Exploration of Latent Spaces of 3D Shapes via Isomap and Inverse Mapping

by

Raghad Rowaida, **B.Sc.**

A thesis submitted to the
Faculty of Graduate and Postdoctoral Affairs
in partial fulfillment of the requirements for the degree of

Master of Computer Science

Department of Computer Science
Carleton University
Ottawa, Ontario
December, 2021

The undersigned hereby recommends to the
Faculty of Graduate and Postdoctoral Affairs
acceptance of the thesis

Exploration of Latent Spaces of 3D Shapes via Isomap and Inverse Mapping

submitted by **Raghad Rowaida**, **B.Sc.**

in partial fulfillment of the requirements for the degree of

Master of Computer Science

Professor Oliver van Kaick, Thesis Supervisor

Professor David Mould, School of Computer Science

Professor WonSook Lee,
School of Electrical Engineering and Computer Science

Professor Evangelos Kranakis, Chair,
School of Computer Science

Department of Computer Science
Carleton University
December, 2021

Abstract

In this thesis, we introduce a method for exploring a latent space of 3D shapes learned by a deep neural network. The main idea of our method is to enable the exploration of the latent space through the navigation of a 2D embedding. The method is based on a combination of Isomap dimensionality reduction and an inverse mapping function. Specifically, given a dataset of 3D shapes, we first train an autoencoder neural network to learn a latent representation for the dataset. Then, we reduce the dimensionality of the latent space to two dimensions with the Isomap method. The 2D embedding of the latent space allows a user to easily navigate through the embedding and sample new points. Our method then translates the sampled points back into latent vectors with an inverse mapping function, while the latent vectors are decoded by the neural network into 3D shapes that can be inspected by the user. The inverse mapping is made possible by posing it as a radial basis function (RBF) scattered interpolation problem. We demonstrate with qualitative experiments that our exploration method has advantages compared to alternative approaches such as interpolation and principal component analysis. We also show with quantitative experiments that our method enables a meaningful exploration of the latent space.

Acknowledgments

First, and foremost, I would like to express my deepest gratitude to Dr. Oliver van Kaick, my research supervisor, for giving me the opportunity and immense support to make this research a reality. His encouragement, guidance and patience truly helped me to shape this research into what it is today. I would also like to thank Dr. David Mould, Dr. WonSook Lee for their thorough review of my work. Their valuable suggestions and useful critiques towards the thesis improved the quality of the work. I would like to thank GIGL lab and all the members for their encouragement and suggestions to prepare myself for the defense. I am also grateful to the School of Computer Science, Carleton University for accepting me as a research student and facilitating me with financial assistance to complete my degree. At last, I would like to thank my parents, sister, family members and my beloved husband for their unconditional love and support that encouraged me to complete my research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Creating 3D shapes is an important task in many areas related to computer graphics, such as asset design in game development, virtual world creation, architectural design, and furniture design. Creating a 3D shape in a manual fashion is difficult and time consuming. Given that a moderately-detailed 3D object represented as a triangle mesh is comprised of thousands of vertices and triangles, to create and manipulate such a mesh, the user needs to navigate the 3D space and make meaningful changes to the shape, which involves precisely selecting and then modifying the intended mesh elements from thousands of vertices and faces. This is why methods for facilitating shape creation are desirable, especially to allow the user to create a shape at a higher level than the manipulation of mesh elements.

Shape creation at a high level. Several methods have been proposed to enable shape creation at a high level. Note that, in the literature, these methods can be referred to as shape modeling, creation, or synthesis methods. One of the early techniques for shape modification was Free-Form Deformation (FFD) [38]. FFD allows the user to edit shapes at a high level, where the user selects the overall shape or a part of it for modification and fits a cage structure to the geometry, which will guide a deformation of the geometry. After moving one or more control points on the cage, the change is propagated to the shape in the form of a deformation. Although this method provides the possibility of deforming the shape at a high level, it has several limitations. The deformation is completely dependent on the user guidance, since the system has no awareness of shape semantics. Thus, the movement of the control points has to be precise, which is not easy for novice users. Given these limitations, subsequent work has focused on ways to include shape semantics in the shape

editing/creation process.

Parametric shape editing methods enable a user to synthesize shapes at a high level through the selection of values for a set of parameters. One class of methods for parametric editing learns a statistical shape model from training data with techniques such as Principal Component Analysis (PCA) [4,5]. The parameters of the statistical model can then be selected to synthesize new shapes of human bodies or faces. One limitation of these approaches is the dependence on training meshes with dense point-to-point correspondences, which is difficult to prepare. In more recent methods, shape segments are annotated with parameter information which helps in editing specific parts of the shapes in a meaningful way without the use of control points. The user can stretch, compress or deform a particular segment of a shape based on the modification of parameters [30]. In addition, new shapes can be synthesized by combining different segments from a database of parametric shapes of a specific class. One limitation of these approaches is that they require a precise semantic segmentation or parametric model of the shapes, which are difficult to obtain. For example, the method of Yumer et al. [50] learns the parametric model from manually-defined training data. Some approaches automate the process of learning the parametric model by analyzing the geometry and topological similarity of the shapes [6]. However, this type of method works best on shapes with simple and regular geometric structure.

Deformation methods allow to modify existing shapes, but it is difficult to create entirely new shapes with deformation. On the other hand, parametric models allow the creation of new models, but the user cannot provide direct constraints to the synthesis process; exploration of a set of parameters is required. Thus, sketch-based approaches were introduced to allow a user to freely create a shape by drawing it. The technical problem addressed by these methods is the synthesis of a 3D shape form one or more 2D sketches. Earlier sketch-based approaches converted 2D sketches to 3D by inflating a drawn shape outline based on the silhouette guideline [23]. The fundamental limitation of sketch-based shape modeling is that the outcome depends entirely on the artistic skills of the user. Thus, one advanced sketching method addresses this limitation by combining sketching with procedural modeling [22], where the sketch guides the selection of parameters in the model, enabling the user to synthesize high-quality shapes from relatively simple sketches. However, the sketches still need to convey enough and accurate information in order to guide the synthesis towards a meaningful model.

Shape synthesis based on latent spaces. Recently, shape generation methods based on deep learning have been introduced, which provide more effective ways of creating shapes at a high level [25,28,32]. One class of deep learning synthesis methods involve the idea of learning a latent space, which is advantageous as the latent spaces serve as a simplified shape representation that can be manipulated for synthesis. Although a 3D shape is a high-dimensional representation, it can be represented as a point on a low-dimensional manifold (the latent space) learned by a deep network [43]. A shape in this low-dimensional manifold is typically represented as a point or vector with hundreds of dimensions. The representation of shapes as points in this manifold is more suitable for tasks such as exploration of the dataset and shape synthesis.

Learning of the latent space can be accomplished with the use of an autoencoder network composed of an encoder and decoder [24]. The encoder takes a 3D shape as input, e.g., encoded as a 3D volume, and returns the latent representation of the shape. The decoder takes the latent representation and decodes it back into a 3D shape. The autoencoder is trained with a dataset of shapes in a fully-automatic manner by minimizing the reconstruction error of the shapes when encoding and then decoding the shapes with the autoencoder. Moreover, in contrast with parametric methods, these latent spaces can be learned with minimal data pre-processing, e.g., requiring mainly a consistent alignment of the shapes rather than a dense point-to-point correspondence.

The learned latent representation of the shapes can be thought of as encoding the high level attributes of the shapes into a set of numbers. Meaningful changes in these latent vectors then produce new shapes after decoding. Thus, manipulation of the latent vectors can be used for synthesizing new shapes, e.g., through the interpolation or combination of latent vectors of existing shapes.

However, this synthesis framework also has its limitations. After encoding, the shapes are converted into latent vectors with hundreds of dimensions. This implies that users have to explore a latent space of hundreds of dimensions to synthesize a shape that fits their requirements. It is impractical to manually explore such a large space. A first approach to facilitate the exploration of these latent spaces is to use interpolation of latent vectors, where the user selects two endpoint shapes that are part of the training set, and synthesizes shapes in between the two endpoints [3, 29]. However, the exploration is limited to blending the endpoint shapes.

A second approach to facilitate the exploration of the shape space is to reduce the

dimensionality of the latent space further more. Härkönen et al. [21] use Principal Component Analysis (PCA) to decompose various latent spaces into a small set of principal components that capture the maximum variation in the data. The principal components can then be combined with different weights to synthesize new data. This simplifies the exploration of the latent spaces. However, not all of the possible combinations of components lead to meaningful shapes. More importantly, there is no direct association between principal components and specific shapes or shape attributes, which makes it more difficult to add specific constraints to the synthesis.

Finally, a third category of approaches enables the exploration of latent spaces with the use of natural language. For example, Jahan et al. [24] introduce a method to enable the exploration of latent spaces through keywords, while Chen et al. [13] introduce a method to directly synthesize a shape from a description in natural language. These methods require additional pre-processing for associating keywords to shapes or establishing a connection between shapes and sentences in natural language.

In contrast to the methods discussed above, the goal of our work is to introduce a method that allows a user to freely explore the latent space and synthesize new shapes, while also combining the advantages of dimensionality reduction and interpolation approaches. Our goals are: (1) To reduce the large latent space into an embedding with a manageable number of dimensions that preserves the neighborhood relations among latent vectors, so that the exploration of different regions of the embedding reflects in meaningful attribute changes in the synthesized shapes. (2) The latent space should be able to be explored freely and easily. (3) There should not be much manual pre-processing involved in the method. We describe our proposed method as follows.

1.1 Overview of our method

Our method that provides an effective manner to explore the latent space is composed of the following steps: (a) We apply the Isomap method for reducing the dimensionality of the learned latent space into a 2D embedding. (b) The user freely explores different regions of the embedding by clicking on specific points in the embedding. (c) We apply an inverse mapping on each selected point to map it back to the original latent space and synthesize a shape from the latent vector. Our method is illustrated in Figure 1.1.

Specifically, given a dataset of 3D shapes, we first train an autoencoder with the dataset. Then, all the shapes in the dataset are passed through the encoder of the trained autoencoder (Stage 1 in Figure 1.1) which provides us with the latent vector of each shape. These latent vectors have 128 dimensions. The Isomap method is then responsible for reducing the dimensionality of the latent vectors from 128 dimensions to a more manageable 2D representation. The red dots in Figure 1.1 show the points in the 2D embedding corresponding to shapes in the training set. The inverse mapping (Stage 4) works in the opposite direction, mapping any given point in the 2D embedding back to the 128-dimensional space via a scattered data interpolation. The resulting latent vectors can then be decoded back into 3D shapes.

For exploration of the latent space, the user can easily sample new candidate shapes by clicking on the 2D embedding, such as the blue dots shown in Stage 3 of Figure 1.1. These points are then fed to Stage 4 (inverse mapping) and 5 (decoding) to obtain the corresponding shapes.

Note that, our proposed two-stage approach, composed of mapping to a 128-dimensional latent space followed by mapping to a 2D embedding, is necessary since autoencoders cannot learn a meaningful encoding by mapping directly to two dimensions. Typically a minimum of hundred dimensions are needed [16].

With our method, the user is able to freely explore the 2D embedding and correspondingly the latent space. In consequence, the user is still able to perform interpolation between existing latent vectors of shapes, but is also able to interpolate between any two points in the embedding. The exploration process is easy as the user mainly needs to click on points in a 2D plot and at the same time the exploration can be controlled with fine precision based on the resolution of the plot. In comparison to the PCA-based method [21], the user has only two dimensions that can be explored but sees the existing shapes on the plot as reference for the exploration, rather than combining multiple dimensions without any reference.

Finally, an obvious benefit of our method in comparison to methods based on natural language [13, 24] is that it needs no manual labeling of the data to enable the exploration. In comparison to sketching approaches [18], artistic skills are not required from the user, while in comparison to classic parametric models [8], the requirements on data pre-processing are not as stringent. Nevertheless, if the user has additional information about some of the training shapes in the form of textual labels, we show in Chapter 3 that we can also take this information into consideration

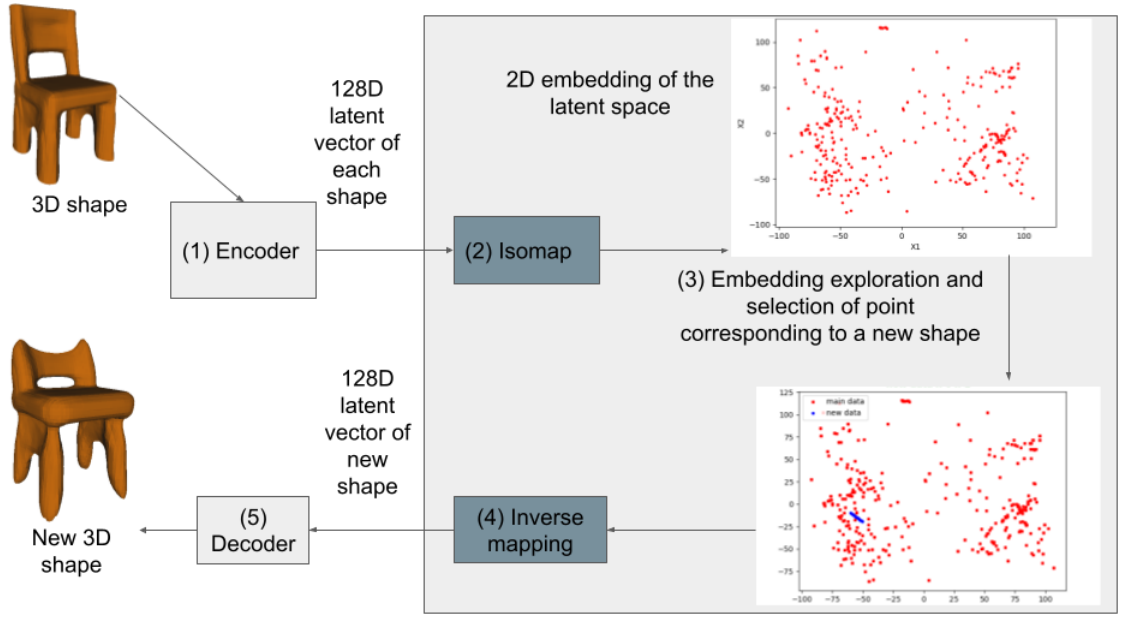

Figure 1.1: Overview of our method. The system takes a set of 3D shapes as input and obtains their corresponding 128D latent vectors form an (1) Encoder network. (2) Isomap reduces the latent vectors to a manageable 2D embedding. (3) Exploration of the embedding allows the user to select new points that correspond to new shapes. (4) The inverse mapping of the selected new points produces new 128D latent vectors. (5) A decoder network generates new 3D shapes from the latent vectors.

to better organize the embedding used for exploration of the latent space.

In this thesis, we first describe the proposed exploration method and then evaluate it with a set of qualitative and quantitative experiments, where we compare to alternative methods and evaluate the different components of our method. The two goals of our evaluation are to demonstrate the advantages of our method over interpolation and PCA, and provide some evidence that the Isomap and inverse mapping allow to synthesize shapes that combine characteristics of nearby shapes in the embedding, generating meaningful shapes. For our evaluation, we focus on the synthesis of manufactured shapes such as furniture, as opposed to natural shapes such as humans, human faces, and animals.

1.2 Contributions

We introduce a method for shape synthesis based on interpolating shapes from an input set. Our method allows non-expert users to explore a set of shapes in search of shapes with desired characteristics. Then, selected shapes can be blended together by simply selecting points on a 2D plot. Since this framework does not allow the creation of entirely new shapes from scratch, it is aimed at users who would like to create variations of existing shapes via blending. Our focus was to making shape editing that do not require expensive preprocessing of input data and additional labeling information to work with. The core of our method is the learning and manipulation of a latent space of 3D shapes through an Isomap embedding and inverse mapping.

Although the different components used by our method have appeared in previous work [31], to the best of our knowledge, our work is the first to introduce a method that combines these components to enable a user to freely navigate a latent space of shapes through exploration of a 2D embedding. Our method has several advantages over the existing methods [21], and allows the user to synthesize smooth shapes with the use of an implicit decoder (Chapter 3).

In addition, a second contribution of our work is that we evaluate the different components of our method in a qualitative and quantitative manner to demonstrate that it enables a meaningful exploration of the latent space, where shapes with similar characteristics are grouped together in the embedding space, and reasonable shapes are synthesized for newly explored regions of the embedding. We tested different methods for the dimensionality reduction step of our method, and selected the Isomap method as it performs the best in conjunction with the inverse mapping function.

1.3 Organization

The remaining chapters of this thesis are organized as follows: Chapter 2 provides a discussion of the existing work and how it relates to our work. Chapter 3 introduces our method, explaining how the different components of the method are integrated together to allow a user to synthesize shapes. Chapter 4 discusses the datasets used and presents results, along with an evaluation of our method. Chapter 5 finalizes the thesis with our conclusions and suggestions for future work.

Chapter 2

Related work

An automated shape creation system should be based on some kind of shape model, possibly involving a high-level understanding of shape characteristics, so that the system can synthesize new shapes with meaningful variations. The system also needs to minimize the user effort by letting the user guide the synthesis in an easy and intuitive manner. Research work in this area has gradually developed different shape creation methods to achieve these goals. When considering 3D shape editing, there are two main classes of shapes that can be considered [49]: (a) fixed topology non-rigid shapes, and (b) rigid shapes with no fixed topology

Different types of techniques are suitable for creating and modifying these two shape classes. Fixed topology non-rigid shapes have an underlying skeleton structure that can be used to guides synthesis and editing methods. Organic shapes such as human figures fall in this category. Editing of this type of shapes typically done by deforming the underlying skeleton with specific constraints such as volume-preserving deformation [49]. On the other hand, the creation and editing of rigid shapes with no fixed topology objects requires a different class methods, since the objects can have a variety topological structures. Man made shapes such as furniture, usually falls into this category. A creation or editing method for these shapes should preserve the internal structural relations between different shape parts. This may require knowledge of the shape semantics in order to modify particular attributes of the shapes [20]. Learning the semantics of shapes is the focus of many deep learning based methods [50].

In the remaining of this chapter, we will discuss editing and creation of man-made shapes in more detail, since it is the set of methods most related to our work. We will also discuss relevant work making use of machine learning for shape creation.

2.1 Traditional shape creation methods

Developing methods for the creation of 3D shapes is an active area of research. One possible manner of creating a 3D shape is to scan an exiting object. However, scanning and reconstruction is a complex problem, and it can only provide models of existing shapes, while the user may want to create new objects. Moreover, shape editing is one approach to create new shapes from existing shapes. However, low-level editing where the user has to manually modify the shape topology and geometry is a laborious process not suitable for large-scale shape creation. Thus, high-level shape editing techniques were introduced, which enable to more easily create shapes from existing ones. One class of high-level editing techniques make use of deformation methods to change the geometry of existing shapes, while another class of methods make modifications to the shapes based on additional knowledge of the shape structure. In this section, we discuss these more traditional shape creation methods in more detail.

Deformation approaches such as free-form deformation [38] and Laplacian mesh deformation [26] allow users to create new shapes by deforming existing shapes. In free-form deformation, the user encloses a cage structure over a shape and deforms the shape by moving control points of the cage. In Laplacian mesh deformation, the user selects certain mesh elements as anchors that should not move and certain elements as a handle that can be moved freely in space. The method then deforms the remaining portion of the mesh to satisfy the deformation constraints imposed by the anchor and handle elements. The type of modification that can be obtained by these methods is limited, as it pertains mainly to the shape geometry, while selecting control points or mesh elements and manipulating them in proper manner requires some experience with these systems.

In contrast, sketch-based shape creation methods are relatively easy to use for non-technical users who have artistic skills. Sketch-based methods create a 3D shape out of one or more 2D shape drawings. One of the earliest sketch-based methods converted a 2D sketch into 3D shape by inflating the region bounded by the sketch outline [23]. Specifically, the method makes some region more inflated and some areas narrower according to a selected threshold on the width of the area. This method also allowed step by step modification on the created 3D object to create more complex shape. This approach facilitates the creation of simple 3D shapes. However, creating more complex shapes is difficult as the sketching steps become more complex for a novice user.

Parametric models learn a shape model from a collection of existing shapes, which then allows a user to create new shapes. These methods are relatively easy to use for novice users. For example, Talton et al. [39] introduced a parametric shape modeling technique that lets the user create a new shape by exploring and modifying a database of existing shapes. The system maintains a database that holds a collection of segments (shape parts) of a class of shapes. The user interacts with the system through an interface that allows the user to specify the attributes of the shapes, and then the system combines selected parts together to form a complete 3D shape. The user can then make small adjustments to the segments using sliders. As a starting point, this method requires a rich database of shapes. The method requires direct user involvement for shape creation, as the system does not automatically combine segments together. This implies that the shape creation is slow.

Subsequent research worked to make the parametric shape creation more autonomous. Chaudhuri et al. [12] introduced a suggestion system that explores the shape part database automatically and suggests appropriate parts that the user can combine to create a new shape. The system analyzes the shape creation in progress and based on the user's creative need and the available database can suggest parts to be incorporated. In this manner, the user does not have to manually search through the database or tweak sliders to get matching shapes.

Alhashim et al [2] introduced a method to interpolate topologically varying shapes from a source shape to a target shape. The method blends the source and the target shape topologically and geometrically, producing a continuous series of in-between new shapes. But this process requires a precise correspondence between the source and the target shape segments.

Bokeloh et al. [6] introduced a method that infers a parametric model for a set of input shapes by analyzing the local shape geometry. The method recursively analyzes each shape to infer patterns present in the shapes' geometry. The method can also infer changes to the shapes with simple algebraic operations. Once the model is learned, the user can modify the shapes by changing parameter values. This is an ideal manner of creating shapes out of a parametric specification. However, the inference is suitable mainly for shapes with simple geometric structures which can be easily segmented in different patches based on geometric similarity.

Chaudhuri et al. [11] created a method to make 3D shape creation easier for novice users. Given a segmented shape database with semantic labels of parts, the method

allows users to describe the desired shape in language form. The system associates user-specified keywords to the shape parts in the database and selects appropriate segments that can be combined in a meaningful way to create a new shape based on the user's description. However, the system is not completely automated. The system uses an interactive process where the user can select parts that are suggested by the system based on the available shape segments.

2.2 Shape representation learning

An essential step in applying machine learning methods to shape analysis is to reduce the high dimensionality of the shapes into a more manageable low-dimensional representation, where the main defining characteristics of shapes are encoded. Such a representation can be computed from a set of shapes in a specific category. The reduced representation captures the main features of shapes and represents them with a small amount of information. Early methods for dimensionality reduction used techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) [35, 48]. However, their capability in extracting shape features is limited. Thus, with the advancement of deep learning, more advanced methods for dimensionality reduction were developed.

The autoencoder (AE) is a deep learning based dimensionality reduction method [48]. The main idea of the AE is to learn a compressed representation for a set of data through minimizing the reconstruction error of a neural network [36, 43]. This is often possible since high dimensional data commonly lies on a low-dimensional manifold, which is referred to as the intrinsic space of the data. Figure 2.1 shows the basic workflow of an autoencoder network. Specifically, an encoder maps the input to a reduced latent representation and the decoder reconstructs the input from the learned latent representation.

Many other methods extended the concept of the autoencoder. We discuss one of these methods here. The generalized autoencoder (GAE) is an extension of the autoencoder [43] which modifies the loss function of the AE to learn from relationships provided during training. It considers data similarities along with a weighting function in the loss function used for training an AE network. Specifically, let us represent the original shape instance as x_i and the latent representation as y_i. The loss function of the GAE network measures the error of reconstructing a set of instances rather than

just a single data point form y_i. The association between instances is provided by a weighted relation function that typically extracts nearest neighbors from the data based on some similarity measure.

Guan et al. [19] applied the GAE to learn a latent space of shapes represented as 3D volumes. In this method, similarities of data points are estimated using the Chamfer distance. As this method enables a more meaningful interpolation of latent vectors, we used it as the encoding network in our work.

On the other hand, the Generative Adversarial Network (GAN) can also be seen as learning a latent representation for a dataset [17]. The work flow of a GAN is illustrated in Figure 2.2. The GAN is composed of a generator and a discriminator network which are trained simultaneously. The generator network tries to generate new shapes from an input noise vector so that the shapes are close to the training shapes. The discriminator network takes both the training data and the generator output and tries to learn to distinguish among them. The measurement of the discriminator network is used as a feedback to both generator network and the discriminator network to fine tune their performance. Based on the feedback, the generator network tries to generate more convincing outcomes and feeds the improved outcomes to the discriminator. This process is repeated until the generator network generates shapes that looks similar to training samples to the discriminator network. With this outcome, the network has learned to create realistic shapes based on the training shape collection. Although in general GANs provide better generalization than AEs, GANs have the disadvantage that the architecture does not include an encoder, which implies that input shapes cannot be mapped to the latent space, which is useful to generate reference latent vectors for interpolation and exploration purposes.

2.3 Shape creation methods using deep networks

In the previous section, we discussed how different dimensionality reduction techniques can reduce shapes into a low-dimensional latent representation for easier analysis and manipulation. In this section, we discuss how this representation can be used to enable shape creation using deep learning methods.

One important question when using deep learning for shape analysis and creation is how to represent a 3D shape for inputting it into a neural network. One of the

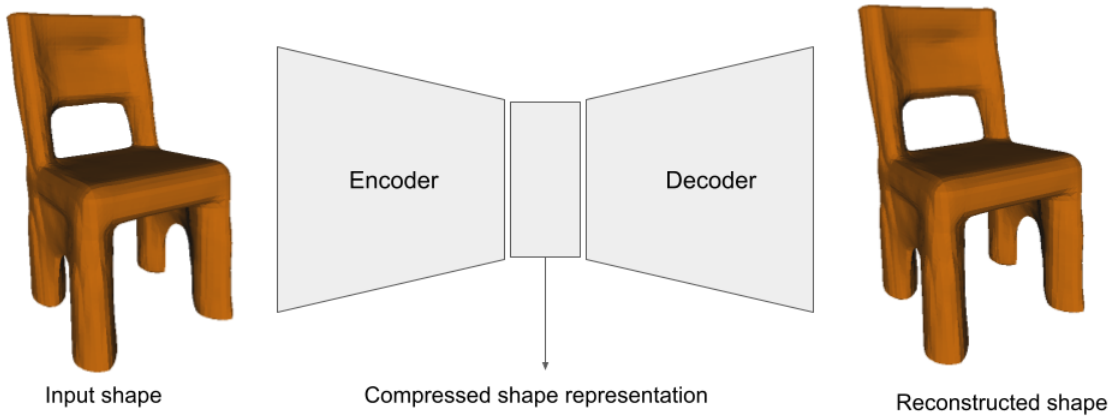

Figure 2.1: Workflow of a basic autoencoder network. An input shape x_i is passed to the encoder network which encodes the shape into a compressed representation: the latent vector y_i. The decoder network decodes y_i back into a shape x_i'. The autoencoder is trained so that the reconstructed shape x_i' from y_i is as close as possible to the original input shape x_i.

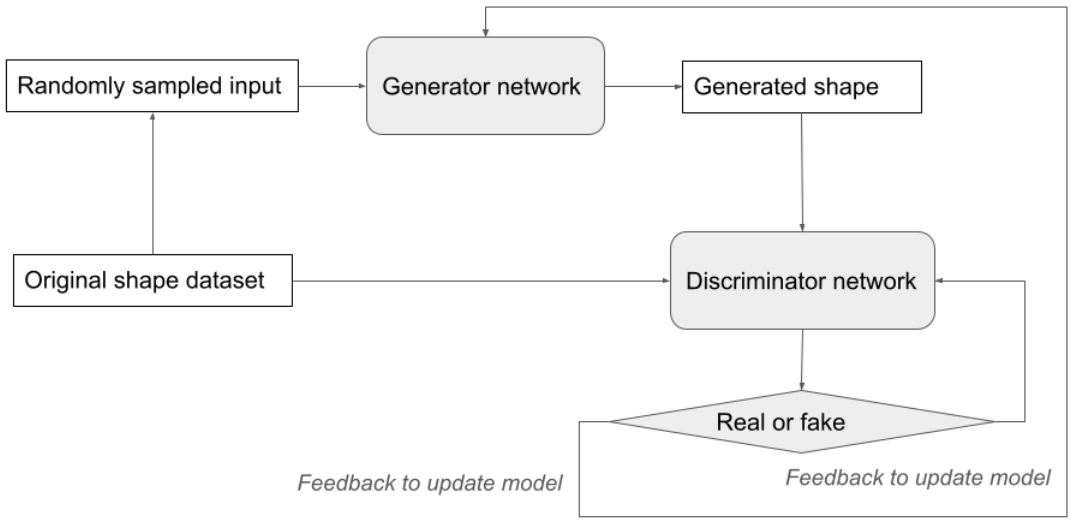

Figure 2.2: Workflow of the Generative Adversarial Network (GAN). The generator and discriminator networks work in conjunction so that the generator can learn a low-dimensional shape representation.

classic representations for 3D objects is the triangle mesh. A triangle mesh represents the geometry of a shape as a set of vertices, which are points sampled from the shape's surface, and also stores topological information in the form of a graph that encodes triangles connecting the vertices. However, since meshes contain this additional topological information, it is difficult to process them with neural networks. Thus, methods based on deep networks have been based on alternative shape representations. Point clouds are one possible representation of shapes, where a point cloud only contains a set of 3D points sampled from a shape's surface. Point clouds are lightweight, but do not represent the topology of the shape. Voxel-based shape representations encode a shape as a set of voxels that are part of a 3D volume, where the voxels can be empty or filled with the shape. Finally, implicit functions represent an object as the zero-set of a function that returns a real value for any query point in 3D space. In the following sections, we discuss important shape creation methods that use these different shape representations.

2.3.1 Point cloud based shape creation

Qi et al. [10, 34] introduced PointNet and PointNet++, neural network models that can be used to classify and segment point clouds. These networks are based on symmetric functions that allow to process points given in any order. However, the networks do not encode generative models, mainly enabling analysis tasks. Achlioptas et al. [1] introduce an autoencoder architecture to learn compact representations of shapes in the form of point clouds. The representations can then be used to reconstruct unseen shapes from input samples. The network can also be used for shape editing through algebraic manipulations, such as semantic part editing, shape analogies, shape interpolation, and shape completion.

2.3.2 Volumetric shape creation

Convolutional Neural Networks are commonly used to learn generative models of 3D shapes from voxel-based shape representations [7, 46]. These methods can be used for reconstructing and interpolating shapes. However, the quality of the generated shapes is poor, given that the shapes are represented in voxelized form.

The latent descriptor networks of Xie et al. [47] model 3D shape data with bottom-up descriptive features learned from the data. This method improves the visual shape

quality from regular CNNs, but still uses a volumetric shape representation.

Octree-based Convolutional Neural Networks (O-CNN) provide an efficient way to represent shapes in volumetric format for shape classification, segmentation and retrieval [42]. The O-CNN can handle shapes with high resolution efficiently as the shapes are represented with an octree that adapts to the data density, while the network can perform 3D convolution operations by benefiting from the sparsity of the data. However, to provide shapes with enough smoothness and details, a high number of octree levels is needed. Dai et al. [15] introduce a method for 3D shape completion from partial volumetric scans using 3D-Encoder-Predictor CNNs. A partial volumetric scan of a shape is passed through a 3D shape classifier network, which estimates the closest match of the scan to a training shape, and then fills the missing parts to reconstruct the shape. The method can be applied mainly for completion of small missing regions of the shapes, not for synthesis of entirely new shapes.

Liu et al. [27] propose a method for interactive 3D modelling using a generative adversarial network (GAN). The GAN model learns a latent representation of a set of shapes. Then, the user can model a partial shape in a volumetric representation, and the system retrieves the closest matching shape in the latent space, updating the user input with a more refined shape. The process is then repeated after each user interaction. Thus, the method allows users to interactively create shapes, although the resulting shapes are created in volumetric form.

2.3.3 Implicit function based shape creation

DeepSDF encodes shapes in the form of an implicit function [33]. Specifically, DeepSDF uses the concept of a Signed Distance Function (SDF) to learn and render a class of 3D shapes. The SDF encodes the closest distance of a query point in 3D space to the surface of the shape. The function is signed, implying that any point that is outside of the shape has a positive distance, any point inside the shape has a negative distance, and points on the surface have a distance of zero. The goal of DeepSDF is to learn a generative representation for a set of shapes with this implicit representation, in order to render, reconstruct, and complete the shapes with high quality, especially when the shapes have a complex topology.

The implicit field decoder, or IM-NET [14], is a shape decoder network that generates both 2D and 3D shapes by decoding latent vectors generated from various encoder networks. By using an implicit representation, IM-NET generates shapes

with good visual quality. The quality of the generated shapes of this decoder network can be sampled at any resolution and is not limited by the resolution of the training shapes. The IM-NET decoder can be embedded into existing autoencoder or generator networks to render high quality shapes. We combine this decoder with the autoencoder used in our work to generate shapes with high visual quality.

DualSDF uses the SDF concept for shape manipulation rather than just shape reconstruction and rendering. DualSDF uses two levels of granularity, one for capturing fine-grained detail and the other encoding a coarse structural decomposition of the shape [20]. DualSDF can perform shape manipulation using the two-level representation, because the the underlying latent vector representation for the two different granular representations is coupled together. A change in one representation is directly reflected in the other. Specifically, the user can easily select a section from the coarse-grained representation and modify it. Then, the modification is transferred to the fine-grained representation via a latent shape decoder. Although the method allows for high quality and relatively easy modifications to a shape, it requires the user's direct participation in modifying the shape with editing operations. Also, the structure of the coarse representation can limit the amount of editing that can be done on the shapes.

2.3.4 Shape creation form other input representations

Several sketch-based 3D shape creation methods were already introduced in the pre-deep learning era, since sketching allows users to create shape out of 2D description of shapes. With the rise of deep learning, methods were introduced which combine learning with sketching. this area became more robust. Huang et al. [22] introduced a sketching-based method which creates shapes with procedural modelling (PM). The user draws a sketch and then a convolutional neural network (CNN) infers the parameters of the procedural model which generate a shape similar to the sketch. Specifically, since a user sketch is merely an approximation of the desired shape, the system produces multiple corresponding shapes that closely resemble the drawing. Then, the user can choose one of the resulting shapes or make further modification on the PM parameters to generate their desired shape. One of the advantages of this method is that it can turn moderate sketches into good quality 3D shapes.

Wu et al. [45] introduced a sketching-based shape generation method, named MarrNet, which derives 2.5D sketches from input 2D sketches, and then reconstructs 3D

shapes from the 2.5D sketches. They demonstrated that this two-step approach provides better results than direct sketch to 3D shape translation. The main limitation of the sketch-based shape creation systems is that they requires a good sketching ability from the user to synthesize high-quality 3D shapes.

Deep learning based shape analysis makes it easy to parametrize shape attributes and to assign natural language words to shapes. This enables users to create, edit and retrieve shapes based on meaningful word tags. Text2Shape is a method that learns a joint embedding between shapes and natural language descriptions to synthesize shapes [13]. The method first establishes a many to many relation between shapes and word descriptions. Then, based on these relations, the method trains a generative joint embedding model. After training, the model can be used to synthesize shapes based on a textual description.

The method Jahan et al. [24] allows to explore a latent space of shapes with textual keywords. The user can specify word tags that reflect the characteristics of the desired shape, and the method maps the tags to a specific vector in the latent space. The user can then modify the shape by further exploring the latent space via a slider interface that allows to navigate the space around the latent space.

2.4 Exploration of latent spaces

The dimensionality reduction networks discussed above can reduce the high dimensionality of 3D shapes to low-dimensional latent vectors via their encoder networks, and reconstruct the latent vectors back to high-dimensional 3D shapes through their decoder networks. This is possible after the encoder and decoder are trained with a set of shapes. Moreover, exploration of the latent vector space can allow a user to synthesize new shapes. However, exploring the latent space is a difficult task as the latent vectors are high-dimensional vectors that represent the compressed information of a complete shape. Usually the latent vectors have hundreds of dimensions. This section discusses a few methods that seek to facilitate the exploration of latent spaces.

2.4.1 Exploration via interpolation

Interpolation is a very basic manner of exploring a latent space, where the user does not need any knowledge of shape attributes or distribution. In general, the user can

apply basic arithmetic operations to two or more latent vectors to explore the intermediate space between the vectors. Using a decoder network, new shapes can be synthesised from the modified latent vectors. For example, the user can linearly interpolate between the latent vectors of two selected shapes, to synthesize shapes that are a weighted combination of the selected shapes, where the quality of the blend will be determined by the model learned by the decoder network. A meaningful interpolation should capture a gradual transformation of one shape to another. However, a recent study has found that interpolation introduces a mismatch between the distribution of the interpolated points and the original prior distribution [29]. Moreover, interpolation only allows to explore the space that falls between the paths connecting the training shapes. It is not easy to meaningfully explore the rest of the latent space.

2.4.2 Exploration via Principal Component Analysis

Härkönen et al. [21] introduced a method for latent space exploration based on Principal component analysis (PCA). The PCA method decomposes a set of high dimensional data into a linear subspace so that the subspace captures the distribution of the data points preserving maximum information. Principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set. The second components accounts for the second largest variance in a direction orthogonal to the first component, and so on. Härkönen et al. apply PCA to different latent spaces learned by deep networks. Figure 2.3 illustrates the method, where $\mathbf{z}_j$ is one of the N randomly selected latent vectors from the set $Z_{1:N}$ of latent vectors learned with a GAN encoder. By applying PCA analysis on the latent space, they extracted PCA directions that capture specific attributes of shapes. They referred to those directions as the activation space. The data encoded in the activation space can then be mapped back to the latent space. In this manner, a user can blend different PCA directions together and generate new data from the blends. However, there is not clear reference for how to select different directions and weight them together.

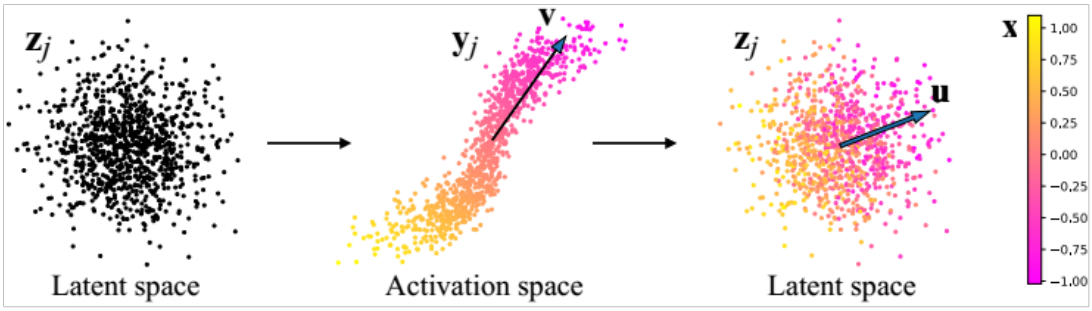

Figure 2.3: GAN control analysis using PCA, where the activation space is composed of the PCA directions of the original latent space. Image source: Härkönen et al. [21].

Chapter 3

Methodology

In this chapter, we first provide an overview of the proposed method for exploring latent spaces of shapes, followed with details for the different components of the method. The exploration method is divided into five steps: latent space embedding, Isomap reduction, new data sampling, inverse mapping, and decoding. Figure 3.1 illustrates the method with a diagram, while a more detailed overview of the method's steps is presented in Table 3.1.

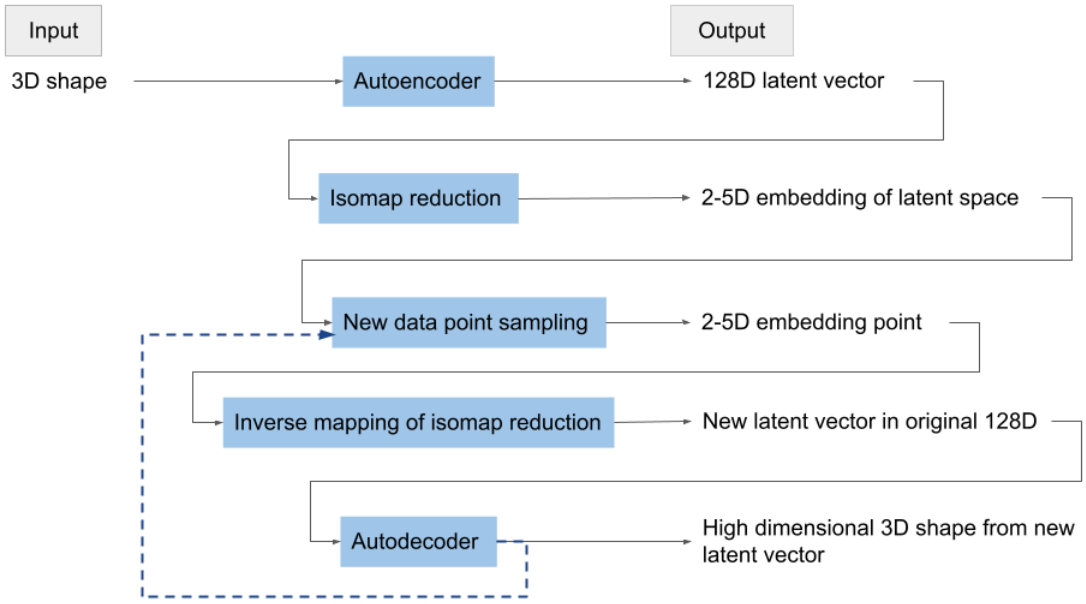

Figure 3.1: Summary of our method for exploration of latent spaces of shapes. The illustration shows the input, output, and data flow in each step of the method.

Table 3.1: Summary of our method for latent space exploration.

Step	Name	Description
1	Latent space learning with autoencoder	Given an input set of 3D shapes $\mathcal{S} = \{S_1, \ldots, S_n\}$, where n is the number of shapes in the set, we train an autoencoder neural network $\mathcal{AE}$ to learn a latent representation of $\mathcal{S}$, as described in Section 3.1. Then, we pass each shape $S_i \in \mathcal{S}$ through the trained autoencoder to obtain the shape's reduced representation (latent vector) z_i, where z_i has $d = 128$ dimensions.
2	Isomap to reduce dimensionality of latent space	We reduce the dimensionality of the latent space from $d = 128$ to e dimensions by computing an Isomap embedding of the latent space, where $e < d$. The value of e is typically between 2 and 5. We choose $e = 2$ in the case when users would like to visually explore the latent space. The Isomap method is described in Section 3.2.
3	New data point sampling	Given the Isomap embedding of the latent space, the user can now visualize the shapes as data points in the 2D embedding and select new points in the embedding.
4	Inverse mapping of data point to latent vector	Given a point selected on the Isomap embedding, we apply the inverse mapping described in Section 3.3 to map a data point p of dimension e back into a latent vector z of dimension d.
5	Shape synthesis from latent vector	The latent vector z is provided as input to the autoencoder network $\mathcal{AE}$ trained in Step 1 to generate an output shape S. By repeating Steps 3-5, the user can effectively explore the latent space and visualize the shapes corresponding to the points selected in the Isomap embedding.

3.1 Latent space learning

In our experiments, we learn a latent space for a set of 3D shapes with the 3D-GAE encoder [19] discussed in Chapter 2. The reason for selecting this variation of an autoencoder network is that the 3D-GAE has a better generalization ability, especially supporting more meaningful interpolation. The encoder network takes as input a shape voxelized into a $32 \times 32 \times 32$ volume. Occupied voxels are set to one, and empty voxels are set to zero. Both the encoder and decoder network are symmetric in structure. In the network, the input shape is downsampled to a $4 \times 4 \times 4$ volume through 3 convolutional layers. The filter sizes of the three convolutional layers are respectively 32, 16, and 8, where a kernel of size $4 \times 4 \times 4$ and stride of 2 are used for all the convolutional layers. The final layer of the network is a fully-connected layer with a rectified linear unit (ReLU) that outputs a latent vector with 128 dimensions.

To decode shapes from latent vectors, we use an implicit decoder which can reconstruct high quality shapes with smooth surfaces, and whose outcome resolution is not limited by the input shape structure. We use the IM-NET [14] discussed in Chapter 2 for this purpose. The decoder network takes as input a latent vector and position of a point in 3D and returns a value that indicates whether the point is inside or outside of the corresponding shape. Based on the values of this function, we can reconstruct a surface with iso-surface extraction and marching cubes.

3.2 Isomap reduction

Isomap is a nonlinear dimensionality reduction method. If we have a complex nonlinear manifold of high dimensional data, isomap reduces the dimensionality of the input dataset while capturing the intrinsic geometry of the data in the embedding [40]. Specifically, the Euclidean distances in the embedding approximate the geodesic distances in the manifold. In contrast, other dimensionality reduction techniques such as PCA and Multidimensional Scaling (MDS) fail to capture the data point interrelation which is represented by the geodesic distance and neighborhood proximity between them. The Isomap method operates in the following manner:

1. The method starts by selecting the r nearest neighbors for each data point and builds a neighborhood graph. The parameter r should be a positive integer less than the total number of data points.

2. Then, based on the neighborhood graph, the method computes the matrix of shortest paths in the graph between all pairs of data points. The shortest paths in the graph are taken as an approximation of the shortest geodesic paths in the data manifold.

3. Finally, the method creates an embedding of reduced dimensionality by applying multidimensional scaling (MDS) based on the computed geodesic distances in the manifold. MDS is a dimensionality reduction method where a high-dimensional set of data is approximated in a lower-dimensional embedding such that a specific relation between the high dimensional data is preserved in the low-dimensional embedding. This relation can be the pairwise Euclidean distances of the points, geodesic distances, or any other relationships of the data.

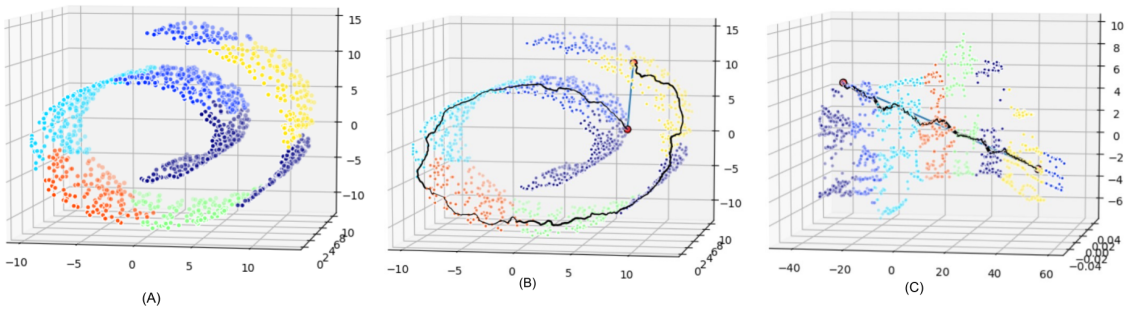

Figure 3.2: Isomap example: (A) Complex 3D Swissroll manifold data. (B) Two arbitrary points on the dataset (circled in red) are indicated along with their Euclidean distance with a blue dotted line and their geodesic distance with a black solid line. (C) The reduced-dimensionality 2D embedding of the data computed by the Isomap method, where the blue and black lines correspond to the Euclidean and geodesic distances shown in (B).

As described above, the isomap method requires to specify a parameter r that indicates the number of nearest neighbors to be used in the construction of the isomap, where $1 < r < n$. The specific value used for r is indicated in Chapter 4.

In Figure 3.2, we see an example where the two-dimensional embedding recovered by Isomap in Figure 3.2(C) reflects the shortest path distances in the neighborhood graph. Specifically, since the blue and black lines are almost similar in the embedding, the embedding preserves the graph distances.

In our method, we use the Isomap dimensionality reduction method to reduce the dimensionality of the latent vectors from $d = 128$ to $e \in [2, 5]$. The reduced points provide an embedding of the data that can be easily visualized and explored by a user if $e = 2$ or 3. Most of our experiments are conducted for $e = 2$. We selected the reduced dimension as 2 because 2D navigation is easier for non-expert users. We used $e > 3$ mainly to compare changes across multiple dimensions when comparing our results to a PCA-based method.

3.3 Inverse mapping of Isomap embedding

Isomap provides an embedding with reduced dimensionality which is easy to visualize and sample. However, points sampled from the embedding have a different dimensionality than the latent space. Thus, our goal for the inverse mapping is to define a function that maps the low-dimensional points back to the original latent dimension in a reasonable manner. We define this inverse mapping function as the solution to a scattered interpolation problem. Since our goal is to map multiple input variables into multiple output variables, one of the best options to perform such a scattered interpolation is to use Radial Basis Function (RBF) interpolation [31]. Note that RBF interpolation has been used in computer graphics for surface reconstruction, where the goal is to interpolate samples to obtain an implicit surface representation [9].

Given the latent vectors of shapes $\mathcal{Z} = \{z_1, z_2, \ldots, z_n\}$ of dimension d and their corresponding Isomap points $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ of dimension e, our goal is to find a function $\phi(p) : \mathbb{R}^e \to \mathbb{R}^d$. In the context of RBF interpolation, we formulate ϕ as a set of functions ϕ_j such that

$$\phi_j(p) = \sum_{i=1}^{n} \alpha_i^j \, k(p_i, p) \tag{3.1}$$

provides the j-th coordinate of the latent vector z corresponding to point p in the embedding. In Eq. 3.1, $k(p_i, p_j)$ is a radial basis function and α_i^j are real constants. In our work, we use the Euclidean distance function $k(p_i, p_j) = \sqrt{(p_i - p_j)^2}$. We tested our method with other RBF functions such as the Gaussian kernel, the cubic function, etc. However, for our specific dataset, the Euclidean distance provided the best inverse mapping. We speculate that this may be due to the sparsity of the data. Since the radial basis function is fixed and independent of the data, ϕ is fit to the

data mainly by determining the α_i^j coefficients.

We find the α_i^j coefficients such that the function outputs the expected values z_i for the known samples p_i. This can be stated as the solution of a linear system. Specifically, we first define the following $n \times n$ kernel matrix K:

$$K = \begin{bmatrix} k(p_1, p_1) & k(p_1, p_2) & \dots & k(p_1, p_n) \\ k(p_2, p_1) & k(p_2, p_2) & \dots & k(p_2, p_n) \\ \dots & \dots & & \dots \\ k(p_n, p_1) & k(p_n, p_2) & \dots & k(p_n, p_n) \end{bmatrix}. \tag{3.2}$$

Then, the j-th dimension of the expected output of the function for the known samples can be encoded in the following linear system:

$$\begin{bmatrix} k(p_1, p_1) & k(p_1, p_2) & \dots & k(p_1, p_n) \\ k(p_2, p_1) & k(p_2, p_2) & \dots & k(p_2, p_n) \\ \dots & \dots & & \dots \\ k(p_n, p_1) & k(p_n, p_2) & \dots & k(p_n, p_n) \end{bmatrix} \begin{bmatrix} \alpha_1^j \\ \alpha_2^j \\ \dots \\ \alpha_n^j \end{bmatrix} = \begin{bmatrix} z_1^j \\ z_2^j \\ \dots \\ z_n^j \end{bmatrix}. \tag{3.3}$$

For example, for the first point p_1, the linear system states that:

$$\phi_j(p_1) = \sum_{i=1}^{n} \alpha_i^j \, k(p_i, p_1) = z_1^j. \tag{3.4}$$

When considering all the dimensions d, the linear system can be extended as:

$$\begin{bmatrix} k(p_1, p_1) & k(p_1, p_2) & \dots & k(p_1, p_n) \\ k(p_2, p_1) & k(p_2, p_2) & \dots & k(p_2, p_n) \\ \dots & \dots & & \dots \\ k(p_n, p_1) & k(p_n, p_2) & \dots & k(p_n, p_n) \end{bmatrix} \begin{bmatrix} \alpha_1^1 & \alpha_1^2 & \dots & \alpha_1^d \\ \alpha_2^1 & \alpha_2^2 & \dots & \alpha_2^d \\ \dots & \dots & & \dots \\ \alpha_n^1 & \alpha_n^2 & \dots & \alpha_n^d \end{bmatrix} = \begin{bmatrix} z_1^T \\ z_2^T \\ \dots \\ z_n^T \end{bmatrix}. \tag{3.5}$$

In short form, this is:

$$KA = Z, \tag{3.6}$$

where the $n \times d$ matrix A is found by solving the linear system.

Once the coefficients α_i^j have been found by solving the linear system, we can compute ϕ for any point with Eq. 3.1, including newly sampled points.

3.4 Exploration of latent space via Isomap

Sections 3.2 and 3.3 discussed how to reduce the dimensionality of latent vectors and then map them back to the original latent space. This section explains how to use these two components to explore the latent space and synthesize new shapes.

The Isomap method reduces the dimensionality of the latent vectors from 128 to a number specified by the user, which is usually between 2 and 5. For visual exploration, we reduce the dimensionality to 2, since navigating a 2D plot is easier for non-expert users. Thus, our system first presents a 2D plot of the distribution of the latent vectors in the reduced embedding, as shown in Figure 3.3(A). The user can then freely select new sample points in this 2D latent space. Once a point is selected, we compute the inverse mapping of the point, which results in a latent vector. Then, we synthesize a 3D shape from the latent vector, which is presented to the user for inspection. In order to explore the reduced embedding in a meaningful manner, the user can use other points in the plot as reference, or freely explore the embedding.

Before generating any shape, the user can first explore the embedding by clicking on different regions or points of the plot and acquiring some knowledge of the characteristics of the shapes that exist in different regions of the plot. Then, the user can use this knowledge to explore the plot and generate shapes that combine different characteristics of the shapes as desired.

For example, as shown in Figure 3.3(B), the user can interpolate between two shapes in the dataset. By inspecting different shapes along the interpolation path, the user is able to generate shapes with gradual changes that combine the characteristics of the shapes at the endpoints. The user can also select one known point and extrapolate in a given direction. Finally, as shown in Figure 3.3(C), the user can also inspect the shapes interpolated along the path formed by two newly selected points. In Chapter 4, we discuss how these different types of exploration provide advantages over the alternative methods.

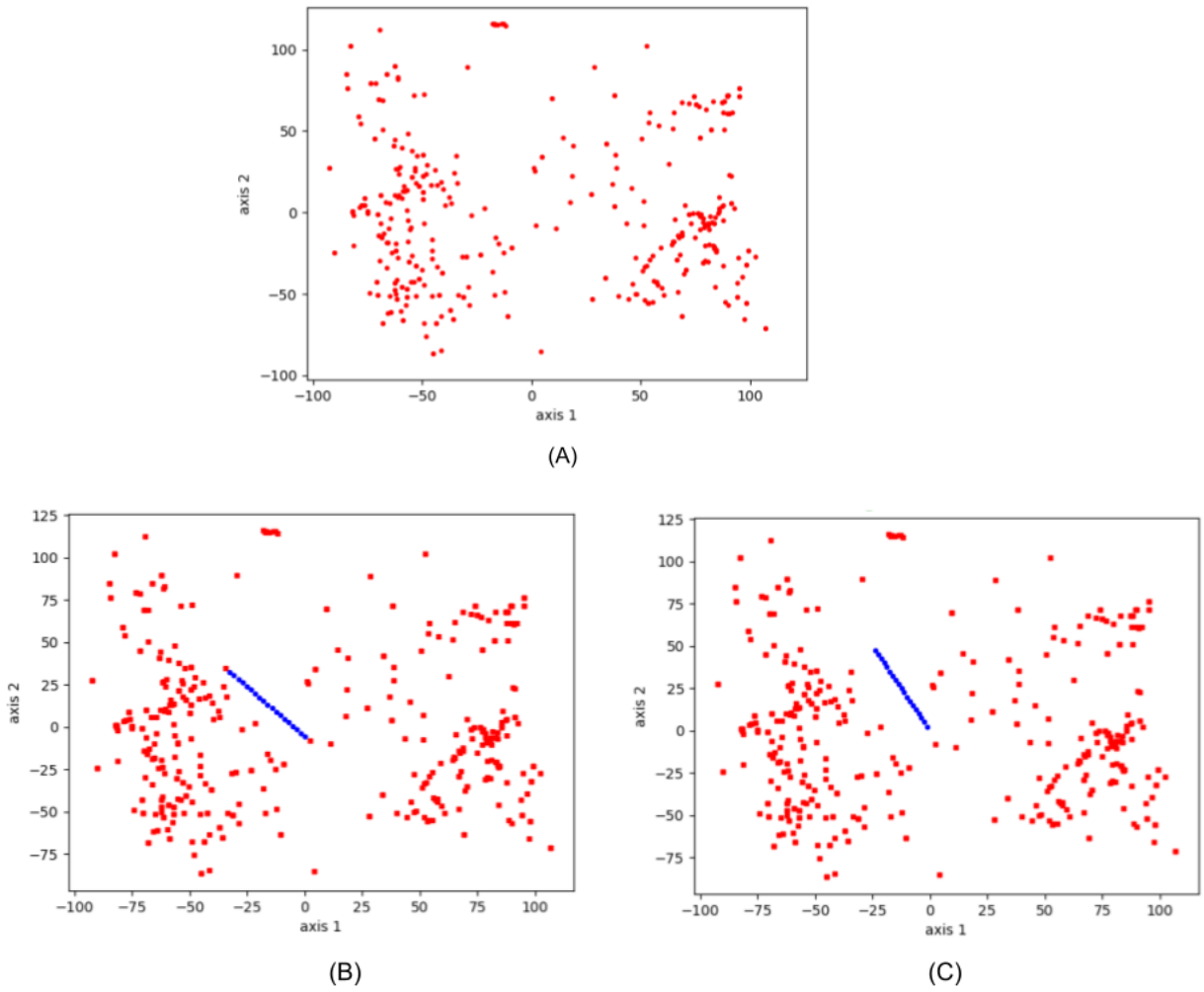

Figure 3.3: Exploration of the latent space: (A) Isomap embedding of a dataset of chairs, where each red dot represents a specific chair in the dataset. (B) Exploration by interpolating between two known shapes, where the blue dots represent new points selected by the user. (C) Exploration by inspecting the shape interpolated between two new sample points.

3.5 Constrained Isomap embedding

The method described so far can be applied to a set of shapes without the need of any labels that describe the characteristics of the shapes. The embedding created by Isomap is derived solely from the learned latent representation. However, there may also exist contexts where the user may have labeled data available describing the characteristics of a subset of the shapes. Thus, it would be beneficial to take the

labels into account when organizing the shapes in the Isomap embedding.

An advantage of the Isomap method over other dimensionality reduction methods is that it can be used to create an embedding constrained by additional data relationships. As described above, the first step of the Isomap method is to retrieve nearest neighbors of points and create a neighborhood graph of the data. Then, the method computes the geodesic distances among the data. Finally, the method reduces the dimensionality of the data in a manner so that the Euclidean distances in the embedding approximate the geodesic distances. This ensures that the placement of data in the low-dimensional embedding is based on the similarities of the data in the high-dimensional representation. Data points that are similar in characteristics are placed close to each other in the embedding, and dissimilar data points are placed far apart in the embedding.

The similarity of the data is reflected by the geodesic distances, which in turn are derived from the neighborhood graph. Thus, modifying the geodesic distances or neighborhood graph would allow to include additional constraints in the embedding. Specifically, in our method, we choose to modify the neighborhood graph, which ensures that the geodesic distances computed from the graph are consistent. Modifying the geodesic distances directly could result in inconsistent distances among the points. Note that we have not found any reference in the literature that proposes such a constrained method. However, the Isomap method was originally designed to embed qualitative and categorical data, where point similarities are dictated by human labeling rather than measurements of numerical properties. Thus, the method is able to handle input graphs of any form.

In our context, given a subset of the data with labels, after constructing the neighborhood graph for all the data based on the similarity of latent vectors, we add additional edges to the graph between data points with the same labels, e.g., if two chairs are described as having "four legs", we add a shortcut edge between the corresponding points in the graph. This ensures that these two points should be maintained close in the embedding. Once the geodesic distances are computed, the Isomap embedding brings closer together not only the labeled points, but any neighbor points of the labeled points, resulting in a grouping in the embedding that also takes into account the label similarity.

Note that it is also possible to remove edges from the graph, e.g, if two data points should be embedded far apart, or to create more complex rules to add edges between

data points, such as considering multiple labels. However, we did not experiment with these options as they are not relevant in the context of our data. We show in Chapter 4 that this method successfully groups together shapes with similar labels.

3.6 Encoding and Decoding networks

To encode the shape dataset into latent vectors, we use the Generalized Autoencoder (GAE) network. A description of the network is already given in Chapter 2. The justification for using this specific encoder is that this method enables a more meaningful interpolation of latent vectors, since the GAE reconstructs multiple samples similar to a given input, based on a weighted similarity metric. Meaningful interpolation among latent vectors is particularly important for our application scenario. Our GAE network takes $32 \times 32 \times 32$ voxel shapes as input and produces 128 dimensional latent vectors. The encoder uses 3 levels of downsampling from 32^3 to 4^3, and the decoder is symmetric. We use a kernel of size $4 \times 4 \times 4$ and stride 2 for all the convolutional layers. The reason for using 128 as the size of the latent vector is that it provides a balance between the reconstruction error and the complexity of the model. Reducing the size of latent vectors will cause the loss of important details from the shapes. In our experiments, we used IM-GAN as the decoder network, which is also discussed in Chapter 2. We selected this particular decoder as it has the special benefit of rendering high quality shapes regardless of the input shape resolution.

Chapter 4

Experimental results

In this chapter, we first provide some background on the datasets used in our experiments, the pre-processing performed on the data, and the parameters used by our method. Then, we present qualitative results of our method in order to compare it to existing methods. After that, we show the results of additional experiments to demonstrate other features and benefits of our method, and also present results of a quantitative evaluation of the different components of our method.

4.1 Shape dataset

We conducted our experiments on three sets of shapes: chairs, tables, and lamps. The chair set is taken from the COSEG [44] dataset and consists of 400 different types of chairs. The table and lamp sets are taken from the ModelNet40 [37] dataset. The table set contains 400 different shapes while the lamp set contains 50 shapes. Most of our experimental results are demonstrated on the Chair dataset as the shapes in this dataset are larger in quantity and also of high quality. On the other hand, the Table dataset has the same quantity of shapes, but the shapes are noisy and some shapes are incomplete. The Lamp dataset is the smallest dataset, and the shape quality is moderate. Data samples from each category are presented in the appendix.

Moreover, for quantitative evaluation, we use a dataset of procedurally-generated shapes composed of 400 chairs with parameters that encode the characteristics of the shapes [3]. The chairs in the dataset are divided into three main regions: back, seat, and leg. The properties of the backs of the chairs are described with 12 parameters, seats are described with 2 parameters, and legs are described with 11 parameters. Thus, a total of 25 parameters describe the overall composition of a chair.

4.2 Data pre-processing

For training the encoder neural network, the shapes were voxelized to volumes of $32 \times 32 \times 32$ resolution. Each voxel in a volume is a binary number, where voxels occupied by a shape are denoted by one, and empty voxels are denoted by zero. For training the implicit decoder network, we selected samples around the shapes according to the procedure discussed in Chen and Zhang [14]. The implicit decoder is trained to reconstruct high-quality shapes regardless of the input resolution. This is achieved by learning an implicit function that provides the distance to a query point in space based on the point's coordinates and a latent vector.

4.3 Parameters

Regarding the value r mentioned in Section 3.2, which determines the neighborhood size used by the Isomap method, we experimented with values of r in the range of 5 to 10. The value 7 provided the most meaningful embeddings for the chair and table datasets, as both of them have 400 shapes. For the lamp dataset, we got more meaningful embeddings with $r = 8$, as the dataset has a smaller number of shapes (50 shapes).

4.4 Qualitative evaluation

In this section, we present several visual examples of experiments on the selected three shape sets to compare our method to existing methods. Our goal is to demonstrate the advantages of our method over existing methods for latent space exploration and also discuss limitations. Since PCA-based exploration and latent vector interpolation are two methods proposed for exploring latent spaces for new shape synthesis, we show results of our method while comparing to these alternative methods. After that, we highlight other benefits of our method with additional qualitative examples.

4.4.1 Comparison to PCA-based latent space exploration

The PCA method provides several principal directions that can be navigated to synthesize shapes with different features. Since we cannot predict beforehand which of the directions produce meaningful variations in the shapes, we need to inspect the

directions individually. In Figure 4.1, we show the variation of an individual chair along four different PCA directions. The original chair when exploring each direction is always the same, and is highlighted with a red box in the figure. Each PCA direction is varied from -1.5 to +1.5 with a step change of 0.5.

We see that in Direction 1 (Row 1), the chair stays well structured in the range -1.0 to 0.5. Outside of this range, the chair gets distorted. In Direction 2 (Row 2), valid shapes are found in the range 0.0 to 0.5. In Direction 3, the valid range is -0.5 to 1.0. In the last row, the valid range in Direction 20 is 0.0 to 0.5. Thus, the valid range differs for each PCA direction. In addition, not every direction produces unique shapes. Direction 2 and Direction 20 produce fairly similar shapes. Only the first few directions provide unique changes to the shapes.

In addition, it is difficult to determine the appropriate step size to explore a specific PCA direction. Figure 4.2 illustrates this. The top two rows of the figure show the PCA shape reconstructions obtained with gradual change along direction 1 from factor -1.75 to 1.5 with constant step change of 0.25. The first shape of the second row (with factor value 0) is the original chair on which the direction change is applied. Here we can see that reconstructed shapes in both positive and negative directions get distorted as the factor value goes away from zero. If we use a smaller step change to see the reconstruction changes in this PCA direction, we see that the variation in the shapes is limited (Rows 3 to 4 of the figure). Thus, it is difficult to determine an optimal step size to explore the latent space. If the step is too large, then the reconstructed shapes might get quickly distorted. If the step size is small, then the shapes may change slowly for each step. Thus, the step size depends on the PCA direction and dataset, which requires tedious manual selection for the navigation.

Our method reduces the latent space into a more manageable low-dimensional embedding, making it easier to explore the latent space. Exploration along the axes of the embedding more frequently results in well-structured shapes with unique feature changes. An example is shown in Figure 4.3. In order to compare to PCA, we use an Isomap embedding with 4 dimensions. Thus, we show the changes to a base shape along the full range of the four axes of the embedding. We selected the same chair which is used to demonstrate the changes in Figures 4.1 and 4.2 for PCA exploration. We see that, as we gradually navigate along each axis, the chair shape

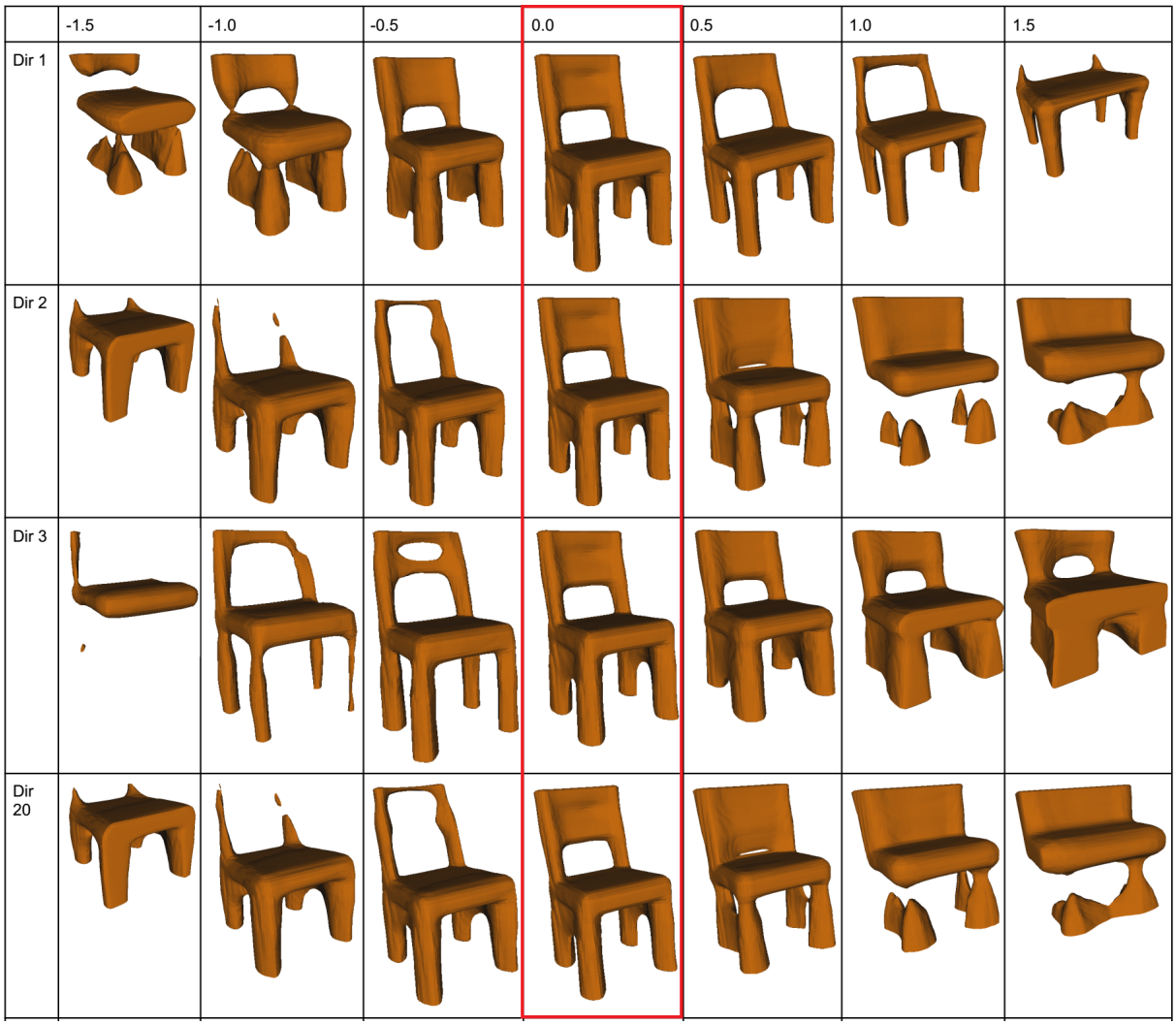

Figure 4.1: Shapes synthesized by moving the latent vector of a chair along different PCA directions. Each column corresponds to the displacement indicated in the header row, where the center marked in red corresponds to the original shape without any change (displacement 0.0). The first row of the table shows the shapes synthesized by moving along direction 1 of the PCA decomposition. The second and third rows show the motion along directions 2 and 3, respectively. The fourth row shows the shapes obtained along direction 20.

changes gradually. Every time the change gives a complete shape which has slightly different characteristics from the other shapes. The reason for this behavior is that the inverse mapping performs a blending of the latent vectors of the training shapes, weighted by the proximity to the selected point. Thus, the blending results in valid

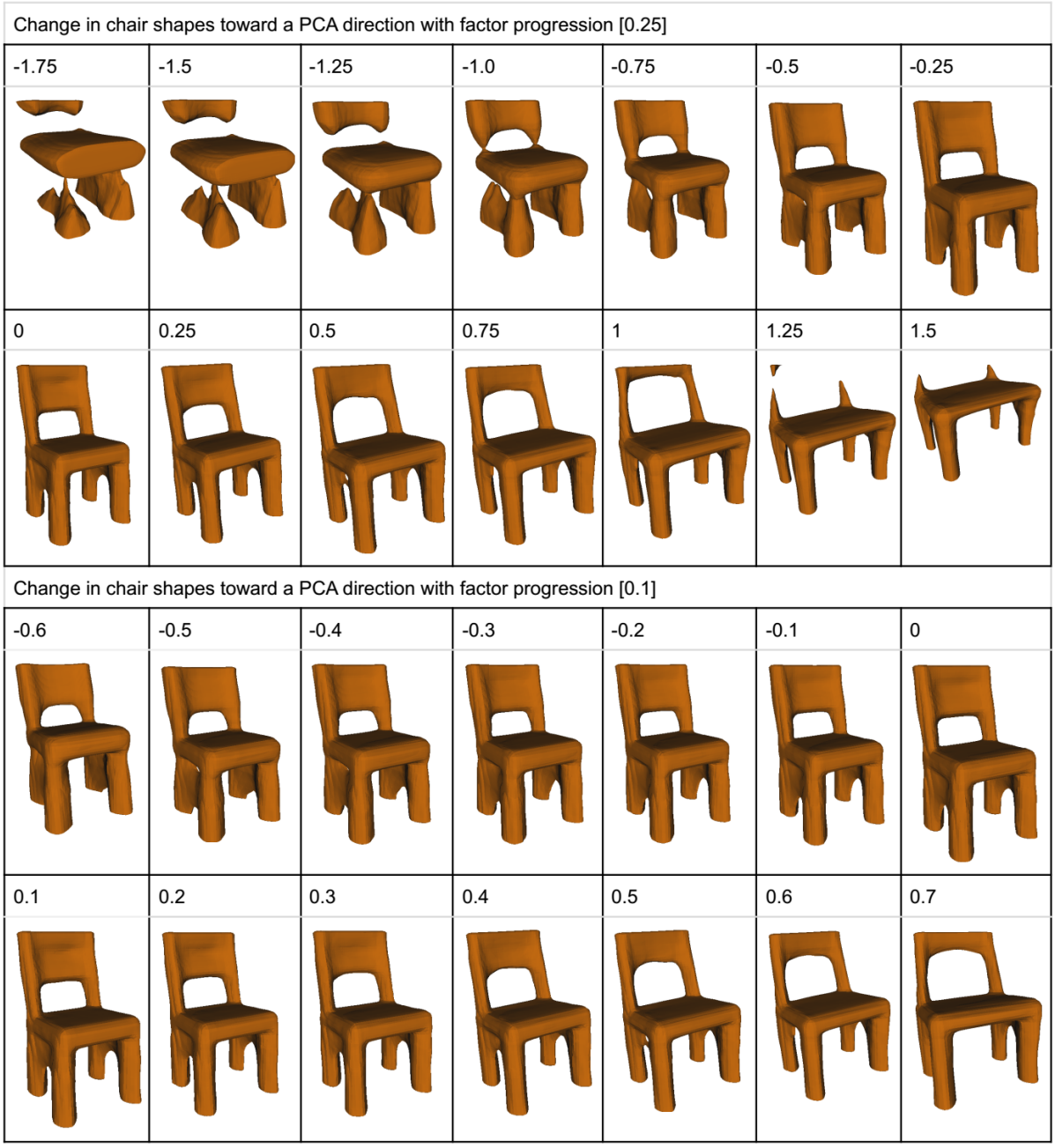

Figure 4.2: Shapes synthesized by moving the latent vector of a chair along direction 1 of the PCA decomposition with different step sizes. The first and second rows show the shapes obtained with a large step of [0.25], while row 3 and 4 show the shapes obtained with a smaller step of [0.1]. The original chair is indicated by the step with value 0.

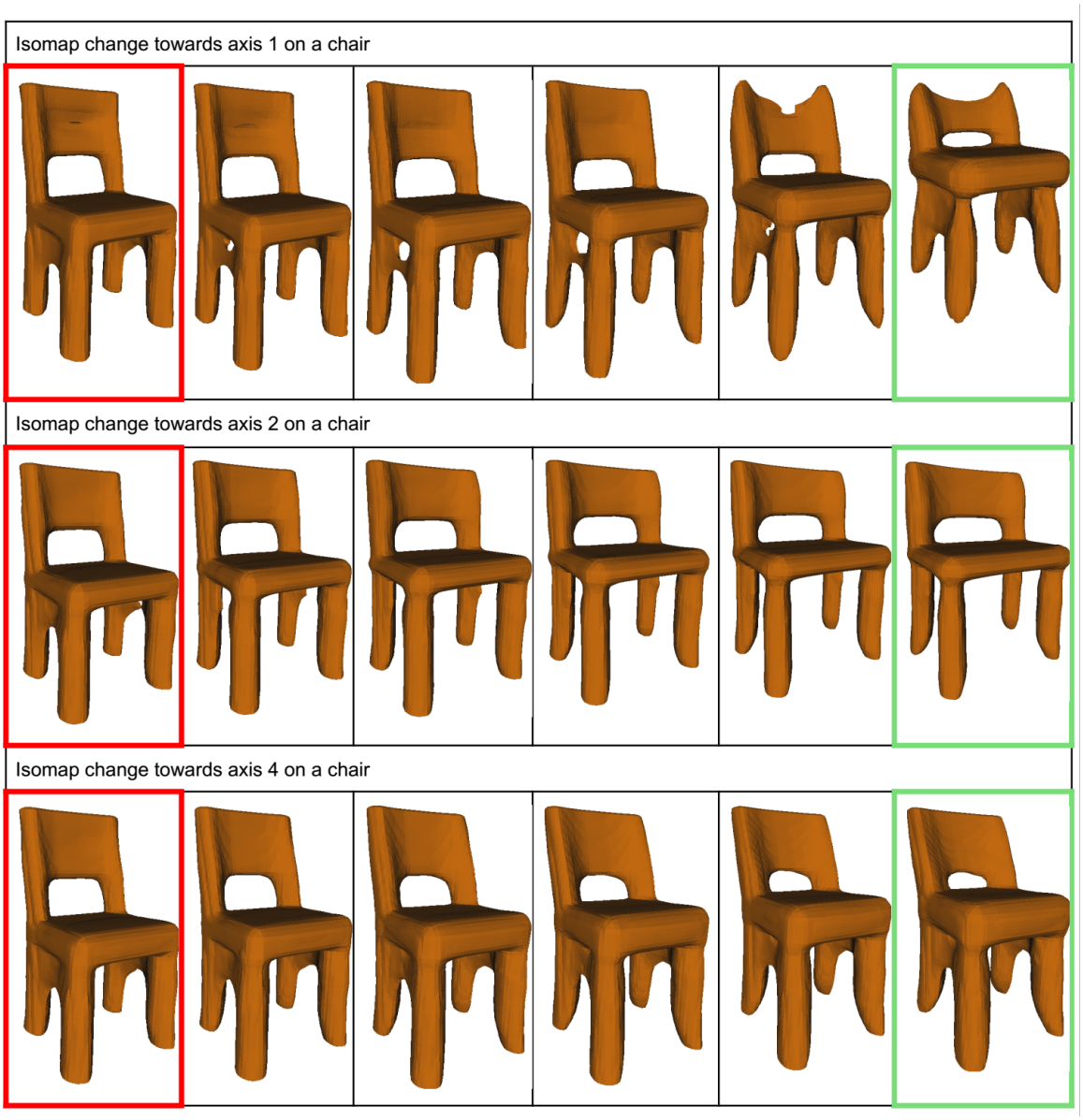

Figure 4.3: Isomap reconstruction corresponding to changes along different axes on a chair. First row shows changes along axis 1. Second row shows changes along axis 2. Third row shows changes along axis 4. Note that, in the second row, the shapes change in their height.

shapes with different characteristics.

4.4.2 Comparison to latent vector interpolation

Interpolation is an effective manner to synthesize new latent vectors from existing latent vectors. However, this method is limited to combining the base shapes. Thus, it can only explore the space that falls in the interpolation path. Other regions of the latent space cannot be explored using interpolation.

Our method allows to explore the latent space freely regardless of any base latent vectors. Figure 4.4 shows a comparison between our method and interpolation. Specifically, Figure 4.4(B) shows an example of regular interpolation. On the top of the figure, we see the distribution of shapes in the embedding, where red dots are the original training shapes, and blue dots are newly selected latent vectors which are obtained by interpolating two base shapes at regular intervals. A total of 20 dots are connecting the two base points. The bottom of the figure shows the shapes reconstructed from the blue dots. Figure 4.4(A) shows the advantage of our method over regular interpolation, where the user freely selected two points in the embedding, and interpolated a path along the two new points. Note that the connecting path does not coincide with any interpolation path between two base shapes. We see that, as with traditional interpolation, the blue dots result in new and meaningful chair shapes. The shapes along the two paths are similar as the two blue lines are close to each other in the embedding. We purposefully selected two paths close to each other to demonstrate that the resulting shapes have slight differences, showing that the extra freedom allows to synthesize shapes different from endpoint interpolation. This allows users to synthesize shapes from regions which are not accessible via regular interpolation.

On the other hand, our method still allows the user to select two existing shapes and walk along the path between the two shapes. Figure 4.5 shows a comparison between interpolation done directly on the 128D latent vectors and interpolation done in the Isomap embedding via the inverse mapping. The columns in the figure alternate the two methods so that we can compare the corresponding shapes side by side. The side by side comparison of these two types of interpolation shows that these two methods produce almost the same shapes at the two extremities. The middle of the paths differs more. In Figure 4.6, we plot the Euclidean distance between the latent vectors obtained with each of the two interpolation techniques. The plot confirms the visual results and shows that the difference between shapes is higher in

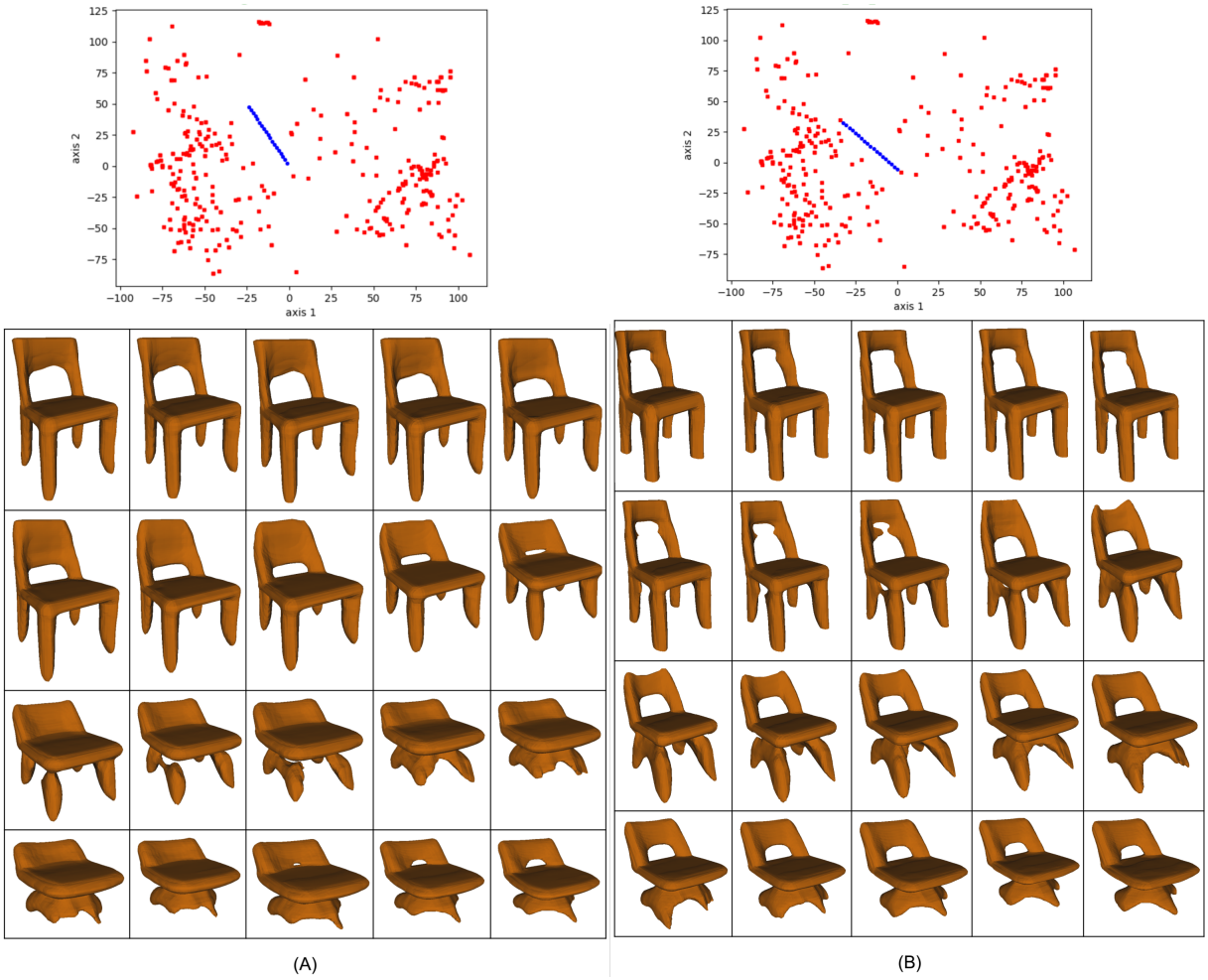

Figure 4.4: Regular interpolation compared to our method. The red points denote the training shapes, while the blue points are newly selected. The bottom row shows the shapes along the blue paths in reading order. (A) Our method can synthesize new shapes from points freely selected in the 2D embedding of the latent space. (B) Interpolation is guided by the base points and can only synthesize new shapes that fall in the interpolation path between the base points.

the middle of the path. The reason is that, when we perform the interpolation on the 2D embedding, we apply the inverse mapping on the interpolated new samples to get corresponding 128D latent vectors. The inverse mapping produces better estimated latent vectors when the input points are closer to the original data points used by the

inverse mapping.

4.4.3 Exploration of the latent space in different directions

Figure 4.7 shows an additional feature of our method, where the user can explore an interpolation path and then change direction along the path towards a different group of shapes. In Figure 4.7(A), we selected a few new data points (blue dots) in the 2D embedding along a specific path. As these points correspond to a straight line, the change reflected in the shapes is gradual. Then, given a specific point, highlighted with the green box, we can continue the exploration into a different direction of the embedding. Figure 4.7(B) shows such an example. We have selected new test points in 4 different directions to see how the starting chair changes. The new test points are indicated in green. We can see that, as we explore in different directions, the attributes of the selected chair change differently. This allows the user to freely explore the space while still gradually inspecting changes to the shapes, which is not possible with regular interpolation.

4.4.4 Shape neighborhood exploration

In Figure 4.8, we show some examples of shape retrieval according to the Isomap embedding for the chair dataset. We selected the shapes that are within 5 units of radial distance from a query shape. We that, in Figure 4.8(A), all the shapes retrieved are structurally similar to the query. All the shapes have four legs, a rectangular back and seat, and the height of legs and back maintains the same ratio for all the chairs. In Figure 4.8(B), all the chairs are stools just as the query chair. In Figure 4.8(C), all the chairs are narrow and tall. This shows qualitatively how the latent representation combined with Isomap group shapes with similar characteristics together.

4.4.5 Comparison to other embeddings

t-SNE is a dimensionality reduction method that is commonly used for visualizing high-dimensional data [41]. We experimented with using t-SNE for producing the embedding in our method in place of Isomap. Specifically, we experimented with applying the inverse mapping on this embedding. We randomly selected two chairs

from our chair dataset to inspect the reconstructions obtained from each embedding. Figure 4.9 shows these results. We see that Isomap paired with the inverse mapping produces good shapes, while the reconstruction from the t-SNE embedding provided inaccurate reconstructions.

The reason for these results can be seen in Figure 4.10. We explored the neighborhood of points in the 2D embedding obtained with t-SNE, similarly as Figure 4.8. The central shapes are the same in Figure 4.10 and Figure 4.8. We can see that the retrieved chairs are not that similar to the query shape. That means that t-SNE does not group similar chairs closely as Isomap does. Similar issue is visible in PCA neighborhood. We can see that in figure 4.11. For this reason, the Isomap embedding paired with the inverse mapping gives a much better result.

4.4.6 Results on other shape categories

Apart from the chair dataset, we also experimented with two other categories of shapes: tables and lamps. Figure 4.12 shows results for the table dataset, while Figure 4.13 shows results for the lamp dataset.

4.5 Quantitative evaluation

To provide some experimental evidence that, in general, our method allows the user to explore a set of shapes in a meaningful manner, we evaluate the two main components of our method in a quantitative manner. First, we evaluate the quality of the lower-dimensional Isomap embedding in terms of how well it organizes the shapes in the embedding. Then, we evaluate the accuracy of the inverse mapping in taking points in the embedding back to the latent space. If we can conclude that the shapes in the embedding are organized in a reasonable manner, and the inverse mapping of a point takes it back to a relevant latent vector, then the method should enable a meaningful exploration of the shape space.

4.5.1 Quality of shape organization in the Isomap embedding

Our requirement for the lower-dimensional embedding is that it should organize shapes in a meaningful manner, where different regions of the embedding should capture shapes with similar characteristics. For example, if the top-left region of the 2D embedding of a set of chairs captures all the chairs with four-legs, and the top-right captures all the chairs with one leg, then the user can use this knowledge to obtain variations of the desired type of chair leg by exploring the corresponding region in the embedding.

In order to evaluate the organization of the embedding in an objective manner, we perform experiments with a dataset where the characteristics of the individual shapes are known. Specifically, we use the dataset of procedurally-generated shapes introduced by Ali et al. [3], where each shape is summarized with a parameter vector. For example, chairs in the dataset have different characteristics such as varying number of legs, round or square geometry for the chair seats and backs, the presence or not of armrests, different lengths for legs and backs, etc. These characteristics are represented as integer or real parameter values, which are stacked together in a parameter vector for each shape.

Thus, when computing an Isomap embedding of this dataset, we expect that points which are close to each other in the embedding are also parametrically similar, implying that they have some characteristics in common. Based on this assumption, we calculate how well specific regions of the embedding capture certain sets of parameters. To perform this calculation, we divide the embedding space into small regions, and calculate how many parameter values are confined to each specific region. Specifically, we search for a subset of parameters that remains unchanged (or changes very little) in each region.

To find the parameters that are unchanged in each region, we sort all the points in the embedding, once for each axis, and count the corresponding parameter changes along each axis. For each possible parameter, we simply count the number of times the parameter value changes according to the sorted order along the axis. Then, the parameters which have a small number of changes (below a threshold) in a region correspond to the parameters tied to that region. In our experiments, we set the threshold to be the 25th percentile of the average number of changes along an axis.

Figure 4.14 shows an example of this computation where we divide the embedding into a regular rectangular grid. In our experiments, we performed this evaluation on

three different grid sizes: 2×2, 3×3, and 4×4. Tables 4.1 and 4.2 present the results. We see that, in the 3×3 grid, most cells contain 5 to 6 parameters with number of changes below the threshold along at least one axis. The zero value indicates that, in those specific areas, none of the parameters remain stable along the corresponding axis. In the 4×4 grid, the number of parameters with small changes is smaller. In the 2×2 grid, each cell contains 6 parameters with a small number of changes. We conducted the same experiments on other 2D embedding methods, such as t-SNE, and PCA. The result shows that on average Isomap shows better stability of parameters in regions for different grid sizes. On average, Isomap has 6 stable parameters per region for a grid of size 2×2, while t-SNE and PCA have 5.125 and 5.25, respectively. For grid size 3×3, Isomap has on average 4.22 stable parameters per region, while both t-SNE and PCA have 3.22 stable parameters. Thus, we can conclude that the Isomap roughly organizes shapes according to their characteristics, although the organization is not too fine, as shown by the 4×4 grid.

To visualize the similarity of shapes based on their parameters, we performed spectral clustering based on the parametric similarity of the shapes, where we use as a similarity metric the count of identical parameters between two shapes. Figure 4.15 shows the cluster distributions of different embeddings (Isomap, t-SNE, and PCA) for the chair dataset, where we divided the 2D embedding in 5 different clusters. We see that shapes with similar parameters are locally grouped together in all of the embeddings, although the global distribution of clusters is not that coherent.

4.5.2 Inverse mapping accuracy

To evaluate the accuracy of the inverse mapping process, we conducted a leave-one-out evaluation on our three datasets. For each shape in each dataset, we computed the inverse mapping with $n - 1$ latent vectors to produce the latent vector left out of the mapping. Then, we measured the Euclidean distance of the produced latent vector to the original left-out vector. We can consider the distance as the amount of error of the inverse mapping. Table 4.3 shows statistics on the errors computed for all the shapes in the datasets.

To understand the significance of the error, we need a reference to compare with. For that purpose, we computed the average distance to the 8 nearest neighbors of each

shape according to the latent vectors. If the average error of leave-one-out evaluation is close to average neighborhood distance, then the inverse mapping is approximating the latent vector with reasonable accuracy. The average neighborhood distances are also reported in Table 4.3. We see that the average error for the chair dataset (23.29) is close to the average neighborhood distance (22.43). This indicates that the inverse mapping results are close to the original shape. The error for the table dataset is also close to the average neighborhood distance. However, the distance is quite large

Cell	Axis 1	Axis 2
1,1	6	6
1,2	6	6
1,3	6	6
2,1	0	5
2,2	0	6
2,3	0	5
3,1	0	0
3,2	6	6
3,3	6	6

Table 4.1: Number of parameters with small number of changes or no change in each cell of the Isomap embedding of the chair dataset, when divided into a 3×3 grid. Here Axis 1 and Axis 2 are from the 2D embedding plot of Figure 4.14.

Cell	Axis 1	Axis 2
1,1	0	0
1,2	0	0
1,3	6	6
1,4	0	5
2,1	0	0
2,2	0	0
2,3	5	0
2,4	0	0
3,1	0	5
3,2	0	0
3,4	5	0
4,1	6	6
4,2	0	0
4,3	5	5
4,4	5	0

Table 4.2: Number of parameters with small number of changes or no change in each cell of the Isomap embedding of the chair dataset, when divided into a 4×4 grid. Here Axis 1 and Axis 2 are from the 2D embedding plot of Figure 4.14.

for the lamp dataset. This may be due to the small number of shapes present in the dataset (50 shapes), where the sparser sampling of shapes may lead to worse scattered interpolation.

Dataset	Minimum error	Maximum error	Average error	Average neighborhood distance
Chair	0.083019	75.188494	23.294272	22.434888
Table	0.399150	74.076806	33.0235208	27.261888
Lamp	5.133253	362.952199	48.933471	26.151401

Table 4.3: Leave-one-out evaluation of the inverse mapping for three datasets. The first three columns show the minimum error, maximum error, and average error respectively for each dataset. The last column shows the average distance of 8 closest latent vectors of a shape for each dataset.

4.5.3 Constrained Isomap embedding

Figure 4.16 shows an example of the constrained Isomap embedding used with our method. In the dataset of 400 chairs, 10 chairs have two legs, and 31 chairs have three legs. For this experiment, we selected 6 two-leg chairs and 6 three-leg chairs and added constraints to keep chairs with the same label close in the embedding. Other chairs were kept unlabeled and no constraints were added for them. We see in Figure 4.16(C) and (D) that, with the constrained embedding, the 6 two-leg chairs and 6 three-leg chairs are arranged in two closely tied groups which look almost like single points. Unlabeled chairs are brought closer to their respective type of chair, but the grouping is not perfect. We can see that the grouping in the original embedding in (A) and (B) is quite different. This constrained embedding is useful if the user has additional semantic information about the shapes and would like to group shapes with specific characteristics in 2D embedding based on this information, so that the user can further explore the latent 2D embedding constrained by this information.

4.6 Timing and machine configuration

The machine setup used in our experiments comprises an Intel CPU (Skylake IBRS) 2.1GHz with an NVIDIA GeForce GTX 1070 Ti GPU with 8GB of memory. Our method can be divided into two parts: (1) Training the encoder and decoder networks to learn the latent representation. (2) Exploration of the latent space with Isomap and inverse mapping. The training takes on average around two hours. The average time to compute an Isomap, inverse mapping, and shape generation on different datasets are given in Table 4.4. The last column of the table shows the total time needed to synthesize a complete new shape using our method. We see that the average time to produce new shapes be exploring the latent space is around 5 seconds.

Dataset	Isomap time	Inverse mapping time	Shape construction time	Total time (sec)
Chair	0.38	3.52	1.27	5.17
Table	0.46	3.56	1.28	5.30
Lamp	0.30	3.54	1.22	5.06

Table 4.4: Average computation time (in Seconds) needed for Isomap reduction, inverse mapping, and shape reconstruction on different datasets.

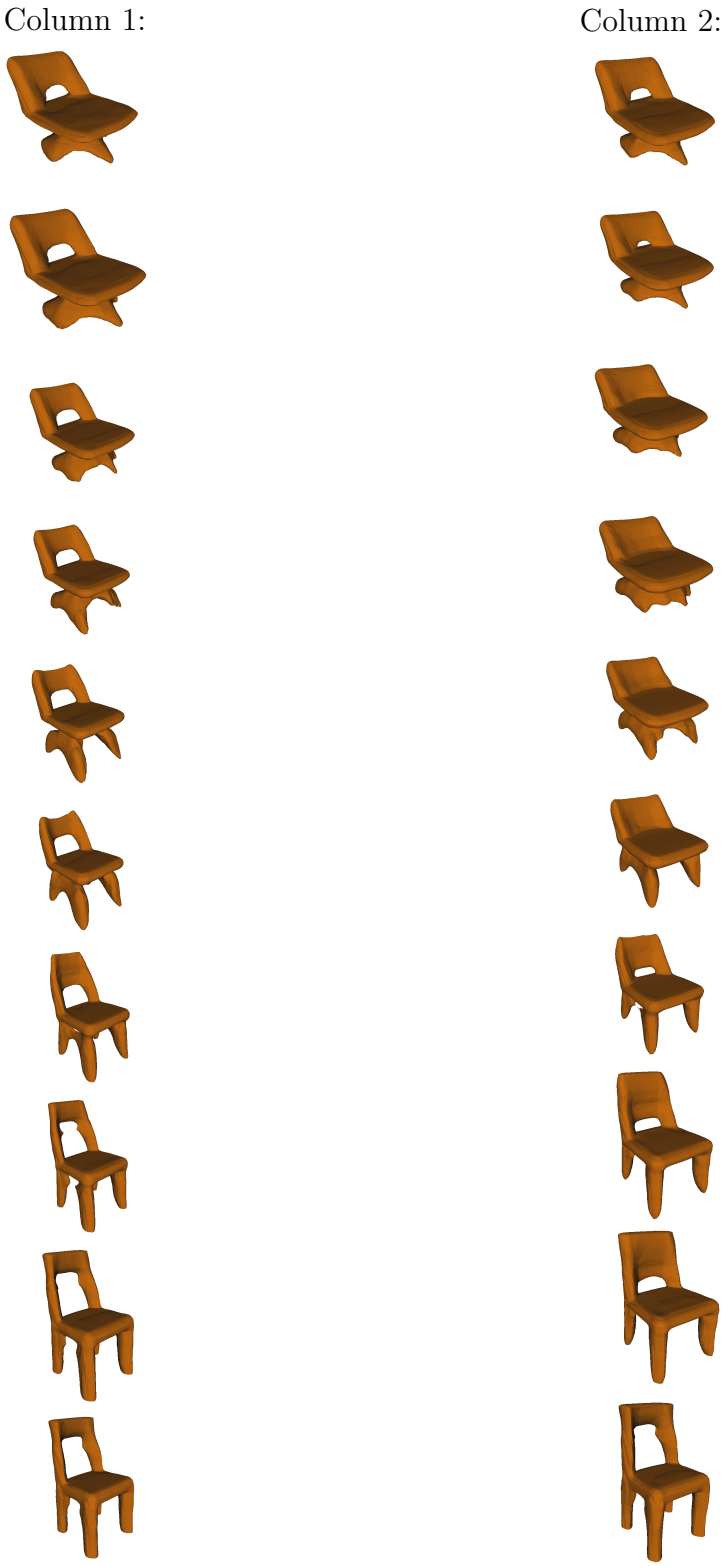

Figure 4.5: Comparison of interpolation in the latent space versus interpolation in the embedding. Columns 1 shows the gradual change from one chair to another when using 128D latent vector interpolation. Columns 2 shows the corresponding chairs reconstructed from 2D interpolation in the embedding.

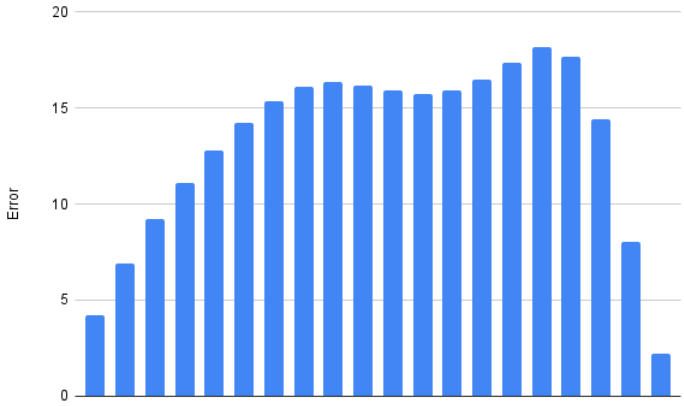

Figure 4.6: Euclidean distance between latent vectors interpolated via the embedding and vectors obtained with 12D8 latent vector interpolation.

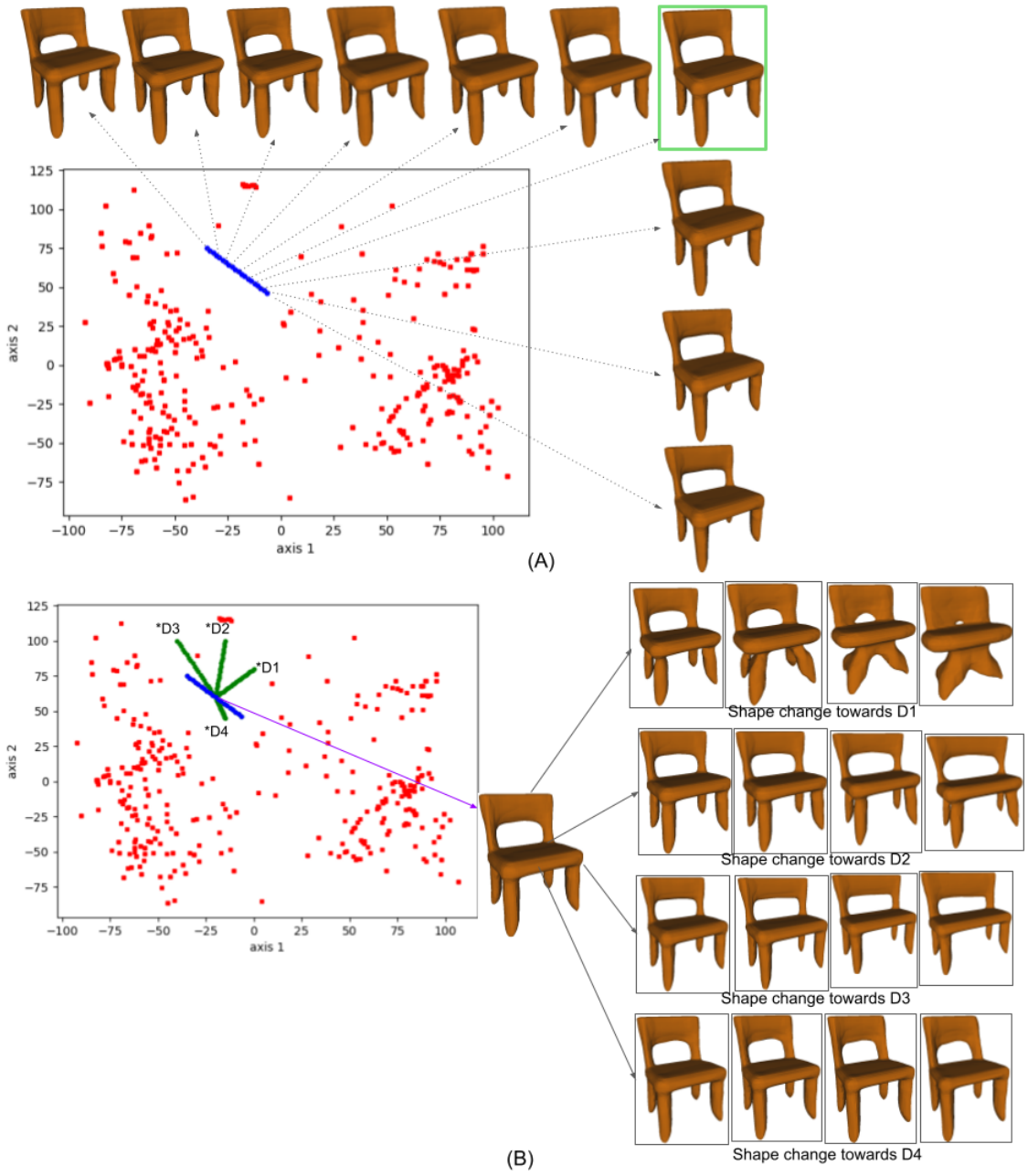

Figure 4.7: Multiple direction change to synthesize new shapes form latent space exploration. (A) The blue dots correspond to user-selected points. Their corresponding synthesized shapes are connected with the arrows. The shape in the green box is selected as the starting point for exploring in different directions. (B) The selected chair is changed by exploring four different directions. *D1 = direction 1, and similar for other directions. The arrows show the newly synthesized chairs in each direction.

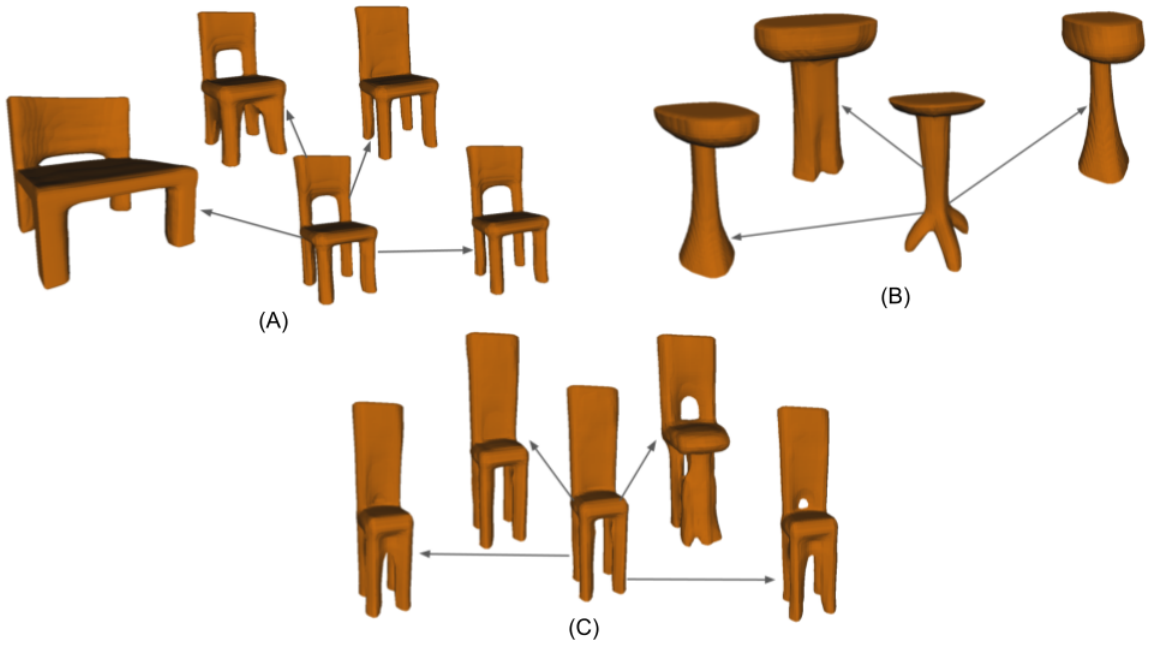

Figure 4.8: Neighborhood shape retrieval within small radial distance on the Isomap embedding of the latent vectors. (A) Query shape and retrieved 4 similar shapes from the dataset. (B) Query shape and 3 retrieved similar shapes. (C) Query shape and 4 retrieved shapes.

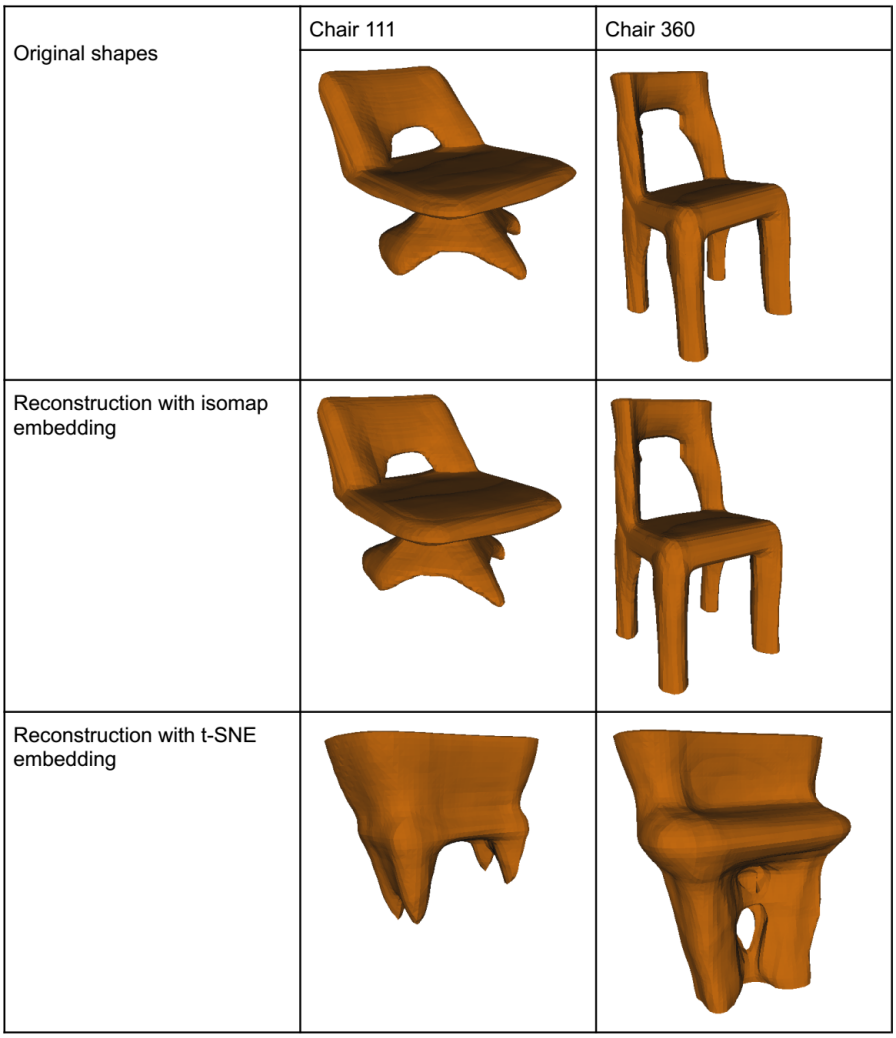

Figure 4.9: Comparison of Isomap and t-SNE embeddings for two chairs. The first row shows the original chairs. The second and third rows show the reconstructions obtained with inverse mapping applied to Isomap and t-SNE embeddings.

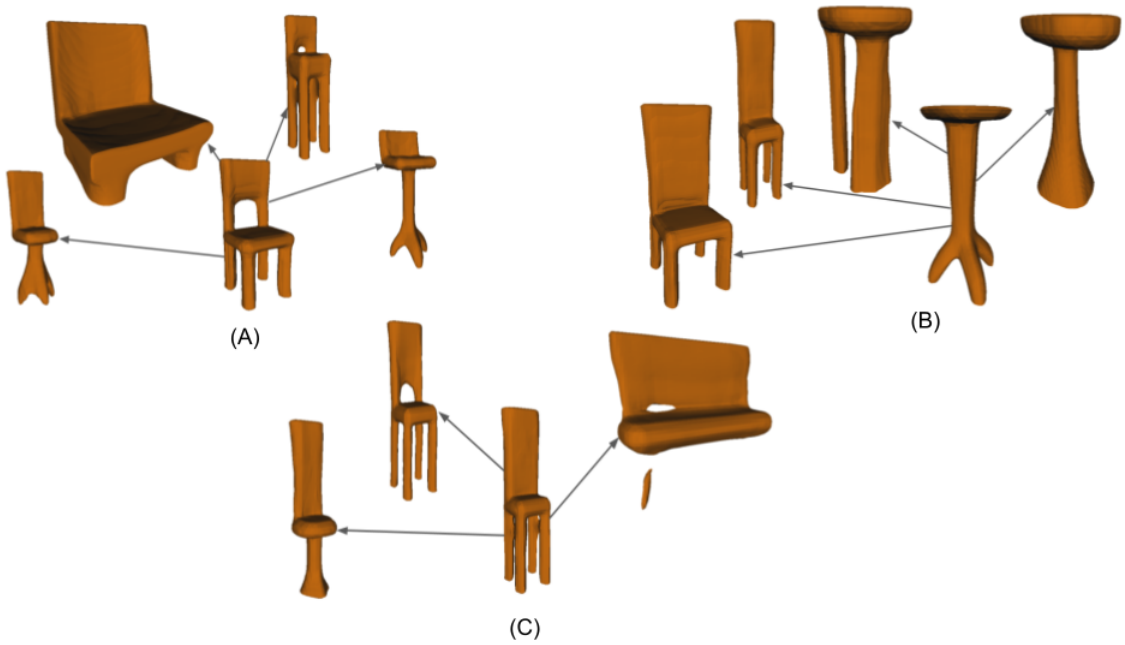

Figure 4.10: Neighborhood shape retrieval within a small radial distance of a query shape, performed on the t-SNE embedding of the latent space. (A) Query shape and four closest shapes retrieved from the dataset. (B) Query shape and three closest shapes retrieved. (C) Query shape and four retrieved shapes.

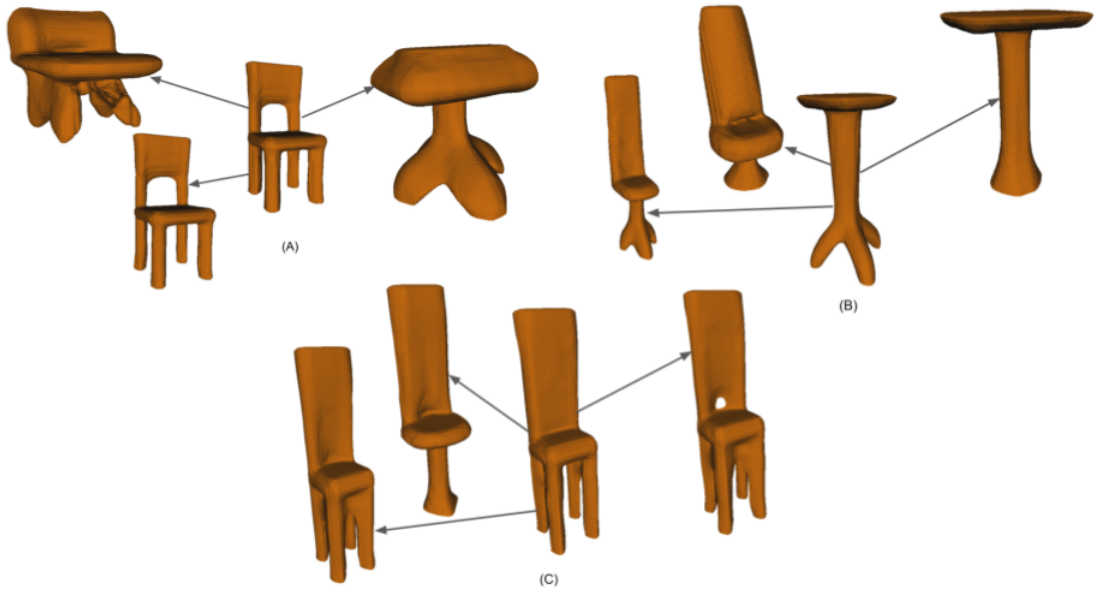

Figure 4.11: Neighborhood shape retrieval within a small radial distance of a query shape, performed on the PCA embedding of the latent space. (A) Query shape and three closest shapes retrieved from the dataset. (B) Query shape and three closest shapes retrieved. (C) Query shape and three retrieved shapes.

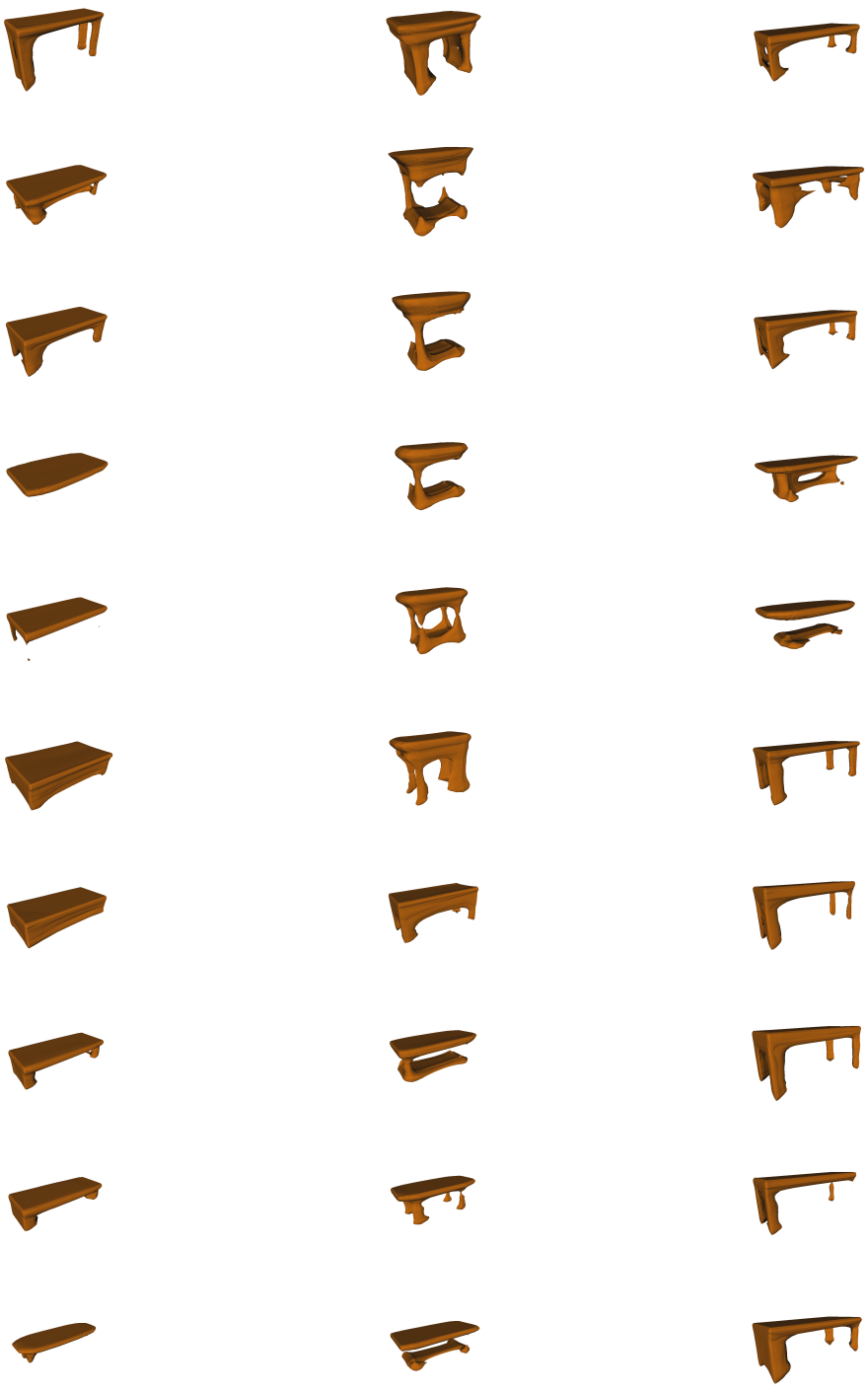

Figure 4.12: Latent space exploration result for the table dataset. Each column shows the gradual change of a shape due to navigation through the embedding.

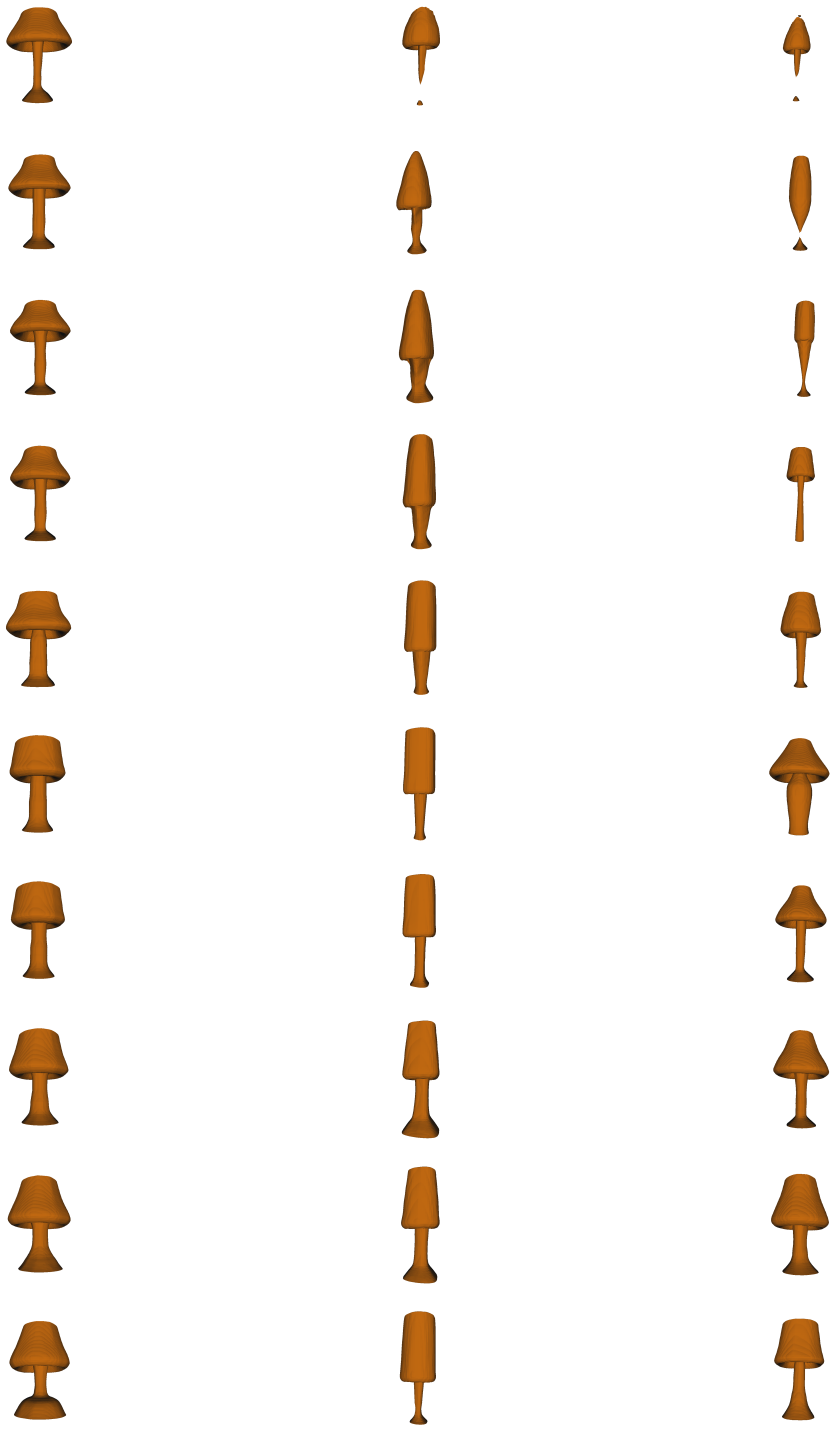

Figure 4.13: Latent space exploration result for the lamp dataset. Each column shows the gradual change of a shape due to navigation through the embedding.

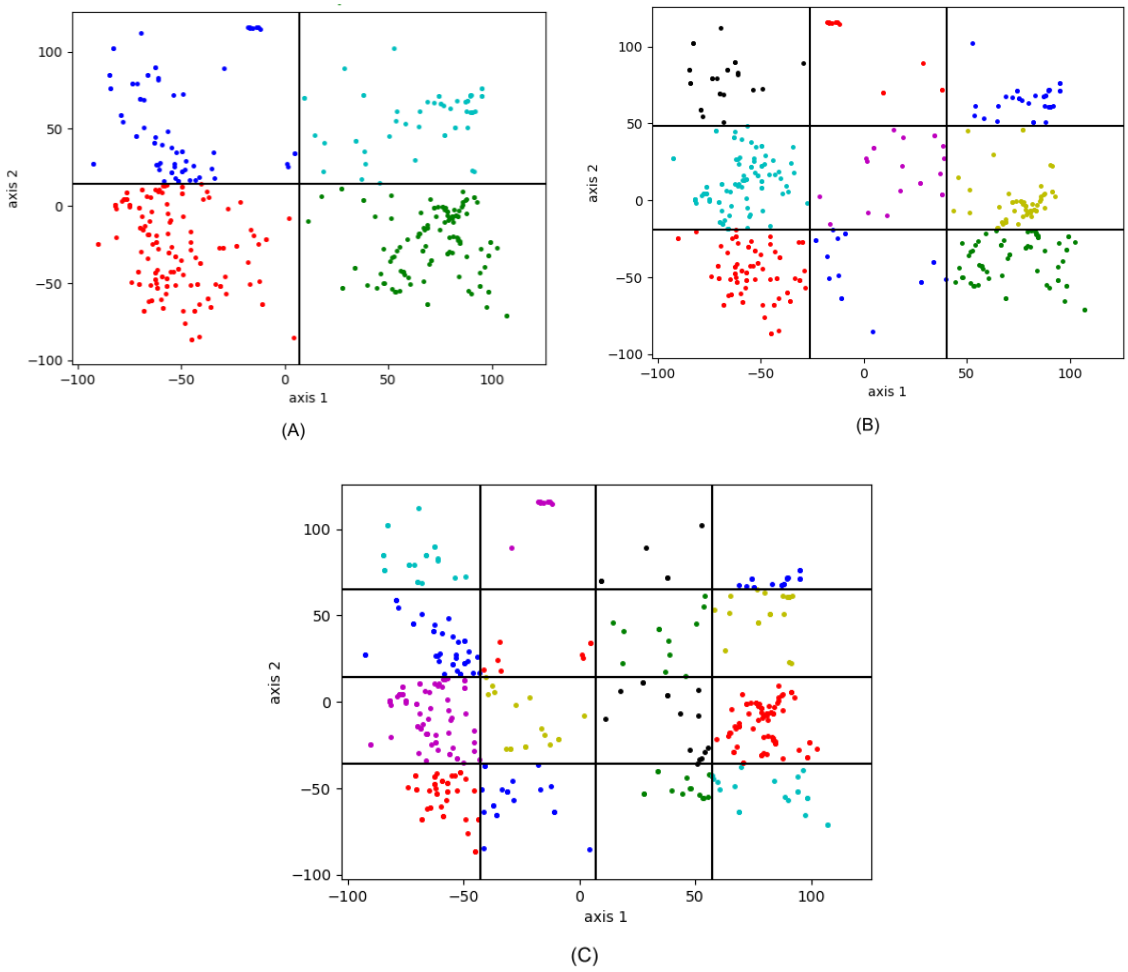

Figure 4.14: Parameter change analysis for an Isomap embedding divided into rectangular grids of different sizes. fig(A) Embedding divided into a 2×2 grid. (B) 3×3 grid. (C) 4×4 grid. Points separated by each cell are assigned different colors for clarity.

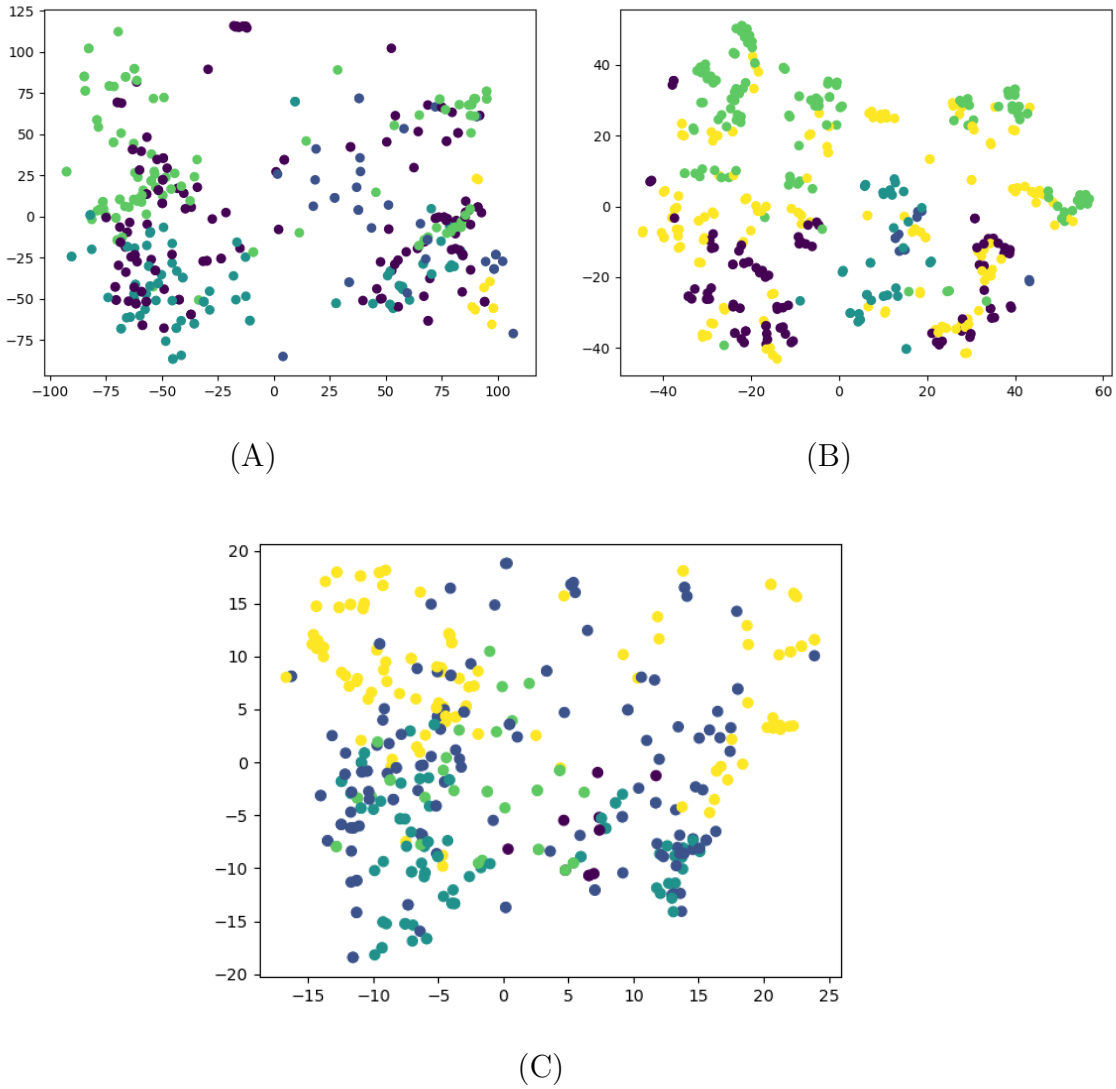

Figure 4.15: Clustering of the 2D embedding based on parametric similarity. (A) Parametric similarity based clustering on Isomap embedding. (B) Parametric similarity based clustering on t-SNE embedding. (C) Parametric similarity based clustering on PCA embedding.

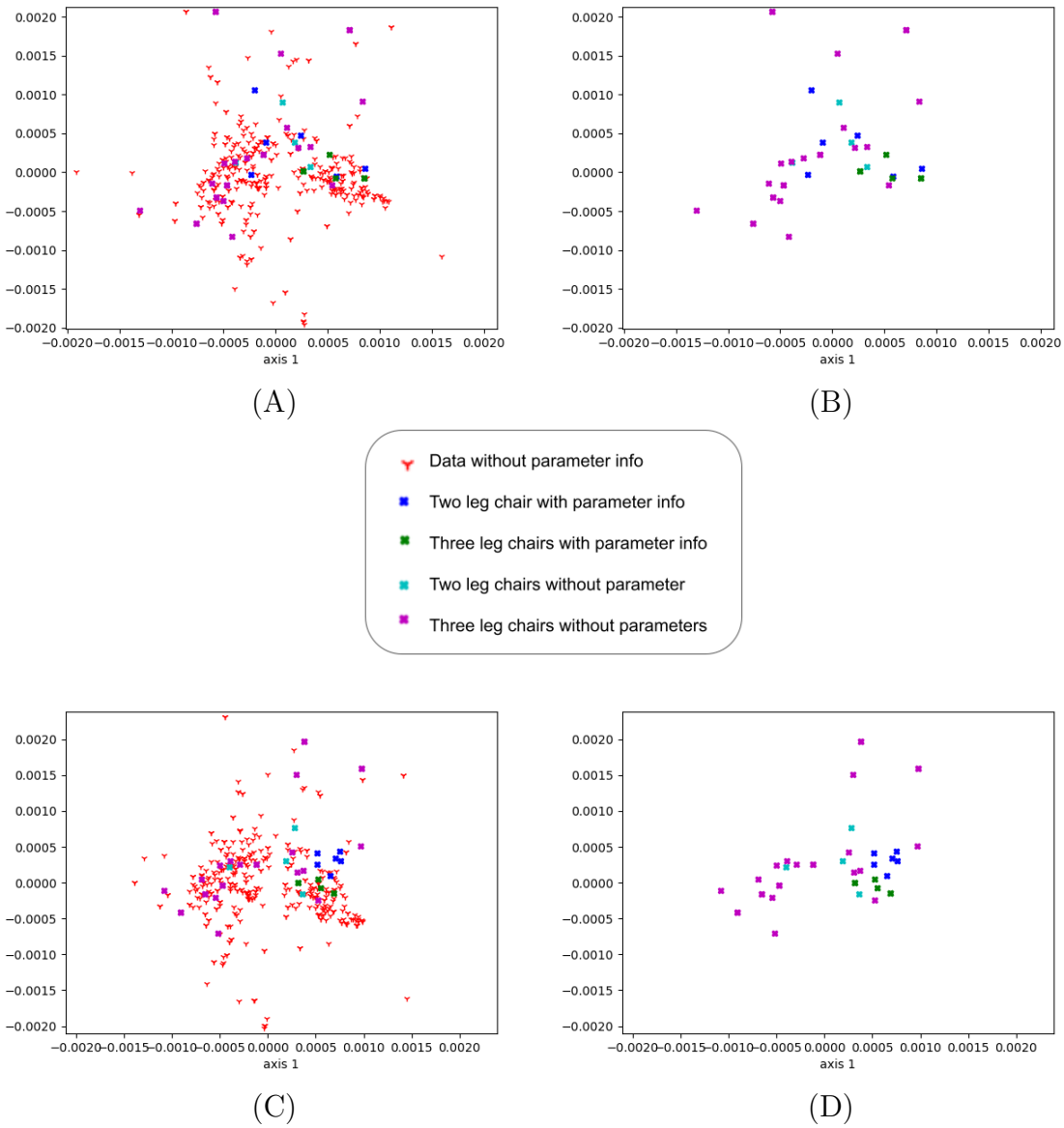

Figure 4.16: Example of constrained Isomap embedding. The blue dots are 6 two-leg chairs labeled by the user, while the green dots are 6 three-leg chairs labeled by the user. The cyan dots are the unlabeled two-leg chairs, while the magenta dots are the unlabeled three-leg chairs. The red dots are all the remaining chairs. (A) The 2D embedding of 400 chairs using Isomap reduction and no labels. (B) The portion of the plot showing only the two-leg and three-leg chairs. (C) The constrained 2D embedding of 400 chairs using the labeled data. (D) The portion of the plot showing only the two-leg and three-leg chairs in the constrained embedding.

Chapter 5

Conclusion

In this thesis, we introduced a method that allows a user to explore a latent space of 3D shapes by navigating an embedding of reduced dimensionality. The method allows the user to synthesize new shapes that are different from the existing shapes in the dataset, beyond just the interpolation of two existing shapes. We demonstrated our method on datasets of three different types of manufactured shapes, ranging from 50 to 400 shapes. Based on our experiments, we conclude that our method can work with relatively small datasets, in the order of hundreds of shapes. We also showed with a qualitative analysis comparing to existing methods for latent space exploration that our method offers advantages over the existing methods, such as providing more freedom to navigate the latent space and a better reference on the relationship between the embedding and the generated shapes. Our method allows the user to work on an embedding with any number of dimensions, although we focused on the 2D case to enable the visual exploration of the latent space.

Regarding the components of our method, we used the Isomap method for dimensionality reduction, but also compared to other methods such as t-SNE, concluding that Isomap works the best for our purpose. We introduced a method to evaluate the relationship between the Isomap embedding and specific shape parameters, to demonstrate that the Isomap embedding groups similar shapes together based on their characteristics. We also applied a constrained version of Isomap to explicitly group shapes in the embedding according to user-provided labels. In this manner, the user has a better reference for navigating the embedding. We evaluated the reconstruction error of the inverse mapping used by our method. The average error is in an acceptable range and the shapes synthesized for each category are usually complete and of good quality.

5.1 Limitations and future work

By analyzing our results, we have found that the system does not work well when the number of shapes in the input dataset is too small, such as the lamp dataset with 50 shapes. A dataset with a minimum of hundreds of shapes is needed. Moreover, we experimentally selected the value r that determines the Isomap neighborhood size (details in Section 4.4), and thus this value can differ based on the category of shapes in the dataset and the number of shapes. In the future, we plan to automate the selection of the neighborhood size according to a reconstruction error threshold.

We focused mostly on using our method with a 2D embedding. In the future, it would be interesting to use embeddings with more than two dimensions, and investigate manners of providing a reference to the user when navigating these higher-dimensional spaces, similarly to how the existing shapes in the dataset are shown as points in the 2D embedding.

Moreover, the method is implemented in Python and the code is not optimized. The method takes on average 5 seconds to produce a complete shape. Thus, in the future, we would like to optimize our implementation to enable real-time performance, with the synthesis taking only a second or less. Lastly, we did not conduct a user study to confirm that the advantages of our method provide a better exploration navigation in practice. We opted instead to demonstrate different use cases and evaluate the components of the method with quantitative experiments. However, a user study could be valuable to reveal other areas for improvement of the method.

List of References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas, "Representation learning and adversarial generation of 3d point clouds," *CoRR*, vol. abs/1707.02392, 2017. [Online]. Available: http://arxiv.org/abs/1707.02392

[2] I. Alhashim, H. Li, K. Xu, J. Cao, R. Ma, and H. Zhang, "Topology-varying 3d shape creation via structural blending," *ACM Trans. Graph.*, vol. 33, no. 4, jul 2014. [Online]. Available: https://doi.org/10.1145/2601097.2601102

[3] S. Ali and O. van Kaick, "Evaluation of latent space learning with procedurally-generated datasets of shapes," in *ICCV Workshop on Deep Learning for Geometric Computing*, 2021, pp. 2086–2094.

[4] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: Shape completion and animation of people," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 408–416, 2005.

[5] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 187–194.

[6] M. Bokeloh, M. Wand, H.-P. Seidel, and V. Koltun, "An algebraic model for parameterized shape editing," *ACM Trans. Graph.*, vol. 31, no. 4, jul 2012. [Online]. Available: https://doi.org/10.1145/2185520.2185574

[7] A. Brock, T. Lim, J. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," Dec. 2016, pp. 1–9, workshop contribution; Neural Inofrmation Processing Conference : 3D Deep Learning, NIPS ; Conference date: 05-12-2016 Through 10-12-2016. [Online]. Available: https://nips.cc/Conferences/2016

[8] A. Brunton, A. Salazar, T. Bolkart, and S. Wuhrer, "Review and comparative analysis of statistical shape spaces of 3d data," *Computer Vision and Image Understanding*, vol. 128, 09 2012.

[9] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 67–76. [Online]. Available: https://doi.org/10.1145/383259.383266

[10] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[11] S. Chaudhuri, E. Kalogerakis, S. Giguere, and T. Funkhouser, "Attribit: Content creation with semantic attributes," in *Proc. Symp. on User Interface Software and Technology*, 2013, pp. 193–202.

[12] S. Chaudhuri and V. Koltun, "Data-driven suggestions for creativity support in 3D modeling," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 183:1–10, 2010.

[13] K. Chen, C. Choy, M. Savva, A. Chang, T. Funkhouser, and S. Savarese, "Text2shape: Generating shapes from natural language by learning joint embeddings," 03 2018.

[14] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5932–5941.

[15] A. Dai, C. R. Qi, and M. Nießner, "Shape completion using 3d-encoder-predictor cnns and shape synthesis," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[16] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 2672–2680.

[18] Y. Gryaditskaya, F. Hähnlein, C. Liu, A. Sheffer, and A. Bousseau, "Lifting freehand concept sketches into 3d," *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3414685.3417851

[19] Y. Guan, T. Jahan, and O. van Kaick, "Generalized autoencoder for volumetric shape generation," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 1082–1088.

[20] Z. Hao, H. Averbuch-Elor, N. Snavely, and S. Belongie, "Dualsdf: Semantic shape manipulation using a two-level representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[21] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, "GANSpace: Discovering interpretable GAN controls," *CoRR*, vol. abs/2004.02546, 2020.

[22] H. Huang, E. Kalogerakis, E. Yumer, and R. Mech, "Shape synthesis from sketches via procedural models and convolutional networks," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 8, pp. 2003–2013, 2017.

[23] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A sketching interface for 3D freeform design," in *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 409–416.

[24] T. Jahan, Y. Guan, and O. van Kaick, "Semantics-guided latent space exploration for shape generation," *Computer Graphics Forum (Proc. Eurographics)*, p. to appear, 2021.

[25] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 1558–1566.

[26] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H. Seidel, "Differential coordinates for interactive mesh editing," in *Proc. Shape Modeling Applications*, 2004, pp. 181–190.

[27] J. Liu, F. Yu, and T. Funkhouser, "Interactive 3d modeling with a generative adversarial network," in *2017 International Conference on 3D Vision (3DV)*, 2017, pp. 126–134.

[28] L. Mi, T. He, C. F. Park, H. Wang, Y. Wang, and N. Shavit, "Revisiting latent-space interpolation via a quantitative evaluation framework," *CoRR*, vol. abs/2110.06421, 2021. [Online]. Available: https://arxiv.org/abs/2110.06421

[29] ——, "Revisiting latent-space interpolation via a quantitative evaluation framework," 2021.

[30] N. J. Mitra, M. Wand, H. Zhang, D. Cohen-Or, V. Kim, and Q.-X. Huang, "Structure-aware shape processing," in *ACM SIGGRAPH 2014 Courses*, ser. SIGGRAPH '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2614028.2615401

[31] N. D. Monnig, B. Fornberg, and F. G. Meyer, "Inverting nonlinear dimensionality reduction with scale-free radial basis function interpolation," *Applied and Computational Harmonic Analysis*, vol. 37, no. 1, pp. 162–170, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S106352031300095X

[32] C. Nash and C. K. I. Williams, "The shape variational autoencoder: A deep generative model of part-segmented 3d objects," *Comput. Graph. Forum*, vol. 36, no. 5, p. 1–12, Aug. 2017. [Online]. Available: https://doi.org/10.1111/cgf.13240

[33] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 165–174.

[34] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *Adv. Neural Inform. Process. Syst.*, vol. 30, 2017.

[35] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.290.5500.2323

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, D. E. Rumelhart and J. L. Mcclelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.

[37] N. Sedaghat and T. Brox, "Unsupervised generation of a viewpoint annotated car dataset from videos," in *Int. Conf. Comput. Vis.*, 2015, pp. 1314–1322.

[38] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986, p. 151–160. [Online]. Available: https://doi.org/10.1145/15922.15903

[39] J. O. Talton, D. Gibson, L. Yang, P. Hanrahan, and V. Koltun, "Exploratory modeling with collaborative design spaces," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–10, 2009.

[40] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, p. 2319, 2000.

[41] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[42] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, jul 2017. [Online]. Available: https://doi.org/10.1145/3072959.3073608

[43] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 496–503.

[44] Y. Wang, S. Asafi, O. van Kaick, H. Zhang, D. Cohen-Or, and B. Chen, "Active co-analysis of a set of shapes," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 165:1–165:10, 2012.

[45] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum, "MarrNet: 3D shape reconstruction via 2.5D sketches," in *Adv. Neural Inform. Process. Syst.*, 2017.

[46] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: a deep representation for volumetric shape modeling," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015.

[47] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu, "Learning descriptor networks for 3d shape synthesis and analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[48] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, pp. 40–51, 01 2007.

[49] Y.-J. Yuan, Y.-K. Lai, T. Wu, L. Gao, and L. Liu, "A revisit of shape editing techniques: from the geometric to the neural viewpoint," *J. Comput. Sci. Technol.*, vol. 36, pp. 520–554, 2021.

[50] M. E. Yumer, S. Chaudhuri, J. K. Hodgins, and L. B. Kara, "Semantic shape editing using deformation handles," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 1–12, 2015.

Appendix A

Dataset shape and quality

We present a sample of shapes from the datasets used in our experiments.

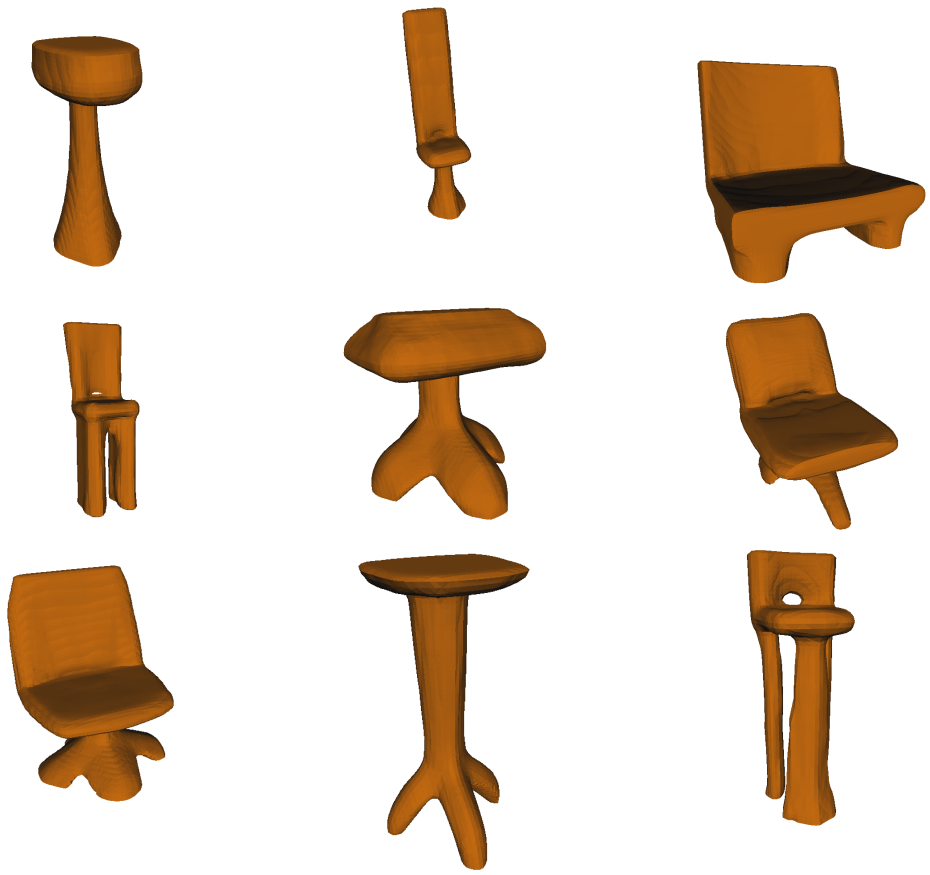

Figure A.1: Sample chairs from chair dataset. Number of legs varies from 1-4. Variation is also visible in chair back, seat shape, height of seat area, etc.

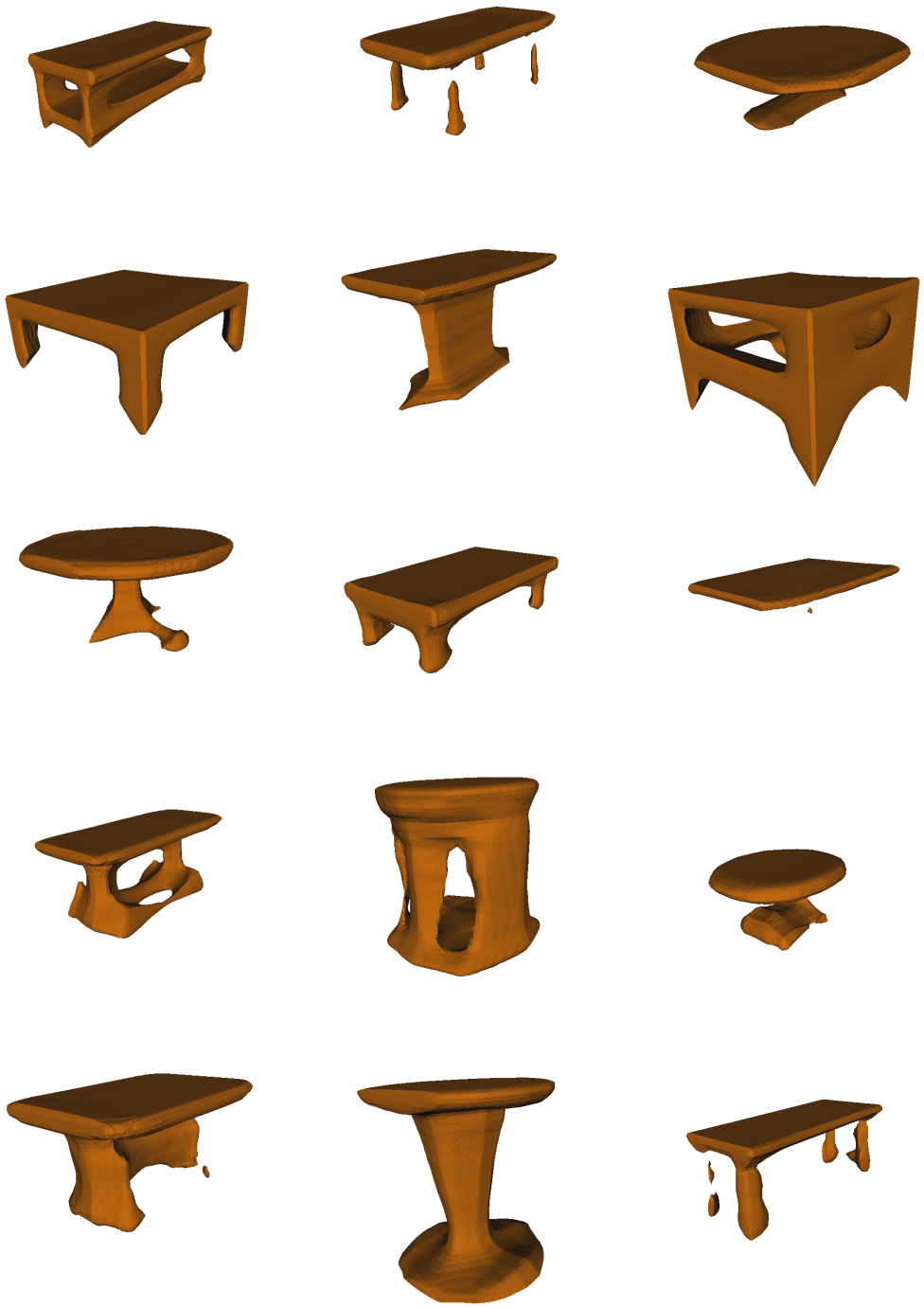

Figure A.2: Sample tables from table dataset. Shape quality is visibly worse than chair dataset. Many shapes are incomplete.

A.1 Implementation details

The encoder is implemented using tensorflow.keras. The decoder network is implemented using tensorflow. Both networks are tested with Python 3.5, TensorFlow 1.8.0, CUDA 9.1 and cuDNN 7.0 on Ubuntu 16.04. The encoder and decoder networks are modifications of the implementations of Guan et al. [19] and Chen et al. [14], respectively. All the other components of our method such as the latent space dimensionality reduction and exploration are implemented in python. Our isomap implementation reduces the dimensionality of the input dataset of a complex nonlinear manifold of high dimensional data while capturing the intrinsic geometry of the data in the embedding. The implementation of the inverse mapping function is the solution to a scattered interpolation problem. Inverse mapping defines a function that maps the low-dimensional points back to the original latent dimension in a reasonable manner.

Figure A.3: Sample lamps from lamp dataset. This dataset has fewer shapes that the other two. But they are mostly of good visual quality. Still it has few incomplete shapes (middle shape of last row)