Grasp Diffusion Network

Greif-Diffusionsnetzwerk
Master thesis by Qiao Sun
Date of submission: November 1, 2024

1. Review: Prof. Dr. Jan Peters
2. Review: M.Sc. Joao Carvalho
3. Review: M.Sc. An Thai Le

Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB der TU Darmstadt

Darmstadt, 1. November 2024

Q. Sun

Acknowledgments

I would like to express my sincere gratitude to my advisors, João Carvalho and An Thai Le, for their invaluable guidance, support, and mentorship throughout my research. Their expertise was instrumental in shaping this thesis, and I am deeply thankful for their continuous encouragement.

I am also profoundly grateful to my family and friends for their unwavering support and understanding. Their encouragement and companionship provided the strength and motivation I needed throughout this journey.

Additionally, I would like to acknowledge the Intelligent Autonomous Systems (IAS) for providing the opportunity, computational resources, and robotic equipment essential for the completion of this thesis.

Thank you all for making this journey possible.

Abstract

Grasp pose generation is crucial for robotic object manipulation because it directly affects a robot's ability to interact with diverse objects. However, vision-based grasping faces significant challenges such as limited viewpoints, complex grasp patterns, and insufficient datasets, which impede the development of robust systems.

To overcome these obstacles, this thesis explores the use of generative models to learn grasp distributions. Traditional generative models like GANs and VAEs have limitations in producing high-quality grasp samples due to architectural constraints. In contrast, diffusion models offer a promising alternative by effectively generating reliable grasp distributions.

This research proposes a Grasp Diffusion Network (GDN) that operates on the Lie group SO(3) and Euclidean space $\mathbb{R}^3$. Using partial point cloud features from a single camera, our method generates accurate grasp poses. We represent the rotational component of grasp poses on the Lie group SO(3), which naturally captures continuous rotational symmetries and avoids the singularities and ambiguities inherent in Euclidean space. Additionally, various strategies are employed to accelerate the sampling process and refine the generated grasps.

Our method is evaluated using the Isaac Gym simulator and a real-world Franka Emika Panda robot. Results demonstrate that our approach outperforms existing methods in terms of success rate, generation speed, and practical applicability, highlighting its potential as a valuable tool for robotic grasping tasks.

Zusammenfassung

Die Generierung von Greifposen ist entscheidend für die robotische Objektmanipulation, da sie die Fähigkeit des Roboters beeinflusst, mit unterschiedlichen Objekten zu interagieren. Allerdings stehen visionsbasierte Greifsysteme vor erheblichen Herausforderungen wie begrenzten Blickwinkeln, komplexen Greifmustern und unzureichenden Datensätzen, die die Entwicklung robuster Systeme behindern.

Um diese Hindernisse zu überwinden, untersucht diese Arbeit den Einsatz generativer Modelle zur Erlernung von Greifverteilungen. Traditionelle generative Modelle wie GANs und VAEs haben aufgrund architektonischer Einschränkungen Schwierigkeiten, qualitativ hochwertige Greifmuster zu erzeugen. Im Gegensatz dazu bieten Diffusionsmodelle eine vielversprechende Alternative, indem sie zuverlässig Greifverteilungen generieren.

Diese Forschung schlägt ein Grasp Diffusion Network (GDN) vor, das auf der Lie-Gruppe SO(3) und dem euklidischen Raum $\mathbb{R}^3$ operiert. Mithilfe von partiellen Punktwolkenmerkmalen einer einzelnen Kamera generiert unsere Methode präzise Greifposen. Wir repräsentieren die Rotationskomponente der Greifposen auf der Lie-Gruppe SO(3), was kontinuierliche Rotationssymmetrien natürlich erfasst und Singularitäten sowie Mehrdeutigkeiten im euklidischen Raum vermeidet. Zudem werden verschiedene Strategien eingesetzt, um den Sampling-Prozess zu beschleunigen und die generierten Griffe zu verfeinern.

Unsere Methode wird mithilfe des Isaac Gym Simulators und eines realen Franka Emika Panda-Roboters evaluiert. Die Ergebnisse zeigen, dass unser Ansatz bestehende Methoden hinsichtlich Erfolgsrate, Generierungsgeschwindigkeit und praktischer Anwendbarkeit übertrifft und sein Potenzial als wertvolles Werkzeug für robotische Greifaufgaben unterstreicht.

Figures and Tables

List of Figures

1.1. Structures of GAN, VAE, and Diffusion Models. GANs consist of a generator and discriminator working adversarially. VAEs include an encoder and decoder to transform data. Diffusion models uniquely add noise and iteratively denoise to generate samples. 2

2.1. Conversion between Lie Algebras and Lie Groups. 8

2.2. The U-Net based noise prediction network for DDPM. The input $\mathbf{x}_t$ and the time step t are processed through the U-Net to predict the noise ϵ. 11

2.3. Simplified Flowchart of DDIM 15

3.1. 6-DoF GraspNet generates grasps using a grasp sampler and iteratively refines negative grasps through gradients from the grasp evaluator. 18

3.2. SE(3)-DiffusionFields Architecture: The object point cloud and grasp pose are encoded through respective encoders to derive features, which are then processed by a decoder to calculate the energy e. 19

3.3. Grasp Latent Diffusion Model (GraspLDM): A model combining a point cloud encoder, a grasp decoder, and a latent diffusion module with a score network to enable direct sampling of grasp latents and task-conditioned generation. 22

4.1. Framework of the grasp diffusion network(GDN). 23

4.2. Forward process of the grasp diffusion network(GDN). 24

4.3. Reverse process of the grasp diffusion network(GDN). 25
4.4. Grasp generation process (from left to right). 25
4.5. Noise prediction network of the grasp diffusion network(GDN). 26
4.6. The Grasp Evaluator framework. 29
4.7. Grasp Diffusion Network(GDN) sampling process using DDIM for accelerated grasp pose generation. 33

5.1. Grasp testing in the Isaac Gym simulation environment (left) and the real world (right). 35
5.2. An example object from the ACRONYM dataset, its corresponding grasp, and the partial point cloud sampled from the object's mesh. 35
5.3. Performance comparison of grasp generation methods. EMD (x-axis) and Success Rate (y-axis) are shown for CAT10 objects, with our method GDN demonstrating superior mean performance and lower variance. 42
5.4. Comparison of grasp generation time between GDN (our method) and SE3Diffusion (baseline). The top plot shows the average generation time per iteration, and the bottom plot shows the cumulative grasp generation time. 44

A.1. Evaluating GDN in the real world: the Panda robotic arm successfully grasps a bowl using a single depth camera. 54
A.2. The GDN grasp generation (denoising) process: effective grasp distributions are generated for objects of various shapes and categories. 55

List of Tables

5.1. Comparison of Grasp Generation Methods in terms of EMD and Success Rate. All results are based on the test dataset. CAT10 includes 10 different object categories. 41

Abbreviations, Symbols, and Operators

List of Abbreviations

Abbreviation	Definition
GAN	Generative Adversarial Network
VAE	Variational Autoencoder
DDPM	Denoising Diffusion Probabilistic Models
DDIM	Denoising Diffusion Implicit Models
SDF	Signed Distance Function
EMD	Earth Mover's Distance

List of Symbols and Operators

Symbol/Operator	Definition
$\mathbb{R}^n$	Euclidean space
$SO(3)$	Lie group of rotations
$SE(3)$	Lie group of rotations and translations
$\mathfrak{so}(3)$	Lie algebra of $SO(3)$
$\mathfrak{se}(3)$	Lie algebra of $SE(3)$

Contents

1. Introduction **1**
1.1. Motivation 1
1.2. Contributions 4
1.3. Outline 4

2. Background **5**
2.1. Lie Groups and Lie Algebras 5
2.2. Diffusion Models 8

3. Related Works **17**
3.1. 6-DoF Grasp Synthesis 17
3.2. Grasp Diffusion Models 19

4. Grasp Diffusion Network **23**
4.1. Overview 23
4.2. Grasp Generation 24
4.3. Grasp Refinement 27
4.4. Sampling Process Acceleration 32

5. Experiments **34**
5.1. Setup 34
5.2. Capturing Point Cloud Features 36
5.3. Comparative Experiments 38
5.4. Training Implementation Details 38
5.5. Results 39

6. Conclusion **45**

7. Discussion **46**
7.1. Limitations 46
7.2. Future Work 46

A. Further Results **54**

B. Detailed Explanation of Diffusion Model Principles **56**
B.1. Derivation of DDPM in Euclidean Space 56
B.2. Derivation of DDPM on $SO(3)$ Space 57
B.3. Derivation of DDIM in Euclidean Space 58
B.4. Derivation of DDIM on $SO(3)$ Space 58
B.5. Pseudocode for DDPM and DDIM 58

1. Introduction

1.1. Motivation

1.1.1. Why Use Generative Models?

In vision-based grasping tasks, objects have countless possible grasps, making the optimal grasp distribution highly complex. Existing datasets cannot cover all optimal grasps for every object. Generative models, a key area of unsupervised learning, address this challenge by learning and generating these complex distributions. They are designed to create new samples resembling the training data. During training, their primary task is density estimation—mapping probability distributions to approximate the unobservable data distribution. Once trained, these models can generate new samples from the distribution, providing multiple outputs from a single input and producing more samples that fit the data distribution.

1.1.2. Why Not Other Generative Model Methods?

Explicitly modeling the density distribution for grasping tasks and updating the model using gradient ascent and maximum likelihood is impractical due to the complexity and high dimensionality of grasp distributions. This approach assumes a specific distribution form, leading to costly parameter estimation and high computational demands. The rigid structure of explicit modeling also limits flexibility in handling diverse data.

Generative Adversarial Networks (GANs) [1] suffer from mode collapse, which reduces diversity in generated samples, and require extensive parameter tuning and experimentation. Variational Autoencoders (VAEs) [2], while offering probabilistic interpretation, often produce blurrier and lower-quality outputs compared to GANs and diffusion models.

The reconstruction loss in VAEs results in less detailed samples, insufficient for the high precision required in robotic grasping.

Diffusion models [3] overcome these issues by generating new samples through a process of adding noise and training the model to denoise it in reverse. This approach ensures stable training and avoids mode collapse. The gradual denoising allows for finer recovery of data details, producing more realistic and diverse samples. Additionally, it enables adjusting the direction of data generation at each step, allowing for higher quality or expectation-aligned distributions. For example, using gradient-guided grasp generation based on grasp success rate prediction networks or avoiding table collisions by considering the distance to the table. Therefore, we choose diffusion models to generate high-quality grasps that meet our requirements.

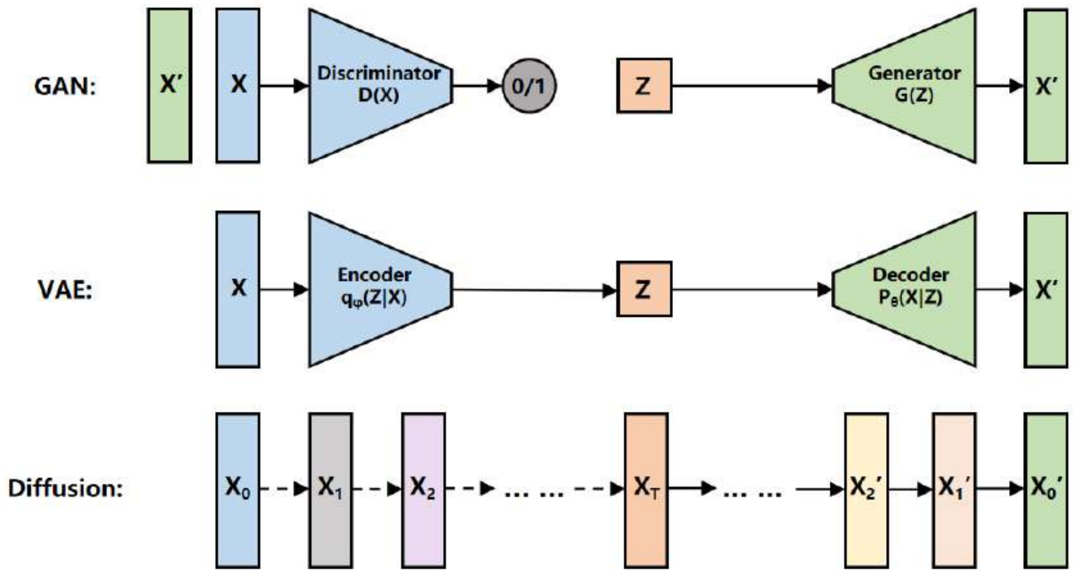

Figure 1.1.: Structures of GAN, VAE, and Diffusion Models. GANs consist of a generator and discriminator working adversarially. VAEs include an encoder and decoder to transform data. Diffusion models uniquely add noise and iteratively denoise to generate samples.

1.1.3. Why Represent Rotation on Lie Groups Instead of in Euclidean Space?

Accurate rotation representation is crucial in robotic grasping for generating and computing grasp poses. The Lie group $\mathrm{SO}(3)$ (Special Orthogonal Group) offers distinct advantages for 3D rotations, making it preferable over Euler angles or quaternions due to its mathematical properties and computational convenience.

Euler angles represent rotations using three angles (roll, pitch, and yaw) but suffer from gimbal lock—a singularity where degrees of freedom are lost at certain angles, causing instability. Quaternions, although free from gimbal lock, present optimization and integration challenges. With four components but only three degrees of freedom, quaternions can deviate from the unit sphere during optimization, leading to invalid rotations.

Using Lie groups for rotation representation avoids these issues. $\mathrm{SO}(3)$ has a well-defined differential structure, and its Lie algebra $\mathfrak{so}(3)$ naturally represents rotation differentials, facilitating optimization and control. The exponential and logarithmic mappings between Lie groups and Lie algebras allow for effective operations like interpolation. Therefore, we use Lie groups for rotation representation, as demonstrated in related works [4, 5, 6], which show their superiority over Euler angles and quaternions.

1.1.4. Why Use Partial Point Clouds Instead of Complete Point Clouds?

Obtaining and processing point cloud data is fundamental for pose generation in robotic grasping. While complete point clouds offer comprehensive information about the target object, acquiring them in practical applications is often unrealistic. Deploying multiple cameras increases costs and system complexity. Real-world robots may face spatial and perspective limitations, making it difficult to obtain complete point clouds—for example, the underside of objects on a table cannot be captured. Therefore, we aim to work with a single camera, enabling the model to generate grasps based on partial point clouds from a single viewpoint. This approach reduces deployment costs but increases implementation difficulty, as incomplete point clouds may lower success rates. Improving the success rate under these conditions is part of our work, making our method easily applicable in practice.

1.2. Contributions

This thesis makes the following key contributions:

Introduction of Grasp Diffusion Network (GDN): We present a novel Grasp Diffusion Network designed for vision-based grasping tasks. GDN generates high-quality grasps for target objects using partial point cloud data from random viewpoints.

Adherence to DDPM Principles: Our approach closely follows the principles and formulas of Denoising Diffusion Probabilistic Models (DDPM) [3]. By directly modeling and scaling the noise, our method outputs denoised vectors, enhancing grasp generation.

Leveraging DDIM Techniques: Using an architecture inspired by DDPM, our method incorporates techniques from Denoising Diffusion Implicit Models (DDIM) [7], significantly accelerating the sampling process.

Optimization of Generated Grasps: We refine the generated grasps using a grasp evaluation network and classifier guidance, improving overall performance and reliability.

1.3. Outline

The remainder of this thesis is structured as follows:

Chapter 2 explains relevant theories, providing a foundation for the subsequent content.

Chapter 3 discusses other related works, compares their advantages and disadvantages.

Chapter 4 details the Grasp Diffusion Network (GDN) method.

Chapter 5 describes the evaluation of our approach against other baselines.

Chapter 6 summarizes the key points and main contributions of this thesis.

Chapter 7 explores the limitations of our method and potential future work.

The appendix presents additional visual results and Diffusion Model details.

2. Background

This chapter provides the necessary background for the methods proposed in this thesis. In Section 2.1, we introduce the concepts of Lie Groups $\mathrm{SO}(3)$ and Lie Algebras $\mathfrak{so}(3)$, laying the mathematical foundation. Section 2.2 explores the original Denoising Diffusion Probabilistic Model (DDPM) in Euclidean space, its extension to Lie Groups $\mathrm{SO}(3)$, and other optimized diffusion model variants such as DDIM and classifier guidance.

2.1. Lie Groups and Lie Algebras

This section delves into the mathematical foundation necessary for implementing diffusion models on the Lie Group $\mathrm{SO}(3)$. We begin by introducing Lie Groups and Lie Algebras, focusing on $\mathrm{SO}(3)$ and its corresponding Lie algebra $\mathfrak{so}(3)$, which are essential for our proposed method. For a comprehensive explanation of these theories and their application in robot state estimation, the reader is referred to [8].

2.1.1. Lie Groups

To understand Lie Groups, we first review the concept of a group. A group is a set G equipped with an operation $*$ satisfying the following conditions:

- **Closure**: For all $X, Y \in G$, $X * Y \in G$.
- **Associativity**: $(X * Y) * Z = X * (Y * Z)$ for all $X, Y, Z \in G$.
- **Identity**: There exists an element $E \in G$ such that $E * X = X * E = X$ for all $X \in G$.

- **Invertibility**: For each $X \in G$, there exists an inverse element $X^{-1} \in G$ such that $X * X^{-1} = X^{-1} * X = E$.

$$\mathrm{SO}(3) = \left\{ R \in \mathbb{R}^{3\times3} \mid RR^\top = I, \det(R) = 1 \right\}$$

$$\mathrm{SE}(3) = \left\{ T = \begin{bmatrix} R & t \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \mid R \in \mathrm{SO}(3),\, t \in \mathbb{R}^3 \right\}$$

A Lie Group G is a smooth manifold that is also a group, where the group operations of multiplication and inversion are smooth functions. A smooth manifold is a space that locally resembles Euclidean space and allows for calculus operations. Each point on the manifold has a unique tangent space, a linear space where calculus can be performed. The Special Orthogonal Group $\mathrm{SO}(3)$ is a Lie Group consisting of rotation matrices, which are orthogonal matrices with determinant equal to one. Including both rotations and translations, we obtain the Special Euclidean Group $\mathrm{SE}(3)$. In these groups, the group operation is matrix multiplication.

2.1.2. Lie Algebras

$$\mathfrak{so}(3) = \left\{ \boldsymbol{\phi} \in \mathbb{R}^3 \mid \boldsymbol{\phi}^\wedge \in \mathbb{R}^{3\times3} \right\}, \quad \boldsymbol{\phi}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}$$

$$\mathfrak{se}(3) = \left\{ \boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix} \in \mathbb{R}^6 \mid \boldsymbol{\rho} \in \mathbb{R}^3,\, \boldsymbol{\phi} \in \mathfrak{so}(3),\, \boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \right\}$$

The tangent space of a Lie Group at the identity element is called the Lie algebra of that group. The Lie algebra $\mathfrak{so}(3)$ corresponding to the Special Orthogonal Group $\mathrm{SO}(3)$ consists of skew-symmetric matrices. Similarly, the Lie algebra $\mathfrak{se}(3)$ corresponding to the Special Euclidean Group $\mathrm{SE}(3)$ includes both skew-symmetric matrices and translation vectors.

2.1.3. Conversion Between Lie Algebras and Lie Groups

We now focus on the relationship between Lie Groups and Lie Algebras, particularly the conversion between them. For $\mathrm{SO}(3)$, a rotation matrix R evolves over time, and its corresponding Lie algebra element ϕ consists of skew-symmetric matrices. The exponential and logarithmic mappings facilitate the conversion between an element in $\mathrm{SO}(3)$ and its Lie algebra $\mathfrak{so}(3)$, allowing operations performed in the Lie algebra to be transformed back to the Lie group. The following derivation demonstrates this relationship:

Given $R(t) \in \mathrm{SO}(3)$,

$$R(t)R(t)^\top = I \quad \Rightarrow \quad \dot{R}(t)R(t)^\top + R(t)\dot{R}(t)^\top = 0 \quad \Rightarrow \quad \dot{R}(t)R(t)^\top = -\left(\dot{R}(t)R(t)^\top\right)^\top$$

Since $\dot{R}(t)R(t)^\top$ is skew-symmetric, it can be represented as:

$$\dot{R}(t)R(t)^\top = \phi(t)^\wedge \quad \Rightarrow \quad \dot{R}(t) = \phi(t)^\wedge R(t)$$

Using Taylor expansion near $t = t_0$:

$$R(t) \approx I + \dot{R}(t)(t - t_0)$$

This leads to:

$$\dot{R}(t) = \phi_0^\wedge R(t)$$

Solving the differential equation with $R(0) = I$:

$$R(t) = \exp\left(\phi_0^\wedge t\right)$$

This demonstrates the exponential map between the rotation matrix $R(t)$ and its corresponding Lie algebra element $\phi_0^\wedge$.

For the logarithmic map, given a rotation matrix R:

$$\phi^\wedge = \log(R)$$

Here, $\phi^\wedge$ is the skew-symmetric matrix that corresponds to the vector $\phi \in \mathbb{R}^3$.

In practical applications, such as computer graphics, robotics, and control systems, the vector representation of Lie algebras is more compact and computationally efficient. The conversion between vector and matrix forms is achieved using the Hat ($\wedge$) and Vee ($\vee$) operators. Our Grasp Diffusion Network (GDN), which we will introduce in the next chapter, utilizes this vector form.

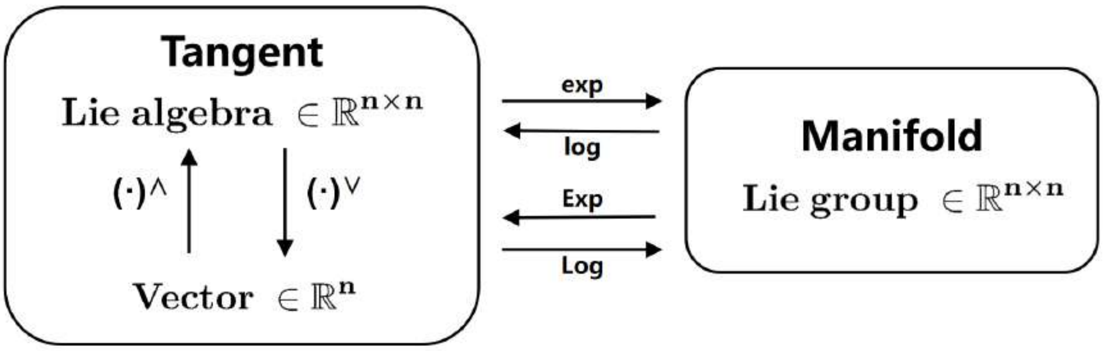
Tangent
Lie algebra $\in \mathbb{R}^{n\times n}$
$(\cdot)^\wedge$
$(\cdot)^\vee$
Vector $\in \mathbb{R}^n$
exp
log
Exp
Log
Manifold
Lie group $\in \mathbb{R}^{n\times n}$

Figure 2.1.: Conversion between Lie Algebras and Lie Groups.

To address the non-closure of addition in Lie groups, we use exponential and logarithmic mappings. By converting Lie group elements to Lie algebra elements, we can perform addition in the Lie algebra, then convert the result back to the Lie group. For example, combining two rotation matrices R_1 and R_2:

$$R_3 = R_2R_1 = \exp\left(\log(R_1) + \log(R_2)\right) = \exp\left(\mathbf{a}_1 + \mathbf{a}_2\right)$$

where $\mathbf{a}_i = \log(R_i)$. The Adjoint mapping is used to move vectors between tangent spaces before addition:

$$\varepsilon_\tau = \mathrm{Ad}_\chi\ {}^\chi\tau$$

This conversion relationship is crucial for interpolation (scaling from 0 to 1) in the linear Lie algebra space and subsequent conversion back to the Lie group, forming the basis of our core algorithm:

$$\lambda(\gamma, \mathbf{x}) = \exp\left(\gamma \log(\mathbf{x})\right)$$

2.2. Diffusion Models

In the previous chapter, we explored the rationale for selecting diffusion models over other generative models, such as GANs and VAEs, for generating grasping actions. This section delves into the principles of implementing diffusion models in both Euclidean space and on Lie groups, along with related optimization techniques.

2.2.1. Denoising Diffusion Probabilistic Models (DDPM)

Denoising Diffusion Probabilistic Models (DDPM) generate high-quality samples from complex distributions, showing remarkable results in audio synthesis and image applications. Inspired by non-equilibrium thermodynamics, DDPM defines a Markov diffusion chain that gradually adds random noise to the data, learning to reverse this process to reconstruct data samples from the noise. Unlike VAEs, diffusion models operate through a fixed process where latent variables retain the same high dimensionality as the original data. By mastering the denoising process, diffusion models achieve higher precision compared to GANs. As the number of samples and the duration of training increase, these models exhibit improved performance. The majority of current diffusion models are based on the work "Denoising Diffusion Probabilistic Models" [3] or its variants.

The diffusion model comprises two main processes: the forward process and the backward process.

Forward Process

In the forward process, Gaussian noise is continuously added to the original data $\mathbf{x}_0$ until it becomes random noise $\mathbf{x}_T$. This process is modeled as a one-step transition density of an inhomogeneous discrete-time Markov chain, where each state depends only on its immediate predecessor.

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}\right)$$
$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right)$$

where α_t $(t = 1, \ldots, T)$ represents a variance schedule, and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Given suitable conditions, the terminal value $\mathbf{x}_T$ is expected to follow a Gaussian distribution such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$.

The proportions of the original data and added noise are controlled by parameters α_t and $\beta_t = 1 - \alpha_t$. As time steps increase, the original data proportion decreases while the noise proportion increases, approximating Gaussian noise at large time steps. This allows for direct data restoration from Gaussian noise during the backward process.

A noise prediction network is trained to predict the noise added at each time step. The procedure involves sampling original data $\mathbf{x}_0$, selecting a random time step t, and adding noise from a normal distribution. The mean squared error (MSE) loss function between the actual and predicted noise is used to update the network. The equation $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ represents the data with added noise at time step t.

Algorithm 1 Training of DDPM

```
repeat
    Sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
    Sample $t \sim \text{Uniform}(\{1, \ldots, T\})$
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    Take gradient descent step on
        $$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left( \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t \right) \right\|^2$$
until converged;
```

Backward Process

The backward process reverses the forward process, removing noise step by step. By learning this reverse process, the model can generate data by transforming noise back into its original structure.

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\right)$$

Using the trained noise prediction network, the backward process begins by sampling from a Gaussian distribution to obtain $\mathbf{x}_T$. The process then reverses the time steps from T to 1, iteratively removing noise and reconstructing the data $\mathbf{x}_{t-1}$.

Algorithm 2 Sampling of DDPM

```
Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ do
    if $t > 1$ then
        Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    else
        $\boldsymbol{\epsilon} = \mathbf{0}$
    end
    Update $\mathbf{x}_{t-1}$ using
        $$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon}$$
end
return $\mathbf{x}_0$
```

Noise Prediction Network

DDPM employs the U-Net [9] architecture for encoding and decoding, enhanced with self-attention layers to improve global modeling capabilities. Each network layer includes embedding layers to predict noise at each time step t.

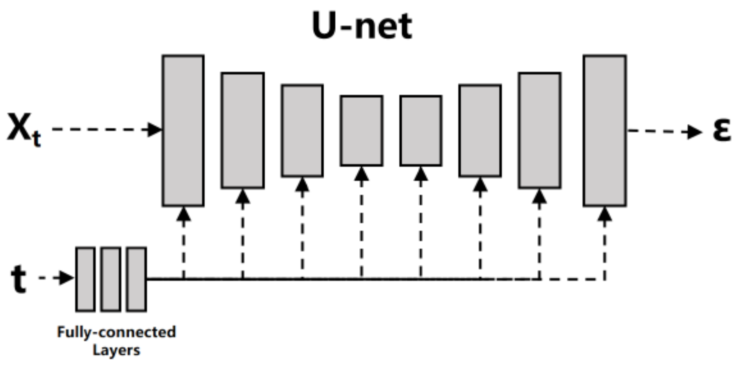

Figure 2.2.: The U-Net based noise prediction network for DDPM. The input $\mathbf{x}_t$ and the time step t are processed through the U-Net to predict the noise ϵ.

While we omit the complex derivations here, the probability density functions for the forward and backward processes are based on normal distributions in Euclidean space. Specifically, the convolution of two normal distributions yields another Gaussian distribution, presenting challenges for implementing the diffusion model on the Lie group $\mathrm{SO}(3)$. The subsequent section will address this issue.

$$\mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

Despite DDPM's advantages, it requires incremental noise additions, necessitating a large number of sampling steps and resulting in relatively slow generation speeds.

2.2.2. DDPM on $\mathrm{SO}(3)$

In the previous section, we discussed that the original DDPM method relies on the properties of Gaussian distributions in Euclidean space. An isotropic Gaussian distribution on $\mathrm{SO}(3)$ (IG) [10] exhibits similar properties, enabling the use of diffusion models on Lie groups [11].

$$\mathcal{IG}_{\mathrm{SO}(3)}(R_1, \epsilon_1) * \mathcal{IG}_{\mathrm{SO}(3)}(R_2, \epsilon_2) = \mathcal{IG}_{\mathrm{SO}(3)}(R_1 R_2, \epsilon_1 + \epsilon_2)$$

Forward Process

In the forward process, we incrementally add noise to the original data $\mathbf{x}_0$. Here, both the original data $\mathbf{x}_0$ and the noise represent rotations. Similar to DDPM, the forward process Markov chain $q(\mathbf{x}_t)$ depends solely on the previous state $q(\mathbf{x}_{t-1})$. A scaling factor ensures that as time progresses, the influence of the original data diminishes while the noise contribution increases.

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{IG}_{\mathrm{SO}(3)}\left(\lambda\left(\sqrt{\bar{\alpha}_t}, \mathbf{x}_0\right), 1 - \bar{\alpha}_t\right)$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ and $0 < \alpha_t < 1$.

Scaling is achieved through exponential and logarithmic mappings between Lie groups and Lie algebras, as discussed in Section 2.1. Combining two rotational matrices in Lie group form is done via matrix multiplication, and scaling is performed in the Lie algebra.

Algorithm 3 Training of DDPM on SO(3)

```
repeat
    Sample $t \sim \text{Uniform}(\{1, \dots, T\})$
    Sample $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
    Sample $R \sim \mathcal{IG}_{\text{SO}(3)}\left(\mathbf{I}, \sqrt{1-\bar{\alpha}_t}\right)$
    Compute $S(\mathbf{v}) = \frac{\log(R)}{\sqrt{1-\bar{\alpha}_t}}$
    Compute $\mathbf{x}_{\text{scale}} = \exp\left(\sqrt{\bar{\alpha}_t} \log \mathbf{x}_0\right)$
    Update $\mathbf{x}_t = R\mathbf{x}_{\text{scale}}$
    Take gradient descent step on
        $$\nabla_\theta \left\|\mathbf{v} - \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right\|^2$$
until converged;
```

Here, $S(\mathbf{v})$ denotes the skew-symmetric matrix in the Lie algebra $\mathfrak{so}(3)$ corresponding to vector $\mathbf{v}$.

Backward Process

The backward process mirrors the forward process, converting the noise back into the original data structure.

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{IG}_{\text{SO}(3)}\left(\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\right)$$

where

$$\tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0) = \lambda\left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}, \mathbf{x}_0\right) \lambda\left(\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}, \mathbf{x}_t\right)$$

The algorithm is as follows:

Algorithm 4 Sampling of DDPM on $\mathrm{SO}(3)$

```
Sample $\mathbf{x}_T \sim \mathcal{U}_{\mathrm{SO}(3)}$ for $t = T, \ldots, 1$ do
    if $t > 1$ then
        Sample $R \sim \mathcal{IG}_{\mathrm{SO}(3)}\left(\mathbf{I}, \tilde{\beta}_t\right)$
    else
        $R = \mathbf{I}$
    end
    Compute $\mathbf{v} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$
    Compute $a_1 = \exp\left(\frac{1}{\sqrt{\alpha_t}} \log(\mathbf{x}_t)\right)$
    Compute $a_2 = \exp\left(S\left(\frac{1}{\sqrt{\alpha_t}}\mathbf{v}\right)\right)$
    Compute $\tilde{\mathbf{x}}_0 = a_1 a_2^{-1}$
    Update $\mathbf{x}_{t-1} = \tilde{\nu}(\mathbf{x}_t, \tilde{\mathbf{x}}_0)R$
end
```

Although this method enables a diffusion model on $\mathrm{SO}(3)$, sampling from the isotropic Gaussian distribution (IG) on $\mathrm{SO}(3)$ involves integrating the cumulative distribution function (CDF), which affects the sampling speed.

2.2.3. Denoising Diffusion Implicit Models (DDIM)

In practical applications, especially for the real-time requirements of robotic operations, we desire a faster generation process. Diffusion models like DDPM require many sampling steps due to their principles, and the use of the cumulative distribution function (CDF) also causes each sampling step to take longer. Therefore, reducing the number of sampling steps is crucial for accelerating the generation process.

The denoising diffusion implicit models (DDIM) [7] method addresses this issue by breaking the Markov chain, allowing $\mathbf{x}_{t-1}$ to be represented using $\mathbf{x}_t$ and $\mathbf{x}_0$ while still meeting the conditions for reverse inference in DDPM. This approach significantly reduces the number of required sampling steps.

The update equation in DDIM is:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t) + \sigma_t \boldsymbol{\epsilon}$$

where σ_t controls the amount of noise added at each step. When $\sigma_t = 0$, the generation process becomes deterministic, and the model is referred to as DDIM.

Since DDIM does not require strict incremental Markov steps in the forward process, a shorter forward process can be defined by sampling a subsequence from the original Markov sequence.

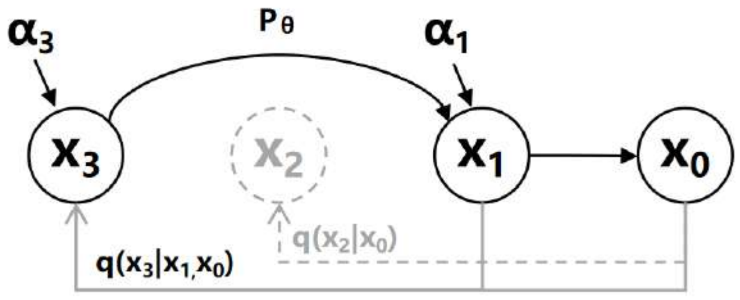

Figure 2.3.: Simplified Flowchart of DDIM

2.2.4. Classifier Guidance

Classifier Guidance [12] enables diffusion models to produce outputs targeted to specific classes. This concept was later expanded to Semantic Diffusion [13], allowing diffusion models to generate content conditioned on image, text, and multimodal inputs. For instance, stylization can be directed by using gradient guidance from both content and style.

Classifier Guidance involves adding the classifier's gradient to the process of conditional generation. By considering the score function and applying Bayes' theorem, the conditional generation probability can be decomposed:

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t \mid y) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(y \mid \mathbf{x}_t)$$

Here, $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$ represents the unconditional score, while $\nabla_{\mathbf{x}_t} \log p(y \mid \mathbf{x}_t)$ denotes the gradient from the classifier.

This approach requires training a classifier on noisy data and computing the classifier's gradient at every time step. With classifier guidance, the denoising process leverages the classifier's gradient, making the generation more responsive and adaptive. The classifier's feedback allows the generation process to adjust the denoising direction dynamically at different stages, effectively enhancing control over the output.

In our approach, we train a grasp evaluation network to estimate the likelihood of a successful grasp. The evaluation network's gradients guide the generation, improving efficiency and accuracy. As a result, the final generated grasp pose has a higher probability of success.

The diffusion algorithms incorporating classifier guidance for DDPM and DDIM are as follows:

Algorithm 5 Classifier-Guided Diffusion Sampling

```
Input: Diffusion model $(\boldsymbol{\mu}_\theta(\mathbf{x}_t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t))$, classifier $p_\phi(y \mid \mathbf{x}_t)$, scale $s$, class label $y$
Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ do
    Compute $\boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow \boldsymbol{\mu}_\theta(\mathbf{x}_t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t)$
    Update $\mathbf{x}_{t-1} \leftarrow$ Sample from $\mathcal{N}\left(\boldsymbol{\mu} + s\boldsymbol{\Sigma}\nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t),\ \boldsymbol{\Sigma}\right)$
end
return $\mathbf{x}_0$
```

Algorithm 6 Classifier-Guided DDIM Sampling

```
Input: Diffusion model $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t)$, classifier $p_\phi(y \mid \mathbf{x}_t)$, scale $s$, class label $y$
Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ do
    Compute $\hat{\boldsymbol{\epsilon}} \leftarrow \boldsymbol{\epsilon}_\theta(\mathbf{x}_t) - \sqrt{1 - \bar{\alpha}_t}\nabla_{\mathbf{x}_t} \log p_\phi(y \mid \mathbf{x}_t)$
    Update $\mathbf{x}_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\hat{\boldsymbol{\epsilon}}}{\sqrt{\alpha_t}}\right) + \sqrt{1 - \bar{\alpha}_{t-1}}\hat{\boldsymbol{\epsilon}}$
end
return $\mathbf{x}_0$
```

This chapter has discussed relevant diffusion model methods, which will be applied in Chapter 4 to our Grasp Diffusion Network (GDN).

3. Related Works

This chapter reviews related works and contrasts them with our proposed method GDN.

3.1. 6-DoF Grasp Synthesis

Data-driven approaches to dexterous grasp sampling commonly include methods based on prior work [14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27], reinforcement learning techniques [28, 29], grasp strategies learned from human demonstrations [30], and various grasp synthesis methods [31]. A significant limitation of these methods is the assumption of complete object observations, which is challenging in real-world scenarios. Additionally, the scarcity of grasp training data and the simulation-to-reality gap hinder their practical application.

Grasp synthesis involves finding stable grasp poses based on sensor data and varies with different gripper types. Generalizable six-degree-of-freedom (6-DoF) grasp synthesis is a current research focus [14] and is typically categorized into discriminative and generative methods. Discriminative methods use manual pose sampling and learn to distinguish successful grasps from unsuccessful ones using loss functions as metrics [32]. In contrast, as discussed in Chapter 1, generative methods [30, 33, 15] directly learn to generate grasp poses from observations [34, 35] through implicit sampling, allowing for more efficient direct sampling from learned models [34]. Some studies on 6-DoF grasping also incorporate auxiliary tasks like shape completion and 3D reconstruction to enhance grasp generation [36]. Unlike these methods, we propose a generative approach that models a continuous distribution of object-centric grasps conditioned on point clouds.

Our method is inspired by 6-DoF GraspNet [34], which generates diverse grasps for unknown objects using three main components: a grasp sampler, a grasp evaluator, and a grasp refinement module. Initially, a Variational Autoencoder (VAE) served as the grasp sampler to generate grasp poses. While the VAE captured a variety of grasp patterns, it often produced unsuccessful grasps in practice, necessitating an additional pose refinement stage. Subsequent work [37] extended this approach by incorporating a learned network for collision checking, yet the VAE still underperformed compared to discrete regression models [35].

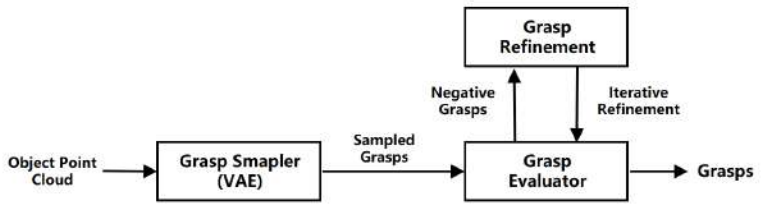

Figure 3.1.: 6-DoF GraspNet generates grasps using a grasp sampler and iteratively refines negative grasps through gradients from the grasp evaluator.

In contrast to [34], which utilizes a VAE for grasp sampling, we employ diffusion models for grasp generation. As detailed in Chapter 1, diffusion models offer advantages over other generative models.

Since the generator is trained only on successful grasps, it may inadvertently produce unsuccessful ones. To detect these negative grasps, a grasp evaluator is trained to predict the success probability $P(S \mid g, X)$ based on the grasp pose g and the point cloud X, using cross-entropy loss between true labels and predicted probabilities.

The findings in [34] indicate that sampled grasps with low evaluator scores are often near higher-scoring grasps in the grasp space, suggesting that local refinement can enhance success probability. This refinement uses the gradient of the predicted success probability from the evaluator to adjust the grasp pose, transforming near-successful negative grasps into successful ones. By iteratively refining and discarding grasps below a certain threshold, a set of high-quality grasps is obtained. Building on this insight, we incorporate the concept of local refinement [38] into our method.

For point cloud processing, they utilized PointNet++ [39], a neural network architecture designed for handling point sets in metric space. Since grasping requires only local features rather than full reconstruction, only the encoder part is employed. This approach offers advantages such as rotational invariance, improving the stability of grasp generation from partial point clouds. We integrate this method into our approach.

3.2. Grasp Diffusion Models

Building on the advantages of diffusion models over VAEs and GANs, this section compares several studies that employ diffusion models for grasp generation.

3.2.1. SE(3)-DiffusionFields

SE(3)-DiffusionFields [40] introduces an algorithm for learning data-driven cost functions in task space. It generates robot motion by optimizing a set of combined cost functions through joint gradient-based optimization. By employing diffusion learning, this method learns smooth cost functions that facilitate joint grasping and motion optimization.

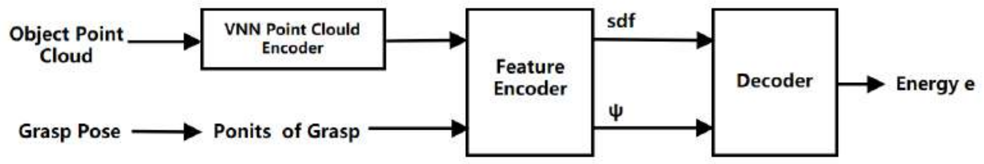

Figure 3.2.: SE(3)-DiffusionFields Architecture: The object point cloud and grasp pose are encoded through respective encoders to derive features, which are then processed by a decoder to calculate the energy e.

As illustrated in Figure 3.2, SE(3)-DiffusionFields relies on an energy model that maps object and grasp poses to energy values, effectively measuring grasp quality. In contrast, our method directly models the noise, outputting a denoised vector in the Lie algebra (i.e., a gradient in $\mathrm{SO}(3)$ space). This approach is more consistent with the standard operation of Denoising Diffusion Probabilistic Models (DDPM) and reduces computation time by avoiding backpropagation through energy-based models. It also eliminates the need to compute $\mathrm{SO}(3)$ gradients, thereby reducing computational complexity and enhancing stability.

Algorithm 7 Grasp SE(3)-DiF Training

```
Data: $\theta_0$: initial parameters for $z$, $F_0$, $D_0$; Datasets: $D_0 : \{m, H_u^0\}$ (object IDs and poses),
      $D_{\text{sdf}}^m : \{x, \text{sdf}\}$ (3D positions $x$ and SDF values for object $m$), $D_g^m : \{H\}$ (successful
      grasp poses for object $m$);
Result: Optimized parameters $\theta^*$;
for $s \leftarrow 0$ to $S-1$ do
    Sample $k, o_k \leftarrow [0, \ldots, L]$;
    Sample $m, H_u^0 \in D_0$;
    $z \leftarrow$ shape codes$(m)$;

    SDF Training
    Sample $x, \text{sdf} \in D_{\text{sdf}}^m$;
    $\text{sdf}_{\text{pred}} \leftarrow -F_0(H_u^0, x, z, k)$;
    $L_{\text{sdf}} \leftarrow C_{\text{mse}}(\text{sdf}_{\text{pred}}, \text{sdf})$;

    Grasp Diffusion Training
    Sample $H \in D_g^m$;
    $\epsilon \sim \mathcal{N}(0, \sigma_k I)$;
    $\hat{H} \leftarrow H\,\text{Expmap}(\epsilon)$;
    $x_0^n \leftarrow H\hat{x}$;
    $\text{sdf}_n, \phi_n \leftarrow F_0(x_0^n, z, k)$;
    $\psi \leftarrow \text{Flatten}(\text{sdf}_n, \phi_n)$;
    $e \leftarrow D_0(\psi)$;
    $L_{\text{dsm}} \leftarrow C_{\text{dsm}}(e, \hat{H}, H, \sigma_k)$;

    Parameter Update
    $L \leftarrow L_{\text{dsm}} + L_{\text{sdf}}$;
    $\theta_{s+1} \leftarrow \theta_s - \alpha \nabla L$;
end
return $\theta^*$;
```

As indicated by the algorithm step $\hat{H} \leftarrow H\,\text{Expmap}(\epsilon)$, SE(3)-DiffusionFields does not employ a scaling factor related to α_t to control the data-to-noise ratio when adding noise, unlike DDPM or DDPM on $\text{SO}(3)$. As a result, it cannot achieve the effect where the noised data asymptotically approaches pure noise as the time step approaches infinity.

SE(3)-DiffusionFields samples complete point clouds from objects and processes the features using a Vector Neuron Network (VNN) encoder. However, as discussed in Chapter 1, utilizing partial point clouds is more practical. For partial point clouds, methods like the VNN encoder are unsuitable because their encodings vary with object rotation, potentially leading to unstable grasp poses. Therefore, we adopt the rotationally invariant PointNet++ method to handle partial point clouds.

3.2.2. GraspLDM

GraspLDM [41] is a generative framework for object-centric six-degree-of-freedom (6-DoF) grasp synthesis using latent diffusion. This framework employs a diffusion model as a prior in the latent space of a VAE, learning a generative model of object-centric $\mathrm{SE}(3)$ grasp poses conditioned on point clouds. The lower quality of samples produced by VAE methods is often due to the prior gap problem, especially when the encoding distribution does not match the prior [38]. Unlike 6-DoF GraspNet [34], which relies on grasp refinement, GraspLDM bridges the gap between the VAE's prior and posterior distributions during training. It achieves this by using a Denoising Diffusion Model (DDM) in a low-dimensional latent space to learn the distribution of successful grasps on object point clouds, retraining a task-specific denoising network. Similar to our method, GraspLDM also utilizes diffusion models based on DDPM [3], employing Denoising Diffusion Implicit Models (DDIM) to accelerate the sampling process with minimal performance loss.

GraspLDM performs diffusion in the Euclidean latent space and then projects the diffusion back to $\mathrm{SO}(3)$ using Modified Rodrigues Parameters (MRP). In contrast, our method performs diffusion directly in the tangent space of rotations—the Lie algebra space. This approach better leverages the properties of diffusion models. For instance, we combine gradients to progressively guide the generation process because we aim to learn noise related to the gradient of the log-probability:

$$\epsilon_\theta(x_t) = -\sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t}\log p_\theta(x_t)$$

Unlike GraspLDM, we do not train a VAE by maximizing the Evidence Lower Bound (ELBO). Instead, we formulate the problem as diffusion in the pose space. Additionally, to prevent the strict ELBO objective from causing the Kullback-Leibler (KL) divergence term D_{KL} to become extremely small in the early stages of training, GraspLDM needs to use linear annealing of the λ parameter, which increases computational cost.

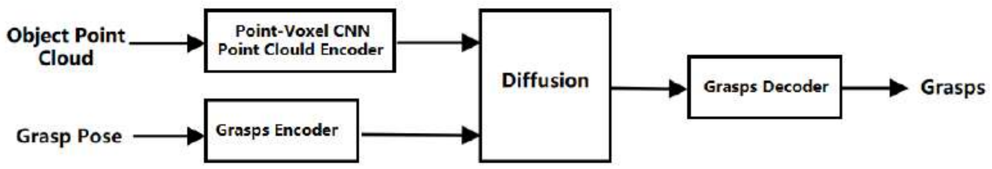

Figure 3.3.: Grasp Latent Diffusion Model (GraspLDM): A model combining a point cloud encoder, a grasp decoder, and a latent diffusion module with a score network to enable direct sampling of grasp latents and task-conditioned generation.

While other grasp generation methods yield good results, our approach offers advantages in practicality, stability, and success rate. In the next chapter, we will delve into the implementation details of our method.

4. Grasp Diffusion Network

In the previous chapters, we discussed our motivations, background, and related work. In this chapter, we provide a detailed explanation of our approach. Section 4.1 presents an overview of our method. Section 4.2 outlines the grasp generation process, and delves into the diffusion process and the architecture of our neural network. Section 4.3 describes the use of classifier guidance for refining the grasp generation process. Section 4.4 explains acceleration techniques of our method.

4.1. Overview

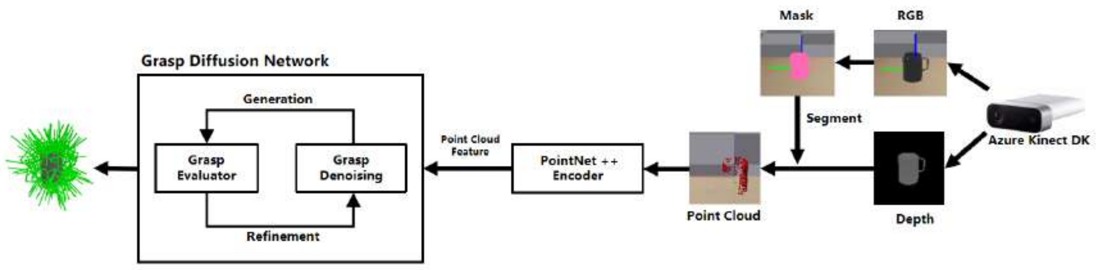

Figure 4.1.: Framework of the grasp diffusion network(GDN).

Our method's framework is illustrated in Figure 4.1. We begin by capturing RGB and depth images using the Azure Kinect DK depth camera. The RGB image undergoes segmentation to produce a mask that isolates the object of interest. Using this mask, we extract the object's depth information and convert it into a point cloud representation. This object point cloud is then encoded into point cloud features using the PointNet++ encoder [39], which serves as one of the conditional inputs for our grasp diffusion network.

In addition to grasp generation, we train a grasp evaluation network to estimate the success probability of each grasp. During the denoising process, we apply classifier guidance [12] at each timestep, using gradients from the evaluation network to steer the grasp generation towards higher success rates. This approach effectively refines the generated grasps. Finally, we sample from the learned distribution to determine the final grasp pose.

4.2. Grasp Generation

We utilize a diffusion model [3, 11, 7] to generate grasps from partial point clouds through a forward and reverse process.

4.2.1. Forward Process

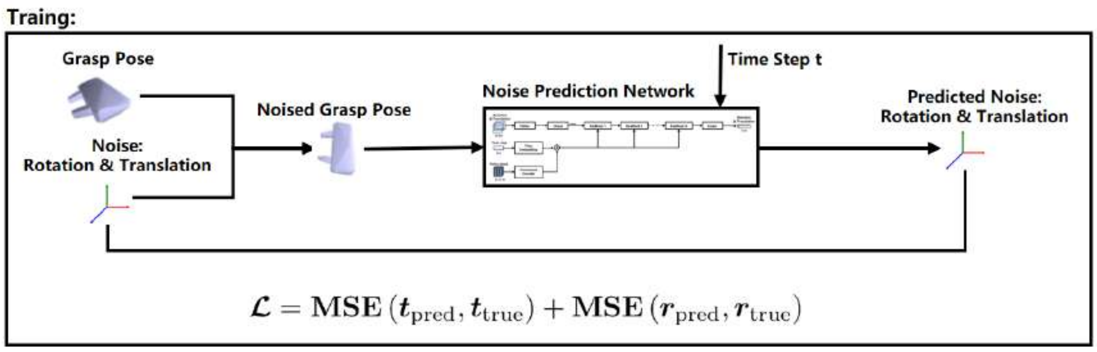
Traing:
Grasp Pose
Noise: Rotation & Translation
Noised Grasp Pose
Noise Prediction Network
Time Step t
Predicted Noise: Rotation & Translation
$\mathcal{L} = \mathrm{MSE}(t_{\mathrm{pred}}, t_{\mathrm{true}}) + \mathrm{MSE}(r_{\mathrm{pred}}, r_{\mathrm{true}})$

Figure 4.2.: Forward process of the grasp diffusion network(GDN).

As outlined in Chapter 2, diffusion models involve both a forward and a reverse process. Unlike basic Diffusion Probabilistic Models (DDPM) [3], our approach incorporates both the translation and rotational components of the grasp pose. In the forward process, noise is added at each timestep, modifying both the translation and rotation of the grasp pose. By knowing the added noise and the grasp pose at each step, we train a Noise Prediction Network to estimate the noise at each timestep. The loss is calculated as the mean squared error between the predicted and actual noise.

4.2.2. Reverse Process

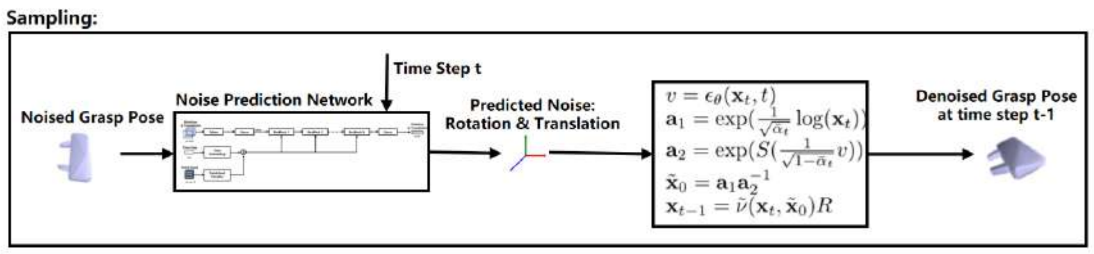
Sampling:
Time Step t
Noised Grasp Pose
Noise Prediction Network
Predicted Noise: Rotation & Translation
$v = \epsilon_\theta(\mathbf{x}_t, t)$
$\mathbf{a}_1 = \exp(\frac{1}{\sqrt{\bar{\alpha}_t}} \log(\mathbf{x}_t))$
$\mathbf{a}_2 = \exp(S(\frac{1}{\sqrt{1-\bar{\alpha}_t}} v))$
$\tilde{\mathbf{x}}_0 = \mathbf{a}_1 \mathbf{a}_2^{-1}$
$\mathbf{x}_{t-1} = \tilde{\nu}(\mathbf{x}_t, \tilde{\mathbf{x}}_0) R$
Denoised Grasp Pose at time step t-1

Figure 4.3.: Reverse process of the grasp diffusion network(GDN).

Once the Noise Prediction Network is trained, we employ the reverse diffusion process to reconstruct a clean grasp pose from the noisy one, iterating backward through the timesteps until the final grasp pose is obtained.

By progressively reducing noise, we transform a noisy distribution into a specific grasping distribution. As illustrated in Figure 4.4, the entire denoising or grasp generation process moves sequentially from left to right.

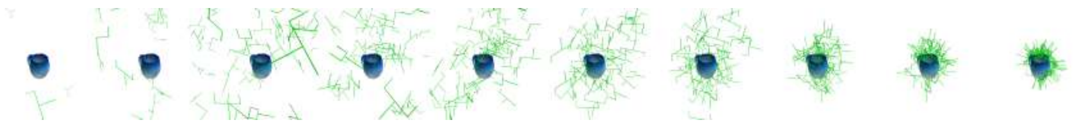

Figure 4.4.: Grasp generation process (from left to right).

4.2.3. Noise Prediction Network

The Noise Prediction Network takes as input the translation and rotation matrices, the current timestep, and the object's point cloud. The output is represented as a rotation and translation vector in Lie algebra. The timestep and point cloud features act as conditional inputs, allowing the model to account for the variability in noise at different stages and adapt to the object features accordingly.

To handle these conditional inputs, we use ResNet [42] and Feature-wise Linear Modulation (FiLM) [43]. ResNet's residual connections help mitigate common training issues like gradient vanishing or explosion, improving training stability and efficiency. This is crucial for our deep model as it needs to capture complex, nonlinear relationships effectively.

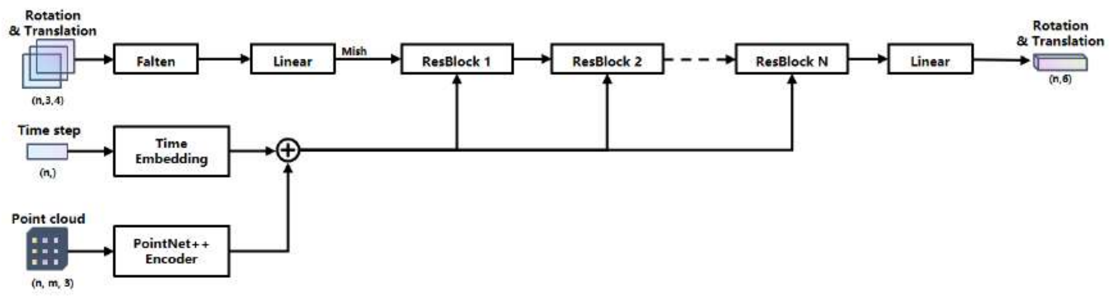

Figure 4.5.: Noise prediction network of the grasp diffusion network(GDN).

FiLM is employed to dynamically adjust feature representations based on conditional inputs, linearly modulating intermediate features to incorporate timestep and point cloud data. This enables the model to flexibly adapt to varying timesteps and object features, resulting in more accurate noise predictions.

ResNet enhances stable deep feature extraction, while FiLM provides the flexibility to adjust based on conditional inputs. Together, these components enable the Noise Prediction Network to handle complex data and accurately predict noise in the grasp pose.

For noise representation, we utilize either a matrix or a 6D vector form. When working with SO(3), we found that directly predicting noise vectors is more effective than learning residual rotations, as it simplifies capturing the stochastic nature and uncertainties of rotation. For SE(3), we chose to learn noise vectors over the 6D representation because working with noise in a linear space avoids the complications of optimizing high-dimensional constraints, making the learning process more stable and efficient.

Directly learning noise vectors also allows for better adaptation to the data distribution, particularly given the varying nature of noise at each timestep in diffusion models. This approach helps the model capture the complex variations and uncertainties inherent in vision-based grasping tasks. Consequently, this method improves learning efficiency, simplifies training, and enhances the model's ability to express uncertainty in both rotation and translation, leading to better performance overall.

4.3. Grasp Refinement

4.3.1. Classifier Guidance

We employ a classifier-guided optimization strategy [44] within the diffusion model to refine the grasp. This choice is motivated by several key reasons.

In diffusion-based generation without classifier guidance, the denoising process relies solely on the capabilities of the Noise Prediction Network. In contrast, classifier guidance leverages an additional quality gradient provided by a classifier, offering an extra guiding signal during generation. This gradient helps adjust the direction of each denoising step, ensuring a gradual improvement in the quality of the generated result. Without classifier guidance, the generation process can suffer from mode collapse or instability. By incorporating classifier guidance, we provide supplementary information on generation quality at each timestep, enabling the model to converge towards higher-quality outcomes.

In the context of grasp generation, we specifically avoid using classifier-free guidance because its approach to handling "bad" grasps contradicts our task objectives. Classifier-free guidance requires the model to learn both conditional and unconditional distributions during training, which means it learns to generate not only "successful" grasps but also "failed" ones. However, generating failed grasps holds no practical value for our task, as our primary goal is to maximize the success rate of grasps. Unlike image generation tasks, where there is no definitive distinction between "good" and "bad" outcomes, grasp generation inherently has a clear objective—to maximize success. Generating unsuccessful grasp postures reduces overall system performance and increases the likelihood of ineffective actions. Therefore, we employ classifier guidance to provide quality gradients that guide the generation towards higher success rates, effectively optimizing both the quality of generated grasps and the overall success rate.

4.3.2. Grasp Evaluator Guidance

Similar to 6-DOF GraspNet, we use a classifier trained on the ACRONYM dataset—referred to as the Grasp Evaluator—to predict the success rate of grasps. The detailed workflow is as follows:

In the network architecture diagram (Figure 4.6), we illustrate the design of the deep learning model for grasp evaluation and generation. The model consists of two main components: the Grasp Evaluator module and the diffusion model module described earlier. These two modules work together to enhance the quality of grasp generation.

Grasp Evaluator Module

The Grasp Evaluator module is designed to score each potential grasp. The evaluation process involves several steps:

1. **Point Cloud Input and Feature Extraction**: The point cloud data is processed through a point cloud encoder, which consists of a linear layer, batch normalization, and a ReLU activation layer. These components extract features from the raw 3D point cloud, resulting in a feature vector z.
2. **Grasp Feature Encoding**: The grasp features, including rotation and translation information, are encoded through a grasp feature encoder. This encoder also uses linear layers, batch normalization, and ReLU activation to process the grasp features.
3. **Feature Fusion and Prediction**: The point cloud features and grasp features are concatenated and input into the prediction network. The prediction network contains two residual blocks (ResNet Block 1 and ResNet Block 2) and a final linear layer to predict the quality score and confidence of the grasp.
4. **Output Quality Score**: The prediction network outputs a quality score and confidence value for the grasp, which are then used in the subsequent noise adjustment process.

Here, f_{encoder} and g_{encoder} represent the point cloud encoder and grasp feature encoder, respectively, while $h_{\text{predictor}}$ represents the prediction network. The predicted quality score is denoted as $\hat{q}$, and the confidence is denoted as $\hat{c}$.

Diffusion Model Module

In the diffusion model module, the score provided by the Grasp Evaluator is used to guide and optimize the generation process. At each timestep of the diffusion model, the predicted noise value is adjusted based on the gradient information from the Grasp Evaluator. This adjustment can be represented as follows:

$$\epsilon_{\text{new}} = \epsilon_{\text{pred}} - w_1 \cdot \nabla_x \log p(y|x)$$

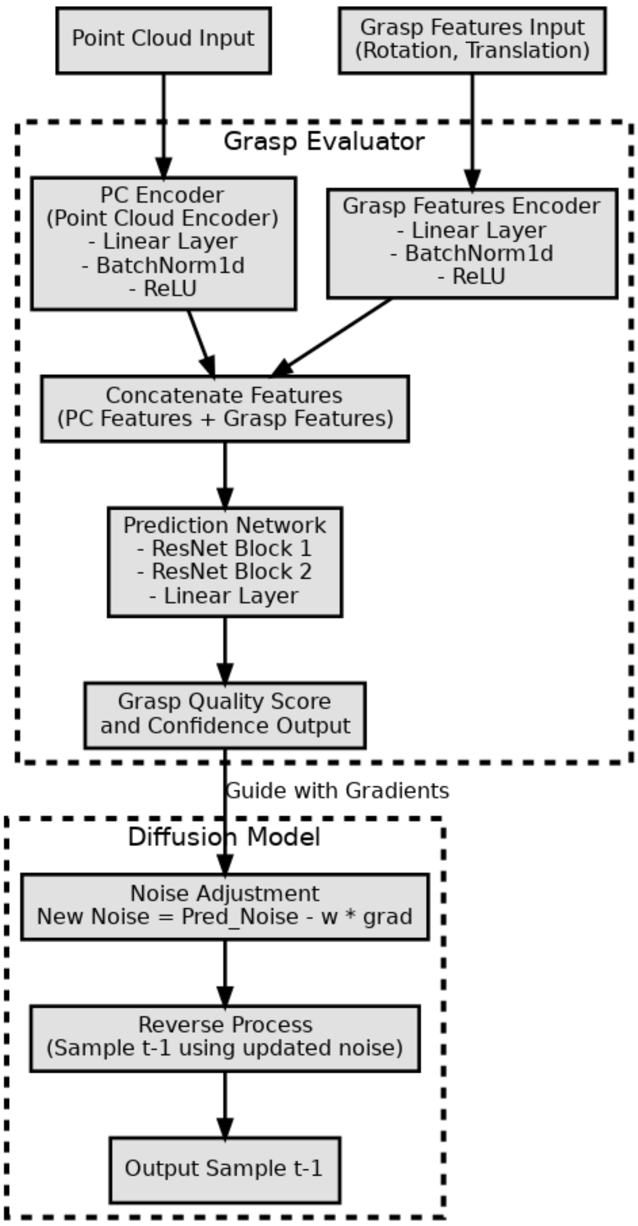
Point Cloud Input
Grasp Features Input (Rotation, Translation)
Grasp Evaluator
PC Encoder (Point Cloud Encoder) - Linear Layer - BatchNorm1d - ReLU
Grasp Features Encoder - Linear Layer - BatchNorm1d - ReLU
Concatenate Features (PC Features + Grasp Features)
Prediction Network - ResNet Block 1 - ResNet Block 2 - Linear Layer
Grasp Quality Score and Confidence Output
Guide with Gradients
Diffusion Model
Noise Adjustment New Noise = Pred_Noise - w * grad
Reverse Process (Sample t-1 using updated noise)
Output Sample t-1

Figure 4.6.: The Grasp Evaluator framework.

where ϵ_{new} is the updated noise value for the reverse sampling process, ϵ_{pred} is the original noise predicted by the diffusion model, w_1 is the guidance coefficient, and $\nabla_x \log p(y|x)$ is the gradient provided by the Grasp Evaluator to adjust the grasp posture towards higher success probability.

The subtraction term reflects the fundamental opposition between the predicted noise ϵ_{pred} and the gradient-guided correction. In this context, the noise ϵ_{pred} represents the inherent uncertainty in the generation process, while the gradient $\nabla_x \log p(y|x)$ works to guide the sample toward the desired outcome, i.e., a successful grasp. By subtracting the gradient, we actively reduce noise components that may lead to failed grasps, thereby shifting the generation process toward states that increase the probability of success. This relationship between noise and gradient correction can be viewed as:

$$\epsilon_{\text{new}} = \epsilon_{\text{pred}} - \sqrt{1-\bar{\alpha}_t} \cdot \nabla_x \log p(y|x),$$

where the term $\sqrt{1-\bar{\alpha}_t}$ scales the gradient as per the current timestep t, aligning the adjustment with the denoising process over time. The negative sign is critical in ensuring that the gradient opposes the noise, guiding the sample toward minimizing the likelihood of grasp failure.

In this formulation, the gradient term $\nabla_{H_t} \log p(y|H_t, t, O)$ represents the rate of change in the success probability $p(y|H_t, t, O)$ of the grasp posture H_t at the current timestep t, with partial point cloud information O. This gradient, computed through Bayesian inference, indicates the optimal direction for improving the grasp.

The combined network of the Grasp Evaluator and the diffusion model aims to generate high-quality grasps through iterative refinement. The Grasp Evaluator serves as a scorer, providing gradient information to the diffusion model, which guides the optimization process. This collaborative mechanism effectively prevents issues like mode collapse during generation, thereby enhancing the diversity and stability of the generated grasps.

Table-SDF Guidance

In addition to classifier guidance, we integrate other mechanisms like the Signed Distance Function (SDF) [44] to prevent collisions with the environment, such as the table. By combining the quality gradient from the Grasp Evaluator with the collision avoidance gradient from the SDF, we achieve a multi-objective optimization that enhances grasp quality while ensuring safety.

Collision Avoidance Using SDF The SDF describes the distance between a point and a surface. For grasp generation, we use SDF to ensure the gripper avoids collisions with the environment. First, we segment the table and obtain its depth information to create a Box-SDF that describes the table's location and boundaries. By computing the table's SDF gradient, we determine a direction for collision avoidance, which is used to optimize the grasp pose.

Gradient-Guided Optimization During grasp generation, the diffusion model predicts a grasp pose and a corresponding noise correction at each timestep. By incorporating both the feedback from the Grasp Evaluator (grasp quality gradient) and the SDF (collision avoidance gradient), we can optimize the grasp effectively. The updated noise is given by:

$$\epsilon_{\text{new}} = \epsilon_{\text{pred}} - w_1 \cdot \nabla_x \log p(y|x) - w_2 \cdot \nabla_x \text{SDF}_{\text{table}}$$

where w_1 and w_2 control the influence of the quality and collision avoidance gradients, respectively. The negative sign ensures that both gradients contribute to reducing noise in a direction that improves grasp success and avoids collisions.

Collaborative and Adversarial Guidance The gradients from the Grasp Evaluator and the SDF may either conflict or complement each other:

- If a grasp pose has high quality but risks collision, the SDF gradient pushes it away from the table until the collision is eliminated.
- If there is no collision risk but the grasp quality is poor, the Grasp Evaluator's gradient guides it towards higher quality.

This multi-objective optimization allows the diffusion model to adjust noise at each timestep, ensuring safe and high-quality grasp generation. This collaborative mechanism enhances grasp effectiveness while avoiding issues like mode collapse, ultimately increasing the diversity and stability of the generated results.

4.4. Sampling Process Acceleration

In our Grasp Diffusion Network, we extend the acceleration techniques of Denoising Diffusion Implicit Models (DDIM) [7] to reduce computational cost while enhancing precision. By tailoring the diffusion and sampling processes specifically for grasping tasks, we introduce several enhancements that outperform traditional DDIM methods. These key modifications are detailed in the following subsections.

4.4.1. Adaptive Skip-Step Sampling Strategy

We propose an adaptive skip-step sampling strategy that adjusts based on task complexity. Unlike the fixed skip patterns used in traditional DDIM methods (e.g., uniform, quadratic, exponential), our approach dynamically identifies critical time points based on the unique characteristics of the grasping scenario. This adaptive selection captures essential state changes with fewer sampling steps, ensuring efficiency without sacrificing precision. The selection of time steps is adjusted based on task complexity, allowing more sampling steps in regions with greater complexity to effectively model critical changes.

4.4.2. Dynamic Noise Control for Enhanced Determinism

To enhance determinism during the generation process, we introduce a dynamic noise control mechanism. Unlike the fixed parameter η used in DDIM, our approach adjusts η based on the grasping stage. As the generated grasp approaches the target, η is gradually reduced, increasing determinism and enhancing precision in the final output.

Algorithm 8 Dynamic Noise Control

```
Input: Time step $t$, Current stage $stage$
Output: Noise parameter $\eta$
if far_from_target(t) then
    $\eta \leftarrow$ high value (exploration phase)
else if near_to_target(t) then
    $\eta \leftarrow$ low value (exploitation phase)
end
return $\eta$
```

This dynamic adjustment of η reduces uncertainty when the model is near the target, resulting in more precise and stable grasp poses.

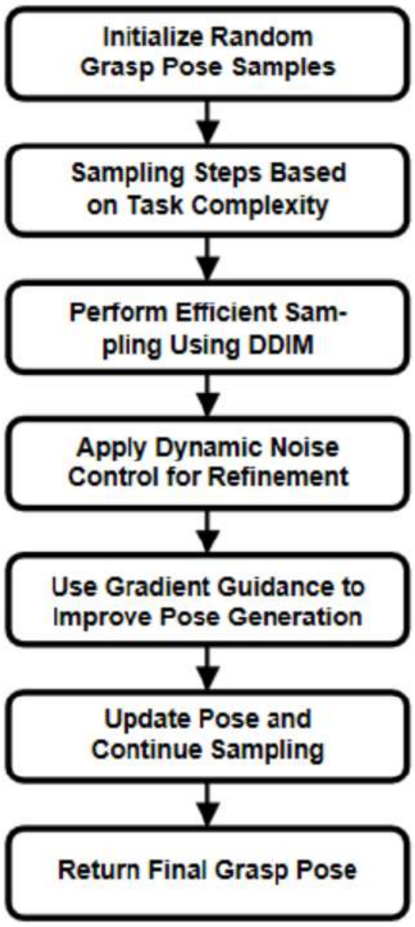
Initialize Random Grasp Pose Samples
Sampling Steps Based on Task Complexity
Perform Efficient Sam-pling Using DDIM
Apply Dynamic Noise Control for Refinement
Use Gradient Guidance to Improve Pose Generation
Update Pose and Continue Sampling
Return Final Grasp Pose

Figure 4.7.: Grasp Diffusion Network(GDN) sampling process using DDIM for accelerated grasp pose generation.

5. Experiments

In this chapter, we evaluate our proposed method through experiments. We detail the setup, environments, dataset, comparative analysis, training procedures, and evaluation results.

5.1. Setup

5.1.1. Environments

Our research aims to develop a grasp diffusion model based on partial point cloud features. To validate its effectiveness, we created benchmark environments that significantly influence the model's learning ability and the quality of generated grasps. We tested our model in both the Isaac Gym simulation environment and the real world, as shown in Figure 5.1. Various objects were placed at the center of a table, with a fixed camera at the side. Using partial point cloud data, our model generated grasp poses to control a Panda robotic arm. We measured the success rate of these grasps and compared the generated grasp distributions against ground truth data using the Earth Mover's Distance (EMD).

5.1.2. Dataset

We trained and evaluated our model using the ACRONYM dataset [45], which contains approximately 177 million parallel gripper grasps across 8,872 objects in 262 categories, each labeled as successful or failed. Each category contains training and testing subsets, respectively used for model training and evaluation. To improve data quality and grasp evaluation accuracy, we further preprocessed the object mesh models.

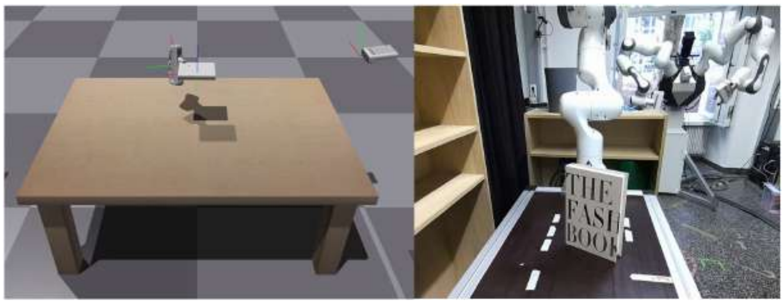
Figure 5.1.: Grasp testing in the Isaac Gym simulation environment (left) and the real world (right).

Starting with raw object meshes from ShapeNet [46], we used the Manifold tool to make the models watertight, generating closed meshes without holes. This step is crucial for reliable collision detection and accurate grasp pose evaluation, as watertight meshes better represent object shapes and reduce errors in grasp quality assessment. We also simplified the meshes to lower model complexity and speed up training and evaluation.

We combined the watertight, simplified meshes with grasp information from the ACRONYM dataset, ensuring each grasp file correctly referenced its corresponding object model. Focusing on categories such as "Mug" and "Cup," we divided the dataset into training and testing sets for evaluation. With this carefully prepared data, we effectively trained the Grasp Diffusion Network.

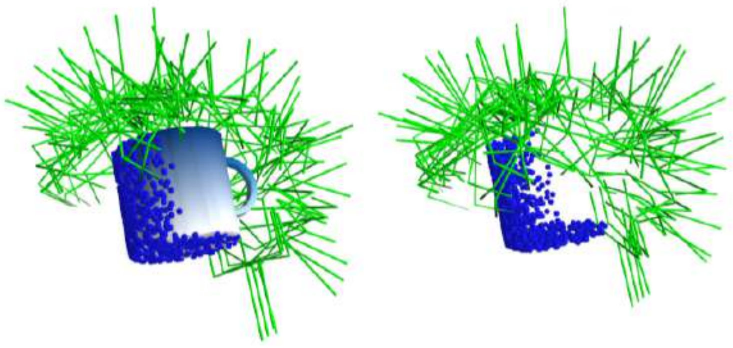
Figure 5.2.: An example object from the ACRONYM dataset, its corresponding grasp, and the partial point cloud sampled from the object's mesh.

5.2. Capturing Point Cloud Features

5.2.1. Depth Map to Point Cloud

We converted depth maps into point clouds to capture the three-dimensional structure of objects. This process involved reading pixel depth values and mapping each pixel to real-world coordinates using the camera's projection and view matrices.

To improve accuracy and avoid background interference, we first performed object segmentation. In the Isaac Gym simulation environment, we used the rendering ID of each pixel for direct segmentation, resulting in depth maps containing only the target object. In the real world, we applied the FastSAM [47] segmentation method on RGB images, then used the segmentation result on the depth map to extract the target object's depth information. This provided a depth map of only the target object, which we transformed into a point cloud.

Each pixel in the depth map represents its distance from the camera. Using the camera's intrinsic parameters, we converted the 2D pixel coordinates (u, v) and depth value Z into 3D points (X_c, Y_c, Z_c) in the camera coordinate system:

$$X_c = \frac{(u-c_u)Z}{f_u}, \quad Y_c = \frac{(v-c_v)Z}{f_v}, \quad Z_c = Z$$

Here, f_u and f_v are the camera's focal lengths, and c_u and c_v are the coordinates of the optical center on the image plane. In simulation environments like Isaac Gym, these parameters are available through the API. In real-world scenarios, we obtained them through camera calibration, often using a chessboard pattern from multiple viewpoints.

After converting to the camera coordinate system, we transformed the point cloud to the world coordinate system using the camera's extrinsic parameters (the view matrix). The view matrix describes the camera's position and orientation in the world coordinate system. Using its inverse, we mapped points from the camera to the world coordinate system:

$$P_{\text{world}} = V^{-1} P_{\text{cam}}$$

For linear transformations, we represented point clouds in homogeneous coordinates by adding an additional dimension, converting them into $(X, Y, Z, 1)$. This facilitates transformations like translation and rotation, allowing the point cloud to be transferred from the camera coordinate system to the environment coordinate system for subsequent analysis.

In the simulation environment, intrinsic and extrinsic camera parameters are obtained directly from the API, simplifying the conversion. In the real world, these parameters are determined through camera calibration, with the intrinsic matrix K defined as:

$$K = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

The extrinsic parameters, including the rotation matrix R and translation vector t, describe the camera's orientation and position. With these parameters, we accurately mapped each pixel in the depth map to 3D world coordinates.

5.2.2. Point Cloud to Point Cloud Features

To process the point cloud data and extract low-dimensional features from high-dimensional input, we used the PointNet++ encoder [39]. PointNet++ is a neural network with rotational invariance and local feature extraction capabilities, well-suited for handling partial point clouds. By progressively sampling and aggregating local point features, PointNet++ generates a compact, representative feature set that effectively captures the object's shape and spatial relationships.

Its rotational invariance makes PointNet++ highly advantageous for grasp generation in partial point cloud scenarios, as the grasp must be robust to various object orientations. The extracted features effectively describe the geometry, supporting grasp generation and other 3D tasks.

Using object segmentation to preprocess depth maps significantly improved point cloud accuracy. By focusing depth information on the target object and minimizing background noise, we enhanced the quality of the point clouds. PointNet++ then extracted low-dimensional features from the high-dimensional point cloud, which were crucial for our grasp generation tasks.

5.3. Comparative Experiments

5.3.1. Experimental Setup

To evaluate the performance of various methods in grasp generation, we conducted comparative experiments involving different grasp generators. This comprehensive exploration aimed to compare their performance in handling different point cloud data distributions and grasp pose generation tasks.

5.3.2. Grasp Generator Comparison

We evaluated the performance of several grasp generators, including existing methods and our proposed approach:

- **GraspGeneratorVAE (Alternative generative method) [41]**: This generator employs a variational autoencoder (VAE) architecture to model the distribution of feasible grasps and generate corresponding configurations.
- **GraspGeneratorSE3Diffusion (Baseline) [40]**: Serving as our baseline, this method adapts SE(3) diffusion models into an energy-based framework for grasp generation, capturing necessary spatial transformations.
- **GraspGeneratorGDN (Our method)**: Our proposed Grasp Diffusion Network (GDN) leverages diffusion processes to synthesize grasps, aiming to enhance generation quality and performance over existing methods.

5.4. Training Implementation Details

5.4.1. Data Loading and Batching

Each batch consisted of 32 objects, each with 32 grasp poses, resulting in a total batch size of 1,024. We used 16 worker threads to parallelize data loading and speed up the preparation process.

5.4.2. Training Environment and Configuration

We conducted the training on a Slurm cluster using either an NVIDIA RTX3090 or A5000 GPU. Each task was assigned one GPU and eight CPU cores.

5.4.3. Training Epochs and Learning Rate Scheduling

During the initial 100,000 epochs, we did not apply learning rate decay to stabilize early optimization. Afterward, we used a lambda-based scheduling strategy with cosine annealing restarts for dynamic adjustment, starting with an initial learning rate of 0.0003.

5.5. Results

5.5.1. Comparison of EMD and Success Rate Across Methods

Table 5.1 and Figure 5.3 compare the performance of SE3Diffusion (baseline), Variational Autoencoder (VAE), and our proposed Grasp Diffusion Network (GDN) across multiple object categories: Bowl, Cup, Mug, and a broader set called CAT10, which includes Book, Bottle, Hammer, Milk Carton, Rubik's Cube, Shampoo, Teapot, and the aforementioned objects. We evaluate the models using two primary metrics: Earth Mover's Distance (EMD) and success rate. A lower EMD indicates a closer alignment between the predicted and ground-truth grasp distributions, while a higher success rate reflects more reliable grasp generation. Figure 5.3 visualizes these results, plotting EMD on the x-axis and success rate on the y-axis for the CAT10 objects.

Earth Mover's Distance (EMD): EMD measures the difference between the predicted and ground-truth grasp distributions; lower values indicate better performance. As shown in Table 5.1 and Figure 5.3, GDN generally achieves lower median EMD values compared to SE3Diffusion across various categories. For instance, in the mug category, GDN's median EMD is 0.140, outperforming SE3Diffusion's 0.154. However, VAE achieves a slightly lower median EMD of 0.139 in this category. In the CAT10 category, GDN achieves a median EMD of 0.135, which is lower than SE3Diffusion's 0.154. These results demonstrate the robustness of GDN across diverse object shapes.

Success Rate: Success rate measures the proportion of successful grasps generated by the model; higher values indicate better grasp quality. BothTable 5.1 and Figure 5.3 show that GDN generally performs better than the baseline models in terms of success rate. In the mug category, GDN achieves a median success rate of 0.860, surpassing SE3Diffusion's 0.790 and VAE's 0.800, indicating superior reliability in practical grasping scenarios. In the CAT10 category, GDN's median success rate is 0.840, which is higher than SE3Diffusion's 0.800, reflecting improved performance across different objects.

Mean, Standard Deviation, and Median Analysis: Analyzing the mean, standard deviation, and median values for both EMD and success rate highlights GDN's strengths in terms of accuracy and consistency. GDN consistently shows lower EMD values and higher success rates compared to the baselines, with reduced variability across object categories. As depicted in Figure 5.3, GDN achieves a balance between low EMD and high success rate, demonstrating superior mean performance and lower variance compared to SE3Diffusion, particularly in the CAT10 object category. This underscores GDN's robustness and suitability for real-world grasping tasks, where both accuracy and reliability are critical.

In summary, the results across both metrics demonstrate that GDN generally outperforms both SE3Diffusion and VAE, particularly in terms of achieving lower EMD values and higher success rates with reduced variance. These findings validate the effectiveness of GDN in generating accurate and dependable grasps across various object categories.

Table 5.1.: Comparison of Grasp Generation Methods in terms of EMD and Success Rate. All results are based on the test dataset. CAT10 includes 10 different object categories.

Methods	Category	EMD (Mean/Std/Median)	Success Rate (Mean/Std/Median)
GroundTruth	Bowl	0.101/0.015/0.102	0.854/0.202/0.920
SE3Diffusion (Baseline)	Bowl	0.168/0.089/0.134	0.735/0.318/0.890
VAE	Bowl	0.134/0.038/0.121	0.737/0.262/0.840
GDN(Our)	Bowl	0.145/0.045/0.129	0.700/0.261/0.810
GroundTruth	Cup	0.130/0.016/0.126	0.896/0.128/0.950
SE3Diffusion (Baseline)	Cup	0.172/0.043/0.157	0.881/0.149/0.940
VAE	Cup	0.168/0.026/0.160	0.850/0.087/0.880
GDN(Our)	Cup	0.171/0.031/0.168	0.875/0.107/0.900
GroundTruth	Mug	0.110/0.016/0.108	0.954/0.045/0.960
SE3Diffusion (Baseline)	Mug	0.164/0.039/0.154	0.751/0.166/0.790
VAE	Mug	0.144/0.029/0.139	0.779/0.106/0.800
GDN(Our)	Mug	0.143/0.024/0.140	0.834/0.111/0.860
GroundTruth	CAT10	0.108/0.023/0.105	0.857/0.195/0.940
SE3Diffusion (Baseline)	CAT10	0.162/0.051/0.154	0.721/0.244/0.800
VAE	CAT10	0.144/0.039/0.135	0.703/0.197/0.775
GDN(Our)	CAT10	0.145/0.042/0.135	0.755/0.225/0.840

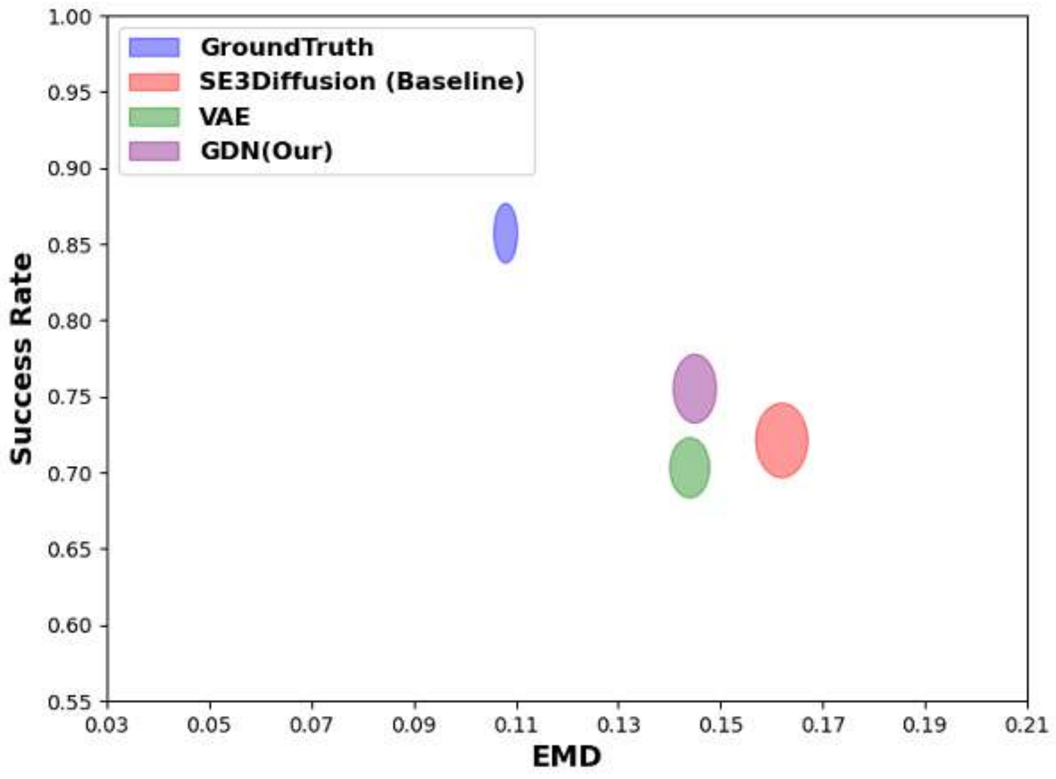

Figure 5.3.: Performance comparison of grasp generation methods. EMD (x-axis) and Success Rate (y-axis) are shown for CAT10 objects, with our method GDN demonstrating superior mean performance and lower variance.

5.5.2. Comparison of Grasp Generation Speed Across Methods

Figure 5.4 compares the grasp generation speeds of our proposed Grasp Diffusion Network (GDN) and the baseline model SE3Diffusion. The top plot shows the average time per iteration, while the bottom plot illustrates the cumulative grasp generation time over all iterations.

GDN achieves a per-iteration time approximately 0.3 seconds faster than SE3Diffusion. While this difference may seem minimal, it accumulates significantly over multiple iterations. This cumulative advantage is evident in the bottom plot, where the total time difference between GDN and SE3Diffusion widens progressively. By the final iteration, GDN demonstrates a substantially lower cumulative time, underscoring its efficiency.

This efficiency gain offers a significant advantage for robotic manipulation tasks requiring real-time responsiveness. The faster performance of GDN makes it more suitable for applications where rapid and reliable grasp generation is critical, positioning it as a superior alternative to SE3Diffusion.

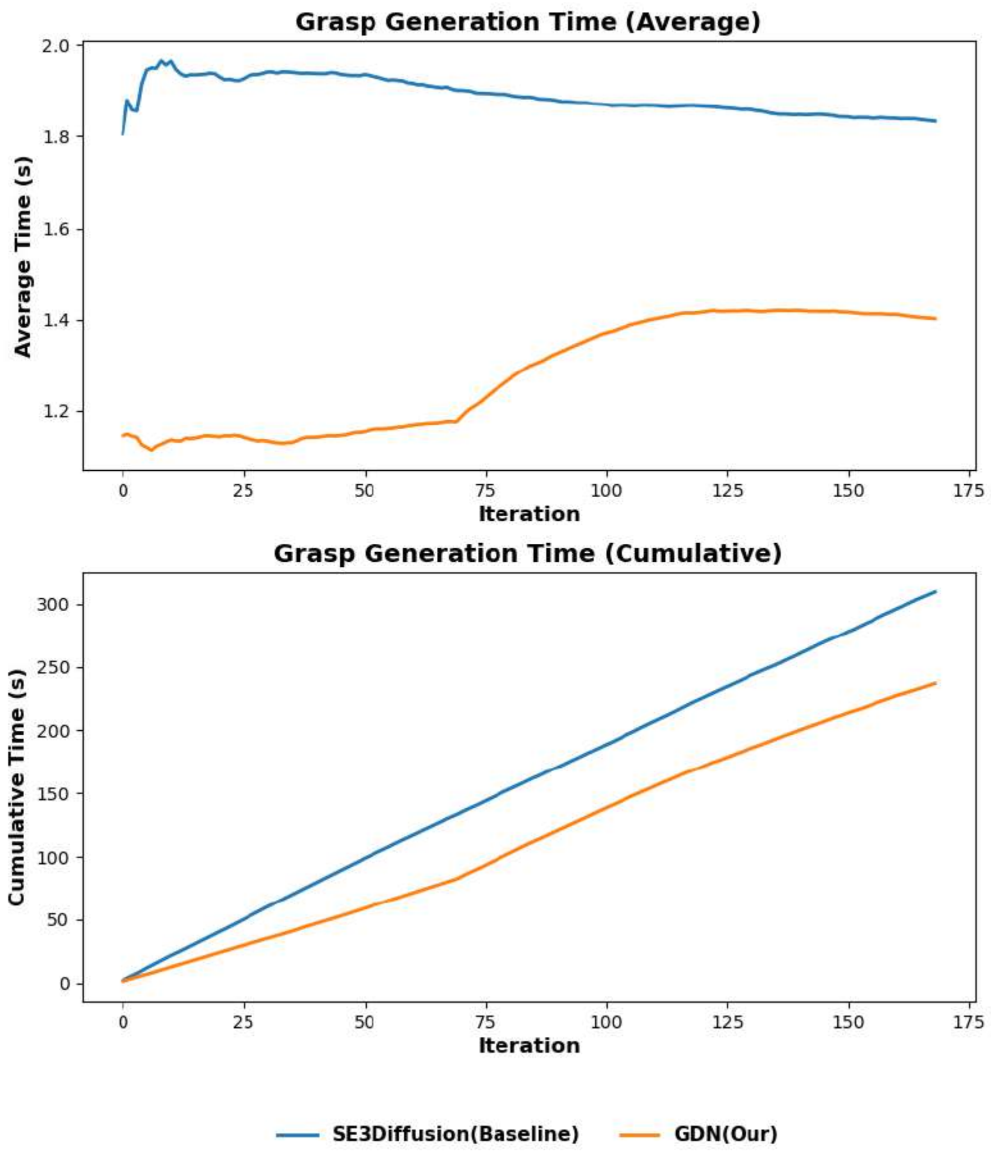

Figure 5.4.: Comparison of grasp generation time between GDN (our method) and SE3Diffusion (baseline). The top plot shows the average generation time per iteration, and the bottom plot shows the cumulative grasp generation time.

6. Conclusion

This thesis addressed the challenge of vision-based robotic grasping, specifically focusing on learning generalized grasp representations from demonstrations. To achieve this goal, we introduced the Grasp Diffusion Network (GDN), a novel framework for grasp generation. GDN is among the few data-driven approaches capable of directly generating grasp poses from partial point clouds, enabling efficient learning of complex, object-centered 6-DoF grasp distributions.

We conducted extensive simulations to evaluate GDN's performance, which demonstrated superior results, particularly in sampling speed, compared to baseline methods. Compared to existing generative modeling approaches, GDN effectively generated grasps and exhibited strong scalability to large object sets. Additionally, we validated our approach by training the GDN model on synthetic single-view point clouds and testing it in real-world environments, demonstrating effective transfer from simulation to reality.

Given the improvements introduced by diffusion models in GDN, we believe this approach can be extended to address other complex dexterous manipulation tasks, such as in-hand object manipulation or grasping in cluttered environments. Addressing these challenges and the current limitations of our method represents a promising direction for future research.

7. Discussion

7.1. Limitations

Despite its contributions, the proposed method has several notable limitations:

1. The grasp generation relies on partial point clouds from a single viewpoint, potentially reducing the quality and success rate of the generated grasps.
2. Even with acceleration strategies like DDIM, the diffusion model remains computationally intensive, posing challenges for tasks that require high real-time performance.
3. The approach does not account for complex environmental constraints, such as avoiding collisions with other objects; therefore, both simulation and real-world experiments were conducted in relatively simple environments.
4. Sampling from an isotropic Gaussian distribution in $SO(3)$ necessitates integrating the cumulative distribution function, which is computationally expensive and lacks a closed-form solution.

7.2. Future Work

To overcome the current limitations, we plan to focus our future work on several key areas:

1. **Improving Grasp Quality and Success Rates**: Currently, our diffusion model uses partial point clouds from single viewpoints, which limits performance. To address this, we intend to reconstruct more complete object models from these limited observations using techniques such as Iterative Closest Point (ICP) [48],

Neural Radiance Fields (NeRF) [49], and Point Completion Networks (PCN) [50]. Enhancing the input data quality through these reconstruction methods should significantly improve grasp accuracy and success rates.

2. **Speeding Up the Sampling Process**: While we have utilized DDIM to accelerate sampling, we plan to explore additional techniques to further enhance the diffusion model's sampling speed. Methods such as automated diffusion optimization [51] and progressive distillation [52] could reduce the number of diffusion steps required without compromising performance.

3. **Adapting to Complex Environments**: To improve adaptability in complex settings, we plan to incorporate additional constraints, such as classifiers or guidance terms, inspired by conditional diffusion models. This approach should enhance the model's robustness and performance in diverse environments.

4. **Improving Efficiency in Rotational Transformations**: Addressing inefficiencies in sampling rotations is a priority. We will investigate efficient numerical methods tailored for rotation groups like SO(3), such as Runge-Kutta-Munthe-Kaas (RK-MK) integrators [53], to enhance sampling efficiency in rotational transformations.

5. **Exploring Alternative Generative Models**: Finally, to further optimize model efficiency and broaden applicability, we will assess other generative models—such as flow matching [54] and recent diffusion model variants—for their potential benefits in specific tasks.

Bibliography

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[4] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5745–5753, 2019.

[5] F. S. Grassia, "Practical parameterization of rotations using the exponential map," *Journal of graphics tools*, vol. 3, no. 3, pp. 29–48, 1998.

[6] V. De Bortoli, "Convergence of denoising diffusion models under the manifold hypothesis," *arXiv preprint arXiv:2208.05314*, 2022.

[7] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[8] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.

[9] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241, Springer, 2015.

[10] T. Savjolova, "Preface to novye metody issledovanija tekstury polikristalliceskich materialov," *Metallurgija, Moscow*, vol. 4, no. 2, pp. 6–2, 1985.

[11] A. Leach, S. M. Schmon, M. T. Degiacomi, and C. G. Willcocks, “Denoising diffusion probabilistic models on so (3) for rotational alignment,” 2022.

[12] P. Dhariwal and A. Nichol, “Diffusion models beat gans on image synthesis,” *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[13] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical text-conditional image generation with clip latents,” *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[14] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic, *et al.*, “Deep learning approaches to grasp synthesis: A review,” *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3994–4015, 2023.

[15] V. Mayer, Q. Feng, J. Deng, Y. Shi, Z. Chen, and A. Knoll, “Ffhnet: Generating multi-fingered robotic grasps for unknown objects in real-time,” in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 762–769, IEEE, 2022.

[16] M. Ciocarlie, C. Goldfeder, and P. Allen, “Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem,” in *Robotics: Science and systems manipulation workshop-sensing and adapting to the real world*, 2007.

[17] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, “Diffusion models: A comprehensive survey of methods and applications,” *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.

[18] P. Li, T. Liu, Y. Li, Y. Geng, Y. Zhu, Y. Yang, and S. Huang, “Gendexgrasp: Generalizable dexterous grasping,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8068–8074, IEEE, 2023.

[19] J. Lu, H. Kang, H. Li, B. Liu, Y. Yang, Q. Huang, and G. Hua, “Ugg: Unified generative grasping,” *arXiv preprint arXiv:2311.16917*, 2023.

[20] A. Wu, M. Guo, and C. K. Liu, “Learning diverse and physically feasible dexterous grasps with generative model and bilevel optimization,” *arXiv preprint arXiv:2207.00195*, 2022.

[21] S. Ottenhaus, D. Renninghoff, R. Grimm, F. Ferreira, and T. Asfour, “Visuo-haptic grasping of unknown objects based on gaussian process implicit surfaces and deep learning,” in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pp. 402–409, IEEE, 2019.

[22] J. Lundell, E. Corona, T. N. Le, F. Verdoja, P. Weinzaepfel, G. Rogez, F. Moreno-Noguer, and V. Kyrki, “Multi-fingan: Generative coarse-to-fine sampling of multi-finger grasps,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4495–4501, IEEE, 2021.

[23] J. Lundell, F. Verdoja, and V. Kyrki, “Ddgc: Generative deep dexterous grasping in clutter,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6899–6906, 2021.

[24] M. Van der Merwe, Q. Lu, B. Sundaralingam, M. Matak, and T. Hermans, “Learning continuous 3d reconstructions for geometrically aware grasping,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11516–11522, IEEE, 2020.

[25] Q. Lu, M. Van der Merwe, and T. Hermans, “Multi-fingered active grasp learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8415–8422, IEEE, 2020.

[26] Q. Lu, M. Van der Merwe, B. Sundaralingam, and T. Hermans, “Multifingered grasp planning via inference in deep neural networks: Outperforming sampling by learning differentiable models,” *IEEE Robotics & Automation Magazine*, vol. 27, no. 2, pp. 55–65, 2020.

[27] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, “Generating grasp poses for a high-dof gripper using neural networks,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1518–1525, IEEE, 2019.

[28] W. Wei, D. Li, P. Wang, Y. Li, W. Li, Y. Luo, and J. Zhong, “Dvgg: Deep variational grasp generation for dextrous manipulation,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1659–1666, 2022.

[29] Y. Qin, B. Huang, Z.-H. Yin, H. Su, and X. Wang, “Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation,” in *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[30] M. Popović, G. Kootstra, J. A. Jørgensen, D. Kragic, and N. Krüger, “Grasping unknown objects using an early cognitive vision system for general scene understanding,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 987–994, IEEE, 2011.

[31] A. Bicchi and V. Kumar, “Robotic grasping and contact: A review,” in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and*

automation. Symposia proceedings (Cat. No. 00CH37065), vol. 1, pp. 348–353, IEEE, 2000.

[32] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[33] C. Choi, W. Schwarting, J. DelPreto, and D. Rus, "Learning object grasping for soft robot hands," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2370–2377, 2018.

[34] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2901–2910, 2019.

[35] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13438–13444, IEEE, 2021.

[36] Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, and Y. Zhu, "Synergies between affordance and geometry: 6-dof grasp detection via implicit representations," *arXiv preprint arXiv:2104.01542*, 2021.

[37] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6232–6238, IEEE, 2020.

[38] Y. Zhou and K. Hauser, "6dof grasp planning by optimizing a deep learning scoring function," in *Robotics: Science and systems (RSS) workshop on revisiting contact-turning a problem into a solution*, vol. 2, p. 6, 2017.

[39] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[40] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki, "Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5923–5930, IEEE, 2023.

[41] K. R. Barad, A. Orsula, A. Richard, J. Dentler, M. Olivares-Mendez, and C. Martinez, "Graspldm: Generative 6-dof grasp synthesis using latent diffusion models," *arXiv preprint arXiv:2312.11243*, 2023.

[42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[43] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[44] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, “Deepsdf: Learning continuous signed distance functions for shape representation,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

[45] C. Eppner, A. Mousavian, and D. Fox, “Acronym: A large-scale grasp dataset based on simulation,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6222–6227, IEEE, 2021.

[46] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, “Shapenet: An information-rich 3d model repository,” *arXiv preprint arXiv:1512.03012*, 2015.

[47] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, “Fast segment anything,” *arXiv preprint arXiv:2306.12156*, 2023.

[48] J. Yang, H. Li, D. Campbell, and Y. Jia, “Go-icp: A globally optimal solution to 3d icp point-set registration,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2241–2254, 2015.

[49] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[50] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, “Pcn: Point completion network,” in *2018 international conference on 3D vision (3DV)*, pp. 728–737, IEEE, 2018.

[51] L. Li, H. Li, X. Zheng, J. Wu, X. Xiao, R. Wang, M. Zheng, X. Pan, F. Chao, and R. Ji, “Autodiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7105–7114, 2023.

[52] T. Salimans and J. Ho, “Progressive distillation for fast sampling of diffusion models,” *arXiv preprint arXiv:2202.00512*, 2022.

[53] Y. Jagvaral, F. Lanusse, and R. Mandelbaum, “Unified framework for diffusion generative models in so (3): applications in computer vision and astrophysics,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 12754–12762, 2024.

[54] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” *arXiv preprint arXiv:2210.02747*, 2022.

A. Further Results

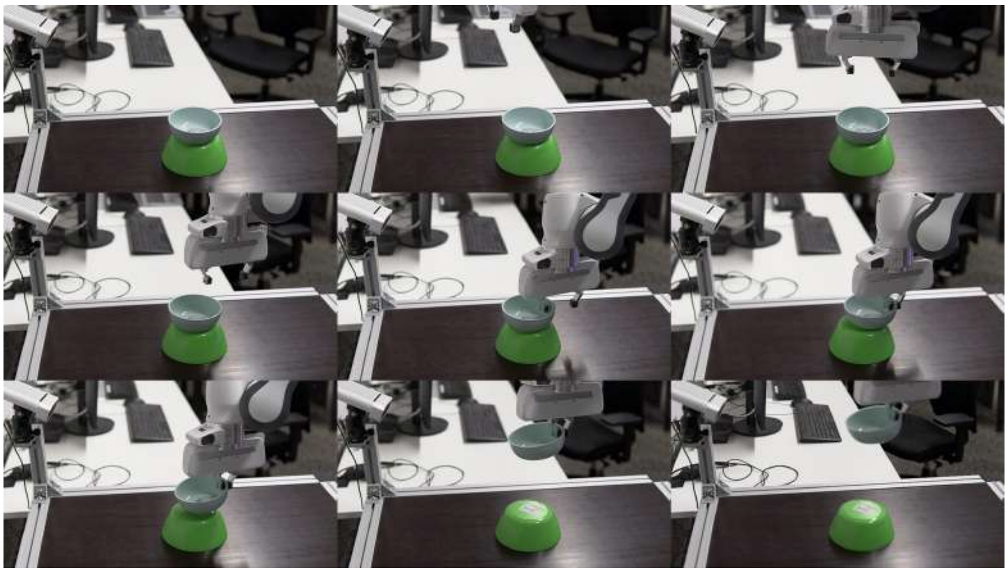

Figure A.1.: Evaluating GDN in the real world: the Panda robotic arm successfully grasps a bowl using a single depth camera.

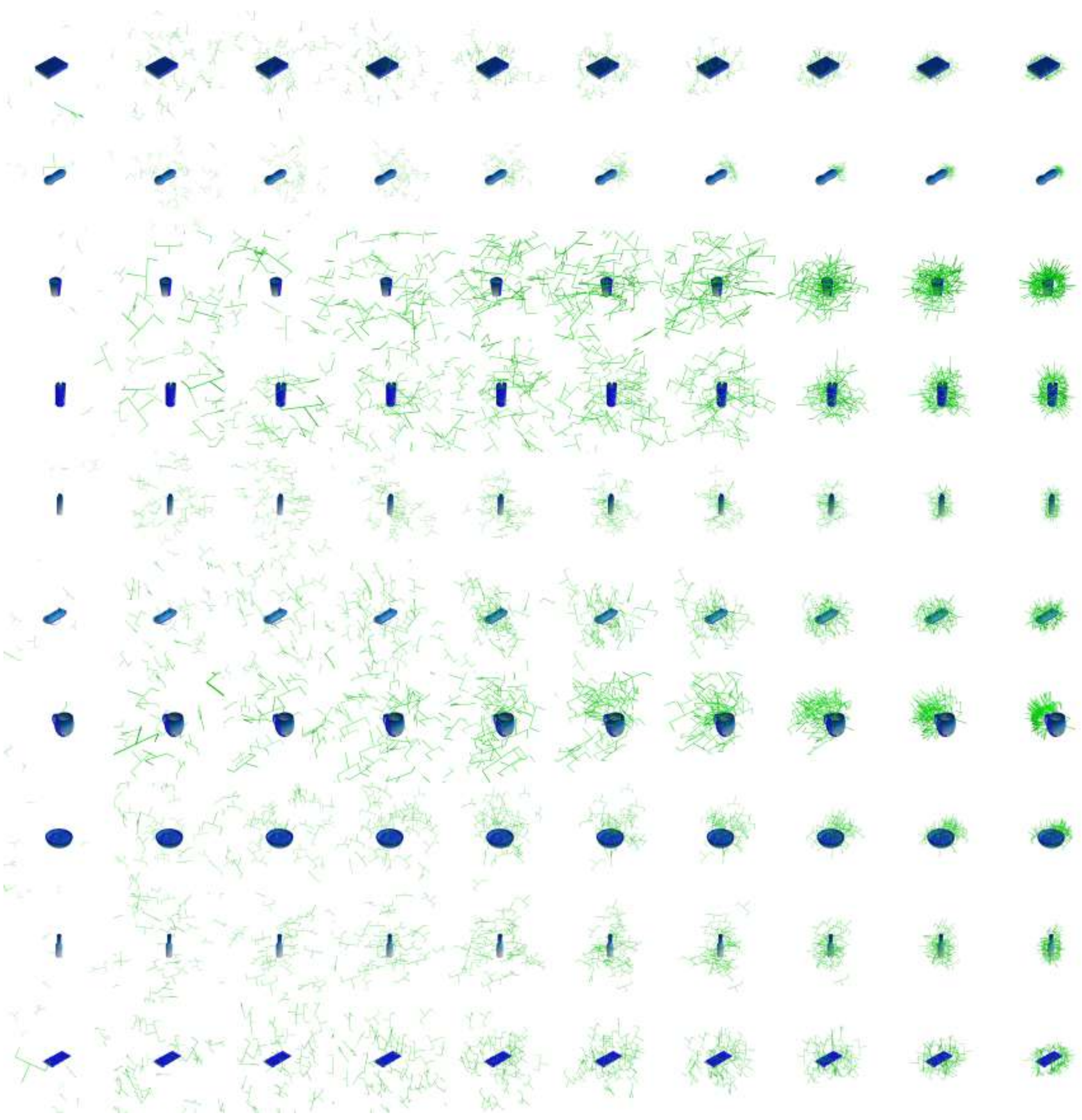

Figure A.2.: The GDN grasp generation (denoising) process: effective grasp distributions are generated for objects of various shapes and categories.

B. Detailed Explanation of Diffusion Model Principles

B.1. Derivation of DDPM in Euclidean Space

B.1.1. Forward Diffusion Process

In Denoising Diffusion Probabilistic Models (DDPM), the forward diffusion process gradually adds Gaussian noise to a data sample x_0 over discrete time steps. At each time step t, the noisy sample x_t is generated from x_{t-1} using:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t \mid \sqrt{\alpha_t}\, x_{t-1}, (1-\alpha_t)\mathbf{I}\right),$$

where $\alpha_t \in (0,1)$ controls the noise scale, and $\mathbf{I}$ is the identity matrix.

By composing these steps, the distribution of x_t conditioned on x_0 becomes:

$$q(x_t \mid x_0) = \mathcal{N}\left(x_t \mid \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t)\mathbf{I}\right),$$

with $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

B.1.2. Reverse Diffusion Process

The reverse diffusion process aims to recover x_{t-1} from x_t. Using Bayes' theorem, the conditional distribution is:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(x_{t-1} \mid \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \mathbf{I}\right),$$

where:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t,$$

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t, \quad \beta_t = 1 - \alpha_t.$$

B.1.3. Reconstructing the Original Sample x_0

By training a neural network $\epsilon_\theta(x_t, t)$ to predict the added noise, we can estimate x_0 from x_t as:

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(x_t - \sqrt{1-\bar{\alpha}_t}\, \epsilon_\theta(x_t, t) \right).$$

B.2. Derivation of DDPM on $SO(3)$ Space

B.2.1. Forward Diffusion Process

Extending DDPM to the rotation group $SO(3)$, we define the forward process using the exponential map:

$$q(R_t \mid R_{t-1}) = \exp\left(-\frac{1}{2}\sigma_t^2 \left\| \log\left(R_t R_{t-1}^\top \right) \right\|^2 \right),$$

where $R_t, R_{t-1} \in SO(3)$, $\sigma_t^2 = 1 - \alpha_t$, and $\log(\cdot)$ denotes the matrix logarithm.

B.2.2. Reverse Diffusion Process

The reverse process on $SO(3)$ utilizes the logarithmic and exponential maps:

$$R_{t-1} = \exp\left(\log(R_t) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(R_t, t) \right),$$

where $\epsilon_\theta(R_t, t)$ predicts the noise in the Lie algebra associated with $SO(3)$.

B.3. Derivation of DDIM in Euclidean Space

B.3.1. Reverse Sampling Process

In Denoising Diffusion Implicit Models (DDIM), the deterministic reverse process is:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\epsilon_\theta(x_t, t).$$

B.4. Derivation of DDIM on $SO(3)$ Space

B.4.1. Reverse Sampling Process

For $SO(3)$, the DDIM reverse process becomes:

$$R_{t-1} = \exp\left(\sqrt{1-\bar{\alpha}_{t-1}}\,\epsilon_\theta(R_t, t)\right)\hat{R}_0,$$

where $\hat{R}_0 = R_t \exp\left(-\sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(R_t, t)\right)$.

B.5. Pseudocode for DDPM and DDIM

B.5.1. DDPM on $SO(3)$ Training Algorithm

Algorithm 9 Training DDPM on $SO(3)$

```
repeat
    Sample time step $t \sim$ Uniform($\{1, \dots, T\}$);
    Sample $R_0$ from the data distribution;
    Sample noise $\epsilon$ from a suitable distribution on $\mathfrak{so}(3)$;
    Compute $R_t = R_0 \exp\left(\sqrt{1-\bar{\alpha}_t}\,\epsilon\right)$;
    Predict $\hat{\epsilon} = \epsilon_\theta(R_t, t)$;
    Update $\theta$ to minimize $\|\epsilon - \hat{\epsilon}\|^2$;
until converged;
```

B.5.2. DDPM on $SO(3)$ Sampling Algorithm

Algorithm 10 Sampling from DDPM on $SO(3)$

```
Initialize $R_T$ from the uniform distribution on $SO(3)$;
for $t = T, \dots, 1$ do
    Predict $\hat{\epsilon} = \epsilon_\theta(R_t, t)$;
    if $t > 1$ then
        Sample noise $\epsilon$ from a distribution on $\mathfrak{so}(3)$;
    else
        Set $\epsilon = 0$;
    end
    Update $R_{t-1} = \exp\left(-\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}\right) R_t \exp\left(\sqrt{\beta_t}\,\epsilon\right)$; end
```

B.5.3. DDIM on $SO(3)$ Sampling Algorithm

Algorithm 11 Sampling from DDIM on $SO(3)$

```
Initialize $R_T$ from the uniform distribution on $SO(3)$;
for $t = T, \dots, 1$ do
    Predict $\hat{\epsilon} = \epsilon_\theta(R_t, t)$;
    Compute $\hat{R}_0 = R_t \exp\left(-\sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}\right)$;
    Update $R_{t-1} = \exp\left(\sqrt{1-\bar{\alpha}_{t-1}}\,\hat{\epsilon}\right)\hat{R}_0$; end
    return $R_0$;
```