above. We compare the

objective function values and computing time of the original optimization trajectories with neural network generated trajectories. In addition, we simulate each configuration trajectory to check the validity of the motion.

Finally, we summarize the research process and the main result to conclude our thesis in Chapter 5.

2

Rescue Robot Platform and Objective

This chapter covers some ground concept of robot dynamic optimization and Long Short Term Memory - Mixture Density Network(LSTM-MDN) that are the basis our thesis.

2.1 Hurcules Rescue Robot

In this study, we used Hercules robots to create, learn and imitate optimal motion. The Hercules rescue Robot is a new humanoid robot made by Agency for Defense Development(ADD) and is responsible for rescuing the wounded from the battle-field [10]. This section describes the hardware and software of this robot to justify the target motions we have created in the next section. The conceptual working environment of Hurcules robot is shown in Figure 2.1

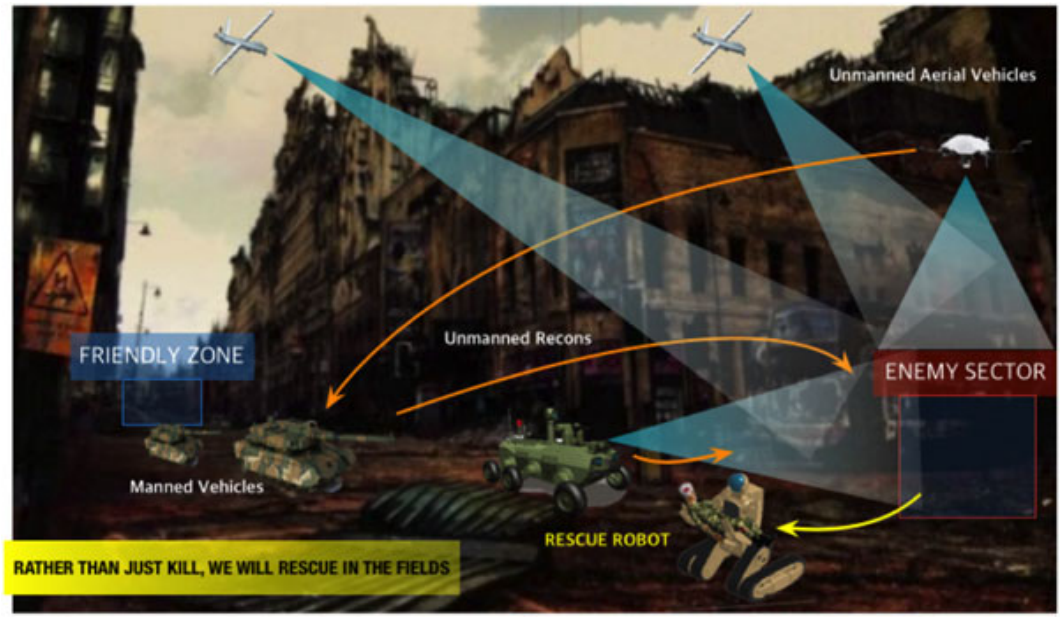


Figure 2.1: The conceptual working environment of Hurcules, repriented form [10].

2.1.1 Hardware Architecture

The robot consists of a tracked lower body(TLB) and an upper body with dual arms attached. The gripper can grasp or lift an object. In addition, the camera is attached to the head to visualize the surrounding environment. It has a total of 19 to 21 DoFs, which has 10 DoFs in the arms, 2 DoFs in the neck, 7 DoFs in the lower body and can add 0 or 2 DoFs depending on the structure of the hand.

The structure of the hand can be changed according to the mission required to the robot, and all movements are focused on the rescue mission. The Hercules robot uses worm gears on all joints because all joints are designed to stop for the safety of the wounded being rescued even when the power is off. The overall shape of the robot can be seen in Figure 2.2. More detailed information is written

on [11].

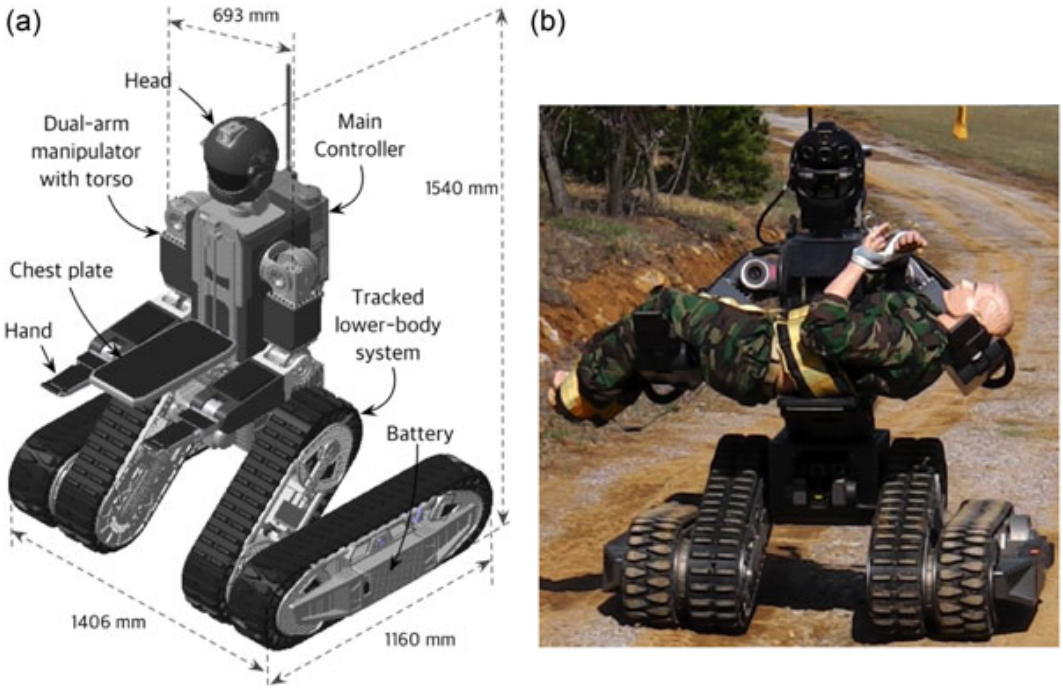


Figure 2.2: Hardware architecture of the Hurcules repriented form [10]. (a) Mechanical design, (b) Hardware configuration holding a casualty.

2.1.2 Software Architecture

The Hurcules robot runs on three computers. The first computer handles robot global positioning calculation and path generation. The second is responsible for visualizing the data from the sensors in the head, and the final computer generates the robot's motion trajectory and manipulates joints.

The motion of the robot is based on Ubuntu Linux 12.04 (Linux Kernel 3.4.6)

and the Xenomai 2.6.3 real-time patch. Hercules communicates with the joint manipulator and all operations are performed under the Xenomai real-time task application programming interfaces.

When creating the real-time trajectory and transmitting the data, the memory is shared to increase the convenience and communication speed of the robot [12]. Figure 2.3 shows the software structure of the Hurcules rescue robot [10]. More specific information can be found at [13].

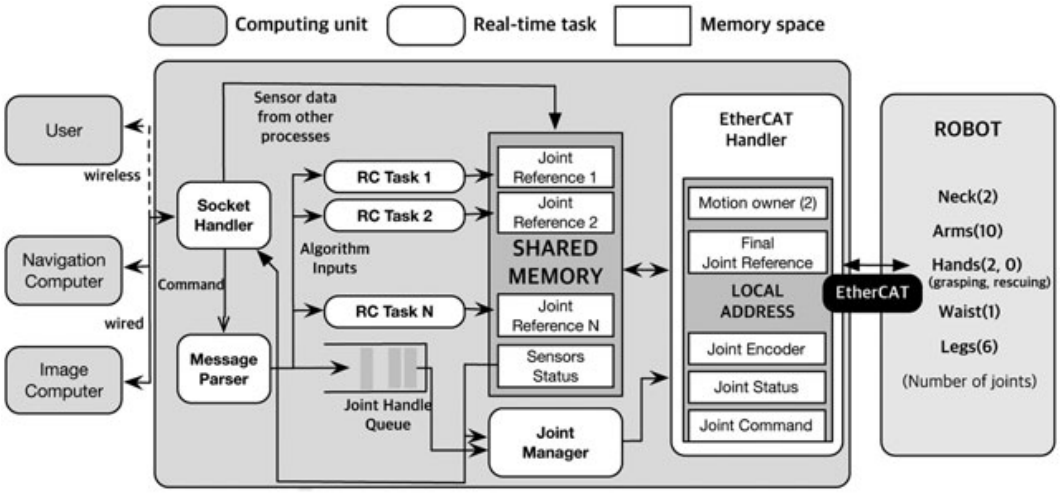


Figure 2.3: Software architecture of the Hurcules repriented form [10].

2.2 Robot Dynamics Optimization

2.2.1 Problem Definition

Modeling of robot dynamics is needed to generate the optimal path of a joint system considering dynamics. The dynamic equation of the articulated robot model

is as follows.

$$\tau = M(q)q'' + C(q, q')q' + G(q) + f(q') \tag{2.2.1}$$

Where $q(t)$ is the joint angle vector, $M(q)$ is the mass matrix, $C(q, q')q'$ is the Coriolis vector, $G(q)$ is the force vector by gravity, $f(q')$ is the frictional force on the joint, and τ is the torque vector on the joint. The most used and simple frictional force model is as follows [14].

$$f(q') = K_v q' + K_c sgn(q') \tag{2.2.2}$$

Here, K_v and K_c are diagonal matrices, viscous and Coulomb friction coefficients, respectively. Input torque from the motors can be expressed as joint torque vector as below.

$$\tau_m = G^{-1}\tau \tag{2.2.3}$$

Where G stands for the diagonal matrix combining gear ratio from each motor. We can efficiently solve the inverse kinematics of the dynamics model presented above by using the iterative Lie group Newton-Euler algorithm shown as Algorithm 1 [3].

Then, the point-to-point motions planning problem can be defined as an optimal control problem as follows.

Algorithm 1 Recursive Inverse Dynamics

initialization $V_0, \dot{V}_0, F_{n+1}$
procedure - forward recursion
for $k = 1$ **to** n **do**
    $f_{k-1,k} = M_k e^{S_k q_k}$
    $V_k = \mathrm{Ad}_{f_{k-1,k}^{-1}} V_{k-1} + S_k \dot{q}_k$
    $\dot{V}_k = \mathrm{Ad}_{f_{k-1,k}^{-1}} \dot{V}_{k-1} + S_k \ddot{q}_k + \mathrm{ad}_{V_k} S_k \dot{q}_k$
end for
procedure - backward recursion
for k=n **to** 1 **do**
    $F_k = \mathrm{Ad}^*_{f_{k,k+1}^{-1}} F_{k+1} + J_k \dot{V}_k - \mathrm{ad}^*_{V_k} J_k V_k$
    $\tau_k = S_k^{\mathrm{T}} F_k$
end for

$$
\begin{aligned}
&\text{Minimize} \quad J = \int_{t_0}^{t_f} L(q, q', q'')dt \\
&\text{Subject to } f(q, q', q'') = \tau \\
&\qquad \underline{q} \leq q(t) \leq \overline{q},\ \underline{q'} \leq q'(t) \leq \overline{q'} \\
&\qquad \underline{q''} \leq q''(t) \leq \overline{q''},\ \underline{\tau} \leq \tau(t) \leq \overline{\tau} \\
&\qquad q(t_0) = q_0,\ q'(t_0) = q'_0,\ q''(t_0) = q''_0 \\
&\qquad q(t_f) = q_f,\ q'(t_f) = q'_f,\ q''(t_f) = q''_f
\end{aligned}
\tag{2.2.4}
$$

Here, f means the robot dynamics mentioned above. There are joint angles and velocity acceleration constraints, and there is also a range of values for the torque applied to the joints. Boundary conditions in the initial and final postures are also given. If the objective function is defined as the final time of operation,

it becomes an optimization problem that generates the fastest operation, and if it is defined as the mechanical power, it becomes an optimization problem that minimizes mechanical power.

2.2.2 Objective Function

The objective function to optimize the energy can be defined in various ways, but in this study, the objective function that minimizes the mechanical power is defined as follows.

$$J_{mp} = \int_{t_0}^{t_f} \sum_{j=0}^{n} max(\tau_j q'_j, 0) dt \tag{2.2.5}$$

When torque and joint velocity directions are opposite, the motors cannot obtain energy from the joints. Since this energy is not able to use, we define energy consumption as 0 in this case.

Numerical integration must be performed when calculating the objective function. In this study, a numerical integration method called Gaussian quadrature (GQ) was used.

An n-point Gaussian quadrature states as a weighted sum of function values which is defined over[-1, 1], at specified points:

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i), \quad i = 1, \cdots, n. \tag{2.2.6}$$

There are several ways to get points x_i and weights w_i. We use the method called Gauss-Legendre quadrature which uses Legendre polynomials [15]. Before using the Gaussian quadrature method, an integral over[0,T] must be changed into an integral over[-1,1]. Changing interval can be done by following way:

$$\begin{aligned}\int_0^T \tilde{E}(\tilde{t}(t))dt &= T\int_0^1 \tilde{E}(\tilde{t})d\tilde{t} \\ &= \frac{T}{2}\int_{-1}^1 \tilde{E}(\frac{1}{2}\tilde{t}+\frac{1}{2})d\tilde{t}\end{aligned} \tag{2.2.7}$$

Applying the Gaussian quadrature method, we can get the following equation:

$$\int_0^T \tilde{E}(\tilde{t}(t))dt = \frac{T}{2}\sum_{i=1}^n w_i\tilde{E}(\frac{1}{2}\tilde{t}_i'+\frac{1}{2}) \tag{2.2.8}$$

where $(w_i, \tilde{t}_i')$ are the weights and the points of an n-point Gaussian quadrature.

2.2.3 Optimization Algorithm

There are two main ways to solve the optimal control problem. One is the direct method and the other is the indirect method. In the case of the indirect method, the optimal control problem is transformed into a given differential equation problem by using the principle of optimality.

In the direct method, the optimal control problem is transformed into a finite dimensional nonlinear problem by parameterizing the parameter to be optimized with the finite number of parameters. From the objective function above :

$$\begin{aligned}&\text{Minimize} \quad J_{mp} = \int_0^{t_f} \tau^T(t)\dot{q}(t)dt \\ &\text{Subject to } q_{i,min}\leq q_i(t)\leq q_{i,max},\ \tau_{i,min}\leq\tau_i(t)\leq\tau_{i,max} \\ &\qquad 0\leq t\leq t_f, p_{ZMP}\in S, \text{where} S \text{ is support polygon}\end{aligned} \tag{2.2.9}$$

In this study, the direct method was used to solve the optimal control. Therefore, we approximate objective function to a finite nonlinear function.

$$J_{mp} = \int_0^{t_f} \tau^T(t)\dot{q}(t)dt \approx \int_{i=1}^{n} w_i \tau^T(t_i)\dot{q}(t_i) \tag{2.2.10}$$

Then, find the gradient of J with respect to p

$$\frac{\partial J_{mp}}{\partial p} = \int_{i=1}^{n} 2w_i(\dot{q}^T(t_i)\frac{\partial \tau(t_i)}{\partial p} + \tau^T(t_i)\frac{\partial \dot{q}(t_i)}{\partial p}) \tag{2.2.11}$$

The parameter to be optimized was mediated through the spline method called B-spline [2].

The B-spline curve basically consists of a basis $B_i(t)$ function and a control point $p_j = (p_{j1}, p_{j2}, \cdots, p_{jk})$. The angular value of the joint can be mediated through B-spline, which is expressed as:

$$q_j(t, p_j) = \sum_{i=1}^{k} B_i(t)p_{ji} \tag{2.2.12}$$

When using the direct method to solve the optimal control problem, it is very important to obtain the analytic gradient of the objective and the constraint function in terms of stability of speed and convergence.

If the torque is included in the objective or the constraint function, a value $\frac{\partial \tau}{\partial p_{ji}}$ that is differentiated the torque by a parameter is required. To do this, it is necessary to differentiate the robot dynamics differential equation as a parameter.

It is very complicated to directly differentiate the Lagrangian type equation as a parameter. Using the Lie group based repetitive differential dynamics algorithm, the torque can be obtained as a parameterized result very efficiently. The algorithm is written as Algorithm 2 [3].

Algorithm 2 Derivative of the Recursive Inverse Dynamics

initialization $\frac{\partial V_0}{\partial p}, \frac{\partial \dot{V}_0}{\partial p}$

procedure - forward recursion

for $i = 1$ **to** n **do**

$\frac{\partial V_i}{\partial p} = \mathrm{Ad}_{T_{i-1,i}^{-1}} \frac{\partial V_{i-1}}{\partial p} + S_i \frac{\partial \dot{q}_i}{\partial p} - \mathrm{ad}_{S_i} V_i \frac{\partial q_i}{\partial p}$

$\frac{\partial \dot{V}_i}{\partial p} = \mathrm{Ad}_{T_{i-1,i}^{-1}} \frac{\partial \dot{V}_{i-1}}{\partial p} + S_i \frac{\partial \ddot{q}_i}{\partial p} - \mathrm{ad}_{S_i} \left\{ \frac{\partial q_i}{\partial p} \mathrm{Ad}_{T_{i-1,i}^{-1}} \dot{V}_{i-1} + \frac{\partial V_i}{\partial p} \dot{q}_i + V_i \frac{\partial \dot{q}_i}{\partial p} \right\}$

end for

procedure - backward recursion

for i=n **to** 1 **do**

$\frac{\partial F_i}{\partial p} = \mathrm{Ad}^*_{T_{i,i+1}^{-1}} \left(\mathrm{ad}^*_{-S_{i+1}} F_{i+1} \frac{\partial q_{i+1}}{\partial p} + \frac{\partial F_{i+1}}{\partial p} \right) + \mathcal{J}_i \frac{\partial \dot{V}_i}{\partial p} - \mathrm{ad}^*_{\frac{\partial V_i}{\partial p}} \mathcal{J}_i V_i - \mathrm{ad}^*_{V_i} \mathcal{J}_i \frac{\partial V_i}{\partial p}$

$\frac{\partial \tau_i}{\partial p} = S_i^{\mathrm{T}} \frac{\partial F_i}{\partial p}$

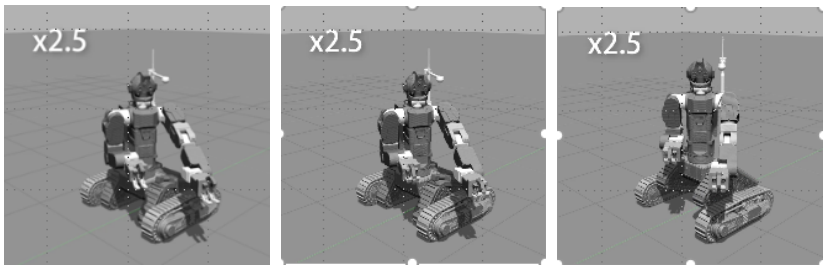
end for

2.3 Target Motion Description

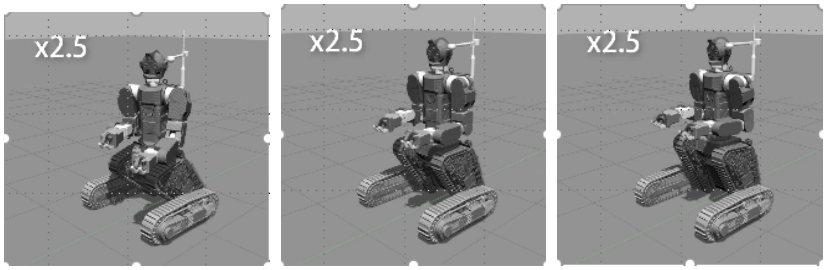
In this study, We hypothesize the most possible motions that the robot can perform in a rescue situation. We select three motions for our study: Single-arm grasping, Dual-arm cylinder lifting and Dual-arm human rescuing.

2.3.1 Single-Arm Grasping

Single-arm grasping assumes that the rescue robot collects necessary item before starting to operate. Therefore, we generate one-hand optimization paths of grasping an object on the left or right side of the body in accordance with the weight of the object. Figure 2.4 shows Single-arm grasping motion.



(a) Step 1 (b) Step 2 (c) Step 3



(d) Step 4 (e) Step 5 (f) Step 6

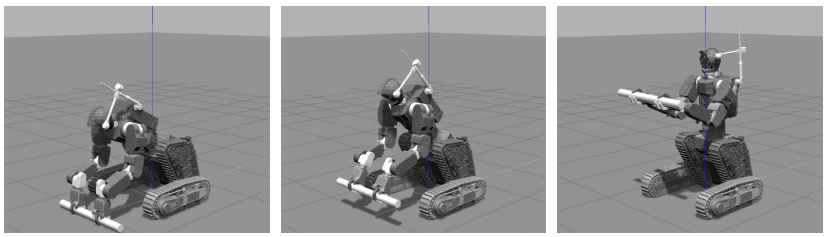
Figure 2.4: Example of Single-arm grasping

2.3.2 Dual-Arm Cylinder Lifting

Dual-arm cylinder lifting assumes that the rescue robot lifts a metered object such as a cylinder using both hands. This time, we optimized two hands trajectory according to the weight of the object. Figure 2.5 exhibits Dual-arm cylinder lifting motion.

2.3.3 Dual-Arm Human Rescuing

Lastly, Dual-arm human rescuing is the case of changing the grippers of the rescue robot and rescuing a human being. Similar to cylinder lifting, the robot uses both hands to rescue a person. However, depending on the height and weight of the

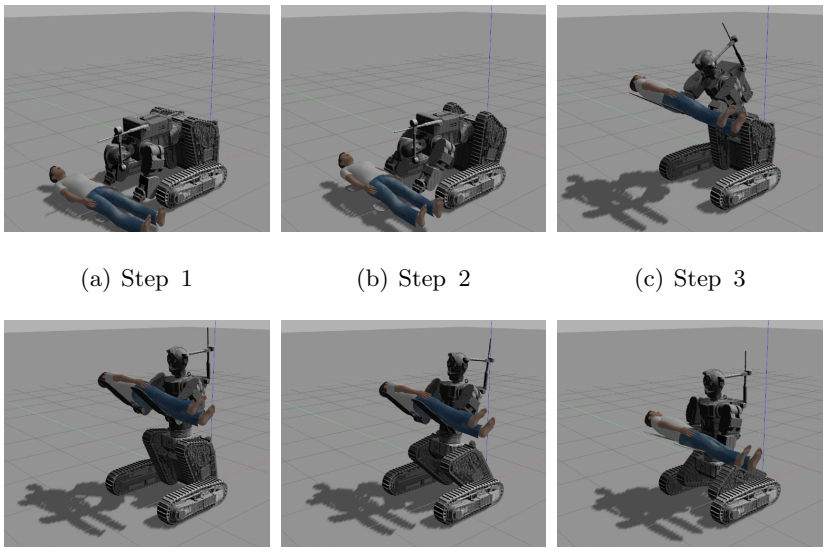


(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

Figure 2.5: Example of dual-arm cylinder lifting

rescuer, the final position of two hands must change. Figure 2.6 presents Dual-arm human rescuing motion.



(a) Step 1 (b) Step 2 (c) Step 3

(d) Step 4 (e) Step 5 (f) Step 6

Figure 2.6: Example of dual-arm human rescuing

3

Long Short Term Memory - Mixture Density Network(LSTM-MDN)

LSTM-MDN is an artificial neural network that combines LSTM and MDN. MDN parameters are expressed as the result of LSTM, and the learned LSTM-MDN expresses state x_{t+1} at time $t+1$ as probability distribution with respect to state x_t at time t [16].

3.1 Long Short-Term Memory(LSTM)

In order to learn motion data over time, it is appropriate to use Recurrent Neural Network (RNN) artificial neural network model. RNN is an artificial neural network that forms a regression structure and is an algorithm suitable for learning data sequentially appearing such as voice, text, and the like.

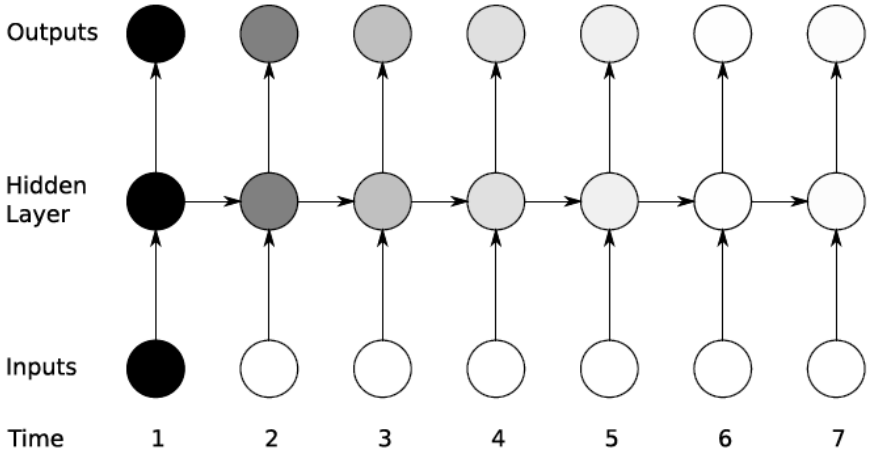


Figure 3.1: The vanishing gradient problem, repriented form [17].

In the case of conventional RNN, the vanishing or exploding gradient problem which is the gradient of the learning error converges to zero or infinite over time occurs through training process [18]. Figure 3.1 shows the vanishing gradient problem.

A modification of this vanilla RNN that solves these problems is a long short-term memory(LSTM) [19]. LSTM solved the problem of the vanishing or exploding gradient by adding input, output and forget gates to conventional RNN. The structure of the single cell of the LSTM is shown in Figure 3.2 [20].

By adopting the gated structure, LSTM controls the total amount of gradient from the past cells. The activation process during the forward pass at each gate can be expressed as follow [17]:

Input Gates

$$\begin{aligned} \alpha_{input}^{t} &= \sum_{i=1}^{I} w_{i,input} x_i^t + \sum_{h=1}^{H} w_{h,input} \beta_h^{t-1} + \sum_{c=1}^{C} w_{c,input} s_c^{t-1} \\ \beta_{input}^{t} &= f(\alpha_{input}^{t}) \end{aligned} \tag{3.1.1}$$

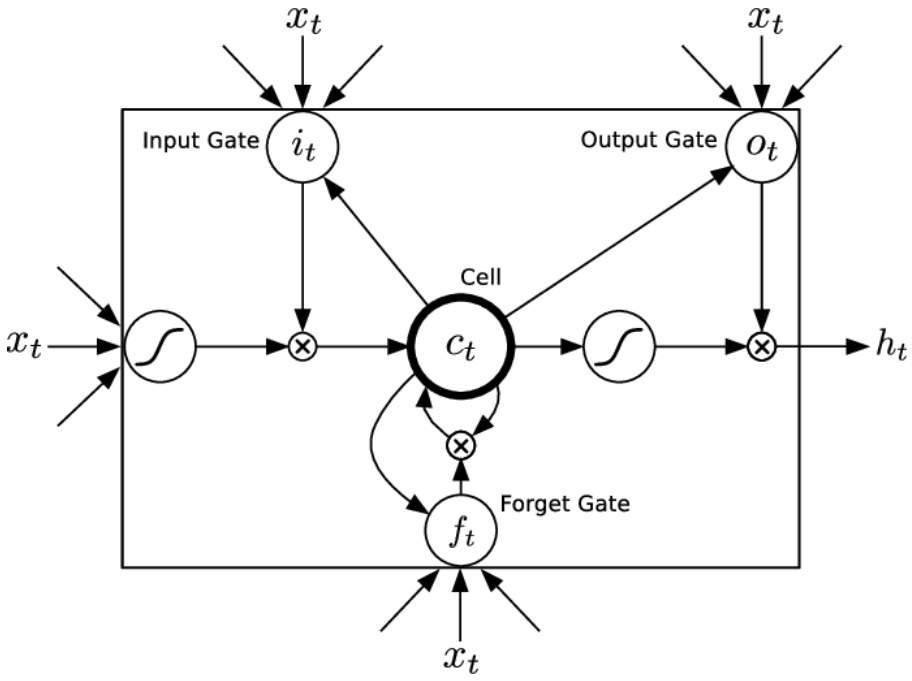


Figure 3.2: The structure of LSTM cell, repriented form [17].

Forget Gates

$$\begin{aligned} \alpha_{forget}^t &= \sum_{i=1}^{I} w_{i,forget} x_i^t + \sum_{h=1}^{H} w_{h,forget} \beta_h^{t-1} + \sum_{c=1}^{C} w_{c,forget} s_c^{t-1} \\ \beta_{forget}^t &= f(\alpha_{forget}^t) \end{aligned} \tag{3.1.2}$$

Cells

$$\begin{aligned} \alpha_c^t &= \sum_{i=1}^{I} w_{i,c} x_i^t + \sum_{h=1}^{H} w_{h,c} \beta_h^{t-1} \\ s_c^t &= \beta_{forget}^t s_c^{t-1} + \beta_{input}^t g(\alpha_c^t) \end{aligned} \tag{3.1.3}$$

Output Gates

$$\begin{aligned}\alpha_{output}^t &= \sum_{i=1}^{I} w_{i,output} x_i^t + \sum_{h=1}^{H} w_{h,output} \beta_h^{t-1} + \sum_{c=1}^{C} w_{c,output} s_c^t \\ \beta_{output}^t &= f(\alpha_{output}^t)\end{aligned} \tag{3.1.4}$$

Cell Outputs

$$\beta_c^t = \beta_{output}^t h(s_c^t) \tag{3.1.5}$$

The gradient calculation during backward pass at each gate is shown as below [17]:

$$\epsilon_c^t \stackrel{\text{def}}{=} \frac{\partial O}{\partial \beta_c^t} \quad \epsilon_s^t \stackrel{\text{def}}{=} \frac{\partial O}{\partial s_c^t} \tag{3.1.6}$$

Cell Gates

$$\epsilon_c^t = \sum_{k=1}^{K} w_{ck} \delta_k^t + \sum_{h=1}^{H} w_{ch} \delta_h^{t+1} \tag{3.1.7}$$

Output Gates

$$\delta_{output}^t = \dot{f}(\alpha_{output}^t) \sum_{c=1}^{C} h(s_c^t) \epsilon_c^t \tag{3.1.8}$$

State

$$\begin{aligned}\epsilon_s^t =& \beta_{output}^t \dot{h}(s_c^t) \epsilon_c^t + \beta_{forget}^{t+1} \epsilon_s^{t+1} + w_{c,input} \delta_{input}^{t+1} \\ &+ w_{c,forget} \delta_{forget}^{t+1} + w_{c,output} \delta_{output}^{t+1}\end{aligned} \tag{3.1.9}$$

Cells

$$\delta_c^t = \beta_{input}^t \dot{g}(\alpha_c^t) \epsilon_s^t \tag{3.1.10}$$

Forget Gate

$$\delta_{forget}^{t} = \dot{f}(\alpha_{forget}^{t}) \sum_{c=1}^{C} s_{c}^{t-1} \epsilon_{s}^{t} \tag{3.1.11}$$

Input Gates

$$\delta_{input}^{t} = \dot{f}(\alpha_{input}^{t}) \sum_{c=1}^{C} g(\alpha_{c}^{t}) \epsilon_{s}^{t} \tag{3.1.12}$$

3.2 Mixture Density Network(MDN)

Mixture Density Network (MDN) is an artificial neural network suitable for learning continuous variables [21]. By fitting the Gaussian probability distribution mixture to the training data by learning the MDN parameter, the output value y is expressed as a probability distribution with respect to the input value x. The draft concept of MDN is shown as Figure 3.3

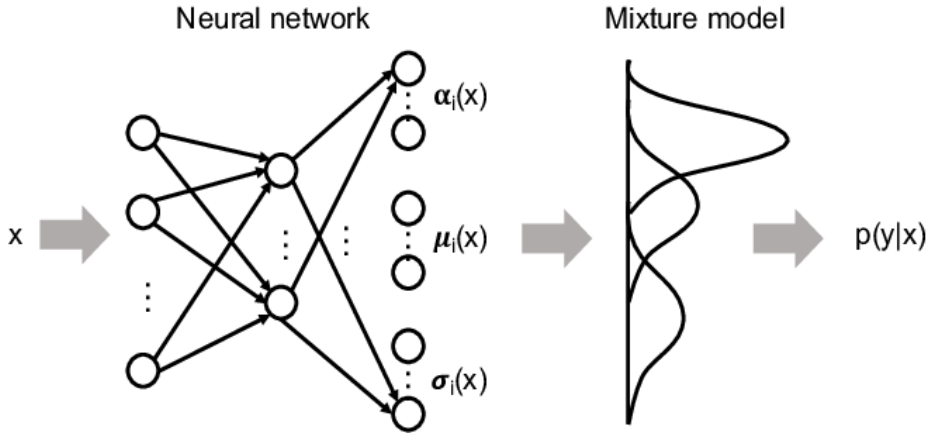


Figure 3.3: The concept of MDN, repriented form [22].

The MDN parameters are:

$$\alpha_i(x): \text{ mixing coefficients of Gaussian mixtures}$$
$$\mu_i(x): \text{ center of } i^{th} \text{ Gaussian mixture}$$
$$\sigma_i(x): \text{ variance of } i^{th} \text{ Gaussian mixture}$$

$\alpha_i(x)$ decides the weight of the Gaussian mixtures. $\mu_i(x)$ shows the center value of the i^th Gaussian mixture. Lastly, $\sigma_i(x)$ determines the variance of the i^th Gaussian mixture. By learning MDN parameters, we can train MDN artificial neural network. The probability distribution of output y with respect to the input x as follows [21]:

$$\begin{aligned} p(y|x) &= \sum_{i=1}^{m} \alpha_i(x)\phi_i(y|x) \\ \phi_i(y|x) &= \frac{1}{(2\pi)^{\frac{c}{2}}\sigma_i^2(x)} \exp\left(-\frac{\|y-\mu_i(x)\|^2}{2\sigma_i^2(x)}\right) \end{aligned} \tag{3.2.13}$$

By using MDN, we can express multiple output y on a single input x through the Gaussian Mixture Model. The comparison between MDN and normal L2 loss function can be seen as Figure 3.4, 3.5. Here, black dots are training data. Red and blue dots mean the prediction from L2 loss function and MDN. From Figure 3.4, 3.5, MDN predicts significantly better than normal L2 loss function.

3.3 LSTM-MDN

By combining LSTM and MDN, the dimension of the result value of LSTM is $K(c+2)$, as follows.

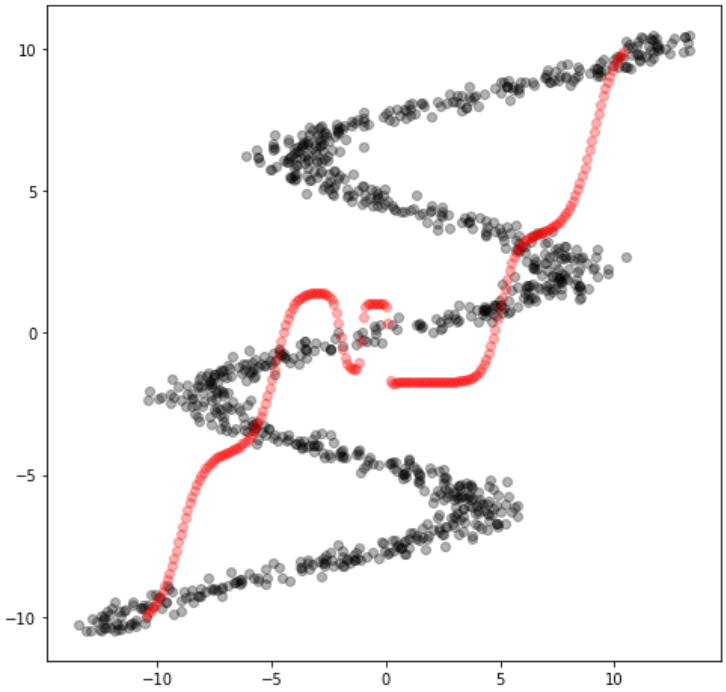


Figure 3.4: The result of L2 loss function.

$$z_i^\alpha \in \mathbb{R},\ i = 1 \backsim K$$

$$z_i^\sigma \in \mathbb{R},\ i = 1 \backsim K$$

$$z_{ij}^\mu \in \mathbb{R},\ i = 1 \backsim K,\ j = 1 \backsim c$$

Using above, we can express the MDN parameter as below.

$$\alpha_i = \frac{\exp(z_i^\alpha)}{\sum_{j=1}^{K} \exp(z_j^\alpha)} \in \mathbb{R},\ i = 1 \backsim K$$

$$\sigma_i = \exp(z_i^\sigma) \in \mathbb{R},\ i = 1 \backsim K$$

$$\mu_i = [z_{i_1}^\mu, z_{i_1}^\mu, \cdots, z_{i_c}^\mu] \in \mathbb{R}^c,\ i = 1 \backsim K$$

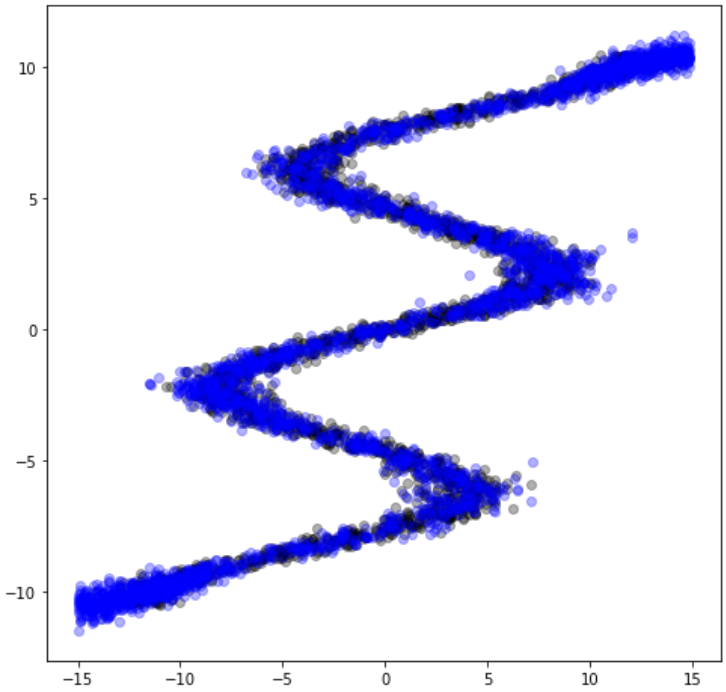


Figure 3.5: The result of MDN.

We use the softmax of z_i^{α} in the result of LSTM as a mixing coefficient of MDN and use the exponential of z_i^{σ} as the Gaussian mixture variance of MDN. For the Gaussian mixture center of MDN, z_{ij}^{μ} is used. The LSTM is learned by learning the LSTM-MDN, and as a result, the MDN parameter is learned. The probability distribution of state x_{t+1} for state x_t is [16].

$$\begin{aligned} p(x_{t+1}|x_t) &= \sum_{i=1}^{K} \alpha_i(x_t) N(x_{t+1}|\mu_i(x_t), \sigma_i^2(x_t)) \\ N(x_{t+1}|\mu_i(x_t), \sigma_i^2(x_t)) &= \frac{1}{(2\pi)^{\frac{c}{2}} \sigma_i^2(x_t)} \exp\left(-\frac{\|x_{t+1} - \mu_i(x_t)\|^2}{2\sigma_i^2(x_t)}\right) \end{aligned} \tag{3.3.14}$$

This can be used to define the learning error equation of the artificial neural network. When the number of learning data is n and $x_i^q, i = 0 \backsim t_f$ is defined as the q-th learning data, the learning error Negative Log-Likelihood Loss (NLL Loss) can be defined as.

$$\begin{aligned} E_{NLL} &= \sum_{q=1}^{n} E^q \\ E^q &= -\log\{\sum_{i=1}^{K} \alpha_i(x_t^q) N(x_{t+1}^q | \mu_i(x_t^q), \sigma_i^2(x_t^q))\} \end{aligned} \tag{3.3.15}$$

It should be noted here that the value contained in the NLL learning error is zero at the beginning of learning, so that the learning error may not be calculated correctly. Therefore, we should set max to ϵ value close to 0 before taking the log.

Our training model architecture and process is shown as Figure 3.6.

After training is completed, the network generates a predicted motion trajectory by using $p(x_{t+1}|x_t)$. We need to choose proper Gaussian mixture i with $\arg\max(\alpha_i(x_t))$. Then, we estimate state x_{t+1} at time $t+1$ through the selected Gaussian mixture.

The prediction model architecture and process is as Figure 3.7.

The network consists of 3 hidden layers with 256 nodes for each layer. It is unrolled for 32-time steps. All parameters were commonly run from -0.08 to 0.08 [16], and the mini-batch size was set to 50, using the stochastic gradient descent method. Adam Optimizer was applied with a learning rate of 0.001 and a decay rate of 0.99. In order to solve the exploding gradient problem, we adopt a gradient clipping technique. Lastly, 80% of the data is used for training and the remaining is used for validation [5].

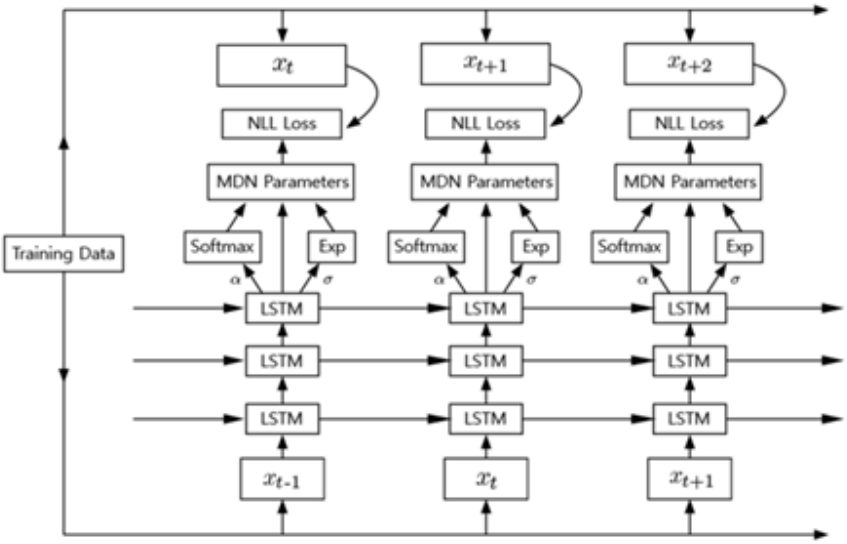


Figure 3.6: LSTM-MDN training model architecture.

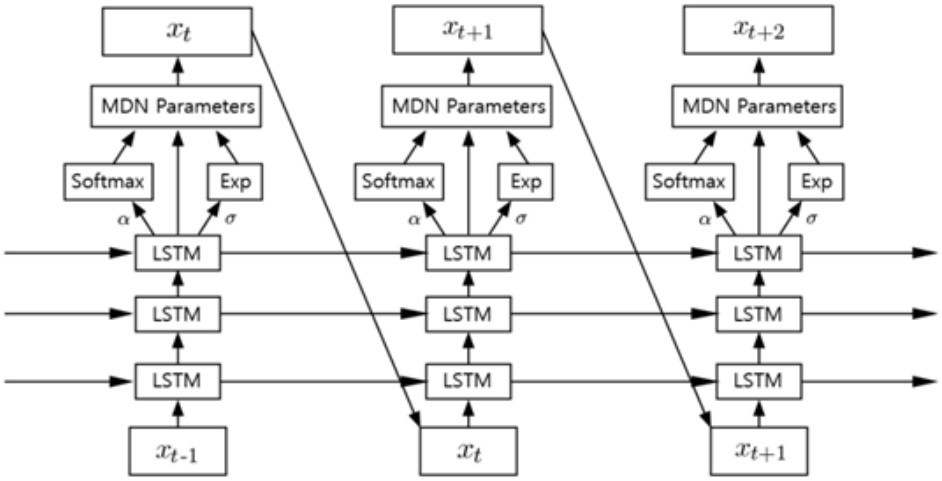


Figure 3.7: LSTM-MDN prediction model architecture.

3.4 Pre-Process of Target Data

Appropriate quantity and quality data play an important role in the field of machine learning. This section describes the most important part of learning, the process of generating and preprocessing of learning data. We use the optimization trajectory of target motion above.

$$q(t) \in \mathbb{R}^9 \text{or} \mathbb{R}^7 : \text{ Joint trajectory of both arms}$$
$$m \in \mathbb{R} : \text{ Characteristics of target object(mass or length)}$$

Here, the measurement of motion data is carried out in SrLib which is C++ based simulator from SNU Robotics Laboratory.

3.4.1 Data Preprocessing

Before learning the LSTM-MDN, it is necessary to modify existing motion trajectory data to suit learning. We include $q(t)$ to express the change of each joint according to the time. The difference between $q(t)$ and term $q(t_f)$ for time is used to express the final joint path $q(t_f)$, which is the constraint of motion.

Furthermore, in order to reflect the characteristics of the target object, the characteristic term m is multiplied by $q(t)$ and reflected. Taken together, one modified motion data $[x_0, x_1, \cdots, x_{t_f}]$ is shown as follows.

$$[x_0, x_1, \cdots, x_{t_f}] = \begin{bmatrix} q(0) - q(t_f) & q(1) - q(t_f) & \cdots & 0 \\ q(0) & q(1) & \cdots & q(t_f) \\ m \times q(0) & m \times q(1) & \cdots & m \times q(t_f) \end{bmatrix} \in \mathbb{R}^{27 \times t_f} \text{ or } \mathbb{R}^{21 \times t_f}$$

Normalize the above deformed training data to mean 0, variance 1. The final $q(t)$ is calculated by denormalizing the motion path generated through the LSTM-MDN. When the motion data is X the average is $E(X)$, and the variance is $\sigma(X)$, the normalization and denormalization processes are as follows.

$$\begin{aligned} &\text{Normalization} \ : \ Z = \frac{X - E(X)}{\sigma(X)} \\ &\text{Denormalization} \ : \ X = \sigma(X)Z + E(X) \end{aligned} \tag{3.4.16}$$

3.4.2 Frequency Sampling

Training an artificial neural network requires a significant amount of training data. By making sub-sample of original data, it is able to increase the number of training data [5]. Figure 3.8 shows the general concept of frequency sampling.

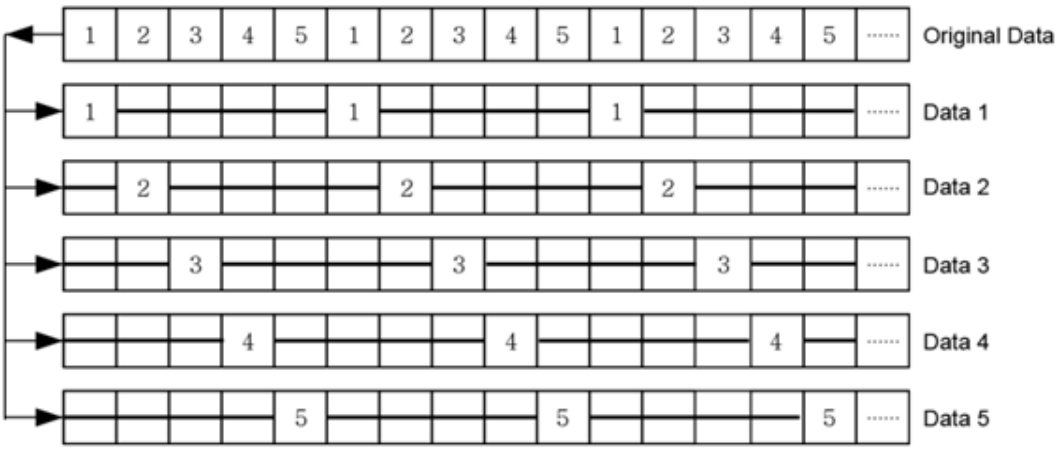


Figure 3.8: Example of frequency sampling.

We generate 220, 150 and 350 raw data for each target motion. Through frequency sampling explained above, we finally obtain 27,500, 18,750 and 43,750 training data.

3.5 Post-Process of Generated Data

After the training is completed, the trained model generates a trajectory up to t_f based on the final target motion $q(t_f)$ and object characteristics m at a given $x_0 \in \mathbb{R}^2 7$.

Therefore, it is necessary to examine the validity of the generated path through trajectory projection

The idea of trajectory projection started in Constrained Bi-directional Rapidly-exploring random trees(CBi-RRT) [23]. CBi-RRT is developed for problems that cannot use existing RRT due to constraints. In the joint space of the robot, the area satisfying these constraints forms a manifold. Since the constraint is generally given in the form of an implicit function $f(q) = 0$ for the joint value $q(t)$ of the robot, the overall manifold structure is unknown, but any point within the joint space can be projected onto the manifold using the Newton method.

In CBi-RRT, the next node is arbitrarily created and projected onto the manifold, but in our study, all nodes are already generated. However, what we need to do is make sure that the existed node is projected onto the manifold and if not project it onto the manifold.

To do this, we first need to create a manifold that satisfies constraint conditions. In this study, constraint conditions are set so that the relative positions of the left and right hands are constant in all paths.

$$SE_L^{-1}(3) \times (SE_R(3) \times SE_{LR}(3)) = I \tag{3.5.17}$$

The algorithm is written as Algorithm 3.

Algorithm 3 Trajectory Projection

initialization $q(t)$, C, T_c^o

for $t = 0$ **to** t_f **do**

$T_{obj}^o = (T_c^o)^{-1} T_{obj}^o$

$$d^c = \begin{bmatrix} t_{obj}^c \\ arctan2(R_{obj32}^c, R_{obj33}^c) - arcsin(R_{obj31}^c) \\ arctan2(R_{obj21}^c, R_{obj11}^c) \end{bmatrix}$$

$$\Delta x = \begin{cases} d^c - C_{max}, \text{ if} > d^c C_{max} \\ d^c - C_{min}, \quad \text{if} < d^c C_{min} \\ 0, \text{ otherwise} \end{cases}$$

if $\|\Delta x\| < \epsilon$ then return $q(t)$

$J \leftarrow$ GetJacobian(q(t)) $\Delta q_{error} \leftarrow J^T (J J^T)^{-1} \Delta x$

$q(t) \leftarrow (q(t) - \Delta q_{error})$

if $|q(t) - q_{old}(t)| >$ OutsideJointLimit$(q(t))$
then return NULL $q(t)$

end for

4

Results and Analysis

In this chapter, we confirm the results of the experimental architecture model described above and compare and analyze the difference between optimal motion trajectory and imitation motion trajectory.

Figure 4.1 shows the training loss for each motion. Cylinder lifting shows the highest loss followed by human rescuing and Single-arm grasping. This result is because the complexity of the movement affects the training outcome. The number of joints involved in the target motion differs from one movement to another. Furthermore, the range of movement is different for each joint, even if the same number of joints move. As a result, we observe that the fewer the number of joints involved in the motion, the less training loss we are able to observe.

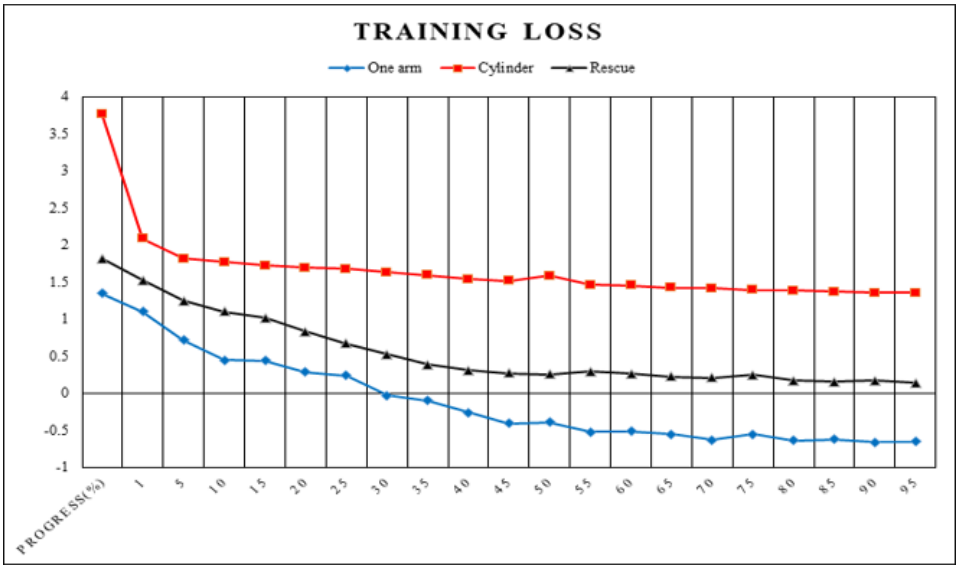


Figure 4.1: Training loss.

4.1 Result of LSTM-MDN Trajectory

4.1.1 Trajectory Comparison

Figure 4.2 shows the single-arm grasping target movement between the original optimal trajectory and imitation model generated trajectory by LSTM-MDN at the different target characteristics. Here, we set 3 different mass as target characteristics. Red lines on the figures mean original optimal trajectories and blue lines show the imitation model trajectory.

Figure 4.3 represents the results for the dual-arm cylinder lifting. As in the previous figure, we generate and compare the motion for three masses.

Lastly, we present the result of the dual-arm human rescuing in Figure 4.4. Experiments are carried out on three masses in the same manner as in the previous figure, but in this target motion, the trajectory was generated in addition to the

characteristics of the target person, the height.

It can be seen from the figure that all three motions generated in LSTM-MDN mimic the learned optimal trajectories. As a result, we can confirm that the preprocessing process and model change that we have made are well done apart from the training loss. However, we also observe that at the very first moment of prediction time the imitated trajectories are not following the ground truth.

This is caused by the nature of our model since our model contains many possibilities but lacks information to decide a suitable path at the beginning of prediction. This can be solved by reducing time step size to minimize a gap between steps or adopting additional smoothing process to ease the deviation.

4.1.2 Result of Trajectory Generation by Mass

Another point to consider is that an appropriate trajectory based on the weight of the object should be generated by LSTM-MDN. In other words, the optimal trajectory is generated differently depending on the characteristics of the object to be grasped or rescued. The trajectory generated through our model should not be significantly different from the original trajectory.

Figure 4.5 and 4.6 show show these results. As shown in the figure, it can be seen that it shows different trajectories according to the characteristics of the object, in this case, the mass.

As described above, the third target movement reflects the height of the person in the characteristics of the object, and the resulting path is also generated reflecting this. Therefore, if the characteristics of other objects are reflected in the learning data, this can be trained and imitated.

4.2 Result of Post-Processing

The path changes before and after post-treatment can be found in the Figure 4.7. As we predicted, our model can not generate perfect forecasting at the early stage. An incomplete prediction is due to the characteristics of the RNN that predict the value for the next time by applying back-propagation. At the beginning of the prediction, the time to perform back-propagation is small and therefore the value becomes incomplete.

As shown in the Figure 4.7, the initial machine learning-based trajectory shows some instability in the early stages. This is not fatal to the system, but it is not efficient. Therefore, we perform post-processing. As a result of the processing, the motion trajectory has become gentle and has no sudden movement.

Here, we intentionally put a huge perturbation in the LSTM-MDN trajectory to simulate a more dangerous situation.

As shown in the Figure 4.8 the change of the initial path is very rapid, which is very dangerous to the robot system in actual application and is practically inaccessible. After the post-processing, these sudden movements are projected according to the constraints of the robot, as a result, the robots can carry out the movements they can afford.

4.3 Result Analysis

Table 4.1, 4.2and 4.3 explains detailed comparison for three target motion. At each motion, we average generation time and the objective function improvement ratio. Here, generation time shows a time to generate an optimized trajectory. Objective function value is a sum of mechanical power that optimized path used. Lastly,

	Optimization			LSTM-MDN		
mass(kg)	1.0	20.0	45.0	1.0	20.0	45.0
Generation Time(ms)	6001	6044	6209	211	230	252
Objective Function Value(J)	1532.1	1936	2509.5	1634.9	1991.1	2393.1
Improvement Rate(%)	34.7	33.57	31.5	30.27	31.68	34.69

Table 4.1: Comparison of dual-arm single-arm grasping

improvement rate means a reduction of objective function value compared to the unoptimized trajectory.

In term of objective function value, LSTM-MDN based trajectory shows more than the optimized one. Thus, the improvement rate is also similar from 2 to 4%. However, we can observe significant improvement in trajectory generation time. LSTM-MDN costs only less than 200 ms which means almost instant time. Although artificial neural network based path uses few more energy, this disadvantage can be compensated by the fast generation time. Considering the working environment of the robot, the reduction of time is more important than the energy. Therefore, we see that our LSTM-MDN is able to achieve optimization in a short time with a relatively small loss of objective function value.

4.4 Generate Untrained Trajectory

Related to our study, we perform an additional simulation to confirm the utility of the model. LSTM based model has seen great improvement. However, when it predicts the result from untrained initial data with perturbation, the performance

	Optimization			LSTM-MDN		
mass(kg)	1.0	15.0	30.0	1.0	15.0	30.0
Generation Time(ms)	6053	6306	6192	156	162	172
Objective Function Value(J)	1131.1	1732	2203.4	1028.9	1537.8	1938
Improvement Rate(%)	33.2	32.1	35.7	30.2	28.5	31.4

Table 4.2: Comparison of dual-arm cylinder lifting

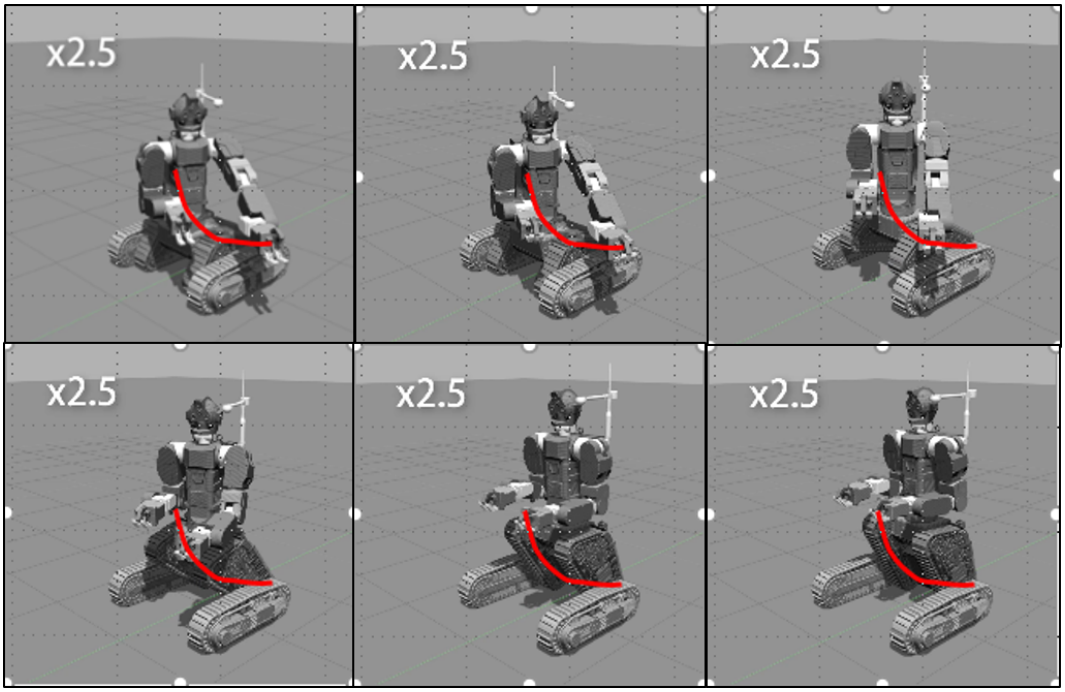
	Optimization			LSTM-MDN		
mass(kg)	80.0	100.0	120.0	80.0	100.0	120.0
Generation Time(ms)	6168	6192	6354	187	165	187
Objective Function Value(J)	3429.4	3921.4	4294.6	3587.1	4021.6	4471.4
Improvement Rate(%)	28.9	29.2	29.5	25.7	26.4	26.7

Table 4.3: Comparison of dual-arm human rescuing

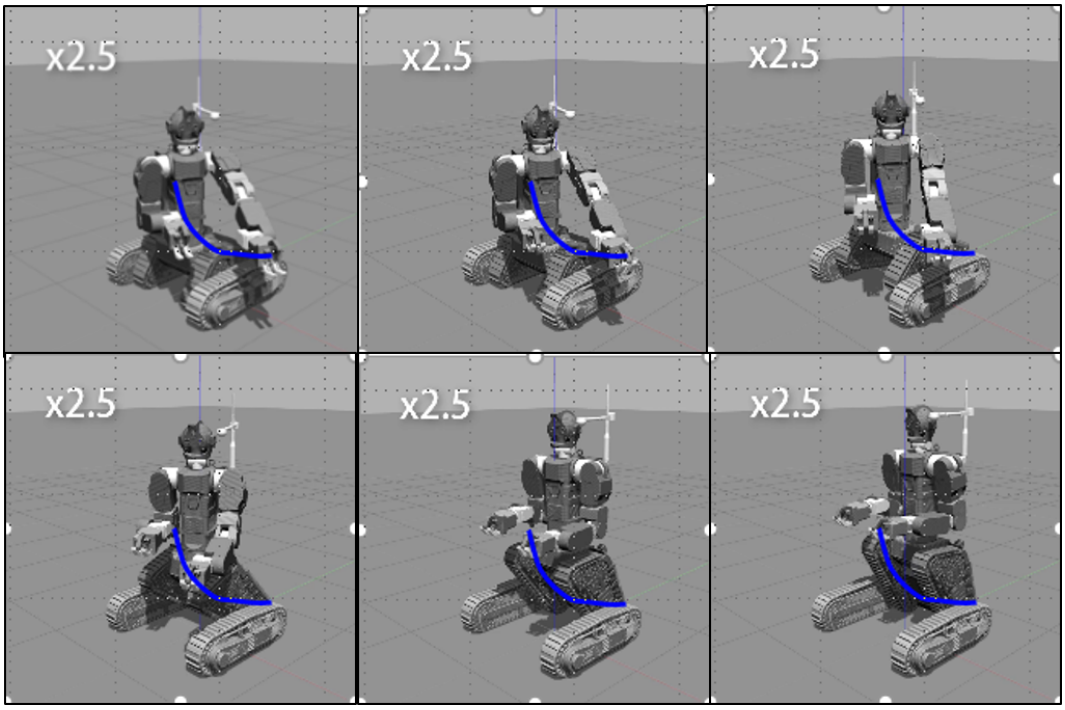
	Optimization			LSTM-MDN		
mass(kg)	7.5	22.5	42.5	7.5	22.5	42.5
Generation Time(ms)	6303	6044	6196	219	235	248
Objective Function Value(J)	1663.3	1889.3	2074.5	1689.8	1970.2	2140
Improvement Rate(%)	34.5	36.8	42.2	33.46	34.1	40.37

Table 4.4: Comparison of untrained trajectory

is decreased [24]. This is because LSTM is a part of supervised learning. Therefore, we also generate a robot trajectory with untrained initial values. Figure 4.9 shows the simulation result and Table 4.4 presents the detail comparison. From the result, our model can handle perturbation within our training data.

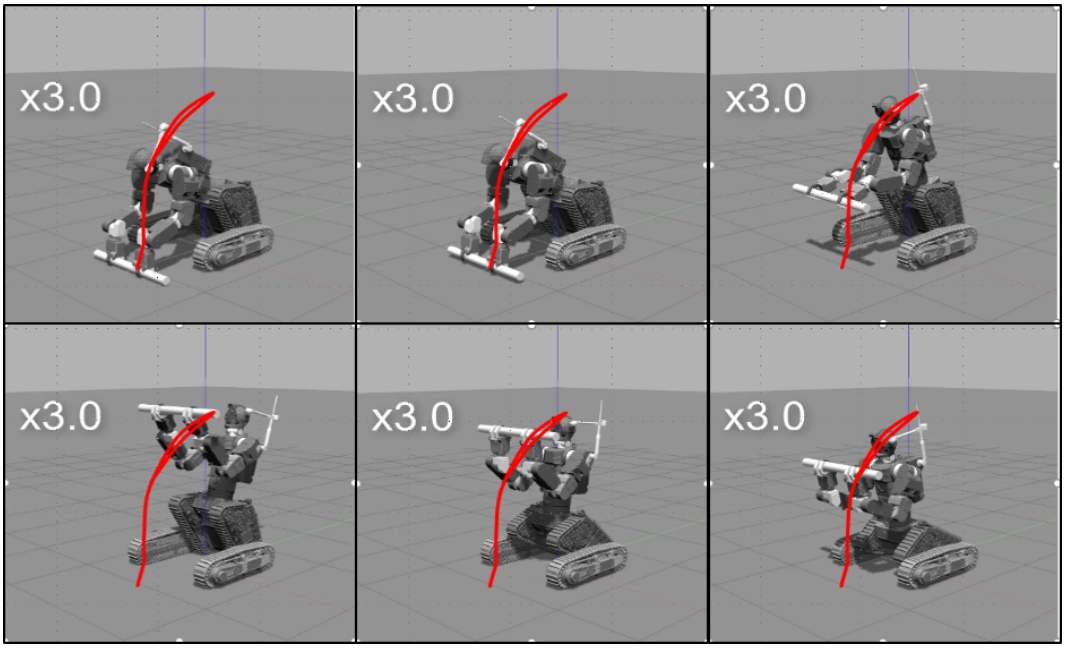


(a) Optimization

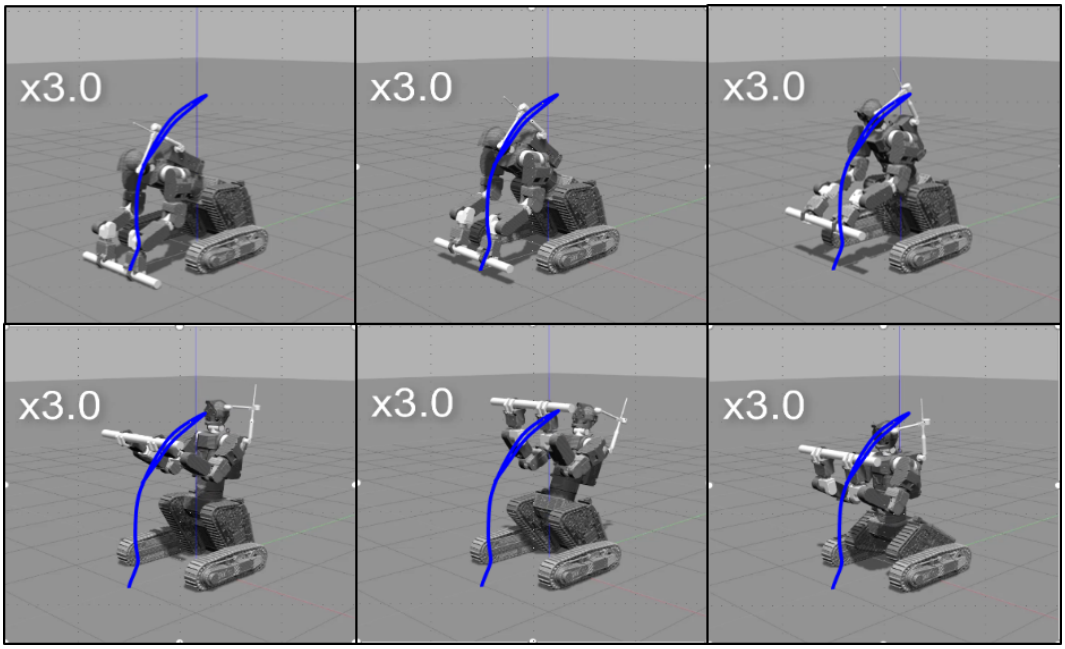


(b) Prediction

Figure 4.2: Result of single-arm grasping

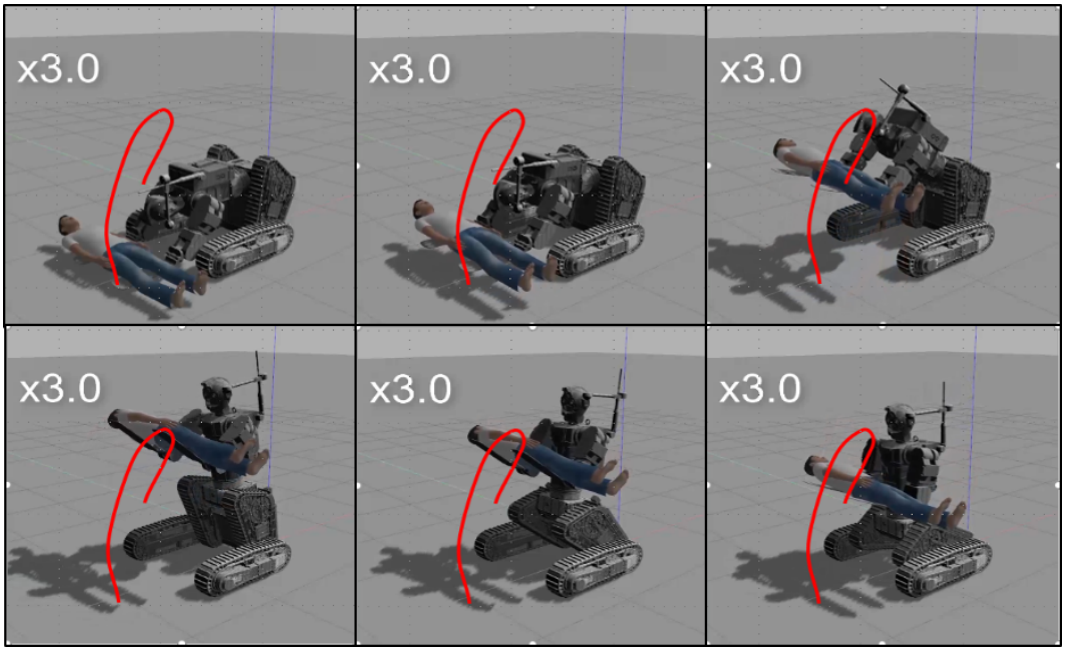


(a) Optimization

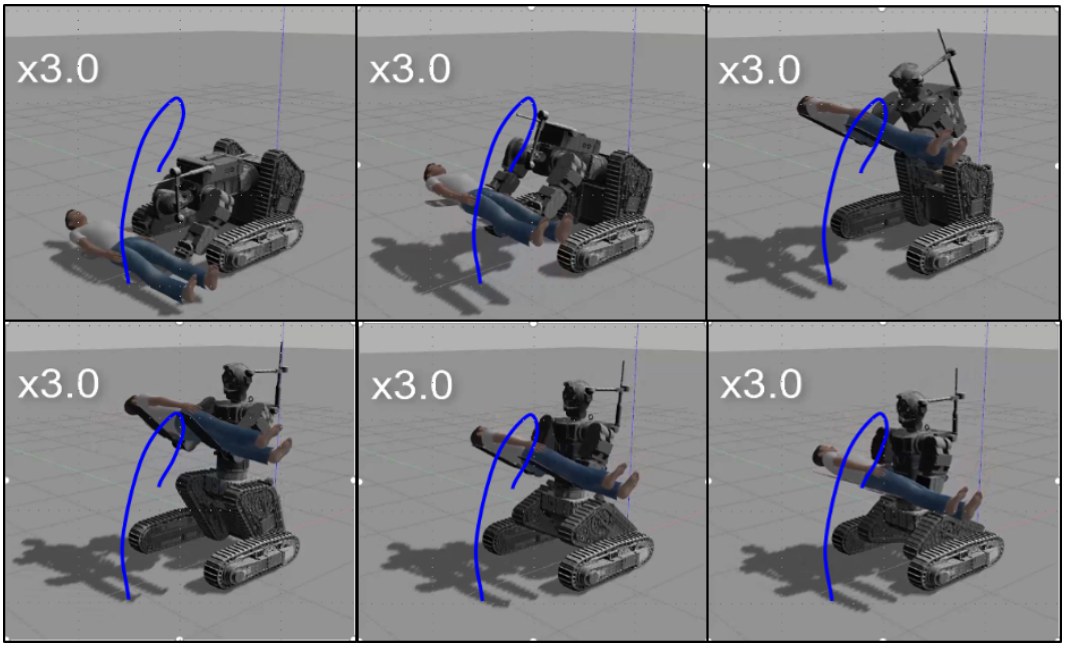


(b) Prediction

Figure 4.3: Result of dual-arm cylinder lifting

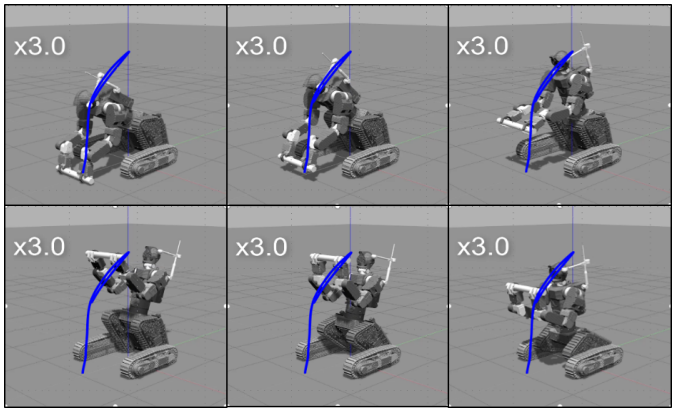


(a) Optimization

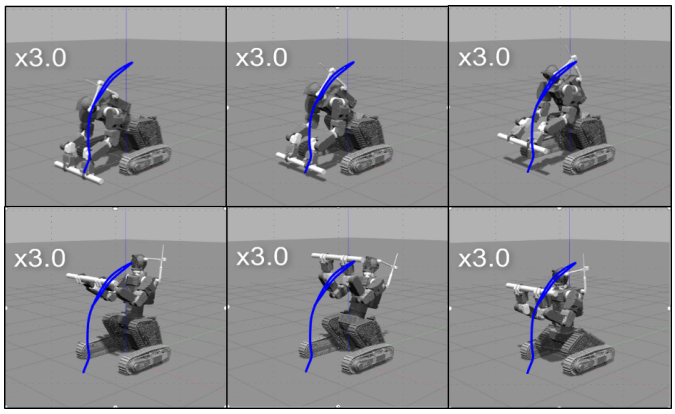


(b) Prediction

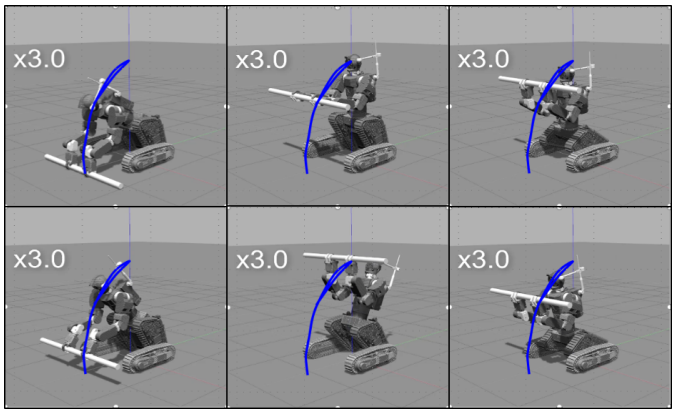
Figure 4.4: Result of human rescuing



(a) 1.0kg Lifting

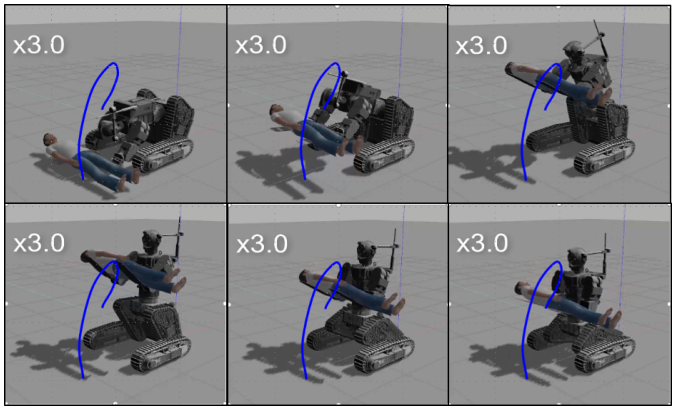


(b) 20.0kg Lifting

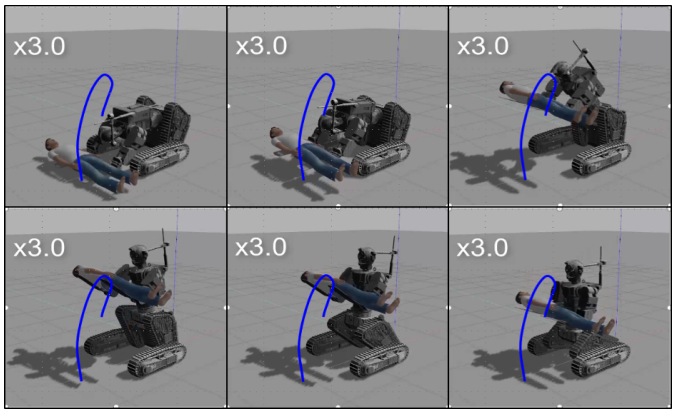


(c) 45.0kg Lifting

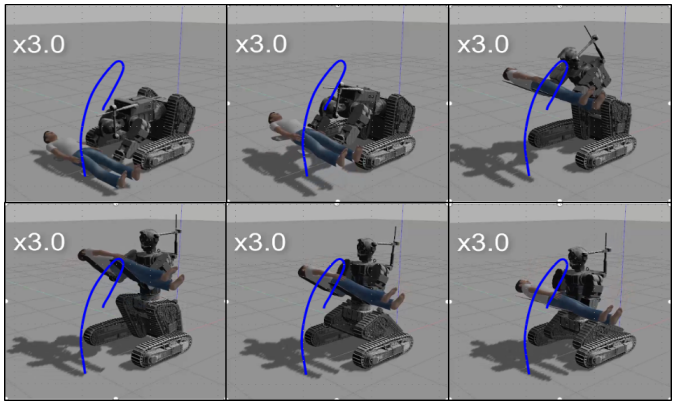
Figure 4.5: Dual-arm lifting trajectory by object weight



(a) 80.0kg Rescuing

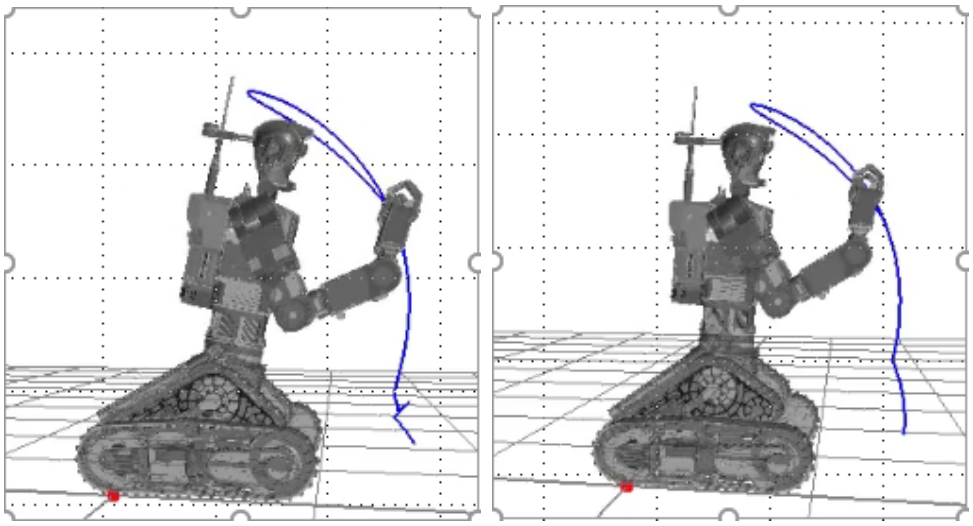


(b) 100.0kg Rescuing



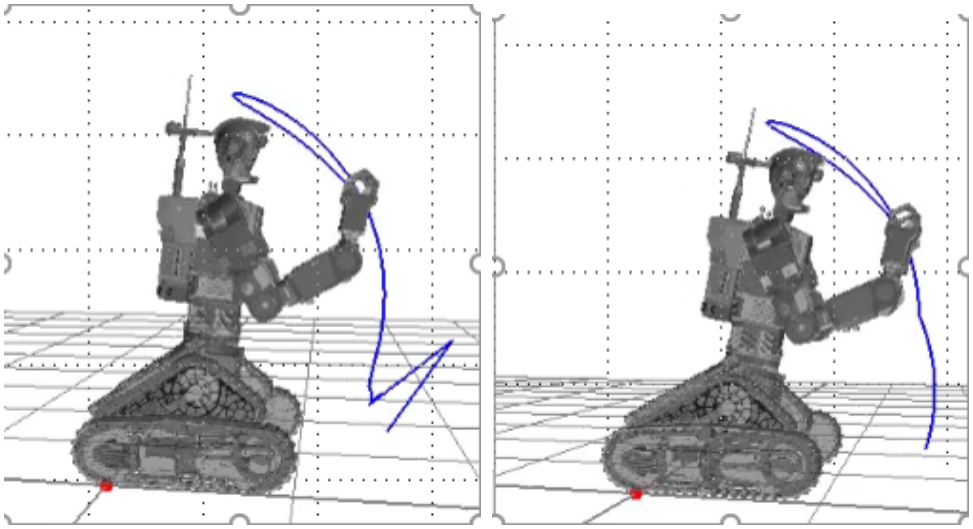
(c) 120.0kg Rescuing

Figure 4.6: Dual-arm rescuing trajectory by object weight



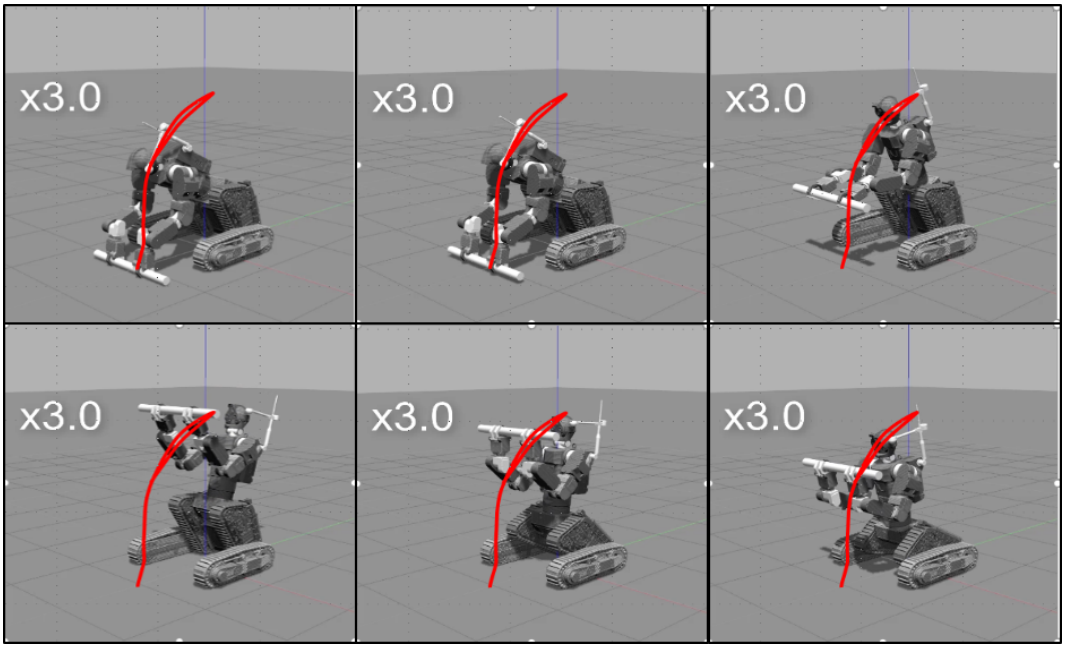
(a) LSTM-MDN prediction (b) After post-processing

Figure 4.7: Result of post-processing

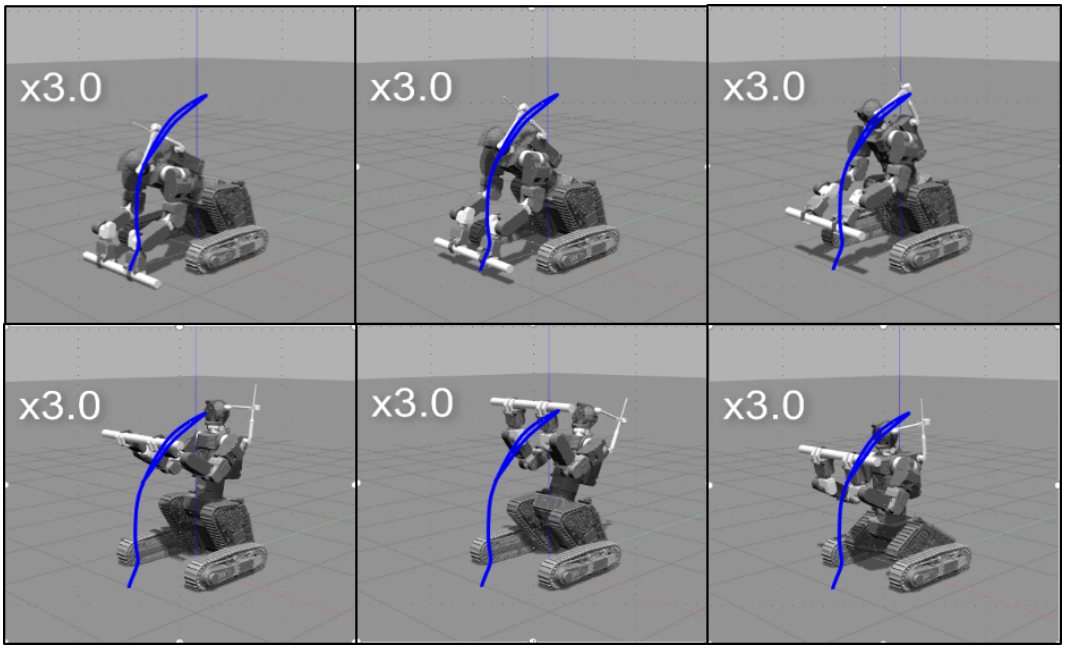


(a) Perturbed trajectory (b) After post-processing

Figure 4.8: Result of perturbation



(a) Optimization



(b) Prediction

Figure 4.9: Generation of untrained trajectory

5

Conclusion

In this thesis, we propose a motion trajectory generation method by using the LSTM-MDN artificial neural network to learn an optimized trajectory. Rescue robots are designed to be multi-jointed due to the nature of the operating environment, such as a disaster situation, and should be capable of performing heavy weight work under limited energy source conditions. Due to these constraints, optimization has been an important requirement for the robot motion generation. However, generating optimal motion in a high dimensional space requires considerable time and computing resources under any optimization algorithm. Many researchers has been working on combining artificial intelligence with robot control due to the development of machine learning.

Our approach is using LSTM-MDN. LSTM adopts gates to deal with exploding or vanishing gradient problem. Also, by predicting mixture weights over a Gaussian mixture model at the output with mean and variance for the mixture component, it is able to learn the diverse possible outcome.

However, LSTM-MDN has been used only relatively low-dimensional motions

such as one-arm pick and place. We modify the model to learn high-dimensional data. Then, we use joint angle data $q(t)$, motion execution time t_f to generate the trajectory of 21 degrees of freedom dual arm robot.

Three motion paths are generated in accordance with the purpose of rescue robot which are single-arm grasping, dual-arm cylinder lifting and dual-arm human rescuing. We preprocess original data and sub-sample them to increase the number of data for stable training. When training is done, we generate imitated trajectory only with initial position value.

We also applied the post - processing method to check whether the machine learning - based motion trajectory meets the physical limitations. We started our intuition form CBi-RRT. In the joint space of the robot, the area satisfying these constraints forms a manifold. Since the constraint is generally given in the form of an implicit function $f(q) = 0$ for the joint value $q(t)$ of the robot, the overall manifold structure is unknown, but any point within the joint space can be projected onto the manifold using the Newton method.

Simulation and comparison between the optimized and LSTM-MDN based trajectories have been carried out. The simulation itself shows only little difference. However, we observe that LSTM-MDN path need modification for improvement. We, on the other hand, observe a huge improvement at trajectory generation time while the loss of objective function values is less than 5% in comparison.

Considering the working environment of the robot, the reduction of time is more important than the energy. Therefore, it is confirmed that the efficient imitation trajectory can be generated by using the LSTM-MDN model. We confirm that LSTM-MDN generates different trajectories according to object characteristics, suggesting that not only weight or height but also other characteristics can be trained.

In addition, paths that are difficult to apply in some generated trajectories are modified to satisfy the constraints through the post-processing method described above. Finally, to check the utility of our model, we perform the additional experiment by generating untrained trajectory.

Bibliography

[1] Y. V. Bangalkar and S. M. Kharad. Review paper on search and rescue robot for victims of earthquake and natural calamities. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2015.

[2] J. E. Bobrow F. C. Park and S. R. Ploen. A lie group formulation of robot dynamics. *The International Journal of Robotics Research*, 14(6):609–618, 1995.

[3] F. C. Park M. Kim S.-H. Lee, J. Kim and J. E. Bobrow. Newtontype algorithms for dynamics-based robot movement optimization. *IEEE Transactions on robotics*, 21(4):657–667, 2005.

[4] Karen Simonyan* Ioannis Antonoglou David Silver*, Julian Schrittwieser*. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[5] Abolghasemi P. Behal A. Bölöni L. Rahmatizadeh, R. Learning real manipulation tasks from virtual demonstrations using lstm. *arXiv preprint arXiv:1603.03833v4*, 2016.

[6] Alexandre Attia and Sharone Dayan. Global overview of imitation learning. *arXiv preprint arXiv:1801.06503v1*, 2018.

[7] Geoffrey J. Gordon Stephane Ross and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *In Proc. of Int'l Conference on Artificial Intelligence and Statistics(AISTATS)*, 2011.

[8] Khanh Nguyen. Imitation learning with recurrent neural network. *arXiv preprint arXiv:1607.05241v1*, 2016.

[9] A Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2016.

[10] Gyuhyun Park Youngwoo Lee Jihong Min Seongil Hong Byunghun Choi, Wonsuk Lee. Development and control of a military rescue robot for casualty extraction task. *Journal of Field Robotics*, 2018.

[11] Park G. Choi, B. and Y Lee. Practical control of a rescue robot while maneuvering on uneven terrain. *Journal of Mechanical Science and Technology*, 32(5):2021–2028, 2018.

[12] Lofaro D. M. Bobick A. Egerstedt M. Oh P. Dantam, N. T. and M. Stilman. A new framework for real-time communicationn. *IEEE Robotics and Automation Magazine*, 22:76–85, 2015.

[13] Lee W. Choi B. Park G. Lee, Y. and Y. Park. Reliable software architecture design with ethercat for a rescue robot. *Proceedings of IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, 2016.

[14] B. J. Martin and J. E. Bobrow. Minimum-effort motions for openchain manipulators with task-dependent end-effector constraints. *The International Journal of Robotics Research*, 18(2):213–224, 1999.

[15] W. T. Vetterling W. H. Press, S. A. Teukolsky and B. P. Flannery. Numerical recipes:the art of scientific computing. *Cambridge University press*, 3, 2007.

[16] Vinyals O. Sutskever, I. and Q. V. Le. Sequence to sequence learning with neural networks. *In Advances in neural information processing systems(NIPS)*, 2014.

[17] Alex Graves. Supervised sequence labelling with recurrent neural networks. *University of Toronto*, 2012.

[18] P. Simard Y.Bengio and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[19] S.Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[20] A. Mohamed A. Graves and G. Hinton. Speech recognition with deep recurrent neural networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (3):6645–6649, 2013.

[21] C. M. Bishop. Mixture density networks. *Technical report*, 1994.

[22] Baptiste Feron Julian Vossen and A. Monti. Probabilistic forecasting of household electrical load using artificial neural networks. *IEEE Probabilistic Methods Applied to Power Systems (PMAPS)*, 2018.

[23] D. Ferguson D. Berenson, S. S. Srinivasa and J. J. Kuffner. Manipulation planning on constraint manifolds. *IEEE International Conference on Robotics and Automation*, 2009.

[24] Takahiro Nishida Takeshi. Inoue, Masaya Yamashita. Robot path planning by lstm network under changing environment. *Advances in Computer Communication and Computational Sciences*, pages 317–329, 2017.

국문초록

구난 로봇은 재난 상황과 같은 운영 환경의 특성으로 인해 다중 관절로 설계되었으며 제한된 에너지 원 하에서 고중량 작업을 수행 할 수 있어야한다. 이러한 제약 때문에 최적화는 로봇 동작 생성에 중요한 요구 사항이다. 그러나 고차원 공간에서 최적의 동작을 생성하는 것은 어떠한 최적화 알고리즘 하에서도 상당한 시간과 컴퓨팅 리소스가 필요하다. 본 연구에서는 모방 학습의 하나인 LSTM-MDN (Long-Short Term Memory-Mixture Density Network)을 통해 최적화된 경로를 학습하고 이를 모방하는 방법을 제안한다. 이 방법을 통해서, 다 관절 로봇의 최적화 경로 생성 시간을 단축할 수 있고, 학습 데이터가 적은 환경 하에서도 효율적인 학습을 수행할 수 있다. 이번 연구에서는 이를 Hurcules 구난 로봇의 다양한 동작에 적용하여 기존의 최적화 알고리즘에 대한 개선점을 보였다.

주요어: 모방학습, LSTM-MDN, 로봇 최적화, 구난 로봇.

학번: 2017-24254