mathematics

IMPACT FACTOR 2.4

CITESCORE 3.5

Article

A First Approach to Quantum Logical Shape Classification Framework

Alexander Köhler, Marvin Kahra and Michael Breuß

MDPI

https://doi.org/10.3390/math12111646

mathematics

MDPI

Article

A First Approach to Quantum Logical Shape Classification Framework

Alexander Köhler *, Marvin Kahra and Michael Breuß

Institute of Mathematics, Brandenburg University of Technology (BTU) Cottbus-Senftenberg, Platz der Deutschen Einheit 1, 03046 Cottbus, Germany; kahramar@b-tu.de (M.K.); breuss@b-tu.de (M.B.)
* Correspondence: koehlale@b-tu.de

Abstract: Quantum logic is a well-structured theory, which has recently received some attention because of its fundamental relation to quantum computing. However, the complex foundation of quantum logic borrowing concepts from different branches of mathematics as well as its peculiar settings have made it a non-trivial task to devise suitable applications. This article aims to propose for the first time an approach using quantum logic in image processing for shape classification. We show how to make use of the principal component analysis to realize quantum logical propositions. In this way, we are able to assign a concrete meaning to the rather abstract quantum logical concepts, and we are able to compute a probability measure from the principal components. For shape classification, we consider encrypting given point clouds of different objects by making use of specific distance histograms. This enables us to initiate the principal component analysis. Through experiments, we explore the possibility of distinguishing between different geometrical objects and discuss the results in terms of quantum logical interpretation.

Keywords: quantum logic; principal component analysis; shape classification; Hilbert spaces; eigenvectors

MSC: 81P99; 03G12

Citation: Köhler, A.; Kahra, M.; Breuß, M. A First Approach to Quantum Logical Shape Classification Framework. *Mathematics* **2024**, *12*, 1646. https://doi.org/10.3390/math12111646

Academic Editors: Jonathan Blackledge and David Carfì

Received: 19 February 2024
Revised: 30 April 2024
Accepted: 21 May 2024
Published: 24 May 2024

1. Introduction

Quantum logic [1–3] was developed as an approach to understanding the quantum physics structures of a logical system. The starting point for this was, above all, the formulation of the Schrödinger equation and Heisenberg's uncertainty principle. The latter, in particular, led to the conclusion that an uncertainty principle violates the distributive law in a logic-based framework. Over the last 70 years, various attempts that allow incorporating the uncertainty principle into logic have been developed from this, with the pioneers being Birkhoff and von Neumann, Reichenbach, and Mittelstaedt. In addition to the explanation of quantum mechanical phenomena, other areas of application for quantum logic have also emerged. The largest of these is quantum computing, where the underlying logic is featured in so-called quantum gates; see, for example, [4]. Other potential fields of applications have also been proposed, such as the evaluation of large datasets using contingency tables [5] or for quantum-inspired cognitive agents [6–8]. However, arguably because of the complex theoretical foundation of quantum logic and the peculiar settings described by it, until now, the range of applications has been limited.

The aim of this paper is to present the first approach using quantum logic for shape classification. In any case, our work is not to be confused with works related to quantum computation, in which quantum-based methods are used to accelerate existing algorithms. However, this does not refer to the use of quantum logic as a modeling framework but rather elaborates on the design and combination of quantum gates for quantum algorithms that are supposed to implement corresponding methods on quantum computers. For examples of this, see [9,10] for works on translating machine learning mechanisms into

a quantum computational approach. To the best of our knowledge, this article thereby represents the first attempt to make use of a quantum logical framework in image analysis.

In this paper, we base our study on probably the most popular interpretation of quantum logic by Birkhoff and von Neumann [1]. This, in turn, is based on the mathematical structures within quantum mechanics, namely Hilbert spaces and the associated operators. To do so leads us to the use of an orthomodular logic, which is a generalization of Boolean logic. Together, these form a projective geometry, which is one of the most important tools of quantum logic.

As an important computational component within our approach, we make use of principal component analysis (PCA). The PCA method was proposed by Karl Pearson in the year 1901 [11]. Later, in 1933, Harold Hotelling developed, independently of Pearson, an analogous method [12]. An overview of this method can be found in the book by Jolliffe [13]. The PCA is a statistical method to decouple meaningful information from high-dimensional data, reducing its dimensionality at the same time. This method is a powerful tool that is widely used in various fields, including machine learning [14,15], image processing [16,17], genetics [18,19], and finance [20,21]. As we discuss in detail in this article, the mechanism of dimension reduction in PCA appears to be a natural fit to quantum logic. One of the contributions of this work is exploring this connection.

Turning to computational ways to implement quantum logic, this paper represents, in several ways, a significant extension of an idea sketched by Wirsching [22]. In his work, he introduced a concept inspired by quantum logic that assigns to an incoming signal a similarity measure, making use of the Gram–Schmidt method.

Our novel contribution may be described as follows. We illuminate in detail the connection between quantum logic to the possible building blocks of an application-oriented framework. By doing this, we clarify in detail which properties of technical components are demanded for potential future applications. We consider a concrete application in the form of shape classification, which allows us to assign a concrete meaning to the rather abstract quantum logical concepts. The shape classification is a simple decision problem, with the inherent binary structure, that represents the basic assumption of every application of quantum logic. As a technical difference to [22], we employ the PCA instead of the Gram–Schmidt method, which allows a more natural explanation of a probability measure in accordance with quantum logic. The PCA also allows a much stronger interpretation of computed results. We want to point out that we will not provide any comparison to other shape classification frameworks.

The paper is structured into two main parts. The first part is presenting our model. Here, we introduce the three important concepts for our paper. We will provide insides of quantum logic, the principal component analysis, and how to connect these two topics with each other. With these concepts, we can apply quantum logic for a shape classification framework. The second part, of this paper, showcases our experiments, where we start to explain how we preprocess our data, fulfilling all the required properties derived from the theory part of this paper. Additionally, we want to verify our approach by presenting some experiments. And finally, we end this paper with the conclusions and possible future work.

2. Description of Our Model

This section is dedicated to introducing all the necessary details for the use of quantum logic for shape classification. First, we explain the concepts and operations in quantum logic. Thereafter, we build the bridge from quantum logic toward the application of shape classification. Then we briefly recall the principal component analysis, and we show how to construct a probability measure based on these concepts.

2.1. Quantum Logic

We begin by introducing the fundamental framework of quantum logic, which is in close relation to the algebraic foundations of complete lattices. It was developed as an algebraic tool to provide an axiomatic construct for quantum mechanics where conventional

approaches failed. The resulting axioms became known primarily through Birkhoff and von Neumann [1] and Piron [2] with his Geneva school's approach [23].

2.1.1. Fundamental Lattice Setting of Quantum Logic

The idea of quantum logic is to operate on a quantum mechanical system in the form of a special lattice, which we will introduce now step-by-step for the reader's convenience. We begin with the orthocomplemented lattice $\mathcal{L} = \big(\mathbb{H}(\Omega), \mathbf{0}, \wedge, \vee, \mathbf{1}, \cdot^{\perp}, \leq\big)$, where $\mathbf{0}, \mathbf{1}$ represent the minimal and maximal element, respectively, $\wedge$ is the conjunction or meet, $\vee$ the disjunction or join, $\cdot^{\perp}$ the orthocomplement and $\leq$ a partial order.

The orthocomplement is defined by the following three properties with the elements or propositions a, b:

$$\text{complement law:} \quad a \vee a^{\perp} = \mathbf{1} \quad \text{and} \quad a \wedge a^{\perp} = \mathbf{0}, \tag{1a}$$

$$\text{involution law:} \quad a^{\perp\perp} = a, \tag{1b}$$

$$\text{order-reversing:} \quad a \leq b \Longrightarrow b^{\perp} \leq a^{\perp}. \tag{1c}$$

This quantum mechanical system contains a set Ω of possible *states*, e.g., the elements of $\mathbb{R}^n$, $n \in \mathbb{N}$. These states contain the set of all properties that are relevant for a particular realization of the system. In the quantum mechanical setting, if one considers a physical system that consists of multiple subsystems, quantum physics allows certain mixed states, namely entangled states, which cause the difference between quantum theory and alternative classical models.

The no-longer reducible states are called *atoms*. Piron [2] stated the following axioms:

$$\text{P}: \text{ If } a_1 \leq a_2, \text{ then the sublattice generated by } a_1 \text{ and } a_2 \text{ is Boolean.} \tag{2}$$

and

$$\text{A}_1: \text{ For any } a \in \mathcal{L} \text{ with } a \neq 0, \text{ there is an atom } \tilde{a} \text{ with } 0 \leq \tilde{a} \leq a. \tag{3a}$$

$$\text{A}_2: \text{ If } \tilde{a} \text{ is an atom and } a \leq x \leq a \vee \tilde{a}, \text{ then } x \in \{a, a \vee \tilde{a}\}. \tag{3b}$$

Axiom P means that physical measurements, which correspond to propositions a_1, a_2 satisfying the relation $a_1 \leq a_2$, are compatible (see [22]).

2.1.2. Reasoning Behind the Projective Geometry of Quantum Logic

By the combination of Equations (1a)–(3b), we may arrive at Piron's result (see [2]), in which the propositions from $\mathcal{L}$ are in a one-to-one relation with the closed linear subspaces of a Hermitian vector space or pre-Hilbert space. Let us note that this result will be crucial to construct a computational approach relying on quantum logic.

In accordance with Piron's result, the mentioned Hermitian vector space or pre-Hilbert space has a non-degenerating Hermitian form $\langle \cdot \mid \cdot \rangle$, which allows the formation of the orthocomplementation $S^{\perp}$ of any subset S in the sense of an orthogonality relation concerning the Hermitian form. Here, a subset is described as closed if the involution law (1b) applies.

At this point, we will briefly summarize how the lattice operations affect the subspaces of the Hilbert space according to Birkhoff and von Neumann [1]:

- $H \wedge \tilde{H} := H \cap \tilde{H}, \quad H, \tilde{H} \in \mathbb{H}(\Omega)$,
- $H \vee \tilde{H} := H + \tilde{H}, \quad H, \tilde{H} \in \mathbb{H}(\Omega)$,
  where the $+$ represents the vector sum, such that the join or union of two subspaces is the smallest subspace encompassing both these subspaces,
- $H^{\perp}$ is the orthogonal complement of $H \in \mathbb{H}(\Omega)$, such that $v \perp w$ for all $v \in H$ and $w \in H^{\perp}$, $H \vee H^{\perp} = H + H^{\perp} = \mathbf{1} = \mathbb{H}$ and $H \wedge H^{\perp} = \mathbf{0} = \{0\}$, where $\{0\}$ is the singleton that only consists of the null vector,
- $H \leq \tilde{H}$ is equal to $H \subseteq \tilde{H}$ for all $H, \tilde{H} \in \mathbb{H}(\Omega)$.

The lattice of subspaces of a Hermitian vector space with the orthogonal complement operation is, in general, not distributive, i.e., $a \wedge (b \vee c) \neq (a \wedge b) \vee (a \wedge c)$. As an example for this statement, we consider the lattice of the subspaces of $\mathbb{F}_2^2 = \mathrm{GF}(2)^2 = \{(0,0),(0,1),(1,0),(1,1)\}$, which consists of the two-dimensional elements of $\{0,1\} \times \{0,1\}$, with the component-wise addition

$$0+0=0=1+1, \quad 0+1=1=1+0$$

and component-wise multiplication

$$0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0, \quad 1 \cdot 1 = 1.$$

To make it clear that we are talking about subspaces here, we write the ordered pairs in $[\,\cdot\,]$ notation. The following then applies:

$$[(1,1)] \wedge \Big([(1,0)] \vee [(0,1)]\Big) = [(1,1)] \cap \Big([(1,0)] + [(0,1)]\Big) = [(1,1)] \cap [(1,1)] = [(1,1)]$$

and at the same time

$$\begin{aligned}\Big([(1,1)] \wedge [(1,0)]\Big) \vee \Big([(1,1)] \wedge [(0,1)]\Big) &= \Big([(1,1)] \cap [(1,0)]\Big) + \Big([(1,1)] \cap [(0,1)]\Big) \\ &= [(0,0)] + [(0,0)] = [(0,0)].\end{aligned}$$

holds true. This is a contradiction to distributivity.

The non-distributivity is probably the best-known property of quantum logic. A classic physical example of this is Heisenberg's uncertainty principle, according to which it is not possible to determine the position and the momentum at the same time, as the measurement influences the system. To be precise, let there be three propositions a, b, c, which correspond to a momentum measurement a and a position measurement that are divided into a left interval b and a right interval c. Then, according to Heisenberg's uncertainty principle, the terms $a \wedge (b \vee c)$ and $(a \vee b) \wedge (a \vee c)$ could produce different results since in the second term, we always have a momentum and a position measurement at the same time. Therefore, this represents another contradiction to distributivity.

For this reason, we use a weakened form of distributivity in the form of the following definition of

$$\text{orthomodularity:} \quad a \leq b \Longrightarrow a \vee (a^{\perp} \wedge b) = b. \tag{4}$$

Specifically, there is a connection between the orthomodularity and the Hilbert space, which was proven by Solér [24] and states that the pre-Hilbert space is complete if and only if the propositions a, b satisfy the orthomodular law (4). Because of this orthomodularity property, we will use so-called *orthomodular lattices* in the following. This leads to the main result that Birkhoff and von Neumann [1] as well as Piron [2] have proven in the form of the aforementioned connection of the lattice $\mathcal{L}$ and the Hilbert space.

2.1.3. Projective Geometry of Quantum Logic

After having introduced some basic algebraic concepts of quantum logic, we will introduce the axiomatic theory presented by von Neumann for the remainder of this section, leading to the foundation of a projective geometry. Our exposition of this is based on the book by Chiara et al. [3].

As for a beginning, let us point out that the quantum logical states have associated *state vectors* in terms of wave functions that form, in general, an infinite-dimensional complex Hilbert space $\mathbb{H}$. The wave functions are the reason the corresponding Hilbert space is called *phase space*. Thereby, a distinction is made between the *pure states*, which are unit vectors of the Hilbert space, and the already mentioned *mixed states*, which are weighted combinations of pure states that will be represented by density operators ρ of the Hilbert space. However, it has already been shown that this can be realized with (not necessarily

complex) finite-dimensional spaces of at least rank four, see for example, [2]. This means for the case $\Omega = \mathbb{R}^n$ that the elements of $\mathbb{H}(\Omega)$ are the vector subspaces of $\mathbb{R}^n$.

In the following, we will replace $\mathbb{H}(\Omega)$ with the set $\mathcal{C}(\mathbb{H})$ of closed subspaces of $\mathbb{H}$. Each pure state or unit vector ψ enables a projection $\mathcal{P}_{[\psi]}$, which is the projection onto the one-dimensional closed subspace $[\psi] \in \mathcal{C}(\mathbb{H})$ corresponding to ψ. However, as for the mixed states, it should be noted that not every possible density operator can be represented by a projection $\mathcal{P}_{[\psi]}$.

The mentioned one-to-one connection by Piron's result, between the state space and the phase space, relies on the fact that one can map one-dimensional subspaces of the Hilbert space into one point (atom) and the linear subspaces of the Hilbert space onto linear sets, i.e., straight lines through at least two points. Consequently, we have a one-to-one correspondence between the set $\Pi(\mathbb{H})$ of all projection operators and the set $\mathcal{C}(\mathbb{H})$ of closed subspaces of $\mathbb{H}$. This leads to a projection from the phase space into the state space since the projective space $\mathcal{C}(\mathbb{H})$ consists of the states of the system. The resulting projective geometry is the main reason why quantum logic is used in general.

2.1.4. Construction of the Projection Operators and the Probability Measure

The aforementioned Hilbert space formalism allows us to use corresponding multidimensional models and assign probabilities to them. For this purpose, we consider the variables $A_1, \ldots, A_n$, $n \in \mathbb{N}$, which take on one of the finitely many values $\lambda_1, \ldots, \lambda_m$, $m \in \mathbb{N}$, in a measurement that defines the state of the system. Thereby, A_i will be called *observables* and the output of the measurement of an observable leads to an event E, which we will interpret as a projection.

As for the relation between a physical measurement and its mathematical realization, this means that every measurement will be represented by a self-adjoint operator on the Hilbert space, which one refers to as observable. The eigenvectors of such an operator form an orthonormal basis for the Hilbert space, and each possible result of this measurement corresponds to one of the vectors that form the basis. The individual events are subspaces of the Hilbert space, and we associate each subspace (or the corresponding event E) with a projector $\mathcal{P}_E$, which projects vectors into the corresponding subspace and fulfills $\mathcal{P}_E = \mathcal{P}_E^2 = \mathcal{P}_E^H$, where $\mathcal{P}_E^H$ represents the Hermitian transpose of $\mathcal{P}_E$.

This projector allows us to assign an expected value to an observable, i.e., that the observable is in a state, which is represented by the pure state ψ. That means, if ψ is any unit vector, we can define a probability measure $\rho_{[\psi]}$ as

$$\begin{aligned} \rho_{[\psi]}(\mathcal{P}_E) := \langle \mathcal{P}_E\psi \mid \psi\rangle = \psi^\intercal \mathcal{P}_E^\intercal \psi = \psi^\intercal \mathcal{P}_E \psi = \psi^\intercal \mathcal{P}_E^2 \psi = \psi^\intercal \mathcal{P}_E^\intercal \mathcal{P}_E \psi \\ = \langle \mathcal{P}_E\psi \mid \mathcal{P}_E\psi\rangle = \|\mathcal{P}_E\psi\|^2. \end{aligned} \tag{5}$$

For practical applications of quantum logic, it appears to be essential to create appropriate projectors. So far, we have only indicated some properties that they must fulfill, but not how to construct one. We will now do this as a first step for projectors on pure states. If we explain the projection operator on the one-dimensional subspace $[\psi]$ belonging to the pure state ψ as $\mathcal{P}_{[\psi]}(x) = \mid \psi\rangle\langle\psi \mid x\rangle = \psi\psi^\intercal x$ for all $x \in \mathbb{H}$, we obtain the notation

$$\rho_{[\psi]}(\mathcal{P}_E) = \langle \mathcal{P}_E\psi \mid \psi\rangle = \sum_{i,j} p_{ij}\psi_j\psi_i = \mathrm{Tr}(\mathcal{P}_E\psi\psi^\intercal) = \mathrm{Tr}\Big(\mathcal{P}_E\mathcal{P}_{[\psi]}\Big), \tag{6}$$

where $\mathcal{P}_E = (p_{ij})$, $\psi = (\psi_1, \psi_2, \ldots)^\intercal$ and Tr is the trace functional, which is more common in quantum mechanics. This means $\mathcal{P}_E$ will have value 1 in the state ψ or, in other words, we have the certain verification of an event that belongs to an arbitrarily closed subspace E by a pure state ψ iff ψ is an element of $E \in \mathcal{C}(\mathbb{H})$:

$$\rho_{[\psi]}(\mathcal{P}_E) = 1 \text{ iff } \psi \in E \text{ iff } \mathcal{P}_E\psi = \psi. \tag{7}$$

This representation can also be extended to mixed states. To achieve this, we consider a mixed state as a superposition of several pure states ψ_i and can thus explain the density operator δ mentioned above as

$$\rho = \sum_i t_i \mathcal{P}_{[\psi_i]}, \quad t_i \in [0,1], \quad \sum_i t_i = 1. \tag{8}$$

This allows us to specify the corresponding probability measure as a convex combination:

$$\begin{aligned} &p(\mathcal{P}_E) = \sum_i t_i p_{[\psi_i]}(\mathcal{P}_E) = \sum_i t_i \operatorname{Tr}\left(\mathcal{P}_E \mathcal{P}_{[\psi_i]}\right) = \operatorname{Tr}\left(\sum_i t_i \mathcal{P}_E \mathcal{P}_{[\psi_i]}\right) = \operatorname{Tr}(\rho \mathcal{P}_E), \\ &t_i \in [0,1], \quad \sum_i t_i = 1, \end{aligned} \tag{9}$$

which is the usual way of representing the probability that the system is in a mixed state ρ and fulfills an event E by the projection $\mathcal{P}_E$ according to the Born rule. In particular, Gleason [25] showed that this representation is the only one that allows probabilities to be assigned to a subspace in the sense of a positive measure for dimensions greater than two.

2.2. Towards Application of Quantum Logic for Shape Classification

So far, we have looked at the basic properties and some conclusions in quantum logic. Now we want to connect these to an application and show how to make use of quantum logic in a shape classification framework as exemplified.

To make this connection more precise, we explain L in a symbolic sense as the collection of all properties of our system $\mathcal{S}$, or in other words, as a collection of "yes" or "no" experiments. A distinction is not necessary in this sense, since each property can be queried with a yes or no question and vice versa. If we transfer this to a quantum system, L corresponds to the collection of all closed subspaces $\mathcal{C}(\mathbb{H})$ of our Hilbert space $\mathbb{H}$, and we say that such an experiment is "true" if the state vector lies in the corresponding closed subspace E of $\mathbb{H}$. In this case, we say "E is true" and we only deny it with certainty if the state vector lies in $E^{\perp}$.

2.2.1. Quantum Logical Meaning of a Shape

By transferring this idea to shapes, we may understand a state vector as a vector that contains certain predetermined information about the shapes under consideration. Since our goal is to decide whether a given shape S corresponds to a certain shape class or not, we can also identify this question as a "yes" or "no" question.

Consequently, we would answer "yes" if the shape under consideration S corresponds to a specific shape class and if the corresponding state vector lies in a closed subspace of the Hilbert space. This closed subspace would then have to be associated with this specific shape class in this sense.

2.2.2. Idea Behind Shape Classification with Quantum Logic

In this context, the quantum logical setting offers the advantage that we can not only say whether a shape S belongs to a certain class or not, but even with what probability. In this sense, it should be noted that S could have any shape regardless of the experiment we conduct. However, if it is a type of shape that we have used to span the Hilbert space, then S would correspond to a pure state, and we could use Equation (5). Otherwise, if S was a superposition of the types of shapes we used to construct $\mathbb{H}$, we would use Equation (9). At this point, it should also be emphasized that when we discuss the Hilbert space spanned by these types of shapes, we mean a subspace of the Hilbert space of all possible shapes, which is itself a Hilbert space to apply quantum logic to.

2.2.3. Necessary Considerations for Shape Classification

If we follow this train of thought further, the underlying projection operator would have to map a vector onto a one-dimensional subspace belonging to the type of geometric shape under consideration. This means that we need a separate operator for each type of shape, i.e., triangles, squares, etc., with which we can decide to what extent the state vector under consideration corresponds to this type of shape.

In order to express the shape classification task quantum logically, we therefore need a Hilbert space of shapes on which we can work. We have to span this space with axes that we associate with certain types of shapes, and to obtain a probability for the classification, we have to compare the considered shape S with the representatives of these axes.

To achieve this, we first have to convert all shapes into normalized vectors of the same length, which contain enough information about these spans to make meaningful decisions. We will divide these vectors into directions that have the greatest influence, which we realize here by using the principal component analysis. These should then be the directions that contribute significantly to categorizing a shape as a triangle, for example. We, therefore, use them to construct our projection operators. We will devote the rest of this section to realizing this prepared setting.

2.3. Summary of Principal Component Analysis (PCA)

The PCA is a statistical technique to reduce the dimension of a large dataset while preserving most of the contained information. It maps a large dataset into a smaller number of new variables called principal components.

The first principal component belongs to the direction where the dataset has the highest variance, and the second principal component belongs to the direction with the second-highest variance, and so on. All directions produced this way are orthogonal to each other, or in terms of quantum mechanics, they are uncorrelated.

Let us recall the PCA method step-by-step. Assume a given dataset consisting of $r \in \mathbb{N}$ random vectors $X \in \mathbb{R}^k$ of length k. The r random vectors will be merged into a matrix $\mathbf{X} \in \mathbb{R}^{k \times r}$

$$\mathbf{X} = (X_1, X_2, \ldots, X_r) \tag{10}$$

The first step in the PCA method is to center the data. Therefore, we calculate the mean value for each row of $\mathbf{X}$ and obtain the mean value vector $\mu \in \mathbb{R}^k$. Thereafter, we can subtract μ from each random vector X in $\mathbf{X}$. We will write the subtraction as $\mathbf{X} - \mu$.

Second, we compute the *covariance matrix* $\mathbf{C} := \mathbf{C_{XX}}$ via

$$\mathbf{C} = \mathrm{cov}(\mathbf{X}, \mathbf{X}) = \frac{1}{k-1}(\mathbf{X} - \mu)(\mathbf{X} - \mu)^\intercal \in \mathbb{R}^{k \times k} \tag{11}$$

As a third step, we calculate the spectral decomposition of the matrix $\mathbf{C}$:

$$\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\intercal, \tag{12}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k) \in \mathbb{R}^{k \times k}$ is the matrix of the sorted $\lambda_1 > \lambda_2 > \ldots > \lambda_k \geq 0$ eigenvalues. The matrix $\mathbf{V} = (V_1, V_2, \ldots, V_k)^\intercal \in \mathbb{R}^{k \times k}$ contains the normalized eigenvectors V_i with $\|V_i\| = 1$ of $\mathbf{C}$ belonging to eigenvalue λ_i, denoting the corresponding eigenspace. The vectors V_i will act as the directions mentioned at the beginning of this section.

The fourth and last step will be the calculation of the principal components Y of an already-centered random vector X

$$Y_i = V_i^\intercal X = \langle V_i \mid X \rangle \quad \text{or} \quad Y = \mathbf{V}^\intercal X = \sum_i e_i \langle V_i \mid X \rangle \tag{13}$$

with Y_i the ith principal component, e_i the ith unit vector, and $\langle \cdot \mid \cdot \rangle$ the dot product in a Hilbert space.

A useful property of the principal components is that the variance of the ith component will be the ith eigenvalue, λ_i

$$\operatorname{var}(Y_i) = \lambda_i \tag{14}$$

This will ensure that the first principal component has the largest variation since the eigenvalues are ordered. One may interpret the ith principal component Y_i as the share of X in the direction V_i.

2.4. Classification by Quantum Logic

In this section, we want to build the bridge between the probability measure of the quantum logic and the principal components from the previous section. Before we can connect these two components, we want to expand the notation of the principal components.

2.4.1. Expanding the Notation

Consider a set of geometrical objects O with the index set G. The set O consists of the geometrical shape classes O_s and $s \in G$ and will be used to store the various classes we want to classify. In this context, $s, t \in G$ refers to two different shape classes if $s \neq t$ and refers to the same shape class if $s = t$.

By using only random vectors X^s of shape class O_s with $s \in G$ for the PCA formalism, we obtain the eigenspace $\mathbf{V}^s$ and mean vector μ_s for this specific shape class.

In Figure 1, we illustrate the obtained eigenspaces for a triangle and square. We make use of the geometrical object sets O_s with $s \in G = \{\triangle, \square\}$ containing only triangles and squares. In this sense, $\mathbf{V}^{\triangle}$ is the eigenspace of the triangle and $\mathbf{V}^{\square}$ belongs to the square shape class. Analogously, we depict the mean vector μ.

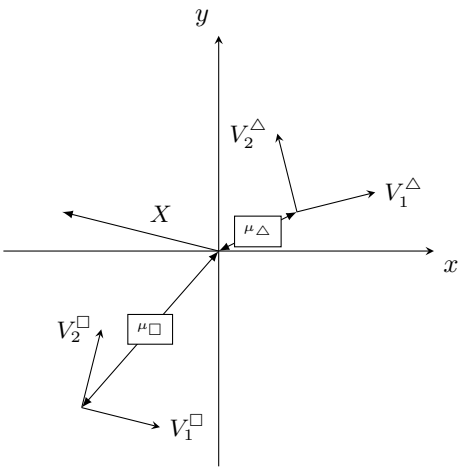

Figure 1. A visualization of made-up results of the principal component analysis. This two-dimensional example shows the eigenvectors V_1^s, V_2^s and mean vectors μ_s for two different shapes from the sets O_s for $s \in G = \{\triangle, \square\}$, namely triangle and square. Additionally, we show a random vector X.

We will adapt the notation even further. In Equation (13), we want to make sure that we can distinguish between the shape class O_t that provides a random vector X^t and the eigenspace $\mathbf{V}^s$ of another shape class O_s, for $s, t \in G$. This will enhance the notation of the principal component vector.

$$\prescript{t}{s}{Y} = \mathbf{V}^{s\mathsf{T}}(X^t - \mu_s) = \left\langle \mathbf{V}^s \mid X^t - \mu_s \right\rangle, \qquad s, t \in G \tag{15}$$

As an example, $\prescript{\triangle}{\square}{Y}$ will describe the principal components obtained from a random vector from the triangle shape class $X^{\triangle}$ and the square shape class eigenspace $\mathbf{V}^{\square}$.

2.4.2. Probability Measure Constructed by Quantum Logic

The theory of quantum logic, in particular, Equation (5), will build the foundation of the presented classification approach. This formula allows us to construct a probability measure $p_{[\psi]}(\mathcal{P}_E)$ for any unit vector ψ and projection operator $\mathcal{P}$.

In our setting, the random vector X of an arbitrary shape class acts as the unit vector ψ. In Section 2.2, we introduced a projection operator for pure states and a density operator for mixed states that consist of pure states. The eigenvectors V_i, from the PCA, will take the role of the pure states. Hence, we need to show that a projection operator $\mathcal{P}$ obtained from the eigenvectors, fulfills the conditions ($\mathcal{P}$ is Hermitian and idempotent) presented in Section 2.1 for a suitable projection operator.

We construct the operator $\mathcal{P}$ via

$$\mathcal{P} = \sum_{i=1}^{k_0} |V_i\rangle\langle V_i| \,, \quad k_0 \in [1,k]. \tag{16}$$

The resulting matrix is symmetric and real-valued, i.e., the condition $\mathcal{P} = \mathcal{P}^H$ is fulfilled for this operator. To prove the idempotence, we calculate

$$\mathcal{P}^2 = \mathcal{P}\mathcal{P}^\intercal = \sum_{i,j=1}^{k_0} |V_i\rangle \underbrace{\langle V_i \mid V_j\rangle}_{=\delta_{ij}} \langle V_j| \;= \sum_{i,j=1}^{k_0} \delta_{ij} |V_i\rangle\langle V_j| \;= \sum_{i=1}^{k_0} |V_i\rangle\langle V_i| \;= \mathcal{P}, \quad k_0 \in [1,k].$$

As such, our $\mathcal{P}$ fulfills all the requirements of a projection operator, and we can now use it to calculate a probability measure $p_{[X]}$ for a pure state or a normalized random vector, X.

$$\begin{aligned} p_{[X]}(\mathcal{P}) &= \langle \mathcal{P}X \mid X\rangle = \sum_{i=1}^{k_0} \langle X \mid V_i\rangle\langle V_i \mid X\rangle = \sum_{i=1}^{k_0} X^\intercal V_i V_i^\intercal X = \sum_{i=1}^{k_0} \left(V_i^\intercal X\right)^\intercal V_i^\intercal X \\ &= \sum_{i=1}^{k_0} Y_i^\intercal Y_i = \sum_{i=1}^{k_0} \|Y_i\|^2, \quad k_0 \in [1,k], \quad \|X\| = 1 \end{aligned} \tag{17}$$

This formula allows us to compute the probability measure via the principal components. Therefore, we can check if a normalized random vector X belongs to a certain shape class O_s with $s \in G$ by computing the principal components ${}_sY$ according to (15).

Since the maximal dimension of our constructed Hilbert space $\mathbb{H}$ is k, we will face some issues that will be addressed now. As a recall of Equation (7), setting $k_0 = k$ will result in a probability measure of one, since we make use of all principal components. As we mentioned at the end of Section 2.3, we can interpret the principal component Y_i as the share of X onto the direction V_i. Using $k_0 = k$ would lead to the fact that we make use of k orthogonal directions in a k-dimensional Hilbert space. Hence, the eigenvectors will span the complete space, regardless of the considered shape class. Computing the probability measure $p_{[X]}(\mathcal{P}_s)$ of a random vector X belonging to shape class O_s with $s \in G$ will lead to a value of one. To avoid this issue, we changed the condition $k_0 = [1,k]$ to $k_0 = [1,k)$.

In Figure 1, we visualize this issue. There we have a two-dimensional space with two results from the PCA, namely the eigenvector spaces from a triangle and a square shape class, respectively.

Considering the projection of the now normalized random vector X onto the first eigenvectors of the triangle and square, we can argue that the probability that X will be labeled as a member of the square shape class is much higher than the probability of being labeled as a triangle, since the share of X onto $V_1^{\square}$ is bigger than the share onto $V_1^{\triangle}$.

If we used both eigenvectors for each shape and computed the probability measure, we would obtain one for both shape classes since each eigenvector pair is a suitable description of the two-dimensional plane; therefore, the vector X is fully described by both pairs.

In the end, we will rewrite Equation (17) to be fit for the classification and consider these remarks:

$$p_{[X^t]}(\mathcal{P}_s) = \sum_{l=1}^{k_0} \left\| {}^t_s Y_l \right\|^2, \quad k_0 \in [1,k), \quad \|X\| = 1, \quad s,t \in G \tag{18}$$

Finally, we expect that our approach will lead to relatively high probability measures for the case $s = t$ in ${}^s_t Y$ since the first principal components should have greater values than the case $s \neq t$ for $s, t \in G$.

2.4.3. Quantum Logical Interpretation

A quantum logical interpretation of the combination would be that each of the eigenvectors produced from the PCA will represent an assertion about a specific shape class. We will not know the exact wording of these assertions, but we know two things.

The first thing is that the initial assertion should be the most controversial since the principal component Y_1 varies the most; see (14). Adding more assertions, i.e., increasing k_0, will lead to more and more specific perceptions of a shape. As the variation in the last assertion is small or negligible, we can safely disregard it.

Second, using all possible assertions, we should be able to confidently tell whether a random vector X belongs to a specific shape class or not. And, with (18), we can also assign a percentage value to this "belonging".

3. The Experiment and the Preparation

We would like to point out, again, that one of the contributions of this paper is the exposition of the connection between PCA and quantum logic. In this section, we will dedicate ourselves to having a closer look at a possible way to convert our data into a format that can utilize this connection.

To be more precise, we will now explain our data format and then move on to illustrate the preprocessing to generate the random vectors X from our shape dataset. Then, we will discuss a method of shape classification and provide an algorithmic pipeline. At last, we will show and discuss the produced results.

3.1. Shape Data

For the start, we want to briefly explain the concept of a shape as we employ it. A shape S is considered to be realized by a closed curve in a two-dimensional space, $S \subset \mathbb{R}^2$. Such a curve is established as a point cloud $P \subset \mathbb{R}^2$ with $n \in \mathbb{N}$ points $(x_i, y_i) = p_i \in P$ for $i \in \{1, 2, \ldots, n\}$. The coordinates x and y of a single point are denoted via x_i and y_i. In this way, the point cloud P will represent a discrete realization of the shape S.

The Dataset

The core of the used dataset is the geometric shape dataset [26]. This dataset consists of 10,000 pictures per shape and contains pictures of triangles, squares, pentagons, hexagons, heptagons, octagons, nonagons, circles, and stars (with a centered pentagonal hole) in different rotations, sizes, positions, and colors of the shapes themselves and the background. We also used this dataset in a previous paper, see [27]. There, we already transformed the images into point clouds. These point clouds define the starting point for our preprocessing.

The process we employ to boil down a color image to a point cloud is sketched in Figure 2 and will be shortly reviewed now. First, we select one color channel of the image and obtain a gray-scaled version of the image. The second step is the transformation into a binary black-and-white image. Thereafter, we use the *bwboundaries* MATLAB R2021b routine to compute the point clouds.

The reason behind using only one color channel of the image to create a grayscale image is that the pictures from the dataset were created to have a constant mean value over the whole image. Doing otherwise, we would end up with no difference between background and shape in the image, and we would not be able to produce a binary image for further processing.

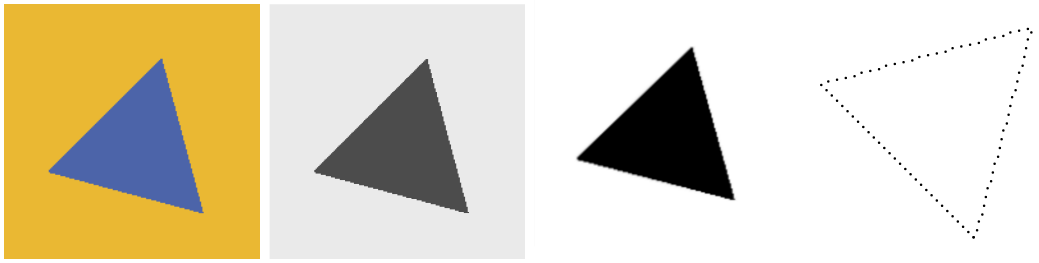

Figure 2. Illustration of basic steps we used to generate point clouds from color images (**left**). First, we just use a gray-valued version of the image (**middle left**) and transform it into a black-and-white image (**middle right**). As the last step, we use the MATLAB R2021b routine *bwboundaries* to create a point cloud (**right**); see also [27].

For the stars, the MATLAB routine bwboundaries produced two to three point clouds, i.e., one point cloud of the star, one pentagon point cloud, and occasionally an image border point cloud. These point clouds swapped their order from image to image. In the end, we were unable to automatically select only the useful point cloud. Since the proceeding is troublesome when producing point clouds from the star images, we chose to ignore these data and consider only the remaining geometrical shapes of the given dataset.

With the remaining dataset, we have a *geometrical shape set*

$$O = \bigcup_{s \in G} O_s \quad \text{with} \quad G = \{3, 4, 5, 6, 7, 8, 9, \infty\},$$

where we denoted the different shapes with their number of vertices. For the circle, we decided to use here the symbol ∞ to encode the number of vertices.

The point clouds P that result from this process are ordered. Therefore, the point $p_i \in P$ is the predecessor of the point $p_{i+1} \in P$ in the images and in the dataset. Since the geometric shapes have different sizes in the images, the number of points per point cloud differs for different realizations of a shape class. Therefore, we end up with point clouds with a size ranging from close to 100 to about 500.

3.2. Preprocessing

The point clouds are the starting point for the preprocessing that produces the random vectors X. Thus, the aim of the preprocessing is to convert all point clouds into normalized vectors of the same length, which are supposed to store enough information to make meaningful classifications possible.

3.2.1. Shape Descriptors and Signature

We will start to convert the point clouds P with their set of two-dimensional coordinates into a one-dimensional object.

This process should encrypt the geometry of the shapes in a meaningful, fast, and computable way. To this aim, we will adopt the D1 shape descriptors presented in [28].

This shape descriptor will calculate the distance $d(\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ between a point p from the point cloud P and a chosen center point $p_c = (x_c, y_c)$, as

$$d(p_i) = \|p_i - p_c\|_2 \quad \text{with} \quad p_c = \frac{1}{n} \sum_{i=1}^{n} p_i \tag{19}$$

We make use of the Euclidean norm $\|\cdot\|_2$ and the barycentre of the point cloud as the designated center point p_c. The center point will thus be calculated by the arithmetic average of all n points of the point cloud. In Figure 3, we provide a visualization of this process for a triangle point cloud.

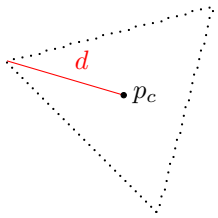

Figure 3. The visualization of a triangle point cloud, the center point p_c, and the distance d.

Now, we can create an ordered set of distance samples, also called an ordered collection,

$$D = \{d(p) \colon p \in P\} \tag{20}$$

with a collection size of n, which corresponds to the number of points in the point cloud. The phrasing "ordered" refers to the fact that we will keep the intrinsic order of the points from the underlying point cloud P.

In Figure 4, we plotted the ordered samples of a triangle, pentagon, nonagon, and circle. In these ordered samples, the vertices and the mid-edge points are recognizable through the higher and lower distance values. Additionally, we notice that the size of the shape impacts the produced distances d. This indicates that we still need some further preprocessing, which will be addressed in the next section.

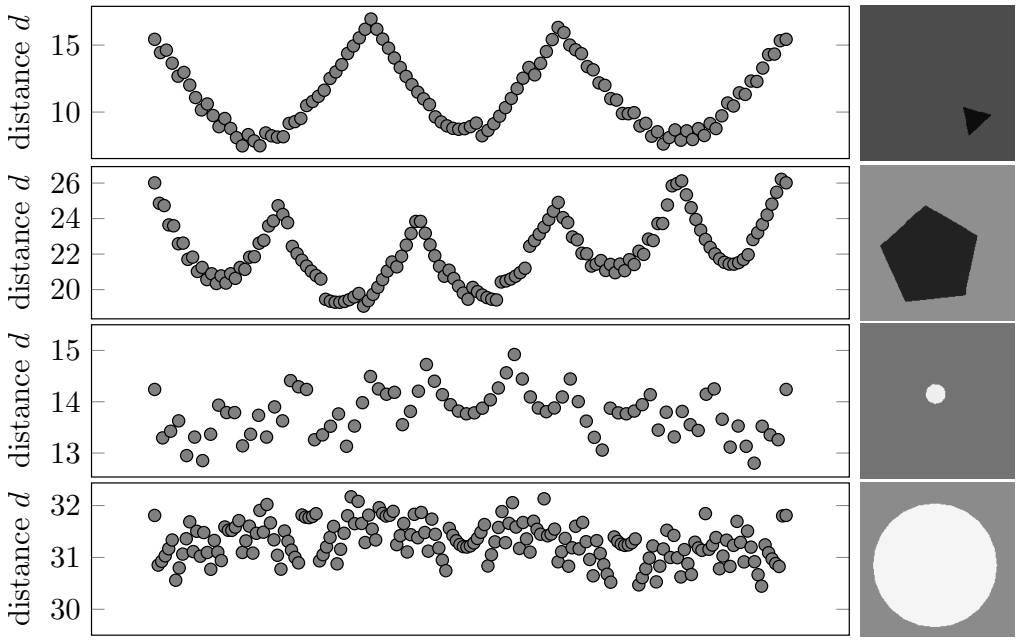

Figure 4. We plotted here the ordered D samples for four geometrical shapes. The y-axis shows the distance, and the x-axis is just an integer that indicates the position in the sample. Therefore, we left the x-axis empty. From top to bottom, we see the samples of a triangle, pentagon, nonagon, and circle. Additionally, we show the resource images of the samples on the right side. Compare [27].

3.2.2. Normalization

With the creation of the sample D, we eliminated dependency on rotation and translation. The sample is still correlated to the size of our shape, as we already mentioned. For example, bigger shapes or image-filling shapes will produce, on average, larger shape descriptors d since the distance between the center point and the boundary points is larger; see Figure 4. The geometrical shapes of the triangle (first row) and nonagon (third row) are similar in size, and therefore, the range of distance values is roughly similar. The pentagon

(second row) and circle (fourth row) produce distances that vary around a value of 23 and 31. And these effects occur between the samples within a certain shape class as well.

Since we can have large and small shapes, we want to unitize the samples for the next preprocessing steps and thus make the samples more comparable with each other. A first approach to this problem is normalization. Since there are multiple ways of normalization, we want to present the methods used:

Mean-Normalization To ensure that all collections have the same mean value, we compute the mean value of a reference collection $\mu_{D_{\text{ref}}}$. The mean value for a collection will be calculated via

$$\mu_D = \frac{1}{n} \sum_{k=1}^{n} d(p_k) \quad \forall d(p) \in D \text{ and } p \in P \tag{21}$$

The mean normalization of a collection D can then be calculated by

$$\widetilde{D} = \frac{\mu_{D_{\text{ref}}}}{\mu_D} D \tag{22}$$

Max-Normalization Similar to mean normalization, we will store the maximum of a reference collection $\max(D_{\text{ref}})$. Then, we ensure that every collection D will have the same max value as the reference collection via

$$\widetilde{D} = \frac{\max(D_{\text{ref}})}{\max(D)} D \tag{23}$$

After the mean-Normalization, all samples from a certain shape class have the same mean value, which is analogous to the max-Normalization, where all samples will have the same maximum value.

As for important implementation details, from the 10,000 samples per shape class, we simply take the first sample as the reference and normalize all other samples of that shape accordingly. The normalization is performed per shape class; that is, triangles are normalized with respect to the reference triangle, squares with respect to the reference square, and so on.

3.2.3. Histogram Technique

After normalization, the samples are no longer dependent on the shape size. However, the samples still consider the size of our point clouds. As the object size in the image increases, the number n of points in the point cloud P increases and so does the number of elements in D. This is why we have different sample sizes up to now.

Another issue is that the single entries do not contain any real information, which may be crucial for the PCA. The first element in the sample is the normalized distance from the first point in the point cloud to the center of the shape. This first point could be anywhere on the shape curve, as we have no control over the original sorting of the points, nor do we want to, as we do not want to limit the generalization properties of our approach unnecessarily.

To address these two points, we make use of a histogram technique. The length will be standardized into predetermined bins, providing an approximate representation of the distance distribution for each sample, denoted as D. Furthermore, we can store the results in the elements of a vector, and therefore even the position in a vector will encrypt information, which we consider helpful for the usage of the PCA. We thus expect that the distribution of distances for a triangle compared to another triangle has more in common than the comparison to, e.g., a distribution of distances for a square.

Let us now briefly explain the idea behind the histograms and how to construct them to obtain a vector from a shape that meets our desired requirements. The main idea of a histogram, as we use it, is to distribute a sample over $k \in \mathbb{N}$ bins, where each bin represents a range of values—in our case, the distances d from a sample D. Then, we can count how

many elements fall into each of the bins. Therefore, it is crucial that the bins do not overlap so that all values can only fall into one bin. In the end, we obtain a vector of length k, which stores the number of elements in each bin. And to keep the results comparable and ensure that the entries store the same information, it is mandatory to use the same bins for all histograms of a single shape.

Since the sample size still differs from sample to sample, the bins of larger samples contain more elements than the bins of smaller samples. To solve this, we can normalize the histograms so that each histogram sums up to an area of one. And the area can be calculated as the sum of all products of the number of elements in a bin and the width of the bin. We make use of this approach because most of the libraries used in programming have a density option for creating a histogram.

To summarize, with these histograms we have constructed a mathematical object in the form of a vector with meaningful axis entries from a shape that is independent of rotation, translation, and the size of the shape. In addition, the resulting vector has a defined length of $k \in \mathbb{N}$, and each dimension of the vector has a relationship to a shape class that may differ from class to class. This means that we can use them as a starting point for the PCA analysis presented in Section 2.3.

Summary of Preprocessing

We started with the two-dimensional point cloud obtained from geometrical objects. From these point clouds, we have calculated the D1-distance. Then, we normalized all D1-distances within a certain shape class. In the end, we compute a histogram from a normalized D1-distance. These histograms lay the foundation for the PCA.

3.3. Shape Classification

Our classification process is made up of two parts: the actual classification, where we determine the extent to which a shape belongs to a certain shape class, and the calculation of the hit rate to quantify the quality of our classification.

3.3.1. Classification Procedure

The core fundamentals for the classification are provided by Equation (18). If we have a normalized random vector X ($\|X\| = 1$) from an unknown class, we compute the principal component vector ${}_tY$ obtained from the eigenvector spaces $\mathbf{V}^t$ for $t \in G$. Thereafter, we compute the probability $p_{[X]}(\mathcal{P}_t)$ for different $k_0 \in [1, k)$ of X using Equation (18). In the end, we need to choose the class, in which X has the highest probability.

$$s = \arg\max_{t \in G} p_{[X]}(\mathcal{P}_t) = \arg\max_{t \in G} \sum_{l=1}^{k_0} \|{}_tY_l\|^2 = \arg\max_{t \in G} \sum_{l=1}^{k_0} \langle V_l^t \mid X - \mu_t \rangle^2, \tag{24}$$

where we make use of the Euclidean norm for $\|\cdot\|$, and since the single term in the sum is made up of scalars, it simplifies to the square of the terms.

This procedure can be seen in Table 1, where we use the values from Figure 5 to support this method with numbers. In the image, we show the square of principal components ${}^3_3Y$ (left) and ${}^3_4Y$ (right) in percentages. We see that the square of the first component of ${}^3_3Y$ contributes 73.14% to the total probability and, on the other hand, the first component of ${}^3_4Y$ will only contribute 4.12%. Therefore, the outcome of Equation (24) would always be 3, since we obtained the highest probability measure for the projection operator $\mathcal{P}_3$ of the triangle shape class, regardless of the chosen parameter k_0.

Table 1. The probabilities of a triangle random vector $X^3 \in \mathbb{R}^{10}$ ($k = 10$) acting on different projection operators over different values for $k_0 \in [1, k)$. In the second row, we used the projector $\mathcal{P}_3$ obtained from the triangle eigenvector space, and in the third row, we make use of the projector $\mathcal{P}_4$ from the square shape eigenvectors. For all values of k_0, the second row stores the higher values, which would lead to the conclusion that X is obtained from a triangle point cloud.

k_0	1	2	3	4	5	6	7	8	9
$p_{[X^3]}(\mathcal{P}_3)$	73.14	80.09	88.91	88.91	91.08	95.35	97.01	97.30	99.04
$p_{[X^3]}(\mathcal{P}_4)$	4.13	15.78	16.54	16.96	17.11	19.66	88.63	95.14	95.96

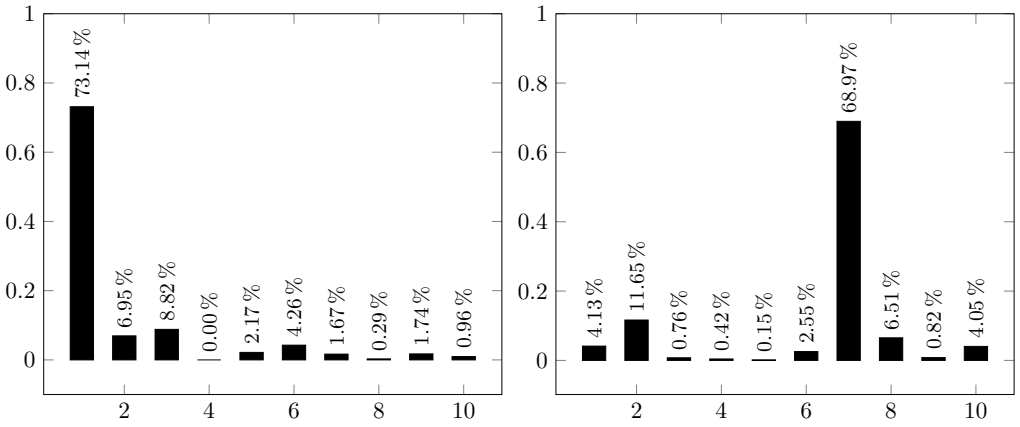

Figure 5. The square of principal components ${}^3_3Y$ (**left**) and ${}^3_4Y$ (**right**). We notice that a normalized triangle histogram X to a triangle eigenvector base will produce larger percentages in the first few components than the normalized triangle histogram that acts on the eigenvector base of a square.

3.3.2. Hit Rate

Consider having multiple shape classes stored in O and each shape class O_s for $s \in G$ has $m \in \mathbb{N}$ random vectors X. We want to know how many vectors are classified into a specific shape class.

Generally speaking, we will call a classification *successful* if $s = t$, and call it *fail* if $s \neq t$ for two shape classes O_s and O_t with $s, t \in G$. To quantify the quality of the classification, we introduce the *hit rate* $h(s,t)\colon G \times G \to [0,1]$, where we compute the quotient of the number of vectors of shape class O_t classified as shape class O_s over the total number of elements m_t in class O_t:

$$h(s,t) = \frac{\#X^t \text{ classified to } O_s}{m_t}, \qquad s,t \in G \tag{25}$$

The hit rate $h(s,t)$ will be visualized in a matrix-styled plot, see Figure 6. The x-axis will represent the result of the classification, shape class O_s, and the y-axis indicates the original shape class O_t, for $s, t \in G = \{3,4,5,6,7,8,9,\infty\}$. The gray value in the entries of the hit rate matrix encodes the value for the hit rate. A black entry will represent a value of one for $h(s,t)$, and for a value of zero, we will produce a white entry in the hit rate matrix.

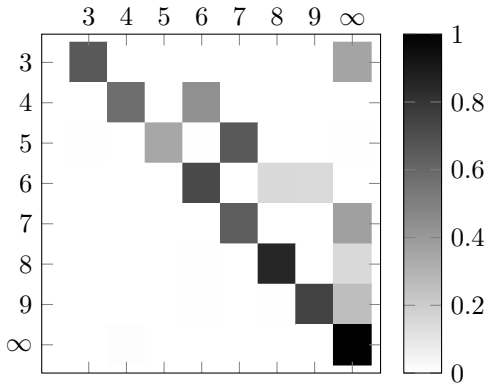

Figure 6. The visualization of a hit rate matrix $h(s,t)$ (25). The y-axis shows the source shape class O_t, and the x-axis presents the result of the classification process, e.g., shape class O_s, for $s,t \in G$. We illustrate the percentage as a gray valued box. Values closer to one, respectively 100%, yield darker boxes, and lighter boxes indicate lower percentages.

3.4. The Experimental Pipeline

In this section, we want to present the pipelines and algorithms we used to create our results.

First, we need to preprocess the loaded point cloud as discussed so that they can be used for the PCA. Algorithm 1 provides an overview of this process.

Algorithm 1: Algorithm for preprocessing the data.

```
Data: Point clouds of different shapes
Result: for PCA useable random vectors X
loading point clouds;
for every shape class O_s with s ∈ G do
    for each point cloud P in shape class O_s do
        Compute features via (19);
        Normalize the features via (22) or (23);
    end
    Compute max and min over all normalized features to unify the k histogram
     bins per shape class;
    for each normalized sample D̃ in class O_s do
        Compute the histograms and store them as random vectors X;
    end
end
```

For all shape classes O_s with $s \in G = \{3,4,5,6,7,8,9,\infty\}$, and for all point clouds belonging to class O_s, we produce normalized samples. Then, we computed the max- and min-values of all normalized samples $\widetilde{D}$ of each shape class O_s to define the width of the $k = 10$ bins. With these bins, we produced the histograms X from the samples $\widetilde{D}$. After this step, the preprocessing is finished.

The next step would be the separation between the training dataset and the test dataset. We chose a ratio of 7:3. From the 10,000 histograms X in shape class O_s, we used 7000 for the training with PCA, and the remaining ones for testing. These remaining 3000 histograms will be used to test the matching quality of this approach.

In the third step, we produced the eigenspaces $\mathbf{V}^s$ and mean values μ_s with the PCA and the histograms labeled for training. The PCA is performed separately for each shape class O_s with $s \in G$. We stored the produced eigenspaces and mean values for the classification process later on.

The fourth step calculates the classification via (15). This is illustrated in Algorithm 2. Here, we normalize the histograms labeled for testing again, so that $\|X\|_2 = 1$. We will make use of the 2-norm for this since $\|\cdot\|$ in (18) refers to the induced norm from the inner product. In this case, it is the 2-norm since we are using the Hilbert space $\mathbb{R}^k$. Thereafter, we compute the principal values ${}^t_sY$ for all combinations of $s, t \in G$, using Equation (15).

Algorithm 2: Compute the probability and construct the hit rate matrix.

Data: For testing random labeled vectors X, stored mean values μ and eigenspaces $\mathbf{V}$ of shape classes in O
Result: Hit rate matrix h
for *every shape class* O_t *with* $t \in G$ *and every* X^t *in* O_t **do**
  Normalizing the random vectors X^t;
  for *every eigenspace* $(\mathbf{V}^s, \mu_s)$, $s \in G$ **do**
    Compute ${}^t_sY$ via (15);
  end
  for $k_0 \in [1, k)$ **do**
    Compute the probability measure $p_{[X^t]}(\mathcal{P})$ with (18);
    Use these results to compute the best classification with (24);
    Compute the hit rate $h(s, t)$ via (25);
  end
end

With ${}^t_sY$, we can thus calculate probabilities via (18) and different $k_0 \in [1, k)$. Thereafter, we can produce a hit rate matrix using Equations (24) and (25).

Finally, we would like to emphasize the two main properties that the generated random vectors X should have. All vectors must have the same length, and each dimension should encrypt some information.

Since we do not make use of any complex calculation, the presented framework is quite fast. We need only 10 s from loading the point clouds to produce an image of the hit rate matrix, regardless of the chosen normalization.

The GitHub link to our experiment can be found in the section "Data Availability Statement".

3.5. Experiments

For conducting the experiments, we would like to formulate two theses. First, we would like to see that we are actually able to distinguish between different shape classes in O. Second, we would like to see that this prediction improves with an increasing value for k_0.

For this aim, we will make use of the presented pipeline and switch between the two normalizations, namely mean-Normalization and max-Normalization.

The resulting hit rate matrices for different $k_0 \in [1, k)$ and $k = 10$ are depicted in Figures 7 and 8. There, we increase $k_0 \in [1, 10)$ row-wise from the top left image down to the bottom right image. Now, we will examine the hit rate matrices with respect to our two theses.

Both figures have a visible diagonal line, which is more dominant for shapes with a higher number of nodes, i.e., hexagon up to circle. The mean-normalization tends to have more problems with low node shape classes than the max-normalization because the diagonal line is faintly recognizable.

Concerning the second hypothesis, we notice that the diagonal becomes more and more dominant if we increase the value of k_0. It is remarkable that we achieve the best results for values of $k_0 \in \{7, 8\}$. However, we observe for $k_0 = 9$ an overall increase in the number of failed classifications. These failed attempts are more spread over multiple shape classes in the max-normalization and fixed to the hexagon shape class for the mean-normalization.

In addition to the failed classifications, we notice other side effects and attempt to give an explanation for these.

We start with a potential explanation for the failed classifications. One reason could be the inherent structure of the principal component calculation. Therefore, we subtract the mean value μ_s of a specific shape class O_s with $s \in G$ from a random vector X. As a result, the random vector X belonging to the shape class O_s should be closer to the origin of the eigenvector space spanned by this shape class than the random vectors of the other shape classes. Since points close to the origin are more sensitive to small errors, a small error could already change the share of different eigenvectors considerably and, therefore, the principal components, too. Random vectors of other shape classes may not have this problem because their mean value differs from this sketched scenario. Therefore, random vectors of other shape classes than the one used for testing tend to stay away from the sensitive origin region. In consequence, these random vectors vary less in the principal components and may even be higher.

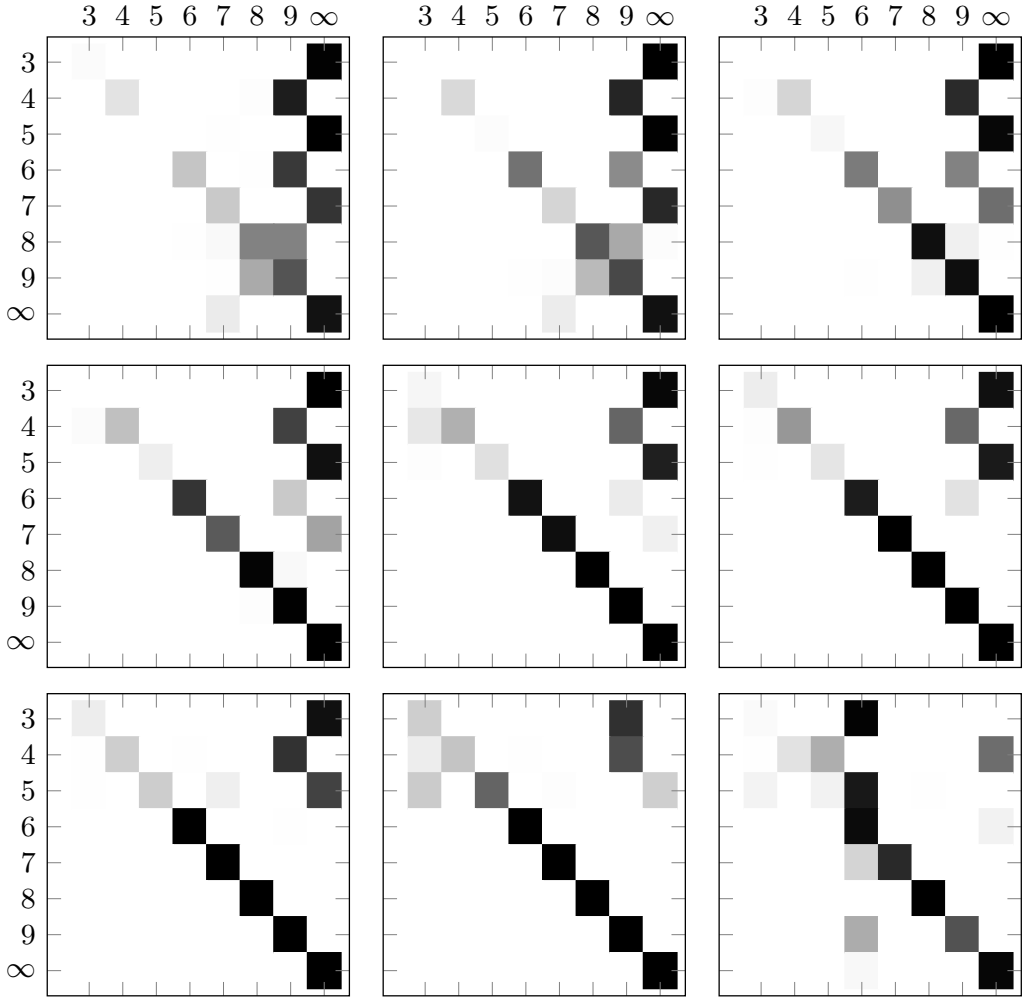

Figure 7. Hit rate matrices (25) for different values of $k_0 \in [1, k)$, $k = 10$. Row-wise, from the top left image down to the bottom right image, we are increasing the value k_0. Starting at one and ending at nine. The D samples were normalized with the mean-Normalization.

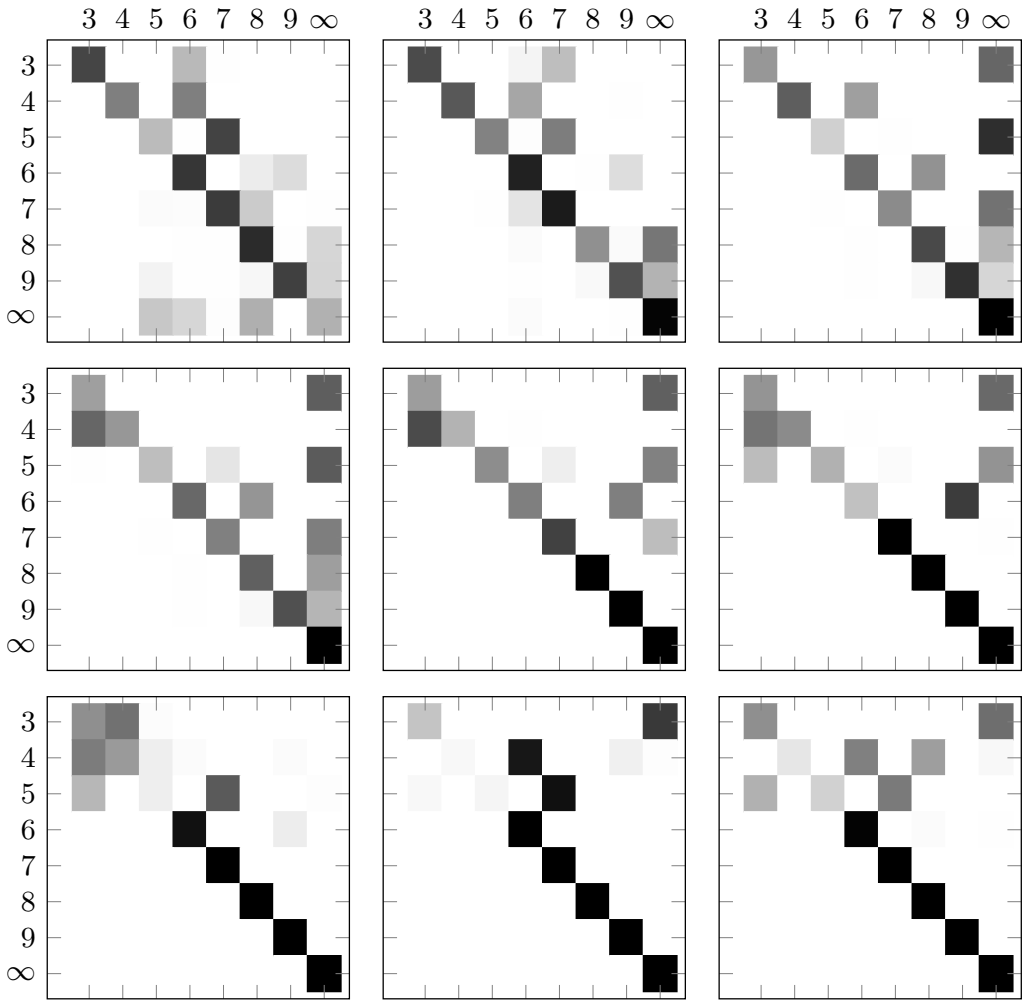

Figure 8. Hit rate matrices (25) for different values of $k_0 \in [1, k)$, $k = 10$. Row-wise, from the top left image down to the bottom right image, we are increasing the value k_0, starting at one and ending at nine. The D samples were normalized with the max-Normalization.

Another effect is that with increasing values of k_0, the eigenvalues converge to zero. The latter used eigenvectors are, therefore, not meaningful enough. And so, these eigenvectors do not add much to the information describing the testing shape, but allow random vectors from other shapes to increase their probability measure. Ending up with the failed classifications in the bottom left images in the presented figures.

To summarize, on the one hand, we notice that an interpretable, meaningful classification is possible. Even with this relatively simple approach, we can partially obtain correct classifications. Note that an advantage of such a simple approach is, for example, that it can be easily extended to 3D shape point clouds. On the other hand, we know that the presented preprocessing can be optimized in some aspects for better shape classification.

4. Conclusions and Future Work

In the first part of the article, we presented the main theoretical aspects. We illuminated the mathematical backbone of quantum logic, principal component analyses (PCA), and the connection between these two topics. With the presented formalism, the reader can use other methods, which provide Hermitian and idempotent operators, to establish a connection to quantum logic.

The presented theory is general enough to work with mixed states, e.g., with triangle-square hybrid shapes, but the formalism of the PCA inherently refers to pure states without putting in adjustments. One still has to find out, which may be a topic for future work,

how to mix eigenvector spaces, e.g., of triangles and squares, while keeping the general procedure of calculating probabilities via the eigenvector spaces.

The presented preprocessing may be optimized in future work for better results. In this paper, the main idea was to study a first approach to work on a meaningful task in image processing with the theoretical aspects of quantum logic. Especially, we think that the classification results can be improved with better preprocessing. The usage of the histograms, while at first glance adequate, seems not to preserve the inherent structure of the point clouds adequately enough. For example, one may have different point cloud distributions that could lead to the same histogram.

Another point for optimization is the described classification proceeding. We labeled a random vector to the shape class with the highest probability and ended up in some cases with a failed classification. This process leaves the question of whether the correct shape class was close to or far from the final one. With this in mind, we think that a ranking of the classification results would be more useful than committing to one shape class.

A final point that we would like to discuss in the future for optimizing our approach is that we did not fully utilize the possibilities of PCA formalism. The ordered eigenvalues allow us to reduce the eigenspaces to only necessary directions, and with the quantum logic, we can also quantify the error by doing so. This reduction could thus be explored more consequently in our presented pipeline, which could lead to some better classification results.

Author Contributions: Conceptualization, A.K. and M.K.; methodology, A.K.; software, A.K.; validation, A.K., M.K. and M.B.; formal analysis, M.K. and A.K.; investigation, A.K.; resources, A.K., M.K. and M.B.; data curation, A.K.; writing—original draft preparation, A.K. and M.K.; writing—review and editing, A.K., M.K. and M.B.; visualization, A.K.; supervision, A.K., M.K. and M.B.; project administration, A.K., M.K. and M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset of all images can be found via [26]. The code is available at: https://github.com/koehlale/A_First_Approach_to_Quantum_Logical_Shape_Classification_Framework, accessed on 16 February 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Birkhoff, G.; Von Neumann, J. The Logic of Quantum Mechanics. *Ann. Math.* **1936**, *37*, 823–843. [CrossRef]
2. Piron, C. Axiomatique de la théorie quantique. *Les Rencontres Physiciens-Math. Strasbg.-RCP25* **1973**, *16*, 1–13. Available online: http://www.numdam.org/item/RCP25_1973__16__A10_0/ (accessed on 6 May 2024).
3. Dalla Chiara, M.L.; Giuntini, R.; Greechie, R. *Reasoning in Quantum Theory: Sharp and Unsharp Quantum Logics*; Springer Science & Business Media: Dordrecht, The Netherlands, 2013; Volume 22. [CrossRef]
4. Rieffel, E.; Polak, W. An introduction to quantum computing for non-physicists. *ACM Comput. Surv. (CSUR)* **2000**, *32*, 300–335. [CrossRef]
5. Busemeyer, J.; Zheng, W. Data fusion using Hilbert space multi-dimensional models. *Theor. Comput. Sci.* **2018**, *752*, 41–55. [CrossRef]
6. Huber-Liebl, M.; Römer, R.; Wirsching, G.; Schmitt, I.; Wolff, M. Quantum-inspired cognitive agents. *Front. Appl. Math. Stat.* **2022**, *8*, 909873. [CrossRef]
7. Wolff, M.; Huber, M.; Wirsching, G.; Römer, R.; Graben, P.B.; Schmitt, I. Towards a Quantum Mechanical Model of the Inner Stage of Cognitive Agents. In Proceedings of the 2018 9th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Budapest, Hungary, 22–24 August 2018; pp. 147–152. [CrossRef]
8. Schmitt, I.; Romer, R.; Wirsching, G.; Wolff, M. Denormalized quantum density operators for encoding semantic uncertainty in cognitive agents. In Proceedings of the 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Debrecen, Hungary, 11–14 September 2017; pp. 165–170. [CrossRef]
9. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [CrossRef] [PubMed]
10. Duan, B.; Yuan, J.; Yu, C.H.; Huang, J.; Hsieh, C.Y. A survey on HHL algorithm: From theory to application in quantum machine learning. *Phys. Lett. A* **2020**, *384*, 126595. [CrossRef]
11. Pearson, K. LIII. On lines and planes of closest fit to systems of points in space. *Lond. Edinb. Dublin Philos. Mag. J. Sci.* **1901**, *2*, 559–572. [CrossRef]

12. Hotelling, H. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.* **1933**, *24*, 417–441. [CrossRef]
13. Jolliffe, I.T. *Principal Component Analysis*, 2nd ed.; Springer Series in Statistics; Springer: New York, NY, USA, 2002. [CrossRef]
14. Chen, J.; Jenkins, W.K. Facial recognition with PCA and machine learning methods. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017. [CrossRef]
15. Howley, T.; Madden, M.G.; O'Connell, M.L.; Ryder, A.G. The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data. In *Applications and Innovations in Intelligent Systems XIII*; Springer: London, UK, 2005; pp. 209–222. [CrossRef]
16. Bouwmans, T.; Javed, S.; Zhang, H.; Lin, Z.; Otazo, R. On the Applications of Robust PCA in Image and Video Processing. *Proc. IEEE* **2018**, *106*, 1427–1457. [CrossRef]
17. Patil, U.; Mudengudi, U. Image fusion using hierarchical PCA. In Proceedings of the 2011 International Conference on Image Information Processing, Shimla, India, 3–5 November 2011. [CrossRef]
18. Reich, D.; Price, A.L.; Patterson, N. Principal component analysis of genetic data. *Nat. Genet.* **2008**, *40*, 491–492. [CrossRef] [PubMed]
19. McVean, G. A Genealogical Interpretation of Principal Components Analysis. *PLoS Genet.* **2009**, *5*, e1000686. [CrossRef] [PubMed]
20. Nobre, J.; Neves, R.F. Combining Principal Component Analysis, Discrete Wavelet Transform and XGBoost to trade in the financial markets. *Expert Syst. Appl.* **2019**, *125*, 181–194. [CrossRef]
21. Le, T.H.; Le, H.C.; Taghizadeh-Hesary, F. Does financial inclusion impact CO_2 emissions? Evidence from Asia. *Financ. Res. Lett.* **2020**, *34*, 101451. [CrossRef]
22. Wirsching, G. Quantum-Inspired Uncertainty Quantification. *Front. Comput. Sci.* **2022**, *3*, 662632. [CrossRef]
23. Stubbe, I. The Geneva School Approach to the Axiomatic Foundations of Physics. DEA Dissertation, Université Catholique de Louvain, Ottignies-Louvain-la-Neuve, Belgium, 1999; Excerpt from DEA dissertation "Physics and Categories". Available online: https://www-lmpa.univ-littoral.fr/~stubbe/PDF/DEAChapterOne.pdf (accessed on 2 May 2024).
24. Soler, M.P. Characterization of Hilbert spaces by orthomodular spaces. *Commun. Algebra* **1995**, *23*, 219–243. [CrossRef]
25. Gleason, A.M. Measures on the closed subspaces of a Hilbert space. In *The Logico-Algebraic Approach to Quantum Mechanics*; Springer: Dordrecht, The Netherlands, 1975; pp. 123–133. [CrossRef]
26. Korchi, A.E. 2D geometric shapes dataset. *Mendeley Data* **2020**, V1. [CrossRef]
27. Köhler, A.; Rigi, A.; Breuss, M. Fast Shape Classification Using Kolmogorov-Smirnov Statistics. *Comput. Sci. Res. Notes* **2022**, *3201*, 172–180. [CrossRef]
28. Osada, R.; Funkhouser, T.; Chazelle, B.; Dobkin, D. Matching 3D models with shape distributions. In Proceedings of the Proceedings International Conference on Shape Modeling and Applications, Genova, Italy, 7–11 May 2001; pp. 154–166. [CrossRef]