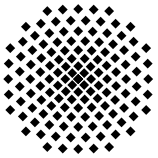


University of Stuttgart
Institute for Signal Processing and System Theory
Professor Dr.-Ing. B. Yang

Master's Thesis D1475

6 DoF Pose Estimation with Diffusion Models from a Single RGB-D Image

Schätzung der 6-DoF-Pose mit Diffusion Modelle aus einem einzelnen RGB-D-Bild

Author: Haoqing Wu

Date of work begin: 01.06.2023
Date of submission: 30.11.2023

Supervisor: Junwen Huang

Keywords: Deep Learning, Computer Vision, 6 DoF Pose Estimation, Diffusion Probabilistic Models

6 DoF pose estimation, a crucial aspect in the realms of computer vision and robotics, has witnessed considerable attention, particularly with the advent of deep learning methodologies. Recent years have seen the emergence of diverse techniques, encompassing RGB-based methods, depth-focused approaches and RGB-D fusion, aimed at enhancing the accuracy of pose estimation. However, challenges persist in scenarios involving textureless objects, occlusion, and cluttered backgrounds. This thesis introduces a pioneering pose estimation method founded on diffusion models, which have demonstrated notable success in image synthesis tasks. The objective is to leverage the strengths of diffusion models, such as their scalability to large-scale data and the diversity of generation, to address the complexities inherent in 6 DoF pose estimation. A comprehensive diffusion framework is proposed, utilizing a single RGB-D image to generate multiple hypotheses for the 6 DoF pose of an object. The proposed method is rigorously evaluated on real-world datasets featuring noise and occlusion. Results indicate that our approach achieves comparable performance with state-of-the-art methods, highlighting its effectiveness in challenging scenarios. This study not only contributes a novel perspective to pose estimation but also underscores the potential of diffusion models in advancing the state of the art in this critical domain.

Schätzung der 6-DoF-Pose, ein entscheidender Aspekt in den Bereichen Computer Vision und Robotik, hat erhebliche Aufmerksamkeit erfahren, insbesondere mit dem Aufkommen von Deep Learning Methoden. In den letzten Jahren sind verschiedene Techniken aufgetaucht, darunter RGB-basierte Methoden, auf Tiefen fokussierte Ansätze und RGB-D Fusion, die darauf abzielen, die Genauigkeit der Schätzung zu verbessern. Herausforderungen bleiben jedoch in Szenarien mit strukturlosen Objekten, Okklusion und überladenem Hintergrund bestehen. Diese Arbrit stellt eine bahnbrechende Pose-Schätzungsmethode auf Grundlage von Diffusionsmodellen vor, die in Bildsyntheseaufgaben bemerkenswerte Erfolge gezeigt haben. Das Ziel besteht darin, die Stärken der Diffusionsmodelle, wie ihre Skalierbarkeit für große Datenmengen und die Vielfalt der Generierung, zu nutzen, um den in der 6-DoF-Pose-Schätzung inhärenten Komplexitäten zu begegnen. Es wird ein umfassender Diffusionsrahmen vorgeschlagen, der ein einzelnes RGB-D-Bild verwendet, um mehrere Hypothesen für die 6-DoF-Pose eines Objekts zu generieren. Die vorgeschlagene Methode wird sorgfältig an realen Datensätzen mit Rauschen und Okklusion evaluiert. Die Ergebnisse zeigen, dass unser Ansatz vergleichbare Leistungen mit modernsten Methoden erreicht, was seine Effektivität in anspruchsvollen Szenarien unterstreicht. Diese Studie trägt nicht nur eine neue Perspektive zur Pose-Schätzung bei, sondern betont auch das Potenzial der Diffusionsmodelle zur Weiterentwicklung des State-of-the-Art in diesem kritischen Bereich.

Contents

List of Acronyms

***k*-NN**	*k*-Nearest Neighbors
1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
6 DoF	six degrees of freedom
AdaLN	Adaptive Layer Normalization
Adam	Adaptive Moment Estimation
ADD	Average Distance
ADD-S	Average Distance for Symmetrical Objects
ALD	Annealed Langevin Dynamics
AR	Average Recall
AR*	Augmented Reality
BOP	Benchmark for 6D Object Pose Estimation
CA	Cosine Annealing
CAD	Computer-Aided Design
CAS	Consistent Annealed Sampling
CD	Chamfer Distance
CDM	Cascaded Diffusion Models
CNN	Convolutional Neural Network
CSM	Concrete Score Matching
CV	Computer Vision
D3PM	Discrete Denoising Diffusion Probabilistic Models
DDIM	Denoising Diffusion Implicit Models
DDPM	Denoising Diffusion Probabilistic Models
DDSS	Differentiable Diffusion Sampler Search
DF	DenseFusion
ELBO	Evidence Lower Bound
EMA	Exponential Moving Average
GAN	Generative Adversarial Network
GICP	Generalized Iterative Closest Point
ICP	Iterative Closest Point

IDM Implicit Diffusion Models

LDM Latent Diffusion Model
LM LINEMOD
LMO LINEMOD-Occlusion
LN Layer Normalization
lr learning rate
LSGM Latent Score-based Generative Model

MLP Multilayer Perceptron
MRP Modified Rodrigues Parameters
MSE Mean Squared Error
MSPD Maximum Symmetry-Aware Projection Distance
MSSD Maximum Symmetry-Aware Surface Distance

NeRF Neural Radiance Fields
NICP Normal Iterative Closest Point
NLP Natural Language Processing

ODE Ordinary Differential Equation
ORB Oriented FAST and Rotated BRIEF

P*n*P Perspective-*n*-Point
PBR Physically Based Rendering
PDF Probability Density Function

R-CNN Region-based Convolutional Neural Network
RANSAC Random Sample Consensus
RDM Riemannian Diffusion Model
RGB red, green, blue
RGB-D red, green, blue, depth
RSGM Riemmanian Score-based Generative Model

SDE Stochastic Differential Equation
SG Stop-Gradient
SGM Score-based Generative Model
SIFT Scale-Invariant Feature Transform
SR3 Super-Resolution via Repeated Refinement
SURF Speeded Up Robust Features
SVD Singular Value Decomposition

VAE Variational Autoencoder
VDM Variational Diffusion Models
ViT Vision Transformer
VQ-VAE Vector Quantized Variational Autoencoder
VSD Visible Surface Discrepancy

List of Symbols

Constants

α_t, β_t	Variance schedule of diffusion models
$\bar{\alpha}_t, \bar{\beta}_t$	Accumulated variance schedule of diffusion models
η	Hyperparemeter that controls the sampling stochasticity
τ	Misalignment tolerence of VSD
θ_e	Threshold of the recall rate calculation
B, b	Batch size
d	Feature dimension
d_k	Dimension of the key vector
I	Identity matrix
n_{nb}	Number of neighbors
T	Diffusion steps

Distributions

$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
$\mathcal{U}(0, T)$	Uniform distribution between 0 and T
$h(\mathbf{T}\|y)$	Distribution of the transformation hypotheses conditioned with the input
$p_\theta(\mathbf{x}_{t-1}\|\mathbf{x}_t)$	Estimated reverse diffusion process between each time step
$q(\mathbf{x}_{t-1}\|\mathbf{x}_t)$	Reverse diffusion process between each time step
$q(\mathbf{x}_t\|\mathbf{x}_{t-1})$	Forward diffusion process between each time step

Functions

$\lambda(t)$	Weighting function of SDE
$\mathbb{E}(\cdot)$	Expectation of a random variable
$\mathbf{w}$	Brownian motion

$\odot$	Element-wise multiplication
D_F	Fisher divergence
f_{GS}	Gram-Schmidt-like process
g_{GS}	Reverse Gram-Schmidt-like process
$H(a, b)$	Cross-entropy function
$KL(P \parallel Q)$	Kullback-Leibler divergence
L	Loss function
$N(\cdot)$	Normalization function
$V(D, G)$	Value function of GAN

Networks

D	Discriminator of GAN
G	Generator of GAN
g_{θ_s}	Teacher network in knowledge distillation
g_{θ_t}	Teacher network in knowledge distillation

Fields

$\mathbb{R}$	Real number set
$\mathbb{SE}(3)$	3D special Euclidean group
$\mathbb{SO}(3)$	3D special orthogonal group

Variables

$\alpha_{1,2}, \beta_{1,2}, \gamma_{1,2}$	Learnable parameters of AdaLN
ϵ	Random noise with normal distribution
ϵ_θ	Estimated noise
ϵ_t	Random noise with normal distribution at time step t
$\hat{\mathbf{P}}, \bar{\mathbf{P}}$	Estimated pose and ground truth pose
$\hat{S}$	Reconstruct point cloud in completion task
$\hat{S}, \bar{S}, S_I$	Estimated, ground truth and test image distance map
$\mu_\theta(\mathbf{x}_t, t)$	Estimated mean at time step t in tensor form
$\mathbf{\Sigma}_\theta(\mathbf{x}_t, t)$	Estimated variance at time step t in tensor form

$\tilde{\mu}_t$	Trackable mean at time step t in tensor form
$\mathbf{f}(\mathbf{x}, t)$	Drift term of SDE
$\mathbf{h}_i$	Output matrix of a single head attention
$\mathbf{q}$	Quaternion in vector form
$\mathbf{q}_i, \mathbf{k}_i, \mathbf{v}_i$	Query, key and value vectors of the attention mechanism
$\mathbf{R}$	3D rotation matrix
$\mathbf{r}$	Representation of 3D rotation in vector form
$\mathbf{s}_\theta$	Estimated score of SGM
$\mathbf{T}$	3D transformation matrix
$\mathbf{t}$	3D translation vector
$\mathbf{T}_r$	Refined transformation
$\mathbf{T}_t$	Transformation at time step t
$\mathbf{W}^O$	Output matrix of the attention mechanism
$\mathbf{W}^{q,k,v}$	Query, key and value matrices of the attention mechanism
$\mathbf{x}_t$	Data at time step t of diffusion models
$\mathbf{y}_i$	Output vector of the scaled dot-product attention
$\mathcal{T}_B$	Subset of the whole timesteps with the batch size B
$\Phi_{x,y,z}$	Euler angles in three orthogonal directions
$\tau_{1,\ldots,s}$	Subset of the complete diffusion steps in DDIM
θ_s	Student parameters in knowledge distillation
$\tilde{\beta}_t$	Trackable variance at time step t
E	Triangles of a 3D mesh
e	One triangle of a 3D mesh
e_{metric}	Error of the evaluation metric
e_p	Position embedding
e_t	Time embedding
f_{2D}	2D global feature
f_{3D}	3D global feature

$g(t)$	Diffusion coefficient of SDE
M	Mesh of an object
O	Object set
q	Quaternion
$q_{w,x,y,z}$	Quaternion coefficients
r_{ij}	Elements in 3D rotation matrix
S	Reference point cloud in completion task
t	Time step variable
$t_{x,y,z}$	Translation scalars in three orthogonal directions
V	Vertices of a 3D mesh
v	One vertice of a 3D mesh
y	Conditional input of conditional diffusion models
z	Latent feature

1. Introduction

1.1. Motivation

In recent years, the field of Computer Vision (CV) has undergone a significant transformation, fueled by the rapid advancement and integration of cutting-edge technologies such as deep learning and the utilization of three-dimensional (3D) data. This transformation has substantially enhanced machines' capabilities to comprehend and interact with the physical world. Within the realm of computer vision, six degrees of freedom (6 DoF) pose estimation, tasked with determining the position and orientation of objects in 3D space, has garnered considerable attention due to its pivotal role in diverse applications, including augmented reality, robotics, automation, autonomous driving, object manipulation, and quality control.

6 DoF pose estimation necessitates the integration of techniques from various fields. Learning-based methods are employed to extract features from input data and estimate poses, while non-learning-based algorithms process and refine these estimations. To accommodate diverse data inputs and enhance accuracy and reliability, the task involves processing data from both two-dimensional (2D) and 3D domains. Scaling to multi-object or multi-view scenarios, commonplace in robotics and autonomous driving, and considering time-series data introduce additional challenges and opportunities. Pose estimation can extend beyond its core function to object tracking in videos or even predicting future object poses. The complexity of these tasks intensifies the challenge of pose estimation in real-world applications, injecting vitality into ongoing research in this field.

Diffusion models, initially introduced by Sohl-Dickstein et al. 2015 [1] and recently refined by Ho et al. 2020 [2], belong to a class of generative models renowned for their capacity to generate high-quality data from random noise. Demonstrating notable success in various image synthesis tasks, diffusion models have outperformed previous methods, extending their influence beyond CV to fields such as natural language processing, temporal data modeling, and multi-model learning. Despite this success, the integration of diffusion models into 6 DoF pose estimation is at an early stage, with limited existing works. The untapped potential of these models in the realm of 6 DoF pose estimation underscores a promising avenue for exploration and advancement in the field.

The widespread use of RGB-D images in computer vision tasks stems from the increased prevalence of RGB-D sensors and the invaluable depth information they provide. Research consistently indicates that leveraging depth information alongside RGB data leads to improvements in network performance, surpassing the capabilities of methods reliant solely on RGB data. Given this, it is of great significance to evaluate the effectiveness of diffusion models when applied to RGB-D data within the context of our pose estimation task. This evaluation will shed light on the model's capability to harness the additional depth information for enhanced pose estimation accuracy and robustness.

1.2. Aims and Objectives

This thesis endeavors to explore the feasibility of integrating diffusion models into the pose estimation task, harnessing the advantages inherent in diffusion models within the 2D domain and extending their utility to the 3D domain. These advantages encompass the capability of handling large-scale data and the diversity of the generated data. To this end, we first review the related works about the 6 DoF pose estimation and diffusion models. Then design the diffusion network architecture with the single RGB-D image as input, which contains the diffusion backbone, 2D feature extractor and 3D feature extractor. Finally, we evaluate the performance of the diffusion models on the 6 DoF pose estimation task and compare it with the preceding methods characterized by varying architectural configurations.

1.3. Structure of the Thesis

The thesis is organized into seven chapters and the structure of the thesis is as follows:

Chapter 2 - Background This chapter gives an overview of the 6 DoF pose estimation and diffusion models. The 6 DoF pose estimation is introduced with the definition, representation of the pose and typical applications. The challenges of the pose estimation task are also discussed. The diffusion models are introduced with an overview of other generative models and the basic theory behind them together with the applications.

Chapter 3 - Related Works In this chapter, some previous works are reviewed which are related to the 6 DoF pose estimation and diffusion models. Different architectures of the pose estimation models are hierarchically introduced and discussed.

Chapter 4 - Methodology The proposed method and models are introduced in this chapter. The structure of the pose diffusion architecture is presented together with each module and algorithm of the model in detail.

Chapter 5 - Experiments The experiments are conducted in this chapter. The datasets and evaluation metrics are introduced first. We describe the setup of the training and evaluation for each model in this work. Then the qualitative and quantitative results of the experiments, the comparison with other methods as well as the ablation study are presented.

Chapter 6 - Discussion The results of the experiments are discussed here. The advantages and disadvantages of the proposed method are analyzed and the possible improvements are also discussed.

Chapter 7 - Conclusion Finally, we summarize the thesis and the contribution from this work and give an outlook of future work.

2. Background

2.1. 6 DoF Pose Estimation

2.1.1. Definition

The term "Six Degrees of Freedom (6DoF) pose" pertains to the six fundamental ways a rigid body can move within three-dimensional space. Specifically, it encompasses the freedom of movement in three orthogonal directions, referred to as translations, as well as the ability to rotate about three orthogonal axes, termed rotations. This concept holds significant relevance, particularly in industrial and automotive domains, where it is instrumental in measuring and analyzing the spatial properties of objects. The comprehensive understanding of an object's movement and orientation provided by the 6DoF pose is crucial for various applications within these fields.

In domain of computer vision and robotics, 6 DoF pose estimation is a fundamental task that aims to estimate the 3D translation $\mathbf{t} = (t_x, t_y, t_z)$ and rotation $\mathbf{R} = (\Phi_x, \Phi_y, \Phi_z)$ of an object related to a canonical coordinate system using the sensor input, such as RGB or RGB-D data.The object M is typically a known 3D Computer-Aided Design (CAD) model, consisting of a set of vertices $V = \{v_1, ..., v_N\}$, with $v_i \in \mathbb{R}^3$ and $V \in \mathbb{R}^{3\times N}$ and triangles $E = \{e_1, ..., e_M\}$, with $e_i \in \mathbb{R}^3$ and $E \in \mathbb{R}^{3\times M}$ connecting the vertices. Furthermore, if the query image is a multi-object scenario with N objects $O = \{M_1, ..., M_N\}$, we need to detect and estimate the pose of each object M_i in the image [3]. The following figure 2.1 shows a general structure of the learning-based 6 DoF pose estimation.

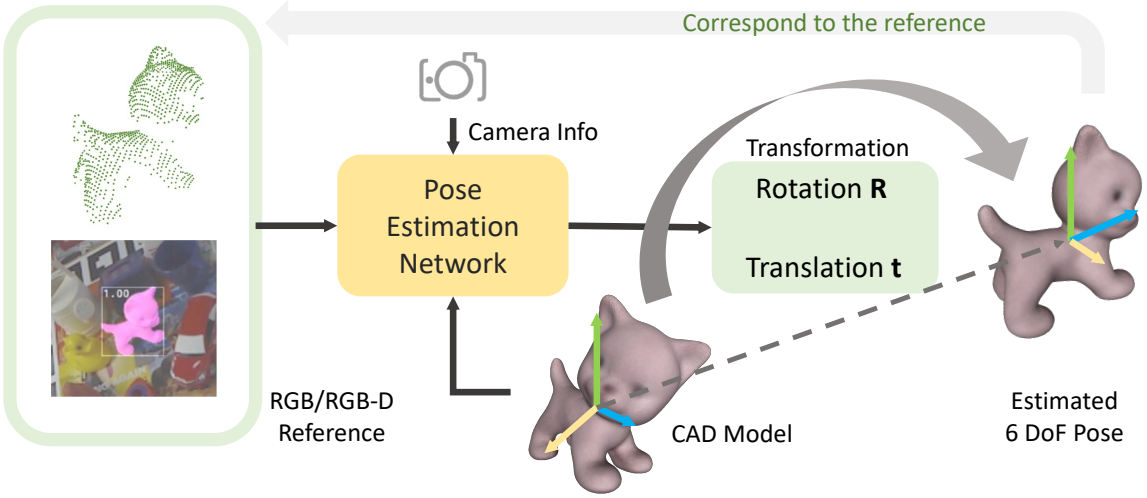


Figure 2.1.: Overview of the general proposal towards 6 DoF pose estimation.

2.1.2. Representing 6 DoF Pose

6 DoF pose can be treated seperately as 3D translation and 3D rotation. The 3D translation is simply represented by 3 scalars along the X, Y, and Z axis of the canonical coordinate system. We can use either the deep learning methods to estimate the depth and the corresponding 2D projection from RGB images or even get the depth information fused from RGB-D data [4]. After that, the object can be shifted back to the camera coordinate system by adding a translation vector to the object vertices V:

$$V^{'} = V + \mathbf{t} \tag{2.1}$$

where $V^{'}$ represents the shifted object vertices. Similarly, the 3D rotation can be represented by 3 rotation matrices around the X, Y and Z axes. The final transformation of rotation can be achieved by multiplying the rotation matrices $\mathbf{R}_i$ defined by Euler angles Φ_i with $i \in \{X, Y, Z\}$. Rotation around X axis is defined as:

$$V^{'} = \mathbf{R}_X(\Phi_x)V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\Phi_x) & -sin(\Phi_x) \\ 0 & sin(\Phi_x) & cos(\Phi_x) \end{bmatrix} V \tag{2.2}$$

Rotation matrix $\mathbf{R}_Y$ and $\mathbf{R}_Z$ can be defined repectively with:

$$\mathbf{R}_Y(\Phi_y) = \begin{bmatrix} cos(\Phi_y) & 0 & sin(\Phi_y) \\ 0 & 1 & 0 \\ -sin(\Phi_y) & 0 & cos(\Phi_y) \end{bmatrix} \tag{2.3}$$

$$\mathbf{R}_Z(\Phi_z) = \begin{bmatrix} cos(\Phi_z) & -sin(\Phi_z) & 0 \\ sin(\Phi_z) & cos(\Phi_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

The rotation matrix $\mathbf{R}$ can be obtained by multiplying the three rotation matrices $\mathbf{R}_X$, $\mathbf{R}_Y$ and $\mathbf{R}_Z$ together, but changing the order of the multiplication will result in different rotation matrix. The common order is defined as $Z - Y - X$ order, which means the rotation around the X axis is performed first, then the Y axis and finally the Z axis. All possible rotations in 3D Euclidean space establish a natural manifold known as special orthogonal group $\mathbb{SO}(3)$ [5].

Togather with the translation vector $\mathbf{t}$, the 6 DoF pose can be represented by a 4x4 transformation matrix $\mathbf{T}$ as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}(3) \tag{2.5}$$

The partitioned transformation matrix with 3x3 rotation matrix $\mathbf{R}$ and a column vector $\mathbf{t}$ that represents the translation is also called the homogeneous representation of a transformation. All possible transformation matrices of this form generate the special Euclidean group $\mathbb{SE}(3)$:

$$\mathbb{SE}(3) = \{\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} | \mathbf{R} \in \mathbb{SO}(3), \mathbf{t} \in \mathbb{R}^3\} \tag{2.6}$$

Normally, we use the shift in 3 orthogonal directions in the cartesian coordinate system to represent the translation. However, there are some different ways to represent the rotation.

One simple method to represent the rotation is to use the Euler angles ϕ, θ and ψ which are also marked as roll angle (around X axis), pitch angle (around Y axis) and yaw angle (around Z axis) respectively. The main drawback of this representation is the gimbal lock problem, which means the rotation around two axes will cause the rotation around the third axis to be the same as the rotation around the first axis.

An alternative representation of 6 DoF pose is a 4-dimensional vector that consists of translation and rotation quaternion which has a compacter form:

$$\mathbf{r} = (q_w, q_x, q_y, q_z)^T \tag{2.7}$$

Where the quaternion q is defined as:

$$q = q_w + q_x i + q_y j + q_z k \quad \text{with} \quad i^2 = j^2 = k^2 = ijk = -1 \tag{2.8}$$

Normally, regressing the rotation matrix directly is not a common choice since the same rotation can be achieved via different combinations of Euler angles. And the unit quaternion form is in many cases preferred because it can ensure the uniqueness by restricting the quaternion on the upper hemisphere of $q_w = 0$ plane and can also guarantee a gimbal-lock free rotation in $\mathbb{SO}(3)$ [6].

Another representation that can be considered is called Modified Rodrigues Parameterss (MRPs) which is a 3-dimensional vector $\mathbf{r} = (r_1, r_2, r_3)^T$. They are triplets in $\mathbb{R}^3$, bijectively and rationally mapped to quaternions through stereographic projection [7]. The MRP vector $\mathbf{r}$ is defined as:

$$\mathbf{r} = \frac{\mathbf{q}}{1 + q_w} = \frac{1}{1 + q_w}(q_x, q_y, q_z)^T \tag{2.9}$$

where $\mathbf{q}$ is the quaternion representation of the rotation. The MRP vector $\mathbf{r}$ is also a unit vector, the advantage of using MRP is that a random assignment of the vecter within the unit sphere will always result in a valid rotation. That property makes this representation more robust in the forward and reverse process of the diffusion pipeline.

Zhou et al. 2019 [8] proposed a novel representation of rotation called 6D continuous rotation representation. The mapping from the rotation matrix to the 6D representation with generally n dimensional rotation is defined as:

$$g_{GS}\left(\begin{bmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{bmatrix}\right) = \begin{bmatrix} | & & | \\ a_1 & \cdots & a_{n-1} \\ | & & | \end{bmatrix} \tag{2.10}$$

The mapping from $\mathbb{SO}(3)$ to the 6D representation can be simplified as:

$$g_{GS}\left(\begin{bmatrix} | & | & | \\ a_1 & a_2 & a_3 \\ | & | & | \end{bmatrix}\right) = \begin{bmatrix} | & | \\ a_1 & a_2 \\ | & | \end{bmatrix} \tag{2.11}$$

The reverse mapping follows the Gram-Schmidt-like process:

$$f_{GS}\left(\begin{bmatrix} | & | \\ a_1 & a_2 \\ | & | \end{bmatrix}\right) = \begin{bmatrix} | & | & | \\ b_1 & b_2 & b_3 \\ | & | & | \end{bmatrix} \tag{2.12}$$

$$b_i = \begin{bmatrix} \begin{cases} N(a_1) & \text{if } i = 1 \\ N(a_2 - (b_1 \cdot a_2)b_1) & \text{if } i = 2 \\ b_1 \times b_2 & \text{if } i = 3 \end{cases} \end{bmatrix}^T \tag{2.13}$$

Here $N(\cdot)$ is the normalization function. It was proved by the sanity tests introduced in the paper that this kind of representation is an efficient way for training in deep neural networks, compared with quaternions and Euler angles that are not continuous and have singularities.

2.1.3. Applications

6 DoF pose estimation is a central technology that can be the critical part of many computer vision applications such as augmented Augmented Reality (AR*), robotics, 3D scene understanding and autonomous driving. In this section, we will introduce some typical applications of 6 DoF pose estimation.

Augmented Reality

AR* applications use 6 DoF pose estimation to accurately place the virtual objects in the real world. Precise estimation and quick inference of the pose guarantee an immersive and interactive experience which is the direction of the development of AR* applications [9]. Furthermore, 6 DoF pose estimation can also be utilized to track real-world objects, enabling more natural interactions.

Robotics

6 DoF pose estimation helps robots understand the scene so that the grasping and manipulation of objects can be achieved. In the field of medical robotics, it can be used to track the surgical instrument or a patient's body part [10]. In manufacturing, robots use the estimated pose to identify, sort and assemble the objects in the field like automatic logistic sorting and manufacturing lines.

3D Scene Understanding

In order to register the 3D objects into the scene or reconstruct the 3D environment from 2D images or 3D point clouds, 6 DoF pose estimation is required. The alignment of the 3D objects or 3D scenes is realized by estimating the rigid transformation using methods like correspondence matching [11] or direct transformation estimation [12] follows the ideas of Iterative Closest Point (ICP) [13].

Autonomous Driving

Autonomous driving is also a cross-domain topic that requires many different technologies to work together. A well-estimated pose of the vehicle inside the scene is the basis of many other subtasks of autonomous driving such as collision avoidance, trajectory planning and so on. Subtle errors in the pose estimation may lead to fatal consequences [14] because the vehicle moves normally at high speed and the heading direction causes a large deviation in a long distance considering also the reaction time of the vehicle.

2.1.4. Challenges

6 DoF pose is widely used in many applications and has become a popular research topic of computer vision in recent years. However, solving this problem is not trivial and even challenging in many cases.

The first constraint would be the auto-occlusion or symmetries of the object since the object cannot be clearly and unequivocally observed from all angles [15]. The auto-occlusion means that the object itself is partially occluded by other parts of the object such as LINEMOD-Occlusion (LMO) dataset [16]. This is common in many real-world objects such as tables or chairs. The symmetry of the object means that the object has the same appearance from different angles, which will cause ambiguity in the estimation such as the T-LESS dataset [17]. Imagining an image of a mug with the handle hidden behind it, it is hard to tell the orientation of the mug without the handle.

Textureless object is also a challenge for 6 DoF pose estimation since many methods rely not only on the geometry of the object but also on the texture. It is hard for RGB-only methods [18] or keypoint-based method [19] to extract enough local features if the object is completely textureless.

Another difficulty is the domain gap between the training and testing data. Normally, the training data consists of synthetic CAD models and images which are clean and annotated with the ground truth pose to have precise supervision. But lacking the information of the real world, for example, lighting and occlusion, the model trained on the synthetic data cannot generalize well to the real world data. Some dataset provides real-world data or 3D rendered images which can reduce the domain gap to some degree [20], but the noise and invalid training samples still confuse the model.

If facing the multi-object scenario, which is common in applications like robotics and autonomous driving, the unknown number and type of objects will increase the difficulty of pose estimation for each object in the scene.

2.2. Diffusion Models

2.2.1. Generative Models

One of the most fascinating and distinctive features of the human brain is the ability to create or imagine objects that do not immediately exist in reality. Humans can spontaneously learn

the underlying properties of the world and generate hypotheses about the future. This procedure is similar to supervised learning and reinforcement learning with little amount of labeled data but generalizes very well to many unseen scenarios and has a high level of robustness [21].

In order to achieve a similar ability of the generative process from the human brain, many generative models have been proposed in recent years, to not synthesize the unseen data but to recover or modify the seen data with given constraints. Some of the most popular generative models are introduced below.

Generative Adversarial Networks

Generative Adversarial Networks (GANs) [22] is a smart idea to train a generative model by playing a min-max game between two neural networks. The generator G is trained to generate data that is indistinguishable from the real data, while the discriminator D is trained to distinguish the real data from the fake data generated by G. The training process can be formulated as the value function $V(D, G)$, and for the classification objective using cross-entropy loss, the optimization problem can be written as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.14}$$

The generator is optimized to maximize the probability that the discriminator will classify the generated data as real, which explains the word "adversarial" in the name of GAN.

GANs have shown great success in many applications such as generating high-resolution images that are difficult to distinguish from the real ones and the ability to learn the complicated distributions. However, the main challenge of GANs is the instability of the training process, which increases the difficulty of training and tuning the model. It will sometimes suffer from the mode collapse problem, where the generator only learns to generate a subset of the data distribution [23].

Some work has been done to solve these problems. For example, Wasserstein GANs [24] uses a different loss function to stabilize the training process and avoid the mode collapse. Spectral Normalization [25] is another method to stabilize the training process by constraining the Lipschitz constant of the discriminator.

Variational Autoencoders

Variational Autoencoders (VAEs) [26] is another popular generative model that is based on the encoder-decoder architecture. It allows the model to learn the latent representation of the input data and generate new data from the latent space. The encoder E is trained to map the input data x to the latent space z with a distribution $q(z|x)$, while the decoder D is trained to reconstruct the input data from the latent space z with a distribution $p(x|z)$. The training process can be formulated as:

$$\min_{E,D} \mathbb{E}_{x \sim p_{data}(x)}[\mathbb{E}_{z \sim q(z|x)}[\log p(x|z)]] - KL(q(z|x)||p(z)) \tag{2.15}$$

The first term is the reconstruction loss, which is the negative log-likelihood of the input data x given the latent representation z. The second term is the regularization term, which is the

Kullback-Leibler divergence between the latent distribution $q(z|x)$ and the prior distribution $p(z)$. With the regularisation term, we prevent the model from encoding the input data far apart in the latent space, which will cause the model to generate unrealistic data.

The reparameterization trick [27] is introduced afterward to make the stochastic part of the loss function which is the latent representation z differentiable so that the model can be trained with backpropagation. The latent representation z is sampled from a distribution $q(z|x)$, which is normally a Gaussian distribution. The trick constructs the random variable z into the following expression where ϵ is a random variable sampled from a standard Gaussian distribution.

$$z \in \mathcal{N}(\mu, \sigma^2) \longrightarrow z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1) \tag{2.16}$$

VAEs allow us to easily sample the latent representation z from the prior distribution $p(z)$ and generate novel data from the decoder D. It can also be used to make data compression and denoising, which is the main application of autoencoders. Since the flexibility and the robustness of VAEs, It is widely used in many applications such as image manipulation, text generation and speech synthesis.

Normalizing Flows

Normalizing flows [28] are a family of generative models with tractable marginal likelihood which can not be achieved with VAEs. A normalizing flow is a transformation of a simple distribution into a more complex distribution by a series of invertible and differentiable mappings. By repeating the rule of transformation, the initial probability density "flows" through the sequence of invertible mappings and becomes a valid distribution.

The basic rule for the transformation of densities considers an invertible, smooth mapping $f : \mathbb{R}^D \to \mathbb{R}^D$, with inverse $f^{-1} = g$. Transforming a random variable z with distribution $q(z)$ through f results in a random variable $z^{'} = f(z)$ has a distribution:

$$q(z^{'}) = q(z) \left| \det \frac{\partial f^{-1}}{\partial z^{'}} \right| = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} \tag{2.17}$$

The last term is the Jacobian determinant of the transformation f, which is the determinant of the matrix of partial derivatives of f with respect to z. Given a chain of invertible mappings $f_1, ..., f_K$, the transformation of the random variable z through the sequence of mappings and the density $q_K(z)$ can be written as:

$$z_K = f_K \cdot ...f_2 \cdot f_1(z_0) \tag{2.18}$$

$$lnq_K(z_K) = lnq_0(z_0) - \sum_{k=1}^{K} ln \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right| \tag{2.19}$$

The path of the transformation can be seen as a flow of the probability density from the initial distribution $q_0(z_0)$ to the final distribution $q_K(z_K)$. If the length of the normalizing flow tends to infinity, the model becomes an infinitesimal flow which is described by a differential equation.

Normalizing flows provides a flexible framework for modeling complex distributions, which is difficult to achieve with previous generative models. However, the samples that are generated through flow-based models are not as realistic as the samples from GANs or VAEs, and the data will be projected into also high dimensional space, which is hard to interpret.

Diffusion Models

Diffusion models are a new class of state-of-the-art generative models that can synthesize high-quality images in recent years. The representative one, which is the Denoising Diffusion Probabilistic Models (DDPM) was initialized by Sohl-Dickstein et al 2015 [1] and proposed recently by Ho. et al 2020 [2].

A diffusion probabilistic model (diffusion model), inspired by the nonequilibrium thermodynamics, is a parameterized Markov chain trained using variational inference to produce samples from a given target distribution after finite steps. The basic idea behind diffusion models is trivial. Given an input data x_0, we first gradually add Gaussian noise to it and finally get a sequence of noised data $x_1, ..., x_T$, which we call it forward process. Afterward, a neural network is trained to recover the original data by estimating the noise and reversing the forward process, which we call it sampling process or reverse process.

Figure 2.2 shows an overview of four different types of generative models that we mentioned in this section. Unlike VAE and the flow-based model, the diffusion model is based on the Markov chain and the latent variable has high dimensionality which has the same size as the input data. We recover the original data by estimating the noise and reversing the forward process iteratively, which ensures the high quality of the generated data.

The great success of some architecture using the diffusion model such as GLIDE [29] and DALLE-2/3 [30] has shown the potential of the diffusion model in the field of generative models. The advantage of the diffusion model is that it is large-scale, flexible and offers high-quality samples. With the tradeoff of the relatively longer training time and inference time because of its 2-phases architecture, it can synthesize the highest-quality images than other generative models. This potential motivates us to apply the diffusion model also to 3D domain and the related tasks.

2.2.2. Theory and Fundamentals

In this section, we will introduce the details of the diffusion model, the mathematical background in the forward process and the sampling process, and the conditional diffusion model. The extended version of the classic DDPM will also be briefly introduced.

Forward Process

Given an input data $\mathbf{x}_0$ from the target data distribution $q(\mathbf{x})$, we first define a forward process that gradually adds Gaussian noise to $\mathbf{x}_0$ with variance $\beta_t \in (0, 1)$ at each step t and finally get a sequence of noised data $\mathbf{x}_1, ..., \mathbf{x}_T$. At each step t, we have the new data $\mathbf{x}_t$ with the conditional distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ defined as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2.20}$$

where $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a normal distribution with mean $\sqrt{1-\beta_t}\mathbf{x}_{t-1}$ and variance $\beta_t\mathbf{I}$. Thus, we can derive the posterior distribution from the input data $\mathbf{x}_0$ to $\mathbf{x}_T$ in a tractable way:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.21}$$

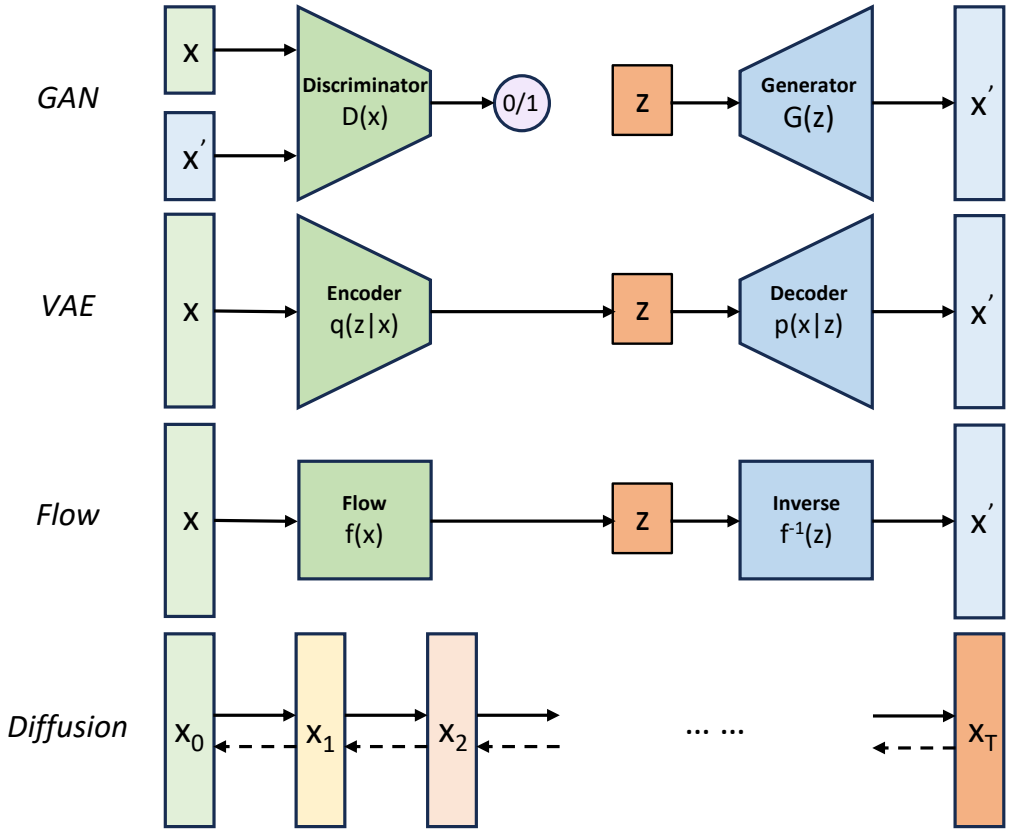


Figure 2.2.: Overwiew of different types of generative models.

Our goal is to track the noised data at an arbitrary step t with a closed-form posterior distribution $q(\mathbf{x}_t|\mathbf{x}_0)$. So the reparameterization trick is introduced so that we don't need to calculate the $\mathbf{x}_t$ iteratively from $t = 0$.

Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ with Gaussian noise $\boldsymbol{\epsilon}_0, ..., \boldsymbol{\epsilon}_{t-2}, \boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we can simplify the noised the data $\mathbf{x}_t$ in such a recursive way:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1}
\end{aligned}$$

Notice that when we merge two Gaussian distributions with different variance, $\mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I})$ and $\mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{I})$, the new merged distribution is $\mathcal{N}(\mathbf{0}, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$. So we can merge the second and third term in the equation above where $\bar{\boldsymbol{\epsilon}}_{t-2}$ is the new Gaussian and get:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= ... \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}
\end{aligned} \tag{2.22}$$

Finally, we can represent the sample $\mathbf{x}_t$ with the following distribution:

$$\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.23}$$

where α_t and $\bar{\alpha}_t$ can be precomputed for any arbitrary step t from β_t. The variance hyperparameter β_t is normally chosen as a linear, quadratic or cosine schedule. The original design of DDPM used a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$ which is also commonly used in other diffusion models.

Reverse Process

The purpose of the reverse process is to reverse the forward process above and recover the original data $\mathbf{x}_0$ from a random Gaussian noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Practically, the reverse conditional distribution is not directly tractable, because the computations involve the whole data distribution. Therefore, we need to train a model $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ to estimate the reverse conditional distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Since the variance β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ can be treated as Gaussian distribution, so does $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, which can be defined as follow:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{2.24}$$

Applying the estimated reverse conditional distribution for all timesteps we get:

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2.25}$$

The reverse conditional probability is only trackable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}) \tag{2.26}$$

With the help of Bayes' Rule and the properties of the Gaussian probability density function, we can prove that:

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \tag{2.27}$$

$$\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x_0} \tag{2.28}$$

Thanks to the reparemeterization trick, we can represent $\mathbf{x_0} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x_t} - \sqrt{1-\bar{\alpha}_t}\epsilon_t)$ from 2.22 and further simplify the expression of $\tilde{\boldsymbol{\mu}}$ as:

$$\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right) \tag{2.29}$$

Notice that such a setup of p and q is similar to VAEs, so we can optimize the negative log-likelihood using the variational bound:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{KL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\right] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] =: L
\end{aligned} \tag{2.30}$$

To make the lower bound L computable, the expression can be further rewritten after some manipulations in Appendix of [2] as:

$$L = \mathbb{E}_q \left[\underbrace{D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)||p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T} \underbrace{D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right] \tag{2.31}$$

Each term L_i with $i \in \{0, ..., T\}$ compares the forward and reverse conditional distributions at each timestep i and in closed form, where L_T is constant and can be ignored during training, L_0 is the reconstruction term and is learned using a separate decoder in the original model [31].

The second term L_{t-1} describes the difference of $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$ against the posteriors in forward process, which we need to learn during the training process. Replace $t-1$ with t and t with $t+1$ in the equation above in order to express it in a natural way, we use L_t in the following calculation.

Revisit the reverse process from 2.24, we need to train $\boldsymbol{\mu}_\theta$ to approximate $\tilde{\boldsymbol{\mu}}_t$ in 2.29, where ϵ_t can be reparameterized as the prediction from the input $\mathbf{x}_t$ at time step t. Finally, we can have the expression of the approximation of the mean:

$$\boldsymbol{\mu}_\theta = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) \tag{2.32}$$

The lost term L_t can be formulated using l_2 distance:

$$L_t = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{1}{2 \left\| \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) \right\|_2^2} \left\| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) \right\|^2 \right] \tag{2.33}$$

which can be simplified by ignoring the weighting term according to the original paper [2] as:

$$L_{simple} = \mathbb{E}_{t \sim [1,T], \mathbf{x}_0, \boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}_t, t) \right\|^2 \right] + C \tag{2.34}$$

where C is a constant term which not related to θ and can be ignored during training. And we the variance is not considered in the loss function and it is improved in the later research [32] to let the network also learn the covariance matrix $\boldsymbol{\Sigma}_\theta$.

An overview of the forward and reverse process using a cat image as an example is shown in figure 2.3.

Conditional Diffusion Model

Conditional diffusion, also called guided diffusion is very practical in many applications since we normally want to generate the data in a particular style, direction or distribution and not in an arbitrary way. Typical usage of conditional diffusion is to sample data from a given class or category, as well as text prompt, image prompt and so on.

Mathematically, condition means the prior distribution $p(\mathbf{x})$ is conditioned on a given input y. By modifying the equation 2.25, we get:

$$p_\theta(\mathbf{x}_{0:T}|y) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, y) \tag{2.35}$$

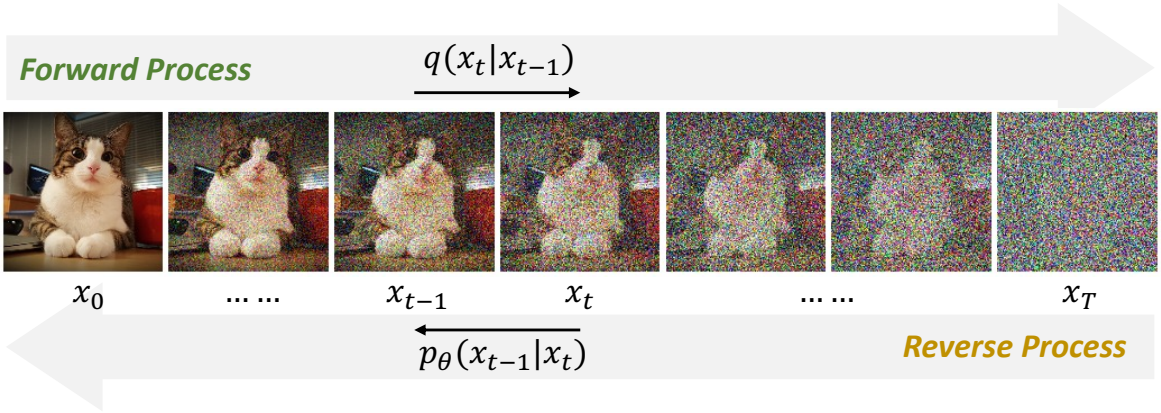


Figure 2.3.: Forward and reverse process of diffusion model using 2D image as example, cat images are adapted from [33].

Using the idea of the score-based generative model [34], we can train a score network for an unconditioned diffusion with the score function:

$$\mathbf{s}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) \tag{2.36}$$

Extend the score function with condition y, we can get the conditional score function after applying Bayes' Rule:

$$\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|y) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p_\theta(y|\mathbf{x}_t) \tag{2.37}$$

Based on the score function we can derive the conditional diffusion model with two variations, namely the classifier guidance and classifier-free guidance.

Classifier guidance is a method that balances the trade-off between mode coverage and sample fidelity post-training. It combines the score estimate of a diffusion model with the gradient of an image classifier, which requires training a classifier f_ϕ separate from the diffusion model and uses the gradients of the classifier as the guidance.

Without the separate classifier f_ϕ, it is still possible to let the conditional and unconditional score functions share the same network, which is called classifier-free guidance. The diffusion model is trained by randomly dropping the condition y during training. The result turns out to be that the conditional and unconditional score estimates are combined to attain a good tradeoff between quality and diversity [35].

The training and sampling algorithm of the conditional denoising diffusion probabilistic model can be summarized as the following algorithm 1 and 2.

Algorithm 1 Training of Conditional Diffusion Model

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, conditional input y
3: $t \sim$ Uniform($\{1, ..., T\}$)
4: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take a gradient descent step on $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t, y) \right\|^2$
6: **until** converged

Algorithm 2 Sampling of Conditional Diffusion Model

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T$ to 1 **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t, y)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

2.2.3. Applications

Computer Vision

The majority of the applications of diffusion models lie in the field of computer vision, including super-resolution, translation, inpainting and so on [36]. Diffusion models have shown a great performance in these 2D based manipulation tasks compared with other generative models such as GANs and VAEs.

Super-Resolution via Repeated Refinement (SR3) [37] and Cascaded Diffusion Models (CDM) [38] are two representative works in the field of super-resolution. They use either an iterative way or a concatenation of diffusion models to generate high-resolution images from low-resolution inputs. Figure 2.4 illustrates the process of cascaded super-resolution, which is taken from the project page of the SR3. Implicit Diffusion Models (IDM) for Continuous Super-Resolution [39] integrates an implicit neural representation in the decoding process.

Inpainting and image translation are also two popular image manipulation tasks with different conditional inputs. Typical works are RePaint [40], Palette [41] and Diffusion-based Image Translation using Disentangled Style and Content Representation [42].

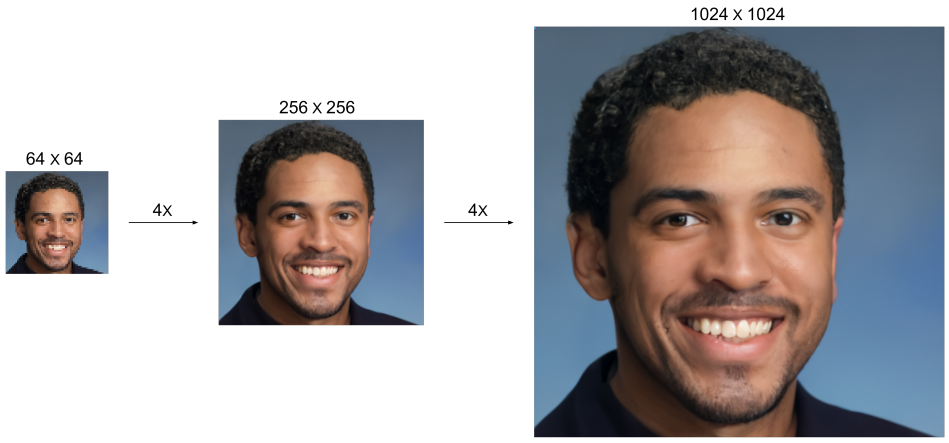


Figure 2.4.: Super resolution using diffusion model, image taken from the project page of SR3 [37].

In 3D domain, the diffusion model is also applied to the task of point cloud generation and completion. Luo et al.2021 [43] and Zeng et al.2022 [44] have presented the diffusion models

for point cloud generation by treating point clouds as particles in a thermodynamic system. Lyu et al.2022 [45] have proposed a coarse-to-fine point cloud completion diffusion model and also established a point-wise mapping between the output and ground truth. Figure 2.5 adapted from the paper shows the point cloud generation using diffusion architecture.

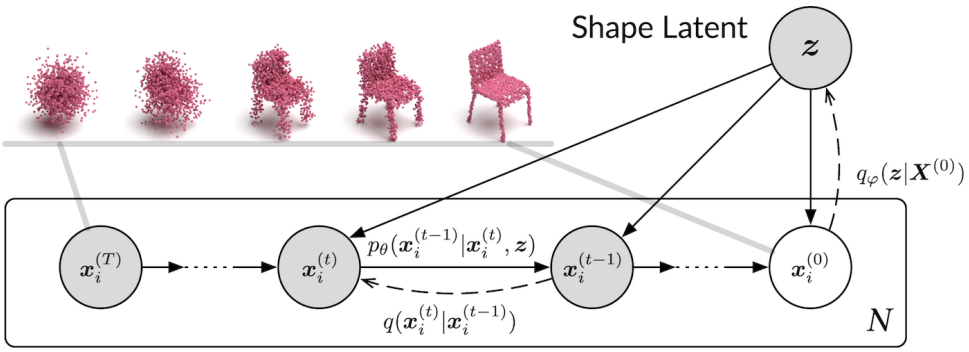


Figure 2.5.: Point cloud generation using diffusion model, image taken from [45].

Natural Language Processing

Natural Language Processing (NLP) has been dramatically developed in recent years. The iconic models like BERT [46], GPT series [47] and LLaMA [48] are all based on the Transformer architecture. However, some diffusion-based methods can generate text with high quality and diversity. In fact, diffusion models have been shown to have significant advantages over autoregressive models in terms of parallel generation, text interpolation, token-level controls such as syntactic structures and semantic contents, and robustness [49].

Discrete Denoising Diffusion Probabilistic Models (D3PM) [50], is a diffusion-like generative model for discrete data that generalizes the multinomial diffusion model [51], by going beyond corruption processes with uniform transition probabilities.

Diffusion-LM [52] proposes a non-autoregressive language model based on continuous diffusions, which iteratively denoises a sequence of Gaussian vectors into word vectors, yielding a sequence of intermediate latent variables, which makes it possible for simple, gradient-based methods to achieve complex control.

Multi-Modal Learning

Multi-modal learning is a field that combines different modalities such as text, image, video and audio. It tends to become the mainstream of future research in the field of machine learning because of the higher requirement of real-world applications.

Text-to-image generation is a typical task in this field. A common pipeline is to first train a prior model that can generate image embedding conditioned on a text prompt,e.g. CLIP [53]. Then we use the prior output as a condition to train a diffusion model to generate the final image. Famous works like Stable Diffusion [54] and DALLE-2 [30] followed this pipeline and achieved state-of-the-art results in text-to-image generation. Following Text-to-Image samples 2.6 are generated using Stable Diffusion.

"Watch assembling with cinematic style"

"Waterfall before sunset with watercolor style"

"Spaceship travelling in the universe with pixel style"

Figure 2.6.: Image synthesis with text prompt in different styles using Stable Diffusion [54].

ControlNet [55] attempts to control pre-trained large diffusion models to support additional semantic maps, like edge maps, segmentation maps, key points, shape normals, depths, etc. The authors use the "trainable copy" of the original weights of the pre-trained diffusion model and connect these "copy" blocks in the original model with zero convolution layers. Thus, we don't need to retrain the whole model and also guarantee the quality as well as the flexibility of the model.

Text-to-3D generation and Image-to-3D are novel tasks in the field of multi-modal learning and have the potential to be applied in many cases such as 3D object reconstruction, 3D scene generation and so on. DreamFusion [56] adopts a pre-trained 2D text-to-image diffusion model to perform text-to-3D synthesis. It optimizes a randomly-initialized 3D model based on Neural Radiance Fields (NeRF) with a probability density distillation loss, which utilizes a 2D diffusion model as a prior for optimization of a parametric image generator. Figure 2.7 illustrates the process of the text-to-3D generation using DreamFusion, and the image is adapted from the demo in the original work.

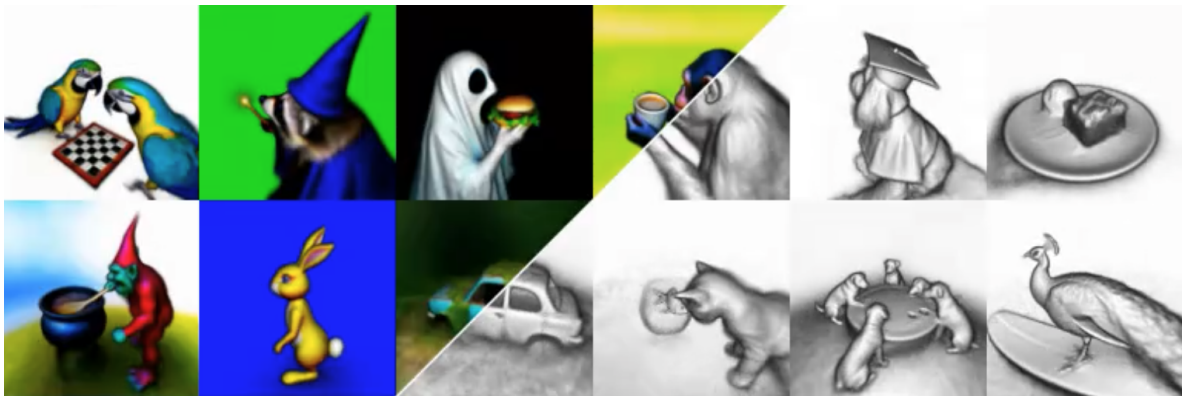


Figure 2.7.: 3D model NeRF synthesis prompt using DreamFusion, image taken from [56].

3. Related Works

3.1. 6 DoF Pose Estimation

3.1.1. Learning-Free Methods

In the early stage of the 6 DoF pose estimation research, the methods are mainly based on the hand-crafted feature descriptors such as Scale-Invariant Feature Transform (SIFT) [57], Speeded Up Robust Features (SURF) [58] and Oriented FAST and Rotated BRIEF (ORB) [59]. The main idea of these methods is to extract the feature points from the image and find the 2D-3D correspondences. After that, the pose of the object can be estimated by solving the Perspective-n-Point (PnP) problem. The PnP problem is a classic problem in computer vision which is to estimate the pose of an object given a set of 3D points in the object coordinate system and the corresponding 2D projections in the image plane. The PnP problem can be solved by using the Random Sample Consensus (RANSAC) [60] algorithm to find the inliers and then use the least square method to estimate the pose. The main drawback of these methods is that they are sensitive to occlusion, texture-less objects and illumination change.

ICP [13] and its improved version such as Generalized Iterative Closest Point (GICP) [61] and Normal Iterative Closest Point (NICP) [62] are also used to estimate the pose of the object when the depth information (point cloud) is available as reference. The task is also called point cloud registration where a partial point cloud is registered to a complete one or the alignment between two 3D scenes is estimated. However, the ICP is basically a time-consuming iterative method that is sensitive to the initial pose and the convergence is not guaranteed. Using ICP alone is not sufficient to achieve a robust pose estimation, but it can be used as a post-processing method to refine the pose estimation at the coarse level. The optional ICP module as the learning-free algorithm can effectively reduce the residual error existing in the learning-based methods and improve the final accuracy. In our framework, we also adapt the ICP as our refinement to further boost the performance.

3.1.2. Template-Based Methods

Template-based methods have been widely used to handle the texture-less object in the pose estimation task. In 2D object detection, a template is created from the object of interest and we scan the reference image with the template to locate the object. Adapting the same idea, in 6 DoF pose estimation a template is usually rendered from the 3D CAD model of the object and the pose of the object can be estimated by calculating the similarity score between the template and the reference image.

Gu et al. 2010 [63] propose to use a mixture of holistic templates and discriminative learning for joint object categorization and viewpoint classification. Different objects are learned

in the mixture and related to the canonical viewpoints with different supervision strategies. Hinterstoisser et al. 2012 [64] confront the time-consuming shortcoming of the template-based methods by using the gradient response maps and we only need to test a subset of all possible pixel locations. Wohlhart et al. 2015 [65] provide a powerful approach to compute the descriptors for object views which memorize both object class and pose information. The scalable nearest neighbor search is utilized to handle a large number of objects as well as viewpoints. Generalization of the template-based problem is emphasized in the recent work from Nguyen et al. 2022 [66] and increases the robustness against the occlusion and cluttered background by using the Convolutional Neural Network (CNN)-based feature extraction. A similar idea is used in OSOP [67], which is a one-shot method combining template matching with dense 2D-3D correspondence prediction. Additional P*n*P+RANSAC or Kabsch [68]+RANSAC and pose hypothesis verification are used to further refine the pose estimation.

3.1.3. Keypoint-Based Methods

Instead of directly estimating the pose from the reference input, keypoint-based methods first detect the key points or super points in the reference image and then solve the correspondence matching problem between the RGB image and the 3D model by P*n*P. In the case of point cloud registration, the task becomes to find the correspondences between the partial point cloud and the 3D model.

RGB-only keypoint matching is the most common method because the 2D data is more accessible than the data with additional depth information and the networks dealing with 2D image are more mature as well. Besides the hand-crafted feature descriptor that we introduced before, methods like [19, 69, 70, 71, 72] use the CNN-based architecture to extract the 2D semantic keypoints and estimate the pose by P*n*P algorithm. Especially, Oberweger et al. 2018 [73] propose a RGB-only method that can estimate the pose under large occlusion. Their solution is to predict heatmaps from multiple patches independently and calculate the accumulated score to ensure a robust estimation. ZebraPose [74] assigns a hierarchical binary grouping to encode the object surface. It replaces the sparse correspondence matching with dense maps learning to handle the occluded object and make the learning of 2D-3D correspondences more efficient.

With the promising performance of the 3D networks [75, 76] in recent years, some methods are proposed to directly tackle the 3D key points in the pose estimation task. D3Feat [77] leverage a 3D CNN-based architecture to jointly learn the detection and description of 3D local features. SpinNet [78] aims to learn efficient descriptors which are rotationally invariant. The Spatial Point Transformer is introduced to map the local surface into a cylindrical space for end-to-end optimization with $\mathbb{SO}(2)$ equivariant representation. YOHO [79] is another work that ensures the rotation invariance with the help of group equivariant feature learning. The novel descriptors make the correspondence matching more accurate and fewer P*n*P iterations are needed. GeoTransformer [11] leverages the efficient local feature extraction by using the geometrical embedding and follows the coarse-to-fine pipeline to achieve RANSAC-free pose estimation. The enhanced version of this work, RolTr [80] further improved the performance by utilizing the local-to-global attention-based rotation-invariant feature learning. And it outperforms the state-of-the-art methods in the low-overlapping scenarios.

3.1.4. Direct Regression Methods

Another popular group of 6 DoF pose estimation methods follows an end-to-end pipeline which requires the estimation process to be differentiable.

PoseCNN [81] is an early research that leverages the CNN on 2D image. It estimates the translation through the object center with the distance to the camera and regresses the quaternion representation of rotation. The symmetrical object is also considered by introducing a novel loss function of the rotation. Similar work from Wu et al. 2018 [82] trains a CNN-based model only on the synthetic data instead of expensively annotated object pose data. They overcome the domain gap between real and synthetic data by using object masks as an intermediate representation. Deep-6DPose [83] extends the instance segmentation network Mask Region-based Convolutional Neural Network (R-CNN) [84] and decouples the translation and rotation to make the rotation be regressed in Lie algebra representation. GDR-Net [85] is a novel method that directly regresses the 6 DoF pose from the RGB data in an end-to-end manner. It takes the zoomed-in region of interest as input and predicts hierarchical intermediate geometric features. Then the pose is regressed from the introduced tricks of Dense Correspondances and Surface Region Attention.

Direct regression of the transformation between two point clouds is more straightforward than the correspondence-based method in the point cloud registration task. PointNetLK [86] combines the PointNet in 3D domain with the Lucas & Kanade algorithm [87] for 2D image alignment together to regress the transformation between two frames. Huang et al. 2020 [88] propose a method that optimizes the registration by minimizing a feature-matrix projection error without matching the correspondence. OMNet [89] is another global feature-based iterative network for partial-to-partial registration. It utilizes overlapping masks to reject non-overlapping regions, transforming partial registration into same-shape registration.

Thanks to the decreasing cost of RGB-D sensor in recent years, RGB data with additional depth information enable us to more precisely handle texture-less objects in poorly illuminated scenarios. Some RGB-D based dense methods are proposed to estimate the pose in a fine-grained way. Kehl et al. 2016 [90] use regressed descriptors of locally sampled patches with a convolutional auto-encoder and then cast 6D voting for each patch to estimate the pose. Pix2Pose [91], PVNet [92] and DenseFusion (DF) [93] are typical pixel-wise regression methods, where glsdf is a representative method that pixel-wise votes the 6 DoF pose prediction. The network concatenates the 2D feature extracted from CNN and the 3D feature from PointNet together with the global feature through average pooling. The pixel-wise dense predictions are voted through the trainable confidence for each pixel. And FFB6D [94] as its successor, builds a bidirectional network to enhance the feature fusion from both domains. HybridPose [95] utilizes more information from the image like key points, edge vectors and symmetry correspondences to achieve more diverse feature representation for regression.

Common structures of 6 DoF pose estimation methods are summarized in the following figure 3.1.

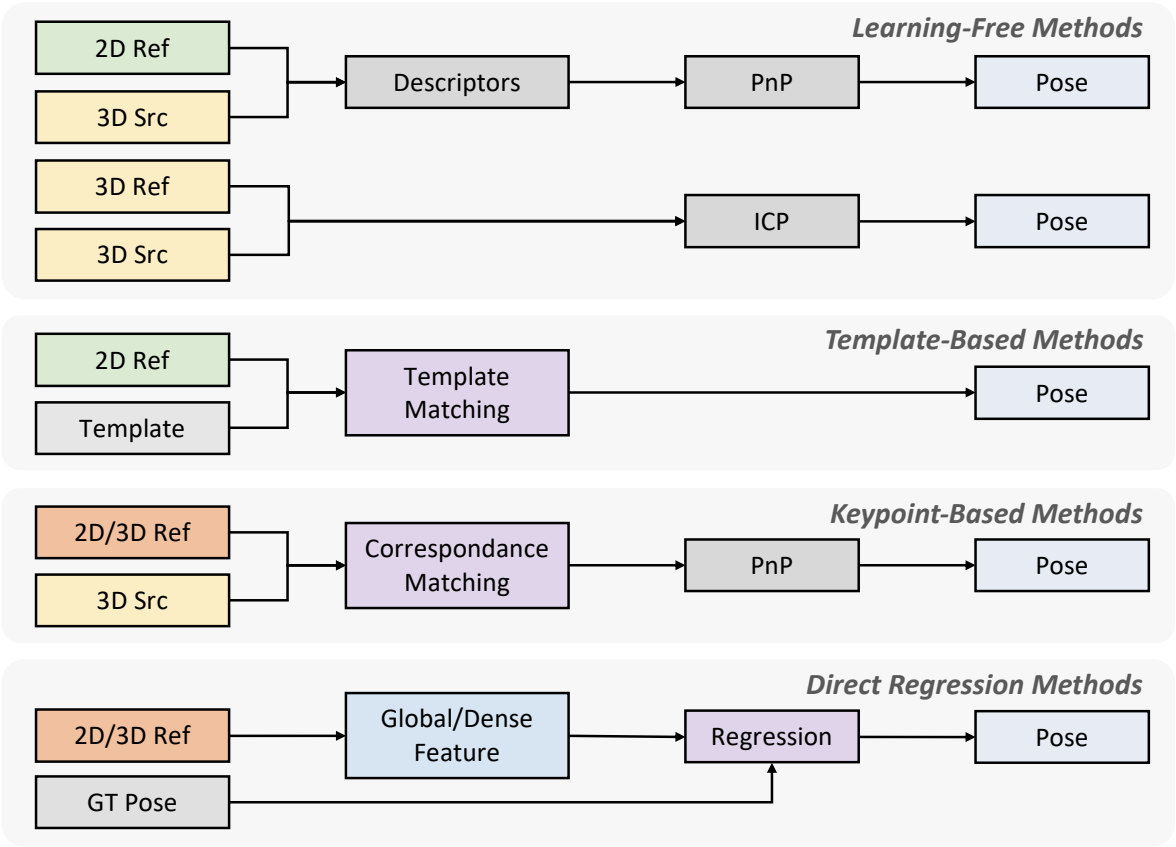


Figure 3.1.: Structures of some common 6 DoF pose estimation methods.

3.2. Diffusion Models

3.2.1. Foundations

Besides the original DDPM [1, 2] which is introduced in section 2.2.2, another two fundamental models of all works based on diffusion models are the Score-based Generative Models (SGMs) [34, 96] and Stochastic Differential Equations (SDEs) [97, 98].

The concept of SGMs is to use the Stein-score function to optimize the estimated distribution. The score $\nabla_{\mathbf{x}} log p(\mathbf{x})$ is defined as the gradient of the log-density of the Probability Density Function (PDF) $p(\mathbf{x})$. It represents a vector field that points in the direction of the steepest ascent of the PDF. The advantage of the score-based model is that we can parameterize the model without considering the normalization constant of the PDF by the operation of the derivative. That means we don't need any special treatment to make the normalizing constant trackable compared with the likelihood-based models. The training of the score-based models is to minimize the Fisher divergence between the model and the data distribution, which is defined as:

$$D_F = \mathbb{E}_{p(\mathbf{x})}\left[\left\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\right\|^2\right] \tag{3.1}$$

where $\nabla_{\mathbf{x}} log p(\mathbf{x})$ is hard to track in practice because we don't have the knowledge of the data distribution $p(\mathbf{x})$. However, we can utilize the method of score matching [99, 100] to minimize the Fisher divergence without knowing the ground truth score of the data.

The likelihood-based or score-based diffusion models can be further extended from the fixed step size to the continuous time domain using the SDEs. Perturbing the data distribution with a continuously developing noise following the SDEs process, we have the following equation:

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)\mathrm{d}\mathbf{w} \tag{3.2}$$

where $\mathbf{f}(\mathbf{x}, t)$ is called drift term and $g(t)$ is the diffusion coefficient, $\mathbf{w}$ is a standard Wiener process (or Brownian motion). The forward processes of DDPMs and SDEs have the discrete version of the continuous-time SDEs. And the corresponding reverse SDE can be represented as:

$$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} log p_t(\mathbf{x})\right]\mathrm{d}t + g(t)\mathrm{d}\mathbf{w} \tag{3.3}$$

where $p_t(\mathbf{x})$ is the density of the data distribution at time t. After that, the time-based training objective can be formulated with modification from equation 3.1 as:

$$D_F = \mathbb{E}_{t\in\mathcal{U}(0,T)}\mathbb{E}_{p_t(\mathbf{x})}\left[\lambda(t)\left\|\nabla_{\mathbf{x}} log p_t(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, t)\right\|^2\right] \tag{3.4}$$

where $\mathcal{U}(0, T)$ is a uniform distribution over the interval $[0, T]$ and $\lambda(t)$ is a weighting function that is used to balance the contribution of each timestep. The SDEs based diffusion models are more flexible and can be applied to the continuous-time domain, which is more fine-grained than the discrete-time models.

3.2.2. Improvements on Sampling

The sampling process of the original diffusion models takes a large amount of time because of its iterative architecture during the reverse process. In order to improve the efficiency of the sampling process, some work has been done to accelerate the sampling process.

Learning-Free Sampling

The first group of methods can be categorized as learning-free sampling. The main works of the advanced sampler focus on the efficient discretizing of the time-continuous solver. The SDE-based sampler such as Consistent Annealed Sampling (CAS) [101] which is improved from Annealed Langevin Dynamics (ALD) [102] proposes a modified SDE with consistent scaling of the added noise and ensures the expected geometric progression of the noise and make the generated samples closer to the data distribution. Another work from Jolicoeur-Martineau et al. 2021 [103] introduces a new SDE solver with adaptive step size controlled by comparing the high-order and low-order SDE solvers' outputs.

There are also many methods that rely on the Ordinary Differential Equations (ODEs) solver. Song et al. 2021 [98] states the probability flow ODE with the marginals same as the reverse-time SDE. The ODE solvers are deterministic, therefore the convergence is much faster than the stochastic solvers with the small tradeoff of the sample quality. Denoising Diffusion Implicit Models (DDIM) [104] is the pioneer of the work in this field, which extends the DDPM to non-Markovian perturbation. Karras et al. 2022 [105] propose a sampling process using Heun's 2$^{\text{nd}}$ order method. DPM-Solver [106] is a fast dedicated high-order solver for diffusion ODEs with the convergence order guarantee. The advanced ODE solver makes the sampling process much faster than DDIM and also improves the sample quality.

Learning-Based Sampling

Another sampling strategy is called learning-based sampling. By training the reverse process, the sampler is able to select the critical steps so that the training objective can be efficiently optimized. Watson et al. 2021 [107] introduce an exact dynamic programming algorithm that finds the optimal discrete-time schedules for any pre-trained DDPM. After that, Differentiable Diffusion Sampler Search (DDSS) [108] is proposed to accelerate the sampling by differentiating through sample quality scores.

Instead of optimizing the schedule of the sampling process, works like Early-Stopped DDPM [45] and Truncated DDPM [109] propose to train the diffusion model without the complete forward and start the denoising from a non-gaussian distribution, so that the denoising steps can be significantly reduced. Another approach is to utilize the knowledge distillation [110, 111] to compress the sampling steps into just a few steps. Progressive Distillation, for example, is able to reduce the steps from over a thousand to four steps without losing much sample quality by progressively parameterizing the sampler.

3.2.3. Improvements on Likelihood

Original DDPM need to optimize the variational Evidence Lower Bound (ELBO) because of the intractable log-likelihood (see section 2.2.2). When we use time-continuous diffusion such as SDEs, some methods are proposed to estimate the likelihood of the data distribution. Song et al. 2021 [97] realize a connection between the maximum likelihood estimation with the weighted combination of score matching with a specific weighting scheme. Variational Diffusion Models (VDM) [112] introduce a simplification of the time-continuous variational lower bound which is invariant to the noise schedule and accelerates the optimization. Besides combining the score matching the maximum likelihood estimation, Nichol et al. 2021 [32] propose a hybrid way to define the training objective by adding the variational lower bound with the score matching loss with a control parameter λ.

3.2.4. Extended Data Structures

Most of the works based on diffusion models focus on the 2D image generation and manipulation tasks, where the data is continuous in 2D space. However, there is also some work that applies diffusion models to other data structures such as text, graphs, point clouds and so on. The standard method of the diffusion model may not be suitable for these data structures, so some modifications are needed to adapt the diffusion model to different data structures.

Discrete Data

To adapt the perturbation of the Gaussian noise to the discrete data, some works extend the original DDPM to the discrete domain, especially in NLP field. For instance, VQ-Diffusion [113] inspired from Vector Quantized Variational Autoencoder (VQ-VAE) [114] replaces the Gaussian noise with a mask-and-replace diffusion strategy to avoid the accumulation of error, which is the first application of diffusion model on vector-quantized data. Concrete Score Matching (CSM) [111] proposes an analogous score function called "Concrete score",

which is a generalized Stein score for discrete data. It is defined by the rate of change of the probabilities with respect to local directional changes of the input. Liu et al. 2023 [115] propose a method dealing with the data on constrained and structured domains, including discrete data as a special case. The diffusion model is driven by a drift force that is a sum of two parts: one singular force designed by Doob's h-transform that enforces the outcome in the constrained domain, and one non-singular neural force field that ensures statistical consistency.

Manifold

In many real-world applications, data lies in the non-Euclidean space. For instance, the Riemannian manifolds are widely used in the robotic and computer vision field. Riemannian Diffusion Model (RDM) [116] generalizes continuous-time diffusion models to arbitrary Riemannian manifolds and derives a variational framework for likelihood estimation. Similar work like Riemmanian Score-based Generative Model (RSGM) [117] extends the SGMs to the setting of compact Riemannian manifolds.

Graph structure is also a common data structure that represents the relationship between different entities which is an efficient expression in fields like data mining, protein structure prediction and so on. Niu et al. 2020 [118] first propose a permutation invariant equivariant graph neural network called EDP-GNN and integrate it into the score-based generative model. Graph-GDP [119] further constructs a forward diffusion process with a SDE, which perturbs the graph with a known edge probability. A position-enhanced graph score network is trained to extract the structure and position information from the graph for permutation equivariant score estimation.

The majority of the raw data lie in the high-dimensional space but can be represented in the low-dimensional manifold using a network like autoencoder. And working on these latent spaces is more efficient and effective. For this reason, some recent works utilize the diffusion model on the manifold which is compressed from the original data space. The Latent Score-based Generative Model (LSGM) [120] proposes a method to jointly train a score-based diffusion model and a variational autoencoder by optimizing the ELBO of the VAE together with the training objective of the score matching model. Instead of the joint training, Latent Diffusion Model (LDM) [54] used in the influential work of Stable Diffusion, addresses the training into two separate phases. First, the autoencoder is trained to project the data into latent space. Then, the diffusion model is trained to directly sample the latent expression of the data.

3.3. Diffusion models in Pose Estimation

The objective is 6 DoF pose estimation lies on the $\mathbb{SE}(3)$ of Lie group, which is a differentiable manifold that is generalized from the Euclidean space. Recent works from Leach et al. 2022 [121] first explore the usage of the diffusion model to generate the probability density on $\mathbb{SO}(3)$. Jagvaral et al. 2023 [122] further optimized the loss function from the Euclidean space to the manifold $\mathbb{SO}(3)$. Urain et al. 2023 [123] propose smooth cost functions for joint grasp on $\mathbb{SE}(3)$ through diffusion. Yim et al. 2023 [124] introduce a framework called

FrameDiff to learn the $\mathbb{SE}(3)$ equivariant score of the diffusion models over multiple frames in protein structure generation.

Typical work from Hsiao et al. 2023 [125] focuses on the advantages of diffusion models towards the pose ambiguity problem on $\mathbb{SE}(3)$. The denoiser backbone consists of several Multilayer Perceptron (MLP) blocks processing the 2D global feature extracted from a pre-trained ResNet. Figure 3.2 shows the overview of the framework. The authors use the surrogate Stein score calculation on $\mathbb{SE}(3)$ following the score-based diffusion architecture and Fourier-based conditioning mechanism to deal with the ambiguity problem caused by the symmetrical object. Additionally, they also compare the advantage of $\mathbb{SE}(3)$ parametrization over previous works on $R^3\mathbb{SE}(3)$. This work is trained and evaluated on the synthetic dataset SYMSOL [126] with fice texture-less objects, namely tetrahedron, cube, icosahedron, cone, and cylinder. The denoising process is shown in figure 3.3. The real data captured by the camera is not considered in this work. In our proposed method, we focus on pose estimation with diffusion models under the real-world scenario and even with occlusion, which is more general and challenging.

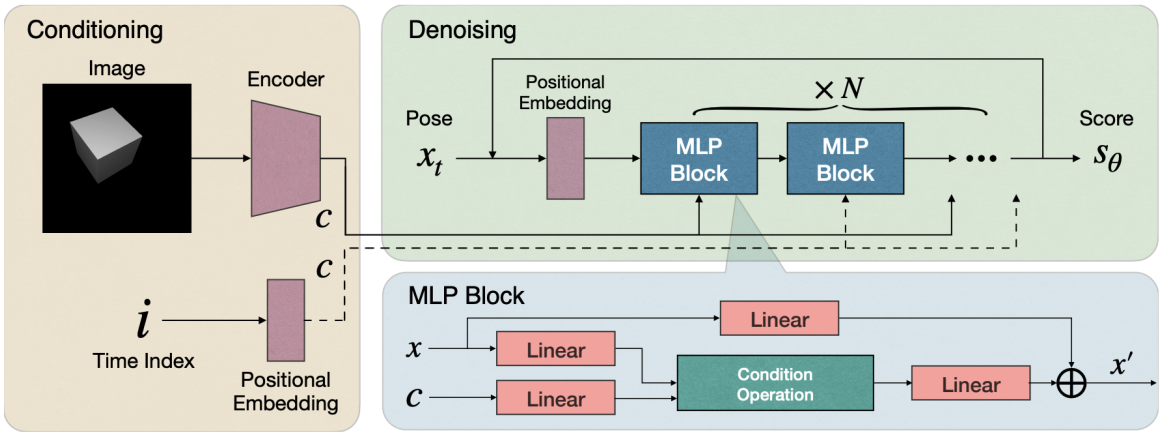


Figure 3.2.: Overview of the network from the work "Confronting Ambiguity in 6D Object Pose Estimation via Score-Based Diffusion on SE(3)", image adapted from the original paper [125].

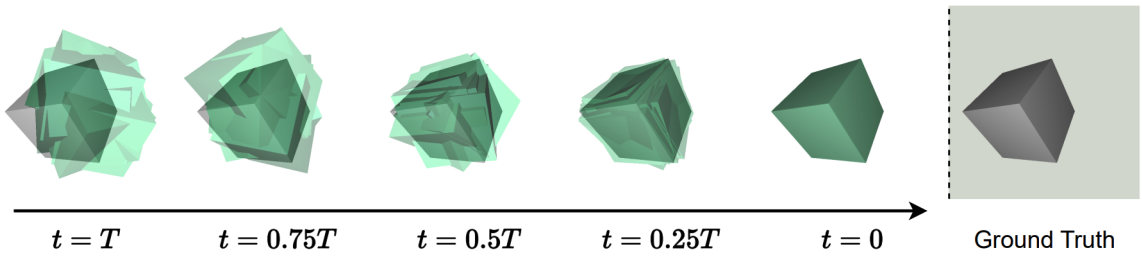


Figure 3.3.: Visualisation of the denoising process from the work "Confronting Ambiguity in 6D Object Pose Estimation via Score-Based Diffusion on SE(3)", image adapted from the original paper [125].

Wang et al. 2023 [127] propose a diffusion-aided bundle adjustment network for the camera

pose estimation with a series of input images. Similar to the previous work, the diffusion network is conditioned by the 2D feature downstream by a pre-trained encoder. The highlight of this method is the geometry-guided sampling of the diffusion model with the Sampson epipolar error minimization so that the image-to-image epipolar constraint can be satisfied in the sampling phase. Figure 3.4 and 3.5 illustrate the overview of the framework and the denoising process of the bundle adjustment. The model is evaluated on CO3Dv2 [128] containing turntable-like objects and RealEstate10K [129] which captures videos of the interior and exterior of real estate. Both datasets aim to estimate the camera poses with a series of input images relative to the object or the scene. Our work focuses on the single-frame object pose estimation with the fixed camera parameters and processes the data further with the depth information.

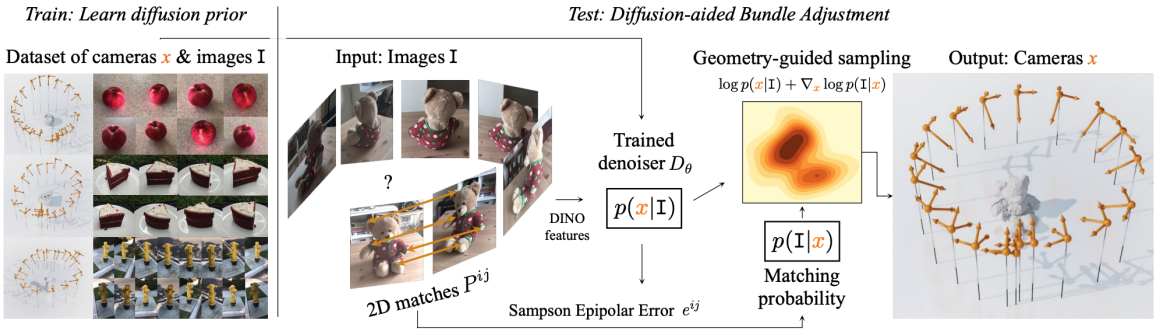


Figure 3.4.: Overview of the network from PoseDiffusion, image adapted from the original paper [127].

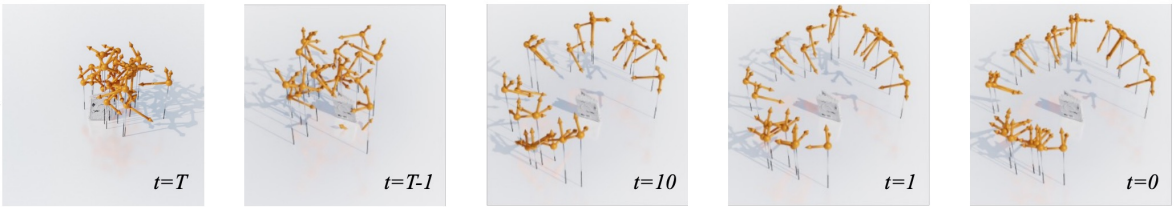


Figure 3.5.: Visualisation of the denoising process from PoseDiffusion, image adapted from the original paper [127].

4. Methodology

4.1. Introduction

In this chapter, we will introduce the details of our proposed method, namely the pose hypotheses diffusion. The intuitive idea is to directly denoise the pose which consists of translation and rotation, $\mathbf{T} = (\mathbf{t}, \mathbf{r})^T$. As the diffusion model is basically one of the generative models, we use the diffusion pipeline to generate the possible pose hypotheses which is similar to the image synthesis but with a different objective. So we call this kind of pose estimation method pose hypotheses diffusion.

Through the experiments using different representations of the rotation, we find that the 6D representation of the rotation introduced in 2.11 is the most efficient one. So we use this representation in the following chapters and the comparison with other forms of rotation will be discussed in the experiment section.

By generating multiple samples for one reference input y, we can get a set of pose hypotheses $\mathbf{T}_1, ..., \mathbf{T}_N$, which compose a hypothesis distribution $h(\mathbf{T}|y)$ that can be used to estimate the pose $\mathbf{T}$ of the reference input $\mathbf{r}$. For a given object without any ambiguity, the distribution $h(\mathbf{T}|y)$ should be a delta distribution in the ideal case, or in other words, squeezed to a single point in the spatial solving space. The pose hypotheses $\mathbf{T}_1, ..., \mathbf{T}_N$ should be close to each other and the variance of the distribution should be small. On the contrary, if the object is symmetrical, the distribution of the pose should fit the corresponding pattern of the symmetry in the solving space. In our work, we focus on the first case and we don't propose any post-processing method to analyze the distribution of the symmetrical object, which can be a future work.

4.2. Framework

This section introduces the overall structure of the pose hypotheses diffusion, including the 2-phase architecture of the diffusion model and the other modules in the whole model.

Similarly to the original diffusion pipeline, we first need to train a model to estimate the noise $\boldsymbol{\epsilon}_\theta$ conditioned with the input $\mathbf{x}_t$, the timestep t as well as the guidance y. Then we can go through the reverse process in which we generate the pose hypothesis $\mathbf{T}_{t-1}$ with conditional distribution $q(\mathbf{T}_{t-1}|\mathbf{T}_t, y)$ step by step using the equations we derived in section 2.2.2.

4.2.1. Training Phase

In the training phase of the pose hypotheses diffusion, we basically let the backbone network predict the noise given by the noised pose $\mathbf{T}_t$, the noised pose can be derived from the ref-

erence pose $\mathbf{T}_0$ with the noise schedule β_t which introduced before in the forward process. Most applications using diffusion have a convolutional UNet-like backbone [130], which performs well in the 2D tasks. However, when dealing with the pose estimation task, we have a different objective and the convolutional neural network is no longer suitable. In our model, we utilize the transformer encoder as the backbone network, which has been proven to be effective and flexible in not only the natural language processing but also the CV tasks.

Additionally, we also need to provide the timestep t and the guidance y to the backbone. The guidance here is the feature of the reference RGB or RGB-D image depending on the requirement of the task or the dataset. We use RGB-D image in our experiments which has both 2D and 3D features that can be extracted and fused into the model. As 2D feature extractor, a pre-trained self-supervised Vision Transformer with DINO [131] is used and the 3D feature is extracted from a pre-trained FoldingNet encoder [132]. The structure of the training process is shown in figure 4.1.

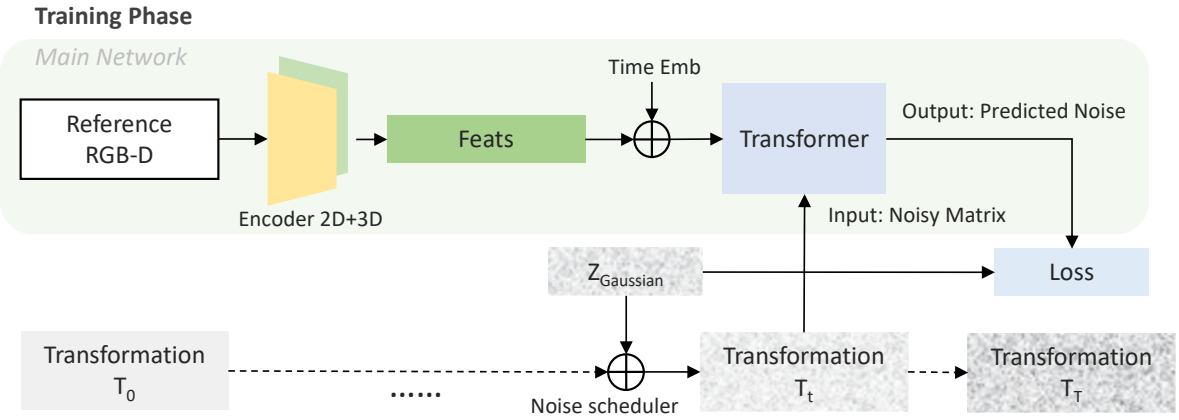


Figure 4.1.: Structure of the training phase of the pose hypotheses diffusion.

4.2.2. Sampling Phase

Assuming that the denoiser is converged in the training phase, we can use the denoiser to iteratively generate the pose hypotheses with the randomly initialized transformation $\mathbf{T}_T$. Same as the training phase, we need to provide the timestep t and the guidance y to the backbone. Given a reference RGB-D image, we use the pre-trained 2D and 3D encoder to extract the downstream features and concatenate them as the conditional embedding of the diffusion backbone.

Since during the training phase, the network has learned the noise distribution conditioned each timestep embedding, so the denoiser has the capability to predict the noise $\boldsymbol{\epsilon}_\theta$ from $t = T$ to $t = 0$. Then we can use the equation 2.24 to get the pose hypothesis $\mathbf{T}_{t-1}$ and repeat the process until we get the final pose hypothesis $\mathbf{T}_0$.

In the training phase, we feed the network with batches of data that are different in the timestep t and the guidance y (different reference images). During the sampling phase, we can easily make the batch size to one and only infer one pose hypothesis $\mathbf{T}_0$ for one reference input. Another efficient way is to simultaneously sample multiple pose hypotheses by

batchifying the randomly initialized transformation and conditioning with the same reference input. The mean of the sampled pose hypotheses can be more precisely estimated as the final pose estimation if the pose of the object is uniquely determined in the space. This can be switched so that the batch of the random Gaussian initialization is conditioned with different reference inputs, which can infer multiple poses for different frames or different objects.

After the optional multi-hypotheses inference, we can further use some algorithms to refine the pose. In our model, we choose ICP[133] algorithm to refine the pose hypotheses and finally get the output transformation $\mathbf{T}_r$. The structure of the sampling process is shown in figure 4.2. Details of each model in the training phase and sampling phase will be introduced in the next section.

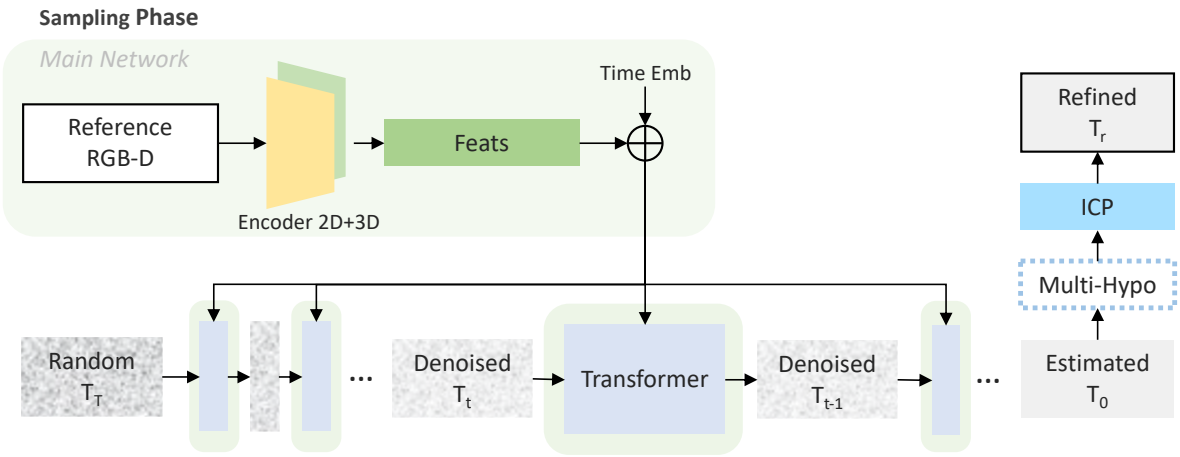


Figure 4.2.: Structure of the sampling phase of the pose hypotheses diffusion.

4.2.3. Speed up Sampling with DDIM

It is relatively slow to generate the sample from DDPM because the denoising process follows the whole Markov chain of the reverse process which is quite long. Slow inference reduced the performance of the real-time application of pose estimation, which we need to optimize. In order to speed up the sampling process, we use the DDIM [104] with the same training procedure as DDPM but more efficient in the reverse process.

First, we rewrite the reverse conditional probability in 2.26 to be parameterized by a desired standard deviation σ_t using the reparameterization trick in 2.22 as:

$$q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1};\ \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I}) \tag{4.1}$$

Compared with the original probability $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I})$, we have:

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \tag{4.2}$$

Then we let $\sigma_t^2 = \eta \cdot \tilde{\beta}_t$ where η is a hyperparemeter that controls the sampling stochasticity. $\eta = 0$ corresponds the best performance of the model and $\eta = 1$ corresponds to the original

DDPM. In the reverse process, we only sample a subset of the complete diffusion steps $\{\tau_1, ...\tau_S\}$ and the reverse process can be formulated as:

$$q_{\sigma,\tau}(\mathbf{x}_{\tau_i-1}|\mathbf{x}_{\tau_i}, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{\tau_i-1};\ \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_{\tau_i} - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I}) \tag{4.3}$$

With DDIM we can use a non-Markovian diffusion process whose reverse process can be much faster than the original DDPM and still keep the quality of the samples. It also remains the consistency of the reverse process when $\eta = 0$, where the reverse process is deterministic. That makes sure that multiple samples from the same reference input have similar high-level features and in our case, the pose hypotheses are consistent with each other.

4.3. Models

4.3.1. Denoiser Network

The following figure 4.3 illustrates the structure of our main network in the diffusion model. As in the previous section mentioned, the transformer encoder processes the translation and rotation vector together with their position embedding conditioned with time embedding and 2D/3D feature embedding of the reference image and predicts the noise added at this timestep.

It is worth mentioning that the fusion of 2D and 3D feature is not pointwise aligned, which means we don't extract the pointwise feature from the 2D and 3D domain and feed to the network. Instead of doing that, we separately extract the global features and concatenate them together in order to reduce the difficulty of the convergence of the backbone with the tradeoff of the robustness and generalization of the model. This part of optimization will be discussed later.

4.3.2. Backbone

A modified transformer encoder is utilized as the backbone of our diffusion model. The Transformer [134] is a sequence-to-sequence model that uses multi-head self-attention layers to understand the relevant token in the sequence and follows the encoder-decoder structure in the NLP tasks. However, in our case, the encoder part is what we need to estimate the noise.

Similar to the vanilla transformer, our model consists of N stacked transformer encoder blocks. As shown in the left part of figure 4.3, each block is made up of two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We use residual connections around each of the two sub-layers and the Layer Normalization (LN) is applied before each sub-layer, which follows the design of Vision Transformer (ViT) [135].

The core of the transformer encoder is the multi-head self-attention mechanism, which is illustrated in figure 4.4. First, we create three vectors from each input vector $\mathbf{x}_i$, namely the query vector $\mathbf{q}_i$, the key vector $\mathbf{k}_i$ and the value vector $\mathbf{v}_i$. These vectors are created by multiplying the input vector $\mathbf{x}_i$ with three matrices $\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$ respectively that we

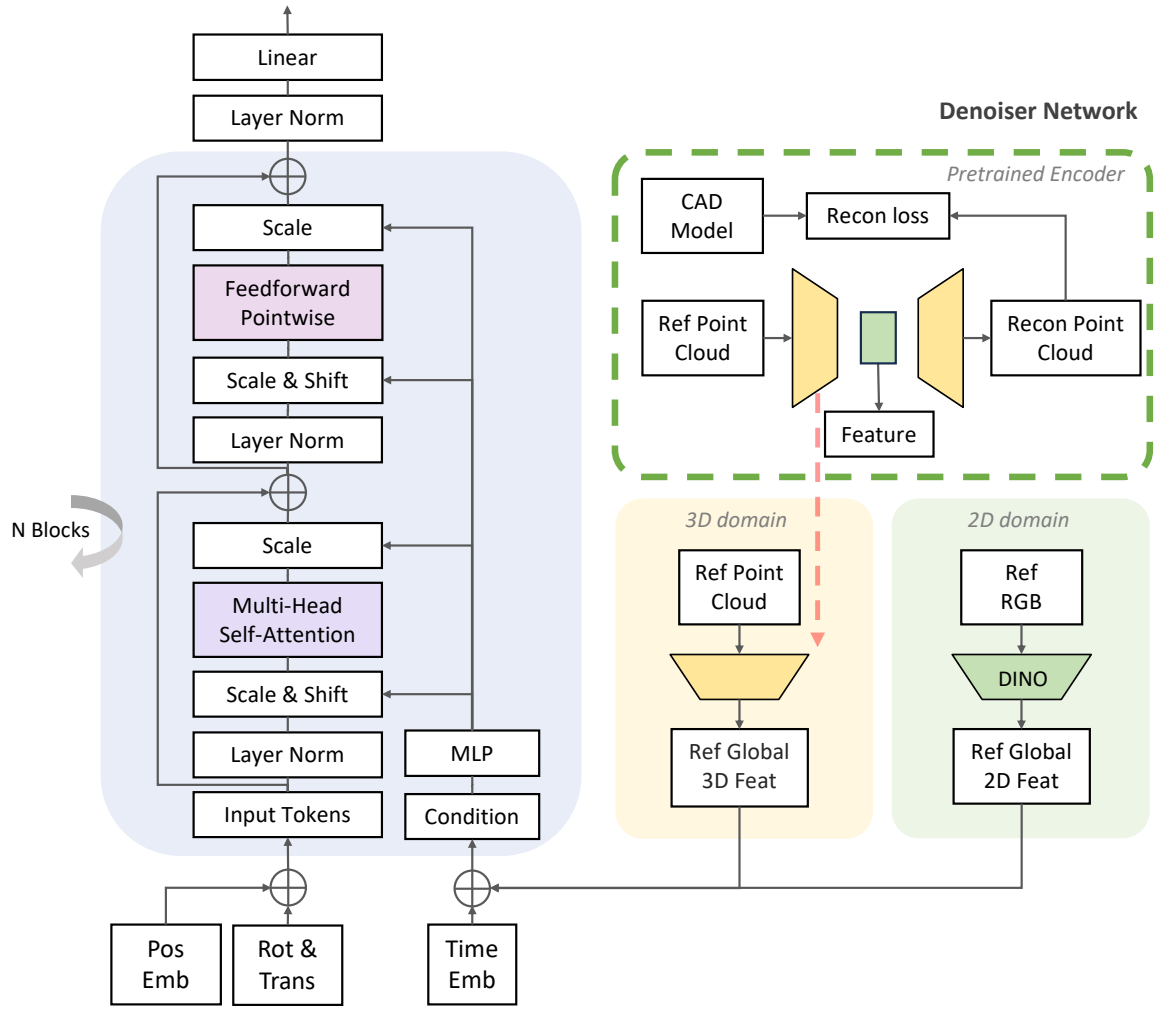


Figure 4.3.: Denoiser network with backbone and feature extractor.

trained during the training phase. Then we use the scaled dot-product attention to calculate the output vector $\mathbf{y}_i$:

$$\mathbf{y}_i = \text{softmax}\left(\frac{\mathbf{q}_i \mathbf{k}_i^T}{\sqrt{d_k}}\right)\mathbf{v}_i \tag{4.4}$$

where d_k is the dimension of the key vector $\mathbf{k}_i$. The scaled dot-product attention is the core of the transformer encoder and the multi-head self-attention is the extension of the scaled dot-product attention. The multi-head self-attention is defined as:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\mathbf{h}_1, ..., \mathbf{h}_n)\mathbf{W}^O \tag{4.5}$$

where $\mathbf{h}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$ and $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ and $\mathbf{W}^O$ are the trainable parameters. The multi-head self-attention allows the model to jointly attend to information from different representation subspaces at different positions. With a linear projection at the end, the model is able to learn a more complex function.

To effectively process the conditional input, we use modified Adaptive Layer Normalization (AdaLN) to replace the original layer normalization in the transformer encoder which

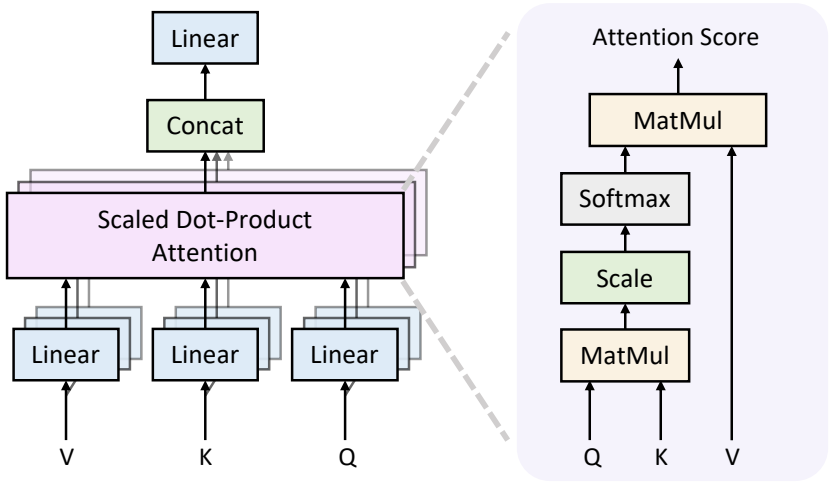


Figure 4.4.: Multi-head self-attention and scaled dot-product attention.

is introduced in [136, 137]. The learnable scale and shift parameters are regressed from the conditional input and applied in each sub-layer of the transformer encoder blocks. The modified transformer encoder block can be formulated as:

$$\mathbf{y}^{'} = \alpha_1(\mathbf{c}) \odot \text{Attention}[\gamma_1(\mathbf{c}) \odot N(\mathbf{x}) + \beta_1(\mathbf{c})] \tag{4.6}$$

$$\mathbf{y} = \alpha_2(\mathbf{c}) \odot \text{FeedForward}[\gamma_2(\mathbf{c}) \odot N(\mathbf{y}^{'}) + \beta_2(\mathbf{c})] \tag{4.7}$$

where $\mathbf{y}$ is the output of the block, $\mathbf{y}^{'}$ is the intermediate expression after the first sub-layer, $\mathbf{c}$ is the conditional input, $\alpha_1(\mathbf{c})$, $\alpha_2(\mathbf{c})$, $\beta_1(\mathbf{c})$, $\beta_2(\mathbf{c})$, $\gamma_1(\mathbf{c})$ and $\gamma_1(\mathbf{c})$ are the learnable parameters and N is the layer normalization. The $\odot$ denotes the element-wise multiplication.

4.3.3. Time Embedding and Postion Embedding

Why do we need time embedding and position embedding in our case? The architecture of the diffusion model determines that we need to let the network learn the influence of timestep on the noise estimation. So we have to encode the timestep information into the network. For the same reason, the sequence of the transformer input which is the transformation vector is also relevant and should be encoded, because each bit of the vector represents a different meaning and can not be shuffled.

The way we embed time and position information is generally called positional encoding. It should satisfy the following conditions [138]:

- It should be unique and deterministic defined for each position (or timestep).
- The encoded distance between any two steps should be consistent across different timesteps.
- The value should be bounded and generalized to any input.

The most common positional encoding is the sine and cosine positional encoding [134],

which is defined as:

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{4.8}$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{4.9}$$

where pos is the position, i is the dimension and d_{model} is the dimension of the input vector. And we add a fully connected layer to the positional encoding to make it trainable. Figure 4.5 shows the 64-dimensional positional encoding for a sequence with a length of 100 using the sine and cosine positional encoding.

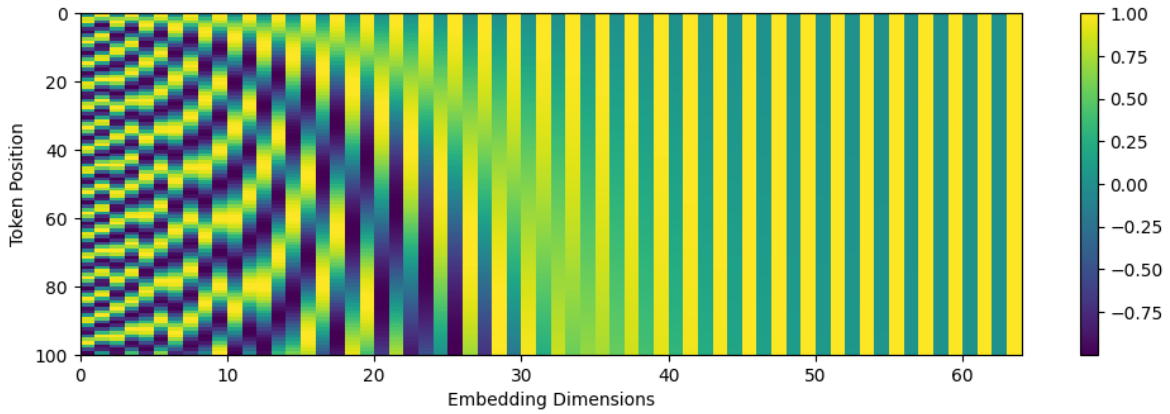


Figure 4.5.: The 64-dimensional positional encoding for a sequence with a length of 100.

4.3.4. 2D Feature Extractor

As the 2D feature extractor, the self-supervised Vision Transformer with DINO [131] is used. DINO is a self-supervised learning method that trains a Vision Transformer with a small set of negative examples. It is a contrastive learning method that maximizes the agreement between differently augmented views of the same image. The architecture of DINO shares the same overall structure with recent self-supervised approaches and also similarities with knowledge distillation.

Knowledge distillation is a learning paradigm where a student network g_{θ_s} is trained to match the output of a given teacher network g_{θ_t}, parameterized by θ_s and θ_t respectively. Given an input image x, both networks output probability distributions over K dimensions denoted by P_s and P_t. The probability P is obtained by normalizing the output of the network g with a softmax function. Given a fixed teacher network g_{θ_t}, the student network g_{θ_s} is trained to minimize the cross-entropy loss between the two distributions w.r.t the student parameters θ_s:

$$L = \min_{\theta_s} H(P_t(x), P_s(x)) \tag{4.10}$$

where $H(a, b) = -a \log b$ is the cross-entropy loss. This optimization problem is adapted to self-supervised learning by constructing different distorted views or crops of an image with a

multi-crop strategy. Making a set of two global views, x_1^g and x_2^g and several local views with smaller resolution, the loss function can be formulated as:

$$L_{DINO} = \min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in V \\ x' \neq x}} H(P_t(x), P_s(x')) \tag{4.11}$$

The structure of the self-supervised architecture is shown in figure 4.6. The model passes two different random transformations of the input image to both networks with the same structure but different parameters. The output of the teacher network is centered and a Stop-Gradient (SG) is applied on the teacher to let the gradients only propagate through the student network. The teacher parameters are updated with an Exponential Moving Average (EMA).

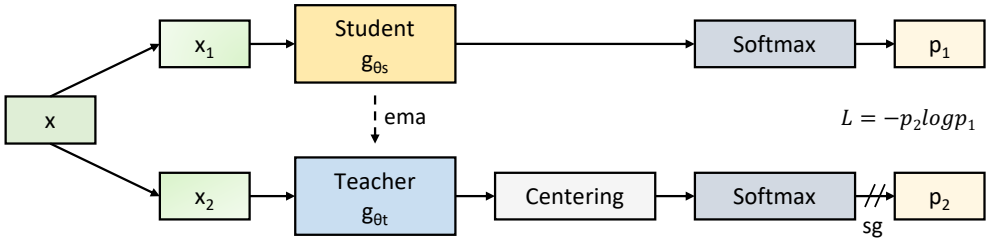


Figure 4.6.: Self-supervised architecture of DINO.

The backbone of the DINO is ViT [135], which proved that the transformer architecture also performs well on the 2D vision tasks. The standard transformer processes one-dimensional (1D) sequences, and in order to handle 2D images, the input is first flattened into a sequence of patches. For a input image $x \in R^{H \times W \times C}$ and patch size p, the patchified input can be denoted as $x_p \in R^{N \times (p^2 C)}$ with the number of patches $N = HW/p^2$. The reason why the patches are fed into the transformer rather than the raw image is that it is relatively easier for the network to understand the relationship between the patches than the raw pixels.

Similar to the vanilla transformer for NLP tasks, the patches need to be embedded with the positional encoding. A standard learnable 1D position embedding is used in the original paper. In order to solve the classification task with ViT, an extra token is added to the sequence of patches, which is called the class token. An additional MLP layer is used for the classification. But for the downstream tasks like ours, we only need the latent feature from the transformer encoder output. The overview of the ViT model is shown in figure 4.7, using the illustration in the original publication.

Compared with the convolutional neural network which has a dominant position in the field of 2D vision tasks, the Vision Transformer has advantages as well as drawbacks under some scenarios. The transformer is more computationally expensive and requires more memory than the convolutional neural network. And the transformer is not as robust as the convolutional neural network in the case of a small dataset. On the other hand, Vision Transformer architecture is more flexible and can be easily applied to different tasks with different input sizes. It offers a more natural way to process the 2D image, which is more similar to the human brain. The transformer is more suitable for tasks that require the global information of the image, such as the classification task. In our case, we use the Vision Transformer pre-trained with DINO to extract the global feature of the reference image and feed it to the transformer encoder as the conditional embedding.

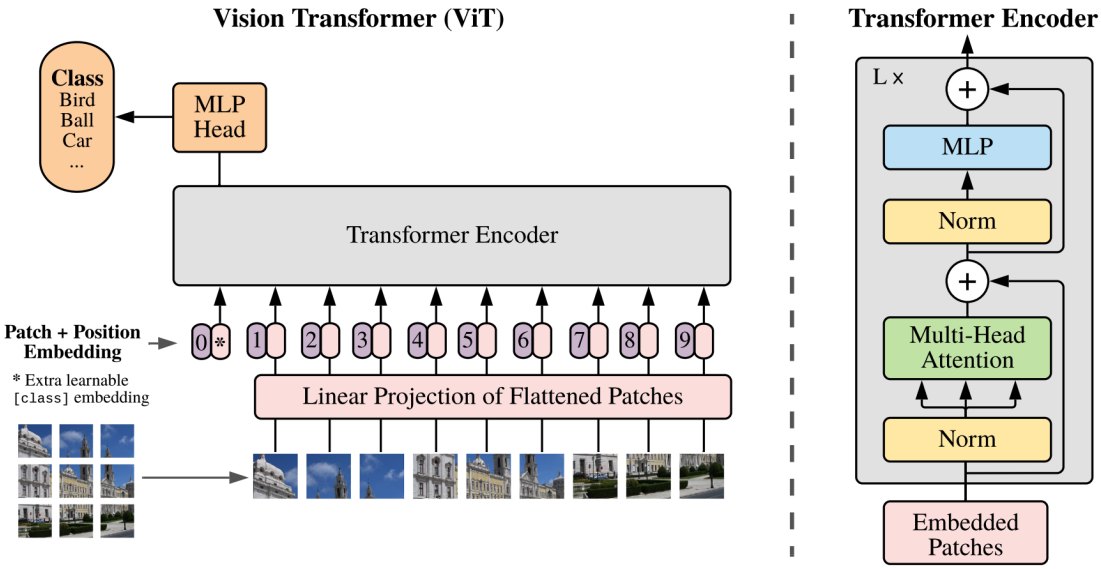


Figure 4.7.: Sturcture of the ViT model, image adapted from the original paper [135].

We directly use the weights of the model pre-trained on ImageNet [139], because the 2D feature downstream network generalizes well on other datasets than 3D feature extractor, which has right now rare well generalized downstream backbones that can cover any 3D objects. For this reason, we train the 3D feature extractor from scratch on the dataset we use and this part will be introduced in the next section.

4.3.5. 3D Feature Extractor

Due to the permutation invariance and transformation invariance of the point cloud, the 3D networks are differently constructed compared with the 2D networks. Famous works like PointNet [75] and PointNet++ [76] are the pioneers of the 3D deep learning. After that, the convolutional neural network is also introduced to the 3D domain such as KPConv [140].

In our case, we use an encoder-decoder architecture to first build up a point cloud completion task of the target dataset and utilize the latent feature from the encoder output as the 3D global feature of the reference RGB-D image. The reason for using the point cloud completion task rather than the classification or semantic segmentation task is that we need the network to learn the geometrical features such as shape, position and orientation of the object. The point cloud completion is more suitable than separately reconstructing the CAD model and reference partial point cloud because the network can learn the relationship between the two point clouds and the latent feature is more robust and generalizable.

FoldingNet consists of a graph-based encoder and a folding-based decoder. The graph-based encoder follows the design of [141] using the graph structure of the point cloud neighborhood. The encoder is built up with MLP and graph-based max-pooling layers, which are constructed from the k-nearest neighbor through the spatial position of the nodes in the point cloud. For each point, a local covariance matrix with the size of 3x3 is computed, flattened to a vector and concatenated with the point position as the input vector fed to the network.

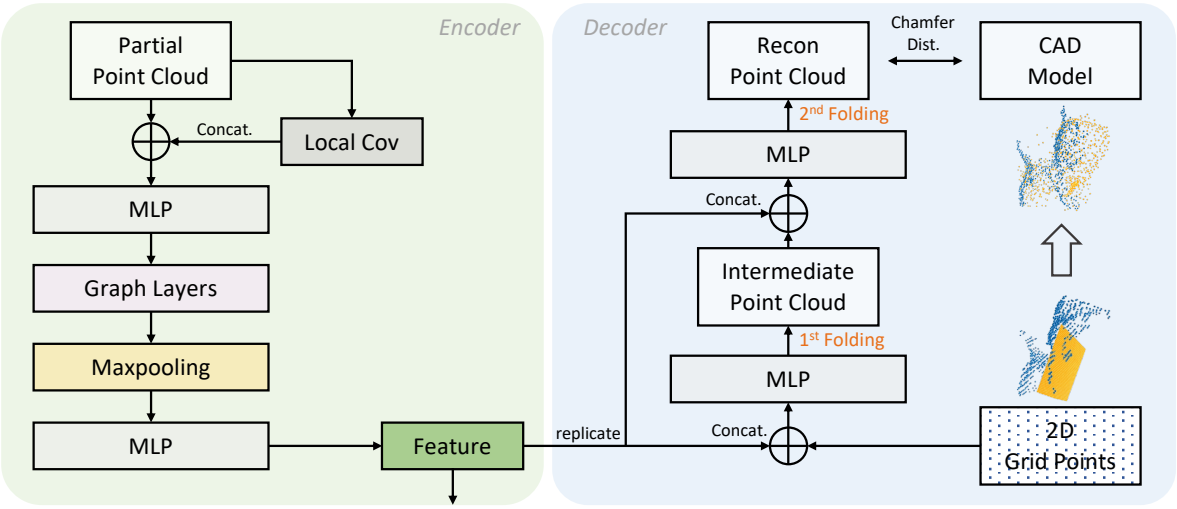


Figure 4.8.: 3D feature extraction with FoldingNet-based point cloud completion.

Afterward, the input vector is passed through MLP and graph layers to get the topology information of the local neighborhood. Then the max-pooling is applied to all nodes and the global feature is projected to a latent space by using another perceptron.

The folding-based decoder is built up with two consecutive MLP to wrap a fixed 2D grid into the shape of the reference point cloud and that explains why this network is called FoldingNet. First, a 2D grid matrix is initialized in a standard square shape at the origin and is concatenated with the replicated latent feature from the encoder output to become the input of the first folding MLP. The output is the first folded point cloud and it is again concatenated with the replicated latent feature and fed to the second folding MLP.

The final output point cloud is aligned with the reference point cloud using the bi-directional Chamfer Distance (CD) as the loss function, which can be formulated as:

$$L_{CD}(S, \hat{S}) = \max \left\{ \frac{1}{|S|} \sum_{x \in S} \min_{\hat{x} \in \hat{S}} \|x - \hat{x}\|_2, \ \frac{1}{|\hat{S}|} \sum_{\hat{x} \in \hat{S}} \min_{x \in S} \|\hat{x} - x\|_2 \right\} \tag{4.12}$$

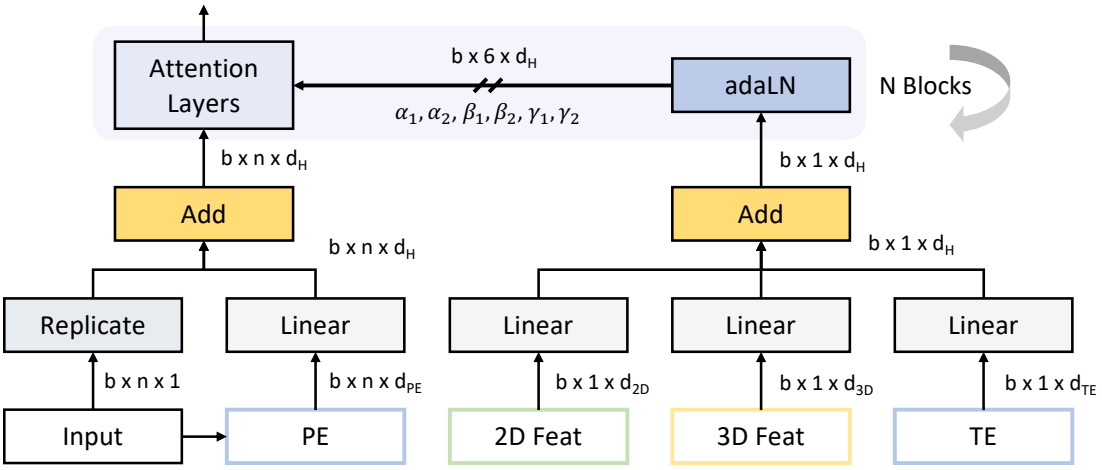
where S is the reference point cloud and $\hat{S}$ is the output point cloud. The formulation enforces that the output point cloud is close to the reference point cloud and vice versa. In our case, given a partial point cloud as input, the reconstructed point cloud is aligned with the CAD model at the pose of the partial point cloud. The structure of the FoldingNet is shown in figure 4.8.

4.3.6. Feature Fusion

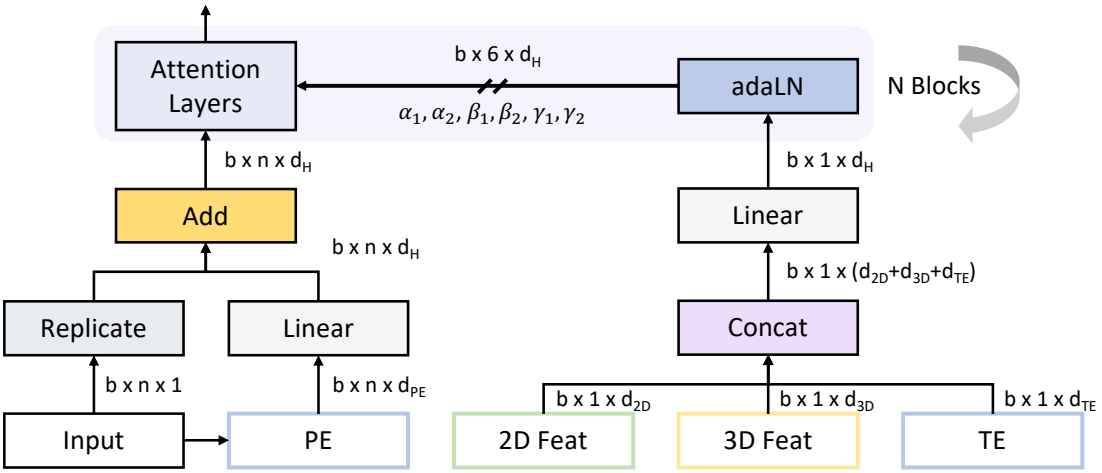
The way of fusing the features and embedding from different domains determines the performance of the model and the convergence of the training phase. Because features may compensate each other or may be redundant in some cases and the network cannot efficiently learn the relationship between the latent feature and the ground truth. In our model, we have the following feature or embedding vectors that need to be integrated into the network: the 2D feature f_{2D} from the pre-trained DINO model, the 3D feature f_{3D} from the pre-trained

FoldingNet model, the positional encoding of the timestep (time embedding) e_t and the positional encoding of the transformation vector (position embedding) e_p.

For the position embedding e_p, we follow the pipeline of the Transformer that e_p is directly added to the input vector x_t which is first replicated to the dimension of the hidden dimension of the network. For the conditional guidance of the diffusion model f_{2D}, f_{3D} and e_p, we have two ways introduced in our setup to merge these features into the backbone. The first method is that these features are projected separately to the hidden dimension of the network with fully connected layers and then added together. Another way is to directly concatenate these features along the feature dimension and then project them to the hidden dimension. After that, the merged feature vector is fed to the Transformer encoder with AdaLN after each attention layer and feed-forward layer. Figure 4.9 illustrates the detail of the feature fusion process using both methods.



(a)



(b)

Figure 4.9.: Structure of the feature fusion process using (a): addition of the conditional features and embedding; (b): concatenation of the conditional features and embedding.

4.3.7. Residual Translation

Compared with the rotation estimation in the 6 DoF estimation task, translation is relatively easier to determine, at least in a rough range. We can use the position of the bounding box of the object in the camera coordinate combined with the camera parameter to calculate the coarse position of the object in the 3D space. In another way, we can directly use the mean of the point cloud's position to roughly approximate the translation if the depth information is available. The error exists mainly because the point cloud is sometimes not complete so the gravity center of the partial point cloud is not the real center of the object. That's also the reason for the error of the 2D only method that the occlusion causes a shift between the center of the visible part and the ground truth.

With the help of the deep learning method, we can minimize the error of the estimation when we use the gravity center of the reference point cloud as the initial translation. The model is trained to learn the residual translation from the gravity center to the ground truth center. First, we calculate the mean of the reference point cloud ($\overrightarrow{OQ}$ in figure 4.10) which is the initial translation and move the point cloud to the origin. Then we let the network minimize the residual error between the estimated translation and the shifted ground truth translation $\overrightarrow{OP'}$. The estimated residual translation is added to the initial translation $\overrightarrow{OQ}$ to get the final translation estimation $\overrightarrow{OP}$. This process basically enables the network to approximate the shift between the mean of the partial point cloud and its original center, instead of the position of the object in the 3D space. This modification reduces the solving space significantly compared with directly estimating the translation $\overrightarrow{OP}$ and makes the training more efficient.

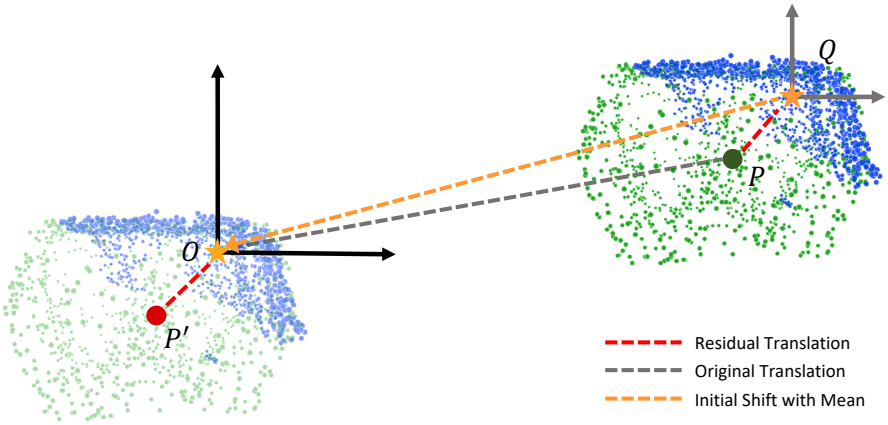


Figure 4.10.: Illustration of the residual translation estimation (projected in 2D space).

4.3.8. Multi-Hypotheses Inference

One of the advantages of the diffusion model is the high quality and diversity of the generated objects. Using the case of image synthesis as an example, we can derive dozens of high-resolution and meaningful images from the diffusion model with the same prompt. In our case, we want to first use the multi-hypotheses module to reduce the variance of the generated poses by taking the mean of all hypotheses as the final estimation and making the final result

more robust. In the feature work, we may sample more hypotheses to analyze the pose distribution of symmetrical objects in order to explore more possibilities when we deal with the ambiguity of the pose estimation.

4.3.9. Pose Refinement

To further refine the 6 DoF pose of the reference input, we can use the coarse pose estimation from the diffusion model as the initial pose and use some algorithms to refine the pose. Iterative and not learning-based methods like RANSAC [60] and ICP [133] are popular in the field of pose estimation. RANSAC, for example, uses repeated random sub-sampling to iteratively estimate the optimal result in the set of data containing outliers. The goal of RANSAC is to find a global optimal solution that excludes the outliers and contains the inliers as much as possible. This technique is widely used in correspondence-based point cloud registration. Our pose hypotheses diffusion model samples directly the estimated 6 DoF pose of the reference input and we utilize ICP to the point cloud transformed via the coarse pose estimation.

The ICP algorithm is also an iterative solution to the rigid point cloud registration problem. Given a set of source points A and a set of target points B, the goal of ICP is to find a rigid transformation $(\mathbf{R}, \mathbf{t})$ that minimizes the distance between the source points and the transformed target points. The first step is to find the corresponding points pair using nearest neighbor search. Let the set of corresponding pairs be:

$$C = \{(i, j) \mid \mathbf{a}_i \in A \text{ and } \mathbf{b}_j \in B \text{ are corresponding points}\} \tag{4.13}$$

After that, we use the sum of the squared distance of the corresponding point pairs as the error to be minimized. The minimization problem can be formulated as:

$$(\mathbf{R}^*, \mathbf{t}^*) = \underset{\mathbf{R},\mathbf{t}}{argmin} \sum_{(i,j)\in C} \|\mathbf{R}\mathbf{a}_i + \mathbf{t} - \mathbf{b}_j\|_2^2 \tag{4.14}$$

To solve this problem efficiently, we normally use the Singular Value Decomposition (SVD) which guarantees the accuracy and speed during a large number of iterations. The ICP algorithm is summarized in the following algorithm 3:

Theoretically, the ICP algorithm can converge to the global optimal solution if the initial pose is close enough to the ground truth pose. However, in practice, the ICP algorithm is sensitive to the initial pose and can easily get stuck in the local optimal solution. Because of that, we use the coarse pose estimation from the diffusion model as the initial pose and run the ICP algorithm within an acceptable number of iterations to have a good tradeoff of inference speed and accuracy.

Algorithm 3 Iterative Closest Point

Require: Source point cloud $A = \{\mathbf{a}_i\}_{i=1}^{N}$, target point cloud $B = \{\mathbf{b}_i\}_{i=1}^{N}$, maximum number of iterations k_{max}, threshold ϵ

Ensure: Rigid transformation $(\mathbf{R}, \mathbf{t})$

Initialize $(\mathbf{R}, \mathbf{t})$ with coarse pose estimation
for $k = 1$ to k_{max} **do**
    $A^* \longleftarrow \text{Trans}(A, (\mathbf{R}, \mathbf{t}))$
    $C = \{(i, j) \mid \mathbf{a}_i \in A \text{ and } \mathbf{b}_j \in B\} \longleftarrow \text{NearstSearch}(A^*, B)$
    $(\mathbf{R}, \mathbf{t}) \longleftarrow \text{SVD}(C)$
    $e = \frac{1}{|C|} \sum_{(i,j)\in C} \|\mathbf{R}\mathbf{a}_i + \mathbf{t} - \mathbf{b}_j\|_2^2$
    if $e < \epsilon$ **then**
        Break
    end if
end for

5. Experiments

In this section, we first introduce the datasets we use in our experiments and then we show the details of the training and evaluation of the pose hypotheses diffusion model and its feature extractors on the datasets. We compare our model with other methods and show the ablation study of our model.

5.1. Datasets

5.1.1. LINEMOD Dataset

The LINEMOD (LM) dataset [142] is a widely used dataset for the 6 DoF pose estimation. It has 15 objects containing ape, bench vise, bowl, camera, can, cat, cup, driller, duck, egg box, glue, hole puncher, iron, lamp and phone. These objects are texture-less with discriminative color, shape and size and are placed in a cluttered scene. As the training dataset, it provides for each object 1312 RGB-D images from different viewpoints annotated with the ground truth. These training data are based on the synthetic CAD Model without any occlusion and noise. There are also vividly rendered training data placed together in the scene to simulate the real scenario. The Benchmark for 6D Object Pose Estimation (BOP) [143] introduced the training data generated from an open-source, light-weight, procedural and photorealistic Physically Based Rendering (PBR) renderer BlenderProc [144] into the LM dataset. This set of PBR-BlenderProc4BOP training images are sorted in 50 different scenes with 50000 images in total which are rendered with different lighting, and occlusion. The test datasets are captured with the Kinect sensor. The following figures 5.1 show the training data and test data of the LM dataset.

Figure 5.1.: LIMEMOD dataset. Left: training data with Synthetic CAD Model; Middle: training data rendered with PBR-BlenderProc4BOP; Right: test data captured with Kinect Sensor.

5.1.2. LINEMOD-O Dataset

To evaluate the model on the occluded scene, Brachmann et al. [16] refined the bench-vise partition of the LM dataset with occluded objects and annotated ground truth pose under different light conditions to make the pose estimation more challenging. The meaning of the high-level occlusion data is to simulate the real scenario where the object is partially occluded by other objects so that the model can learn the robustness of the pose estimation and generalize better to the real-world data. The following figure 5.2 shows the low-level occluded scene and high-level occluded scene.

Figure 5.2.: Comparison of LMO dataset with LM dataset. Left: low-level occluded scene; Right: high-level occluded scene.

5.1.3. Data Format

In order to unify the data format of the different datasets for 6 DoF pose estimation task. We follow the data format of the BOP dataset. The structure of the dataset is shown in the following table 5.1.

Clarification of the naming in table 5.1:

- models[_MODELTYPE]_eval: "Uniformly" resampled and decimated 3D object models used for calculation of errors of object pose estimates.
- MODELTYPE, TRAINTYPE, VALTYPE and TESTTYPE are optional and used if more data types are available (e.g. images from different sensors).

Camera parameters in scene_camera.json:

- cam_K - 3x3 intrinsic camera matrix K (saved row-wise).
- depth_scale - Multiply the depth image with this factor to get depth in mm.
- cam_R_w2c (optional) - 3x3 rotation matrix R_w2c (saved row-wise).
- cam_t_w2c (optional) - 3x1 translation vector t_w2c.
- view_level (optional) - Viewpoint subdivision level.

Ground truth annotation in scene_gt.json:

- obj_id - Object ID.
- cam_R_m2c - 3x3 rotation matrix R_m2c (saved row-wise).

Table 5.1.: Dataset Structure of BOP Format.

Directory	Contents
DATASET_NAME	
|— camera[_TYPE].json	Camera parameters
|— dataset_info.json	Dataset-specific information
|— test_targets_bop19.json	Test targets for the BOP Challenge
|— models[_MODELTYPE][_eval]	
| |— models_info.json	Dimension and symmetry
| |— obj_OBJ_ID.ply	3D file of the object
|— train|val|test[_TYPE]	Training/Validation/Test images
| |— SCENE_ID|OBJ_ID	
| | |— scene_camera.json	Camera parameters of the scene
| | |— scene_gt.json	Ground truth annotation
| | |— scene_gt_info.json	Meta information
| | |— depth	Depth images
| | |— mask	Masks of object silhouettes
| | |— mask_visib	Masks of the visible parts of object silhouettes
| | |— rgb|gray	Color/gray images

- cam_t_m2c - 3x1 translation vector t_m2c.

Meta information of the ground truth in scene_gt_info.json:

- bbox_obj - 2D bounding box of the object silhouette given by (x, y, width, height), where (x, y) is the top-left corner of the bounding box.
- bbox_visib - 2D bounding box of the visible part of the object silhouette.
- px_count_all - Number of pixels in the object silhouette.
- px_count_valid - Number of pixels in the object silhouette with valid depth.
- px_count_visib - Number of pixels in the visible part of the object silhouette.
- isib_fract - The visible fraction of the object silhouette.

5.1.4. Data Augmentation

Although the quantity of the training data from the BlenderProc is large enough, we still need some data augmentation tricks to make the model more robust against noise and invalid data. The data augmentation for the point cloud in our case is firstly the random rotation of the partial point cloud and CAD model with the modification of the ground truth transformation. We add noise to points with a Gaussian distribution scaled by a factor k_n related to the scale of the objects to simulate the real-world data captured by the depth sensor. The symmetry of the object is also considered, and we generate extra data by flipping or rotating the symmetrical CAD models with the corresponding ground truth.

To remove the noise and outliers in the point cloud. We use the statistical outlier filter in the data preprocessing stage. The statistical outlier filter is an algorithm that removes outliers

from a point cloud based on the statistical analysis of point neighborhoods. The algorithm removes points that are further away from their neighbors compared to the average for the point cloud. The filter is controlled by the number of neighbors n_{nb} and the standard deviation multiplier σ.

After that, the point cloud is then downsampled to 1000 points to reduce the computational cost but still maintain the geometry information of the object. The whole original training dataset is divided into the training part (90%) and validation part (10%) randomly and together with the separate test dataset we get the dataset we preprocessed for the training and evaluation of our models.

As the 2D image is combined with 3D point cloud in the pose estimation task and we don't need to train the 2D feature extractor, we only use the data augmentation of the random rotation during the training of the 3D feature extractor. Because the 2D image with 3D depth uniquely defines uniquely the 6 DoF pose of the object. So during the training of the pose estimation task we don't let orginal point cloud rotate, but still add noise and consider the symmetry of the object.

5.2. Implementation Details

In this section, we first present the implementation of the 2D and 3D feature extractors and the corresponding evaluations on the module level. Then we demonstrate the training details of our diffusion model.

5.2.1. 2D Feature Extractor

Pretrained Model

DINO, as our 2D feature encoder, provides many pre-trained weights of the backbone with different numbers of parameters and backbone architecture which are pre-trained on ImageNet for general image downstream tasks. The evaluation of the self-supervised architecture uses the linear and k-Nearest Neighbors (k-NN) classifier on the feature extracted from the model. The following table 5.2 shows the evaluation results of the different pre-trained backbones on the ImageNet dataset according to the original paper [131].

Table 5.2.: Evaluation of the pre-trained DINO model on ImageNet.

Backbone	#Params.	Dim.	#Layers	#Heads	Patch Size	k-NN	Linear
ViT-S/16	21M	384	12	6	16	74.5	77.0
ViT-S/8	21M	384	12	6	8	78.3	79.7
ViT-B/16	85M	768	12	12	16	76.1	78.2
ViT-B/8	85M	768	12	12	8	77.4	80.1

After we register the pre-trained 2D backbone to our model, we need to feed the image with some preprocessing into the ViT. The image is first cropped to a bounding box with

the object we want to estimate its pose. Then we resize the cropped image to the size of 224x224 and normalize the image with the mean and standard deviation of the ImageNet dataset, while the ViT is pre-trained on the ImageNet. The mean and standard deviation of the RGB channels are $(0.485, 0.456, 0.406)$ and $(0.229, 0.224, 0.225)$ respectively. After that, the image is processed following the ViT architecture and as output, we get the feature we can use in the pose estimation task.

The 2D encoder needs to be lightweight since it is just a downstream task for our diffusion model, so in our setup we don't use the backbone with a very deep network and a large number of parameters, e.g.,ViT-L or ViT-G. It needs also to be well generalized because the image in our case is not seen during the training of the 2D encoder. ImageNet used in DINO contains more than 1.4 million annotated images from different categories, which has the capability to let the network learn the latent feature expression from the majority of real-life objects.

Evaluation

Since we use the 2D feature not for the classification or segmentation tasks, but as the input for the pose estimation where the feature is the latent expression of pose that is hard to quantitatively evaluate, we evaluate the influence of the 2D feature extractor together with the pose estimation diffusion model later. Instead, the attention map of the input image can be visualized to indirectly evaluate the network and validate whether the Transformer can grasp the important part of the image. The attention map is the output of the last attention layer in the Transformer encoder. Figure 5.3 shows the attention maps of 3 objects as examples fed to the ViT.

By Comparing the outputs from a different scale of ViT models and patch size, we can find that the patch size significantly determines the attention map. Because the object we dealing with has relatively simple geometry and some sometimes textureless, the transformer can not encode the image in a very accurate way for the pose difference if the patch size is too big. The following figure 5.4 shows the attention maps of the same image fed to the ViT with different patch sizes and scales of ViTs.

From the figure 5.4 we can see that the key points of the cat e.g., ears, tail and eyes are not precisely localized if the patch size is not small enough. For the classification task is this drawback not that obvious because the network can learn the latent feature from the majority of the image. But for the pose estimation task, the network needs to learn the latent feature from the key points or edges of the object, which is more sensitive to the resolution.

Considering the tradeoff of the inference speed and performance, we use ViT-B/8 in our experiments which has a quite satisfied performance and a relatively acceptable number of parameters. The ablation study of different setups will be shown later in this experiments section.

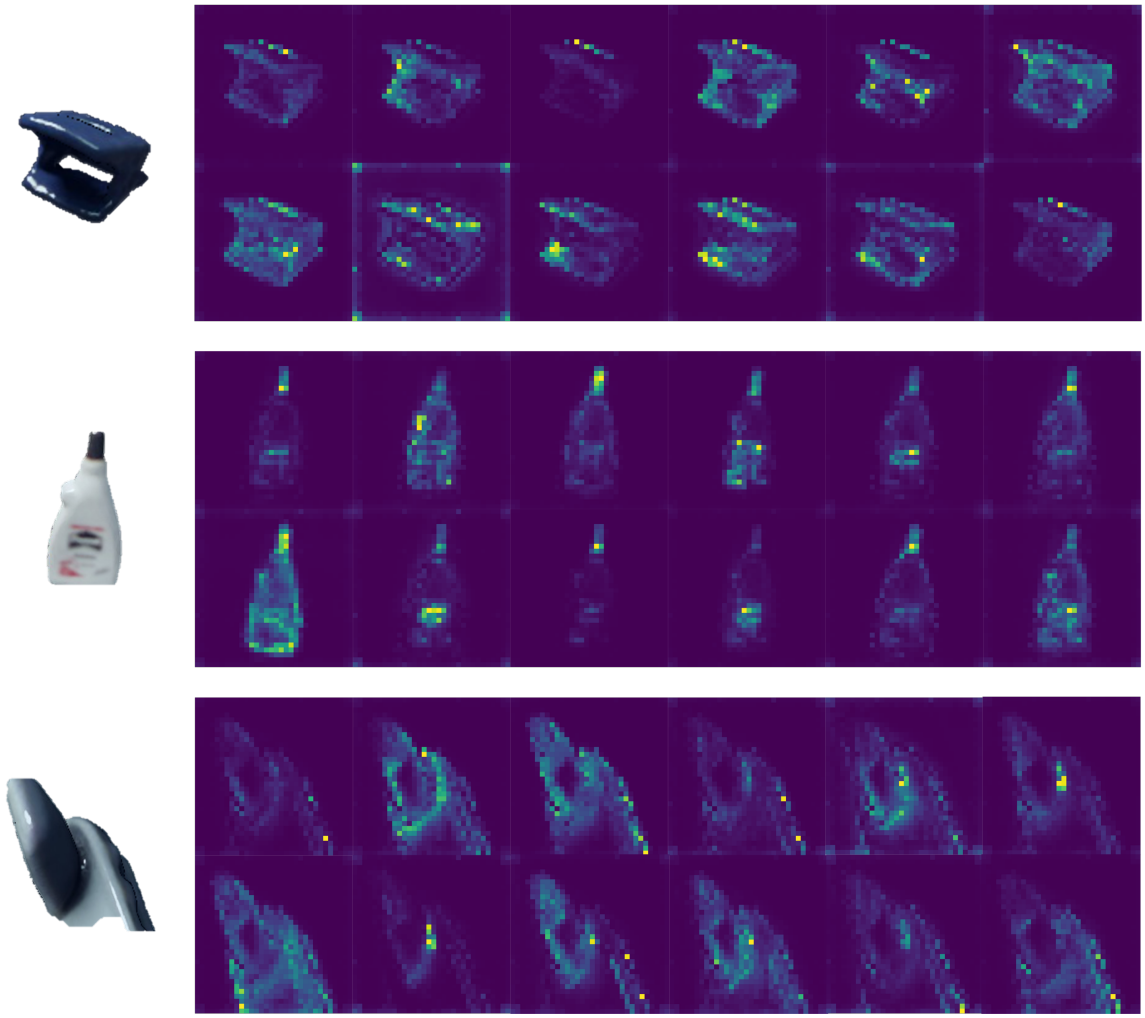


Figure 5.3.: Visualization of the attention maps of the input image, where 12 heatmaps represent 12 attention heads output in the Transformer. Top: hole puncher; Middle: glue; Bottom: phone (occluded).

5.2.2. 3D Feature Extractor

Training

The 3D feature extractor is trained using a point cloud completion task. The reference partial point cloud is fed to the FoldingNet and the output is aligned with the CAD model transformed with the ground truth pose as the previous figure 4.8 shows. The loss function is the bi-directional CD between the output point cloud and the transformed CAD model (see equation 4.12).

The model is trained on a single Nvidia Tesla V100-SXM2 video card with 16GB memory. We use the BlenderProc4BOP training dataset for LM which contains 50k images with 15 objects and we can further crop several objects from each image. To that end, we finally have more than 560k separate partial point clouds for training. After the tuning of the hyperparameters, we use the batch size of 32 and the Adaptive Moment Estimation (Adam) optimizer with the cosine annealing learning rate scheduler. The initial learning rate is $1e^{-4}$ and the

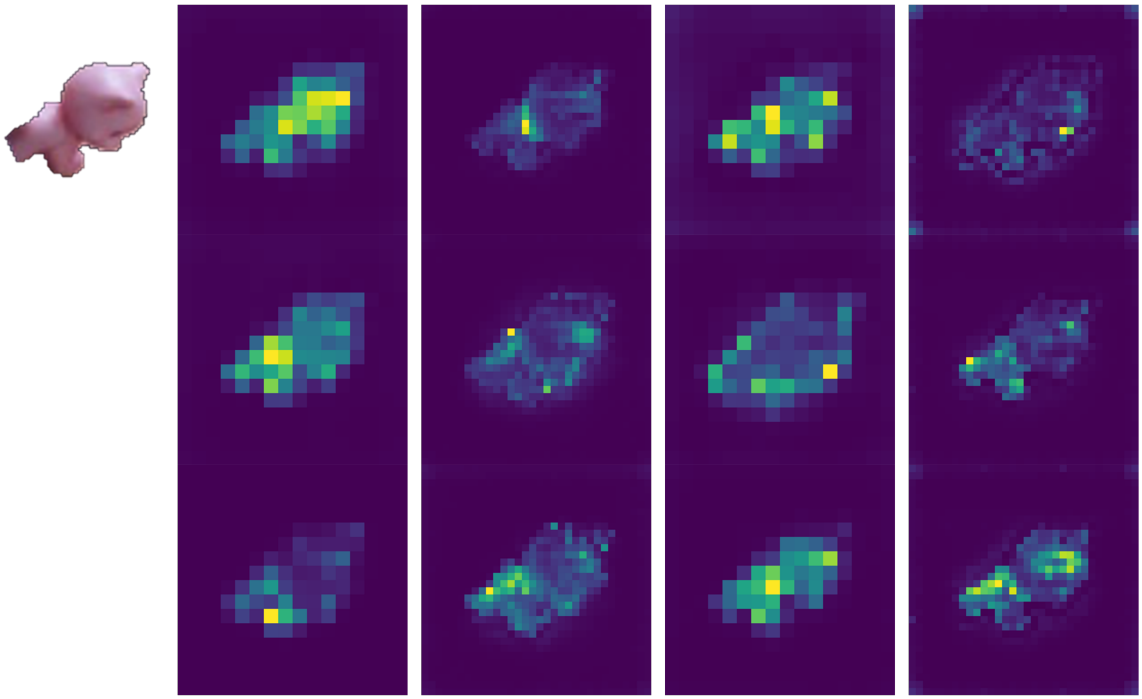


Figure 5.4.: Attension maps of the different scales of ViTs and patch size. From left to right: ViT-S/16, ViT-S/8, ViT-B/16, ViT-B/8. Each column represents 3 random attention heads.

end-up learning rate (lr) is $1e^{-6}$. The model-specific hyperparameter is the number of the folding MLP layers, which is by default set to 2 in our case. The number of k-NN is set to 64 and the number of the graph-based local maxpooling layers is set to 2. The reconstructed point number is set to 2025, whose square root is an integer responding to the size of the 2D grid. The dimension of the latent feature is set to 512 for the downstream task. The model is trained for 200k iteration which is about 114 epochs.

Evaluation

The evaluation metric of the point cloud completion task is the CD between the output point cloud and the ground truth point cloud. The following graph 5.5 shows the training and validation loss of the model during the whole training process.

We stop the training after 200k iterations to prevent the overfitting problem, because the validation loss is not decreasing anymore. And the distribution of each category's CD in the test dataset is shown in figure 5.6.

With the visualization of the reconstructed point cloud (figure 5.7), we can validate that the model has the capability to localize the partial point cloud and complete the missing part of the object. The implicit expression of the 6 DoF poses as well as the object class is learned by the model without giving the class label so that we can use the feature to represent the pose and object class of the reference input in the diffusion model.

The following figure 5.8 shows the development of the output from the FoldingNet decoder during the training process.

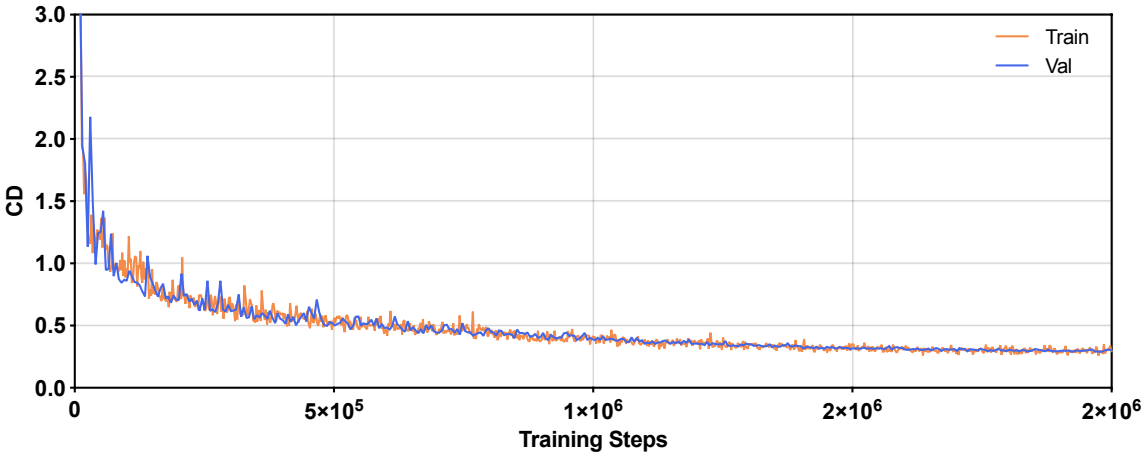


Figure 5.5.: CD of the training and validation dataset during the point completion pretraining.

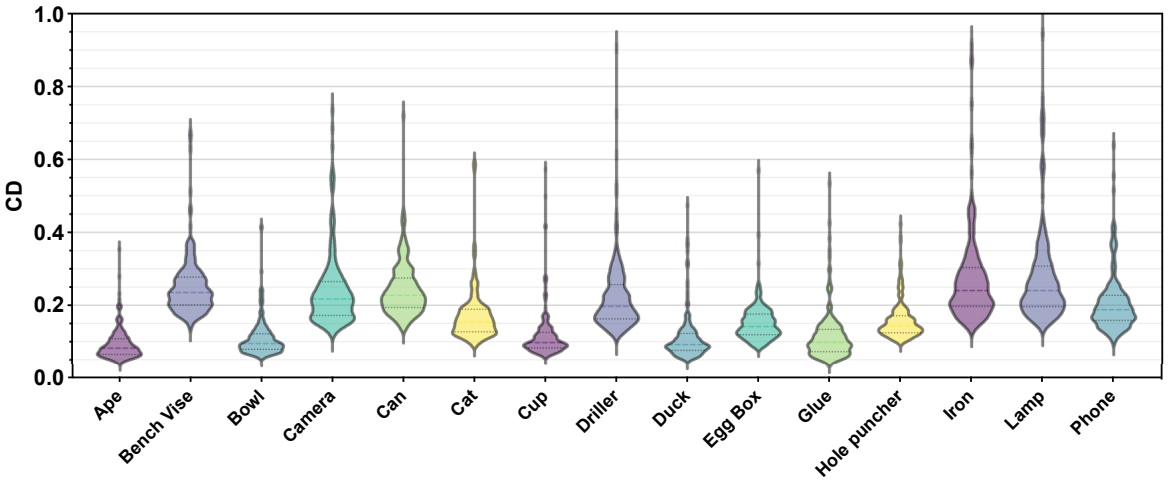


Figure 5.6.: Categorical level CD distance distribution of the test dataset.

From the visualization, we can find that the shape of the reconstructed point cloud is getting more and more similar to the ground truth, but the difference exists because of the capability of the backbone on data with heavy occlusion and noise. We further compare the training of point cloud completion tasks on all 15 categories with overfitted models on a single object to validate the generalization of the network. From figure 5.9 we can find that the CD of the overfitted models has a lower mean and variance than the model trained on all categories. Table 5.3 shows the quantitative comparison.

To evaluate the influence of the number of k-NN on the result, we train the model with different setups and the following table 5.4 shows the corresponding CDs on the test dataset.

The number of the k-NN determines the receptive field of the local graph-based max-pooling layer. The local feature can be more robust to the noise and outliers if the number of k-NN is larger. However, the model can not learn the global feature of the object if the number of k-NN is too large. In our experiments of the following diffusion model, we set the number to 64, which is proven to be optimal within the objects of our dataset.

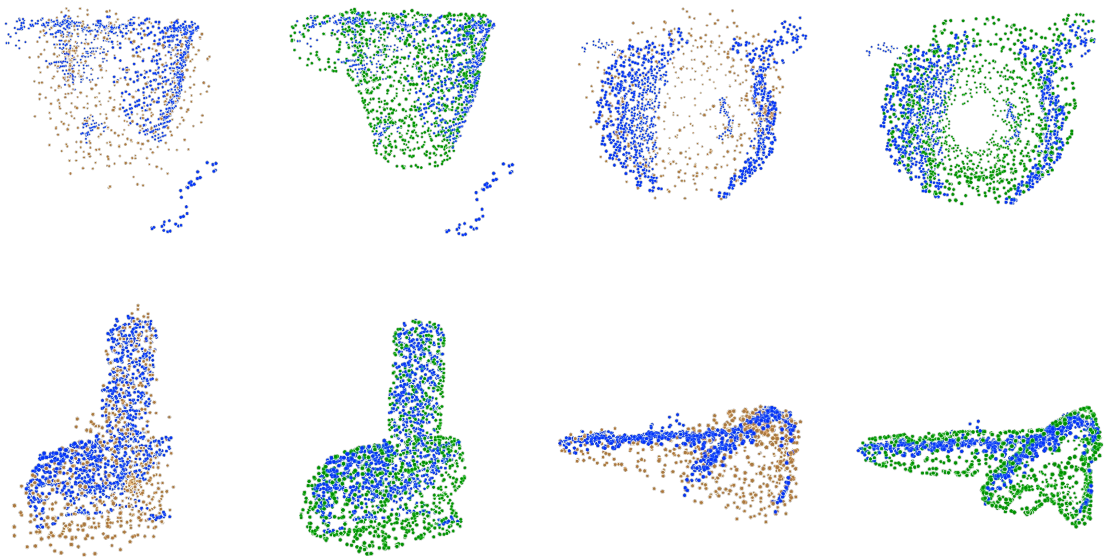


Figure 5.7.: Point cloud completion results of the FoldingNet. Top: cup; Bottom: phone. From left to right: estimation(viewpoint 1), ground truth(viewpoint 1), estimation(viewpoint 2), ground truth(viewpoint 2). Blue points are the input partial point cloud.

Figure 5.8.: Development of the reconstructed point cloud during the training process.

5.2.3. Diffusion Models

The training of the diffusion network for the pose hypotheses estimation can be done after the 2D and 3D feature encoders are pre-trained (for 2D feature extractor is the checkpoint of the pre-trained model loaded from the public cloud storage). We load the checkpoint of the encoders and freeze the network of these two parts to avoid the parameters of the pre-trained model being updated during the training of the diffusion network.

The training is also running on the single Nvidia Tesla V100-SXM2 16G video card. Like the setup of the 3D encoder, we use the BlenderProc4BOP training dataset and divide the original data into 90%/10% partitions for training and validation. After tuning the hyperparameters from the experiments, we use the following setup (table 5.5) to represent the pose hypotheses diffusion model.

Clarification of some hyperparameters in table 5.5:

- CA - Cosine annealing learning rate scheduler.

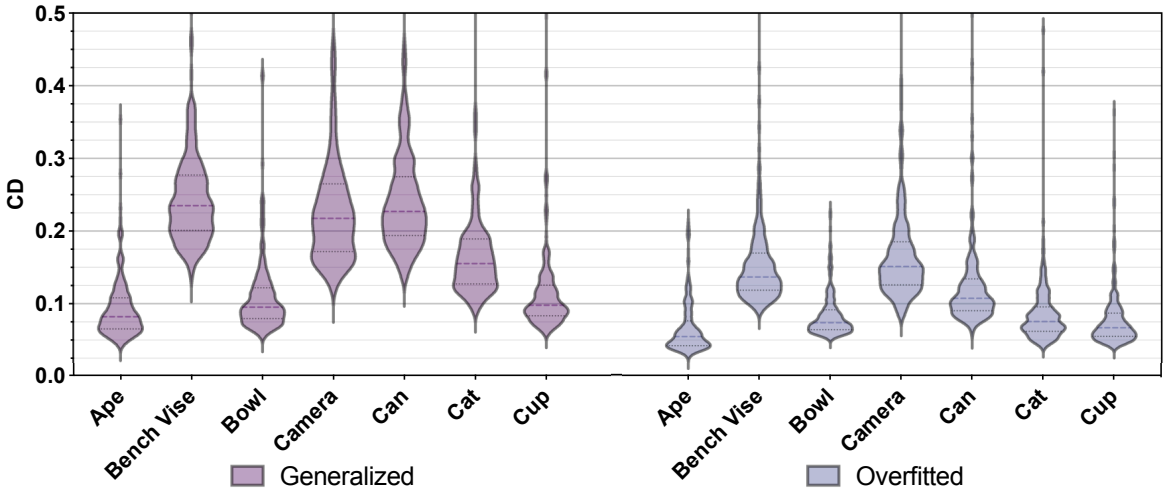


Figure 5.9.: Comparision of the CD distributions between the model trained with the whole data set and the overfitted model on a single object (the first 7 objects of LM are shown).

Table 5.3.: Mean and standard deviation of the CD with the model trained on the whole dataset and the overfitted model on a single object (7 of 15 objects selected).

Trained on the complete LM								
Category	Ape	Vise	Bowl	Cam.	Can	Cat	Cup	Avg.
Mean	0.093	0.251	0.109	0.241	0.243	0.171	0.117	0.175
Std.	0.042	0.079	0.050	0.103	0.073	0.071	0.066	0.069
Overfitted on single object								
Category	Ape	Vise	Bowl	Cam.	Can	Cat	Cup	Avg.
Mean	0.067	0.154	0.084	0.166	0.129	0.087	0.079	0.109
Std.	0.036	0.061	0.031	0.062	0.076	0.047	0.044	0.051

- β_1 - Initial value of the variance schedule at time step 1.
- β_T - Final value of the variance schedule at time step T.
- Scheduler (Diffusion) - Variance scheduler of the forward diffusion process which represents the change of β with the time step t.
- Σ_θ - Diagonal variance of the reverse process which is fixed in the original paper [2] but can be optimized to be learnable [32].
- Rotation - Representation of the rotation matrix for 6 DoF pose. We use the 6D continuous representation in our case. Recall the section 2.1.2.
- Translation - Instead of directly estimating the absolute translation, we let the network learn the residual of the translation using the mean of the partial point cloud as the reference point.
- #layers - Number of the multi-head attention layers.

Table 5.4.: CD with different number of k-NN of the FoldingNet.

#k-NN	8	16	32	64
CD	0.215	0.206	0.208	0.193

Table 5.5.: Hyperparemeters of the pose hypotheses diffusion model.

Training param.	Conf.	Diffusion	Conf.	Backbone	Conf.
Batch size	16	Steps	400	Model	Transformer
Optimizer	Adam	β_1	$1e^{-4}$	#Layers	8
Scheduler	CA	β_T	$2e^{-2}$	#Heads	4
Initial lr.	$1e^{-4}$	Scheduler	Linear	Hidden dim.	512
End-up lr.	$1e^{-6}$	Σ_θ	Learnable	FF dim.	2048
#Iterations	200k	Rotaion	6D	TE dim.	256
		Translation	Residual	PE dim.	512
		Loss func.	MSE	2D dim.	768
				3D dim.	512
				Feat. fusion	Add + adaLN

- #heads - Number of the attention heads.
- Hidden dim. - Dimension of the hidden layer in the Transformer encoder.
- FF dim. - Dimension of the feed-forward layers in the Transformer encoder, which is chosen to be 4 times the hidden dim.
- TE dim. - Dimension of the time embedding in the diffusion model.
- PE dim. - Dimension of the position embedding of the input tokens for the Transformer encoder.
- 2D dim. - Dimension of the 2D feature extracted from the pretrained 2D encoder.
- 3D dim. - Dimension of the 3D feature extracted from the pretrained 3D encoder.
- Feat. fusion - The operation of the feature fused into the backbone. In our case, we first use the addition of the 2D, 3D features as well as the time embedding and then utilize the AdaLN to integrate the features into the network. Recall the section 4.3.6.

The loss function is defined by the Mean Squared Error (MSE) between the noise ϵ added at a particular time step t and the estimated noise ϵ_θ. The loss function is formulated as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{B} \sum_{t \in \mathcal{T}_B} \|\epsilon_t - \epsilon_\theta(\mathbf{x}_0, \epsilon, t, y)\|^2 \quad , \mathcal{T}_B \subseteq \mathcal{T} = \{1, \ldots, T\} \tag{5.1}$$

where B is the batch size, $\mathcal{T}_B$ is the subset of the whole time steps with the batch size and ϵ_θ is the estimated noise depending on the input $\mathbf{x}_0$, the noise ϵ, the time step t and the condition y (see section 2.2.2). The loss is calculated with the MSE of the randomly sampled subset controlled by the batch size instead of the whole time steps, but it will cover each time step during the training process. Figure 5.10 shows the loss of the model during the training process.

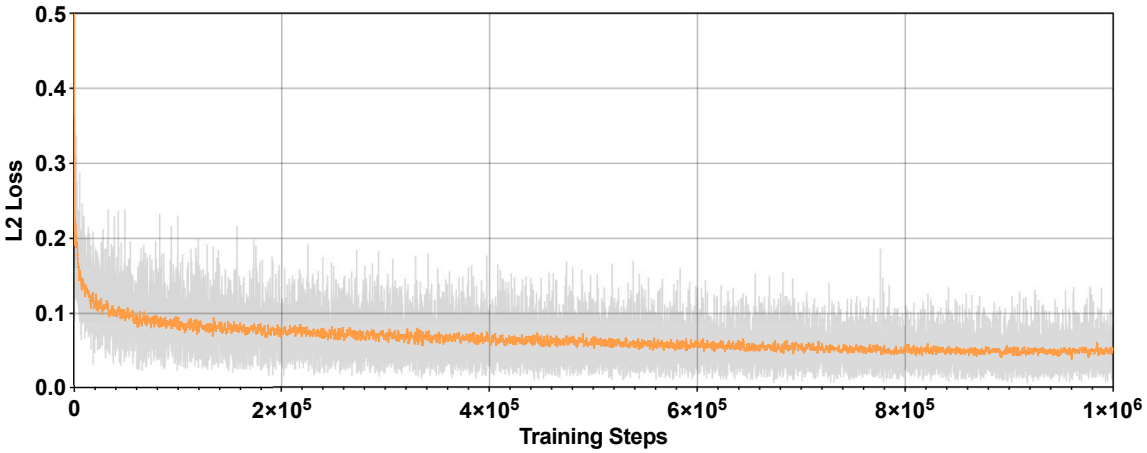


Figure 5.10.: L2 loss of the training phase of the diffusion model. Original data (gray) is downsampled and smoothed with a 2^{nd} order filter (orange) for visualization.

5.3. Evaluation

5.3.1. Evaluation Metrics

No matter which form of the rotation representation we use during the training phase that is introduced in section 2.1.2, we transform it back to the rotation matrix $\mathbf{R}$ to evaluate the pose estimation together with the translation, in order to meet the requirement of the data format of BOP. The pose is represented by a 4x4 matrix $\mathbf{T} = [\mathbf{R}, \mathbf{t}, \mathbf{0}, 1]$, where $\mathbf{R}$ is a 3x3 rotation matrix and $\mathbf{t}$ is a 3x1 translation vector. As evaluation metrics, we adopt the same metrics used in BOP Challenge 2019/2020/2022/2023 [143, 145]. The following metrics are used to evaluate the pose estimation.

Visible Surface Discrepancy (VSD):

$$e_{\text{VSD}}\left(\hat{D}, \bar{D}, \hat{V}, \bar{V}, \tau\right) = \text{avg}_{p \in \hat{V} \cup \bar{V}} \begin{cases} 0 & \text{if } p \in \hat{V} \cap \bar{V} \wedge \left|\bar{S}(p) - \hat{S}(p)\right| < \tau \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

An object model is rendered in two poses: the estimated pose $\hat{\mathbf{P}}$ and the ground truth pose $\bar{\mathbf{P}}$. And the result of the rendering is two distance maps $\hat{S}$ and $\bar{S}$. The distance maps are compared with the distance map S_I of the test image to calculate the visibility masks $\hat{V}$ and $\bar{V}$. The error is controlled by a misalignment tolerance τ.

Maximum Symmetry-Aware Surface Distance (MSSD):

$$e_{\text{MSSD}}\left(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_M, V_M\right) = \min_{\mathbf{S} \in S_M} \max_{\mathbf{x} \in V_M} \left\|\hat{\mathbf{P}}\mathbf{x} - \bar{\mathbf{P}}\mathbf{S}\mathbf{x}\right\|_2 \tag{5.3}$$

where S_M is the set of all symmetry transformations of the object model and V_M is the set of model vertices. The MSSD is the minimum of the maximum symmetric-aware surface distance between the estimated pose $\hat{\mathbf{P}}$ and the ground truth pose $\bar{\mathbf{P}}$. The maximum distance between the model vertices is important for robotic applications, where the maximum surface deviation strongly indicates the chance of a successful grasp.

Maximum Symmetry-Aware Projection Distance (MSPD):

$$e_{\text{MSPD}}\left(\hat{\mathbf{P}}, \bar{\mathbf{P}}, S_M, V_M\right) = \min_{\mathbf{S}\in S_M} \max_{\mathbf{x}\in V_M} \left\| \text{proj}\left(\hat{\mathbf{P}}\mathbf{x}\right) - \text{proj}\left(\bar{\mathbf{P}}\mathbf{S}\mathbf{x}\right) \right\|_2 \tag{5.4}$$

The proj(·) is the 2D projection which results in pixels and other variables that are identical to the MSSD. The MSPD considers global object symmetries and replaces the average by the maximum distance to increase robustness against the sampling of the object model compared with the previous method [146].

After the error functions are determined, we use the threshold of the error to define the correctness of the pose estimation. The pose is correctly estimated if the error $e < \theta_e$, where $e \in \{e_{\text{VSD}}, e_{\text{MSSD}}, e_{\text{MSPD}}\}$ and θ_e is the threshold of correctness.

AR_{VSD} is the average recall rates calculated for the misalignment tolerence τ from 5% to 50% of the object diameter with a step of 5%, and the threshold of correctness θ_{VSD} ranging from 0.05 to 0.5 with a step of 0.05. AR_{MSSD} is the average recall rate calculated for the threshold of correctness θ_{MSSD} ranging from 0.05 to 0.5 of the object diameter with a step of 0.05. AR_{MSPD} is the average recall rate calculated for the threshold of correctness θ_{MSPD} ranging from $5r$ to $50r$ with a step of $5r$, where $r = w/640$ and w is the image width in pixels. The average recall rate is defined as:

$$\text{AR} = \left(\text{AR}_{\text{VSD}} + \text{AR}_{\text{MSSD}} + \text{AR}_{\text{MSPD}}\right)/3 \tag{5.5}$$

The most popular metrics for pose estimation have been the Average Distance (ADD) and the Average Distance for Symmetrical Objects (ADD-S) [142]. The ADD is the average distance between the model vertices transformed with the estimated pose and the model vertices transformed with the ground truth pose and the ADD-S is the version considering the pose ambiguity of the rotationally symmetric object. ADD and ADD-S are defined as:

$$\text{ADD} = \frac{1}{|V|} \sum_{\mathbf{x}\in V} \left\| \hat{\mathbf{P}}\mathbf{x} - \bar{\mathbf{P}}\mathbf{x} \right\|_2 \tag{5.6}$$

$$\text{ADD-S} = \frac{1}{|V|} \sum_{\mathbf{x}_1\in V} \min_{\mathbf{x}_2\in V} \left\| \hat{\mathbf{P}}\mathbf{x}_1 - \bar{\mathbf{P}}\mathbf{x}_2 \right\|_2 \tag{5.7}$$

where V is the set of model vertices. It has some limitations that ADD(-S) comes from a high dependence on the geometry of the object model and the sampling density of its surface, i.e. the average distance is dominated by higher-frequency surface parts such as the thread of a fuse. The maximum distance used in MSSD and MSPD is less dependent on the geometry and sampling of the object model [145].

5.3.2. Pose Estimation Results

We evaluate our model on the test dataset of BOP challenge [143] with overall 3000 images and use the segmentation result from the ground truth as our mask to filter out the background, since we only focus on the pose estimation with an object already detected. In this section, we represent both quantitative results and qualitative estimation of given input. Our method is also compared with the state-of-the-art methods on this task.

Quantitative Evaluation

Following the metrics that we introduced in the previous section, we propose comprehensive evaluations on each object of both LM and LMO datasets. The corresponding results of all categories are shown in table 5.6 and 5.7. In figure A.1 we can further validate the capability of our model facing each object with different thresholds of correctness. The object like cup, whose AR of BOP metrics and ADD(-S) is relatively lower, is hard to estimate. This is because of the ambiguity caused by the handle when it is not visible and this identification part only takes a small partition in the whole object. Besides that, our model achieves quite satisfying results even when dealing with the object with occlusion.

Table 5.6.: Quantitative Evaluation on LM dataset(symmetrical objects annotated with *).

	AR_{MSPD}		AR_{MSSD}		AR_{VSD}		AR		ADD(-S)	
Refinement	–	√	–	√	–	√	–	√	–	√
Ape	85.6	93.9	73.9	87.6	58.7	80.4	72.7	87.3	73.5	92.0
Bench vi.	78.8	87.3	85.2	89.8	61.3	74.7	75.1	83.9	93.0	93.0
Bowl*	80.3	83.6	77.6	82.7	42.9	60.6	66.9	75.7	99.0	99.5
Camera	73.7	82.8	73.0	82.3	57.7	73.2	68.1	79.4	70.0	81.0
Can	73.8	83.1	76.2	84.6	54.5	71.5	68.1	79.8	82.0	86.5
Cat	88.2	97.2	86.1	95.9	60.3	83.4	78.2	92.2	96.0	97.0
Cup	53.7	59.5	46.9	53.8	30.0	51.2	43.5	54.8	31.5	51.5
Driller	78.2	85.6	87.0	92.3	60.4	77.3	75.2	85.1	95.5	95.0
Duck	77.5	85.4	67.0	78.2	53.5	71.7	66.0	78.4	66.5	81.0
Egg box*	65.9	72.0	64.2	70.3	59.0	68.4	63.0	70.2	99.0	99.0
Glue*	72.6	79.8	70.1	77.6	46.0	62.3	62.9	73.2	92.0	93.0
Hole pu.	68.4	73.9	62.8	69.3	38.7	49.9	56.6	64.3	53.5	67.0
Iron	79.5	84.4	88.3	91.9	67.9	77.2	78.6	84.5	93.5	95.0
Lamp	74.9	83.6	86.1	90.5	59.7	71.9	73.6	82.0	91.0	91.5
Phone	83.7	89.2	87.0	92.3	61.5	74.7	77.4	85.4	95.0	96.0
Mean	75.6	82.8	75.4	82.6	54.1	69.9	68.4	78.4	82.1	87.9

Table 5.7.: Quantitative Evaluation on LMO dataset(symmetrical objects annotated with *).

	AR_{MSPD}		AR_{MSSD}		AR_{VSD}		AR		ADD(-S)	
Refinement	–	√	–	√	–	√	–	√	–	√
Ape	68.1	75.3	56.4	66.6	43.4	54.5	56.0	65.4	54.4	76.5
Can	55.4	64.3	58.0	65.6	36.3	54.2	49.9	61.4	61.0	68.7
Cat	55.8	59.4	50.9	54.9	33.5	43.9	46.7	52.7	75.7	88.3
Driller	64.9	68.6	73.0	74.6	43.4	56.9	60.4	66.7	73.6	74.1
Duck	41.3	45.7	34.6	40.7	37.7	49.6	37.9	45.3	35.8	47.0
Egg box*	27.9	29.8	20.5	22.4	21.8	24.2	23.4	25.5	89.8	89.8
Glue*	51.4	57.4	52.1	57.4	32.7	43.9	45.4	52.9	88.5	89.4
Hole pu.	51.6	57.3	46.7	53.5	31.3	42.5	43.2	51.1	32.5	47.7
Mean	52.1	57.2	49.0	54.5	35.0	46.2	45.4	52.6	63.9	72.7

Comparing with other 6 DoF pose estimation methods in recent years (table 5.8, 5.9) including RGB-only and RGB-D approaches, our model owns comparable performance, especially on LMO dataset. It is easy to notice the significant AR drop between the result on LM and

LMO of some methods, which is even more than 40%. Our method however has an acceptable score with and without refinement. Why we only use ADD(-S) as the evaluation metric in this part of the comparison, is because the majority of pose estimation methods only provide ADD(-S) in publications and the result of BOP metrics are not always available. Our ablation study based on all the metrics of BOP is shown in the next section.

Table 5.8.: Comparison of ADD(-S) scores with other methods on LM dataset (symmetrical objects annotated with *).

Method	Pix2-Pose [91]	PVNet [92]	Hybrid-Pose [95]	DPOD [72]		DF [93]		FFB6D [94]	Ours	
Data	RGB					RGB-D				
Refinement	√	–	√	–	√	–	√	–	–	√
Ape	58.1	43.6	63.1	53.3	87.7	79.5	92.3	98.4	73.5	92.0
Bench vi.	91.0	99.9	99.9	95.3	98.5	84.2	93.2	100	93.0	93.0
Camera	60.9	86.9	90.4	90.4	96.1	76.5	94.4	99.9	70.0	81.0
Can	84.4	95.5	98.5	94.1	99.7	86.6	93.1	99.8	82.0	86.5
Cat	65.0	79.3	89.4	60.4	94.7	88.8	96.5	99.9	96.0	97.0
Driller	76.3	96.4	98.5	97.7	98.8	77.7	87.0	100	95.5	95.0
Duck	43.8	52.6	65.0	66.0	86.3	76.3	92.3	98.4	66.5	81.0
Egg box*	96.8	99.2	100	99.7	99.9	99.9	99.8	100	99.0	99.0
Glue*	79.4	95.7	98.8	93.8	96.8	99.4	100	100	92.0	93.0
Hole pu.	74.8	81.9	89.7	65.8	86.9	79.0	92.1	99.8	53.5	67.0
Iron	83.4	98.9	100	99.8	100	92.1	97.0	99.9	93.5	95.0
Lamp	82.0	99.3	99.5	88.1	96.8	92.3	95.3	99.9	91.0	91.5
Phone	45.0	92.4	94.9	74.2	94.7	88.0	92.8	99.7	95.0	96.0
Mean	72.4	86.3	91.3	83.0	95.2	86.2	94.3	99.7	82.1	87.9

Table 5.9.: Comparison of ADD(-S) scores with other methods on LMO dataset (symmetrical objects annotated with *).

Method	Pix2-Pose [91]	PVNet [92]	Hybrid-Pose [95]	GDR-Net [85]	Zebra-Pose [74]	FFB6D [94]	Ours	
Data	RGB					RGB-D		
Refinement	√	–	√	–	√	–	–	√
Ape	22.0	15.8	20.9	46.8	57.9	47.2	54.4	76.5
Can	44.7	63.3	75.3	90.8	95.0	85.2	61.0	68.7
Cat	22.7	16.7	24.9	40.5	60.6	45.7	75.7	88.3
Driller	44.7	65.7	70.2	82.6	94.8	81.4	73.6	74.1
Duck	15.0	25.2	27.9	46.9	64.5	53.9	35.8	47.0
Egg box*	25.2	50.2	52.4	54.2	70.9	70.2	89.8	89.8
Glue*	32.4	49.6	53.8	75.8	88.7	60.1	88.5	89.4
Hole pu.	49.5	39.7	54.2	60.1	83.0	85.9	32.5	47.7
Mean	32.0	40.8	47.5	62.2	76.9	66.2	63.9	72.7

It is worth mentioning that our model is not trained on each object separately, so the result can be further improved if we train the model overfitted to each object with the cost of inflexibility. Table 5.10 shows the difference of the AR with a generalized model and an object-specific one evaluating on LMO.

Table 5.10.: AR drop between generalized model and overfitted one on LMO dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
General.	–	55.8	50.9	33.5	46.7	75.7
Overfit.	–	56.8	51.1	37.4	48.5	86.5
General.	√	59.4	54.9	43.9	52.7	88.3
Overfit.	√	61.2	56.6	46.6	54.8	93.7

Qualitative Evaluation

Here we also demonstrate the qualitative result of our model on the pose estimation task by using the visualization toolkit from BOP [143] challenge. Figure 5.11 and A.5 show some objects from LM and LMO dataset rendered with the estimated pose on the input image. The first row is the objects without occlusion and the second row is the objects with occlusion. We can observe that the 2D-projection of the estimated pose has a very high percentage of overlap with the ground truth pose if there is no occlusion. The highlighted estimation and object in the background are shifted a little bit when the visible part of the object is occluded by other objects.

Figure 5.11.: Visualization of some objects highlighted at estimated pose. First row: objects without occlusion; Second row: objects with occlussion.

Additionally, we can have a closer look at the development of the pose hypotheses through the whole denoising process (figure 5.12). We select the cat object in LM dataset as an example and render the visualization with Blender. The multi-hypotheses (set to 16 in our experiment) of the estimated pose, which is colorized with transparent gray, localize randomly in the space at the early stage of the denoising process and finally overlap together on the ground truth (purple). More examples of the visualization can be found in the appendix A.6, A.7.

Figure 5.12.: Visualization of the 6 DoF pose denoising process.

5.4. Ablation Study

In this section, the ablation study of our method is proposed with the model trained on the PBR dataset of one object (cat) to accelerate the training process. We analyze several aspects that influence the performance of the pose estimation. The comparisons are given on both LM and LMO datasets with and without refinement.

5.4.1. Feature Domain

Here we compare the quantitative results using 2D, 3D, and 2D+3D feature given into the backbone since sometimes the ideal RGB-D data is not available in the real world application. Table A.1 and 5.11 show the evaluation results in different cases. The model utilizing both 2D and 3D features achieves naturally the highest score followed by the model only using 3D feature. 2D-only method has a relatively wide gap with the other two settings. This is probability because the 2D feature extractor is not trained on the target dataset but on the general image dataset, and the 2D network suffers from the texture-less object with only limited information available from the data.

However, the additional 2D information solidly increases the accuracy of the estimation with only 3D feature. It validates that our feature fusion mechanism doesn't negatively compensate each other, but rather complements the missing information in the other domain.

Table 5.11.: Comparison of the different domains of the feature on LMO dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
2D	–	34.7	34.4	25.5	34.6	28.8
3D	–	54.4	49.4	33.1	45.6	78.4
2D+3D	–	56.8	51.1	37.4	**48.5**	**86.5**
2D	√	52.1	43.9	36.2	44.1	56.8
3D	√	57.4	53.5	42.2	51.0	86.5
2D+3D	√	61.2	56.6	46.6	**54.8**	**93.7**

5.4.2. Feature Fusion

As we introduced in section 4.3.6, we run the experiments with entity-wise addition and feature-wise concatenation to evaluate the effect of the different feature fusion. From table A.2 and 5.12 we find that the addition of the features outperforms the concatenation in both cases. Normally, the concatenation of the features can preserve more information than the addition with the tradeoff of the larger dimension of the feature. But in our setting, we find that the addition converges fast and performs better than concatenation in the module of feature fusion.

Table 5.12.: Comparison of the different feature fusion methods on LMO dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
Add	–	56.8	51.1	37.4	**48.5**	**86.5**
Concat.	–	47.0	38.2	27.3	37.5	40.5
Add	√	61.2	56.6	46.6	**54.8**	**93.7**
Concat.	√	54.6	48.0	38.5	47.1	72.1

5.4.3. Transformation Representation

Recall the discussion in section 2.1.2 and 4.3.7, we compare the different representations of rotation including quaternion, MRP and 6D representation [8]. From table A.3 and 5.13 we can find that the 6D representation and quaternion have equally best performance on LM dataset, and 6D achieves better results on the LMO without refinement. Figure A.2 shows the convergence of the rotation error with three different forms. We can also see the difference between each representation.

Table A.4 and 5.14 show the results of the model trained with and without residual translation. The advantage of learning the residual error is clear that the AR^*s are improved in both datasets. We can benefit from this trick with respect to the convergence speed shown in figure A.3. The residual translation is also a good way to prevent the model from being trapped in the local minimum.

Table 5.13.: Comparison of the different rotation representations on LMO dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
MRP	–	55.8	50.4	35.0	47.1	83.8
Quat.	–	56.1	51.0	36.4	47.8	82.0
6D	–	56.8	51.1	37.4	**48.5**	**86.5**
MRP	√	60.4	56.5	45.7	54.2	92.8
Quat.	√	61.0	56.8	46.6	**54.8**	**96.4**
6D	√	61.2	56.6	46.6	**54.8**	93.7

Table 5.14.: Comparison of the model trained with and without residual translation on LMO dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
w/o res.	–	57.1	50.5	28.2	45.3	69.4
w/ res.	–	56.8	51.1	37.4	**48.5**	**86.5**
w/o res.	√	60.9	56.4	44.3	53.9	91.9
w/ res.	√	61.2	56.6	46.6	**54.8**	**93.7**

5.4.4. Backbone Scaling

To test the scaling of our backbone, we compare the results of three different settings of our transformer encoder: Small-L4H2-D128, Base-L8H4-D256 and Large-L12H6-D768 (L, H and D represent the number of layers, heads and hidden dimensions respectively). The results are shown in table 5.15. With a larger and deeper network, the performance of the diffusion model is improved. The Large setting achieves 1 2% higher AR than the Base setting with a tradeoff of additional 20% training time.

Table 5.15.: Effect of the number of denoising steps on the estimation results.

Dataset		LM			LMO		
#Steps		Small	Base	Large	Small	Base	Large
AR	w/o ICP	66.7	68.4	69.2	44.1	45.5	47.6
	w/ ICP	77.0	78.4	79.2	52.2	52.9	55.0
ADD(-S)	w/o ICP	77.8	82.1	80.9	57.3	63.9	64.8
	w/ ICP	85.9	87.9	87.8	69.4	72.7	74.2

5.4.5. Denoising Steps

In the original DDPM setting of the image synthesis task, the diffusion steps are configured as 1000 to ensure the vividness of the generated image. In our pose estimation task, we compare the effects caused by the number of denoising steps on the final result in table 5.16.

The result shows that a large number of steps doesn't necessarily lead to better performance.

Table 5.16.: Effect of the number of denoising steps on the estimation results.

Dataset		LM			LMO		
#Steps		100	400	1000	100	400	1000
AR	w/o ICP	84.0	83.2	82.8	48.3	48.5	48.5
	w/ ICP	94.7	94.0	93.8	54.3	54.8	54.3
ADD(-S)	w/o ICP	98.5	97.5	98.5	89.2	86.5	85.6
	w/ ICP	99.0	98.5	99.0	92.8	93.7	92.8

6. Discussion

In this chapter, we will discuss our methodology, results from the experiments as well as advantages and drawbacks of our method.

Framework Our approach to 6 DoF pose estimation is founded upon the diffusion pipeline, introducing a novel perspective to address this task beyond conventional methods such as direct regression and correspondence matching. By leveraging generative models, specifically the advantageous diversity inherent in diffusion models, our framework enhances the versatility of pose estimation. However, this approach intensifies the intricacy of both training and real-world applications. A pivotal strategy involves segregating the training and inference phases at the network structure level. Inference, due to the iterative nature of the generation process within diffusion models, is inherently intricate and time-consuming.

Despite the incorporation of optimization techniques for sampling, as discussed in Section 3.2.2, the inference time remains noteworthy compared to more expeditious methods. Notably, the encoder networks, responsible for extracting features from the input data and constituting the diffusion backbone, are trained independently of the diffusion networks. Consequently, our model is not an end-to-end system, necessitating the pretraining of feature extractors through subtasks or the utilization of pre-existing generalized models. This delineation of tasks ensures the effective functioning of the overall framework while acknowledging the intricacies inherent in the training and deployment phases.

Featue Fusion Within our methodology, we leverage both 2D and 3D features from the input data to maximize information extraction. However, the fusion of features across different domains poses a non-trivial challenge in the realm of deep learning. Although we initially explored intuitive methods such as addition and concatenation, these operations are not always the most efficient or effective. The ablation study (refer to Section 5.4.2) underscores the substantial impact that the choice of feature fusion methodology has on the final results.

Moreover, our approach involves extracting global features from both the 2D and 3D domains to characterize the object within a specific pose. The fusion of global features is relatively straightforward, involving the matching of feature dimensions through the application of projection layers. However, when extracting local features through patchified sampling or dense feature extraction from key points, aligning these local features across different domains becomes a non-trivial task, adding an additional layer of complexity to our methodology.

2D Feature Extractor In our methodology, we employ a self-supervised 2D network pretrained on a comprehensive image dataset to extract 2D features, rather than training it on the specific target dataset. This choice is motivated by the objective of encoding the image with latent expression, and a general pre-trained model possesses the capability to handle previously unseen RGB data in downstream tasks. As observed in Section 5.4.1, utilizing only 2D features results in suboptimal performance, particularly in scenarios involving occlusion.

While enhancing performance is conceivable through supervised training of the 2D network specifically for the pose estimation task, this approach deviates from our primary objective. We aim to demonstrate the efficacy of the diffusion framework in processing indistinct input features.

The decision to use a pre-trained network, which has learned representations from a diverse array of images, aligns with the overarching goal of showcasing the adaptability of diffusion models across various applications and datasets. This approach obviates the need to retrain the network from scratch for each new application, providing a more flexible and resource-efficient solution.

3D Feature Extractor In contrast to 2D feature extraction networks, there is a notable absence of widely applicable 3D feature extraction networks that can achieve high accuracy on unseen data for downstream tasks. This scarcity arises from the inherent sparsity of 3D data, coupled with the point cloud's permutation invariance and its exclusive reliance on geometric information. These factors introduce greater uncertainty into the feature extraction process. Consequently, we opt to train our 3D network on the target dataset, specifically tailoring it to extract features from a point cloud completion task. As demonstrated in Section 5.4.1, our chosen approach validates the effectiveness of training a 3D network for satisfactory results.

It is important to note that the utilized network exhibits commendable performance when dealing with individual objects. However, there remains room for improvement in terms of generalization, particularly in scenarios involving multiple objects. When the occluded portion surpasses a certain percentage of the entire object, the network encounters challenges in accurately completing the shape of the object without prior knowledge. This limitation underscores the need for continued advancements in 3D feature extraction networks for broader applicability across diverse and complex scenarios.

Diffusion Model Our diffusion model adopts the architecture of the DDPM and incorporates techniques from DDIM to expedite the inference phase. Visualizations in Figure 5.12 demonstrate that, regardless of the initial pose hypothesis, our approach converges to the ground truth pose after a certain number of denoising steps. Notably, the number of denoising steps has a discernible impact on the quality of generated samples in image synthesis tasks. However, as discussed in Section 5.4.5, in our pose estimation task, varying the number of denoising steps does not significantly alter the results. This can be attributed to the fact that the denoising objective, in terms of transformation, represents a low-dimensional representation of a 3D object compared to a point cloud. The 2D image itself is a high-dimensional matrix with more entities (pixels) and channels, necessitating more denoising steps to ensure meaningful representation and quality at both pixel and image levels. In our context, the transformation vector converges more readily to the ground truth pose, requiring fewer denoising steps.

Unlike the original DDPM, our adaptation employs a Transformer encoder as the backbone to accommodate the input data format, diverging from the traditional CNN-based network. This choice introduces the advantage of leveraging the self-attention mechanism to capture weighted relationships between input tokens. This attribute proves beneficial for potential extensions of our work into pose estimation for sequence inputs, such as video or object tracking. The self-attention mechanism can enhance the model's understanding of tempo-

ral relationships between frames, contributing to more robust estimations. The transformer encoder, designed for sequence inputs, aligns well with the nature of such dynamic scenarios.

In our experiments, we adopt the mean of multi-hypotheses as the final output of the generated pose. This approach serves to further minimize errors with the ground truth and expedite model convergence. In cases involving symmetrical objects or ambiguity arising from occlusion, the distribution of pose hypotheses becomes multi-modal rather than a single peak. In such scenarios, it proves advantageous to sample multiple hypotheses and analyze the distribution of poses, particularly in applications like robot grasping, where determining the optimal pose for object manipulation is crucial.

7. Conclusion

In this last chapter of our thesis, we summarize the work and highlight the contributions of our method in the field of 6 DoF pose estimation. Additionally, it gives an outlook on the future work.

7.1. Summary

This thesis introduces a pioneering framework and comprehensive evaluation for 6 DoF pose estimation, a crucial task in computer vision with widespread applications in robotics and augmented reality. Traditional methods, including template-based, keypoint-based, and direct regression approaches, exhibit distinct advantages and constraints in various scenarios. In response, our work proposes a diffusion-based framework for 6 DoF pose estimation.

Diffusion models, a category of generative models, excel in generating high-quality samples from randomly initialized noise. The thesis begins with a review of classical diffusion models and related improvements. Our designed diffusion model, tailored for pose estimation, employs a Transformer encoder as the backbone. This encoder estimates noise added at each training step, guided by the noisy pose and conditional information from the input data. During the inference phase, the pose undergoes gradual denoising from initial random noise to final estimation through iterative passes of a denoiser, implemented as the Transformer encoder. Additionally, optional multi-hypotheses and ICP refinement steps further enhance estimation accuracy.

To extract the feature from the input RGB-D data. A self-supervised ViT called DINO [131] is utilized for 2D feature downstream and a point cloud completion task using FoldingNet [132] is designed for 3D feature. After that, we fuse the feature from both domains to become the conditional embedding form the diffusion model.

To validate the performance of our method, we evaluate our method on LM [142] dataset and the occlusion version LMO [16] dataset with the comparison of other state-of-the-art methods. Results demonstrate comparable performance, even in the presence of occluded objects. Real-world dataset experiments underscore the applicability of the diffusion-based framework in 6 DoF pose estimation. A comprehensive ablation study is conducted, analyzing the impact of various components in our method.

As a pioneering work in diffusion-based 6 DoF pose estimation on real-world datasets, our approach not only introduces a novel methodology for this task but also highlights the potential of diffusion models beyond image synthesis applications.

7.2. Outlook

In future research, several aspects of our work can be further improved and expanded. In our thesis, we use the architecture of DDPM as our diffusion model. The alternative model like SGM [34, 96] can be also applied to our task to avoid the consideration of normalizing constant in likelihood-based models. The sampling techniques like CAS [101], ALD [102] and DDSS [108] can be an alternative to the DDIM to accelerate the inference phase.

Furthermore, attention could be directed towards refining the feature extraction process. While we currently utilize a general pre-trained 2D network, training the network with supervision on the target dataset could lead to further performance improvements. For 3D features, exploring techniques like contrastive learning may present a novel direction. The generalization of the 3D downstream network remains a bottleneck for the flexible application of our method. Discovering ways to extract 3D features from unseen data using a general model could significantly broaden the utility of our approach. Additionally, investigating advanced feature fusion methods, such as cross-attention, could further enhance the understanding of relationships between features from different domains.

Another avenue for extension is adapting our method for object-tracking tasks. Leveraging the transformer encoder as the diffusion backbone could enable understanding not only of the relationship between rotation and translation within a single frame but also the temporal relationships between frames. The use of multi-hypotheses from diffusion models to generate pose distributions, rather than a single solution, would allow for a better understanding of ambiguity in symmetrical objects and facilitate the identification of the optimal pose for tracking objects.

A. Additionally: Evaluation Results

A.1. Quantitative Evaluation

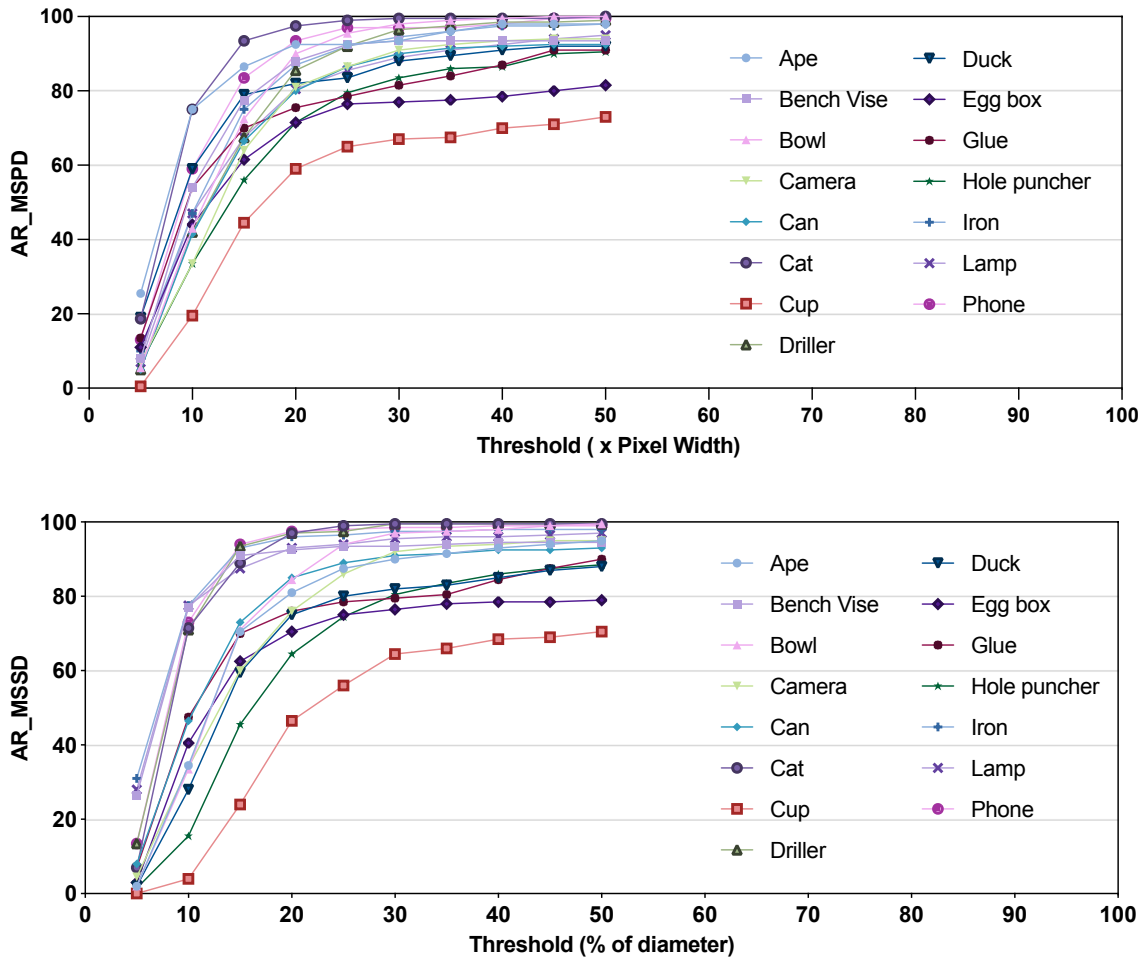


Figure A.1.: Categorical MSPD and MSSD recall rates with different thresholds on LM dataset (without refinement).

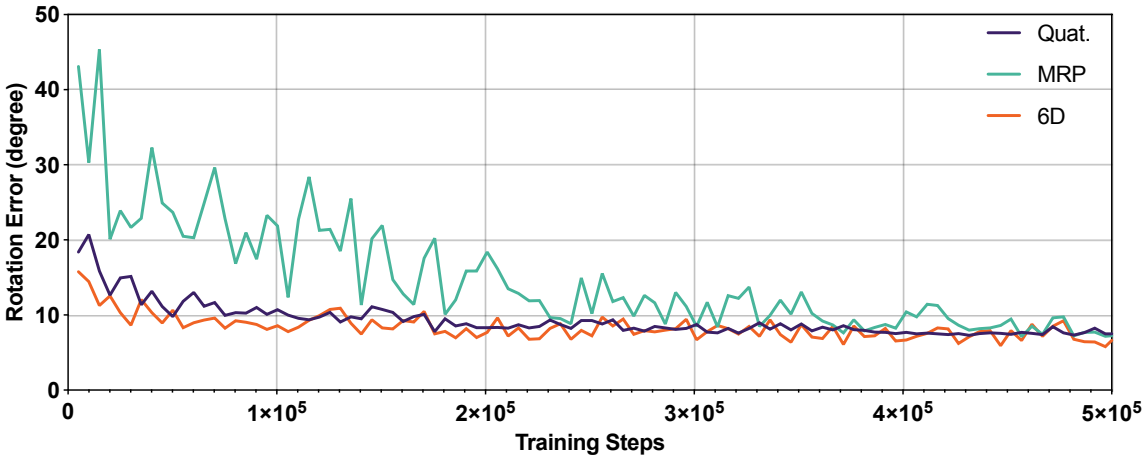


Figure A.2.: Rotation error under different representations of rotation during the training phase (without refinement).

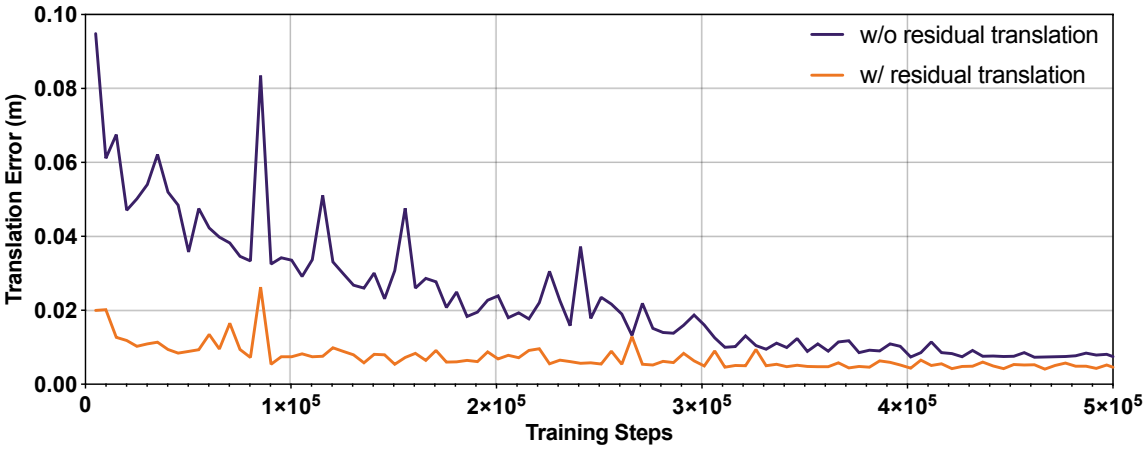


Figure A.3.: Translation error with and without using residual translation during the training phase (without refinement).

Table A.1.: Comparison of the different domains of the feature on LM dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
2D	–	77.3	73.2	47.6	66.0	76.0
3D	–	88.9	87.0	68.3	81.4	93.5
2D+3D	–	91.6	89.1	68.8	**83.2**	**97.5**
2D	√	93.9	91.8	78.1	87.9	91.5
3D	√	94.8	93.9	83.5	90.7	95.5
2D+3D	√	98.5	98.7	85.7	**94.0**	**98.5**

Table A.2.: Comparison of the different feature fusion methods on LM dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
Add	–	91.6	89.1	68.8	**83.2**	**97.5**
Concat.	–	80.9	77.5	53.2	70.5	83.5
Add	√	98.5	98.7	85.7	**94.0**	**98.5**
Concat.	√	94.8	93.7	80.4	89.7	94.0

Table A.3.: Comparison of the different rotation representations on LM dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
MRP	–	89.4	87.0	65.2	80.5	96.0
Quat.	–	91.6	89.4	68.5	**83.2**	**99.0**
6D	–	91.6	89.1	68.8	**83.2**	97.5
MRP	√	97.4	96.3	84.2	92.6	97.5
Quat.	√	99.0	98.3	86.1	**94.5**	**99.0**
6D	√	98.5	98.7	85.7	94.0	98.5

Table A.4.: Comparison of the model trained with and without residual translation on LM dataset.

Method	Refine	AR_{MSPD}	AR_{MSSD}	AR_{VSD}	AR	ADD(-S)
w/o res.	–	92.0	88.3	59.3	79.9	93.5
w/ res.	–	91.6	89.1	68.8	**83.2**	**97.5**
w/o res.	√	98.3	97.3	84.5	93.4	**99.0**
w/ res.	√	98.5	98.7	85.7	**94.0**	98.5

A.2. Qualitative Evaluation

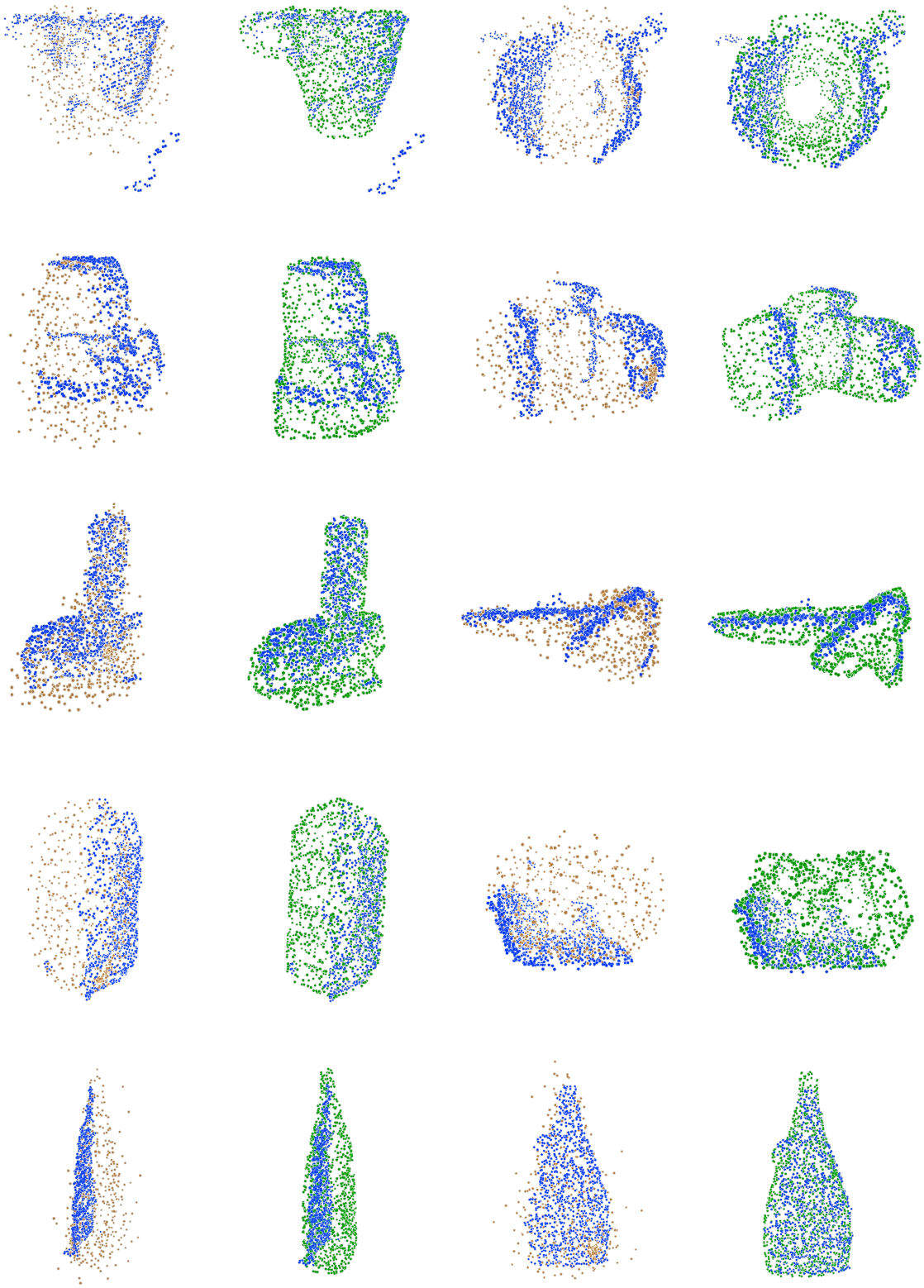


Figure A.4.: Point cloud completion results of the FoldingNet. From top to bottom: cup, camera, phone, egg box, glue. From left to right: estimation(viewpoint 1), ground truth(viewpoint 1), estimation(viewpoint 2), ground truth(viewpoint 2). Blue points are the input partial point cloud.

Figure A.5.: Visualization of some objects highlighted at estimated pose. Row 1-2: objects without occlusion; Row 3-4: objects with occlussion; Row 5: objects with occlusion shown in one scene.

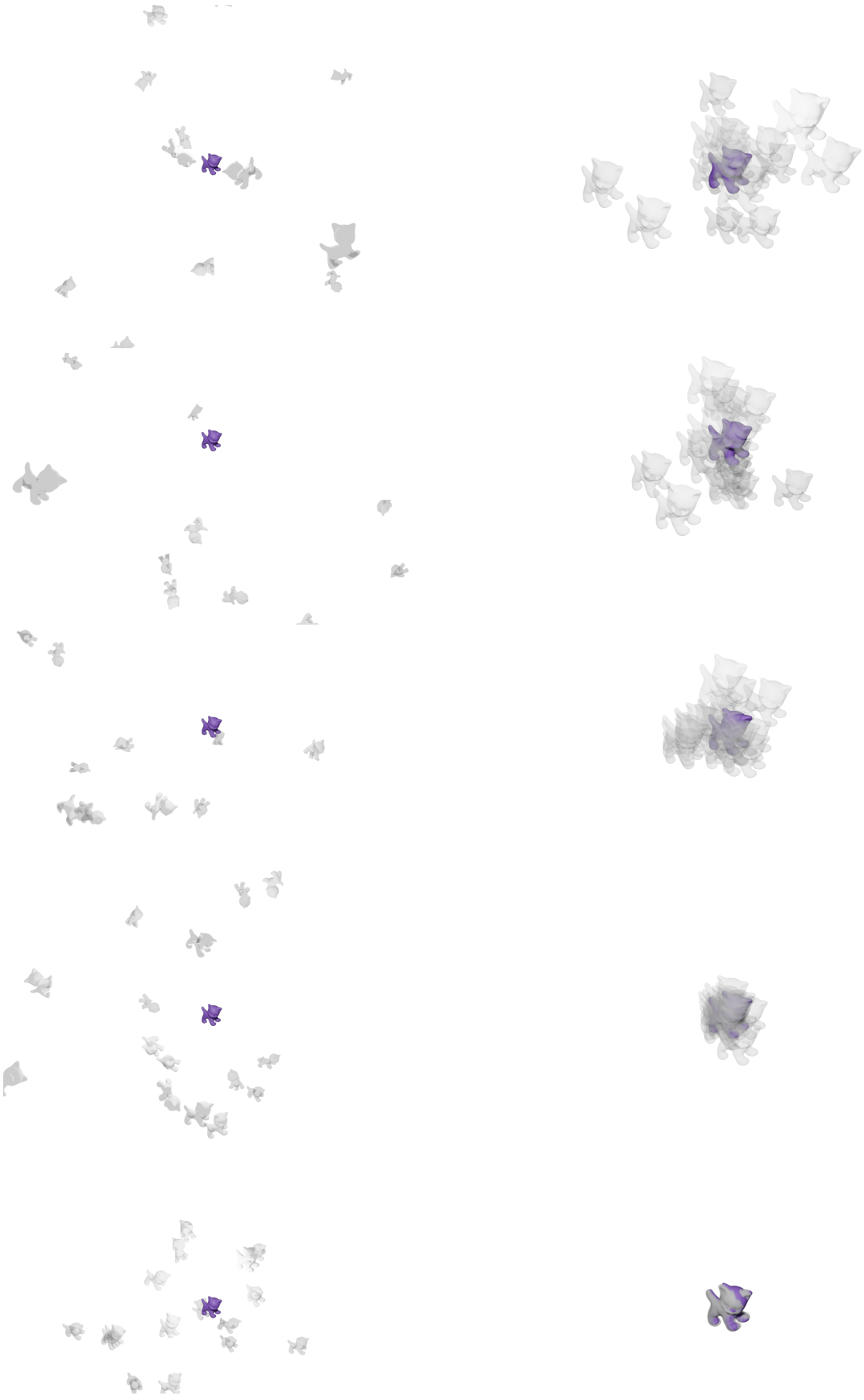


Figure A.6.: Visualization of the denoising process of the pose estimation of cat object (Sequence: top to bottom, left to right).

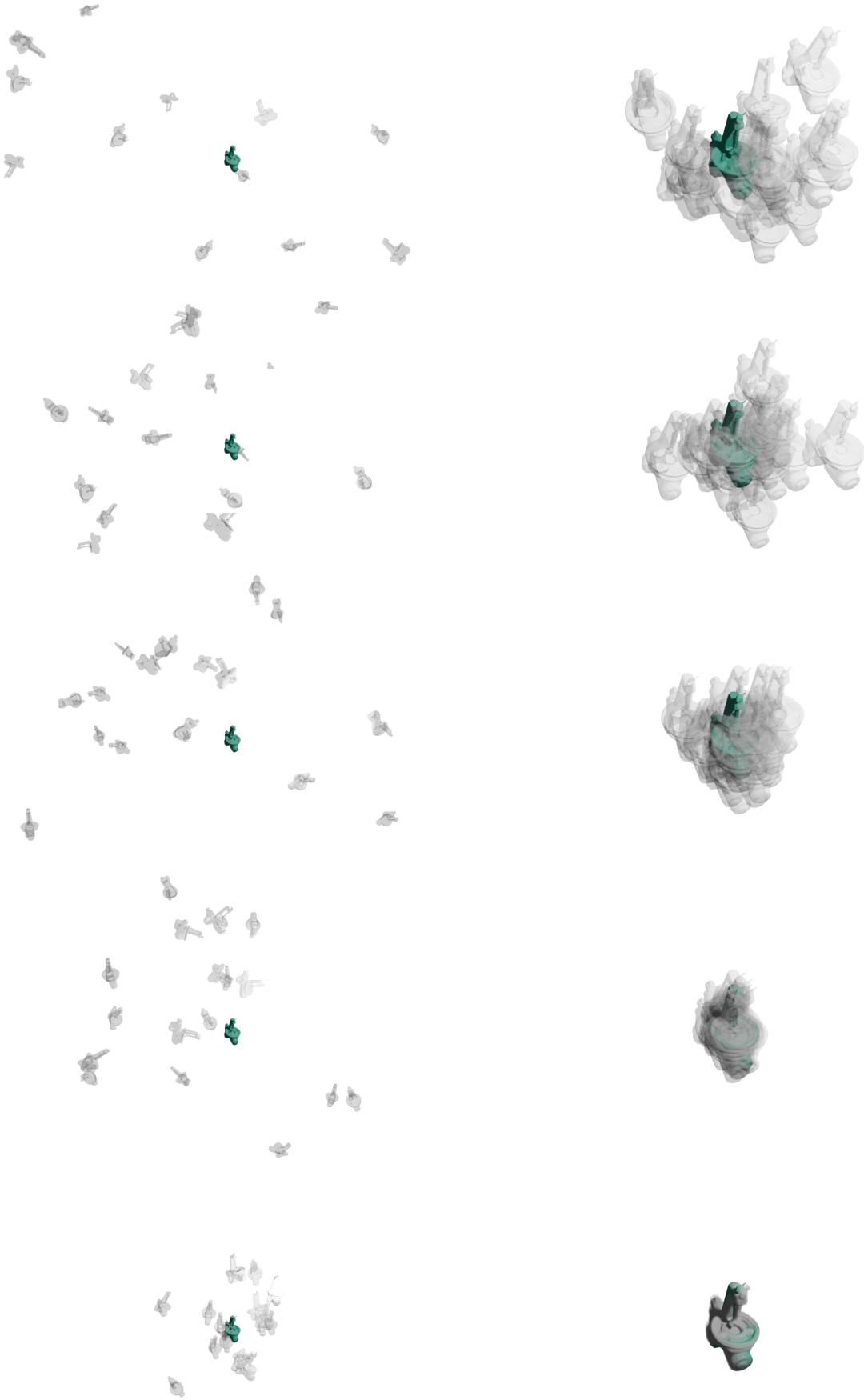


Figure A.7.: Visualization of the denoising process of the pose estimation of bench object (Sequence: top to bottom, left to right).

List of Figures

List of Tables

Bibliography

[1] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *ICML*, 2015.

[2] J. Ho, A. Jain and P. Abbeel, "Denoising diffusion probabilistic models," in *NIPS*, 2020.

[3] F. Manhardt, "Towards monocular 6d object pose estimation," Ph.D. dissertation, Technische Universität München, 2021.

[4] Y. Xiang, T. Schmidt, V. Narayanan and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *CoRR*, 2017.

[5] H. A. Hashim, "Special orthogonal group so(3), euler angles, angle-axis, rodriguez vector and unit-quaternion: Overview, mapping and challenges," in *ArXiv preprint ArXiv:1909.06669*, 2019.

[6] V. Mansur, S. Reddy, S. R and R. Sujatha, "Deploying complementary filter to avert gimbal lock in drones using quaternion angles," in *GUCON*, 2020, pp. 751–756.

[7] G. Terzakis, M. Lourakis and D. Ait-Boudaoud, "Modified rodrigues parameters: An efficient representation of orientation in 3d vision and graphics," in *Journal of Mathematical Imaging and Vision*, 2018.

[8] Y. Zhou, C. Barnes, L. Jingwan, Y. Jimei and L. Hao, "On the continuity of rotation representations in neural networks," in *CVPR*, 2019.

[9] Y. Zhu, M. Li, W. Yao and C. Chen, "A review of 6d object pose estimation," in *ITAIC*, 2022.

[10] H. Cao, L. Dirnberger, D. Bernardini, C. Piazza and M. Caccamo, "6impose: Bridging the reality gap in 6d pose estimation for robotic grasping," in *Frontiers in Robotics and AI*, 2023.

[11] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng and K. Xu, "Geometric transformer for fast and robust point cloud registration," in *CVPR*, 2022.

[12] K. Fu, S. Liu, X. Luo and M. Wang, "Robust point cloud registration framework based on deep graph matching," in *CVPR*, 2021.

[13] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," in *PAMI*, 1992.

[14] R. A. Rill and K. Faragó, "Collision avoidance using deep learning-based monocular vision," in *SN Computer Science*, 2021.

[15] G. Marullo, L. Tanzi, P. Piazzolla and E. Vezzetti, "6d object position estimation from 2d images: a literature review," in *Multimedia Tools and Applications*, 2022.

[16] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton and C. Rother, "Lerning 6D Object Pose Estimation using 3D Object Coordinates," in *ECCV*, 2014.

[17] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *WACV*, 2017.

[18] A. Kendall, M. Grimes and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *ICCV*, 2016.

[19] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis and K. Daniilidis, "6-dof object pose from semantic keypoints," in *ICRA*, 2017.

[20] T. Hodan, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. N. Sinha and B. Guenter, "Photorealistic image synthesis for object instance detection," in *ICIP*, 2019.

[21] A. Lamb, "A brief introduction to generative models," in *arXiv:2103.00265*, 2021.

[22] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial networks," in *NIPS*, 2014.

[23] A. Borji, "Pros and cons of gan evaluation measures," in *Computer Vision and Image Understanding*, 2018.

[24] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein gan," in *PMLR*, 2017.

[25] T. Miyato, T. Kataoka, M. Koyama and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *ICLR*, 2018.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2022.

[27] D. P. Kingma, T. Salimans and M. Welling, "Variational dropout and the local reparameterization trick," in *NIPS*, 2015.

[28] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *ICML*, 2016.

[29] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," in *arXiv:2112.10741*, 2022.

[30] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu and M. Chen, "Hierarchical text-conditional image generation with clip latents," in *arXiv:2204.06125*, 2022.

[31] L. Weng, "What are diffusion models?" *lilianweng.github.io*, 2021. [Online]. Available: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

[32] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *PMLR*, 2021.

[33] Z. Xiao, K. Kreis and A. Vahdat, "Tackling the generative learning trilemma with denoising diffusion GANs," in *ICLR*, 2022.

[34] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *NIPS*, 2020.

[35] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *arXiv:2207.12598*, 2022.

[36] 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," in *ACM Computing Surveys*, 2023.

[37] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet and M. Norouzi, "Image super-resolution via iterative refinement," in *PAMI*, 2021.

[38] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi and T. Salimans, "Cascaded diffusion models for high fidelity image generation," in *The Journal of Machine Learning Research*, 2021.

[39] S. Gao, X. Liu, B. Zeng, S. Xu, Y. Li, X. Luo, J. Liu, X. Zhen and B. Zhang, "Implicit diffusion models for continuous super-resolution," in *CVPR*, 2023.

[40] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte and L. V. Gool, "Repaint: Inpainting using denoising diffusion probabilistic models," in *CVPR*, 2022.

[41] C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet and M. Norouzi, "Palette: Image-to-image diffusion models," in *SIGGRAPH*, 2022.

[42] G. Kwon and J. C. Ye, "Diffusion-based image translation using disentangled style and content representation," in *arXiv:2209.15264*, 2023.

[43] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *CVPR*, 2021.

[44] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler and K. Kreis, "Lion: Latent point diffusion models for 3d shape generation," in *arXiv:2210.06978*, 2022.

[45] Z. Lyu, Z. Kong, X. Xu, L. Pan and D. Lin, "A conditional point diffusion-refinement paradigm for 3d point cloud completion," in *arXiv:2112.03530*, 2022.

[46] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *arXiv:1810.04805*, 2019.

[47] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49313245

[48] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave and G. Lample, "Llama: Open and efficient foundation language models," in *arXiv:2302.13971*, 2023.

[49] H. Zou, Z. M. Kim and D. Kang, "A survey of diffusion models in natural language processing," in *arXiv:2305.14671*, 2023.

[50] J. Austin, D. D. Johnson, J. Ho, D. Tarlow and R. van den Berg, "Structured denoising diffusion models in discrete state-spaces," in *NIPS*, 2023.

[51] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré and M. Welling, "Argmax flows and multinomial diffusion: Learning categorical distributions," in *NIPS*, 2021.

[52] X. L. Li, J. Thickstun, I. Gulrajani, P. Liang and T. B. Hashimoto, "Diffusion-lm improves controllable text generation," in *NIPS*, 2022.

[53] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger and I. Sutskever, "Learning transferable visual models from natural language supervision," in *PMLR*, 2021.

[54] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.

[55] L. Zhang, A. Rao and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *ICCV*, 2023.

[56] B. Poole, A. Jain, J. T. Barron and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," in *arXiv:2209.14988*, 2022.

[57] D. Lowe, "Distinctive image features from scale-invariant keypoints," in *IJCV*, 2004.

[58] H. Bay, T. Tuytelaars and L. Van Gool, "Surf: Speeded up robust features," in *ECCV*, 2006.

[59] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "Orb: An efficient alternative to sift or surf," in *ICCV*, 2011.

[60] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," in *Commun. ACM*. Association for Computing Machinery, 1981.

[61] A. Segal, D. Hähnel and S. Thrun, "Generalized-icp," 2009.

[62] J. Serafin and G. Grisetti, "Nicp: Dense normal based point cloud registration," in *IROS*, 2015.

[63] C. Gu and X. Ren, "Discriminative mixture-of-templates for viewpoint classification," in *ECCV*, 2010.

[64] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, V. Lepetit and b. Lepetit, "Gradient response maps for real-time detection of texture-less objects," in *PAMI*, 2012.

[65] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *CVPR*, 2015.

[66] V. N. Nguyen, Y. Hu, Y. Xiao, M. Salzmann and V. Lepetit, "Templates for 3d object pose estimation revisited: Generalization to new objects and robustness to occlusions," in *CVPR*, 2022.

[67] I. Shugurov, F. Li, B. Busam and S. Ilic, "Osop: A multi-stage one shot object pose estimation framework," in *CVPR*, 2022.

[68] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," in *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 1976.

[69] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *ICCV*, 2018.

[70] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Proceedings of The 2nd Conference on Robot Learning*, 2018.

[71] B. Tekin, S. N. Sinha and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *CVPR*, 2018.

[72] S. Zakharov, I. Shugurov and S. Ilic, "Dpod: 6d pose object detector and refiner," in *ICCV*, 2019.

[73] M. Oberweger, M. Rad and V. Lepetit, "Making deep heatmaps robust to partial occlusions for 3d object pose estimation," in *ECCV*, 2018.

[74] Y. Su, M. Saleh, T. Fetzer, J. Rambach, N. Navab, B. Busam, D. Stricker and F. Tombari, "Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation," in *CVPR*, 2022.

[75] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[76] C. R. Qi, L. Yi, H. Su and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NIPS*, 2017.

[77] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan and C.-L. Tai, "D3feat: Joint learning of dense detection and description of 3d local features," in *CVPR*, 2020.

[78] S. Ao, Q. Hu, B. Yang, A. Markham and Y. Guo, "Spinnet: Learning a general surface descriptor for 3d point cloud registration," in *CVPR*, 2021.

[79] H. Wang, Y. Liu, Z. Dong and W. Wang, "You only hypothesize once: Point cloud registration with rotation-equivariant descriptors," in *ACMMM*, 2022.

[80] H. Yu, Z. Qin, J. Hou, M. Saleh, D. Li, B. Busam and S. Ilic, "Rotation-invariant transformer for point cloud matching," in *CVPR*, 2023.

[81] Y. Xiang, T. Schmidt, V. Narayanan and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," in *RSS*, 2018.

[82] J. Wu, B. Zhou, R. Russell, V. Kee, S. Wagner, M. Hebert, A. Torralba and D. M. Johnson, "Real-time object pose estimation with pose interpreter networks," in *IROS*, 2018.

[83] T.-T. Do, M. Cai, T. Pham and I. Reid, "Deep-6dpose: Recovering 6d object pose from a single rgb image," in *arXiv:1802.10367*, 2018.

[84] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask r-cnn," in *ICCV*, 2018.

[85] G. Wang, F. Manhardt, F. Tombari and X. Ji, "Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation," in *CVPR*, 2021.

[86] Y. Aoki, H. Goforth, R. A. Srivatsan and S. Lucey, "Pointnetlk: Robust & efficient point cloud registration using pointnet," in *CVPR*, 2019.

[87] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981.

[88] X. Huang, G. Mei and J. Zhang, "Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences," in *CVPR*, 2020.

[89] H. Xu, S. Liu, G. Wang, G. Liu and B. Zeng, "Omnet: Learning overlapping mask for partial-to-partial point cloud registration," in *ICCV*, 2021.

[90] W. Kehl, F. Milletari, F. Tombari, S. Ilic and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *ECCV*, 2016.

[91] K. Park, T. Patten and M. Vincze, "Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation," in *ICCV*, 2019.

[92] S. Peng, Y. Liu, Q. Huang, X. Zhou and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *CVPR*, 2019.

[93] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *CVPR*, 2019.

[94] Y. He, H. Huang, H. Fan, Q. Chen and J. Sun, "Ffb6d: A full flow bidirectional fusion network for 6d pose estimation," in *CVPR*, 2021.

[95] C. Song, J. Song and Q. Huang, "Hybridpose: 6d object pose estimation under hybrid representations," in *CVPR*, 2020.

[96] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," in *NIPS*, 2020.

[97] Y. Song, C. Durkan, I. Murray and S. Ermon, "Maximum likelihood training of score-based diffusion models," in *NIPS*, 2021.

[98] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon and B. Poole, "Score-based generative modeling through stochastic differential equations," in *ICLR*, 2021.

[99] A. Hyvärinen, "Estimation of non-normalized statistical models by score matching," in *JMLR*, 2005.

[100] P. Vincent, "A connection between score matching and denoising autoencoders," in *Neural Computation*, 2011.

[101] A. Jolicoeur-Martineau, R. Piché-Taillefer, R. T. des Combes and I. Mitliagkas, "Adversarial score matching and improved sampling for image generation," in *ICLR*, 2020.

[102] T. Dockhorn, A. Vahdat and K. Kreis, "Score-based generative modeling with critically-damped langevin diffusion," in *ICLR*, 2022.

[103] A. Jolicoeur-Martineau, K. Li, R. Piché-Taillefer, T. Kachman and I. Mitliagkas, "Gotta go fast when generating data with score-based models," in *ICLR*, 2021.

[104] J. Song, C. Meng and S. Ermon, "Denoising diffusion implicit models," in *ICLR*, 2022.

[105] T. Karras, M. Aittala, T. Aila and S. Laine, "Elucidating the design space of diffusion-based generative models," in *NIPS*, 2022.

[106] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li and J. Zhu, "Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps," in *NIPS*, 2022.

[107] D. Watson, J. Ho, M. Norouzi and W. Chan, "Learning to efficiently sample from diffusion probabilistic models," in *arXiv:2106.03802*, 2021.

[108] D. Watson, W. Chan, J. Ho and M. Norouzi, "Learning fast samplers for diffusion models by differentiating through sample quality," in *arXiv:2202.05830*, 2022.

[109] H. Zheng, P. He, W. Chen and M. Zhou, "Truncated diffusion probabilistic models and diffusion-based adversarial auto-encoders," in *ICLR*, 2023.

[110] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *ICLR*, 2022.

[111] C. Meng, R. Rombach, R. Gao, D. P. Kingma, S. Ermon, J. Ho and T. Salimans, "On distillation of guided diffusion models," in *CVPR*, 2023.

[112] D. P. Kingma, T. Salimans, B. Poole and J. Ho, "Variational diffusion models," in *NIPS*, 2023.

[113] S. Gu, D. Chen, J. Bao, F. Wen, B. Zhang, D. Chen, L. Yuan and B. Guo, "Vector quantized diffusion model for text-to-image synthesis," in *CVPR*, 2022.

[114] A. van den Oord, O. Vinyals and K. Kavukcuoglu, "Neural discrete representation learning," in *NIPS*, 2018.

[115] X. Liu, L. Wu, M. Ye and qiang Liu, "Learning diffusion bridges on constrained domains," in *ICLR*, 2023.

[116] C.-W. Huang, M. Aghajohari, A. J. Bose, P. Panangaden and A. Courville, "Riemannian diffusion models," in *NIPS*, 2022.

[117] V. D. Bortoli, E. Mathieu, M. Hutchinson, J. Thornton, Y. W. Teh and A. Doucet, "Riemannian score-based generative modelling," in *NIPS*, 2022.

[118] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover and S. Ermon, "Permutation invariant graph generation via score-based generative modeling," in *PMLR*, 2020.

[119] H. Huang, L. Sun, B. Du, Y. Fu and W. Lv, "Graphgdp: Generative diffusion processes for permutation invariant graph generation," in *ICDM*, 2022.

[120] A. Vahdat, K. Kreis and J. Kautz, "Score-based generative modeling in latent space," in *NIPS*, 2021.

[121] A. Leach, S. M. Schmon, M. T. Degiacomi and C. G. Willcocks, "Denoising diffusion probabilistic models on so(3) for rotational alignment," in *ICLR*, 2022.

[122] Y. Jagvaral, F. Lanusse and R. Mandelbaum, "Diffusion generative models on so(3)," 2023. [Online]. Available: https://openreview.net/forum?id=jHA-yCyBGb

[123] 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 *ICRA*, 2023.

[124] J. Yim, B. L. Trippe, V. D. Bortoli, E. Mathieu, A. Doucet, R. Barzilay and T. Jaakkola, "Se(3) diffusion model with application to protein backbone generation," in *ICML*, 2023.

[125] T.-C. Hsiao, H.-W. Chen, H.-K. Yang and C.-Y. Lee, "Confronting ambiguity in 6d object pose estimation via score-based diffusion on se(3)," in *arXiv:2305.15873*, 2023.

[126] K. A. Murphy, C. Esteves, V. Jampani, S. Ramalingam and A. Makadia, "Implicit-pdf: Non-parametric representation of probability distributions on the rotation manifold," in *ICML*, 2021.

[127] J. Wang, C. Rupprecht and D. Novotny, "Posediffusion: Solving pose estimation via diffusion-aided bundle adjustment," in *ICCV*, 2023.

[128] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut and D. Novotny, "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction," in *ICCV*, 2021.

[129] T. Zhou, R. Tucker, J. Flynn, G. Fyffe and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images," in *SIGGRAPH*, 2018.

[130] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[131] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski and A. Joulin, "Emerging properties in self-supervised vision transformers," in *ICCV*, 2021.

[132] Y. Yang, C. Feng, Y. Shen and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *CVPR*, 2018.

[133] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," in *PAMI*, 1992.

[134] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is all you need," in *NIPS*, 2023.

[135] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

[136] E. Perez, F. Strub, H. de Vries, V. Dumoulin and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI*, 2017.

[137] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *ICCV*, 2022.

[138] A. Kazemnejad, "Transformer architecture: The positional encoding," *kazemnejad.com*, 2019. [Online]. Available: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

[139] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[140] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *ICCVW*, 2019.

[141] Y. Shen, C. Feng, Y. Yang and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *CVPR*, 2018.

[142] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *ACCV*, 2012.

[143] T. Hodan, F. Michel, E. Brachmann, W. Kehl, A. G. Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas and C. Rother, "Bop: Benchmark for 6d object pose estimation," in *ECCV*, 2018.

[144] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi and H. Katam, "Blenderproc," in *arXiv:1911.01911*, 2019.

[145] T. Hodan, M. Sundermeyer, B. Drost, Y. Labbe, E. Brachmann, F. Michel, C. Rother and J. Matas, "Bop challenge 2020 on 6d object localization," in *ECCVW*, 2020.

[146] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold and C. Rother, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *CVPR*, 2016.

Declaration

Herewith, I declare that I have developed and written the enclosed thesis entirely by myself and that I have not used sources or means except those declared.

This thesis has not been submitted to any other authority to achieve an academic grading and has not been published elsewhere.

Stuttgart, 30.11.2023 Haoqing Wu