Time Series modeling using Recurrence Plots and Face Recognition techniques

J. M. Medina-Apodaca°, J. Figueroa-Nazuno° & S. García-Benitez*
°Centro de Investigación en Computación, Instituto Politécnico Nacional.
*Instituto de Ingeniería, UNAM.

Abstract

In this paper we present a time series modeling tool which enables their easy recognition and treatment using Recurrence Plots and Eigenfaces. We obtain the recurrence plots of a set of time series, and take the values of each plot as a very long vector, as we would with the pixels of a face image when using the Eigenfaces technique. This way, we build the face space of the plots. By expressing each recurrence plot as a point in the face space, we can represent the plots with a short vector. These vectors can be used as input for an analysis process. Using the inverse path, we can recover a time series represented by the output of the analysis.

Keywords: Recurrence Plots, Eigenfaces, pattern recognition, time series analysis.

1. Introduction.

Time Series modeling and prediction are up to now hard tasks for real phenomena. It is often practically impossible to find a mathematical function which models a real life time series. Furthermore, most of the events in nature and society have too many variables influencing the observed behavior, while several mathematical methods such as autoregression and correlation are limited to linear signals.

One of the most often used methods for data analysis is the Fourier Transform. When used, this transform gives the input signal in the domain of frequency, showing the influence of each frequency component in the whole signal. However, Fourier analysis carries a loss of details about some features of the signals, such as those related to the periodicity or maximums and minimums.

Beside the purely mathematical methods for data analysis are the computational tools. Currently, one of the most powerful and most often used tools are Artificial Neural Networks (ANNs), which can learn the behavior of a time series adapting to a training set of data. However, they have several drawbacks when used for complex models: the correct parameters are difficult to find, the networks can be overfitted to the training set being useless for other data, and a relatively large amount of inputs and/or outputs makes it necessary to count with huge amounts of computational power and time to train the network.

The method described in this paper uses recurrence plots to multiply the information contained in the signal, making some subtle details of the signal visible even for a human observer. Also, recurrence plots may contain information about the underlying process which generated the signal, such as the number of factors involved on its phase space.

A face recognition technique called "eigenfaces" is used to express a family of signals as a combination of others. The result is that a large signal can be expressed by a vector with a reduced number of entries.

2. Recurrence Plots.

Recurrence plots are used to reveal several characteristics of a time series, such as the degree of aperiodicity and stationarity [1]. Let S be a series with N terms

$$s_0, s_1, s_2, s_3, s_4, s_5, \ldots. s_i, \ldots.. s_{N-1}$$

Let V be the set of all the vectors

$$v_i = (s_i, s_{i+d}, s_{i+2d}, \ldots. s_{i+Dd})$$

Where D is called the embedding dimension and d is called the delay. The embedding dimension is formally the number of variables involved in the phase space diagram of the underlying process which generated the series. However, this dimension can be forced to be 1 without affecting too much the model.

The Recurrence plot is a map where the value of the point (i, j) is the Euclidean distance between the vectors v_i and v_j. This is, the point (i, j) in the plot is equal to $\|vi - vj\|$.

Normally, the values in the map are associated with a color code, so it can be represented in a graphical way. There are different color codes that can be assigned to the values on the plot. The most commonly used is the spectrum, in which the color range runs from white to red, green, blue and black. A white point represents a value of 0, and a black point represents the longest possible value.

Figure 1 shows the recurrence plot of a sine signal. Note the well defined patterns characteristics of a periodical signal. Since the distance is a commutative measure (i.e. the distance from vector v_i to v_j is the same than the distance from vector v_j to v_i), recurrence plots are always symmetrical respect to the diagonal.

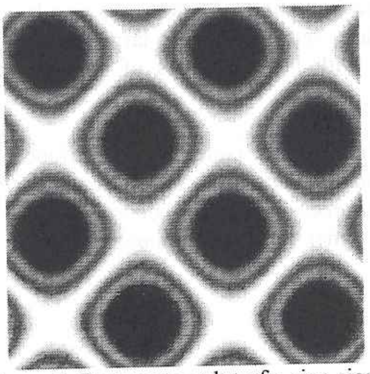

Figure 1. Recurrence plot of a sine signal

Figure 2 shows the recurrence plot of a white noise signal. As this is a random signal there are not well defined patterns as in the case of the sine signal. However, the plot is still symmetric respect to the diagonal.

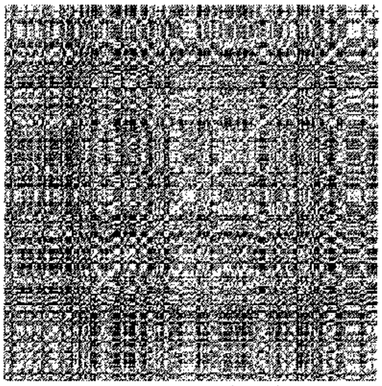
Figure 2. Recurrence plot of a white noise signal

Recurrence plots are useful for identifying special features of a time series, which could hardly be seen in a simple graph. For example, in a Recurrence Plot its easy to identify some kind of periodicity and relations among different parts of the series, even if this is related to a complex phenomena.

3. Eigenfaces.

Eigenfaces is a face recognition technique based on extracting the most important features of a set of face images, and use them to recognize new face images presented to the system.

In mathematical terms, this technique consists on obtaining the eigenvectors of the covariance matrix whose columns are a set of face images expressed as vectors [2]. These vectors are called the eigenfaces of the set of images.

Each face image of the training set can be exactly represented by a linear combination of the resulting eigenfaces. In principle, the number of face images in the training set is equal to the number of eigenfaces. However, the faces can be also represented using the best eigenfaces of the set, i.e. those which have the largest eigenvalues, and which therefore account for the most variance within the set of images. This way, a multitude of face images can be reconstructed by weighted sums of a small collection of characteristic images.

Once normalized, the complete set of eignenfaces form an orthogonal incomplete basis for a very long dimension space. This space is called the face space. Each face from the set can be totally expressed as a point in the face space. The basic idea of the eigenfaces technique is that, since the eigenfaces are orthogonal vectors and the face images are similar (they are all face images of the persons whose face is already in the training set), the faces which are not in the original set can be approximated by a point in the face space.

The eigenfaces technique is able to recognize a face even with small differences in the angle, differences in age, and different illumination environments. This can also easily decide whether or not an image is a face.

Given a point in the face space, it is easy to return to the original face. Since the vector expresses a linear combination of the eigenfaces, the return can be achieved simply by multiplying each component of the vector by the corresponding eigenface.

Although the original eigenfaces technique proposed to compute the covariance matrix, the technique gives better results by making up the matrix with the original values of the faces as its columns, and computing its eigenfaces.

4. The Developed Technique.

The technique presented here can be applied as preprocessing and post-processing stages for other time series analysis techniques. The preprocessing stage is shown in figure 3. Given a set of time series, we compute the recurrence plot for each one and find the face space corresponding to the computed set of recurrence plots.

To find the face space we take the set of recurrence plots and put it in a matrix, where each column is a recurrence plot expressed as a very long vector of point values in the plot. Then, we compute the eigenvectors of the matrix, obtaining the face space.

Once we have the face space, we can express each recurrence plot as a linear combination of the eigenfaces. This results in a point in the face space representing a whole recurrence plot with only a few values.

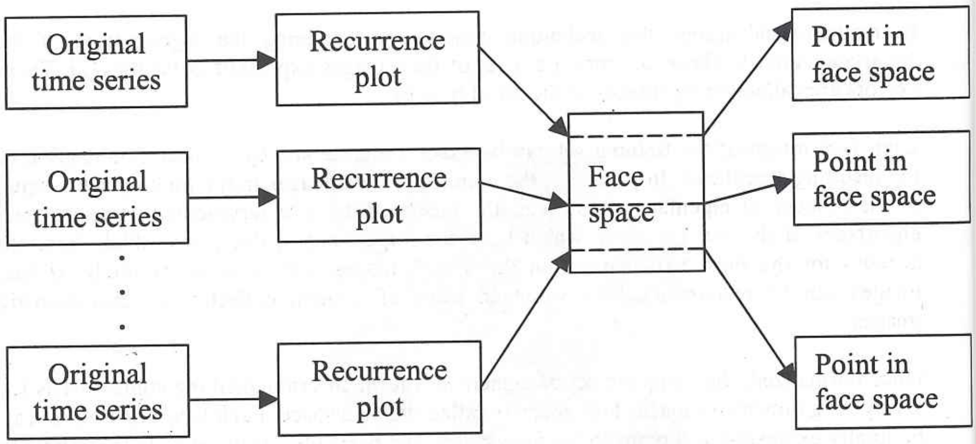

Figure 3. Process to train the ANN from the original time series

Note that dimension of the vectors in the face space will be equal to the number of vectors used as the orthogonal incomplete basis for that face space. The number of eigenfaces that can be computed is equal to the number of recurrence plots.

In principle, we may use the whole set of eigenfaces to obtain an accurate representation of the recurrence plots. However, it is possible to take a subset of the eigenfaces, choosing those which have the longest corresponding eigenvalues.

As the recurrence plots have more expression power than time series, at the end of the process we have a detailed description of each series represented by a short number of values. According to this, the whole process can be used as a preprocessing stage for another analysis or treatment which needs an accurate description of the signals and cannot process a large amount of entries (as in the case of Artificial Neural Networks).

For the post-processing stage, if the output of an analysis process (which received points in the face space as inputs) is another point in the same space, the resulting point can be converted to a recurrence plot and, in turn, to the underlying time series. To achieve this inverse process is necessary for the output point to be referred to a known face space, and for the recovered recurrence plot to have an embedding dimension and delay of 1. This is expected to happen when the output point of the analysis process is applied to the same face space as the one which was used to produce the input for the system, and when all the recurrence plots have been generated using embedding dimensions and delays of 1.

Using this preprocessing and post-processing stages we have a complete cycle for time series analysis using a complete but very short description of them.

5. Computing the Recurrence Plots.

To compute the recurrence plots, the system first obtains the vector of each given point in the time series according to the given embedding dimension and delay. For example, suppose the system is about to compute the point (5, 3) in the plot. If the embedding dimension is 2 and the delay is 3, then the computed vectors would be (x_5, x_8) and (x_3, x_6).

Then, the value of each point of the plot is computed as the magnitude of the difference between the vectors, and the value is added to the plot. The resultant value can be represented in the plot in several manners according to the choice of the user. For example, a point can be added to a text file, or can be represented as a color in a bitmap file or on the user's screen.

As an optimization, if the embedding dimension is 1, the system does not compute the vectors, but simply computes the value of the point as the difference between the values in the given positions. In the previous example, if the embedding dimension is 1, the point (5, 3) is computed as x_5-x_3.

6. Computing the Eigenfaces.

To compute the eigenfaces, the recurrence plots of the training set are taken as very long vectors. If a NxN plot is given, the corresponding vector will have N^2 elements.

In the original technique, the average vector of the plots in the training set is computed, and the difference between each vector and the average vector is placed as a column of a matrix. However, better results are obtained if we take the original vectors instead of the difference with the average vector.

Call A the matrix which has the plots (or the difference between the plots and the average plots) as its columns. The eigenfaces are the eigenvectors of the matrix AxA^t.

A common size for a time series is about 1000 values. So, the recurrence plots are expected to have about 1,000,000 points. In this case, the resultant matrix AxA^t will have 1,000,000 rows and 1,000,000 columns. Computing the eigenvectors of a 1,000,000x1,000,000 matrix is an intractable problem.

A common technique to overcome this problem is to first compute the eigenvectors of the matrix $L=A^txA$ [3]. If we have *m* plots in the training set, then L will be a *m*x*m* matrix, smaller than the previous one.

To compute the eigenvectors of L the system uses the QR algorithm[4], which consists of finding a matrix Q such that $D=Q^tLQ$, where D is a diagonal matrix, and Q is a sequence of rotations, i.e. $Q=Q_1Q_2...Q_m$ where each Q_i is a rotation matrix. At the end, each value in the diagonal of D is an eigenvalue, and each column of Q is the corresponding eigenvector.

If we place these eigenvectors as columns of a matrix V, the eigenfaces can be computed as the result of normalizing the matrix AxV. The system stores the resultant matrix of eigenfaces in order to use it with future plots related to the training set.

Let M be the matrix of eigenfaces obtained from the previous step, and let *u* be a recurrence plot represented as a vector. To compute the representative vector of recurrence plot *u*, the product M x *u* is computed. The resultant vector will have as many elements as the training set of the eigenfaces.

The representative vectors of the recurrence plots can be reduced even more by taking only the most important eigenfaces from the generated set. By doing so, the training cannot be totally recovered from the eigenfaces, but just approximated.

The most important eigenfaces for the training set will be those having the largest eigenvalues generated when the eigenvectors are computed.

7. Recovering the Time Series.

If the output of the analysis stage is a point in the face space, it is necessary to traverse the inverse path in order to obtain the time series represented by this point.

To return to the original series, it is necessary to convert the point in the face space to the corresponding recurrence plot, and then return to the original time series, as shown in figure 4. The first step can be done easily by multiplying each component of the vector by the corresponding eigenface. The last step is a little more complicated, because the recurrence plot holds only the differences between the vectors of the chosen dimension and delay.

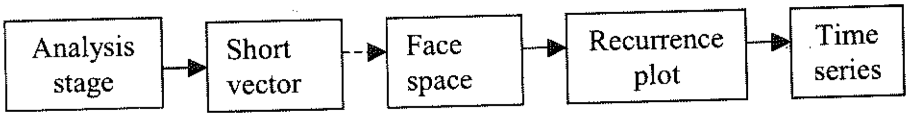

Figure 4. Process to obtain a time series from the output of the analysis

We can return from the recurrence plot to the time series only if the dimension and delay used to build the recurrence plot are both 1. Even in this case we cannot always recover exactly the original time series, since it is impossible to recover the original bias, and the recovered plot may be the negative of the original one if the second point of the series is less than the first.

In the case of a recurrence plot with dimension and delay 1, we actually have the absolute differences of each point of the time series with respect to every other point. To recover the original series, we need to choose the recovered values for the time series in a way that all the "lines" in the recurrence plot represent the same information (or similar information in the case of noisy recovered plots).

The basic algorithm to recover the time series is:

1. Take the values of the first line of the recurrence plot as the values of the time series. Remember that the first line in the time series represents the difference of all the point with respect to the first one, so what you are doing is to consider the first point as zero and all the other points as greater or equal to the first point.
2. Take the next line (line n) from the recurrence plot
3. For each point in the line, compute the sum and the difference between the value at position n (number of line in the recurrence plot) in the current line, and the value of all other positions i in the line. Take the absolute value of the result and call it δ^+ for the sum, and δ^- for the difference.
4. Let λ be the value in the position n of the recovered time series. If $|\lambda-\delta^+|<|\lambda-\delta^-|$ then the sign of the value at position n in the recovered time series is wrong, and should be changed.
5. Repeat step 2 for each line in the recurrence plot.

Since the recovered recurrence plot may be noisy because of the recovery process, we actually compute the value of each point in the time series as the average of the values extracted from the recurrence plot for that specific point. Also, we do not change the sign of the value each time it is detected to be wrong. Instead, for each position in the time series, we have a counter that stores the number of iterations in which the algorithm detects it should be changed. The sign of the value is only changed when the counter crosses by 0, i.e. only when most of the iterations agree in that the sign should be changed.

To explain how our algorithm works, we can observe the relationship among the involved data in figure 5. The figure shows the original time series, the previous recovered data (*RecValue*) and the value of the plot when analyzing line n (*nl*) of the plot.

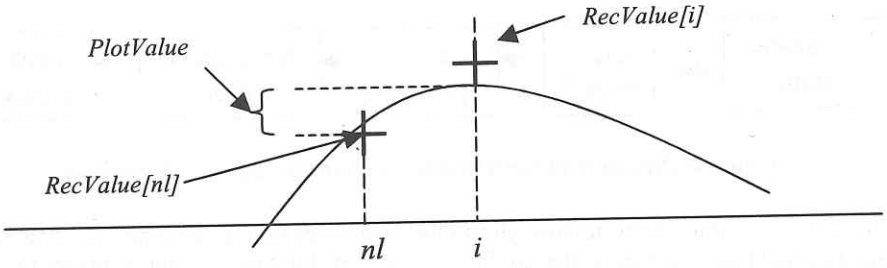

Figure 5. Relationship among recovered data, plot values and the original time series when returning from a noisy plot.

Suppose we want to get a better approximation of point *i*. In a first approach, we assume the recovered value in position *nl* (the number of line we are using on the plot) as correct. For definition of the recurrence plot, the value of the plot in line *nl*, position *i* (*PlotValue*), should be equal to the distance between *RecValue[i]* and *RecValue[nl]*. Note that when the dimension and delay of the plot are both 1, the actual distance is simply the absolute value of the difference between the values.

If we were sure *RecValue[nl]* and *PlotValue[i]* are correct, we could compute the real value in position *i* simply by adding *RecValue[nl]* and *PlotValue[i]*. However, these values are not always accurate, so we should better compute the new value of *RecValue[i]* as the average among all the computed values. This way, in each step we should add or subtract:

$$\frac{\mathrm{Re}\,cValue[i]-(\mathrm{Re}\,cValue[nl]-PlotValue)}{Total\ number\ of\ lines\ in\ recurrence\ plot}$$

to *RecValue[i]* in order to compute its new value.

The complete algorithm to return to the original time series from the recurrence plot is shown below:

```
RecoverOriginalValues ()
{
RecValue[] = values in first line of the recurrence plot
positive[] = {0} //Counter of positive agreements
for nl=1 to data length
      CorrectOriginalValues (RecValue, nl, positive);
}

CorrectOriginalValues (RecValue[], nl, positive[])
{
PlotValue = Value of the plot in line nl, position i
for i=0 to data length
{
if PlotValue<0          // The recurrence plot should not have
PlotValue = 0; // negative values
      if |PlotValue-|RecValue[nl]+RecValue[i]|| <
                                |PlotValue-|RecValue[pos]-RecValue[i]|| {
```

```
        //Incorrect sign
        if RecValue[i]>0 {
            positive[i]--;  // Change the sign every time
            if positive[i]<0       // positive[i] crosses by zero
                RecValue[i] = -1.0*RecValue[i];
        }
        else {
            positive[i]++;
            if positive[i]>0
                RecValue[i] = -1.0*RecValue[i];
        }
    }
    else {          // Correct sign. Correct the value
        if RecValue[i]>0
            positive[i]++;
        else
            positive[i]--;
        AbsDif = |RecValue[pos]-RecValue[i]|;
        if AbsDif>PlotValue { // Increase the difference
            if RecValue[i]>RecValue[pos]
                RecValue[i] = RecValue-(AbsDif-
                    plotValue)/(length of data);
            else
                RecValue[i] = RecValue+(AbsDif-
                        PlotValue)/(length of data);
        }
        else {              // Decrease the difference
            if RecValue[i]>RecValue[pos]
                RecValue[i] = RecValue[i]+(PlotValue-
                    AbsDif)/(length of data);
            else
                RecValue[i] = RecValue-(PlotValue-
                    AbsDif)/(length of data);
            }
        }
    }
}
}
```

8. Example.

As an example consider three input time series, each with 200 values. These signals are shown in Figure 6.

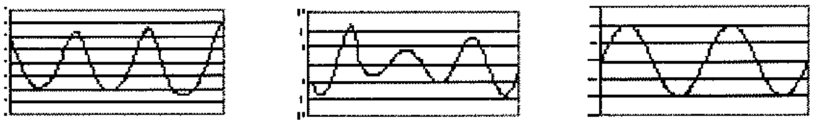

Figure 6. Three input time series

The first step of the preprocessing stage is to compute the recurrence plot of each input time series. The resulting recurrence plots are shown in figure 7.

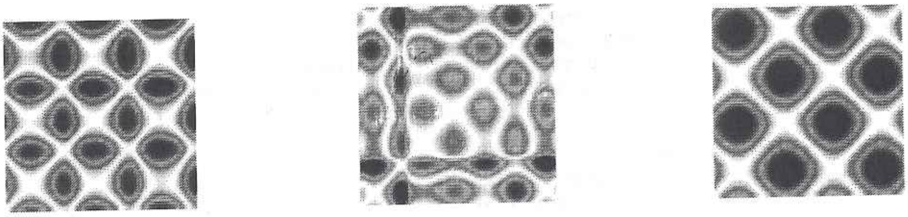

Figure 7. Three recurrence plots corresponding to the input time series

The next step is the generation of the set of eigenfaces corresponding to the recurrence plots shown above. These eigenfaces form the face space, and thus are treated as a unit. However, for this paper the eigenfaces were extracted and their representation with negative values put to zero is shown in figure 8. Note that the eigenfaces do not have a one to one correspondence with the recurrence plots.

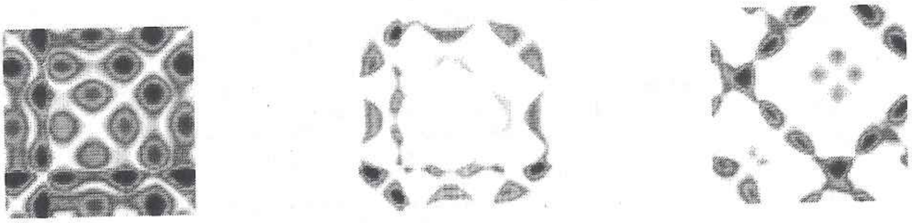

Figure 8. The eigenfaces of the face-space

The vectors in the face-space show the weight each eigenface has in the recurrence plots. These non-normalized vectors are shown below, keeping the order of the above recurrence plots:

(777.1191573465204, -494.3188957993881, 12.325170500631794)
(1448.5151494857735, 268.12345386239616, 6.803736657263384)
(138.87833187137394, -30.50296638025993, -139.9314167509452)

Given the vectors in the face-space, the original recurrence plots and the underlying time series can be recovered, as shown in figures 9 and 10.

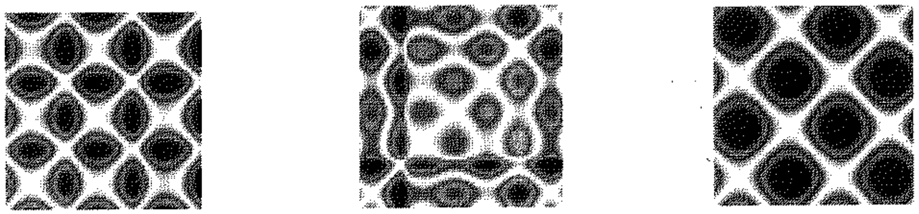

Figure 9. Recovered recurrence plots

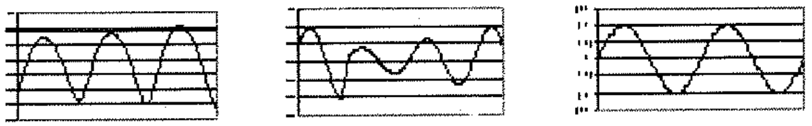

Figure 10. Recovered time series

Note that the first and second recovered time series are the inverse (negative) of the original versions. This happens because the recurrence plots do not have information about the sign of the first point.

9. Conclusion.

Powerful tools for time series analysis exist, such as ANN. Sometimes, a technological problem arises when it is necessary to analyze a large amount of inputs in parallel, because this would make necessary to have an input and an output in the analysis tool for each single value in the time series. In the case of ANNs, would be necessary to have a number of neurons of at least two times the number of values in the time series. Furthermore, depending on the number of hidden layers in the ANN, the number of neurons may be several times the number of inputs.

The number of entries can be dramatically reduced using the eigenfaces technique. However, the result can be improved converting the time series to a recurrence plot before applying the process. The recurrence plots can easily show periodicity and many other features hardly visible in the original time series.

The inverse path is needed to process outputs from the ANN and obtain the corresponding time series. Specifically, we have shown an algorithm to recover the original time series from a possibly noisy recurrence plot.

References

[1] Koebbe, M. & Mayer-Kress, G. (1992). Use of Recurrence Plots in the Analysis of Time-Series Data. *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity, Proc. Vol. XXI,* Addison-Wesley, Santa Fe. 361-378.

[2] Turk, M. & Pentland, A. (1991). Face recognition using eigenfaces. *Proc. IEEE CVPR, Maui, HI.* 586 - 591.

[3] Turk, M. & Pentland, A. (1991). A. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience.*

[4] Watkins S. D. *Fundamentals of Matrix Computations*. John Wiley & Sons. 140-142, 279, 280.

[5] Joliffe, I. T. (1986). *Principal Component Analysis*. Springer-Verlag, New York.

[6] Pentland A., Moghaddam B. & Starner T. (1994). View-Based and Modular Eigenspaces for Face Recognition. *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'94)*