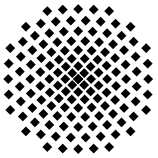

University of Stuttgart
Institute for Signal Processing and System Theory
Professor Dr.-Ing. B. Yang

Masterarbeit Dxxxx TBD

Thesis title TBD

Arbeitstitel, to be defined (TBD)

Author: Student's name TBD

Date of work begin: Date of work begin TBD
Date of submission: Date of submission TBD

Supervisor: Supervisor's name TBD

Keywords: Keyword1, Keyword2 TBD

Abstract TBD

Contents

1. Introduction

1.1. Motivation

1.2. Main Objective

1.3. Structure of the Thesis

As shown in [1], we present an equation

$$H(\omega) = \int h(t)\, \mathrm{e}^{\mathrm{j}\,\omega t}\, \delta t \in \mathbb{N} \tag{1.1}$$

Then we include a graphic in figure 1.1 and information about captions in table 1.1.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Figure 1.1.: A beautiful mind

Table 1.1.: Where to put the caption

	above	below
for figures	no	yes
for tables	yes	no

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2. Background

2.1. 6 DoF Pose Estimation

2.1.1. Definition

Six degree-of-freedom(DoF) pose refers to the six degrees of freedom of movement of a rigid body in three-dimensional space. Especially, it represents the freedom of a rigid body to move in three perpendicular directions, called translations, and to rotate about three perpendicular axes, called rotations. This concept is widely applied in the industial and automotive field to measure and analyze the spacial properties of objects.

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[2]. The object M is typically a known 3D 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].

————————image here————————-

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 translation vector to the object vertices V

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

Similarly, the 3D rotation can be represented by 3 rotation matrics around the X, Y and Z axis. And rotating the object vertices V by the rotation matrix $\mathbf{R}_i$ with $i \in \{X, Y, Z\}$ can be achieved by multiplying them. 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 a $Z-Y-X$ order, which means the rotation around X axis is performed first, then Y axis and finally Z axis. All possible rotations in 3D Euclidean space establish a natual manifold known as special orthognal 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 homogeneous representation of a transformation. All possible transformation matrices of this form generate the special Euclidean group $\mathbb{SE}(3)$

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

An alternative representation of 6 DoF pose is a 7-dimensional vector that consists of translation and rotation quaternion which has a compacter form

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

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 quaterion form is in many case prefered because it can ensure the uniqueness by restricting the quaterion on the upper hemisphere of $q_w = 0$ plane and can also guarantee a gimbal-lock free rotation in $\mathbb{SO}(3)$[6]. However, rotation matrix is widely used in many dataset to represent the ground truth transformation.

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 reality(AR), robotics, 3D scene understanding and autonomous driving.

Augmented Reality

AR applications use 6 DoF pose estimation to accurately place the virtual objects in the real world. With precise estimation and quick inference of the pose guarantee a immersive and interactive experience which is the direction of the development of AR applications[7]. Furthermore, 6 DoF pose estimation can also be utilized to track the real world objects, enabling more natural interactions.

Robotics

6 DoF pose estimation helps robots to 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[8]. In manufacturing, robots use the estimated pose to identify, sort and assemble the objects in field like automatic logistic sorting and manufacturing line.

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 method like correspondance matching[9] or direct transformation estimation[10] follows the ideas of ICP[11].

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[12], because the vehicle move normally in high speed and the heading direction cause 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 became a popular research topic of computer vision in recent years. However, solving this problem is not trivial and even challenging in many cases.

First constrain would be the auto-occlusion or symmetries of the object since the object cannot be clearly and unequivocally observed from all angles[13]. The auto-occlusion means that the object itself is partially occluded by other parts of the object such as LINEMOD-O dataset[14]. This is common in many real world objects such as table or chair. The symmetries of the object means that the object has same appearance from different angles, which will cause ambiguity in the estimation such as T-LESS dataset[15]. Imagining an

image of 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[16] or keypoint based method[17] to extract enough local features if the object is complete 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 in order to have a 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 the real world data or 3D rendered images which can reduce the domain gap in some degree[18], but the noise and unvalid training samples still confuse the model.

If facing the multi-object scenario, which is common in the application 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.

——————image here——————-

2.2. Diffusion Model

2.2.1. Generative Models

One of the most fascinating and distinctive feature of human brain is the ability to create or imagine objects that do not immediately exist in reality. Humans can spontaneouesly learn the underlying properties of the world and generate the hypotheses of the future. This procedure is similiar to supervised learning and reinforcement learning with little mount of labeled data, but generalizes very well to many unseen scenarios and has a high level of robustness[19].

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

Generative Adversarial Networks

Generative Adversarial Networks(GANs)[20] 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

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.9}$$

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

————————image here————————-

GANs have shown a great success in many applications such as generating high-resolution images which are difficult to distinguish from the real ones and the ability to learn the complicated distributions. However, the main challenge of GANs is the instablitity 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[21].

Some works have been done to solve these problems. For example, Wasserstein GANs[22] use a different loss function to stablize the training process and avoid the mode collapse. Spectral Normalization[23] is another method to stablize the training process by constraining the Lipschitz constant of the discriminator.

Variational Autoencoders

Variational autoencoders(VAEs)[24] 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

$$\min_{E,D} \mathbb{E}_{x\sim p_{data}(x)}[\mathbb{E}_{z\sim q(z|x)}[\log p(x|z)]] - KL(q(z|x)||p(z)) \tag{2.10}$$

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 to encode the input data far apart in the latent space, which will cause the model to generate unrealistic data.

————————image here————————-

And the reparameterization trick[25] is introduced afterwards 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 following expression where ϵ is a random variable sampled from a standard Gaussian distribution.

$$z \in \mathcal{N}(\mu, \sigma^2) \longrightarrow z = \mu + \sigma \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1) \tag{2.11}$$

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 alse 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[26] are a familiy 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 densitiy "flows" through the sequence of invertible mappings and become a valid distribution.

————————image here————————-

The basic rule for transformation of densities considers an inverible, smooth mapping $f : \mathbb{R}^D \rightarrow \mathbb{R}^D$, 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:

$$q(z^{'}) = q(z) \left| \det \frac{\partial f^{-1}}{\partial z^{'}} \right| = q(z) \left| \det \frac{\partial f}{\partial z} \right|^{-1} \tag{2.12}$$

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

$$z_K = f_K \cdot ... f_2 \cdot f_1(z_0) \tag{2.13}$$

$$lnq_K(z_K) = lnq_0(z_0) - \sum_{k=1}^{K} ln \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right| \tag{2.14}$$

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.

Normalizating flows provide a flexible framework for modeling complex distributions, which is difficult to achieve with previous generative models. However the samples that are generated througn 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.

Transformer

————————Add if needed————————-

Diffusion Model

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 Denosing Diffusion Probabilistic Models(DDPM) was initialized by Sohl-Dickstein et al[27] and proposed recently by Ho. et al[28].

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 precess, which we call it sampling process or reverse process.

————————image here————————-

The great succuss 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 diffision model is that it is large-scale, flexible and offer high-quality samples. With the tradeoff of the relative longer training time and inference time because of its 2-phases architecture, it can synthesize highest-quality images than other generative models. This potential motivates us to apply the diffusion model also to 3D domain and the related tasks.

————————image here————————-

2.2.2. Theory and Fundamentals

In this section, we will introduce the detail 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 x_t with the conditional distribution $q(x_t|x_{t-1})$ defined as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{2.15}$$

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:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.16}$$

————————image here————————-

Our goal is to track the noised data at an arbitrary step t with a close-form posterior distribution $q(\mathbf{x}_t|\mathbf{x}_0)$. So the reparemeterization 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 $\boldsymbol{\epsilon}_0, ..., \boldsymbol{\epsilon}_{t-2}, \boldsymbol{\epsilon}_{t-1} \sim \mathcal{N}(0, \mathbf{I})$, 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, $\mathcal{N}(0, \sigma_1^2\mathbf{I})$ and $\mathcal{N}(0, \sigma_2^2\mathbf{I})$, the new merged distribution is $\mathcal{N}(0, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$. 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:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} \\
&= ... \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}
\end{aligned} \tag{2.17}$$

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

$$\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{2.18}$$

where α_t and $\bar{\alpha}_t$ can be precomputed for any arbitrary step t from β_t. The variance hyperparemeter β_t is normally chosen as a linear, quadratic or cosine schedule. The orginal 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 $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. Practically, the reverse conditional distribution is not directly tractable, because the computations involve the whole data distribtion. 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:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{2.19}$$

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

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \tag{2.20}$$

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

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbf{I}) \tag{2.21}$$

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

$$\tilde{\beta}_t = \frac{1-\tilde{\alpha}_{t-1}}{1-\tilde{\alpha}_t}\cdot\beta_t \tag{2.22}$$

$$\tilde{\boldsymbol{\mu}}_t = \frac{\sqrt{\alpha_t}(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t}\mathbf{x_t} + \frac{\sqrt{\tilde{\alpha}_{t-1}}\beta_t}{1-\tilde{\alpha}_t}\mathbf{x_0} \tag{2.23}$$

Thanks to the reparemeterization trick, we can represent $\mathbf{x_0} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x_t} - \sqrt{1-\tilde{\alpha}_t}\epsilon_t)$ from 2.17 and further simplify the expression of $\tilde{\boldsymbol{\mu}}$ as:

$$\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\epsilon_t\right) \tag{2.24}$$

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:

$$\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{KL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\right] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] =: L
\end{aligned} \tag{2.25}$$

To make the lower bound L computable, the expression can be further rewritten after some manipulations in Appendix of [28] as:

$$L = \mathbb{E}_q\left[\underbrace{D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0)\|p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^{T}\underbrace{D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t)\|p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}\underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}\right] \tag{2.26}$$

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 seperate decoder in the original model[31].

The second term L_{t-1} describe 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 natual way, we use L_t in the following calculation.

Revisit the reverse process from 2.19, we need to train $\boldsymbol{\mu}_\theta$ to approximate $\tilde{\boldsymbol{\mu}}_t$ in 2.24, 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:

$$\boldsymbol{\mu}_\theta = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \tag{2.27}$$

The lost term L_t can be formulated using l_2 distance:

$$L_t = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\,\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\|_2^2}\left\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\right\|^2\right] \tag{2.28}$$

which can be simplified ignoring the weighting term according to the original paper[28] as:

$$L_{simple} = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\tilde{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\tilde{\alpha}_t}\boldsymbol{\epsilon}_t, t)\right\|^2\right] + C \tag{2.29}$$

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 $\boldsymbol{\Sigma}_\theta$.

Conditional Diffusion Model

Conditional diffusion, also called guided diffusion is very practical in many applications since we normally want to generate the data in paricular 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. Modifiy the equation 2.20, we get

$$p_\theta(\mathbf{x}_{0:T}|y) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, y) \tag{2.30}$$

Extensions

2.2.3. Applications

3. Related Work

4. Methodology

5. Experiments

6. Discussion

7. Conclusion

A. Additionally

You may do an appendix

List of Figures

List of Tables

Bibliography

[1] C. Jones, A. Smith and E. Roberts, "Article title," in *Proceedings Title*, vol. II. IEEE, 2003, pp. 803–806.

[2] S. Peng, Y. Liu, Q. Huang, X. Zhou and H. Bao, "PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, Jun. 2019, pp. 4556–4565. [Online]. Available: https://ieeexplore.ieee.org/document/8954204/

[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," *CoRR*, vol. abs/1711.00199, 2017. [Online]. Available: http://arxiv.org/abs/1711.00199

[5] H. A. Hashim, "Special orthogonal group so(3), euler angles, angle-axis, rodriguez vector and unit-quaternion: Overview, mapping and challenges," *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 *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, 2020, pp. 751–756.

[7] Y. Zhu, M. Li, W. Yao and C. Chen, "A review of 6d object pose estimation," in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 10, 2022, pp. 1647–1655.

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

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

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

[11] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 239–256, 1992. [Online]. Available: https://api.semanticscholar.org/CorpusID:21874346

[12] R. A. Rill and K. Faragó, "Collision avoidance using deep learning-based monocular vision," *SN Computer Science*, vol. 2, 09 2021.

[13] G. Marullo, L. Tanzi, P. Piazzolla and E. Vezzetti, "6d object position estimation from 2d images: a literature review," *Multimedia Tools and Applications*, vol. 82, pp. 1–39, 11 2022.

[14] E. Brachmann, "6D Object Pose Estimation using 3D Object Coordinates [Data]," 2020. [Online]. Available: https://doi.org/10.11588/data/V4MUMX

[15] 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," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

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

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

[18] 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," 2019.

[19] A. Lamb, "A brief introduction to generative models," 2021.

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

[21] A. Borji, "Pros and cons of gan evaluation measures," 2018.

[22] M. Arjovsky, S. Chintala and L. Bottou, "Wasserstein gan," 2017.

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

[24] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

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

[26] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," 2016.

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

[28] J. Ho, A. Jain and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[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," 2022.

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

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

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

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, TBD Date of sign. Student's name TBD