Applications of Shape Data Analysis to Time Series Data and Manifold Learning

Benjamin Shaw
College of Science
Utah State University
Logan, UT, United States
ben.shaw@usu.edu

***Abstract*—Methods of shape analysis are applied to the analysis of time series data and to symmetry detection. Time series data are converted to shape representatives and subsequently compared using methods from shape data analysis. Additionally, 3-dimensional data is segmented into 2-dimensional shapes, and the similarity of the shapes is used to detect symmetry of the original 3-dimensional dataset.**

***Index Terms*—Shape analysis, symmetry, manifold learning, time series.**

I. Introduction

A. Symmetry and Manifold Learning

Recent developments have been made in detecting symmetries within a given dataset [1], [2]. In particular, symmetry detection has been explored in the context of 3-dimensional geometry [3]. Symmetry has recently been shown to enhance machine learning techniques [4], and, in particular, to perform manifold learning [5]. The first part of this paper will explore symmetry detection using techniques from shape data analysis in order to learn an underlying manifold structure.

Shape data is commonly realized as time series data. Conversely, we propose analyzing time series data by converting time series to shape representatives, then using methods for shape data analysis [6].

Colloquially, a symmetry is made manifest as a quantity that remains unchanged under the variation of another quantity. For example, a sphere is said to exhibit rotational symmetry about any axis through the center of the sphere, since the appearance of the sphere remains unchanged under the action of a rotation of the sphere about such an axis.

A more precise formulation of symmetry in the presence of a dataset is given as follows [1]. Suppose that f is a function of the input variables x: $f = f(x)$.[1] Let S be a transformation of the inputs: $x \to S(x)$. The function f is said to be *symmetric under the transformation* S if $f(S(x)) = f(x)$, and we say that the dataset exhibits a symmetry with respect to the transformation S.

Within the context of machine learning, incorporating symmetry has been shown to be of benefit to the predictive performance of neural networks [4]. It has also been shown to lead to more efficient data handling and understanding [1], as well as the reduction in the effective dimension of the dataset [2]. In this paper, the main motivation for the identification of symmetries is the explicit reduction–perhaps nonlinear–of the dimension of the dataset, and for the explicit construction of a Riemannian manifold associated to the dimensionality-reduced dataset. This formulation gives the dimensionality-reduced dataset the interpretation of an explicit embedding within a higher-dimensional space.

In this paper, we will test for structural symmetry under translational and rotational transformations. A given n-dimensional dataset will be segmented into k pieces of dimension $(n-1)$, which pieces are determined by the transformation S. By so doing, we will have constructed a companion dataset to the original, which companion has the interpretation of a sequence of $(n-1)$-dimensional clusters of data, which clusters we consider to be synthetically generated shape data. Under this construction, there are k such shapes..

For each of the k instances of shape data, we will attempt to learn an $(n-2)$-dimensional manifold structure using known manifold learning techniques. The degree to which the original dataset is symmetric with respect to the transformation S is determined by the degree to which the learned manifold structures are similar.

Similarity of shape data has been explored previously [7]. In fact, aspects of the geometric structure on 2-dimensional shapes generated by a parametric curve have also been explored [6]. In this paper, the original datasets will be 3-dimensional, so that the companion dataset is a sequence of 2-dimensional shapes: we will thus seek to compare the similarity of 1-dimensional manifolds, which manifolds have the interpretation of parametric curves. The measure of similarity between shapes will be determined by a similarity measure of parametric curves, and if the dataset is approximately symmetric under the transformation S, the similarity between the 1-dimensional learned manifolds will be approximately independent of the temporal component to the companion dataset.

We use the following formula to determine the similarity of parametric curves. A smooth parametric curve is a function $r : t \to \mathbb{R}^2$ defined by $r(t) = (x(t), y(t))$, where $x(t), y(t) \in \mathcal{C}^\infty(\mathbb{R})$, and where $t \in [t_0, t_1]$. Let u and v be two parametric curves. The following formula defines an inner product on the space of parametric curves:

[1]For example, consider the case of supervised learning: the function f could simply assign each datapoint to its class.

$$\langle u, v \rangle = \int_{t_0}^{t_1} u \cdot v dt. \tag{1}$$

The notion of an inner product allows us to define the angle between two parametric curves, as follows:

$$\cos(\theta) = \frac{\langle u, v \rangle}{||u||\ ||v||}, \tag{2}$$

which angle defines the similarity between u and v.

B. Time series and Shapes

A shape as an object is invariant under translations, rotations, and scaling [4]. Thus, a shape may be thought of as an equivalence class, and so in order to compare shape similarity, a representative shape from each equivalence class must be chosen.

A time series may be realized as a shape representative. The temporal component of the data may be transformed to a real number $\theta \in [0, 2\pi)$, and the remaining component of the time series may be interpreted as a distance r from the origin, normalized so that $r \in (0, 1]$. A Cartesian coordinate system $u, v)$ is obtained via the transformation $u = r\cos(\theta)$, $v = r\sin(\theta)$. The final transformation $(u, v) \rightarrow (x, y)$ can be obtained by mean-centering the data and projecting the data onto the principal components of the data. Under this composition of transformations, invariance under scaling, translation, and rotation have each been addressed.

The representative shape may also be interpreted as a parametric curve $r : t \rightarrow (x(t), y(t))$. With this interpretation, shape comparison is made by considering the similarity as in equation (2). Much of the success of this approach depends on the ability to fit the representative shape data as a suitable parametric curve: a regression task. For more complex shapes, this task becomes increasingly complex, as will be shown.

II. Symmetry and Manifold Learning

A. Example: translational symmetry

The first dataset is synthetically generated using the *S-curve* dataset [8] with 3000 samples and no added noise. A visualization of this dataset is given in Fig. 1 below.

It can be observed in Fig. 1. that the *S-curve* dataset exhibits an approximate translational symmetry in the y-coordinate direction: this is because of the apparent 2-dimensional "S-shape" perpendicular to the y-axis. Our goal is to understand this symmetry without the need for direct visualization. We will do this by outlining an algorithm which quantifies translational symmetry in a particular direction.

After transforming the data so that it is mean-centered, we calculate the principal components. They are given, approximately, as

$$\begin{aligned} v_1 &= 0.089\hat{i} + 0.003\hat{j} - 0.996\hat{k}, \\ v_2 &= -0.996\hat{i} - 0.007\hat{j} - 0.089\hat{k}, \\ v_3 &= 0.008\hat{i} - 0.999\hat{j} - 0.002\hat{k}. \end{aligned} \tag{3}$$

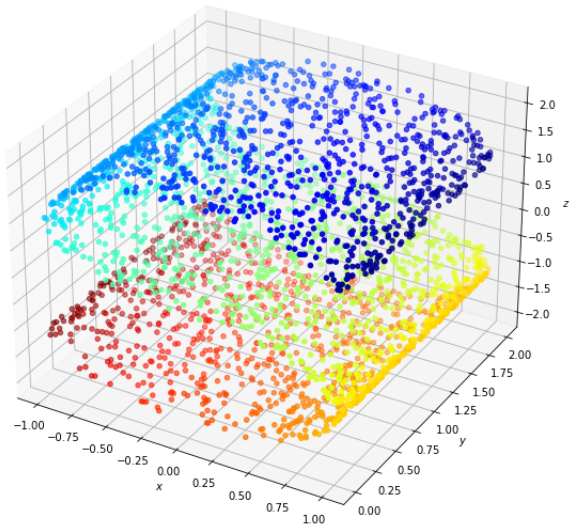

Fig. 1. The S-curve dataset with 3000 samples and no added noise.

Unsurprisingly, the principal components roughly coincide with the original component directions, up to a sign. We will quantify the translational symmetry in each of the principal component directions, and we will show that the best direction for translational symmetry is along the v_3 direction. Later, we will afford a discussion about the choice of principal components in testing for symmetry.

1) The optimal principal component: To quantify the symmetry in the direction of v_3, we proceed as follows. The datapoints are projected onto the nearest of 10 planes which are each orthogonal to the directional vector under consideration.[2] By so doing, we have constructed a dataset associated with the original dataset which can have the interpretation of time series shape data: upon each plane is a collection of datapoints, and the temporal component is defined by the sequence of 10 planes. A visual of one of these planes is given in Fig. 2. below.

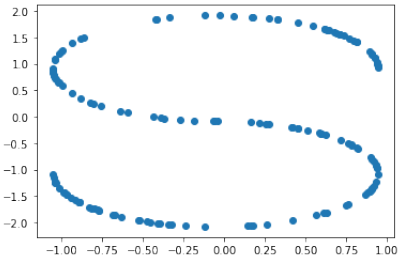

Fig. 2. The original dataset is projected onto 10 planes, each of which are orthogonal to the optimal principal component. This is one such plane.

With each point now projected onto precisely one of 10 planes, we now apply manifold learning techniques to understand the manifold structure of the data on each plane.

[2]In principle, k planes can be considered–however, 10 are selected here.

In principle, there are several manifold learning techniques which are available; for this example, the method of Spectral Embedding is selected. A visualization of a 2-dimensional spectral embedding of one of the 2-dimensional datasets is given in Fig. 3.

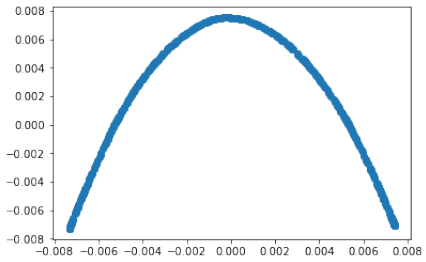

Fig. 3. Spectral Embedding is used for each 2-dimensional s-curve as a preliminary step to curve fitting.

The purpose of spectral embedding is to allow us to parameterize the apparent curve, and it is apparent that our method relies upon the assumption that each of the temporal components–the shapes–can be modeled as parametric curves or, in a general $(n-1)$-dimensional space, a parametric hypersurface. It is this assumption that will later allow for the explicit reduction in dimension.

Spectral embedding allows us to construct a parametric curve in the following way. The datapoints, under the transformation of the spectral embedding, give rise to a natural ordering of the points, as well as intervals between the two points: the horizontal axis of the embedded graph serves as the parameter t for the corresponding 2-dimensional s-curve. A plot of the the t-values vs. the x-values of an s-curve is given in Fig. 4, and a plot of the t-values vs. the y-values of the same s-curve is given in Fig. 5.

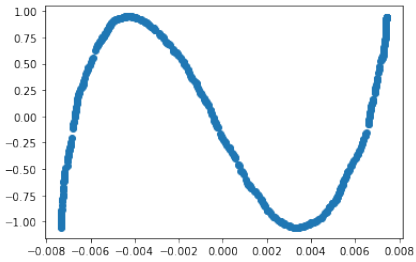

Fig. 4. This graph depicts the plot of the x-values from a 2-dimensional s-curve as a function of the parameter t, which parameter is gotten from the horizontal axis of the spectral embedding.

Thus, we have realized each 2-dimensional s-curve as a parametric curve r_i, where $1 \leq i \leq 10$. We now compute the mean shape μ by projecting all of the original 3-dimensional datapoints onto a 2-dimensional plane perpendicular to v_3. We also interpret μ as a parametric curve r_μ. The similarity between r_μ and r_i is computed as follows:

$$s_i = 1 - \frac{\theta_i}{\pi}, \tag{4}$$

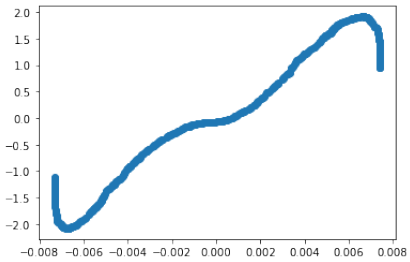

Fig. 5. This graph depicts the plot of the y-values from a 2-dimensional s-curve as a function of the parameter t, which parameter is gotten from the horizontal axis of the spectral embedding.

where θ_i is defined as in (2) with $u = r_\mu$ and $v = r_i$.[3] To quantify the symmetry along v_3, we compute the average similarity: $s = \frac{1}{n}\sum s_i$. Along the direction of v_3, we find that $s \approx 0.6$.

2) A suboptimal principal component: Having examined the symmetry along what is known *a priori* as the most optimal principal component, we now proceed to examine the symmetry in a suboptimal direction: that is, in either of the other principal component directions. We apply the same slgorithm as before, arriving at $s \approx 0.3$ for both v_1 and v_2. Thus, we conclude that the original dataset exhibits more translational symmetry in the direction of v_3 than in either of the other principal component directions.

The application of the spectral embedding algorithm when analyzing the symmetry along v_2 or v_1 is also telling. Several of the cross-sectional shapes do not have a good interpretation of a parametric curve, and this is indeed alluded to by the warning generated by the application of the spectral embedding package [8] that the underlying graph is not fully connected. This warning is a clue in itself that the original data is not well-modeled as being generated by a straight-line translation of a parametric curve.

B. Rotational Symmetry and the Explicit Construction of the Riemannian Metric

We now turn our attention to a dataset which is generated synthetically by the following equation: $z = x^2 + y^2$. After the datapoints are generated, they are rotated and subsequently mean-centered. The result, along with the first principal component, is depicted in Fig. 6.

The shape is approximately generated by a rotation of a parabola about the first principal component, which we now consider to be the x-direction. Applying a rotation (about the x-axis) of all datapoints onto the (x, y) plane (right half), we obtain the following plot, depicted in Fig 7.

The data may thus be said to be approximately generated by a rotation of the curve $x = y^2 - 44$ about the x-axis. Thus, a datapoint may be uniquely represented by its place on the given curve u and by its angular value v.

At the onset, the original 3-dimensional dataset is thought to have points belonging in $\mathbb{R}^3$, equipped with the Euclidean

[3]Our parametric curves are normalized so that the limits of integration are -1 and 1.

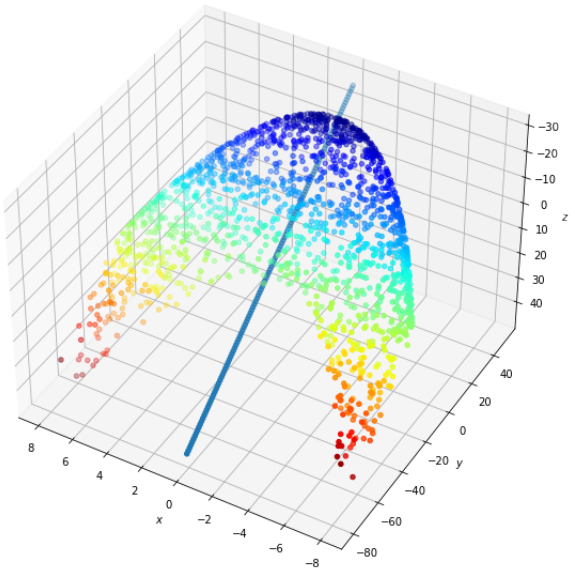

Fig. 6. The graph of the rotated paraboloid, along with the first principal component.

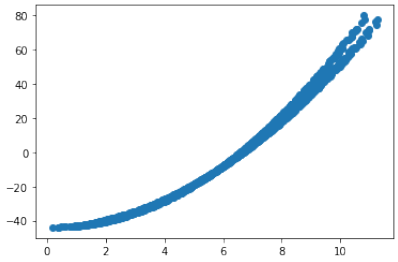

Fig. 7. The datapoints are projected via rotation onto the right half of the (x, y) plane, with the vertical axis being the x-axis. The best fit curve in the form $x = ay^2 + b$ is $x = 0.997y^2 - 43.9$ with an R^2 value of 0.99657.

(flat) Riemannian metric. Writing our datapoints in terms of the principal components is representative of a linear transformation, so that the metric g is still manifestly Euclidean:

$$g = dx^2 + dy^2 + dz^2.$$

The symmetry of our dataset provides a natural embedding of the coordinates (u, v) into $\mathbb{R}^3$. This embedding is given by the following:

$$x = (u^2 - 44), \qquad y = u\cos(v), \qquad z = u\sin(v). \tag{5}$$

The pullback of the metric g using this transformation yields the following metric $\tilde{g}$ in the coordinates (u, v):

$$\tilde{g} = \left(1 + 4u^2\right) du^2 + u^2 dv^2. \tag{6}$$

The Riemannian metric, while not a metric in the sense of endowing the surface with the structure of a metric space, can be used to define such a metric. To begin this process, the geodesic equation must be solved, either exactly or numerically. For demonstration purposes, we have listed the geodesic equations below, using the connection defined by the Christoffel symbols of $\tilde{g}$. The Differential Geometry Software Package [9] was used to compute these equations.

$$u'' - \frac{u((v')^2 - 4(u')^2)}{4u^2 + 1} = 0, \qquad v'' + \frac{u'v'}{u} = 0.$$

The explicit construction of a Riemannian metric in the context of manifold learning is believed to be under-utilized and will likely be explored in forthcoming works.

III. Application of Shape Analysis to Time Series

A. An example using Synthetic data

As was previously stated, tools from shape analysis can be used in the analysis of time series. Time series are represented as shapes, which shapes are represented as parametric curves, which curves can then be compared to other curves. We first consider the following simple example of synthetically-generated time series.

Three time series are generated, as seen in Fig. 8, Fig. 9, and Fig. 10: we refer to these time series as $T1$, $T2$, and $T3$, respectively. $T1$ and $T3$ are generated by the line $y = x$ along with random noise, though the range for x in $T1$ is $[-2, 2]$ while the range for $T3$ is $[0, 2]$. $TS2$ is generated by the following curve with random noise:

$$y = \begin{cases} e^{\frac{3}{(2x)^2 - 1}} + x & -1/2 < x < 1/2 \\ x & x \in [-1, -1/2] \cup [1/2, 1]. \end{cases} \tag{7}$$

The linear density of points varies for each time series.

We will assume that the classes for $T1$ and $T2$ are known and distinct–class 1 and class 2, respectively–and that the task at hand is to classify $T3$ into either the $T1$ class of the $T2$ class. This will be done by comparing the similarity of $T1$ and $T3$ with the similarity of $T2$ and $T3$.

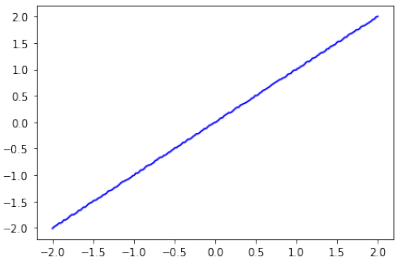

Fig. 8. A synthetically-generated time series generated by the curve $y = x$ with random noise. It is assumed that this time series belongs to class 1.

We first transform each time series to a representative shape as explained in the introduction. Fig. 11 depicts both $T1$ and $T3$ as representative shapes, and Fig. 12 depicts both $T2$ and $T3$ as representative shapes. Visually, it is clear that $T1$ and $T3$ are more similar than $T2$ and $T3$.

We now fit each representative shape to a parametric curve $r : [0, 2\pi) : \mathbb{R}^2$. Let r_1, r_2, and r_3 be parametric curves

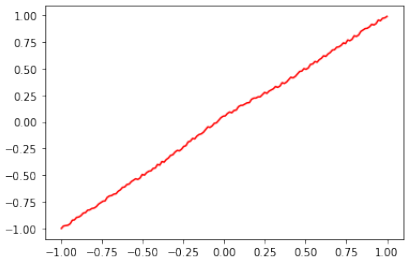

Fig. 9. A synthetically-generated time series generated by the curve given in equation (7) with random noise. It is assumed that this time series belongs to class 2.

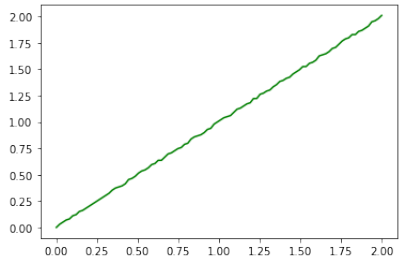

Fig. 10. A synthetically-generated time series generated by the curve $y = x$ with random noise. No class assumption is made: the task is to classify this time series into class 1 or class 2.

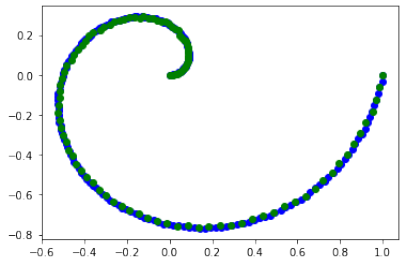

Fig. 11. Representative shapes of $T1$ and $T3$.

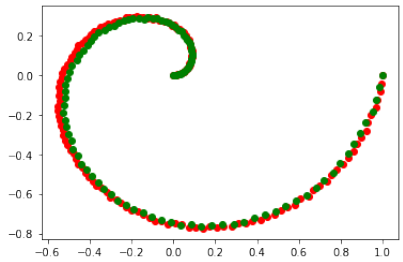

Fig. 12. Representative shapes of $T2$ and $T3$.

corresponding to $T1$, $T2$, and $T3$, respectively. Applying equation (2), we find that the angle between r_1 and r_3 is approximately 0.003, whereas the angle between r_2 and r_3 is approximately 0.14. Therefore, our methods would correctly classify $T3$ as belonging to class 1.

B. Trial data: leaf data

We will now attempt to scale our synthetically-generated example to a more realistic dataset. We load in the "OSU leaf" dataset [10], which is given as a time series, though originally "shape data"[4] corresponding to the boundaries of leaves.

Having loaded the dataset, the time series are converted to representative shapes, as discussed earlier. Fig. 13 depicts two such shapes, which shapes correspond to leaves belonging to two separate classes. Fig. 14 depicts two shapes belonging to the same class. At issue already is the in-class variation of shape representatives.

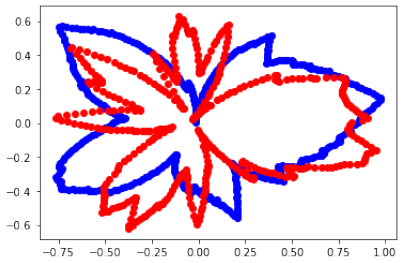

Fig. 13. Shape representatives corresponding to two distinct classes.

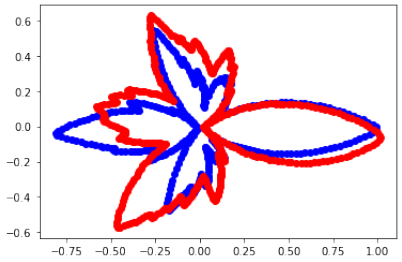

Fig. 14. Shape representatives belonging to the same class.

Although each time series can be associated with a well-defined parametric curve, applying the parametric fit in practice is of concern, since the outline of a particular leaf is more complicated as a shape than what was encountered previously. Indeed when we apply a curve fit of the blue shape in Fig. 14, we obtain Fig. 15.

The parametric curve in Fig. 15 was assumed to be a 9^{th}-order polynomial in both components, and yet the complexity of this model was wholly insufficient to capture the complexity of the shape. Thus, our proposed method of time series classification via the determination of the similarity of parametric curves becomes less practical when the associated shape data is complex.

[4]Strictly speaking, the shape data may not have been transformed so as to address the issue of invariance under translation, rotation, and scaling.

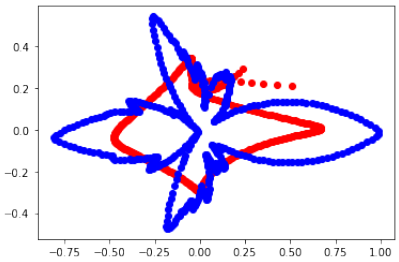

Fig. 15. A plot of a shape representative (blue) and the fitted curve (red).

IV. Conclusion and Future Work

We have shown that shape data analysis can be used to detect symmetry and to quantify the similarity of time series data. When the associated shape data is modeled as a parametric curve, the method is limited by the ability to fit the shape data to a parametric curve.

When symmetry is detected, it can be used to explicitly reduce the dimensionality of the original dataset, and it can be used to explicitly construct a Riemannian metric. The explicit construction of a Riemannian metric leads to a more concrete manifold model, and may allow one to compute other interesting geometric quantities, including a metric to measure geodesic distances between points. These geometric quantities may subsequently be used to advance understanding of a particular dataset.

The utility of constructing a Riemannian metric may be explored in greater length in forthcoming publications. It is known that the Riemannian metric g can be used to construct a metric d which endows the manifold with the structure of a metric space. The metric d may subsequently be used to compute manifold distances between points, thus enabling many conventional machine learning techniques in the intrinsic, dimensionality-reduced manifold setting. This may lead to more efficient data analysis.

References

[1] S. Krippendorf and M. Syvaeri, "Detecting symmetries with neural networks," 2020. [Online]. Available: https://arxiv.org/abs/2003.13679

[2] K. Desai, B. Nachman, and J. Thaler, "Symmetry discovery with deep learning," *Phys. Rev. D*, vol. 105, p. 096031, May 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.105.096031

[3] N. J. Mitra, L. J. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3d geometry," in *ACM SIGGRAPH 2006 Papers*, ser. SIGGRAPH '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 560–568. [Online]. Available: https://doi.org/10.1145/1179352.1141924

[4] D. L. Bergman, "Symmetry constrained machine learning," in *Advances in Intelligent Systems and Computing*. Springer International Publishing, aug 2019, pp. 501–512. [Online]. Available: https://doi.org/10.1007

[5] S. Craven, D. Croon, D. Cutting, and R. Houtz, "Machine learning a manifold," *Physical Review D*, vol. 105, no. 9, may 2022. [Online]. Available: https://doi.org/10.1103

[6] K. Bharath and S. Kurtek, "Analysis of shape data: From landmarks to elastic curves," *WIREs Computational Statistics*, vol. 12, no. 3, p. e1495, 2020. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.1495

[7] M. B. Stegmann and D. D. Gomez, "A brief introduction to statistical shape analysis," Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, p. 15, mar 2002, images, annotations and data reports are placed in the enclosed zip-file. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/403-full.html

[8] F. Pedregosa *et al*., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] I. M. Anderson and C. G. Torre, "The differential geometry software project," May 2022. [Online]. Available: https://digitalcommons.usu.edu/dg_downloads/4

[10] M. Löning, F. Király, T. Bagnall, M. Middlehurst, S. Ganesh, G. Oastler, J. Lines, M. Walter, ViktorKaz, L. Mentel, chrisholder, L. Tsaprounis, RNKuhns, M. Parker, T. Owoseni, P. Rockenschaub, danbartl, jesellier, eenticott shell, C. Gilbert, G. Bulatova, Lovkush, P. Schäfer, S. Khrapov, K. Buchhorn, K. Take, S. Subramanian, S. M. Meyer, AidenRushbrooke, and B. rice, "sktime/sktime: v0.13.4," Sep. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7117735