TÉCNICO LISBOA

Trajectories in Hand Synergy Spaces for in-hand manipulation

Jorge Miguel Costa Telo

Thesis to obtain the Master of Science Degree in

Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Plinio Moreno López

Examination Committee

Chairperson: Prof. João Manuel de Freitas Xavier
Supervisor: Prof. Plinio Moreno López
Members of the Committee: Prof. Bruno Duarte Damas
Prof. Alberto Manuel Martinho Vale

November 2023

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my grandmother. To my brother, thank you for encouraging me to keep fighting for this goal.

I would also like to acknowledge my dissertation supervisor Prof. Plinio Moreno López for his insight, support and sharing of knowledge that has made this Thesis possible.

Abstract

Postural synergies are an important case of study to facilitate the control of dexterous artificial robot hands. To achieve this, latent spaces (synergy spaces) are generated and trained from grasp postures and are used to directly control a robot hand in this space. In this work, we base ourselves in an already achieved model and propose the addition of specific Riemannian metrics to achieve better results than the original model. To evaluate the results, the values for reconstruction error and accumulated smoothness of paths in a space are used to evaluate in-hand regrasping tasks. Different Riemannian metrics lead to different results and should be used with different tasks in mind. The arguments presented in this work are validated by calculating accumulated smoothness based on a set group of in-hand regrasping tasks.

Keywords

Postural Hand Synergies; Latent Space; Dimensionality; Riemannian Geometry; Smoothness

Resumo

Sinergias posturais de mãos são um caso de estudo importante para facilitar o controlo de mãos robóticas hábeis. Para conseguir isto, espaços latentes (espaços de sinergia) são criados e treinados a partir de posturas de pegas e são usados para controlar diretamente uma mão robótica neste espaço. Nesta tese, vamos basearnos num modelo criado num trabalho anterior e propomos a adição de métricas de Riemann para obter melhores resultados do que o modelo original. Para avaliar os resultados, os valores de erro de reconstrução e a suavidade acumulada de caminhos num espaço são usados para avaliar tarefas de reaplicação. Métricas diferentes levam a resultados diferentes e devem ser usadas em tarefas diferentes. Os argumentos presentes neste trabalho são validados ao calcular suavidade acumulada, baseados num grupo de tarefas de reaplicação.

Palavras Chave

Sinergias Posturais de Mão; Espaço Latente; Dimensionalidade; Geometria Riemanniana; Suavidade;

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

AE	Auto-Encoder
BCGPLVM	Latent Variable Model based on Gaussian Processes (GPLVM) with back constraints
CVAE	Conditional Variational Auto-Encoder
DoF	Degrees of Freedom
ELBO	evidence lower bound
GPLVM	Latent Variable Model based on Gaussian Processes
KL	Kullback-Leibler
LVM	Latent Variable Models
MSE	Mean Squared Error
PCA	Principal Component Analysis
VAE	Variational Auto-Encoder

1

Introduction

Contents

This chapter will go through the motivation of this thesis, why the study of synergies in robotic hands is important and how much technology has helped the research in this field. It will also go through what are the goals of this thesis as well as how the rest of the document is organized.

1.1 Motivation

With the evolution of technology and the pursuit of understanding more about how the human brain directs the human body, engineers have been attempting to recreate the human body utilizing artificial tools, like robots, for a multitude of reasons, be it for medical purposes (creation of prosthesis) or for industrial use to reduce human error in production lines.

One specific part of the human body that is a common case of study due to its many uses, is the human hand.

Although in a superficial view, the human hand can be seen as a simple tool our body has to grasp objects or do certain motions, it is, in reality, a complex system of muscles, tendons, nerves and bones, with a high amount of Degrees of Freedom (DoF), [3]. If we were to control such a system in a computer environment, a high amount of processing power would be required. Due to how the human body behaves, this requirement is diminished by the existence of constraints integrated into this system, as for example, the fact that the fingers cannot bend backwards, or the fact that the muscles and tendons are intertwined, which makes moving a finger have an involuntary movement of its neighbors. These constraints are used to reduce number of possible DoF, as shown in [3] and [4]. This leads to a reduction of the control inputs required to control such a system, facilitating the task at hand.

The study of this simplification is the goal of this thesis. To attempt and reduce the control space between the user and the robotic hand used, so that controlling the hand to move or even hold objects require a small amount of control inputs, and not one for each individual DoF. Specifically, this project is a continuation of the work done in [2], in an attempt to improve the discussed solution to the dimensionality question.

1.2 Objectives

This work will utilize the Conditional Variational Auto-Encoder (CVAE) model designed in [2]. The CVAE model generates a lower dimensional space (i.e. latent space), where we aim to plan and execute in-hand manipulation tasks. The models developed should analyze the structure of the real space and, with added constraints, construct a smoother latent space. The main focus of this thesis is to study several options to change the structure of the latent space, providing a smoother latent space for in-hand manipulation, by introducing metrics in the training methodology of the model to better represent

characteristics of the real space that represents the DoF of the robotic hands used, so that regrasping can be done with less cost.

For the training of the models used in this work, 2 data sets, obtained in a previous work, are used as a starting point. This makes it so that we have information regarding each point in the real space, which corresponds to a certain combination of hand grasp and object type.

1.3 Contribution

By analysing and altering the model developed in [2] and introducing Riemannian metrics into the generated latent space this thesis hopes to achieve a smoother modeling of the latent space associated to a robotic hand, so that regrasping can done with less cost. This work also serves as a entry work for more complex modeling of latent spaces, to help understand how different metrics or parameter fine tuning can help latent space generation.

1.4 Organization of the Document

This thesis is organized as follows: Chapter 1, where an introduction to the problem this thesis plans to tackle is given. In Chapter 2 synergy will be discussed as a control problem, as well as, models utilized before to tackle the dimensionality problem synergies are based on. Chapter 3 focuses on the methods used in this work, their planning and gives a quick view on the two robotic hands used for simulation. Chapter 4 will focus on the solutions and their implementation to try and achieve the goal mentioned before. Chapter 5 is where the evaluation of the solutions implemented in this work will be discussed. Chapter 6 will offer conclusions on the results obtained, and possible options of future work to be done, taking advantage of the results obtained in this work.

2

Synergy as an Engineering problem

Contents

Synergy is a problem previous works have tried to solve by using different approaches. Most of these approaches take advantage of learning models to discover a low dimensional space that can represent synergies present in the human hand. This chapter focuses on the understanding of what are synergies in the context of engineering and previous works that have been done to present solutions for this problem.

2.1 Background of Hand Control Synergy

The human body has been the object of study for many centuries, with the brain representing the most interesting. One of the things scientists are most curious about is the way the brain controls the various systems present in the human body, including hand control, the main point of this project. As theorized in [3], [5], [6] and [7], the human brain is able to reduce the amount of control inputs it requires to control the human hand. The study in [3] refers to the existence of at least 60 DoF,by just taking into account the complete biomechanical model while ignoring muscles acting on the wrist and radioulnar joints or additional DoF associated with contact forces, which even for a computer controlled system is a high amount of variables to control. In control systems, a high amount of DoF leads to a higher energy requirement to achieve the tasks required of the system. To achieve a smaller number of DoF in the hand control system, the human brain takes advantage of synergies, and one of the ways it achieves this is by utilizing hand constraints.

2.1.1 Hand Constraints

As all synergy problems, one aspect of the system that helps reduce the dimensionality is the existence of constraints. In the human hand, these are knows as biomechanical constraints. As the starting point, we cannot forget that the human hand is first and foremost a complex biomechanical structure, that includes 27 bones that are actuated by 18 intrinsic and 18 extrinsic muscles, that are controlled by a complex web of tendons. An initial measurement of complexity of the hand requires the consideration of the number of joints existing inside of the structure of the hand. A complete biomechanical model of a human hand would consist of four DoF for each finger, with 5 for the thumb, with an extra DoF at the radioulnar joint and 2 at the wrist. Adding to these, we also need to include the 36 muscles that act on the thumb and fingers, as well as the complex web of tendons that actuate the hand, bringing the number of possible DoF to at least 60, as seen in Figure 2.1. This number still does not include the muscles acting on the wrist or the radio-ulnar joint, or any additional degree associated with contact forces.

Regarding the constraints, as seen in [3] and [5], it is possible to easily test the existence of some very basic ones. One can check that a finger cannot bend backwards, or that moving 1 finger moves the

2 closest to it, or that touching the thumb with any other finger or even touching the middle of the palm with any finger, including the thumb, generates a movement of the entire hand. These constraints are a tool used to simplify how many signals are required to make the hand perform the grasps required for daily actions.

Knowing about these constraints might sound redundant, but the knowledge of their existence serves as a basis for understanding how the brain might achieve a reduction of dimensionality of its control over the hand, and even give an idea of what type of procedure one should use to achieve the same result but with a robotic hand.

Figure 2.1: Anatomy of human hand, showing coupled tendon structure.

As mentioned in Chapter 1, the idea of this thesis is to answer the problem of dimensionality using the idea of synergies mentioned in this chapter, with the goal of simplifying the control of robotic hands.

In control systems, the amount of control inputs needed correspond to how many inputs are needed to be controlled or fed to the system to perform a certain task or tasks that the system is designed to perform. When the system to be controlled increases in complexity, which in the context of this work, is considered to be the amount of DoF, the amount of computational power required to do that control increases. Using the idea of synergies referred in this work, it is possible to implement a model with the goal of reducing the amount of power required to perform the same desired tasks, by utilizing a reduction

of the amount of DoF required.

2.2 Reduction of Dimensionality

The synergies used by the human brain to control the human hand, can be seen in an engineering environment as the reduction of dimensionality of the control space.

A dimensionality reduction technique consists in having an initial n-dimensional space, R^{d_x}, denominated as real space, where the aim is to find a mapping e that encodes a data point $x \in R^{d_x}$ into a point in a low dimensional space, R^{d_z}, denominated as latent space, $z \in R^{d_z}$, where $d_z << d_x$, and the reverse mapping d that decodes the low-dimensional point into the high-dimensional point. These mappings are then parameterized by two vectors, θ and ϕ. The different methods used to recover these vectors use different approaches to recover these vectors, given a dataset of observations X, that minimize a optimization criteria previously chosen.

Current studies to the problem of hand synergy implementation in robotic hands all follow the idea that this is an optimization problem for the reduction of dimensionality, which can be solvable by software solutions [8], [9].

To achieve the results wanted, most attempts based themselves on the following methods.

2.2.1 Principal Component Analysis

The most basic process used is the Principal Component Analysis (PCA). This process assumes that the mapping e, that encodes the high-dimensional point into the low-dimensional point, is linear, following the relation $e(X) = xW$, where $W \in R^{d_x * d_z}$, [10].

The most common used optimization criteria in this approach is the minimization of the mean squared error between the original input x and the reconstructed signal from the embedding. This minimization problem is possible to be solved analytically, computing the optimal value for W, which corresponds to the columns of this matrix being the principal components of X.

2.2.2 Auto-Encoder

The next most used process is the Auto-Encoder (AE). This process consists in an artificial neural network that is then used to learn efficient codings of unlabeled data, also known as unsupervised learning. This encoding is validated and refined by attempts at regenerating the input from the encoding. The auto-encoder learns how to represent a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data (”noise”), [11].

As previously mentioned, the AE has two parts. The encoder, that maps the high-dimensional input, x into a low-dimensional signal, z, and then the decoder, that does the reverse operation.

Unlike the PCA, the mapping from high-dimensional space to low-dimensional space in this case has a non-linear relationship, but both processes utilize the same optimization criteria, the mean squared error. Using the vectors θ and ϕ, this can be represented by:

$$\begin{aligned} \theta &: R^{d_x} \rightarrow R^{d_z} \\ \phi &: R^{d_z} \rightarrow R^{d_x} \end{aligned} \tag{2.1}$$

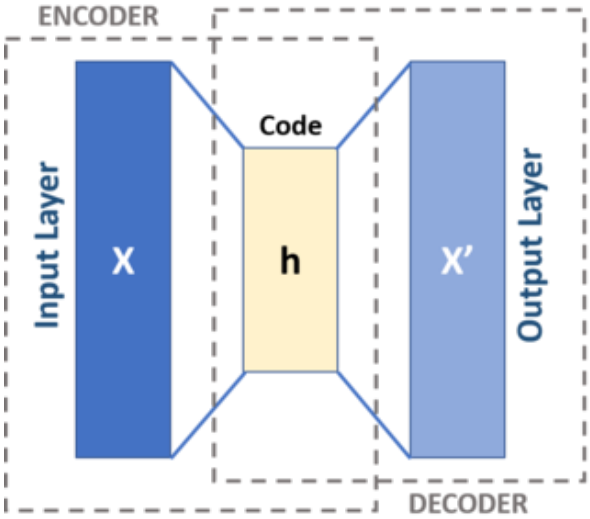

Figure 2.2: Scheme of a basic Auto encoder

2.2.3 Latent Variable Models

The last type of used process are Latent Variable Models (LVM). These models take a probabilistic approach on the problem of reduction of dimensionality, [9]. These models assume that the inputs x, in a high-dimensional space, are generated by unobserved latent variables through a probability distribution. These latent variables are low-dimensional signals that encode the information needed to generate new input samples. In mathematical terms, this is represented by a probability distribution $p(x, z; \theta)$, where x are the observed data points in a high-dimensional space, z are the latent variables in a low-dimensional space and θ is the vector with the parameters to be learned.

One of the most common LVM is the Latent Variable Model based on Gaussian Processes (GPLVM), [12], [9]. As the name indicated, this model utilizes Gaussian Processes to represent the mapping e, which means that each point in latent space is connected by a Gaussian process mapping to a point in a high-dimensional space. For this model, the optimization criteria is based on maximizing the likelihood

of the data, which is normally achieved with a gradient based method. By using the equation for a Gaussian probability distribution, the mapping will have the following distribution:

$$P(\mathbf{Z}|\mathbf{X},\theta) = \prod_{j=1}^{D} \frac{1}{(2\pi)^{\frac{N}{2}}|\mathbf{K}|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{z_j}^T\mathbf{K}^{-1}\mathbf{z_j}} \tag{2.2}$$

This model has a computationally expensive encoding for new data points, due to the optimization process requiring to be repeated in order to perform inference.

2.2.4 Previous Works related to Synergy

In [4] and [3], the author of these studies has shown that the fingers of the human hand are not individually controlled for each possible hand grasp and/or movement, but instead the brain utilizes a synergistic framework that utilizes both neural signals, as well as muscles and kinematic constraints present in the hand. These couplings are what are called synergies, and the hands are not the only part of the body where these synergies can be found.

Just like in [4], both [13] and [14] show the existence of these synergies using a PCA approach. These studies use this model to try and find a possible low dimensional space that is able to represent the 12 previously recorded hand grasps/postures. The recovered basis components of the process used are denominated *eigengrasps* and the combination of these parameters leads to new possible grasps/-postures. They also show that control algorithms have a better performance, in terms of searching for a more stable grasp for certain tasks, by utilizing the low-dimensional space.

In [15], the PCA method is again used, but this time with utilizing precision grasps. This dataset of prevision grasps was achieved by a human operator controlling two dexterous robotic hands. In this work, the discovered synergies are represented in robot joint angle space, which allows for a direct application of grasps/postures. This study concludes that for both hands tested, it is sufficient to use 5 or 6 dimensional spaces to express even the most complex prevision grasps.

[16] and [17] compare several classical dimensionality reduction methods for encoding the control of a robotic hand in 2D subspaces. The methods were applied on recorded hand movements and evaluated based on the inconsistency and continuity of the representations, while a method for denoising graphs consisting of embedded grasp postures was proposed. They conclude that the Isomap method learns more consistent and continuous embeddings.

In [12] and [18], a more sophisticated approach is presented. One utilizing LVM, specifically, GPLVM, changing the deterministic nature of the problem into a more probabilistic nature. In [12] the goal was to try and evaluate a better method for synergy representation and it was done by using two types of evaluation, one for the reconstruction error of the method and another for semantic evaluation. The reconstruction error shows how much the generated mapping that connects the observed and the latent

representation can distort the dataset, and is achieved by pushing a point from the high-dimensional space into the low-dimensional space and pulling it back to the original dimensional space and registering the differences between the 2 points, before and after usage of the mapping. For this reconstruction error, [12] concludes that all attempts with a GPLVM process had a better performance than a PCA process, while utilizing the same dataset, which leads to the idea that human hand synergy is a non linear problem, which might be better solved by using a method that better fits to a non linear nature.

The semantic evaluation presented has 2 parts, the similarity between different test subjects executing the same grasp and dissimilarity across different grasps. For the first part, the author presents a visualization of a Mean Grasp Model which shows that the GPLVM outperforms the PCA, since it is able to follow the fingertips better for any given dimension. This does come as a cost, according to the author, due to the nature of the trajectories created by the GPLVM, which are likely to be problematic in planning, but overall better in most applications, since it's better to have rougher trajectories that stay true to the correct path, rather than smoother trajectories that follow a wrong one. In [18], the same result is reached, where the authors conclude that the problem of synergy might be better solved by a non linear model.

These studies show that utilizing this new approach leads to a smaller reconstruction error than the PCA approach. This result leads to the conclusion that although a PCA model is enough to show the existence of a reduction of dimensionality, a more complex model might be more useful when trying to improve the model used for encoding and decoding grasps. This is important for when the intended goal is not showing the existence of a possible latent space, but the actual use of said space to perform multiple tasks.

The addition of back constraints to the GPLVM, as seen in [19], GPLVM with back constraints (BCGPLVM), ensures that points in a high-dimensional space that are close to each other, translate into points in the latent space that are close to each other as well.

With the improvement of learning models and the increase in calculation power given by hardware, more complex models have been tested. The next approach, with AE, utilizes these new capabilities to create a model based on a non-linear, deterministic dimensionality reduction method, based on neural networks.

In [1] and [20] this approach is shown to improve the performance in terms of reconstruction and is able to take into account additional information such as the object size and/or shape, which enables a bigger scope of simulations and/or grasps to attempt while using a robotic hand.

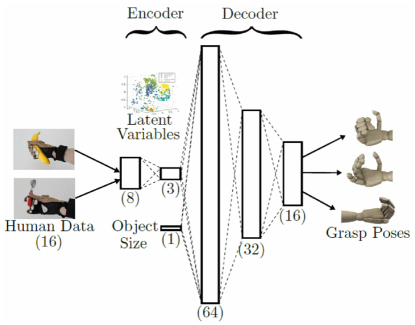

Figure 2.3: Structure of the trained AE, in [1]

2.3 Conditional Variational Auto-Encoder

For this thesis, as mentioned in Chapter 1, the learning model to solve the dimensionality problem is a CVAE, based on the one used in [2].

As the name indicates, this model is based on the AE architecture, most specifically, on the Variational Auto-Encoder (VAE) variant of AE, which provide a computationally efficient framework for parameter learning. These models are based on having an encoder network $q_\phi(z|x)$ and a decoder network $p_\theta(x|z)$. The difference between AE and VAE is that while the former uses the output of the encoder directly as the latent variable, the latter samples the latent variable z from a Gaussian distribution, whose mean $\mu \in R^{d_z}$ and variance $\sigma \in R^{d_z}$ are computed by the encoder, when it is fed a data point x as input.

The training of the VAE model is based on variational inference, with the aim of minimizing the Kullback-Leibler (KL) divergence between the input of the encoder and the output of the decoder. This can be interpreted as an approximate maximum likelihood procedure, with the change that instead of trying to maximize the likelihood of the data, the goal is to maximize the evidence lower bound (ELBO), which is computationally easier.

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta\left(x|z\right)\right] - D_{KL}\left(q_\phi\left(z|x\right)\|p(z)\right) \tag{2.3}$$

In Equation (2.3), the first term represents the Mean Squared Error (MSE), while the second terms represents the KL divergence, which is used to try and resemble a Gaussian distribution of the points in the latent space.

In [2] the author extends the idea of this model, with the addition of a conditional variable, following the procedure described in [21], transforming the model into a CVAE. To do this, the extra variable, denominated c, is used to condition both the encoder and the decoder network. This variable can be either continuous or discrete, changing the ELBO into:

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x,c)}\left[\log p_\theta\left(x|c,z\right)\right] - D_{KL}\left(q_\phi\left(z|x,c\right)\|p(z)\right) \tag{2.4}$$

The optimize the loss function given by Equation (2.4), standard gradient descent algorithms are utilized, like the ones presented in [22].

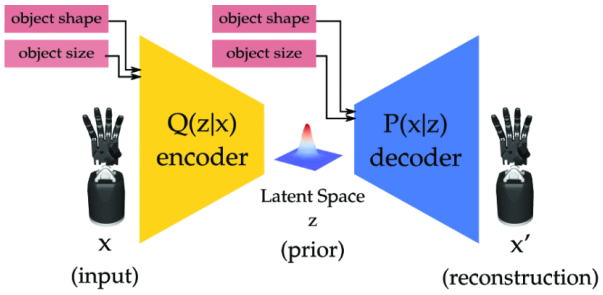

Figure 2.4: Structure of the trained CVAE, in [2]

This study utilized the dataset provided in [15]. As conclusions, the author shows that although the CVAE model shows a worse result regarding MSE as a reconstruction metric in comparison with approaches like the PCA, it does learn a smoother latent space, which translates into a better performance on tasks that are based on changing grasps smoothly..

2.4 Riemannian Geometry

One issue with problems involving dimensional reduction is the possibility that the latent space might not correctly represent geometric relations between paints in the real space, as in, the two closest neighbors in the real space might not be the same pair in the latent space, which might lead to substantial challenges when trying to transverse the latent space to conclude a certain task in the most efficient way.

To grapple with this intricate issue, we turn our attention to the profound and versatile field of Riemannian Geometry. As a sophisticated branch of differential geometry, Riemannian Geometry provides a comprehensive framework for studying the intrinsic geometric properties of smooth, curved spaces, encapsulated within the realm of Riemannian Manifolds.

2.4.1 A brief introduction to Riemannian Geometry

Riemannian Geometry is the branch of differential geometry that studies Riemannian Manifolds.

These manifolds, [23], are well-defined metric spaces, where the inner-product is defined only locally and changes smoothly throughout the entire space.

A smooth manifold is a topological space, which locally is homeomorphic to an Euclidean space. An intuitive way to think of a n-dimensional smooth manifold is as an embedded non-intersecting surface M in an ambient space $X = R^D$, with $D > n$. In this case, the tangent space T_xM is a d-dimensional vector

space tangential to M at the point $x \in M$. Hence,$v \in T_xM$ is a vector $v \in R^D$ and the Riemannian metric is $M_X : M \to R^{DxD}$, as shown in Figure 2.5.

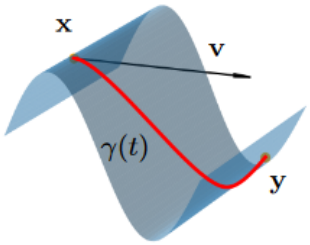

Figure 2.5: Example of a Riemannian manifold

The simplest approach is to assume that X is equipped with the Euclidean metric $M_X(x) = I_D$ and its restriction is utilized as the Riemannian metric on T_xM. Since the choice of $M_X(.)$ has a direct impact on the manifold itself, the metric can be modifiable in a way to be designed to encode high-level information.

The important point of using Riemannian Manifolds for the problem at hand, is that it can be used to simplify calculations. Due to the way on how the lower dimensional space, $Z = R^z$ can be designed using a Riemannian metric, and the fact that the metric, if well designed, can store high-level information from the high dimensional space, $X = R^x$, most importantly, the proximity of points this space, the shortest paths in X can be computed in Z. Since the calculations are done using a lower dimensional plane, they require less energy to do.

One added point, is that by introducing the metric to the learning model used to create the latent space, more sophisticated relations between points in the real space can be learned and/or discovered.

2.4.2 Previous works related to Riemannian Metrics

Previous works utilizing this type of approach, show how introducing these type of metrics can help to create shortest paths in a high dimensional space, while only requiring to transverse the lower dimensional space obtained from the original space.

In [24] the author gives a quick explanation on what Riemannian Geometry is and how using these metrics can be useful to the calculation of shortest paths in manifolds obtained from high dimensional datasets.

In [25], the author shows how considering the high dimensional space as a Riemannian manifold instead of the usual consideration that it is an Euclidean space, which allows the possibility of encoding domain knowledge through well defined Riemannian metrics. Using this, one is able to define shortest paths accordingly in the latent space, which follow both the learned manifold as well as respecting the high dimensional geometry. The author also mentions that if the used metrics are carefully designed for

specific tasks, they can ensure that shortest paths are well-behaved even for deterministic generators, which usually suffer from a misleading bias.

3

Methodology

Contents

The methods presented in this chapter will be used in an attempt to provide a better encoding model of a high dimensional space, such that the new model contains more complex relations between data points in this space in a lower dimensional space.

This can be translated as when transversing the latent space, from one encoded grasp to another, the new model provides a shorter path than the original CVAE model.

The shortest path metric used in this work is the sum of the differences in each dimension of the group of grasps that belong to a path. This value is considered using the high dimensional space, since it is in this space that the hand to be controlled operates.

3.1 Robot hands

For this work, two robotic hands were utilized, with different amounts of DoF. Basing the datasets on the ones tested and obtained in [15], the two hands used for simulations are the iCub Robot hand and the Shadow C5 air-muscle hand.

3.1.1 iCub Robot Hand

The iCub robot hand is the hand attached to the baby robot iCub, based on a 18 month to 2.5 year old child. These hands have a total 19 DoF, not counting the wrist, but only has 9 actuators for each hand. This makes it so that for controlling these hands, only 9 control inputs are required. The dataset for this hand is given in a R^9 space and has 536 recorded grasps. The actuators in this hand are separated as: 3 for the thumb, 2 for the index, 2 for the middle finger, 1 for the coupled ring and little fingers and 1 final one for the adduction/abduction.

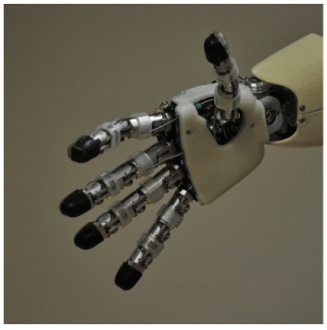

Figure 3.1: iCub Baby Robot hand

3.1.2 Shadow Robot Hand

The Shadow C5 air-muscle hand was developed to closely resemble the size and shape of a human hand and provides a total of 24 DoF. These include 2 DoF for the wrist and 1 for palm flexure, which are not used, dropping the total amount to 21. These 21 DoF are presented as: 4 for each finder and 5 for thumb.

Figure 3.2: Shadow Robot Hand

3.2 Conditional Variational Auto-Encoder

As mentioned in Chapter 1, the model in this worked will be based on the model given in [2], and the grasp dataset will be the same as the author used, which coincides to the grasps obtained in [15].

A CVAE model is based on the VAE architecture, where an encoder is used to encode high dimensional data into a lower dimensional data, and a decoder is used to the reverse operation.

The VAE model is created by utilizing the dataset $X = \{x_1, x_2, x_3, ..., x_n\}, x_i \in R^{d_x}$, from the space R^{d_x} where d_x equals to the amount of DoF a robotic hand has,that contains the saved postures obtained in [15], to train an encoder to model a probability distribution $q(z|x)$. With this, it is possible to encode a data point x_i from the high dimensional space into a data point z_i from the low dimensional space, $R^{d_z}, d_z \leq d_x$. The decoder part of the model is trained to the inverse operation. From a dataset $Z = \{z_1, z_2, z_3, ..., z_n\}, z_i \in R^{d_z}$, the decoder is trained to model a probability distribution $p(x|z)$, the decodes a lower dimensional data point, z_i, into a higher dimensional data point, x_i.

The Loss function used to train the model is given in Equation (2.3), where the first term corresponds to the reconstruction error between the input of the encoder and the output of the decoder. The second term, that corresponds to the KL divergence between the probability distribution of the latent variables and the normal Gaussian distribution. Optimizing the second term of the Loss function can lead to a smoother latent space, while also reducing the reconstruction error of the model. This optimization process is done by utilizing standard gradient descent methods, as shown in [22].

The produced latent (low dimensional) space of $Z \in R^{d_z}$ will represent the synergistic components of the hand, which require decoding before being able to have a physical representation by moving the robotic hand.

To create the CVAE model, an extra variable, c, was added as a condition to the model described. This variable contains information about the object that is to be grasped, specifically it's size and shape. Adding this new condition requires a new formulation of the sets used, and the probability densities. The high dimensional dataset becomes $X_c = \{(x_1, c_1), (x_2, c_2), (x_3, c_3), ..., (x_n, c_n)\}, x_i \in R^{d_x}, c_i \in R^{d_c}$ and the encoder now models the distribution $q(z|x, c$. In reverse, the latent space dataset becomes $Z_c = \{(z_1, c_1), (z_2, c_2), (z_3, c_3), ..., (z_n, c_n)\}, z_i \in R^{d_z}, c_i \in R^{d_c}$ and the decoding distribution $p(x|z, c)$.

As the datasets and the probability densities changed to accommodate the new added condition, the Loss function has to suffer changes, as shown in Equation (2.4).

Adding the conditional variable to the model helps it create a more robust latent space, by grouping a certain grasp to certain object size and/or shape, which might lead to a better learning pattern for hidden relationships in the original data.

3.3 Smoothness as a goal

The goal of this thesis is to attempt to find a way to achieve a better representation of a high dimensional space in a lower dimensional space, to make regrasping procedures more efficient.

Efficiency in the context of this work is considered as smoothness. Smoothness is given by Equation (3.1), where $f(z_i)$ is the joint angle of a decoded posture.

$$S(z_1, z_2) = ||f(z_1) - f(z_2)||_2 \tag{3.1}$$

From this

This notion of smoothness is presented in [26]. This means that the goal is to try and minimize this value and can be described as trying to make it so that the neighboring points in the high dimensional space are also neighboring points in the latent space. If this is achievable, then when transversing the latent space utilizing paths, that are a groups of grasps in a needed for a regrasping task, from one initial grasp to a final grasp, if one uses the shortest path, than that path leads to the smallest movement of the joint angles of the controlled hand.

In this work, we focus on trajectories inside the latent space to replicate in-hand manipulation tasks. Using the notion of smoothness presented in Equation (3.1), the idea of accumulated smoothness is used as an evaluation value. This value is described as the smoothness between joint angles present in a single trajectory, as seen in Equation (3.2), where a trajectory has an n amount of steps in it.

$$AS(z_1, z_2) = \sum_{i=1}^{n-1} ||f(z_i) - f(z_{i+1})||_2 \tag{3.2}$$

3.4 Regrasping Algorithms

To test the trained model, the latent representation will be tested utilizing 3 different pathing algorithms. These are simple procedures and are used to try to understand how the model behaves when training parameters are changed. The first of these algorithms will be used twice. The first use will consider high dimensional distances, while the second use will consider low dimensional distances for the purpose of passing to neighboring points.

All the paths created by these algorithms will then be decoded, if necessary, so that a value for the smoothness of the path can be quantified.

3.4.1 Naive Neighbor

The first algorithm, and the most time consuming since it does not provide the proper shortest path is a Naive Neighbor algorithm. This pathing algorithm is used so that it is possible to compare both spaces and check if it was possible to make the neighboring pairs in the high dimensional space to be modeled correctly in the low dimensional space. In a perfect scenario, the total smoothness of a path using this algorithm would be the same if high dimension neighbors were used or if low dimension neighbors were used.

Algorithm 3.1: Naive Neighbor Pathing

$a, b \in R^{d_z}$ $a \leftarrow starting\,point$
$b \leftarrow ending\,point$
Initialize path array *P* = [ ]
while $a \neq b$ **do**
    Generate grasp posture $g_a = decode(a)$
    for each point $z_i \in R^{d_z}$ **do**
        Generate grasp posture $g_i = decode(z_i)$
        Compute distance $d_{ai} = ||g_a - g_i||_2^2$
        if $d_{ai} = smallest\,dist$ **then**
            Append z_i to *P*

In Algorithm 3.1, at the end of the run time, the array *P* will have the path, given in latent space points, needed to go from a starting point to the ending point of a pair, by only using neighboring points, as in, the pathing is only done by going to the next closest point, with no importance in it being the shortest path or not. Note that in order for this algorithm to not get stuck in a pair of points, it is required to add a condition that the path cannot have the same point twice in it, as in, there can be no backtrack.

3.4.2 A*

The next algorithm to be used is based on the commonly used A* pathing algorithm, that is used to find shortest paths in a dataset. Unlike the previous mechanism, this algorithm is specifically used to see how the smoothest path gets affected by different training parameters.

Since this type of algorithm is used to reach the shortest path between two points in a dataset, it is useful to see if the trained model can have efficient pathing between grasps, and minimize unnecessary movement.

Algorithm 3.2: A* Pathing

$a, b, n \in R^{d_z}$ $a \leftarrow starting\,point$
$b \leftarrow ending\,point$
$h(n) \leftarrow heuristic\,that\,estimates\,cost\,from\,n\,to\,b$
$f \leftarrow point\,priority$
$v(i) \leftarrow distance\,up\,point\,i$ Initialize path array *P* = [ ]
Initialize list of successors *L* = []
Initialize priority queue (min-heap) open_set
Initialize list of predecessors *C* = []
while $open\_set \neq empty$ **do**
    a = point in open_set with lowest f
    Remove Current from open_set
    if $a = b$ **then**
        Reconstruct path using *C*
        Return *P*
    Generate grasp posture $g_a = decode(a)$
    for each point $z_i \in L(Current)$ **do**
        Generate grasp posture $g_i = decode(z_i)$
        Compute distance $d_{ai} = ||g_a - g_i||_2^2$
        Save total distance up to $z_i, accum\_dist = v(a) + d_{ai}$
        if $z_i \notin open\_set || accum\_dist < d_{bi}$ **then**
            Set parent of z_i to a
            Set $accum_dist = d_{bi}$
            Calculate $f(z_i) = d_{bi} + h(z_i)$
            Return *P*
        if $z_i \notin open\_set$ **then**
            Add z_i to $open\_set$

The procedure in Algorithm 3.2 has a more complex implementation than the one in Algorithm 3.1, but the purpose of it is to search for the most cost efficient path possible between two points in the latent space. This means that if the algorithm is well defined, the resulting path will be the smoothest path between two points.

3.4.3 Interpolation

The last algorithm used is a simple interpolation algorithm, where given a number of steps, n a path can have, then $n-2$ intergrasps will be created and used. The main point of this algorithm is that while the previous two procedures only use data points present in the dataset, the interpolation method creates grasps that were not avaiable when training the model. This is an interesting algorithm to use, since it gives an idea if the trained model is able to create grasps that did not exist if it can create new grasps by itself that can be used.

Algorithm 3.3: Interpolation Pathing

$a, b \in R^{d_z}$ $a \leftarrow starting\,point$
$b \leftarrow ending\,point$
$n \leftarrow number\,of\,steps$
Initialize path array *P* = [ ]
for $i = 0; i \leq n; i++$ **do**
    $z = (1-i) * a + i * b$ Append z to *P*

3.5 Riemannian Metrics

As a way to encode more complex information regarding geometric relationships between points in the high dimensional space, the idea of Riemannian metrics, described in [27] were used.

The Riemannian metric utilized serves to capture the geometric properties of the high dimensional space. This metric described how distances between points in this space should be measured while considering the underlying structure and relationships in the dataset used. To promote smoothness in the latent space, the metric was designed to preserve the neighborhood relationships between points in the high dimensional space. This means if two points are neighbors in the original space, then the metric would serve to train the CVAE model in a way that those two points are also neighbors in the latent space.

With the metric defined, a matrix $G \in R^{d_z \times d_z}$ is derived from it and is used to define geodesic distances.

This matrix G, also given the name of metric tensor in Riemannian Geometry. For geodesic distances, the metric tensor captures the local geometry of the manifold at each point. This matrix is a symmetric and positive-definite matrix. Besides these conditions, there are few others that we should be aware of.

- The metric tensor G should be smooth and differentiable on the manifold. This ensures that the geometry varies smoothly across the manifold.

- The metric tensor must be compatible with the intrinsic geometry of the manifold. It should not introduce distortions that are not intrinsic to the manifold.
- The metric tensor should not depend on the specific coordinate system chosen. This property ensures that the geodesic distances calculated are intrinsic to the manifold and not an artifact of the coordinate system.
- The eigenvalues of G should be positive at each point on the manifold. This condition guarantees that the metric tensor defines a valid inner product space.

These geodesic distances represent the shortest path on the curved surface of the Riemannian manifold, which in this case correspond to the original high dimensional space. Using this matrix, it is possible to add terms to the Loss function of the CVAE model so that the metric is considered when training.

The utilization of the metric tensor in the training method can be described by Algorithm 3.4.

Algorithm 3.4: Training with G

Create G with the original data set x
Encode the original data set $z = encode(x)$
Calculate the pairwise distances in z, as $Pdist_z$
Calculate Geodesic distances $D_z = {Pdist_z}^T G Pdist_z$
Compare Geodesic Distances with Pairwise Distances in Real space

The final loss function then considers the introduction of the difference between the calculated geodesic distances between points, D_z, as well as the geodesic distance between each point and the center of the latent space, C_z

The geodesic distance between each point and the center of the latent space utilizes by the same steps as in Algorithm 3.4, but instead of the pairwise distance between points, we utilize the distance between each point and the center of whichever space we are working on.

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x,c)}\left[\log p_\theta\left(x|c,z\right)\right] - D_{KL}\left(q_\phi\left(z|x,c\right)\|p(z)\right) + D_z + C_z \tag{3.3}$$

With the algorithms planned and the metrics designed, implementation of these measures is the next step.

4

Implementation

Contents

In this chapter, an explanation of how the implementation of the procedures referred in Chapter 3 is given. As is common in working with training models, the metrics used for such tasks have to be adapted and updated to achieve the results that are wanted. In the case of this work, this happens when trying to implement the Riemannian metrics. Due to the how these are used, and due to the nature of the dataset, these metrics had to be redesigned as new results were obtained.

4.1 Data Preparation

The data used for this thesis is based on the grasps postures registered in [15]. The model will be trained and tested for both datasets, one for each of the hands shown in Chapter 3. For the iCub robot hand, the dataset presents 536 grasps and each of these grasps has the information of the object it was used on. This information is used to create the conditional variable, also called object type for purposes of implementation, used for training. For the Shadow robot hand, the dataset contains 438 grasps and their respective object information.

Using the dataset, the center of the data points belonging to the dataset is calculated and saved for use in the Riemannian metrics implementation. The centers of each group of data points that belong to a specific object are also obtained, for the possibility of testing different options regarding metric construction.

Pairs of points to test the pathing algorithms were created, with the condition that both points belonging to a pair belong to the same object type.

The last step of data preparation is to calculate the pairwise distance between points, which means creating a $N \times N$ matrix that contains the distance between points i and j.

4.2 Original Model

The CVAE model trained is based on the model used in [2]. The encoding part of the model consists on a linear transformation of the original data input array concatenated with the conditional array, while the decoding part is a linear transformation of the encoded grasp array concatenated with the conditional array.

The loss function is given by Equation (2.4), with the addition of a weight w for the KL term.

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x,c)}\left[\log p_\theta\left(x|c,z\right)\right] - w \times D_{KL}\left(q_\phi\left(z|x,c\right)\|p(z)\right) \tag{4.1}$$

The introduction of the weight can be used to train the model so that the distribution of the latent space resembles the distribution present in the high dimensional space.

4.3 Pathing Calculations

Regarding the pathing algorithms, the use of Algorithm 3.3 is straight-forward, as in, it can just be used as it is, with no need to decide if it should use real space distances or latent space distances.

Algorithm 3.1 has two uses. The first use considers the distance calculation utilizing real space distances, where the neighboring points are determined via their true distances. The second use is done by using latent space distances, as in, the neighboring points are decided via their distance in the latent space. Since it is difficult to maintain neighboring relationships between both spaces, the uses of these algorithms allows for a comparison between paths done using different terms. The objective is that the pathing done utilizing latent space distances can achieve a better result than using real space distances. This would mean that the model was able to learn more complex relationships inside the geometry of the original high dimensional space, which leads to a more smooth space.

It is important to note that for Algorithm 3.1, an extra condition was added, where the loop can only search points belonging to the same object as the starting and ending points. This is done, because when using the entire latent space, the pathing loop has to test every point of the N avaiable, leading to a very high run time. Since Algorithm 3.2 was designed to find the shortest possible path, it can be used with the entire latent space.

The smoothness of a path is given by the accumulative sum of the distances between points that belong to a path, as described in Equation (3.2). This means that paths with a smaller value can be considered smoother, since to achieve the grasps required as steps, the hand suffered less changes in its actuators. This idea was designed with the goal of reducing unnecessary movements between grasps

4.4 Implementing Riemannian Metrics

Introducing the Riemannian metrics into the model depended on what the planning required. Due to how these metrics change the entire model, it is important to know what is wanted from the model being trained. To achieve this, three metrics were planned.

1. Encouraging points to be close to their respective object type cluster center while maximize distances between cluster centers.

2. Encouraging points from one object to be closer to points of the same object and maximize distances between points of different objects.

3. Encouraging points to keep their neighboring relationships from the original high dimensional space.

As mentioned in Chapter 3, the distances obtained in the data preparation section are utilized to create three different G matrixes. Each of these matrixes are used for different purposes and each one has their own specific formulation.

4.4.1 Metric 1

The first metric planned was an attempt to make the latent space readable, in a sense that if the latent space should be able to be representable in a graph. This means that data points in the latent space would have a clear distinction based on their conditional variable.

To achieve this, the G matrix for this metric requires both the distance of points from one object type to their respective object type cluster center point and the distance of the multiple cluster center points to each other.

As mentioned in Section 3.5, the G matrix created is a square matrix, $N \times N$, where N is equal to the latent space dimension.

The diagonal entries of this matrix are determined using the sum of the mean distance of each point of one object type towards their cluster center, scaled using a scaling factor.

The off-diagonal entries of G contain the mean distance between cluster center points of different object types, also scaled using a scaling factor.

These scaling factors allow the control of the force of each individual objective of the metric. This mean that is possible to make the model give different importance to metric objectives. These parameters need to be fine-tuned due to the effect they have on the model training, specifically on the loss function of the training.

4.4.2 Metric 2

For this metric, the goal was to recreate metric 1, but while the previous metric was planned to make points in the latent space separated in clusters with clear distinctions, metric 2 was planned so that while the points are separated by clusters, these same clusters transition to others by using the closest points that belong to different types. This leads to a separation that is not as clear as before, but makes the transition between clusters smoother.

To achieve this, the G matrix was designed as the one in metric 1, but with a difference in the off-diagonal terms.

The diagonal terms in this matrix are the sum of the mean distances of each points to their specific cluster center point.

The off diagonal terms are obtained by calculating the euclidean distance between closest points pf different object types.

Both terms are then scaled using specific scaling factors.

This metric was planned after the implementation of metric 1, due to how results using the previous metric behaved, where the approach of having clear distinctions between object types in the latent space leads to paths with a worse result in regards to smoothness, specifically when considering a initial and final with a high difference in angle joint values.

4.4.3 Metric 3

The last metric was designed without the idea of creating of clusters in the latent space that depend on object type. In this approach, the pairwise distances of all points are used. The goal is to encourage points to keep their relationship from the real space into the latent space, as in, closest points stay the closest and farthest points stay the farthest.

The G matrix diagonal terms are the average distance of the each point towards their closest point in the high dimensional space.

The off diagonal terms are given by the two points with the highest distance between distance each other of any possible combination of two points in the high dimensional space.

This metric was designed so that the model being trained attempts to group grasps depending on their joint angle dimensions, ignoring object types. This helps with regrasping tasks when the task involves grasping and regrasping different objects with different sizes and/or shapes.

4.5 Loss function with Metrics

Adding the Riemannian metrics to the model requires adapting the loss function to take these into consideration. Since the loss function is used utilizing latent space data, it is required to have an argument that can create a link between the real space distances and latent space distances. This is where the G matrix comes into the midst.

Utilizing the G matrix, it is possible to transform the latent space distances into real space distances, and then use this for the Loss function of the model. With this transformation, the loss function of the model can take into account real space pairwise distances between points, so that training the model attempts to keep the same geometry in both spaces.

The new loss function, entitled as Riemannian loss takes the form:

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x,c)}\left[\log p_\theta\left(x|c,z\right)\right] - D_{KL}\left(q_\phi\left(z|x,c\right)\|p(z)\right) + D_z \tag{4.2}$$

Where D_z is the loss given by comparing the high dimensional space pairwise distances, given in the high dimensional space, and the geodesic pairwise distance, given in the low dimensional space.

As an added metric, it was also included the distance of data points towards the center of the high dimensional space, and the geodesic distance of the latent space data points towards their center. This serves as an added measure so that the latent space keeps the real space geometry, even when utilizing the metrics that use clusters.

This leads to the final formula of the Loss function used for training the model, with the added weights of each term.

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x,c)}\left[\log p_\theta\left(x|c,z\right)\right] - w_{KL} \times D_{KL}\left(q_\phi\left(z|x,c\right)\|p(z)\right) + w_{D_z} \times D_z + w_{C_z} \times C_z \tag{4.3}$$

These weights control the importance of each term towards the final loss result, and makes it possible to fine tune the model towards different objectives.

5

Results of Implementation

Contents

This chapter focuses on the results obtained while experimenting the different approaches referred before. Starting with testing the original model shown in [2] for the purpose of possessing a reference value, to the different testing using the pathing mechanisms implemented and the different metrics utilized. The results presented in this chapter refer to experiments using both the iCub and the Shadow robot hand. As a metric, the Reconstruction error of the model trained, given by the MSE error between the original data point and the recreated data point is utilized to grasp how close the model is able to reconstruct the original dataset without much loss of information. This error is not enough to see if the goals explained in Chapter 1 are achieved, so as an added metric the idea of Smoothness, as depicted in Chapter 3. The smoothness values presented in the following graphs is given by calculating the difference in joint angles between the grasps that belong to a path taken by the model from a starting grasps to an ending grasp. As the models are tested in different possible pairs of grasps, the accumulated smoothness values of the paths created is shown. A smaller amount means a smoother path, which means that to perform the regrasping task asked, the hand used suffered a smaller change in joint angle movement, which is what is wanted.

These pairs of starting and ending points in the datasets are carefully selected, so that both grasps in a pair belong to the same object type. As this is to be used as a starting point for a future study, it was decided that for these tests, regrasps should be focused on the same object.

As a reminder, the metrics designed are as follow:

1. Encouraging clustering of points of same object and maximize distance between the cluster centers.

2. Encouraging points from one object to be closer to points of the same object and maximize distances between points of different objects.

3. Encouraging points to keep their neighboring relationships from the original high dimensional space, disregarding object type.

The simulations were done by using latent dimensions between the values of $[2; 9]$, in a way to understand how the models behave when the latent dimension size gets close to the original dimension size.

5.1 Reconstruction Error

Starting with the reconstruction error between the input dataset fed to the encoder and the decoded dataset, given by the output of decoder, the results can be seen in Figure 5.1, with the results for the iCub hand on the left and the results for the Shadow hand on the right.

Figure 5.1: Reconstruction Error on iCub and Shadow Robot hands

From the graphs provided, the simulations done in the iCub hand show that both metric 2 and 3 provide a better value of the reconstruction error compared to the original CVAE model for lower dimensions, which shows a that the new models are able to output a closer resemblance of the original input data. For metric 1, due to the nature of the what the metric is designed to do, it creates a poorer result of the reconstruction error than the original model for higher latent dimension values, but for low values it is similar. This means that, although the latent space is forced to separate points to create the clusters as desired, the model can still use the information given to recreate the dataset.

For the Shadow hand, besides the model using metric 1, all 3 attempts show close values of the reconstruction error. For metric 1, the creation of the clusters creates an issue in the decoding process, which leads to the higher value of error.

5.2 Smoothness Testing

As mentioned at the start of this chapter, although the Reconstruction error is useful to know if the trained model is able to correctly recreated the dataset given to it, so that when using that same model there is a certainty that it can correctly decode a given grasp, it is not enough as a result. For a better understanding if the metrics are achieving the wanted results, pairs of grasps of the same object type are used, one as a starting point and the other as the ending point, to perform paths in the latent space. As mentioned before, to evaluate these paths the notion of Accumulated Smoothness is used, which is the accumulated value of the differences in joint angles in a path, with the amount of paths being decided by the amount of pairs tested.

As a way to represent the results better, these will be presented in two ways, one where for each model, all the values of accumulated smoothness of the different mechanisms are shown, and one where for each pathing mechanism the values obtained in each model are shown.

The algorithms used are:

- Real Neighbors: Use of the Naive Neighbors algorithmn, Algorithm 3.1, utilizing neighboring points in the real space. This is achieved by transversing the latent space and calculating distances between decoded grasps.
- Latent Neighbors: Use of the Naive Neighbors algorithmn, Algorithm 3.1, utilizing neighboring points in the latent space. This is achieved by transversing the latent space and calculating distances between encoded grasps (latent space points).
- A*: Use of the A* algorithm, Algorithm 3.2, which leads to shortest possible using known data points in the dataset.
- Interpolation: Use of the Interpolation algorithm, Algorithm 3.3, which creates new grasps using a interpolation method between the starting and ending grasps.

It should be noted that due to the smoothness of a path being calculated using the Euclidean distance between grasps, when increasing the dimensions of a space, even if the encoded and decoded grasps used for distance calculations originate from the same original data point, the euclidean calculation might increase due to the space being more sparse. This phenomenon happens mostly when utilizing the Naive Neighbors algorithms.

5.2.1 Accumulated Smoothness in a Model

In this subsection, the results of the different pathing algorithms for each model are given. The goal is to achieve the smallest value of accumulated smoothness, which coincides to least amount of energy (difference in values of joint angles) required to perform a regrasping task.

The results will be shown in order from simulations using the original model, followed by the results for metric 1, 2 and 3, in this order.

The graphics presented show the accumulated smoothness for the different pathing mechanisms utilized.

As it is possible to see in Figure 5.2, the paths utilizing Algorithm 3.1, either with the neighboring relationships given from real space or with these relationships given from latent space, the accumulated smoothness is similar in values for the iCub hand (left side graphic). For the Shadow hand, depending on the dimensions present in the latent space, the smoothest path can be obtained using either real space neighbors or latent space neighbors. Regarding the paths using Algorithm 3.2 and Algorithm 3.3, as expected these show the best possible value of smoothness compared to the naive approach. It also shows that the interpolation method, which creates new grasps that might not be present in the original dataset, can compete with the algorithm designed to capture the smoothest path with known data points.

The graphics in Figure 5.2 serve as a reference value for the pathing mechanisms to compare with the implemented metrics.

Figure 5.2: Original Model Pathing results on iCub and Shadow Robot hands

The following figure shows the results utilizing metric 1, where the points are grouped in the latent space depending on their object type, and the center of these clusters are encouraged to be as far apart from other centers.

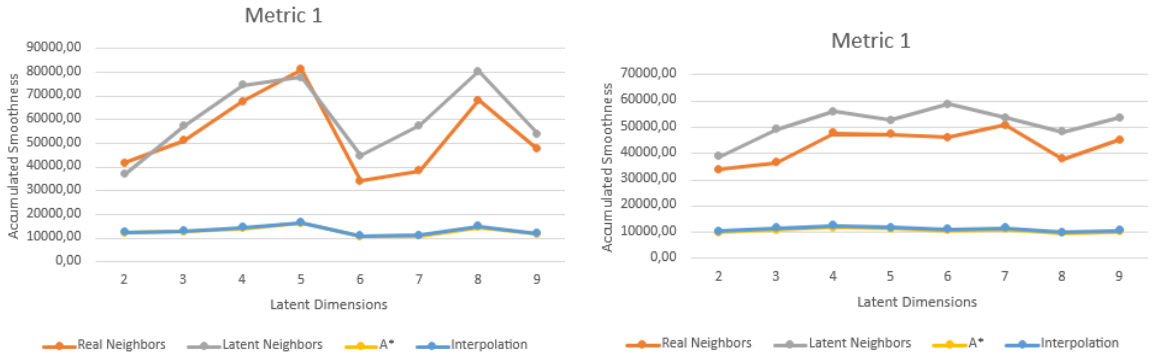

Figure 5.3: Metric 1 Pathing results on iCub and Shadow Robot hands

The first noticeable result is that, once again, both Algorithm 3.2 and Algorithm 3.3 show the best results, with the interpolation method being able to compete with a method that uses known data points while creating new grasps. Between the naive neighbors approach, for both hands utilizing real space neighboring relationships performs better than utilizing latent space neighboring relationships. This is due to the metric deforming the geometry relationships between points in the original dataset to force the clustering procedure. One interesting result is how the naive neighbors approach works with the iCub hand when the latent space has 6 or 7 dimensions. This matches the results from [15], where it is shown that these amounts of dimensions is enough to perform regrasping tasks.

For Metric 2 results, where the metric was designed to group points regarding their object type, and encourage an increase of distance between the two closest points of different object types, the results can be seen in Figure 5.4.

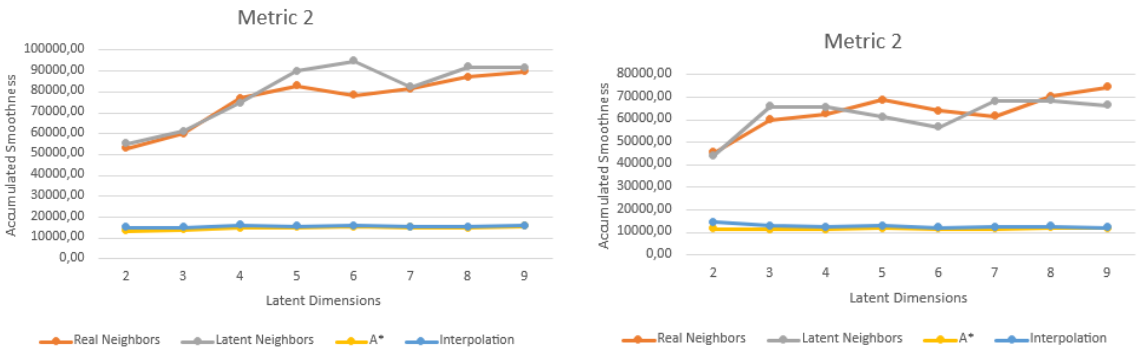

Figure 5.4: Metric 2 Pathing results on iCub and Shadow Robot hands

For this metric, the results show the same behaviour as the previous simulations, with the naive neighbor approach presenting different results to determine which pathing is better depending on how many dimensions the latent space contains. Once again, both the A* and the interpolation approaches present similar results.

The last results of this subsection are for Metric 3, where the idea of grouping grasps depending on their object type is discarded, and instead the metric is designed to make sure the latent space captures the geometric relationships of the real space, and if possible, determines new hidden relationships that might be possible. In Figure 5.5, the results show that the Algorithm 3.1 depends on which hand is used.

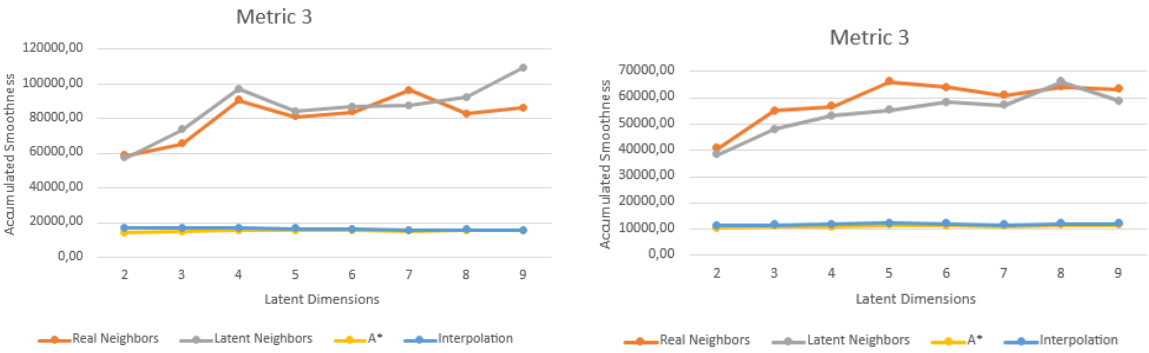

Figure 5.5: Metric 3 Pathing results on iCub and Shadow Robot hands

As it is possible to see, while for the iCub hand utilizing real space neighboring relationships between points leads to smoother paths, when using the Shadow hand, it is actually beneficial to utilize latent space relationships. This means that using the latent space can be better to create regrasping tasks than having to constantly decode grasps to perform these tasks. It also shows that the model was able to recreate these relationships better if the distances are calculated in the latent space than in the

decoded space, which can lead to a simplified use of the hand itself, since it removes the need to having to provide all required inputs of DoF to achieve a regrasping task.

The next subsection shows the same results, but with a different approach. While the previous figure showed all possible pathing mechanisms for 1 model, the next ones will show all possible models for 1 pathing mechanism, so it is easier to compare models relating to pathing smoothness.

5.2.2 Accumulated Smoothness for a Specific Algorithm

To know if the models trained using the new developed metrics perform better or worse than the original CVAE model, each of the models were tested using the different algorithms presented in Chapter 3. These following figures show the results for the different algorithms for each model. The wanted result is that the new metrics can provide a smaller value of accumulated smoothness, preferably for most of the tested latent space possible dimensions sizes. The order used to show these results is the same order as the algorithms were presented.

Starting with the Real Naive Neighbors, where the neighboring relationship is determined via the distance of grasps in the decoded real space, in Figure 5.6.

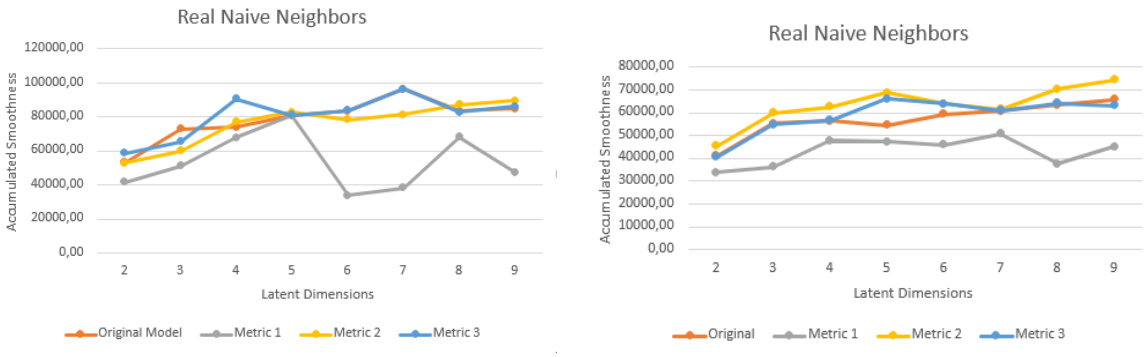

Figure 5.6: Real Naive Neighbors Pathing results on iCub and Shadow Robot hands

For both the iCub and the Shadow robot hands, metric 1 provides the best result in most of the possible dimension sizes for this algorithm. The other three models depend on which used was tested. For the iCub, metric 2 shows the best results following metric 1, while for the Shadow hand the second best model is the original model with no added metrics. It should be noted that although metric 1 shows the lowest amount of accumulated smoothness for most possible latent space dimension sizes, it does show a big variance in the obtained value, while the other three models present a more stable value for different sizes.

Following these results, comes the simulations for Latent Naive Neighbors, where the neighboring relationships are obtained by the distances in the latent space, before the decoding process.

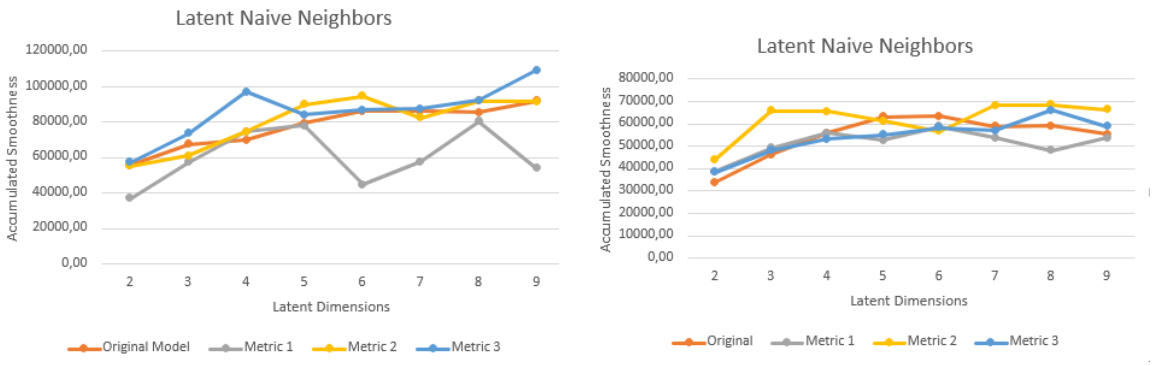

Figure 5.7: Latent Naive Neighbors Pathing results on iCub and Shadow Robot hands

In Figure 5.7, metric 1 presents the best results for the iCub hand, with metric 3 presenting the less smooth paths. Both metric 2 and the original model show the most stable results along the possible sizes of the latent space. For the Shadow hand, for the lowest dimensional latent space, the original model presents the best result. Metric 3 in this case presents a better result than the other 2 metrics, besides metric 1 when the dimensions go over 7.

These results when considering transversing the latent space using neighboring relationships between points present better results when the points in the latent space are grouped depending in their object types. Due to the way this algorithm is used and how the paths are based on the starting and ending points belonging to the same object type, this shows that when performing regrasping tasks where the object being tested does not change, a good approach would be creating the latent space by separating points into clusters and inside these clusters promote the geometric relationships present in real space between grasps of the same object.

With the results shown for the Naive Neighbors approach, the results for the construction of the smoothest path using known data points and the construction of the path by creating new inter grasps are presented. These results are interesting to have to be able to compare if the models perform well when using grasps that are not originally known, so it tests the adaptability of the model to create new grasps after being trained.

In Figure 5.8, the results using Algorithm 3.2 are presented. This approach tries to determine the smoothest possible path, utilizing known grasps. For both hands, the metric 1 approach performs the best in regards to accumulated smoothness, but the results for all of the model are similar and don't fluctuate, even if the amount of latent space dimensions increase, besides metric 1 for the iCub hand.

The interpolation pathing result is used to evaluate if creating new inter grasps is a valid approach when performing these regrasping tasks.

As seen in Figure 5.9 For the iCub hand, the original CVAE shows the most promising results, although the results vary depending on the amount of latent space dimensions. The other 3 metrics

Figure 5.8: A* Pathing results on iCub and Shadow Robot hands

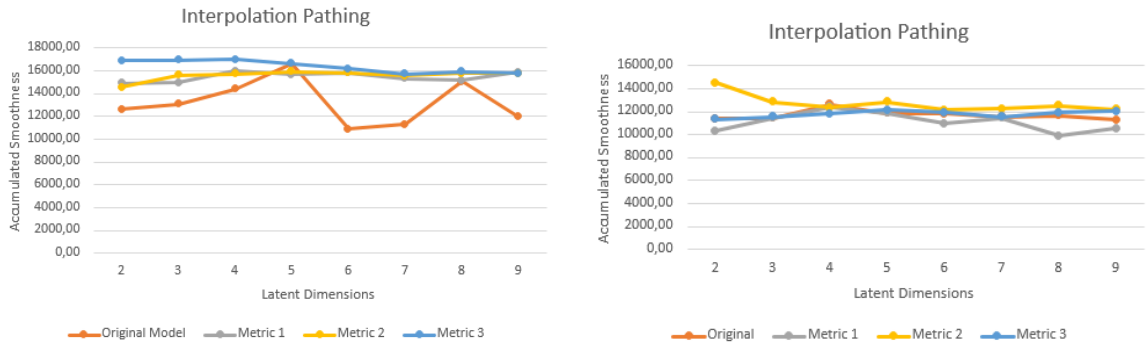

Figure 5.9: Interpolation Pathing results on iCub and Shadow Robot hands

all show a more stable result over the possible values of dimension sizes. For the Shadow hand, all 3 metrics are competitive with the original model, meaning that all 3 metrics behave well when required to create new unknown grasps when performing regrasping tasks.

5.3 KL Weight Effects on the Models

As mentioned in Chapter 3, via the Equation (4.3), there is an added weight to the KL divergence. This divergence is used to make the model approach the distribution of the points of the latent space to the original space. This means, that in theory, increasing this weight forces the model to put more emphasis on trying to maintain how the original space is distributed, but in the latent space. To test the effects of this weight in the model, and knowing that the original space does not have cluster of points depending on object type, the simulation used consist of a model utilizing Metric 3, since this metric only considers the geometric relationships of points in the real space.

In Figure 5.10, the reconstruction error of the model using Metric 3 is given when the KL weight in the loss function is incremented. In the iCub hand, the value of this error increases with the weight

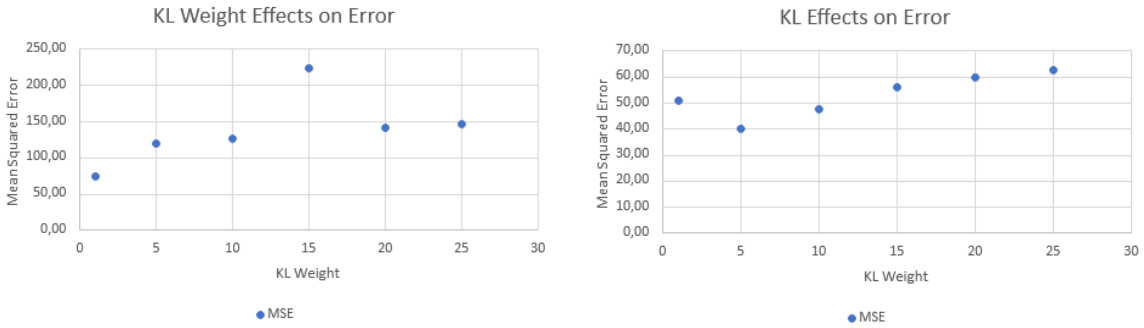

Figure 5.10: Effects of KL weight on Reconstruction Error for iCub and Shadow Robot hands

increase, with a sudden increase for $w = 15$. For the Shadow hand, when increasing the weight from 1 to 5, the value drops, but then slowly increases as the weight increases. This means that a effect of the KL weight on the reconstruction error on the models is already confirmed, making this parameter important to tune. With the Reconstruction error tested, the next step is to test the pathing algorithms with the new weights.

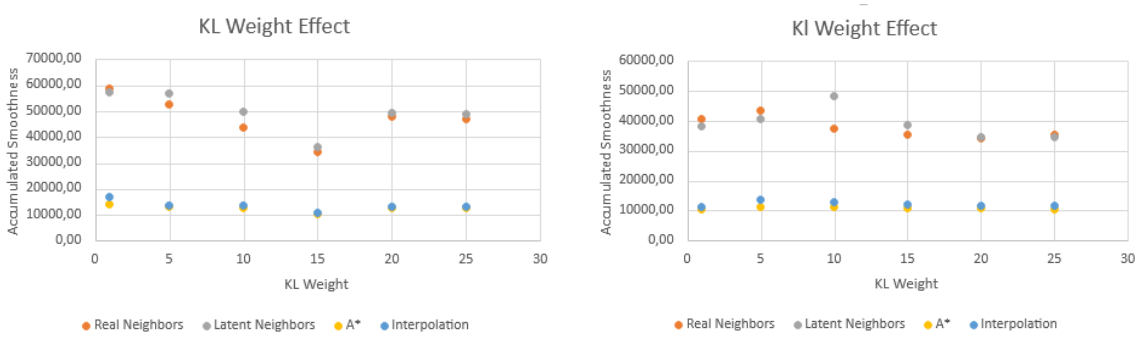

Figure 5.11: Effects of KL weight on Accumulated Smoothness for iCub and Shadow Robot hands

Figure 5.11 shows that the KL weight does have an effect on the results of the model using Metric 3 on the Accumulated Smoothness of a path. This effect is most easily seen in how the paths using the Naive Neighbors approach behave with the weight increase, and how the depending on the weight value, it can change from Real Neighbors being the best approach to Latent Neighbors being the best.

It is also possible to see that for a $w = 15$ using the iCub hand, the interpolation algorithm provides a smoother path for $R^{d_z} = R^2$, than the previous best result in Figure 5.9, although with not a significant difference. For the Shadow hand, the weight does not have a noticeable effect on the accumulated smoothness on the different models, comparing with the previous attempts.

5.4 Simulations in Isaac Gym Environment

The previous results were obtained by the use of calculations. They serve as a way to give a value to smoothness, but it does not give a way to see if the determined path can handle a regrasping task correctly. To confirm that a model can provide a good result without sacrificing utility, the NVIDIA Isaac Gym simulation environment was used, with the Shadow robot hand. This environment provides a visual perspective if a task can be correctly done or not.

To determine if a task was successful or not, using the same pairs that were used for the previous calculations, the paths of each model and metric are recreated using the Isaac Gym environment, for the respective object used in a task. If the object does not leave contact with the simulated hand and is successfully grasped during the entire process, than it is considered a successful regrasp. If, at any moment, the object stops having contact with the hand, then the regrasping task has failed.

The following pictures contain series of the interpolation method, with 9 steps, for all models using a small cylinder as the reference object.

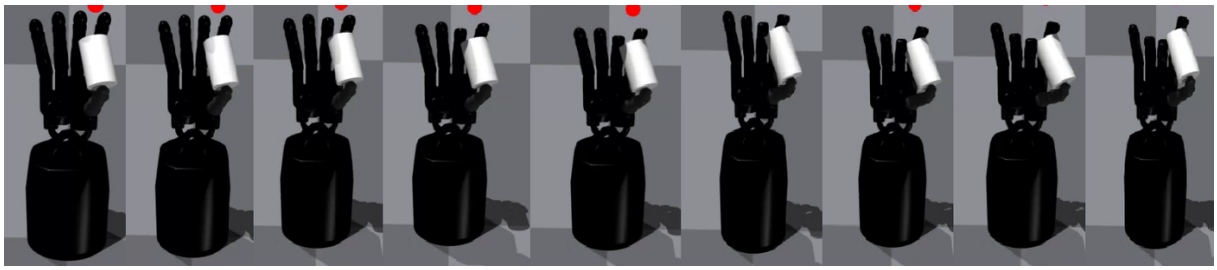

Figure 5.12: Isaac Gym Simulation of Original Model

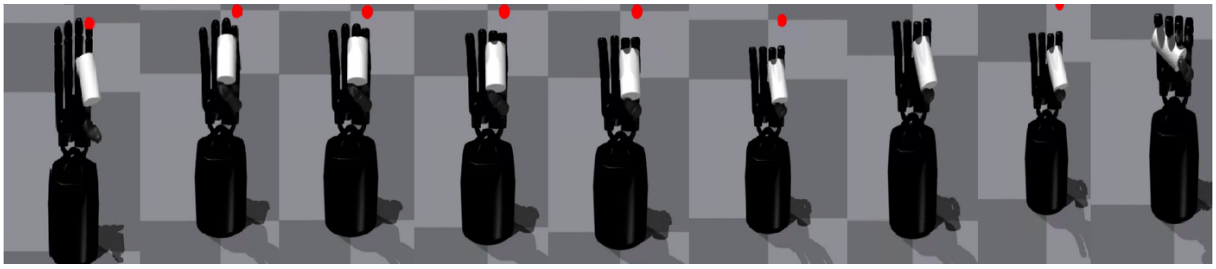

Figure 5.13: Isaac Gym Simulation of Metric 1 Model

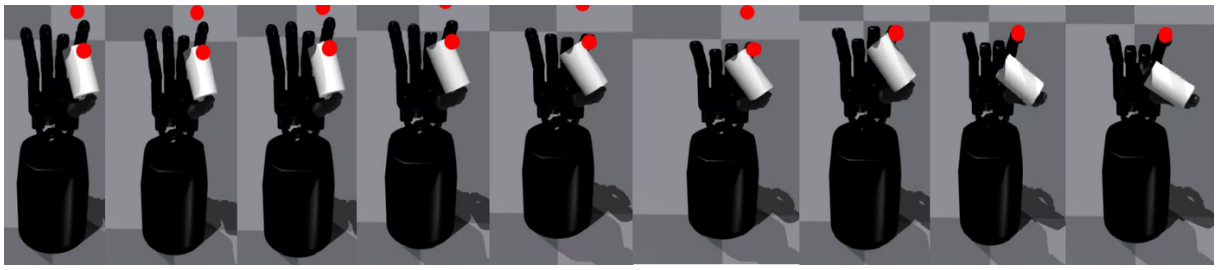

Figure 5.14: Isaac Gym Simulation of Metric 2 Model

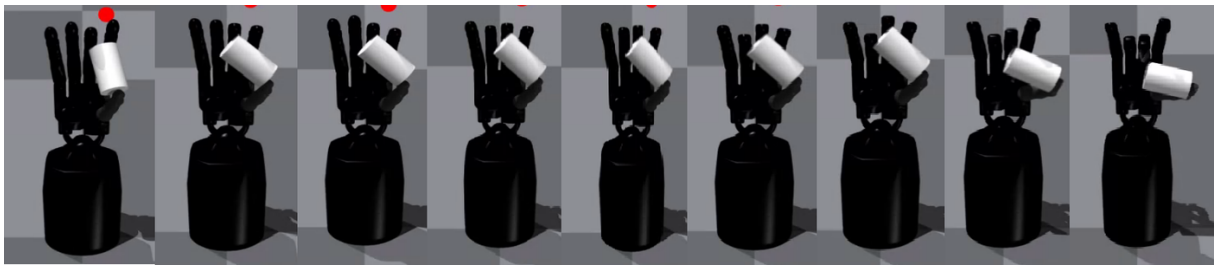

Figure 5.15: Isaac Gym Simulation of Metric 3 Model

As it is possible to see from Figures 5.12 to 5.15, for this specific object, all models can perform a grasp swap without dropping the object, but this does not happen for all possible scenarios. In total, 30 pairs were used and the models were simulated with all. The results are shown in Table 5.1.

Model	Successful Grasp	Failed Grasp
Original	29	1
Metric 1	23	7
Metric 2	28	2
Metric 3	29	1

Table 5.1: Simulation Results with Shadow Robot hand in Isaac Gym Environment with Interpolation algorithm.

These results show that, even though, the model with the implemented Metric 1 had the best value for accumulated smoothness for the Shadow hand in Figure 5.9, it had the most difficulty with regrasping purposes. Specifically, it had difficulty applying a correct movement to the Thumb actuators, leading it to fail most of the regrasps performed with objects with a spherical body.

The original proved to be reliable when performing regrasps with different objects and a good reference point for what results to achieve. Metric 2 failed one more regrasp than the Original model or than the Metric 3 model, specifically one regrasp utilizing a small cylinder, when the object is grasped on the side. Metric 3 had the same performance as the Original model, but they failed on different pairs. While the Original model failed regrasping a small box, Metric 3 failed on a regrasp using a small cylinder on it's side, specifically it couldn't correctly grab the object to start the task.

6

Conclusions

Contents

This chapter focuses on the conclusions reached after analyzing the results obtained after the implementation and the simulations done utilizing both the iCub and the Shadow hand.

6.1 Metric Implementation

For this section, it is imperative to recall that the metrics are structured as follows:

- Promoting the grouping of points belonging to the same object type while also enhancing the separation between the centers of these clusters.
- Encouraging proximity between points that belong to the same object category while simultaneously maximizing the distances between points of different object categories.
- Fostering the preservation of the spatial relationships among points from the original high-dimensional space, irrespective of their object type.

As mentioned in Section 3.5, the metrics that were designed in this work depend on the goals wanted to be achieved. Depending on the certain goals wanted for the model used for dimensionality reduction, one can adapt the metrics to perform better or worse for the tasks planned after implementation. The regrasping tasks in this work were done considering that there was no change on the object being grasped, so this makes it so that, in theory, the best metric to be used would be one that makes it so that the latent space representation of the initial dataset is separated in clusters created by separating objects of the same object type. As seen in Chapter 5, this is what happens, where Metric 1, shows the smoothest creation of paths for regrasping tasks. It is important to note, that since this work deals with the training of models, the fine tuning of all the parameters used is important to the point that changing one of these parameters can lead to different results, making 1 metric better than another.

The interpolation algorithm showing close results to the algorithm specifically implemented for smoothest path discovery is a result that is taken positively, since it shows that the models can create new unknown grasps to perform different regrasping tasks, with these new grasps having a good representation when decoded by the model.

6.2 Kullback-Leibler Weight

Concerning the utilization of KL divergence within the loss function, aimed at preserving the distribution of points from the original real space in the latent space, our experiments highlight the impact of the weight assigned to this parameter. This weight plays a pivotal role in shaping the performance of various metrics, particularly those engineered to prioritize the accurate capture of geometric relationships among points in the original real space over clustering points in the latent space.

While the latent space distribution might initially appear less critical in our work, given that regrasping tasks predominantly involve objects of the same type, it becomes apparent that the value of this parameter becomes increasingly significant when transitioning between different objects is the primary objective. It significantly influences the representation of the latent space and enhances the model's capacity to unveil concealed geometric relationships among data points.

6.3 Simulations on Isaac Gym Environment

The simulations conducted in the NVIDIA Isaac Gym environment were valuable in assessing the practical applicability of our previous findings. They revealed that while Metric 1 initially appeared to be the most promising option, it exhibited shortcomings when applied to real regrasping tasks.

Conversely, Metric 3 demonstrated comparable performance to the original model while delivering superior results in terms of accumulated smoothness. This suggests that by choosing the right metric and implementing it correctly, we can achieve a smoother latent space. This refined latent space, in turn, can effectively control dexterous robotic hands.

6.4 Future Work

This thesis basis itself on an already designed model, given by [2], and in it are used new methods to try to create a smoother latent space representation that is able to recreate the same tasks while suffering less changes in the joint angles of the robot hands used. All of the simulations done in this work utilize the condition that inside a regrasping task, the object being used cannot change. While this work was all done in simulations via software applications and simulation environments, the next step would be attempting the same type of tasks utilizing the real iCub and/or Shadow robot hands to see if the results obtained virtually can be obtained in a real experiment.

Another feature of a future work using the steps taken here should focus on trying to use the same metrics and adapt them or even design new ones that can be used for regrasping tasks with the same efficiency for different dexterous robot hands.

While in-hand regrasping tasks are performed without changing the object the robot hand is currently grasping, in the future it could be possible that a wanted task of a robot hand would involve grabbing two different objects with a continuous motion. Knowing how to create a smoother latent space distribution can be an entry step for the possibility of generating an efficient way to achieve this.

Bibliography

[1] J. Starke, C. Eichmann, S. Ottenhaus, and T. Asfour, "Synergy-based, data-driven generation of object-specific grasps for anthropomorphic hands," vol. 2018-November, 2019.

[2] D. Dimou, J. Santos-Victor, and P. Moreno, "Learning conditional postural synergies for dexterous hands: A generative approach based on variational auto-encoders and conditioned on object size and category," vol. 2021-May, 2021.

[3] M. Santello, G. Baud-Bovy, and H. Jörntell, "Neural bases of hand synergies," *Frontiers in Computational Neuroscience*, 2013.

[4] M. Santello, M. Flanders, and J. F. Soechting, "Postural hand synergies for tool use," *Journal of Neuroscience*, vol. 18, 1998.

[5] M. Santello, M. Bianchi, M. Gabiccini, E. Ricciardi, G. Salvietti, D. Prattichizzo, M. Ernst, A. Moscatelli, H. Jörntell, A. M. Kappers, K. Kyriakopoulos, A. Albu-Schäffer, C. Castellini, and A. Bicchi, "Hand synergies: Integration of robotics and neuroscience for understanding the control of biological and artificial hands," 2016.

[6] J. Lin, Y. Wu, and T. S. Huang, "Modeling the constraints of human hand motion," 2000.

[7] E. Todorov and Z. Ghahramani, "Analysis of the synergies underlying complex hand manipulation," vol. 26 VI, 2004.

[8] G. Salvietti, "Replicating human hand synergies onto robotic hands: A review on software and hardware strategies," *Frontiers in Neurorobotics*, vol. 12, 2018.

[9] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," 2004.

[10] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, 1901.

[11] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” *AIChE Journal*, vol. 37, 1991.

[12] J. Romero, T. Feix, C. H. Ek, H. Kjellström, and D. Kragic, “Extracting postural synergies for robotic grasping,” *IEEE Transactions on Robotics*, vol. 29, 2013.

[13] M. Ciocarlie, C. Goldfeder, and P. Allen, “Dimensionality reduction for hand-independent dexterous robotic grasping,” 2007.

[14] ——, “Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem,” 2007.

[15] A. Bernardino, M. Henriques, N. Hendrich, and J. Zhang, “Precision grasp synergies for dexterous robotic hands,” 2013.

[16] O. C. Jenkins, “2d subspaces for sparse control of high-dof robots,” 2006.

[17] a Tsoli and O. Jenkins, “2d subspaces for user-driven robot grasping,” *Robotics, Science and Systems Conference: Workshop on Robot Manipulation*, vol. 5, 2007.

[18] K. Xu, H. Liu, Y. Du, and X. Zhu, “A comparative study for postural synergy synthesis using linear and nonlinear methods,” *International Journal of Humanoid Robotics*, vol. 13, 2016.

[19] N. D. Lawrence and J. Quiñonero-Candela, “Local distance preservation in the gp-lvm through back constraints,” vol. 148, 2006.

[20] J. Starke, C. Eichmann, S. Ottenhaus, and T. Asfour, “Human-inspired representation of object-specific grasps for anthropomorphic hands,” *International Journal of Humanoid Robotics*, vol. 17, 2020.

[21] K. Sohn, X. Yan, and H. Lee, “Learning structured output representation using deep conditional generative models,” vol. 2015-January, 2015.

[22] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” 2014.

[23] M. P. do Carmo, *Riemannian Geometry*, 1992.

[24] G. Arvanitidis, L. K. Hansen, and S. Hauberg, “A locally adaptive normal distribution,” 2016.

[25] ——, “Latent space oddity: On the curvature of deep generative models,” 2018.

[26] N. Chen, M. Karl, and P. V. D. Smagt, “Dynamic movement primitives in latent space of time-dependent variational autoencoders,” 2016.

[27] G. Arvanitidis, S. Hauberg, and B. Schölkopf, “Geometrically enriched latent spaces,” vol. 130, 2021.

[28] C. Y. Lee, “An algorithm for path connections and its applications,” *IRE Transactions on Electronic Computers*, vol. EC-10, 1961.

[29] M. L. Fredman and R. E. Tarjan, “Fibonacci heaps and their uses in improved network optimization algorithms,” *Journal of the ACM (JACM)*, vol. 34, 1987.

[30] G. Baud-Bovy and J. F. Soechting, “Two virtual fingers in the control of the tripod grasp,” *Journal of Neurophysiology*, vol. 86, 2001.

[31] V. M. Zatsiorsky, F. Gao, and M. L. Latash, “Prehension synergies: Effects of object geometry and prescribed torques,” *Experimental Brain Research*, vol. 148, 2003.

[32] E. J. Weiss and M. Flanders, “Muscular and postural synergies of the human hand,” *Journal of Neurophysiology*, vol. 92, pp. 523–535, 7 2004.

[33] C. Y. Brown and H. H. Asada, “Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis,” 2007.

[34] C. Bishop, “Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn,” *Springer, New York*, 2007.

[35] M. T. Ciocarlie and P. K. Allen, “Hand posture subspaces for dexterous robotic grasping,” *International Journal of Robotics Research*, vol. 28, pp. 851–867, 7 2009.

[36] T. Feix, R. Pawlik, H.-B. Schmiedmayer, J. Romero, and D. Kragi, “A comprehensive grasp taxonomy,” 2009.

[37] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *Journal of Machine Learning Research*, vol. 10, 2009.

[38] S. Karaman and E. Frazzoli, “Incremental sampling-based algorithms for optimal motion planning,” vol. 6. MIT Press Journals, 2011, pp. 267–274.

[39] F. Ficuciello, G. Palli, C. Melchiorri, and B. Siciliano, *Postural synergies and neural network for autonomous grasping: A tool for dextrous prosthetic and robotic hands*. Springer International Publishing, 2013, vol. 1, pp. 467–480.

[40] N. Jarrassé, A. T. Ribeiro, A. Sahbani, W. Bachta, and A. Roby-Brami, “Analysis of hand synergies in healthy subjects during bimanual manipulation of various objects,” *Journal of NeuroEngineering and Rehabilitation*, vol. 11, 2014.

[41] A. Tosi, S. Hauberg, A. Vellido, and N. D. Lawrence, “Metrics for probabilistic geometries,” 2014.

[42] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio, “A recurrent latent variable model for sequential data,” vol. 2015-January, 2015.

[43] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” 2015.

[44] H. V. Hoof, N. Chen, M. Karl, P. V. D. Smagt, and J. Peters, “Stable reinforcement learning with autoencoders for tactile and visual data,” vol. 2016-November, 2016.

[45] M. Heidari and M. H. Moattar, “Discriminative geodesic gaussian process latent variable model for structure preserving dimension reduction in clustering and classification problems,” *Neural Computing and Applications*, vol. 31, 2019.

[46] G. Arvanitidis, S. Hauberg, P. Hennig, and M. Schober, “Fast and robust shortest paths on manifolds learned from data,” 2020.

Object Types and Grasp Types

Object	Size
Ball	Big
Ball	Medium
Ball	Small
Cylinder Top	Big
Cylinder Side	Big
Cylinder Top	Medium
Cylinder Side	Medium
Cylinder Top	Small
Cylinder Side	Small
Pen	
Cube	Small
Blue Box	Large Side
Blue Box	Small Side
Orange Box	Large Side
Orange Box	Small Side
Red Box	Large Side
Red Box	Small Side
Red Box	Medium Side
Yellow Box	Small Side
Yellow Box	Large Side

Table A.1: Object type and Size

Grasps
Tripod
Palmar Pinch
Lateral
Writing Tripod
Parallel Extension
Adduction Grip
Tip Pinch
Lateral Tripod

Table A.2: Grasp Types