A Principal Component Analysis (PCA) Based Method for Shape Extraction

Leontin TUȚĂ, Ioan NICOLAESCU, Radu MARIESCU-ISTODOR, and Angela DIGULESCU-POPESCU

***Abstract*—This paper proposes an algorithm that uses principal component analysis (PCA) in order to extract some of the most commonly used shapes in different design or drawing based applications. Other methods are also discussed in comparison to PCA, the advantages and disadvantages being underlined in each case. Simulations are performed in Matlab for a number of scenarios, with the purpose to test the method's reliability in different cases, especially unfavorable ones. A concept for generalizing the functionality of the algorithm is also presented, and the limitations of this concept are discussed.**

***Index Terms*—PCA, shape extraction, covariance, noise robustness, eigenvector**

I. Introduction

Principal component analysis (PCA) is a processing technique generally used to reduce the dimensions of a measured dataset, by maximizing the variance of the dataset across several dimensions. This is done by means of projecting the data into a new multidimensional space, defined by a number of orthogonal axes, each representing one of the datasets' principal components. Then, having the variance maximized across all dimensions, a decision can be made to discard a number of axes from the representation, more specifically, those which introduce the least variance. If we consider each of the measurements as a linear combination of these orthogonal variables, the dimensionality reduction would be equivalent to removing from the initial measured data those terms that include the variables having the smallest coefficient. Thus, if we project the dimensionally reduced data into the original space, we obtain a filtered version of the dataset, composed of the most significant principal components.

PCA is used in a variety of domains, from nanobiology to a variety of applications in signal processing. In the later domain, PCA proves to be extremely useful in filtering signals affected by noise or reducing redundancy within a measured dataset.

Our purpose is to use PCA to identify certain shapes affected by noise, more specifically to extract an intelligible version of that shape from the real, measured version. The algorithm attempts to the extract the best possible version of the measured shape it can. The shapes in question are ellipses, circles and lines. Though this may seem limited, an argument is made regarding the proposed method's capacity to extract more complex shapes.

One motivation for this is the growing tendency in mobile apps and wireless devices of using shapes to express a different form a media, generally text. Some of these commercial applications include security access by means of finger drawn doodles, mobile games based on finger drawings, or console games that require the movement of a remote controller [1]. Also, more and more specialists in architecture or art are using wireless styluses or markers and digital applications for their 2D or 3D designs. As such, enabling autocorrect functionality for certain shapes could prove practical. In radar imaging, PCA can be used twofold. First, to filter out the unwanted clutter frequency components [2], which produces a better radar image, and after target detection and focalization, PCA can be used to estimate the shape of the target and, if possible, identify it from a number of known options [3].

Other processing methods which can be used for shape extraction are also discussed in this paper, with the purpose of comparing their functionality and to underline the many advantages and few disadvantages offered by the PCA based implementation. Finally, we present simulation results for different scenarios and draw conclusions based on these results.

II. Theoretical Framework

The shapes covered in this paper are ellipses, with their particular cases: circle and line. The most common 2D representation of an ellipse is the canonical form, meaning that the semi-axes are positioned along the coordinate axes and the center is positioned in the origin. The equations that give the coordinate vectors for this representation are:

$$\begin{cases} \mathbf{x_{canon}} = a\cos(\boldsymbol{\varphi}) \\ \mathbf{y_{canon}} = b\sin(\boldsymbol{\varphi}) \end{cases} \tag{1}$$

where 'a' and 'b' are the semi-major and semi-minor axis lengths, and '$\boldsymbol{\varphi}$' varies from 0 to 2π.

For $a = 0$ or $b = 0$, the ellipse becomes a line. For $a = b$, the ellipse becomes a circle and the equal semi-axes both become radii of the circle. So the coordinates of an ideal circle would be, in a 2D representation, a cosine for the abscissa and a sine for the ordinate, both multiplied by the radius, and having an offset given by the center of the drawn circle [4]. The sine and cosine can be interchanged, the effect being a shift in the starting point of the circle. Instead of being on the rightmost point, it will be on the top point of the circle, because the cosine will correspond to the

L. TUȚĂ is with the Military Technical Academy "Ferdinand I", Bucharest, Romania (e-mail: leontin.tuta@mta.ro).

I. NICOLAESCU is with the Military Technical Academy "Ferdinand I", Bucharest, Romania (e-mail: ioan.nicolaescu@mta.ro).

R. MARIESCU-ISTODOR is with the University of Eastern Finland, Joensuu, Finland (e-mail: radum@cs.joensuu.fi).

A. DIGULESCU-POPESCU is with the Military Technical Academy "Ferdinand I", Bucharest, Romania (e-mail: angela.digulescu@mta.ro).

ordinate, and the sense will also reverse to clockwise. Actually any change of the initial phase with a quantity of φ_0, for both the sine and cosine (same change) will shift the starting point. However, the constant $\Delta\varphi_0 = \frac{\pi}{2}$ difference between the sine and cosine must stay the same to preserve the circle shape. If $\Delta\varphi_0$ is changed, then we obtain an ellipse, more specifically a Lissajous ellipse, and not the particular case of a circle. Furthermore, if $\Delta\varphi_0$ is null or a multiple of π, the minor semi-axis of the ellipse will have a length of 0, and the ellipse will become a line. It is intuitive that a sine represented with respect to the same sine (null phase difference) will result in a linear variation with a positive trend (if $\Delta\varphi_0$ is an even multiple of π). The slope of the trend will be $\frac{\pi}{4}$, as it happens with the semi-axis of all Lissajous ellipses. We can write the coordinate vectors corresponding to the any Lissajous ellipse as the following:

$$\begin{cases} \mathbf{x} = r\sin\left(\boldsymbol{\varphi} + \varphi_0\right) \\ \mathbf{y} = r\sin\left(\boldsymbol{\varphi} + \varphi_0 + \Delta\varphi_0\right) \end{cases} \tag{2}$$

where $\boldsymbol{\varphi}$ varies from 0 to 2π, φ_0 corresponds to the change in the starting point position, $\Delta\varphi_0$ gives the ratio of the semi-axes' lengths for the ellipse, as previously discussed, and r increases or decreases the overall size of the ellipse. Because φ_0 is only responsible for shifting the starting point, while not changing in any way the shape or the point distribution of the ellipse, we will ignore it in future equations. Also, for simplicity, we will consider that $\Delta\varphi_0$ only varies within the $\left[0, \frac{\pi}{2}\right]$ interval, because we're only interested in the parameters ability to change the ratio between the semi-axes.

The slope of the semi-axes can be changed by a rotation performed on the data, which is done by multiplying the coordinate matrix composed of the vectors from (1) with a rotation matrix:

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \tag{3}$$

where θ gives the counterclockwise rotation of the ellipse, and $\mathbf{x}'$ and $\mathbf{y}'$ are the new coordinate vectors, after the rotation is performed. If r and $\Delta\varphi_0$ control the size and shape, θ controls the orientation of the ellipse. A particular case here is the clockwise rotation with $\frac{\pi}{4}$. For this rotation, the Lissajous ellipse from (2) should become a canonically represented ellipse. The coordinate vectors after the $\frac{\pi}{4}$ rotation are:

$$\begin{bmatrix} \mathbf{x}' \\ \mathbf{y}' \end{bmatrix} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} r\sin(\boldsymbol{\varphi}) \\ r\sin(\boldsymbol{\varphi} + \Delta\varphi_0) \end{bmatrix} \tag{4}$$

After performing multiplications, we obtain:

$$\begin{cases} \mathbf{x}' = \frac{\sqrt{2}}{2} r\left[\sin\Delta\varphi_0 \cos(\boldsymbol{\varphi}) + (1 + \cos\Delta\varphi_0)\sin(\boldsymbol{\varphi})\right] \\ y' = \frac{\sqrt{2}}{2} r\left[\sin\Delta\varphi_0 \cos(\boldsymbol{\varphi}) - (1 - \cos\Delta\varphi_0)\sin(\boldsymbol{\varphi})\right] \end{cases} \tag{5}$$

Further computation leads to:

$$\begin{cases} \mathbf{x}' = \sqrt{2}\cos\frac{\Delta\varphi_0}{2} r\left[\sin\frac{\Delta\varphi_0}{2}\cos(\boldsymbol{\varphi}) + \cos\frac{\Delta\varphi_0}{2}\sin(\boldsymbol{\varphi})\right] \\ \mathbf{y}' = \sqrt{2}\sin\frac{\Delta\varphi_0}{2} r\left[\cos\frac{\Delta\varphi_0}{2}\cos(\boldsymbol{\varphi}) - \sin\frac{\Delta\varphi_0}{2}\sin(\boldsymbol{\varphi})\right] \end{cases} \tag{6}$$

or

$$\begin{cases} \mathbf{x}' = r \cdot \sqrt{1 + \cos\frac{\Delta\varphi_0}{2}} \sin\left(\boldsymbol{\varphi} + \frac{\Delta\varphi_0}{2}\right) \\ \mathbf{y}' = r \cdot \sqrt{1 - \cos\frac{\Delta\varphi_0}{2}} \cos\left(\boldsymbol{\varphi} + \frac{\Delta\varphi_0}{2}\right) \end{cases} \tag{7}$$

As previously mentioned, adding the same phase difference to both the sinusoidal variations that give the coordinates will only result in shifting the starting point of the ellipse, without changing its shape whatsoever. So, by adding a constant value of $\frac{\pi}{2} - \frac{\Delta\varphi_0}{2}$ to both the phase of $\mathbf{x}'$ and the phase of $\mathbf{y}'$, we obtain:

$$\begin{cases} \mathbf{x}' = r \cdot \sqrt{1 + \cos\frac{\Delta\varphi_0}{2}} \cos(\boldsymbol{\varphi}) \\ \mathbf{y}' = r \cdot \sqrt{1 - \cos\frac{\Delta\varphi_0}{2}} \left[-\sin(\boldsymbol{\varphi})\right] \end{cases} \tag{8}$$

This final form of the coordinate vectors confirms, by comparison with (1), that indeed we obtain an ellipse in its canonical form after performing a quarter quadrant rotation on a Lissajous ellipse. The –1 factor present in the sine's argument only impacts the sense in which the points of the ellipse are represented, without affecting the shape or point distribution. In this case, the points will be represented in the clockwise sense. Also, if $\Delta\varphi_0$ is the phase difference between the sines that shape the ellipse according to (2), then the lengths of the semi-major and semi-minor axes will be, according to (1) and (8):

$$\begin{cases} a = r \cdot \sqrt{1 + \cos\frac{\Delta\varphi_0}{2}} \\ b = r \cdot \sqrt{1 - \cos\frac{\Delta\varphi_0}{2}} \end{cases} \tag{9}$$

Therefore, we can either represent an ellipse in a straightforward manner by knowing the lengths of the semi-axes or by knowing the overall size r and the phase difference $\Delta\varphi_0$ between the sines that give the coordinate vectors.

The advantage of the Lissajous representation from (2) is that it can easily be generalized to a multidimensional space. A 3D ellipse simply requires an additional coordinate vector that differs from the previous two by having a different phase difference [5]:

$$\mathbf{z} = r\sin\left(\boldsymbol{\varphi} + \Delta\varphi_{0z}\right) \tag{10}$$

So, taking into account these final considerations, and after substituting (2) in (3) and performing multiplications, we can express in a general form the abscissa and ordinate vectors corresponding to any representation of a 2D ellipse:

$$\begin{cases} \mathbf{x_g} = x_0 + r\left[\cos(\theta)\sin(\boldsymbol{\varphi}) - \sin(\theta)\sin(\boldsymbol{\varphi} + \Delta\varphi_0)\right] \\ \mathbf{y_g} = y_0 + r\left[\sin(\theta)\sin(\boldsymbol{\varphi}) + \cos(\theta)\sin(\boldsymbol{\varphi} + \Delta\varphi_0)\right] \end{cases} \tag{11}$$

where (x_0, y_0) are offsets that give the center of the ellipse, and the other parameters have been previously defined.

III. PCA Standard Algorithm

Analyzing a great number of interdependent variables means processing a huge dataset, more specifically a matrix of huge dimensions. Thus, a dimensionality problem can be inferred, meaning that by increasing the size of the input data matrix, we unintentionally lower the system's capacity to efficiently manage and process the data. From an experimental point of view, in order to analyze the underlying functionality of a system, we wish to measure different quantities or features of that system. To view how dependent these quantities are to one another, they can be plotted with respect to one another, resulting a multidimensional representation. So, by adding new dimensions, as an effect of registering observations of new quantities, the average distance between the represented points increases. Though this can help in some situations by means of reducing the overall effect of noise, it makes predicting the correct underlying variation of the data increasingly difficult. Furthermore, some of the features may not even contribute enough to the variance of the dataset. These quantities not only do not improve the analyzed model, but they act similar to an additive noise over the useful data. Of course, the final disadvantage to adding more input data is the increase in processing resources and latency.

However, we know from information theory that by increasing the number of measurements, the output data will be more refined, considering the additional inputs are not redundant and the processing is performed efficiently. The problem here is that we do not always have full control of eliminating redundancy while measuring data, and the obvious solution is to use processing techniques to do so. One such technique is principal component analysis, which functions as a critical component in any application based on analytical processes, especially those that use datasets of extremely large dimensions. Not only is PCA capable of identifying the noise within the data, but it also filters it.

Let's consider '*m*' measurements performed to determine the values of '*n*' quantities. In order to organize the data, each of the system's measured quantities or features will represent a column in an $m \times n$ matrix called '**M**'. The features can also be placed on the lines of the matrix, and the PCA algorithm will this give correct results.

$$\mathbf{M} = \left[\mathbf{f_1} \quad \mathbf{f}_1 \quad ... \quad \mathbf{f_n}\right] \tag{12}$$

where

$$\mathbf{f_i} = \begin{bmatrix} f_i[1] \\ f_i[2] \\ ... \\ f_i[m] \end{bmatrix}, i = \overline{1...n} \tag{13}$$

is the *i*-th vector of features.

The first step of implementing the PCA is centering the data, meaning that the mean must be subtracted from each feature column vector. The reason for this is to ensure that the resulting components will only seek to improve the variance of the data, without taking into account the mean as a significant variable. If data centering is not performed, there is a risk that the first and most significant principal component determined with PCA will correspond to the actual mean of the dataset, and not the direction of maximum variance. Another straightforward reason to perform centering is that PCA is a method based on maximizing variance, and variance cannot be of course computed correctly without extracting the mean from the data.

$$\mathbf{M_c} = \left[\mathbf{f_1} - \overline{\mathbf{f_1}} \quad \mathrm{f}_2 - \overline{\mathbf{f_2}} \quad ... \quad \mathbf{f_n} - \overline{\mathbf{f_n}}\right] \tag{14}$$

The covariance matrix is computed afterwards. This will contain the covariances between every pair of column vectors from the centered data matrix.

$$\mathbf{C} = \frac{1}{m-1}\left(\mathbf{M_c}^H \cdot \mathbf{M_c}\right) \tag{15}$$

where $.^H$ denotes the Hermitian (complex conjugate) of a matrix.

The diagonal of the covariance matrix corresponds to the variance of each feature vector, while the values above and under the main diagonal are complex conjugate to one another, being positioned symmetrically with respect to the main diagonal. Because of its structure, the covariance matrix is a symmetric matrix, so it can be diagonalized, meaning that it can be decomposed in a product of matrices based on the eigenvectors and eigenvalues computed from C. Multiplying any of the eigenvectors with the covariance matrix, is equivalent to multiplying the same eigenvector with a scalar, which is the corresponding eigenvalue. Also, because the eigenvectors of the covariance matrix are orthonormal, we can write, in matrix form, the following:

$$\mathbf{C} \cdot \mathbf{V} = \mathbf{V} \cdot \mathbf{D} \Longleftrightarrow \mathbf{C} = \mathbf{VDV}^H \tag{16}$$

where

$$\mathbf{V} = \begin{bmatrix} v_1[1] & v_2[1] & ... & v_n[1] \\ v_1[2] & v_2[2] & ... & v_n[2] \\ ... & ... & ... & ... \\ v_1[n] & v_2[n] & ... & v_n[n] \end{bmatrix} \tag{17}$$

and matrix

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & \lambda_n \end{bmatrix} \tag{18}$$

contains the eigenvalues on the main diagonal.

The eigenvectors define the axes from the new space in which the data is projected, while the eigenvalues indicate the spread of the dataset along each of these axes. The first eigenvector, which corresponds to the greatest eigenvalue will thus correspond to the greatest variance of the projected dataset. Each of the following eigenvectors will be orthogonal with the previous.

The eigenvalues can be used as a sorting criterion for the eigenvectors. The later will be rearranged according to the descending sorting of the corresponding eigenvalues. The first resulting eigenvectors will be the ones that maximize the most the variance of the dataset. The last ones will correspond to the noise present within the data. One can retain a maximum number of $n-1$ principal components out of the total 'n' eigenvectors, though usually, most of the variance resides within the first few eigenvectors. Let's consider the chosen eigenvectors:

$$\mathbf{v}_S = \left[\mathbf{v_{s1}} \quad \mathbf{v_{s2}} \quad \cdots \quad \mathbf{v_{sk}}\right], \tag{19}$$

where $\mathbf{v_{s,i}}, i = \overline{1...k}$ represent the column eigenvectors arranged according to the descending sortation of the eigenvalues. Out of all the eigenvectors, we assume only the first '*k*' are retained, where the value of '*k*' results from:

$$\sum_{i=1}^{k-1} \lambda_{s,i} < p \cdot \sum_{i=1}^{n} \lambda_{s,i} < \sum_{i=1}^{k} \lambda_{s,i} \tag{20}$$

$\lambda_{s,i}, i = \overline{1...n}$ represents the vector of eigenvalues sorted in a descending manner and '*p*' is a subunitary positive value. We can say that '*p*' is a percentage out of the total sum of all eigenvalues represented by only the first '*k*'.

Another option is to simply specify the value '*k*', if we know from the start how many principal components are needed to process the data correctly. This is only feasible though in situations where we have some information regarding the variation of the signals of interest.

After identifying the principal components, the next step of the PCA algorithm is projecting the dataset in the new space defined by the principal components. In order to convert the original values, a projection matrix is formed, containing the chosen eigenvectors on its columns. By multiplying the original centered dataset with the projection matrix, we obtain the projected dataset:

$$\mathbf{M_p} = \mathbf{M_c} \cdot \mathbf{V_s} \tag{21}$$

An example of the spatial projection caused by PCA can be viewed in Figures 1 and 2. Choosing two principal components results in obtaining a 2D projection. The input data used for PCA processing consists of measurements from the "hald.mat" dataset available in Matlab.

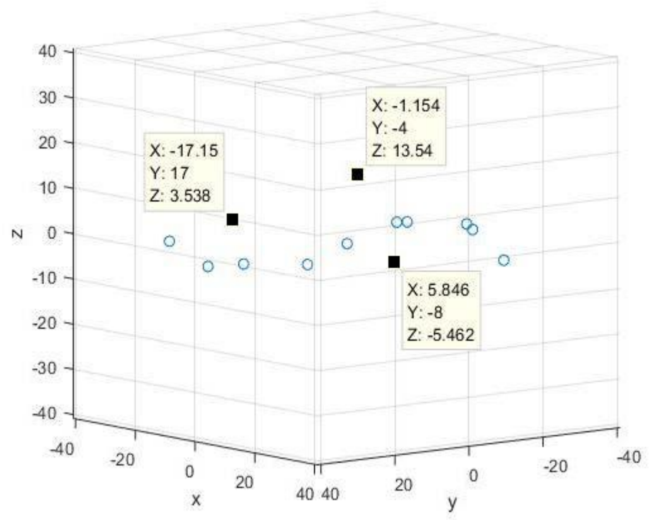

Figure 1. Original dataset with 3 features scattered in a 3D space

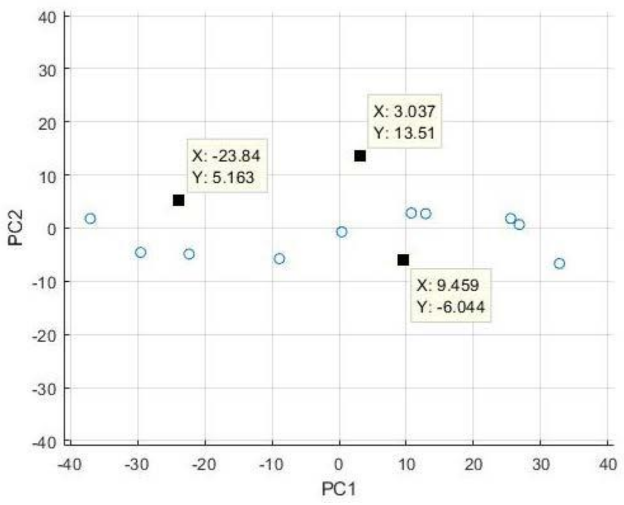

Figure 2. Dataset from Figure 1 projected in a 2D space with axes corresponding to the first two principal components

The algorithm may end here if the projected vectors are considered the signals of interest for the application. This would be the case if we would only want to view the most significant signals that generate any of the features as a result of a linear combination. However, this is not the case for the application which we later propose. After retaining only a selection of eigenvectors, return to the original space is possible by means of multiplying the projected dataset with the Hermitian of the projection matrix:

$$\mathbf{M_f} = \mathbf{M_p} \cdot \mathbf{V_s}^H = \mathbf{M_c} \cdot \mathbf{V_s} \cdot \mathbf{V_s}^H \tag{22}$$

If all eigenvectors would have been chosen as columns of the projection matrix, then the final dataset would have been identical with the original. However, by choosing only those which maximize the variance of the projected data, the final dataset represented in the original coordinate system will be filtered from the noise that affected it. After this is done, the feature means initially extracted from the data may be introduced again as offsets, if a comparison with the original measured data is necessary.

We can state, according to (22), that the final processed features are indeed linear combinations of the principal components, while the features from the original dataset were linear combinations of all eigenvectors, including those that corresponded to noise. The fact that the principal components are linear combinations of the measured features, is also true, according to (21). The coefficients are given in each case by the elements of the eigenvectors.

IV. Shape Extraction Using PCA

Let's assume a 3D designer is drawing an ellipse in space, with a wireless stylus or marker. The first issue is that, because the shape is drawn by hand, some imperfections may appear, that could ideally be smoothed out [6]. Another more important problem is that, because it's drawn in 3D space, the ellipse represented within the design will not be bidimensional. This means that, again, because of hand drawing, the points that form the circle will most likely not belong to the same plane. We can consider this an involuntary 3D noise [7]. Both these issues are within the scope of PCA and can be easily corrected by means of principal component extraction. This problem can be reduced, after some initial processing, to filtering the noise from the sines that shape the 2D or 3D ellipse or circle.

The input variables will be the coordinates of the points sampled from the shape drawn with the wireless stylus. If the user is using a 2D canvas, two input vectors of positions will be measured. Another vector of positions will be added in the case of a 3D design. These input vectors act as the features mentioned in the previous section, while the measurements are simply the positions in space along each axis. Regarding the output, we want to improve the shape, by making it resemble, in this case, a close to perfect ellipse, or at least an intelligible version of the desired ellipse.

So, generally, two coordinate vectors will form the input of our PCA based shape extraction algorithm. We can then form a matrix with ‘*m*’ lines and two columns, ‘*m*’ being the total number of measurements that form the drawn ellipse. Each measurement, present on one of the matrix’s lines, will represent the abscissa and ordinate of a registered point. According to (11), the general expressions for these coordinates are:

$$\begin{cases} \mathbf{x_{measured}} = x_0 + r\left[-\sin(\theta)\sin(\Delta\varphi_0)\cos(\boldsymbol{\varphi}) + \right. \\ \quad \left. +\left(\cos(\theta) - \sin(\theta)\cos(\Delta\varphi_0)\right)\sin(\boldsymbol{\varphi})\right] + \mathbf{n_1} \\ \mathbf{y_{measured}} = y_0 + r\left[\cos(\theta)\cos(\Delta\varphi_0)\sin(\boldsymbol{\varphi}) + \right. \\ \quad \left. +\left(\sin(\theta) + \cos(\theta)\cos(\Delta\varphi_0)\right)\sin(\boldsymbol{\varphi})\right] + \mathbf{n_2} \end{cases} \tag{23}$$

where $\mathbf{n_1}$ and $\mathbf{n_2}$ are the noise vectors that affect the sines and the shape the ellipse.

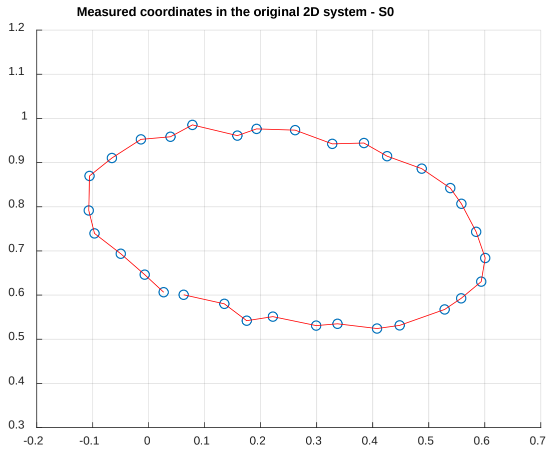

Figure 3. Noisy ellipse measured by registering its coordinate vectors

An ellipse formed by coordinate vectors such as the ones defined in (23) is presented in Figure 3. The $m = 32$ samples are registered in the measurement space, which we will call S_0. Assuming the drawing speed in constant, the measured points will be uniformly distributed along the outline, with small variations caused by noise.

By centering the data, computing the covariance matrix and performing eigendecomposition, we can project the existing measured ellipse, in a plane with new orthogonal axes, so that the variance of the data is maximized along each axis. In this case, the projected data $\mathbf{M_{p,1}}$ given by:

$$\mathbf{M_{p,1}} = \left[\mathbf{x_{measured}} - x_0 \quad \mathbf{y_{measured}} - y_0\right]\cdot\mathbf{V_1} \tag{24}$$

will be an ellipse with the semi-axes positioned along the new coordinate axes. A 2D PCA projection from the original measurement space S_0 to the new 2D coordinate system S_1 will be equivalent to a rotation of the centered data. Of course, both eigenvectors of the 2 × 2 covariance matrix will be used for the projection matrix $\mathbf{V_1}$. This simplifies the problem a lot, by completely eliminating the dependency the data had with respect to the rotation angle θ. Also, centering the data eliminates the offset introduced by x_0 and y_0 and positions the ellipses’ center in the origin of the coordinate system. The PCA transformation is presented in Figure 4.

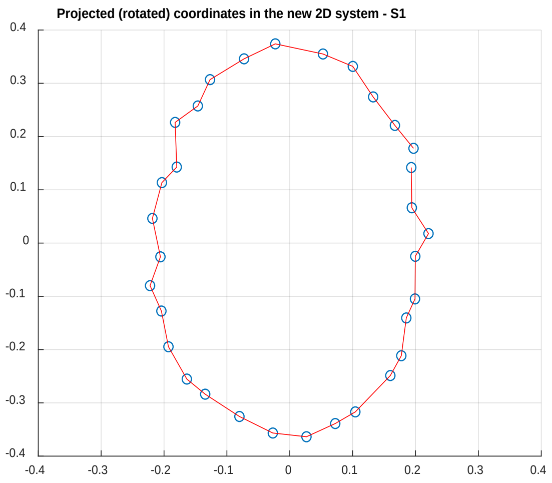

Figure 4. PCA projection of the data from S_0 to S_1

This representation, with the semi-axes positioned along the coordinate axes and the center positioned in the origin is the canonical 2D representation of an ellipse. In this form, the ellipse should be the easiest to process. According to (8), the two columns of $\mathbf{M_{p,1}}$, the projected coordinates, should have the following expressions:

$$\begin{cases} \mathbf{x_{p,1}}[i] = r\cdot\sqrt{1+\cos\dfrac{\Delta\varphi_0}{2}}\cos(\boldsymbol{\varphi}) + \mathbf{n_{p,1}}[i] \\ \mathbf{y_{p,1}}[i] = r\cdot\sqrt{1-\cos\dfrac{\Delta\varphi_0}{2}}\sin(\boldsymbol{\varphi}) + \mathbf{n_{p,2}}[i] \end{cases} \tag{25}$$

where $\mathbf{n_{p,1}}$ and $\mathbf{n_{p,2}}$ correspond to the projected noise. These noise vectors suffer the same transformation as the useful data, so even though its values change, it has the same unwanted variation relative to the ellipse’s shape.

One option here would be to estimate the ellipse’s geometrical parameters while in this simplified form. Determining the difference between the maximum and minimum values of vector $\mathbf{x_{p,1}}$ should give twice the length of the semi-major axis. Similarly, the semi-minor axis length should result from evaluating $\mathbf{y_{p,1}}$. From (9), the value of *r* can be computed by using Pythagoras’ Theorem, it being the hypotenuse, with the semi-axes being the catheti. Then, the phase difference $\Delta\varphi_0$ can be computed using either of the two equations from (9), or it can be determined using both equations, and averaging the results. An ideal ellipse can be represented based on these values. The corresponding coordinate vectors would form, in this case, a new, ideal version, of the projected data matrix. Finally, by multiplying this coordinate matrix with $\mathbf{V_1}^H$, we can return to the original measurement plane S_0, and obtain a corrected version of the original data. The problem with this method is that it can give highly erroneous results for significant levels of noise. The error can be generally minimized by taking into account data from more points in order to better estimate the geometrical parameters. However, if there are amplitude spikes along the represented shape, the values of the parameters can be even more erroneous. Also, the final result will not be the actual processed data of the original measurements, but rather a shape (data) generated based on estimations, that functions as a replacement for the original.

By now, we have transformed the data in a manner that should simplify the processing algorithm. What remains is to filter the noise from the canonically represented ellipse. This is equivalent to filtering the noise from its corresponding Lissajous ellipse or eliminating the noise from both the sines that form the corresponding coordinate vectors.

The idea is based on the fact that the abscissa and ordinate column vector that form $\mathbf{M}_{\mathbf{p,1}}$ are linear combinations of a sine and cosine, both having the same values for the instantaneous phase $\boldsymbol{\varphi}$. In addition, the vectors have added noise. These linear combinations can be observed in (25) and even in the initial representation of the ellipse in equation (23). We want the final data to be just a linear combination between the sine and cosine, without the noise. By using PCA, we know, according to (21), that the projected data will be formed of linear combinations of the original features. Also, the final processed features, obtained by returning to the original space after principal component extraction, will be linear combinations of the projected data, according to (22).

The solution to the noise filtering problem would be then to force the projected data to just contain the components corresponding to the sine and cosine and not the noise. This means forcing the projection to be a circle or noiseless ellipse. The sine and cosine are orthogonal to each other, so components are needed for both. Of course, additional data vectors must be added, because we want to extract two principal components of interest, and have other noise eigenvectors that we can exclude, while only originally having two sets of features. These additional data vectors should act like ideal, reference features, more specifically like a desired version of the sine and cosine we want to obtain, without having any added noise:

$$\begin{cases} \mathbf{x_{ref}}[i] = A\cos(\omega_n i) \\ \mathbf{y_{ref}}[i] = B\sin(\omega_n i) \end{cases} \tag{26}$$

where $i = \overline{0...\left(\frac{1}{f_n}-1\right)}$ are the sample indices for all the points corresponding to a period of the sines, $f_n = \frac{f}{f_s} = \frac{1}{m}$ is the normalized frequency, f_s is the sample rate, m is the number of samples within a sine period or the number of measurements that correspond to a fully drawn ellipse, and $\omega_n = 2\pi f_n$ is the normalized angular frequency.

In order to generate the two signals, we need to know their phase and amplitude. Regarding the reference signals' phases, we must take into account the fact that the samples are registered from real positions. This means the number of values within a period will depend on the sample rate and the speed at which the shape is drawn, which we assume to be constant. If this is not the case, interpolation can be performed on the measured data to even out the initially measured points. Interpolation of the data should be required regardless of the chosen processing method, but in this specific case, it can be easier to implement thanks to principal component analysis, which projects the ellipse to its canonical form [8]. Also, the fact that the sampled points are uniformly spaced due to the measuring process means that the phase will not follow a linear variation with a constant step [9]. Therefore, the point distribution of the measured shape and that of our generated circle/ellipse from (26) will differ. Though this is not a problem for a circle, in the general case of an ellipse, a constant phase step such as the one present in (26) will generate fewer points, with greater separation, near the semi-minor axis of an ellipse, and more points, that are closer to each other, near the semi-major axis. In order for our method to work, the 'm' phases of the generated signals must correspond to the 'm' values of vector $\boldsymbol{\varphi}$ from (23) or (25). Figure 5 presents the difference between an ellipse generated theoretically based on a sine and cosine with linear phase, and a measured ellipse that has a uniform point distribution.

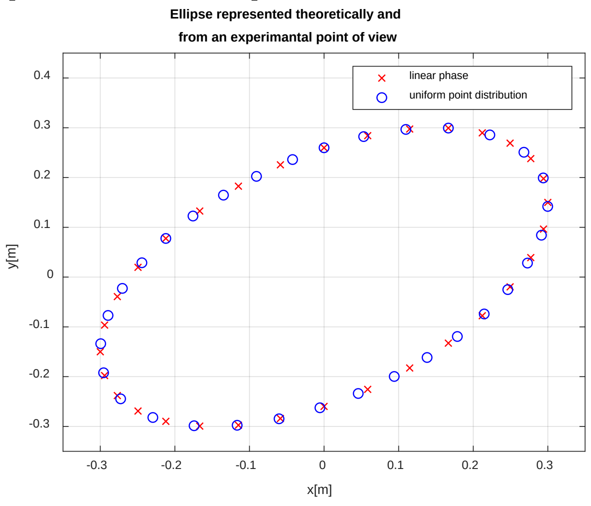

Figure 5. Ellipse measured by registering its coordinate vectors

The required nonlinear phase values can be computed using the following method. First, the semi-axes' lengths can be estimated by evaluating the maximum values of the coordinate vectors $\mathbf{x_{p,1}}$ and $\mathbf{y_{p,1}}$ that correspond to the ellipse projected (rotated) onto S_1. More precise values can be however obtained by generating two length vectors that contain the distances from the origin to the first half and second half of the points that form the outline of the projected ellipse. By adding up these two vectors, the semi-axes' lengths can be determined. The maximum value of the sum vector will correspond to twice the length of the semi-major axis. The semi-minor axis should be at $m/4$ points apart from the semi-major axis, 'm' being the total number of points. So by adding $\pm m/4$ to the index that gives the maximum in the sum vector, we should obtain twice the length of the semi-minor axis. It's recommended to use the semi-major axis as a reference as opposed to independently estimating the length of the semi-minor axis. The reason is that the signal-to-noise ratio will be the lowest near the semi-major axis and the results will have greater precision. Afterwards, an ellipse is generated using these lengths as amplitudes, and a very low value for the normalized frequency, for example $\frac{1}{m^2}$. The ellipse will have a constant phase step, but it will include a much greater number of points, m^2, enough to allow us to choose those that correspond to the phases required for our reference signals. The perimeter of this ellipse can be determined with very good precision using one of Ramanujan's formulas, without having to compute an elliptic integral:

$$P \simeq \pi\left[3\left(a_e + b_e\right) - \sqrt{\left(3a_e + b_e\right)\left(a_e + 3b_e\right)}\right] \tag{27}$$

where a_e and b_e are the estimated semi-axes' lengths.

Knowing that there should be 'm' total points, we can determine the distance any two adjacent points should have so that the ellipse would have a uniform point distribution. Of course, this value is $\frac{P}{m}$. So, from the total number of m^2 points, we choose, as precisely as we can, those 'm' points that are distanced by the mentioned value. Then, the phases corresponding to each of these points should be the ones we require. A method similar to this can be used to interpolate the data in case of point scarcity within certain regions of the drawn shape.

The amplitudes of the two reference signals must be much greater than the semi-axes lengths, with the added noise taken into consideration. The reason is that the PCA 2D projection of the current 4D representation will be performed by maximizing the variance of the data along the two chosen principal component orthogonal axes. So if the amplitudes of the generated sine and cosine are great enough, then the projection should approximately be these two reference signals represented one with respect to the other. Of course, the simple solution is to multiply a constant to the previously estimated semi-axes' lengths, that we determined in order to compute the perimeter. The canvas size of the design can be used each time as a reference value for the amplitude of the generated signals. To avoid surpassing the fixed size limits imposed by the canvas, the $\mathbf{x_{p,1}}$ and $\mathbf{y_{p,1}}$ vectors can be also be scaled down. Taking into account these considerations, we can write the equations that give our desired, up-scaled version of the coordinate vectors:

$$\begin{cases} \mathbf{x_{ref,2}}[i] = c \cdot a_e \cos\left(\boldsymbol{\varphi}_\mathbf{e}[i]\right) \\ \mathbf{y_{ref,2}}[i] = c \cdot b_e \sin\left(\boldsymbol{\varphi}_\mathbf{e}[i]\right) \end{cases} \tag{28}$$

where $\boldsymbol{\varphi}_\mathbf{e}$ is the estimated phase vector and c is a constant chosen so that the amplitudes of the reference signals are much greater than the semi-axes' lengths.

So, after the initial centering and rotation of the data, two more column vectors are added to the $\mathbf{M_{p,1}}$ projected data from (24), and the resulting 4-column matrix will be the input to a new PCA transformation, that aims to eliminate the unwanted noise.

$$\mathbf{M_2} = \left[\mathbf{x_{p,1}} \quad \mathbf{y_{p,1}} \quad \mathbf{x_{ref,2}} \quad \mathbf{y_{ref,2}}\right] \tag{29}$$

After the new 4×4 covariance matrix is computed, and eigendecomposition is performed, two of the four eigenvectors will be retained as principal components. These eigenvectors will be the ones that correspond to the greatest eigenvalues. Because of the constraints imposed on $\mathbf{x_{ref,2}}$ and $\mathbf{y_{ref,2}}$ (much higher amplitude, same phase), the two chosen eigenvalues should be much greater than the others.

The 4D data is projected onto a 2D space, called S_2:

$$\mathbf{M_{p,2}} = \mathbf{M_2} \cdot \mathbf{V_{s,2}} \tag{30}$$

where $\mathbf{V_{s,2}}$'s columns are the two selected eigenvectors. The result is presented in Figure 6, and as expected, it doesn't contain any of the noise present in the first two columns of $\mathbf{M_2}$.

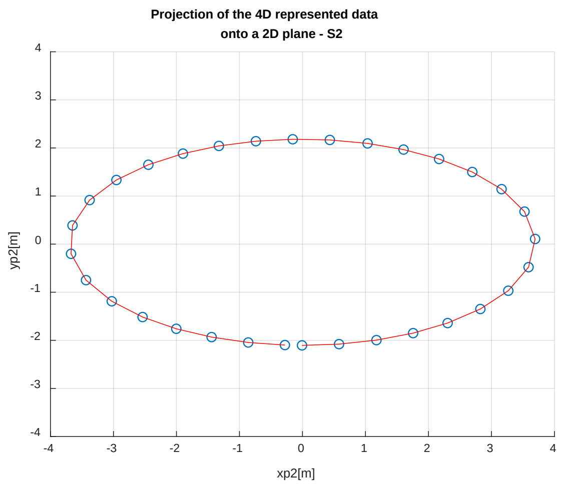

Figure 6. 2D PCA projection of the 4D data from M_2

All that remains is to reverse the PCA transformations, in order to return the data to the original space. So, according to (22), we first compute:

$$\mathbf{M_f} = \mathbf{M_{p,2}} \cdot \mathbf{V_{s,2}^{H}} \tag{31}$$

By extracting and representing the first two columns from $\mathbf{M_f}$, we obtain a filtered version of the canonical ellipse from S_1. This is presented in Figure 7.

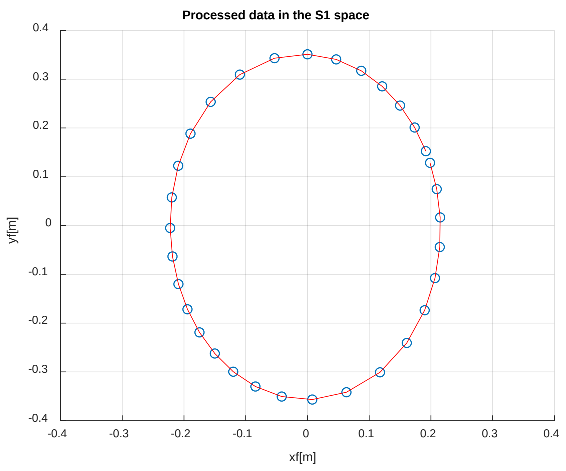

Figure 7. Filtered version of the canonical ellipse

The noise that was previously present within the two coordinate vectors has been significantly attenuated by excluding the eigenvectors that corresponded to the lowest eigenvalues, during the PCA transformation.

The final step is to return to the original space S_0:

$$\mathbf{M_{f,2}} = \begin{bmatrix} \mathbf{x_{f,1}} & \mathbf{y_{f,1}} \end{bmatrix} \cdot \mathbf{V_1}^H \tag{32}$$

where $\mathbf{x_{f,1}}$ and $\mathbf{y_{f,1}}$ are the first two columns of $\mathbf{M_f}$.

In order to visually compare the processed data to the raw measurements (Figure 3), and give the ellipse its original center, the initially subtracted x_0 and y_0 can be added to the abscissa and ordinate vectors present on the columns of $\mathbf{M_{f,2}}$. The final result is presented in Figure 8.

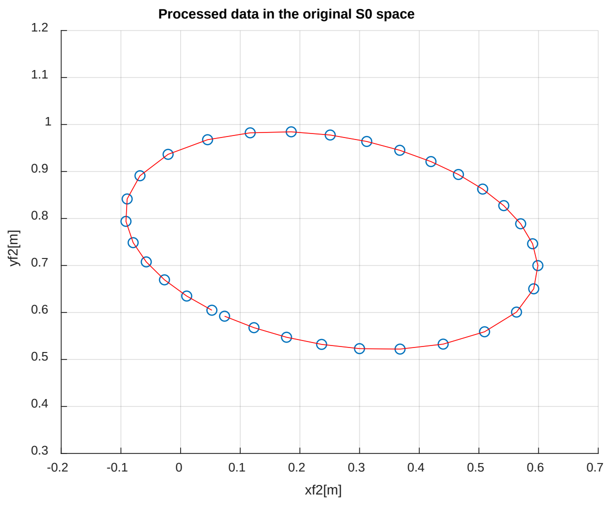

Figure 8. Final version of the processed ellipse

In the context of a 3D design, the algorithm only slightly changes when performing the first PCA projection (corresponding to $S_0 - S_1$). In this case, the input matrix will have three columns, one corresponding to each spatial coordinate, which results in three eigenvectors after covariance matrix computation and eigendecomposition is performed [10]. Also, three rotations (about the x, y and z axes) are performed to simulate a general ellipse's orientation in space. The user's objective is to represent an ellipse in space, not an ellipsoid. So the first PCA projection will be bidimensional. This means that only the two most significant eigenvectors, out of the total number of three, will be selected as principal components. After a bidimensional ellipse is obtained, the problem is reduced to that of a 2D representation, so the algorithm can continue exactly as the one presented in this section.

Simulation results for different scenarios regarding both 2D and 3D cases are presented in Section VI.

V. Generalization and Other Practical Techniques for Shape Extraction

Generalization of the algorithm refers to adapting the ideal reference waveforms generated in (28) to any type of shape. Knowing these waveforms is essential to generating the desired PCA projection of the data. Therefore, the precision with which these ideal waveforms are generated greatly influences how well the noise will be filtered. The ellipse can be considered the most general case, having a sine variation for both axes. A more complex shape will require more complex waveforms, which can be decomposed as sums of sines. By FFT (Fast Fourier Transform) analysis of the measured coordinate vectors, we can identify the frequency of each of these components, as well as an amplitude value for each. Generally, each waveform will have a significant fundamental frequency component, and smaller components corresponding the odd multiples of the fundamental frequency [11]. We can use the spectrum values (frequency and amplitude) of the most significant components to estimate the ideal version the waveforms as a sum of these components, with the amplitudes acting as coefficients. Another option is to generate reference signals for each individual component, without taking into consideration the amplitude values. Any of these approaches can prove rather difficult to correctly implement for higher noise levels, as the noise will also be present within the components of interest.

Knowing beforehand the shape type can greatly simplify the PCA extraction method. For example, in the case of a square, the two orthogonal waveforms will be sawtooth signals, with a phase difference of $\frac{\pi}{2}$. The waveforms and their common spectrum are presented in Figure 9.

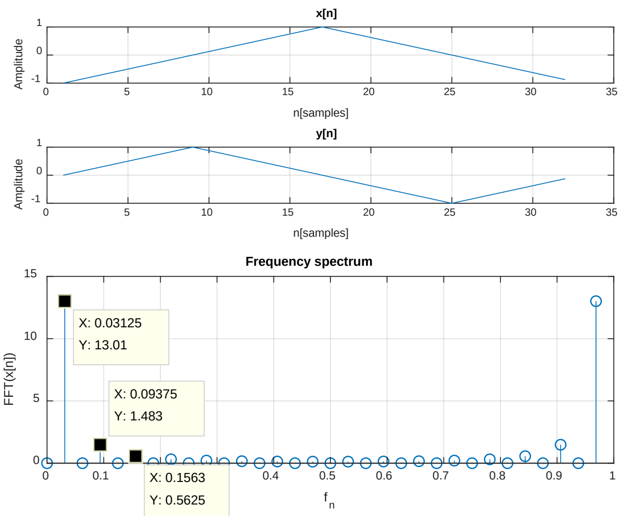

Figure 9. Required waveforms for a square shape

These waveforms will generate a square tilted with $\frac{\pi}{4}$, having the edges on the coordinate axes and the center in the origin, which will correspond to the projection of any measured square after the first PCA transformation, regardless of its original orientation.

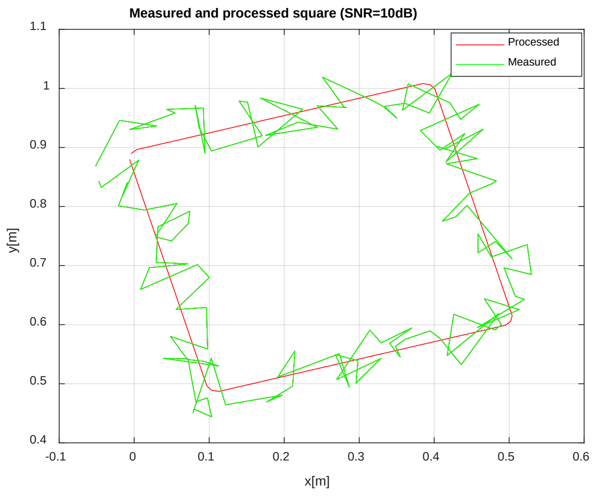

Figure 10. PCA processed square for high noise levels

An example of a processed square, in a scenario with a relatively high noise level, is presented in Figure 10. Point distribution is not an issue here, and smoothing is also applied after noise filtering with PCA, to further improve our results.

A simpler processing method that can be successfully used for basic shapes (ellipses, squares, etc.) is based on determining the geometrical parameters of the shape from its perimeter and area [12]. Usually, the perimeter will give the sum and the area will give the product of the lengths that define the shape [13]. So the first step is to determine the perimeter and area which yield the sum and product. Afterwards, the lengths will be the roots of a quadratic equation based on the sum and product. Knowing the measured coordinates allows us to easily compute the perimeter as the sum of all distances between adjacent points. The area can also be easily computed using the "shoelace" formula:

$$A = \frac{1}{2}\left| x_m y_1 - x_1 y_m + \sum_{i=1}^{m-1} \left(x_i y_{i+1} - x_{i+1} y_i \right) \right| \tag{33}$$

For a square or rectangle, applying this method is straightforward. In the case of an ellipse, the semi-axes product will be the ratio of the determined area and π. The sum can be computed using (27), by performing multiplications and substituting the value of the product. After the geometrical parameters are determined, a perfect version of the shape is generated with the purpose of replacing the original measured one.

This method gives decent results for acceptable or lower noise levels ($SNR = 20$ dB). However, its simplicity and reduced processing requirements come with a number of drawbacks. The obvious one is that for low signal-to-noise ratios, the determined lengths will be highly erroneous and the final shape will not even closely resemble the shape that was intended to be drawn. Also, in the case of amplitude spikes, though the area doesn't suffer much change, the value of the perimeter varies significantly, again affecting the geometrical parameters. Finally, even though the perimeter and area can be computed regardless of the shape's orientation, the initial rotation angle θ, defined in (3), is still necessary. This is because the final shape, after it is generated based on the estimated parameters, needs to be rotated so it can resemble its original measured form. The issue is, without PCA, there is no ideal way to determine or control the value of this angle. It can be estimated, for example, by determining the slope of the semi-major axis, but this approach is also susceptible to high noise levels.

We have mentioned several times by now that the PCA processing method applied to the measured shapes is equivalent to filtering the noise from two orthogonal signals or, in a more general context, from two waveforms that have corresponding orthogonal components. It seems reasonable then to provide a comparison to conventional filtering. Though it may not seem so at first, there are many similarities between the two approaches. Since more complex waveforms would require a great number of FIR bandpass filters, adaptive filtering is considered to be a better option. The desired signals used as references by the FIR adaptive filter are the same waveforms used by the PCA method, similar to those presented in (28). The LMS adaptive approach also requires eigenvalue computation, in order to estimate the optimal value for the convergence coefficient, required to generate the optimal coefficients [14]. The normalized version of this method (NLMS) does not require eigendecomposition, always having a unitary optimal convergence coefficient, though it introduces additional processing by performing normalization with respect to the current signal power at each iteration of the adaptive algorithm. One obvious disadvantage presented by this method is that it depends greatly on the waveforms having a significant number of samples. This would translate in either a high sampling frequency of the positions or drawing the shape with a very low speed. The reason for this is that the number of samples gives the number of iterations the algorithm has to converge the error to zero, which means converging the convolution between the input and the adapting vector of coefficients to the desired signal [15]. The same input signal could be concatenated to itself to artificially create more samples, which would in turn extend the number of iterations. This would only be ideal though if the signal would contain at least one period of each component and the phase would continue to have a linear variation from one copy of the input signal to the other. Also, a higher order filter would be desired for better noise filtering. The problem is that the filter order and number of coefficients are also dependent on the number of samples, a compromise having to be made between this value and the number of iterations used in the convergence process, the latter objectively being more important.

Finally, in the case of compound shapes, an intuitive solution would be to separate each basic shape and filter it using the algorithm discussed at the beginning of this section. The problem is that PCA cannot separate individual features. This is why in the case of square, we did not approach it as a compound shape with 4 lines, and filter each line separately as a particular case of an ellipse. Let's consider a simpler example of two perpendicular segments that intersect at their middle point. The first problem is that if both segments would be affected by significant noise, the overall point distribution would make the segments very difficult to distinguish, and the PCA would have trouble identifying the correct directions of maximum variation. The second problem, is that even in the case of low noise, the best directions of maximum variance PCA can detect are between the two segments. So, in an application required to extract noise from compound shapes, PCA will not suffice, and independent component analysis (ICA) needs to be used [16]. It introduces additional processing by also exploiting the existing kurtosis (fourth moment) information present within the data, and not just the mean and variance like PCA does. In return, ICA can extract individual features, and not average them out, like PCA would do for compound shapes.

VI. Simulation Results

A number of simulations have been performed in Matlab to test the functionality of the method presented in Section IV. Test signals have been generated to simulate measured coordinate vectors, according to all the considerations in Sections II, III and IV, including the uniform point distribution required for a measured real-time drawn ellipse.

As such, some of the input parameters are the number of samples (usually a power of 2 for an ideal FFT result if necessary), the amplitude of the sines r that gives the general size of the ellipse, the θ angle that gives orientation of the ellipse and the (x_0, y_0) offsets that corresponds to the ellipse's center. However, The most important parameters are the phase difference $\Delta\varphi_0$ that affects the semi-axes' lengths and the signal-to-noise ratio, that uses r as the signal amplitude. The first gives us the possibility to test out different ellipses (including circles and lines), while the latter allows us to test the limits of the algorithm's functionality in unfavorable conditions. Amplitude spikes are also introduced, besides noise, to create even more unfavorable initial conditions.

The results will be presented comparatively on the same figure, that is the original measured data (green) and the processed data (red).

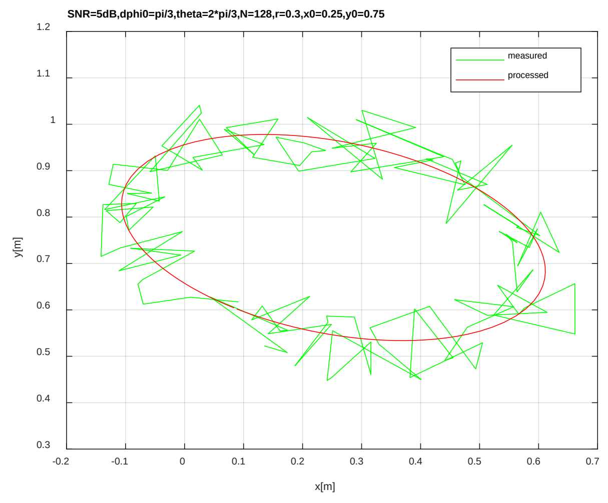

Figure 11. Processed ellipse for high noise levels (PCA)

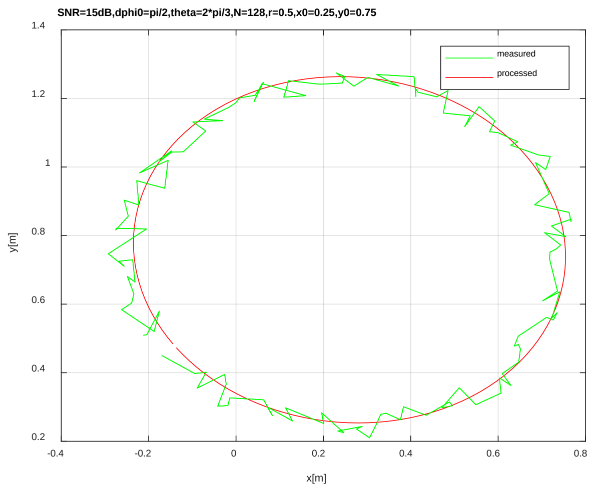

Figure 12. Processed circle for moderate noise levels (PCA)

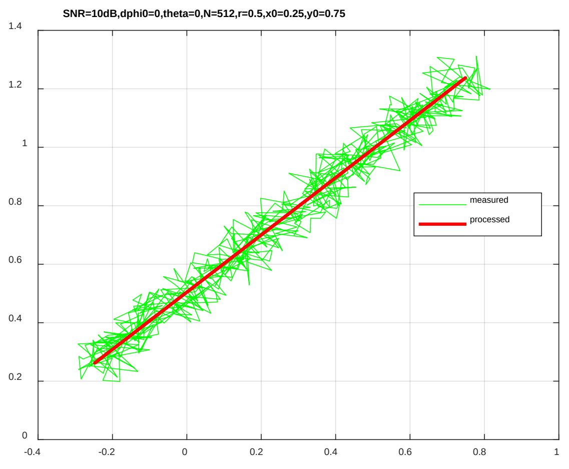

Figure 13. Processed line for high noise levels (PCA)

Figures 11 to 13 present the results obtained from processing different versions of ellipses, each one affected by varying levels of noise. We observe that the method is highly robust to noise. This is to be expected because the noise components will be orthogonal to our signals and by selecting the principal components of interest and excluding the ones corresponding to noise, the noise will be almost entirely eliminated.

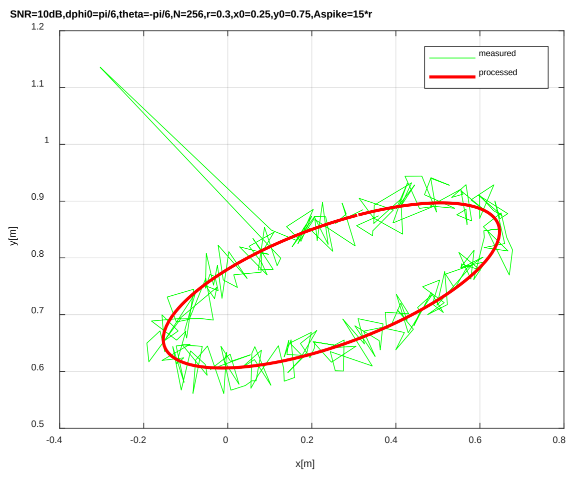

Figure 14. Processed ellipse for a scenario with high noise levels and an amplitude spike (PCA)

Figure 14 presents a scenario where the measurements will also include an amplitude spike. This will have minimal effects on our PCA result, because all the amplitude spike does is introduce an additive impulse to our coordinate signals which will be excluded similarly to the noise, during principal component extraction.

Figures 15-17 present the results obtained by applying the perimeter/area method, presented in the previous section, that is straightforwardly based on estimating the geometrical parameters.

Figure 15 presents one of the scenarios that yields good results for the perimeter/area method or actually any method based on parameter estimation. If the signal-to-noise ratio is high, the method works and it does so requiring less processing than PCA. If, however, the noise's amplitude increases, as in Figure 16, the perimeter value will erroneously increase, and the results will yield an ellipse with semi-axes of significantly different lengths then what would be desired.

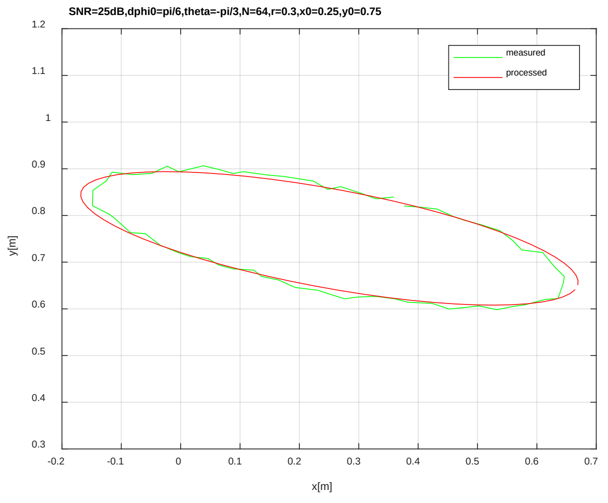

Figure 15. Processed ellipse for low noise levels (perimeter/area)

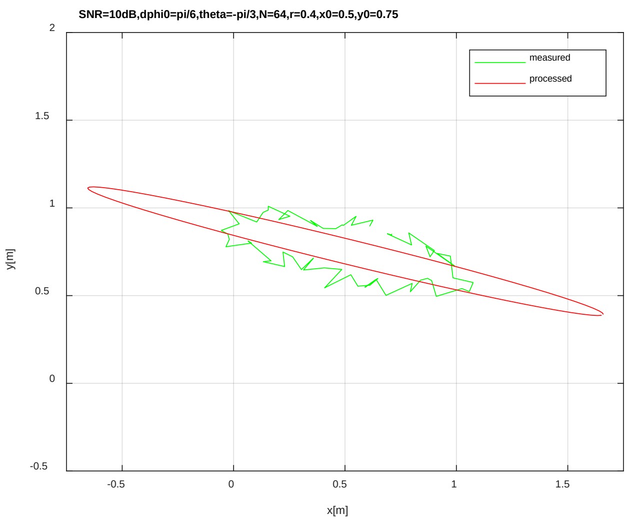

Figure 16. Processed ellipse for high noise levels (perimeter/area)

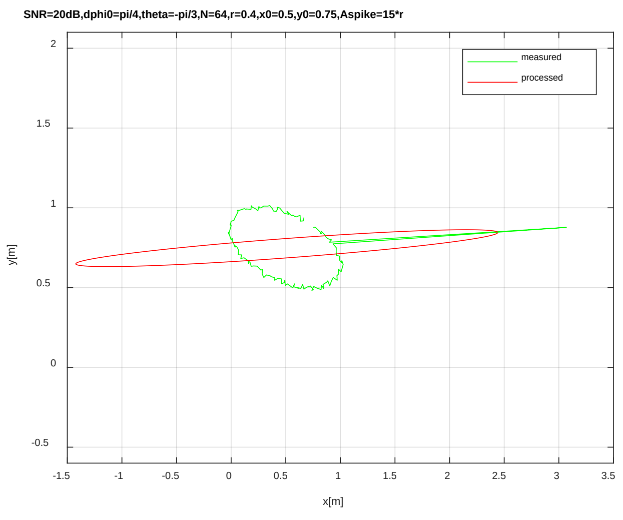

Figure 17. Processed ellipse for a scenario with moderate noise levels and an amplitude spike (perimeter/area)

A greater issue affecting this method is underlined in the case of an amplitude spike, such as the one in Figure 17. Without using PCA, the method can only determine the orientation of the measured ellipse by identifying the slope of the semi-major axis. So in this situation, by confusing the amplitude spike for the semi-major axis, the final processed ellipse will have a false orientation. Sure, the method can be changed to use the semi-minor axis as a reference for determining the orientation. The problem is that, in this case, the evaluated points will have a much lower signal-to-noise ratio, so this approach will not yield any significant improvements. Also, because the area changes very little due to noise or the amplitude spike, the product of the semi-axes is constant. The perimeter on the other hand will increase even further, not only because of the noise, but also due to the presence of the amplitude spike. This means that the sum of the semi-axes will increase greatly. Constant product and increasing sum means that the difference between the semi-axes' lengths will increase, creating an unwanted stretching effect, which can be seen in both Figures 16 and 17.

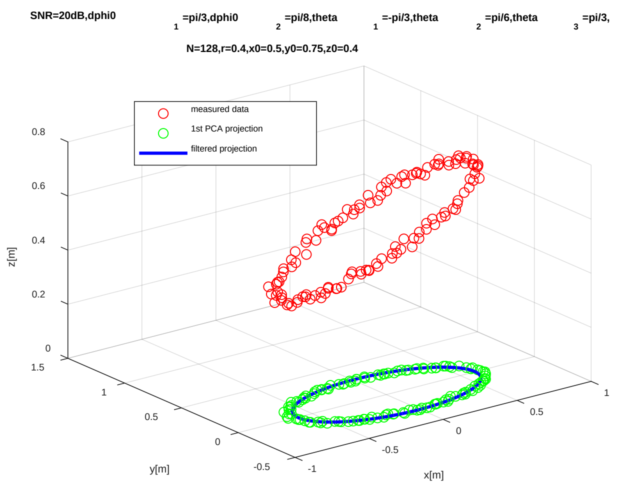

Figure 18. Initial measured data (red) and intermediate results obtained with the PCA extraction method, for a 3D represented ellipse

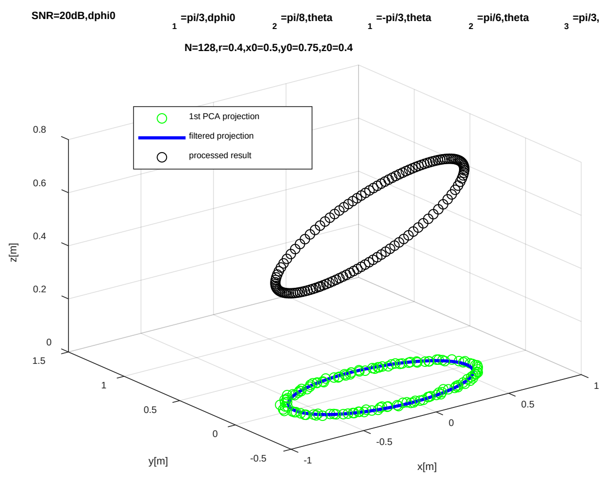

Figure 19. Final processed result (black) and intermediate results obtained with the PCA extraction method, for a 3D represented ellipse

Figures 18 and 19 present the results obtained by applying the PCA extraction method to a 3D ellipse affected by moderate noise. Regarding the algorithm's capacity to filter noise, the results are similar to those obtained in 2D scenarios. This is to be expected, since the method for 3D extraction only varies slightly concerning the first projection of the shape, which can be viewed in the *xy*-plane in both figures. The fact that the final ellipse from Figure 19 has an extremely high signal-to-noise ratio also means that its 3D shape is planar, unlike the measured data from Figure 18.

TABLE I. MEAN SQUARED ERROR VALUES FOR THE SIMULATED SCENARIOS

No	**Method**	**Scenario (Figure index)**	**Mean squared error for x axis**	**Mean squared error for y axis**
1	PCA	10	1.0159e-05	1.3841e-05
2	PCA	11	0.0018	0.0016
3	PCA	12	0.0049	0.0056
4	PCA	13	2.5733e-05	1.8673e-05
5	PCA	14	0.0069	0.0014
6	perimeter/area	15	0.0726	0.0424
7	perimeter/area	16	0.6061	0.1218
8	perimeter/area	17	1.1671	0.0432

Finally, Table I presents the mean squared error (MSE) values computed for the presented scenarios. The reference for the MSE is the original simulated shape without noise. This is compared of course to the processed shape, extracted with PCA or generated by means of geometrical estimation. Each scenario was simulated 100 times, with different noise vectors, and the maximum value for the MSE was retained each time. For the square and line cases, processed with PCA, the MSE has the lowest values for both coordinate vectors. For curbs (ellipses, circle) the MSE value obtained with PCA is higher but still insignificant. Considering the shape sizes range from 0.3m to 0.5m, the error is about 1%. For the 3D simulated version of the ellipse (Figures 18 and 19) the MSE has similar values, more specifically [0.0021, 0.0024, 0.0007]. When using the geometrical estimation method based on computing and perimeter and area, the MSE increases considerably, to a point where the error surpasses the length of the coordinate vectors.

VII. Conclusions and Future Work

The dimensionality reduction provided by PCA can be used to great effect in extracting 2D or 3D basic shapes from measured data affected by noise. Though it introduces a significant amount of processing, it yields good results even for low signal-to-noise ratios of 5dB.

The uniform point distribution that results when measuring real data needs to be taken into account when processing ellipses or other curbs in general.

Even though the presented processing algorithm focuses on extracting ellipse-type shapes, the method can be generalized, but only if generating reference waveforms with high SNR, required for the PCA projection, is possible.

Some of the advantages PCA introduces to shape extraction are: non-dependence on shape orientation, fewer geometrical estimations than other methods, high noise robustness.

Its main disadvantage is its inability to extract individual features from compound shapes. In this case, the ICA algorithm should be implemented and evaluated for different shapes and unfavorable processing scenarios.

References

[1] Mariescu-Istodor R, Fränti P “Grid-based method for GPS route analysis for retrieval”, ACM Trans. on Spatial Algorithms and Systems, 3, paper 8: 1-28, 2017, doi: 10.1145/3125634

[2] C. Qiu, H. Ren, H. Zou and S. Zhou, "Performance comparison of target classification in SAR images based on PCA and 2D-PCA features," 2009 2nd Asian-Pacific Conference on Synthetic Aperture Radar, Xian, Shanxi, 2009, pp. 868-871, doi: 10.1109/APSAR.2009.5374193

[3] A. N. Gaikwad, D. Singh and M. J. Nigam, "Recognition of target in through wall imaging using shape feature extraction," 2011 IEEE International Geoscience and Remote Sensing Symposium, Vancouver, BC, 2011, pp. 957-960, doi: 10.1109/IGARSS.2011.6049291

[4] D. Umbach and K. N. Jones, "A few methods for fitting circles to data," in IEEE Transactions on Instrumentation and Measurement, vol. 52, no. 6, pp. 1881-1885, Dec. 2003, doi: 10.1109/TIM.2003.820472

[5] T. Liu, W. Zhao, Z. Wang and S. Jia, "An Image-Based Circle Extraction Method of Three-Dimensional Point Cloud Data," 2008 Second International Symposium on Intelligent Information Technology Application, Shanghai, 2008, pp. 447-451, doi: 10.1109/IITA.2008.255

[6] F. Mai, Y. S. Hung, H. Zhong and W. F. Sze, "A Hierarchical Approach for Fast and Robust Ellipse Extraction," 2007 IEEE International Conference on Image Processing, San Antonio, TX, 2007, pp. V - 345-V – 348, doi: 10.1109/ICIP.2007.4379836

[7] Z. Shengnan, Y. Shuang, N. Lianqiang and Y. Weiqi, "A Fast Ellipse Detection Method in Planar Target Image," 2012 Third International Conference on Digital Manufacturing & Automation, GuiLin, 2012, pp. 42-45, doi: 10.1109/ICDMA.2012.10

[8] M. Oliveira, A. Sutherland and M. Farouk, "Two-stage PCA with interpolated data for hand shape recognition in sign language," 2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, 2016, pp. 1-4, doi: 10.1109/AIPR.2016.8010587

[9] H. Wang and H. Zhang, "Improving image segmentation via shape PCA reconstruction," 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX, 2010, pp. 1058-1061, doi: 10.1109/ICASSP.2010.5495334

[10] Mao Rui, Tang Yan, Chen Qiang, Zhang Qing-Chen and Gan Yuan-Chao, "3D model feature extraction algorithm based on shape weighted," Proceedings of 2012 International Conference on Measurement, Information and Control, Harbin, 2012, pp. 498-501, doi: 10.1109/MIC.2012.6273350

[11] X. Ning, F. Li, Y. Wang and W. Hao, "Primitive Shape Extraction for Objects in Point Cloud," 2017 International Conference on Virtual Reality and Visualization (ICVRV), Zhengzhou, China, 2017, pp. 144-149, doi: 10.1109/ICVRV.2017.00037

[12] Y. Ye, F. Guangrui and O. Shiqi, "An Algorithm for Judging Points Inside or Outside a Polygon," 2013 Seventh International Conference on Image and Graphics, Qingdao, 2013, pp. 690-693, doi: 10.1109/ICIG.2013.140

[13] J. Zunic and P. L. Rosin, "A new convexity measure for polygons," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 7, pp. 923-934, July 2004, doi: 10.1109/TPAMI.2004.19

[14] G. Eleyan, M. S. Salman and C. Turan, "Two-dimensional sparse LMS for image denoising," 2015 Twelve International Conference on Electronics Computer and Computation (ICECCO), Almaty, 2015, pp. 1-4, doi: 10.1109/ICECCO.2015.7416909

[15] G. H. Costa and J. C. M. Bermudez, "On the Design of the LMS Algorithm for Robustness to Outliers in Super-Resolution Video Reconstruction," 2006 International Conference on Image Processing, Atlanta, GA, 2006, pp. 1737-1740, doi: 10.1109/ICIP.2006.312717

[16] J. Koikkalainen and J. Lotjonen, "Image segmentation with the combination of the PCA- and ICA-based modes of shape variation," 2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821), Arlington, VA, USA, 2004, pp. 149-152 Vol. 1, doi: 10.1109/ISBI.2004.1398496