# Moving Past Principal Component Analysis: Nonlinear Dimensionality Reduction Towards Better Hand Pose Synthesis

Edoardo Battaglia, Michael Kasman and Ann Majewicz Fey

2022 International Symposium on Medical Robotics (ISMR) | 978-1-6654-6928-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISMR48347.2022.9807580

*Abstract*— **Despite their complex kinematic structure with many degrees of freedom, human hands have been shown to have synergistic behavior, with coordinated joint movements being able to explain a large amount of the variance in hand posture measurements. This phenomenon has traditionally been analyzed through Principal Component Analysis (PCA), and has led to important applications in medical robotics, such as the design and control of upper limb prostheses and measurement of hand posture with a reduced number of sensors. However, the use of more complex, nonlinear dimensionality reduction techniques for hand joint measurements has been under-explored in the literature. In this paper, we aim to fill this gap by comparing Principal Component Analysis, Kernel Principal Component Analysis (KPCA), and autoencoders on the same data set, evaluating the performance in terms of Mean Square Error of reconstructed hand poses with respect to the original data set. Results show a better performance for the nonlinear techniques, lowering Mean Square Error up to 25% for the KPCA and 50% for the autoencoders when compared to PCA. Visualization of the reconstructed poses shows a better ability from the autoencoder to reconstruct hand shapes when compared to the two other methods.**

## I. Introduction

Human hands are, at the same time, simple and complex. Simple in the way that they are used, performing advanced grasping and manipulation tasks in everyday life without too much thought from the human user. Yet, they are complex in their anatomy and have a complicated kinematic structure with several degrees of freedom. This dual nature has been highlighted by numerous studies in neurophysiology, which have shown evidence of the existence of *postural hand synergies*, i.e., kinematic patterns in hand postures that can explain complex hand movements with a reduced set of degrees of freedom. This has been observed for static poses [1], reach to grasp [2] and grasping movements [3]. Interestingly, similar patterns can also be observed in muscular activation [4] and there is indication of an overall synergistic framework that begins from the brain and ends in physical grasp postures [5]. Of note, all these studies used Principal Component Analysis (PCA) to linearly reduce the dimensionality of hand pose datasets.

These results from neuroscience have been a source of inspiration for the design of artificial hands, with applications in upper limb prosthetics and robotic end-effectors. A synergy-inspired robotic hand was first presented by Brown and Asada in [6], where synergies extracted from measured human hand postures were used to design a 17 degree of freedom hand with two degrees of actuation. Gabiccini et al. formalized a mathematical model of *soft synergies*, which added compliance to the synergistic behavior[7]. An underactuated robotic hand with 19 joints and one degree of actuation was designed based on this model [8], which was later adapted as a prosthetic system [9], [10]. Other groups have proposed similar approaches [11], [12], [13], [14].

In addition to informing the design, synergies have inspired solutions for control of actuated robotic hands. Ciocarlie and Allen first proposed this approach in [15], where they showed the advantages of using a reduced dimensionality framework and implemented in an online grasp planning. This approach led to several works that mapped human hand measurements to artificial hands [16], [17], [18], while others used similar methods to synthetize grasps without a human in the loop [19], [20].

These concepts were also translated to muscular synergies and used for the control of upper limb prostheses [21], and similar ideas have been explored for hand exoskeletons in rehabilitation [22] and hand posture estimation from a reduced number of sensors [23]. Synergies could also potentially be used to simplify mapping of kinematic data to surgery skill level in robotic surgery [24].

In this paper we compare the performance of PCA and nonlinear reduction techniques on human hand poses, with the evaluation being done on joint angles, and using the difference between reconstructed and original hand poses as a benchmark. We used the open data set released by Glauser at al. in [25], which is a large (over 1 million samples) data set of 10 subjects performing hand movements covering a wide kinematic range, instead of a limited number of grasp shapes. We also focus on reconstruction using a limited number of degrees of freedom (up to three), which is a desirable feature in most medical robotics applications to reduce complexity and keep control easy for the user (e.g., in upper-limb prosthetics [26]). We aim for the results of this work to be general and relevant for multiple synergy-inspired applications, rather than focus on a single specific application. To the best of our knowledge, this is the first work that compares Principal Component Analysis to nonlinear approaches for dimensionality reduction of a hand pose dataset.

This work was supported in part by NSF #1846726.

E.B. is with the Department of Mechanical Engineering, University of Utah, UT 84112, USA. edoardo.battaglia@utah.edu

M.K. is with the Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, TX 75080, USA

A.M.F. is with the Department of Mechanical Engineering, The University of Texas at Austin, TX 78712, and the Department of Surgery, UT Southwestern Medical Center, Dallas, TX 75390, USA

## II. Dimensionality reduction techniques

### A. Principal Component Analysis

Principal Component Analysis can be seen geometrically as describing a data set through a coordinate system where the first axis accounts for the largest amount of variance, the second accounts for the largest amount of remaining variance, and so on until the complete dimensionality of the data is reached [27]. For dimensionality reduction a subset of these *principal components* is chosen, and the data is described through only this subset of coordinates, effectively projecting it on a lower dimension hyperplane [28].

From an operative point of view, PCA is performed by doing a Singular Value Decomposition (SVD) of the covariance matrix of the centered data, obtained by stacking the data by columns in a matrix and subtracting the mean. This can also be equivalently approached using eigenvector decomposition; we refer to the tutorial from Shlens [29] for more details. Because of this, PCA is a linear technique, and while its use is standard and allowed to obtain important results for what concerns hand dimensionality reduction, here we are interested in considering more complex techniques and see if their use can increase performance in hand posture decomposition. To the best of our knowledge, such analysis was never done before.

### B. Kernel Principal Component Analysis

The first nonlinear technique that we will consider is the *Kernel* PCA (KPCA). This technique is similar to the PCA, but is made nonlinear by transforming the data via a nonlinear function, with the PCA being performed in the transformed *feature* space [30]. More formally, we consider a mapping $\phi : \mathbb{R}^n \mapsto F$, where $\mathbb{R}^n$ is the original space for the data set (i.e., measured joint angles) and $F$ is the feature space. It can be shown that as long as a function $k$ respects certain conditions, the relation $k(x,y) = \phi(x)^T \phi(y)$ holds for some $\phi$[27]. In this case $k$ is called *kernel* function, and it allows to calculate a kernel matrix $K$ directly in the feature space. Thanks to this *kernel trick*, it is possible to perform PCA in the feature space (i.e., KPCA) without actually calculating, or even knowing, the function $\phi$.

Because it injects a nonlinearity through the feature space mapping, KPCA can expose patterns in data sets that lie on nonlinear manifolds, which the PCA would not be able to unravel [31]. However, this comes with the drawback of a more complex process for reconstruction of the original data in $\mathbb{R}^n$ from its mapping in $F$ [27]. Furthermore, KPCA comes with a significant additional computational load when compared to PCA, causing problems with large data sets as shown by Kim in [32] and Chin and Suter in [33].

### C. Autoencoders: neural networks for dimensionality reduction

Autoencoders are artificial neural networks that map some input data to itself, and in doing so highlight patterns in the data [34]. In their simplest form, they are composed by two layers: an *encoder*, which usually has a number of neurons lower than the size of the input, and maps it to a lower dimensionality representation known as *latent* representation[1]; and a *decoder*, which maps the latent representation back to the original input [27].

Interestingly, when no activation functions are used and the Mean Square Error is used as cost function for training, such a network builds a latent representation that spans the same subspace that would be represented by a PCA. This makes autoencoders particularly interesting for a comparison that has PCA as baseline, since one can gradually increase the complexity of the basic autoencoder and see how this affects the reconstruction. Such increase in complexity can be achieved by using nonlinear activation functions and adding layers, making the network *deep*. In the following sections we will show how we designed the network architecture using the suggestions laid out by Charte et al. in [34] as general guidelines.

## III. Evaluation Methods

The comparison was done on an open[2] data set of hand pose measurements, introduced by Glauser et al. in [25], where it was used as ground truth to calibrate a sensing glove for hand pose reconstruction. It contains joint angles in radians that were obtained from a depth camera through an off-the-shelf hand tracking algorithm described by Tkach et al. in [35], with 10 participants performing a variety of hand poses that were shown to them during the recording. The hand model used, as introduced in [35], [25], has 25 degrees of freedom and is shown in Figure 1.

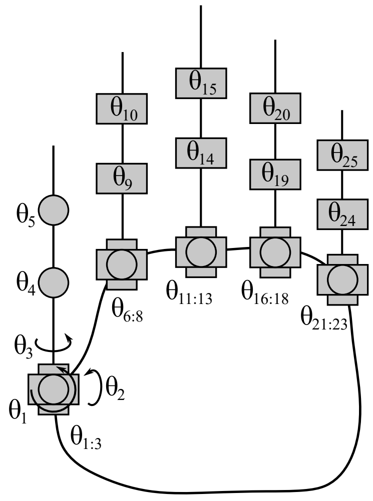


Fig. 1: Hand model as defined in [35], [25] (right hand, view from above).

Compared to other data sets in the state of the art (e.g., [1], [36], [37]), the data from [25] contains a very high number of samples (over one million across all participants).

[1]It is possible to have a number of neurons bigger than the size of the input in the encoder, leading to an *overcomplete* autoencoder. Since we want to use autoencoder for dimensionality reduction here we will focus on *undercomplete* autoencoders.

[2]`https://igl.ethz.ch/projects/stretch-glove`

Additionally, the target poses that were shown to participants were designed to encourage them to explore the complete kinematic range of the hand, instead of being limited to a few target grasp shapes, making this data set well suited to capturing a wider range of kinematic capabilities of the human hand.

The data set was divided in three subsets: a *training* set, that was used to train the dimensionality reduction techniques that we are comparing, a *validation* set, which was used to tune hyperparameters and the autoencoder weights, and a *test* set, on which the final evaluation was done for each technique. We used the data from eight participants for training, while data from the two remaining participants was used for validation and testing, respectively. The hand size of each participant was described in [25] through the bounding box volume in cm$^3$, and we used this as an identifier for the geometry of the hand. In order to preserve as much generality as possible in the training set, we used the median sized subject as test set, and the one that was median once the previous one was dropped as validation set.

## A. Comparison procedure and error metric

A generic dimensionality reduction procedure can be seen as a function that maps a set of joint angles from $\theta \in \mathbb{R}^N$ to $\zeta \in \mathbb{R}^M$, where $M < N$ and $\zeta$ is a set of parameters in $\mathbb{R}^M$. If we assume that this function is invertible, we can obtain a measure of how good a reconstruction in the reduced space is with the following steps:

- Take a numeric vector $\bar{\theta}$ and apply the dimensionality reduction technique to obtain the corresponding $\bar{\zeta}$;
- Use the inverse transform to obtain an estimation of the original $\bar{\theta}$ from $\bar{\zeta}$. Let us call this estimation $\bar{\theta}_e$;
- Calculate the Mean Square Error of $\bar{\theta}_e$ and $\bar{\theta}$.

We can then evaluate performances of different dimensionality reduction procedures on the data set by considering the overall Mean Square Error (which is also the metric being minimized by PCA).

In addition to the quantitative comparison, we also developed a hand visualizer in C++, using the haptic library CHAI3D [38] to link simple 3D geometries according to the hand model, and used it to compare reconstructed hand poses with the originals. Note that, while the bounding box measurements for participants' hands are provided in the data set, measurements of phalanxes lengths are not available. In order to have a realistically proportionate hand for visualization purposes we built the geometry according to average hand proportions as reported by Alexander et al. in [39]. All data analysis was done in Python, using the scikit-learn and tensorflow libraries [40], [41].

## B. Principal Component Analysis

We used the `PCA` method implemented by scikit-learn, which centers the data without scaling it. This is reasonable since all variables have the same dimensionality, and is consistent with the state of the art on dimensionality reduction for joint angles (see e.g. [1]). We then used the scikit-learn method `fit_transform` to obtain the reduced dimensionality representation from the train data, and the scikit-learn method `inverse_transform` to obtain estimates in the original space for the training, validation and test data. This was done for one, two and three principal components. Throughout the paper we will indicate the number of principal components with $n_c$. Results will be reported for the full dataset as well as a few subsampled set, obtained through the pandas library `sample` method while specifying a fixed `random_state` for consistency, in order to show the effect of data set size.

## C. Kernel Principal Component Analysis

As mentioned earlier, KPCA presents two main challenges for application to the problem that we are considering: (i) it is computationally heavy for big data sets, and (ii) obtaining the inverse transformation back to the original joint angles space is not as straightforward as it is for the PCA. There are works that address (i) [33], however doing so while also mantaining (ii) is not trivial. For this reason in this work we decided to use the standard scikit-learn methods, and subsample the data set to deal with the computational burden issues (we considered up to 40000 samples for the KPCA).

We considered as kernel functions the commonly used gaussian radial basis function (RBF), polynomial and sigmoid activation functions in our analysis (Table I). For each we tuned the parameter $\gamma$, the coefficient $r$ for polynomial and sigmoid function, and the exponent $d$ for the polynomial kernel function, by doing a grid search on a 20k subsample and choosing the values that yielded the lowest reconstruction errors on the training and validation sets. We then compared the results on the validation set and chose the kernel function that provided the lowest error as best representative for the KPCA case. All previous steps were followed for one, two and three principal components ($n_c = 1, 2, 3$), as well as a variety of sample sizes as shown in Section IV.

| Kernel Function | | $\gamma$ | $r$ | $d$ |
|---|---|---|---|---|
| Gaussian RBF | $K(\mathbf{x},\mathbf{y}) = \exp(-\gamma\|\mathbf{x}-\mathbf{y}\|^2)$ | 0.05 | - | - |
| Polynomial | $K(\mathbf{x},\mathbf{y}) = (\gamma\mathbf{x}^T\mathbf{y}+r)^d$ | 0.02 | 5.0 | 5.0 |
| Sigmoid | $K(\mathbf{x},\mathbf{y}) = \tanh(\gamma\mathbf{x}^T\mathbf{y}+r)$ | 0.05 | 0.1 | - |

TABLE I: Kernel PCA design.

## D. Autoencoders

There are multiple architectures that could be feasible for an application of autoencoders to our problem. We focused on fully connected layers, and used the suggestions provided in [34] as general guidelines to guide the design process. Some preliminary tests showed indication of a deep architecture giving higher performance than a shallow network, and for this reason we only considered deep autoencoders designs. Similarly, we considered tanh and sigmoid as possible activation functions for the hidden layers in the encoder, and selu and relu for the hidden layers in the decoder. Figure 2 shows the two types of architectures that were examined. The assignment of an activation function to each layer was chosen through a grid search and corresponds to the lowest training and validation errors from trying the different combinations in each layer. For architecture B in particular, we found that using relu for the first layer in the decoder improved performance when compared with using selu as activation function in both layers.

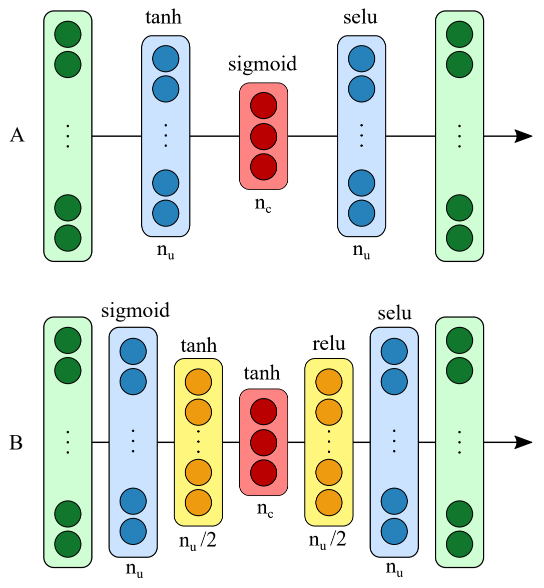


Fig. 2: Autoencoder architectures. Here $n_c$ is the number of neurons in the innermost inner layer and correspons to the number of principal components considered for PCA and KPCA, while $n_u$ is a design parameter.

All networks were trained using the Adam optimizer [42], which has been shown to be generally robust and reliable for most data sets, and by using Mean Square Error as loss function to ensure a fair comparison with PCA and KPCA. In order to limit overfitting, an early stopping callback was added to the training, to stop it if there was no improvement in the validation loss after 30 epochs and restore the weights to the values that yielded the lowest validation error. To simplify the design, the number of units in each layer was determined based on a parameter $n_u$ as shown in Figure 2, which was set to 16 for architecture A and to 22 for architecture B, and was fine tuned at the end of the design process through a final grid search. The number of units in the last hidden layer of the encoder was assigned equal to $n_c$, to produce results to be compared with a PCA with the corresponding number of principal components.

## IV. Results and Discussions

### A. Principal Component Analysis

Table II shows results from PCA. The MSE obtained when fitting the PCA trained from the training set is reported for the training, validation and testing sets. Values are reported for fitting done with one, two and three principal components, and were obtained from the subsampled 40k data set (as we will show later, changing the data set size had no effect on performance for the PCA).

| $n_c$ | Training | Validation | Test |
|---|---|---|---|
| 1 | 0.0895 | 0.0998 | 0.0870 |
| 2 | 0.0630 | 0.0702 | 0.0636 |
| 3 | 0.0487 | 0.0534 | 0.0486 |

TABLE II: Mean Square Errors for PCA for the 40k subsamples dataset.

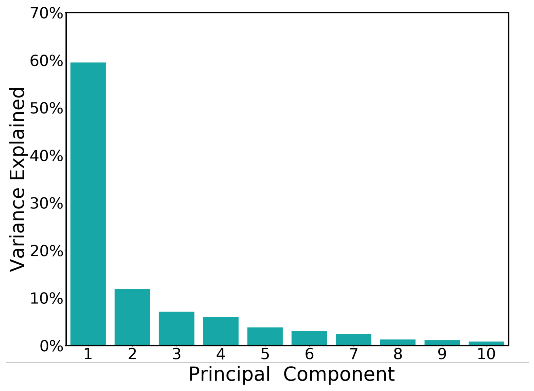


Fig. 3: Explained variance for each Principal Component.

Figure 3 shows the explained variance for the first 10 Principal Components (PCs).The first PC dominates over the others by explaining 59.4% of variance, while the second and third PC explain 11.9% and 7.1% of the variance respectively. This is in line with the results found in the literature, for example by Santello et al. (see e.g. Table 2 in [1]).

### B. Kernel Principal Component Analysis

Polynomial Kernel PCA performed better than RBF and sigmoid in every instance, and for this reason was chosen as the best case representation for using Kernel PCA. Table III shows MSE for polynomial Kernel PCA after training it on the 40k subsample of the training data. When compared with results from the PCA, polynomial KPCA always yields a lower MSE, with the difference being larger when considering three principal components and fitting the testing set (up to 25%).

| $n_c$ | Training | Validation | Test |
|---|---|---|---|
| 1 | 0.0819 | 0.0923 | 0.0787 |
| 2 | 0.0533 | 0.0625 | 0.0513 |
| 3 | 0.0375 | 0.0463 | 0.0366 |

TABLE III: Mean Square Errors for KPCA for the 40k subsamples dataset.

### C. Autoencoders

Table IV shows an overview of the results for the two autoencoder architectures considered, for the 40k subsampled dataset. Architecture B performs better than A in every case. Both architectures consistently show lower MSE than both PCA and KPCA, with up to a 50% improvement when comparing architecture B with the PCA. Table V shows a final overview of results for PCA, KPCA and autoencoders.

| $n_c$ | | Training | Validation | Test |
|---|---|---|---|---|
| 1 | A | 0.0698 | 0.0796 | 0.0675 |
| | B | **0.0586** | **0.0653** | **0.0552** |
| 2 | A | 0.0462 | 0.0507 | 0.0422 |
| | B | **0.0371** | **0.0428** | **0.0342** |
| 3 | A | 0.0341 | 0.0388 | 0.0317 |
| | B | **0.0288** | **0.0348** | **0.0262** |

TABLE IV: Mean Square Errors for the autoencoders for the 40k subsamples dataset.

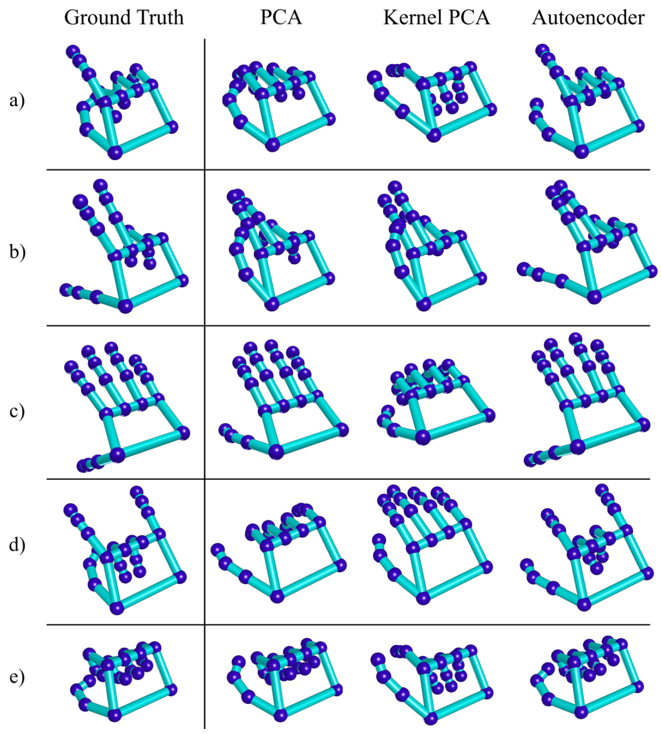


Fig. 4: Hand pose reconstructions comparison for a few hand poses (obtained with $n_c = 3$ principal components/degrees of freedom).

## D. Visual Comparison

We have shown in the previous section that both KPCA and autoencoders yield lower MSE than the PCA in every instance, with autoencoders in particular showing the highest quantitative improvement in performance. Here we will show visual reconstruction of a few hand poses for a qualitative comparison, referring to the video[3] for more views taken from the data set. Figure 4 shows reconstructed hand poses from the original data, taken for a few samples and fitted with

[3]https://www.dropbox.com/s/i6fmb6m8xeybvq4/ISMR_2021_DimensionalityReduction_video.mp4?dl=0

| | $n_c = 1$ | $n_c = 2$ | $n_c = 3$ |
|---|---|---|---|
| | Training | | |
| PCA - 40k | 0.0895 | 0.0630 | 0.0487 |
| PCA - full | 0.0892 | 0.0630 | 0.0487 |
| KPCA - 40k | 0.0819 | 0.0533 | 0.0375 |
| KPCA - full | N/A | N/A | N/A |
| AE-B - 40k | 0.0586 | 0.0371 | 0.0288 |
| **AE-B - full** | **0.0490** | **0.0357** | **0.0261** |
| | Validation | | |
| PCA - 40k | 0.0998 | 0.0702 | 0.0534 |
| PCA - full | 0.0999 | 0.0702 | 0.0532 |
| KPCA - 40k | 0.0923 | 0.0625 | 0.0463 |
| KPCA - full | N/A | N/A | N/A |
| AE-B - 40k | 0.0653 | 0.0428 | 0.0348 |
| **AE-B - full** | **0.0559** | **0.0411** | **0.0334** |
| | Test | | |
| PCA - 40k | 0.0870 | 0.0636 | 0.0486 |
| PCA - full | 0.0872 | 0.0635 | 0.0486 |
| KPCA - 40k | 0.0787 | 0.0513 | 0.0366 |
| AE-B - 40k | 0.0552 | 0.0342 | 0.0262 |
| **AE-B - full** | **0.0453** | **0.0333** | **0.0244** |

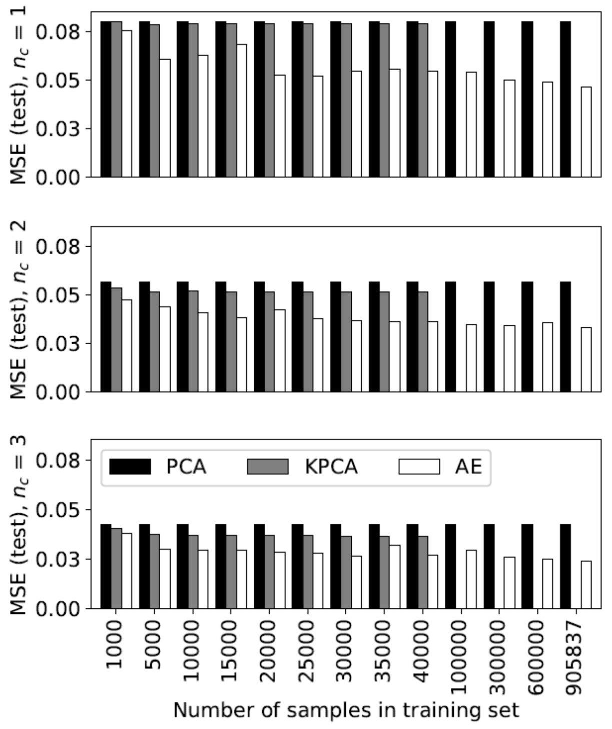
TABLE V: Final overview of MSE for each technique. Note that, as shown in Figure 5a, PCA and Kernel PCA do not increase performance with an increase in the number of samples, and that KPCA could not be applied to the full dataset because of its demanding computational load for large datasets.

$n_c = 3$ for the PCA, polynomial Kernel PCA and the autoencoder network A. It can be seen that reconstructions from the autoencoders shows a more realistic reconstructions of the shape of the original poses, while both Kernel PCA and PCA struggle to capture some of the shapes. More examples of this can be seen in the attached video. This is an important complement to the outcome of the MSE evaluation which, while representative of an overall quantitative evaluation of performance, fails to capture the effect that each joint angle has on defining the overall hand grasp shape. The difference in performance between autoencoders and the other two methods for what concerns this aspect appears remarkable. However, the video also highlights a drawback of the current implementation of the autoencoder method, which being static treats each grasp separately rather than considering them as an evolution over time. This could explain the discontinuities that can be seen when transitioning between some of the hand shapes.

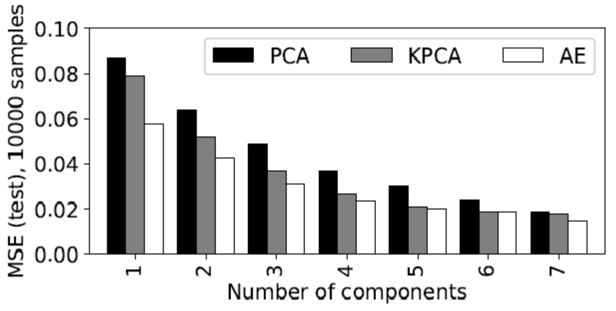
## E. Effect of Sample Size

In the previous subsections we reported results for analysis done on a subset of 40000 samples taken from the original dataset, and the outcome for the full data set was only mentioned for the autoencoder in Table V. The reason for this was that 40000 was the highest number of samples that the KPCA was able to handle, and a fair comparison requires using the different techniques on the same data set.

On the other hand, being unable to deal with larger data sets is a drawback of the KPCA when compared to other methods, and if using more data leads to an increase in



(a) MSE for $n_c$ ranging from 1 to 3 and different values of the number of samples.



(b) MSE for difference values of $n_c$, with number of samples set to 10000.

Fig. 5: Reconstruction error for different numbers of principal components/degrees of freedom ($n_c$), as function of the number of samples used for the training set.

performance, that would be a point in favor of using the other two techniques. In this section we show how the number of samples changed performance for each technique.

Figure 5a shows a detailed overview of the effect of training sample size on the MSE for the test subset, for $n_c$ varying between 1 and 3 and a sample size for the training set ranging from 1000 to 905837 (which is the size of the full set). It can be noted that there is no change in performance for the PCA with an increase in the number of samples. KPCA behaves similarly with a small improvement when increasing the number of samples from 1000 to 5000, followed by no increase in performance with further increases in the number of samples considered (up to 40k, which is the maximum that could be analyzed with this technique). Finally, autoencoders show some fluctuations with a general trend towards improvement of performance for a larger dataset. It is worth pointing out that autoencoders always outperform the other methods no matter the sample size, although the difference becomes noticeably larger as the sample size increases.

In this paper we focused in $n_c \leq 3$, which is desirable in order to keep the complexity low (e.g., in a prosthetic hand or for hand pose sensing with a reduced number of sensors [8], [23]). However, as shown in Figure 5b autoencoders outperform the other techniques for $n_c > 3$ as well. Finally, while MSE was chosen as a main metric of comparison because of the fact that it is the metric being minimized by the PCA, the improvement in performance on MSE also translated to better errors in fingertip distances ($17.2 \pm 5.3$, $14.6 \pm 4.9$ and $13.3 \pm 4.2$ average errors for PCA, Kernel PCA and autoencoders, respectively).

## V. Conclusions

In this work we present a comparison of performance for the application of Principal Components Analysis, Kernel Principal Components Analysis and autoencoders on a large data set of hand postures, described by measurements of joint angles. Results show lower Mean Square Errors from KPCA and autoencoders when compared to the traditional PCA, with a reduction of up to 25% and 50% for KPCA and autoencoders, respectively. Autoencoders in particular seem to offer more realistic reconstructions of hand shapes, as shown by the hand pose visualizer that we developed for this purpose, and show a noticeable increase of performance when using a larger dataset.

The results presented in this paper show the potential of using nonlinear techniques for dimensionality reduction of hand poses, and we hope that this can spark a larger interest from the community towards this under-explored treatment of hand kinematics. We think that the difference in the ability to reconstruct shapes shown through visual reconstruction is especially remarkable. It is worth pointing out that such difference in performance could be increased by the fact that we chose a data set rich in different grasping shapes. The simpler PCA might still be preferrable for smaller and more application specific data sets, where just a few grasp shapes are present.

Future work will focus on using autoencoders to develop a hand sensing method that uses a low number of sensors to capture the whole hand pose, inspired by the approach that was used by Ciotti et al. in [43]. Additionally, we will evaluate the robustness of our analysis by testing the fitting obtained from this data set with autoencoders to other data sets. One limitation of the autoencoder approach is the fact that it is a static technique, and as such does not take into account the evolution of hand shapes over time, potentially causing some of the discontinuities that can be seen in the video reconstruction. To try and address this we will consider a replacement of fully connected layers with Long-Short-Term Memory (LSTM) units [44], which are better at modeling time sequences.

## References

[1] M. Santello, M. Flanders, and J. F. Soechting, "Postural hand synergies for tool use," *Journal of neuroscience*, vol. 18, no. 23, pp. 10 105–10 115, 1998.

[2] C. R. Mason, J. E. Gomez, and T. J. Ebner, "Hand synergies during reach-to-grasp," *Journal of neurophysiology*, vol. 86, no. 6, pp. 2896–2910, 2001.

[3] M. Santello, M. Flanders, and J. F. Soechting, "Patterns of hand motion during grasping and the influence of sensory guidance," *Journal of Neuroscience*, vol. 22, no. 4, pp. 1426–1435, 2002.

[4] E. J. Weiss and M. Flanders, "Muscular and postural synergies of the human hand," *Journal of neurophysiology*, vol. 92, no. 1, pp. 523–535, 2004.

[5] M. Santello, G. Baud-Bovy, and H. Jörntell, "Neural bases of hand synergies," *Frontiers in computational neuroscience*, vol. 7, p. 23, 2013.

[6] C. Y. Brown and H. H. Asada, "Inter-finger coordination and postural synergies in robot hands via mechanical implementation of principal components analysis," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 2877–2882.

[7] M. Gabiccini, A. Bicchi, D. Prattichizzo, and M. Malvezzi, "On the role of hand synergies in the optimal choice of grasping forces," *Autonomous Robots*, vol. 31, no. 2-3, p. 235, 2011.

[8] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive synergies for the design and control of the pisa/iit softhand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 768–782, 2014.

[9] S. B. Godfrey, M. Bianchi, K. Zhao, M. Catalano, R. Breighner, A. Theuer, K. Andrews, G. Grioli, M. Santello, and A. Bicchi, "The softhand pro: Translation from robotic hand to prosthetic prototype," in *Converging Clinical and Engineering Research on Neurorehabilitation II*. Springer, 2017, pp. 469–473.

[10] E. Battaglia, J. P. Clark, M. Bianchi, M. G. Catalano, A. Bicchi, and M. K. O'Malley, "The rice haptic rocker: skin stretch haptic feedback with the pisa/iit softhand," in *2017 IEEE World Haptics Conference (WHC)*. IEEE, 2017, pp. 7–12.

[11] K. Mitsui, R. Ozawa, and T. Kou, "An under-actuated robotic hand for multiple grasps," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 5475–5480.

[12] K. Xu, H. Liu, Y. Du, and X. Zhu, "Design of an underactuated anthropomorphic hand with mechanically implemented postural synergies," *Advanced Robotics*, vol. 28, no. 21, pp. 1459–1474, 2014.

[13] R. Deimel and O. Brock, "A novel type of compliant and underactuated robotic hand for dexterous grasping," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 161–185, 2016.

[14] F. Ficuciello, G. Pisani, S. Marcellini, and B. Siciliano, "The prisma hand i: A novel underactuated design and emg/voice-based multimodal control," *Engineering Applications of Artificial Intelligence*, vol. 93, p. 103698, 2020.

[15] M. T. Ciocarlie and P. K. Allen, "Hand posture subspaces for dexterous robotic grasping," *The International Journal of Robotics Research*, vol. 28, no. 7, pp. 851–867, 2009.

[16] T. Geng, M. Lee, and M. Hülse, "Transferring human grasping synergies to a robot," *Mechatronics*, vol. 21, no. 1, pp. 272–284, 2011.

[17] T. Wimböck, J. Reinecke, and M. Chalon, "Derivation and verification of synergy coordinates for the dlr hand arm system," in *2012 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2012, pp. 454–460.

[18] G. Gioioso, G. Salvietti, M. Malvezzi, and D. Prattichizzo, "An object-based approach to map human hand synergies onto robotic hands with dissimilar kinematics," in *Robotics: Science and Systems VIII*. The MIT Press Sydney, NSW, 2012, pp. 97–104.

[19] F. Ficuciello, G. Palli, C. Melchiorri, and B. Siciliano, "Postural synergies of the ub hand iv for human-like grasping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 515–527, 2014.

[20] G. Salvietti, "Replicating human hand synergies onto robotic hands: A review on software and hardware strategies," *Frontiers in neurorobotics*, vol. 12, p. 27, 2018.

[21] R. Garcia-Rosas, D. Oetomo, C. Manzie, Y. Tan, and P. Choong, "On the relationship between human motor control performance and kinematic synergies in upper limb prosthetics," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2018, pp. 3194–3197.

[22] M. K. Burns, K. Van Orden, V. Patel, and R. Vinjamuri, "Towards a wearable hand exoskeleton with embedded synergies," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2017, pp. 213–216.

[23] M. Bianchi, P. Salaris, and A. Bicchi, "Synergy-based hand pose sensing: Reconstruction enhancement," *The International Journal of Robotics Research*, vol. 32, no. 4, pp. 396–406, 2013.

[24] Z. Wang and A. M. Fey, "Deep learning with convolutional neural network for objective skill evaluation in robot-assisted surgery," *International journal of computer assisted radiology and surgery*, vol. 13, no. 12, pp. 1959–1970, 2018.

[25] O. Glauser, S. Wu, D. Panozzo, O. Hilliges, and O. Sorkine-Hornung, "Interactive hand pose estimation using a stretch-sensing soft glove," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–15, 2019.

[26] E. Biddiss and T. Chau, "Upper-limb prosthetics: critical factors in device abandonment," *American journal of physical medicine & rehabilitation*, vol. 86, no. 12, pp. 977–987, 2007.

[27] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[28] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[29] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[30] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *International conference on artificial neural networks*. Springer, 1997, pp. 583–588.

[31] A. Ghodsi, "Dimensionality reduction a short tutorial," *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, vol. 37, no. 38, p. 2006, 2006.

[32] B.-j. Kim, "Active visual learning and recognition using incremental kernel pca," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2005, pp. 585–592.

[33] T.-J. Chin and D. Suter, "Incremental kernel principal component analysis," *IEEE transactions on image processing*, vol. 16, no. 6, pp. 1662–1674, 2007.

[34] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018.

[35] A. Tkach, A. Tagliasacchi, E. Remelli, M. Pauly, and A. Fitzgibbon, "Online generative model personalization for hand tracking," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 6, pp. 1–11, 2017.

[36] J. Romero, T. Feix, C. H. Ek, H. Kjellström, and D. Kragic, "Extracting postural synergies for robotic grasping," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1342–1352, 2013.

[37] J. Starke, C. Eichmann, S. Ottenhaus, and T. Asfour, "Synergy-based, data-driven generation of object-specific grasps for anthropomorphic hands," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 327–333.

[38] F. Conti, F. Barbagli, R. Balaniuk, M. Halg, C. Lu, D. Morris, L. Sentis, J. Warren, O. Khatib, and K. Salisbury, "The chai libraries," in *Proceedings of Eurohaptics 2003*, Dublin, Ireland, 2003, pp. 496–500.

[39] B. Alexander and K. Viktor, "Proportions of hand segments," *International Journal of Morphology*, vol. 28, no. 3, pp. 755–758, 2010.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] S. Ciotti, E. Battaglia, N. Carbonaro, A. Bicchi, A. Tognetti, and M. Bianchi, "A synergy-based optimally designed sensing glove for functional grasp recognition," *Sensors*, vol. 16, no. 6, p. 811, 2016.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.