Implications of Z-Normalization in the Matrix Profile*

Dieter De Paepe[1 [0000−0002−6661−8474]],
Diego Nieves Avendano[1 [0000−0001−6215−6439]], and
Sofie Van Hoecke[1 [0000−0002−7865−6793]]

IDLab, Ghent University – imec,
Zwijnaarde Technologiepark, Zwijnaarde, Belgium
http://idlab.ugent.be

Abstract. Companies are increasingly measuring their products and services, resulting in a rising amount of available time series data, making techniques to extract usable information needed. One state-of-the-art technique for time series is the Matrix Profile, which has been used for various applications including motif/discord discovery, visualizations and semantic segmentation. Internally, the Matrix Profile utilizes the z-normalized Euclidean distance to compare the shape of subsequences between two series. However, when comparing subsequences that are relatively flat and contain noise, the resulting distance is high despite the visual similarity of these subsequences. This property violates some of the assumptions made by Matrix Profile based techniques, resulting in worse performance when series contain flat and noisy subsequences. By studying the properties of the z-normalized Euclidean distance, we derived a method to eliminate this effect requiring only an estimate of the standard deviation of the noise. In this paper we describe various practical properties of the z-normalized Euclidean distance and show how these can be used to correct the performance of Matrix Profile related techniques. We demonstrate our techniques using anomaly detection using a Yahoo! Webscope anomaly dataset, semantic segmentation on the PAMAP2 activity dataset and for data visualization on a UCI activity dataset, all containing real-world data, and obtain overall better results after applying our technique. Our technique is a straightforward extension of the distance calculation in the Matrix Profile and will benefit any derived technique dealing with time series containing flat and noisy subsequences.

Keywords: Matrix Profile · Time Series · Noise · Anomaly Detection · Time Series Segmentation.

* This work has been carried out in the framework of the Z-BRE4K project, which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 768869.

Preprint version.
The final authenticated version is available online at https://doi.org/10.1007/978-3-030-40014-9_5

1 Introduction

With the lower cost of sensors and according rise of IoT and Industrial IoT, the amount of data available as time series is rapidly increasing due to rising interest of companies to gain new insights about their products or services, for example to do pattern discovery [15], user load prediction or anomaly detection [19].

The Matrix profile is state-of-the-art technique for time series data that is calculated using two time series and a provided subsequence length. It is a one-dimensional series where each data point at a given index represents the Euclidean distance between the z-normalized (zero mean and unit variance) subsequence starting at that index in the first time series and the best matching (lowest distance) z-normalized subsequence in the second time series. Both inputs can be the same, meaning matches are searched for in the same time series. The Matrix Profile Index, which is calculated alongside the Matrix Profile, contains the location of the best match (in the second series) for each subsequence.

The Matrix Profile can be used to find the best matching subsequence in a series, i.e. motif discovery, or to find the subsequence with the largest distance to its nearest match, i.e. discord discovery. It also serves as a building block for other techniques such as segmentation [7], visualizing time series using Multidimensional Scaling [22] or finding gradually changing patterns in time series [24].

The usage of the z-normalized Euclidean distance can be explained by two factors. First, the MASS algorithm [14] was a known method to calculate the z-normalized distance between a sequence of length m and all subsequences obtained by sliding a window of length m over a longer sequence of length n. MASS was a vital part of the original method to calculate the Matrix Profile in reasonable time. Secondly, the z-normalized Euclidean distance can be seen as a two-step process to compare the *shape* of two sequences: the z-normalization transforms each sequence to their normal form, which captures their shape, after which the Euclidean distance compares both shapes. This makes the Matrix Profile well suited for finding patterns in data where a wandering baseline is present, as often occurs in signals coming from natural sources or due to uncalibrated sensors, or where patterns manifest with different amplitudes, which can occur by subtle changes in the underlying system or when comparing signals from different sources.

Although z-normalization is important when comparing time series [10], it has one major downside: when dealing with flat sequences, any fluctuations (such as noise) are enhanced, resulting in high values in the Matrix Profile. This behavior conflicts with our human intuition of similarity and can have an adverse effect on techniques based on the Matrix Profile. A preliminary example of this can be seen in Figure 2, where a discord that is easily detectable using the Matrix Profile becomes hidden once noise is added to the signal. Previous literature has mainly avoided cases with series containing flat and noisy regions, most likely due to this effect.

This paper is an extended version of our previous work [4]. In this version, we diverge less on the many merits of the Matrix Profile and instead discuss several properties of the z-normalized distance relevant for the Matrix Profile.

Furthermore, we have used a new dataset from Yahoo! Webscope in our anomaly detection use case and introduced a new visualization use case.

This paper is structured as follows: Section 2 lists literature related to the Matrix Profile. Section 3 discusses several properties of the z-normalized Euclidean distance that are either directly relevant when using the Matrix Profile or are used in our main contribution. Section 4 provides detail on the effect of flat, noisy subsequences in the Matrix Profile, as well as our solution to compensate for this effect. We demonstrate our technique for anomaly detection in Section 5, for semantic segmentation in Section 6, on data visualization in Section 7 and conclude our work in Section 8.

2 Related Work

In this section, we focus on works related to the Matrix Profile, introduced by Yeh et al. [23] as a new time-series analysis building block, together with the STAMP and STAMPI algorithm to calculate the Matrix Profile in batch or incremental steps respectively.

Internally, STAMP uses the z-normalized Euclidean distance metric to compare subsequences. Originally, all subsequences were compared using the MASS algorithm [14] allowing the Matrix Profile to be calculated in $O(n^2 \log n)$, with n being the length of the series. The later introduced STOMP and SCRIMP algorithms [27, 26] reduced the runtime to $O(n^2)$ for both batch and incremental calculation respectively by applying dynamic programming techniques.

Various variations or enhancements of the Matrix Profile have been published. When users want to track the best earlier and later match of each subsequence, rather than the best global match, the left and right Matrix Profile can be calculated instead [24]. The Multidimensional Matrix Profile tracks the best matches between time-series containing multiple channels [20]. Zhu et al. have suggested a way to calculate the Matrix Profile when the data contains missing values, using knowledge about the range of the data [25]. Lastly, we presented the Contextual Matrix Profile [5] as a generalization of the Matrix Profile that is capable of tracking multiple matches over configurable time spans.

Different distance measures have also been proposed for the Matrix Profile. The Euclidean distance or more general p-norm, might be useful in areas such as finances, engineering, physics or statistics [1]. A distance measure that performs a non-linear transformation along the time axis and can ignore the prefix or suffix of sequences being matched, based on Dynamic Time Warping, has been suggested by Furtado Silva et al. [6]. Recently, we suggested the Series Distance Matrix framework [5] as a way to easily combine different distance measures with the techniques processing these distances in a plug-and-play way.

Once the Matrix Profile and corresponding Matrix Profile index have been calculated, they can be used for motif or discord discovery. In case the user wants to focus on specific parts of the signal, for example based on time regions or high-variance periods in the signal, they can shift the Matrix Profile using the Annotation Vector [3], allowing them to find different sets of discords or motifs.

The Matrix Profile can also be used as a building block for other techniques. Time Series Chains are slowly changing patterns that occur throughout a time series and can be found by analyzing the left and right Matrix Profile [24]. Time series segmentation involves detecting changes in the underlying behavior of a time series and is possible using the offline FLUSS or online FLOSS algorithm [7], which investigate the number of arcs defined by the Matrix Profile index to detect likely transitions. Classification of time series is possible through a dictionary of identifying patterns discovered through the Matrix Profile [21]. The Matrix Profile has also been shown useful for MDS, a data exploration technique that does not work well when visualizing all subsequences in a series, by selecting representative subsequences of series [22]. Lastly, MPDist [8], a distance measure that treats sequences similar if they share many similar subsequences, is calculated using the Matrix Profile and has been used to summarize large datasets for visualisation and exploration [9].

The Matrix Profile has been used in various techniques across many domains. However, series where flat and noisy regions are present have been mostly avoided in related literature, most likely due to the issue mentioned in Section 1. We suspect this issue affects any Matrix Profile based technique using the z-normalized Euclidean distance, and will especially have a negative impact on techniques dealing with discord discovery (such as anomaly detection) or techniques involving matches made on flat sequences (such as the assumption of motifs being present in homogeneous regions when performing time series segmentation). To the best of our knowledge, this issue has not yet been discussed or solved prior to our work. We show how to solve this issue in Section 4, after first discussing several relevant properties of the z-normalized distance measure in Section 3.

3 Properties of the Z-normalized Euclidean Distance

This section gathers aspects of the z-normalized Euclidean distance that are relevant for the remainder of this paper or when working with the Matrix Profile in general. Some properties listed here are obtainable through straightforward mathematical derivation of previously published properties, but have not yet been mentioned in Matrix Profile related literature, despite their high relevance.

3.1 Definition

The z-normalized Euclidean distance D_{ze} is defined as the Euclidean distance D_e between the *z-normalized* or *normal form* of two sequences, where the z-normalized form $\hat{X}$ is obtained by transforming a sequence X of length m so it has mean $\mu = 0$ and standard deviation $\sigma = 1$.

$$\hat{X} = \frac{X - \mu_X}{\sigma_X}$$

$$D_{ze}(X, Y) = D_e(\hat{X}, \hat{Y}) = \sqrt{(\hat{x}_1 - \hat{y}_1)^2 + \ldots + (\hat{x}_m - \hat{y}_m)^2}$$

3.2 Link with Pearson Correlation Coefficient

The z-normalized Euclidean distance between two sequences of length m is in fact a function of the correlation between the two sequences, as originally mentioned by Rafiei D. [16], though without the derivation we provide below.

$$D_{ze}(X,Y) = \sqrt{2m(1 - corr(X,Y))}$$

To derive this property, we first highlight the following property of the inner product of a z-normalized sequence with itself:

$$\sigma_X^2 = \frac{\sum_i^m (x_i - \mu_X)^2}{m}$$
$$m = \sum_i^m \left(\frac{x_i - \mu_X}{\sigma_X}\right)^2$$

Using this, we can derive the equality as follows:

$$\begin{aligned}
D_{ze}(X,Y)^2 &= \sum_i^m \left(\frac{x_i - \mu_X}{\sigma_X} - \frac{y_i - \mu_Y}{\sigma_Y}\right)^2 \\
&= \sum_i^m \left(\frac{x - \mu_X}{\sigma_X}\right)^2 + \sum_i^m \left(\frac{y - \mu_Y}{\sigma_Y}\right)^2 - 2\sum_i^m \left(\frac{x - \mu_X}{\sigma_X}\right)\left(\frac{y - \mu_Y}{\sigma_Y}\right) \\
&= 2m\left(1 - \frac{1}{m}\sum_i^m \left(\frac{x - \mu_X}{\sigma_X}\right)\left(\frac{y - \mu_Y}{\sigma_Y}\right)\right) \\
&= 2m(1 - corr(X,Y))
\end{aligned}$$

3.3 Distance Bounds

Since the correlation is limited to the range $[-1, 1]$, the D_{ze} between two sequences of length m will fall in the range $[0, 2\sqrt{m}]$, where zero indicates a perfect match and $2\sqrt{m}$ corresponds to the worst possible match.

As a result, the upper bound of $2\sqrt{m}$ can be used to *normalize distances* to the range $[0, 1]$, allowing us to compare matches of different lengths and enabling us to define and reuse thresholds to define degrees of similarity when using D_{ze}. This way, we can define a more uniform similarity threshold (e.g.: 0.3) for sequences of any length rather than specifying a threshold that is dependent on m (e.g.: a threshold of 3 for sequences of length 25, 6 for sequences of length 100 and so on). Note that Linardi et al. [12] had already pragmatically found the normalization factor $\sqrt{m}$ to compare matches of different lengths, though without making the connection to the underlying mathematics.

3.4 Best and Worst Matches

The distance bounds of Z_{ed} of 0 and $2\sqrt{m}$ correspond to correlation coefficients of 1 and -1 respectively. This means that for any sequence X of length m with $\sigma_X \neq 0$, $D_{ze}(X, Y) = 0$ and $D_{ze}(X, Z) = 2\sqrt{m}$ if:

$$\begin{aligned} Y &= aX + b \\ Z &= -aX + b \end{aligned}$$

for any values of a and b, where $a > 0$.

3.5 Effects of Noise on Self-Similarity

If we have a base sequence $S \in \mathbb{R}^m$ and two noise sequences $N \in \mathbb{R}^m$ and $N' \in \mathbb{R}^m$ sampled from a normal distribution $\mathcal{N}\left(0, \sigma_N^2\right)$, then the expected distance between the two sequences obtained by adding the noise to the base sequence can be expressed as follows:

$$\begin{aligned} X &= S + N \\ Y &= S + N' \end{aligned}$$

$$\mathbb{E}\left[D_{ze}\left(X, Y\right)^2\right] = (2m + 2)\,\frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2} \tag{1}$$

Note that in (1), σ_N^2 is the variance of the noise and $\sigma_S^2 + \sigma_N^2$ is the expected variance of either noisy sequence. We apply the derivation below, originally published in our previous work [4]. For the remainder of this section, we treat the sequences as random variables.

$$\begin{aligned} \mathbb{E}\left[D_{ze}(X, Y)^2\right] &= \mathbb{E}\left[(\hat{x}_1 - \hat{y}_1)^2 + \ldots + (\hat{x}_m - \hat{y}_m)^2\right] \\ &= m \cdot \mathbb{E}\left[(\hat{x} - \hat{y})^2\right] \\ &= m \cdot \mathbb{E}\left[\left(\frac{x - \mu_X}{\sigma_X} - \frac{y - \mu_Y}{\sigma_Y}\right)^2\right] \end{aligned} \tag{2}$$

Since X and Y are the sum of the same two uncorrelated variables, they both have the same variance.

$$\sigma_X^2 = \sigma_Y^2 = \sigma_S^2 + \sigma_N^2 \tag{3}$$

Next, we decompose μ_X and μ_Y in the component from the original sequence μ_S and the influence of the noise. Here we use n as a random variable sampled from the noise distribution. Note that μ_S can be seen as a constant as it refers to the mean of the base sequence.

$$\begin{aligned} \mu_X = \mu_Y &= \mu_S + \frac{n_1 + \ldots + n_m}{m} \\ &= \mu_S + \mu_N \\ \mu_N &\sim \mathcal{N}\left(0, \frac{\sigma_N^2}{m}\right) \end{aligned} \tag{4}$$

We perform the same decomposition for x and y, where s is an unknown constant originating from the base sequence:

$$\begin{aligned} x = y &= s + n \\ n &\sim \mathcal{N}\left(0,\, \sigma_N^2\right) \end{aligned} \tag{5}$$

Using (3), (4) and (5) in (2), canceling out constant terms and merging the distributions results in:

$$\begin{aligned} \mathbb{E}\left[D_{ze}(X,Y)^2\right] &= m \cdot \mathbb{E}\left[\left(\frac{n_x - n_y - \mu_{N_x} + \mu_{N_y}}{\sqrt{\sigma_S^2 + \sigma_N^2}}\right)^2\right] \\ &= m \cdot \mathbb{E}\left[(\nu)^2\right] \\ \nu &\sim \mathcal{N}\left(0,\, \frac{2+2m}{m} \cdot \frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2}\right) \end{aligned} \tag{6}$$

To finish, we apply the theorem $\mathbb{E}[X^2] = \mathrm{var}(X) + \mathbb{E}[X]^2$:

$$\mathbb{E}\left[D_{ze}(X,Y)^2\right] = (2m+2) \cdot \frac{\sigma_N^2}{\sigma_S^2 + \sigma_N^2} \tag{7}$$

4 Flat Subsequences in the Matrix Profile

While the utility of the z-normalized Euclidean distance as a shape-comparator has been proven by the many Matrix Profile related publications [22, 24, 12], results become counter-intuitive for sequences that contain subsequences that are flat, with a small amount of noise. While humans would consider such sequences as similar, the z-normalized Euclidean distance will be very high.

We can explain this effect in two ways and demonstrate this in Figure 1, where we visualize three pairs of noisy sequences that only differ by their slope. First, considering the Euclidean distance on z-normalized sequences, we can see in Figure 1 (middle) how the effect of noise becomes more outspoken for flatter sequences due to the normalization, resulting in a high Euclidean distance. Alternatively, we can consider the correlation of both sequences, as mentioned in Section 3.2. Looking at both sequences as a collection of points, shown in Figure 1 (bottom), we can see that flatter sequences more closely resemble the random distribution of the underlying noise and are therefor less correlated. Since a correlation of zero corresponds to a z-normalized Euclidean distance of $\sqrt{2m}$, or $\frac{1}{\sqrt{2}} \approx 0.707$ if we rescale this value using the distance bounds mentioned in Section 3.3, we can see that uncorrelated sequences will have a high distance.

The effect of flat, noisy subsequences will have a negative effect on some use cases of the Matrix Profile. Since the flat sequences result in high Matrix Profile values where we would intuitively expect low values, we can estimate which use

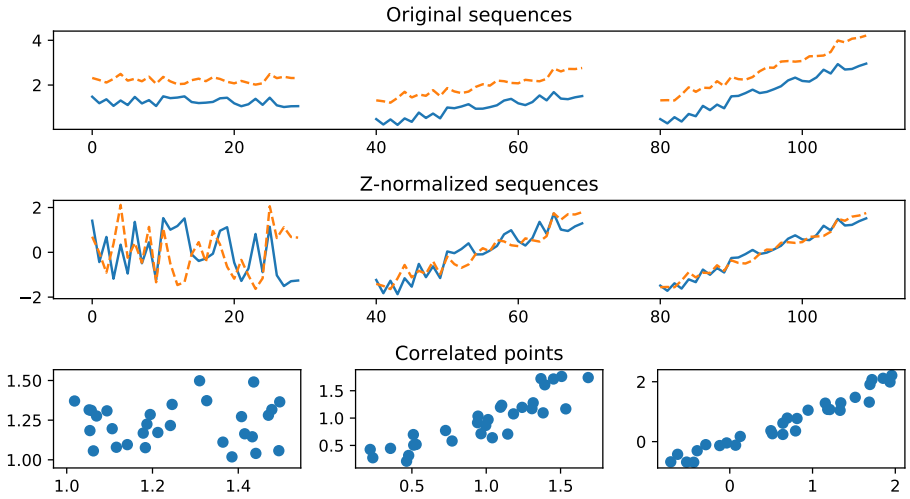

Fig. 1. Three pairs of sequences with varying slopes, each pair has the same noise profile. By looking at the effect of the noise in the z-normalized sequences, we see why the Euclidean distance will return much larger distances for flat sequences. At the bottom we see a visualization of the correlation between both sequences, where we see that the slope of the signal has a major influence on the corresponding correlation.

cases will suffer and which will not. For example, anomaly detection or discord discovery using the Matrix Profile involves finding the highest values in the Matrix Profile. When flat, noisy sequences are present, true discords may be hidden by this effect. Another example is the semantic segmentation technique using the Matrix Profile [7], this technique detects transitions in a signal by analyzing the matches of each subsequence, assuming homogeneous regions will contain many good matches. In this case, homogeneous regions containing flat and noisy sequences will result in poor matches, violating the base principle of the segmentation technique. Notably, motif detection will not suffer from this issue, assuming the user is not interested in flat motifs.

Next, we will discuss seemingly useful resolutions that do not actually manage to solve this effect before presenting our own solution. First however, we introduce a synthetic dataset that will serve as our running example in this section.

4.1 Running Example

We generated a sinusoid signal of 2000 samples and introduced an anomaly in one of the slopes by increasing the value of 10 consecutive values by 0.5 and create a noisy copy by adding Gaussian noise sampled from $\mathcal{N}(0, 0.01)$. The Matrix Profile for both signals was calculated using a subsequence length m of 100 and a trivial match buffer of $\frac{m}{2}$, as recommended in [23]. The signal and

corresponding Matrix Profile are displayed in Figure 2. For the noise-free signal, we see exact matches (distance equal to zero) everywhere except in the region containing the anomaly. For the noisy signal, we see how the Matrix Profile has shifted upwards, as would be expected since exact matches are no longer possible. However, we also see previously non-existing peaks in the Matrix Profile where the signal was more flat, because of this the anomaly is no longer trivial to locate automatically.

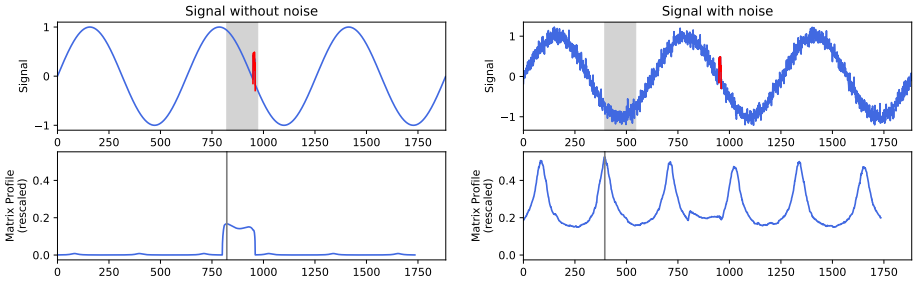

Fig. 2. Top: Sinusoid signal without (left) and with (right) added Gaussian noise. An anomaly of length 10 (red) was introduced at index 950. Bottom: Corresponding Matrix Profile for both signals, rescaled using the method of Section 3.3. The top discord is marked in gray. As can be seen, the presence of noise increases the Matrix Profile values of the flat regions to the degree that they now hide the true anomaly. This figure is modified from our previous work [4].

Let us briefly further investigate how the properties of the noise affect the Matrix Profile in this example. Figure 3 displays our starting sinusoidal signal with anomaly, to which Gaussian noise sampled from different distributions was added. As expected, we see that as the variation of the noise increases, the Matrix Profile becomes more deformed. The anomaly is no longer visually obvious in the Matrix Profile for noise with standard deviation of 0.05 or more. Somewhat surprising is how quickly this effect becomes apparent: when the noise has a standard deviation of around 0.02 (at this point the signal-to-noise ratio is 1250 or 31 dB), the anomaly is already occasionally overtaken as the top discord by the flat subsequences (depending on the sampling of the noise).

Before coming to our solution, we will discuss why some simple, seemingly useful methods to circumvent this problem do not work.

– **Changing the subsequence length m**: as m becomes smaller, the effect of any anomaly on the Matrix Profile will indeed increase. However, as the subsequences become shorter and relatively flatter as a result, the effect of the noise also becomes bigger, resulting in a more eratic Matrix Profile. Increasing m will have a beneficial effect, but this is simply because the longer subsequences will become less flat in this specific example, so this is not a general solution. This approach is demonstrated in Figure 4 (top left).

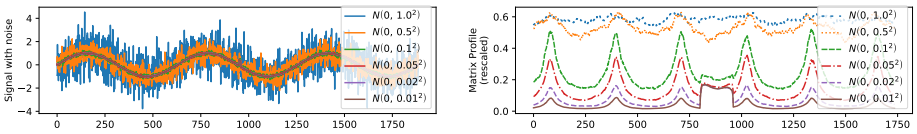

Fig. 3. A demonstration of the effect of varying degrees of noise on the Matrix Profile. Left: the sinusoidal signal with noise sampled from various Gaussian distributions. Right: The corresponding rescaled Matrix Profile. This figure is modified from our previous work [4].

- **Ignoring flat sections**: ignoring subsequences whose variance is below a certain value would result in the removal of the peaks in the Matrix Profile. A first problem with this is that finding the correct cutoff value is not trivial. Secondly, this approach will not be applicable in datasets where the flat subsequences are regions of interest, either as anomalies or for finding similar subsequences, as is demonstrated in the time series segmentation of Section 6. This approach is visualized in Figure 4 (middle left).
- **Smoothing or filtering**: by preprocessing the noisy signal, one could hope to remove the noise altogether. Unfortunately, unless the specifics of the noise are well known and the noise can be *completely* separated from the signal, there will always remain an amount of noise. As was shown in Figure 3, even a small amount of noise can have a large effect on the Matrix Profile. This approach is demonstrated in Figure 4 (bottom left).

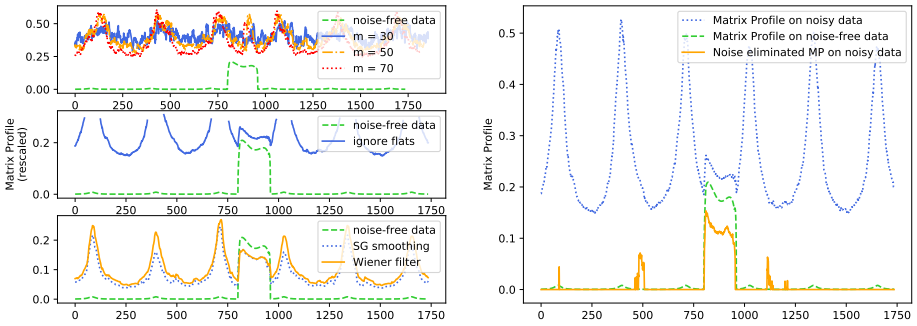

Fig. 4. Left: The effects of several seemingly useful methods to combat the effect of flat, noisy subsequences that in fact do not work. From top to bottom: reducing the subsequence length, ignoring flat sections and smoothing/filtering. None of these methods approach the noise-free Matrix Profile. Right: The effect of our noise elimination technique on the Matrix Profile. We see how the corrected Matrix Profile closely resembles the Matrix Profile of the noise-free signal. This figure is modified from our previous work [4].

4.2 Eliminating the Effect of Noise

Ideally, we want flat subsequences to have good matches with other flat subsequences. This would be the case if those flat subsequences were stretched and/or shifted versions of one another, as mentioned in Section 3.4. Unfortunately, this is not the case due to the effects of noise. We can however, still consider them to be identical, in which case we can use our derivation from Section 3.5, which estimates the effect of the noise on the z-normalized Euclidean distance. By subtracting this estimate during the calculation of the Matrix Profile, we are actively negating the effects of the noise. The only requirement is that we known the standard deviation of the noise that is present in the signal. This may be either known in advance or can be easily estimated by analyzing a flat part of the signal. Note that we also need the standard deviation of the subsequences being compared, but as these are already needed for the distance calculation [27, 26], these are precalculated and available as part of the Matrix Profile calculation.

The algorithm is straightforward, after calculating the squared distance between a pair of subsequences using any of the existing algorithms, we subtract the squared estimate of the noise influence. We do this *before* the element-wise minimum is calculated and stored in the Matrix Profile, because this correction might influence which subsequence gets chosen as the best match. Pseudo code is listed in Algorithm 1 and can run in $O(1)$ runtime.

Algorithm 1: Algorithm for Eliminating the Effects of Noise

Input: d: distance between subsequence X and Y
Input: m: subsequence length
Input: σ_X, σ_Y: standard deviation of subsequence X and Y
Input: σ_n: standard deviation the noise
Output: corrDist: corrected distance between subsequence X and Y

1 $corrDist = \sqrt{d^2 - (2+m)\frac{\sigma_n^2}{max(\sigma_X, \sigma_Y)^2}}$

The only difference between this code and the formula from Section 3.5 is that we use the maximum standard deviation of both subsequences. When processing two fundamentally different subsequences, this choice effectively minimizes the effect of the noise elimination technique.

We demonstrate our technique on the running example in Figure 4 (right). We see that unlike the previously methods, we can closely match the Matrix Profile of the noise-free signal. We do see some small residual spikes, which appear depending on the sampling of the noise, they are caused by local higher-than-expected noise values in that part of the signal.

After demonstrating our noise elimination technique on a limited synthetic dataset, we will use the remainder of this paper to prove the merit of our noise elimination method for several use cases using real-world datasets.

5 USE CASE: ANOMALY DETECTION

One of the original applications for the Matrix Profile is the discovery of discords, where a discord is the subsequence in a series that differs most from any other

subsequence. Discords in fact correspond to the subsequences starting at the indices where the Matrix Profile is highest. When interested in the top-k discords, one can take the top-k values of the Matrix Profile where each value should be at least m index positions away from all previous discord locations. This requirement ensures we cannot select overlapping subsequences as discords, as these basically represent the same anomaly [13].

In this section we demonstrate the benefit of our noise elimination technique when performing anomaly detection utilizing real-world data from Yahoo. In our previous work [4], we performed a similar experiment using the "realAWSCloudwatch" collection from the Numenta Anomaly Benchmark [11].

We use the **Labeled Anomaly Detection Dataset of Yahoo! Webscope**, consisting of both real and synthetic time-series. For this paper we focus on the "A1Benchmark" dataset, which contains real traffic metrics from Yahoo! services, reported at hourly intervals. The benchmark consists of 67 time series with labeled anomalies, ranging from 741 to 1461 data points. The time-series vary considerable, containing diverse ranges, seasonality, trends, variance, among other properties. Figure 5 shows some examples.

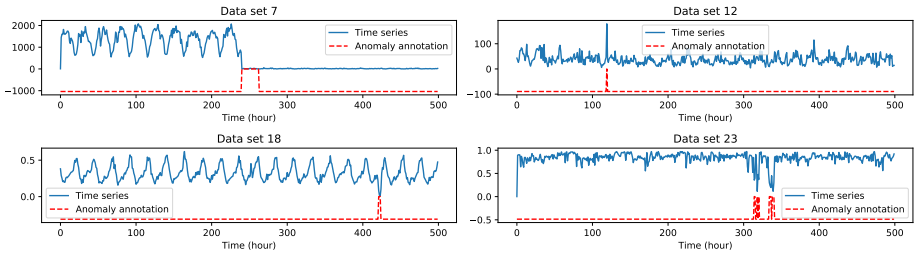

Fig. 5. Extracts of four series from the Yahoo! Webscope anomaly dataset.

Rather than classifying each point in the time series as anomalous or normal, which would involve optimizing a classification threshold, we instead score performance by counting the number of attempts needed before all anomalies in a series are reported, or until 10 incorrect guesses are made, as was done in our previous work for the Numenta benchmark [4]. This way of scoring resembles a user being alerted with suspected anomalies, measuring the capability of the algorithm to present relevant anomalies.

We perform anomaly detection by self-joining each series with a subsequence length of 24 (one day) and using the left Matrix Profile [24] for anomaly detection. The left Matrix Profile only tracks matches preceding each subsequence, similar to how streaming data is processed, and increases the chance to treat sudden changes as discords.

Since we do not know the characteristics of the noise, we would need to estimate the standard deviation using the signal. However, this is not a trivial task since we do not known in advance which signals contain noise and which

do not. Making an inaccurate estimate by assuming a non-noisy signal is in fact noisy would result in poor predictions.

To detect the presence of noise, we devised a heuristic where we evaluate the Matrix Profile values of the first three days of data. If the average rescaled Matrix Profile value is above a certain threshold, we assume the start of the data is noisy and we take the median standard deviation of all subsequences in the first three days as noise parameter. Based on the first 10 datasets and leaving all other datasets as test set, we manually determined a threshold of 0.2.

This heuristic marked 34 out of 67 datasets as noisy. We compared the anomaly detection results for these 34 datasets with and without our noise elimination technique. The results are displayed in Table 1, they show that our noise elimination technique performed better for 32 out of 34 datasets. On average, the regular Matrix Profile found 36 out of a total of 84 anomalies using 291 incorrect guesses, after applying our technique this improved to 79 found anomalies using only 80 incorrect guesses. This means that on average, one in two suspected anomalies turned out to be correct!

Figure 6 shows two close-ups demonstrating the effect of the noise elimination technique. It shows the Matrix Profile having a somewhat consistent high value, whereas the noise eliminated version only increases near the actual anomalies.

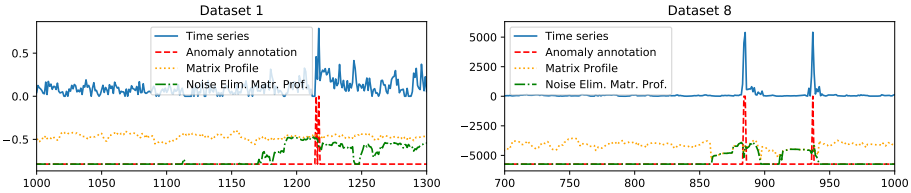

Fig. 6. Two examples displaying the beneficial effect of our noise elimination on anomaly detection. The anomalies are not noticeable in the regular Matrix Profile, but are obvious after applying noise elimination.

Our method was unable to find all anomalies for four of the datasets. Two of these are shown in Figure 7. In dataset 53 the first two anomalies were not detected due to their similarity with other flat series. In dataset 61 the algorithm behaves similarly to the original matrix profile due to the sudden increase in the noise level and becomes unable to differentiate anomalies from noise. In this case the first anomaly is found but the second one is missed.

6 Use Case: Semantic Segmentation for Time Series

Semantic segmentation of time series involves splitting a time series into regions where each region displays homogeneous behavior, these regions typically correspond to a particular state in the underlying source of the signal. Applications of segmentation may include medical monitoring, computer-assisted data

Table 1. Results of anomaly detection using the Matrix Profile with and without noise elimination on the Yahoo! Webscope anomaly dataset. For each dataset, we kept guessing until all anomalies were found or 10 incorrect guesses were made. When using noise elimination we were able to find most anomalies with few attempts.

		Without Noise Eliminaton		With Noise Elimination	
Dataset	# Anomalies	Found Anomalies	Wrong Guesses	Found Anomalies	Wrong Guesses
1	2	0	10	2	0
2	2	1	10	2	3
4	3	0	10	3	0
5	1	0	10	1	0
6	1	0	10	1	0
8	3	0	10	3	5
10	1	0	10	1	0
11	1	0	10	1	0
12	2	1	10	2	1
14	1	1	8	1	0
17	3	1	10	3	2
19	3	0	10	3	0
21	2	1	10	2	1
22	1	1	8	1	0
23	12	5	10	12	3
24	3	3	3	2	10
25	1	1	2	1	0
31	2	0	10	2	1
32	2	2	5	2	1
33	1	0	10	1	1
40	2	1	10	2	9
41	3	1	10	3	3
42	3	0	10	3	7
43	3	2	10	3	1
45	1	0	10	1	0
48	1	0	10	0	10
50	1	1	2	1	0
53	4	4	3	2	10
58	1	0	10	1	0
61	2	0	10	1	10
62	4*	0	10	4	1
63	1	0	10	1	0
66	6	5	10	6	1
67	5	5	0	5	0
Sum	84	36	291	79	80

* Dataset 62 actually contains five anomalies, but because the first anomaly occurs within the first three days which are used to estimate the noise level, we do not consider it in the results.

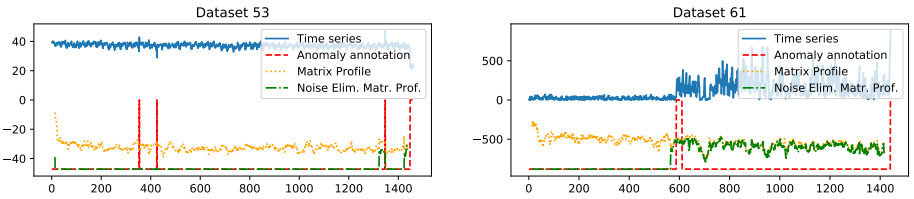

Fig. 7. Left: Dataset containing two anomalies which are not detected as they closely resemble the estimated noise. The noise eliminated Matrix Profile is zero for this segment. Right: Dataset where the original amount of noise is small but increases at one point, causing the Noise Elimination to lose its effect.

annotation or data analysis in general. In this section, we perform semantic segmentation on the PAMAP2 activity dataset using the Corrected Arc Curve (CAC). The CAC is calculated by the FLUSS algorithm for batch data or the FLOSS algorithm for streaming data, using the Matrix Profile index [7].

The CAC was introduced as a domain agnostic technique to perform time series segmentation on realistic datasets, with support for streaming data while requiring only a single intuitive parameter (the subsequence length to consider). During evaluation the CAC was found to perform better than most humans on dozens of datasets, allowing the authors to claim "super-human performance" [7].

The CAC is a vector of the same length as the Matrix Profile, and is constructed by analyzing the Matrix Profile Index. They consider arcs running from each subsequence to the location of its nearest match. To calculate the CAC, they compare the number of arcs running over each location against the amount of arcs expected if all match locations would be determined by uniform sampling over the entire series. This ratio is defined as the CAC, its values are strictly positive without an upper bound, but can be safely restricted to the range $[0, 1]$. Assuming homogeneous segments will display similar behavior while heterogeneous segments will not, a low CAC value is seen as evidence of a change point, though a high CAC value should not be seen as evidence of the absence of one.

We use the **PAMAP2 Activity Dataset** [17], which contains sensor measurements of 9 subjects performing a subset of 18 activities like sitting, standing, walking, and ironing. Each subject was equipped with a heart rate monitor and 3 inertial measurement units (IMU) placed on the chest, dominant wrist and dominant ankle. Each IMU measured 3D acceleration data, 3D gyroscope data and 3D magnetometer data at 100 Hz. The time series are annotated with the activity being performed by the subject and transition regions in between activities. The duration of each activity varies greatly, but most activities last between 3 to 5 minutes.

The PAMAP2 dataset has already been used in the context of segmentation [20], where the authors used the Matrix Profile to classify the activities in passive and active activities. At one point they note that the motif pairs in the passive actions (such as lying, sitting or standing) are less similar and therefore

less useful for segmentation. The underlying problem here is the inability to detect motifs in the passive activities, because they mainly consist of flat, noisy signals. By using our method to compensate for this effect, we will be able to find the needed motifs, resulting in better segmentation results.

We applied time series segmentation on the passive activities present in the PAMAP2 dataset, focusing on the "lying", "sitting" and "standing" activities. We picked these activities since their measurements display very few patterns in the data and they are performed consecutively for all subjects, meaning we did not have to introduce time-jumps in our experiments.

We considered subjects 1 to 8 of the dataset (subject 9 had no recordings of the relevant activities) and tested both the transition from "lying" to "sitting" as well as the transition from "sitting" to "standing". For each subject, we used the 3 accelerometer signals from the IMU placed on the chest of the subject, any missing data points were filled in using linear interpolation. We calculated the CAC by self-joining each sensor channel with and without noise elimination using a subsequence length of 1000 (10 seconds). The standard deviation of the noise was estimated (without optimizing) by taking the 5th percentile of the standard deviations of all subsequences.

An example of the signals spanning over the 3 passive activities can be seen in Figure 8. We emphasize it is not our goal to build the optimal segmentation tool for this specific task, but to simply evaluate the effect of our noise elimination technique on the CAC for sensor signals containing flat and noisy subsequences.

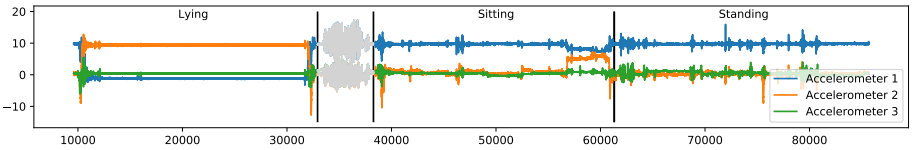

Fig. 8. Three accelerometer channels of subject 6 from the PAMAP2 dataset. We see three activities and one long transition period. No clear patterns are discernible and many flat and noisy subsequences are present. Reproduced from our previous work [4].

There is one unexpected side effect of the noise cancellation technique that needs to be corrected before calculating the CAC. Because most of the flat subsequences will have an exact match (distance equal to zero) to other flat subsequence, there will be many locations to represent the best match. However, since the Matrix Profile Index only stores one value, the selected best match will become determined by the first or last match, depending on the implementation of the Matrix Profile. This creates a pattern in the Matrix Profile Index, that actually violates the CAC's assumption of matches being spread out over a homogeneous region. Note that this effect is in fact also present in the normal Matrix Profile, but typically has goes unnoticed because multiple exact matches are extremely rare.

To prevent this effect, we need to randomly pick one of the best matches and store its location in the Matrix Profile Index. This is straightforward when using the STOMP algorithm [27], as every step in STOMP calculates all matches for one particular subsequence. If we want to calculate the Matrix Profile in an online fashion using the SCRIMP algorithm [26], where the matches for one specific subsequence are spread over many iterations, we need to utilize reservoir sampling [18] in the construction of the Matrix Profile Index. Reservoir sampling allows uniform sampling without replacement from a stream without knowing the size of the stream in advance. We use it to sample the stream of best matches. Implementing reservoir sampling requires us to store an additional vector of the same length as the Matrix Profile, to keep track of the number of exact matches that was encountered so far for each subsequence. Pseudo code to update the Matrix Profile and its indices is listed in Algorithm 2.

Algorithm 2: SCRIMP Matrix Profile Update using Reservoir Sampling

Input: dists: distances on diagonal calculated by SCRIMP
Input: indices: corresponding indices of dists
Input: numMatches: number of exact matches per subsequence
Input: mp: part of Matrix Profile vector being updated
Input: mpi: part of Matrix Profile Index being updated

```
   /* Handle new better matches                                    */
1  better = dists < mp
2  mp[better] = dists[better]
3  mpi[better] = indices[better]
4  numMatches[better] = 1
   /* Handle matches equal to current best match                   */
5  equal = dists == mp ∧ finite(dists)
6  numMatches[equal] = numMatches[equal] + 1
7  for i in equal do
8      if random() < 1/numMatches[i] then
9          mpi[i] = indices[i]
```

The code is straightforward. In lines 1 to 3 we update the Matrix Profile and Index if a better match was found and in line four we reset the tracked number of exact matches. In line five, we gather any matches equally good as the match being tracked in the Matrix Profile. Line six increases the counter of any equally good matches found and line seven to nine perform the reservoir sampling to update the Matrix Profile Index for each newly found equal match.

For both experiments, the CAC was calculated using the Matrix Profile with randomly sampled indices. The activity-transition point was taken where the CAC was minimal, ignoring any values in the first and last 50 seconds (5 times the subsequence length), as suggested by the original paper [7]. We considered 4 segmentations per subject: one CAC for each of the three sensor channels and one obtained by averaging the individual CACs.

To evaluate the ability to predict the transition period, we define the score as the normalized distance between the predicted transition and the ground truth transition. Note that some ground truth transitions are instantaneous, while others consist of a transition period, as can be seen in Figure 8. We also added an additional buffer period equal to the subsequence length m (10 seconds) before and after the transition period that we still consider as correct to consider the detection interval of the Matrix Profile. Pseudo code for our scoring function is listed in Algorithm 3, a score will range from 0 to 100, where lower is better.

Algorithm 3: Scoring Function for Semantic Segmentation

Input: estimate: estimated transition
Input: trueStart, trueEnd: ground truth start and end of transition
Input: n: length of series (both activities and transition period)
Input: m: subsequence length / transition buffer
Output: score

if $estimate < trueStart - m$ **then**
  $score = ((trueStart - m) - estimate)/n * 100$
else if $estimate > trueEnd + m$ **then**
  $score = (estimate - (trueEnd + b)/n * 100$
else
  $score = 0$

Table 2. Scores for the segmentation of the transition from "lying" to "sitting" using the 3 chest accelerometers from the PAMAP2 dataset for subjects 1 through 8, with and without noise elimination applied. Segmentation is performed using the CAC from a single sensor (C1, C2 and C3) and using the average of the 3 CACs (combined). Similar or better performance are achieved when applying noise elimination for all subjects except subject 1. Results are reproduced from our previous work [4].

	Without Noise Elimination				With Noise Elimination			
Subject	C1	C2	C3	Combined	C1	C2	C3	Combined
1	**5.9**	31.3	**31.9**	**31.7**	41.3	31.8	41.8	36.7
2	32.9	1.4	1.4	1.4	28.8	1.4	1.7	1.4
3	35.9	2.8	31.1	33.8	**2.4**	2.3	**2.3**	**2.3**
4	0.0	2.8	5.9	0.0	0.0	1.5	6.6	0.8
5	1.1	7.6	5.1	3.9	1.6	**1.7**	4.9	1.6
6	2.5	1.9	2.3	2.3	2.4	1.9	2.0	2.4
7	0.1	1.8	11.1	2.0	2.1	1.8	**1.9**	1.9
8	0.0	1.4	5.5	1.7	0.0	1.4	1.4	1.4
Average			9.32	9.61			7.71	6.07

Table 3. Scores for the segmentation of the transition from "sitting" to "standing" using the 3 chest accelerometers from the PAMAP2 dataset for subjects 1 through 8, with and without noise elimination applied. Segmentation is performed using the CAC from a single sensor (C1, C2 and C3) and using the average of the 3 CACs (combined). Overall, we see similar or better performance when applying noise elimination, except for the segmentation using the first channel for subject 1, 3 and 8. Results are reproduced from our previous work [4].

	Without Noise Elimination				With Noise Elimination			
Subject	C1	C2	C3	Combined	C1	C2	C3	Combined
1	**32.5**	0.0	3.6	2.2	38.7	0.0	3.7	2.2
2	36.5	37.2	36.4	37.0	**7.1**	**30.0**	32.7	**29.2**
3	**10.0**	30.2	43.1	**30.2**	43.2	**14.0**	43.7	43.2
4	7.8	1.9	1.1	1.2	**0.7**	2.0	1.3	1.3
5	13.1	0.0	28.5	10.6	13.3	1.0	**1.2**	**1.0**
6	36.1	36.6	26.9	36.6	**23.3**	**3.4**	26.5	**3.2**
7	43.1	38.0	16.5	16.5	43.4	**1.6**	**0.0**	**1.6**
8	**2.3**	1.0	24.8	1.0	21.1	0.0	**16.5**	1.0
Average			21.12	16.9			**15.35**	**10.3**

Table 2 lists the results for the segmentation when transitioning from "lying" to "sitting". For all subjects expect subject one, the results show similar of improved scores for segmentation using individual sensors as well as the combined approach when using the noise elimination technique. The average score for the individual sensors improves from 9.32 to 7.71, a modest improvement corresponding to a gain of about 8 seconds. The segmentation based on the combined CACs improves from 9.61 to 6.07, a gain of about 18.5 seconds. Note that most scores without noise elimination were already very good, leaving little room for improvement. The bad results for subject one can be explained by an incorrect early estimate which is caused by movement of the subject near the start of the "lying" activity. Note that subject one has bad scores for both techniques.

Table 3 lists the results for the transition from "sitting" to "standing". Like the previous experiment, we see similar or improved results when applying noise elimination, except for subjects one, three and eight using the first sensor series and for the combined approach for subject three. While the overall scores are worse, the gain by enabling noise elimination is more significant. The average result for a single sensor improves from 21.12 to 15.35, corresponding to a gain of about 27 seconds. When using the combined approach the average score improves from 16.9 to 10.3, a gain of around 31 seconds.

Though limited in scope, these result indicate that the noise elimination technique improves the ability of the CAC to detect transitions in a series containing flat and noisy subsequences.

7 Use Case: Data Visualization

Data visualization is a great tool for exploring newly acquired data or for finding similar data in a larger data collection. Unfortunately, time series data for realistic use cases is typically very lengthy, making trivial visualizations useless as these will fail to simultaneously capture the overall and minute details of a series. More advanced techniques summarize the series in a way that is still useful to gain insight into the data. The visualization technique used in this section is the Contextual Matrix Profile (CMP), recently introduced by the authors [5].

The CMP is a generalization of the Matrix Profile that tracks the best match between predefined ranges whereas the Matrix Profile tracks the best match for every possible sliding window location. The distinction between the two can also be made in terms of the implicit distance matrix, defined by the pairwise distance between all subsequences of two series. Where the Matrix Profile equals the column-wise minimum of the distance matrix, the CMP consists of the minimum over rectangular areas of the distance matrix.

For this use case, we use the **Chest-Mounted Accelerometer Dataset** [2], an activity recognition dataset publicly available at the UCI repository[1]. The dataset contains data of 15 subjects performing seven different activities, measured using a chest-mounted accelerometer sampling at 52Hz. The data is labeled with the corresponding activity, though visual inspection reveals the labels seem misaligned for some subjects. The activities performed are: Working at a computer; standing up, walking, going up/down stairs; standing; walking; going up/down stairs; walking while talking; talking while standing. We selected this dataset for this use case as it contains both activities with a periodic nature as well as passive activities where the accelerometer signal consists of mainly noise.

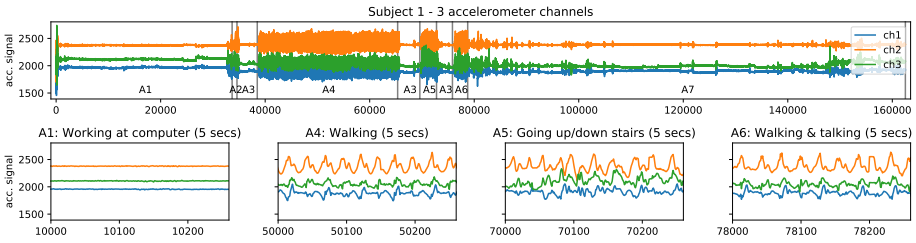

Fig. 9. Top: Uncalibrated accelerometer data (3 channels) for subject 1, sampled at 52Hz, for a total of 52 mins. The annotations A1 to A7 indicate the corresponding activity labels. The activities performed are computer work, standing up/walking/stairs, standing, walking, standing, stairs, standing, walking while talking, and talking while standing. Bottom: four extracts of 4 different activities, each 5 secs long.

[1] https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer

For the remainder of this section, we focus on the first subject of the dataset due to space constraints, though similar results were obtained for all subjects. This series comprises 52 minutes of data containing nine regions of activity, it is visualized in Figure 9. In the top of the figure we see the complete dataset with annotations indicating the activity regions. At the bottom of the figure, close-ups of the signal for four different activities are displayed. While the "working at computer" consists mainly of a flat singal, a periodic pattern is visible in the channels of the other activities, with the "walking" activity having the clearest pattern. Note that the three data channels are uncalibrated, which is not an issue since the z-normalization focuses on the shape of subsequences rather than the absolute values.

The CMP for each data channel is calculated by self-joining the data, using a subsequence length of 52 (1 sec) and specifying contexts of length 469 at 520 (10 secs) intervals. The context length is chosen so that matches can never overlap. Calculating the CMP comes down to dividing the series in non-overlapping, contiguous 10 sec windows and finding the best one second match for each pair of windows. The resulting CMP is a 312 by 312 matrix, each value representing the distance of the best match of the intervals defined by the row and column. The resulting CMPs for each data channel can be seen in Figure 10 (left). To demonstrate the value of the visualization, we also added grayscale band to the CMP outline that corresponds to the activity labels present in the data.

To interpret a CMP visualization, one should consider both axes represent the flow of time, starting at the origin. Each value in the CMP indicates how well one region of time matched another region, low (dark) values represent good matches while high (light) values represent bad matches. Looking at Figure 10 (left), we can observe a number of things. Most obvious is the symmetry of each CMP, this is because we performed a self-join. We also see the dark square centered at index 100 in all three channels, this indicates a period containing a repetitive pattern across all channels. This region corresponds to the "walking" activity in the dataset, as can be seen by referencing Figure 9. Next, for channel two we see similar dark regions for the "stairs" and "walking while talking" activities, meaning there is a similarity between all three activities based on channel two. Though these activities also appear in channel one and three, they are less visually noticeable, especially the "stairs" activity in channel one could be easily missed. The same can be said for the very short "standing up/walking/stairs" activity that precedes the walking activity. One final observation is the presence of high value bands occurring across all channels to some degree between indices 0 and 60 and between 200 and 250. While the first band corresponds to the "computer work" activity, there is no clear-cut corresponding activity for the second band. Most likely, these artefacts are caused by regions with very flat signals, where the noise has a large effect on the distance calculation.

Next, we calculated the CMPs while using the noise elimination technique, keeping all other parameters equal. We estimated the noise parameter for each channel by sliding a one second window over the entire series, calculating the standard deviation for each location and taking the value corresponding to the

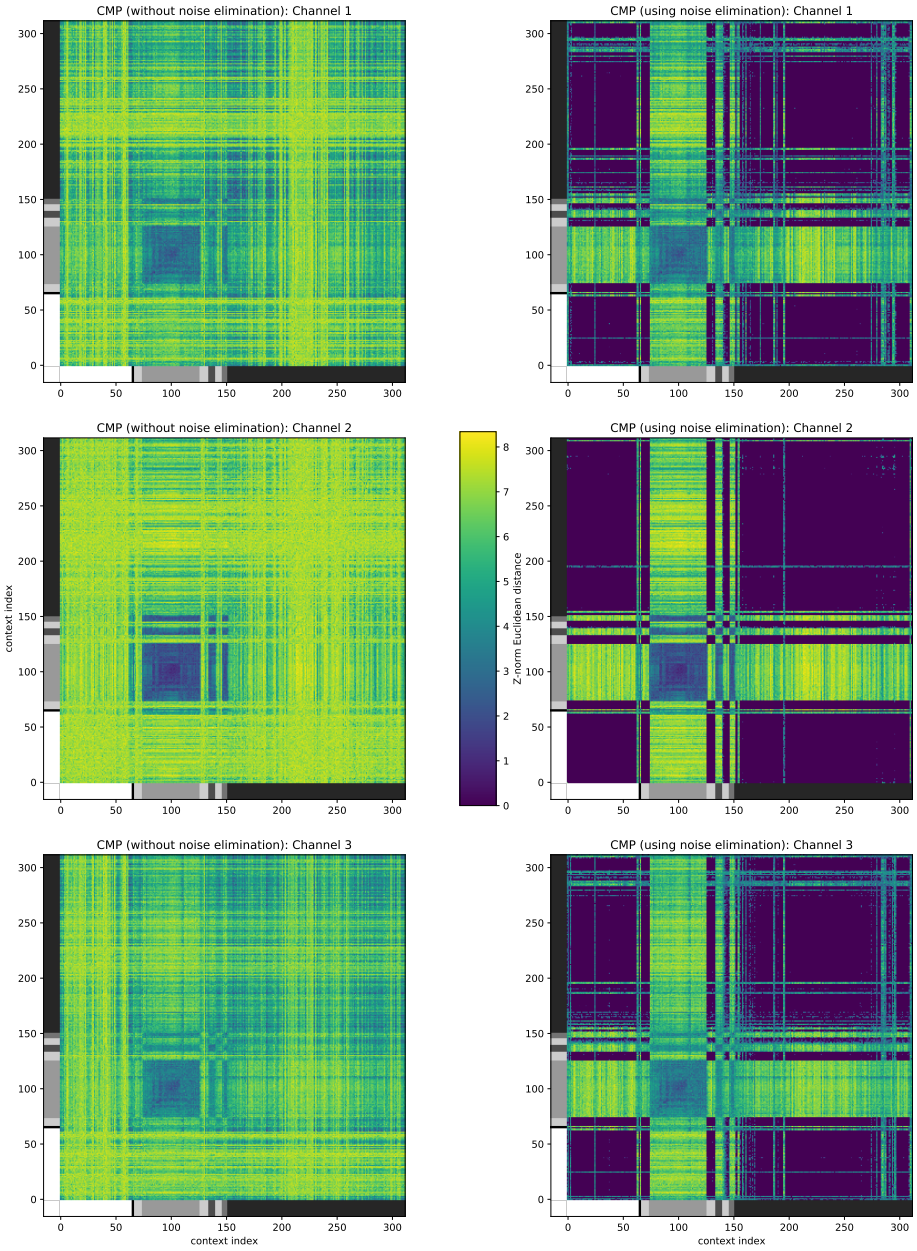

Fig. 10. CMPs produced for each channel of the dataset using z-normalized Euclidean distance without (left) and with (right) compensating for the noise. The left and bottom grayscale bar for each CMP corresponds to the different activity labels for each of the time windows. We see how the noise elimination results in large areas of exact matches for the passive activities, because of this, the different transitions between active and passive activities is clearly visible. In case the activity labels were unknown, the CMP would have given a good indication of the different regimes present in the signal.

fifth percentile, similar as we did in Section 6. For reference, these values were: 2.6, 2.3 and 2.7 respectively. The resulting CMPs are visualized in Figure 9 (right).

We see the CMPs generated using noise elimination now have additional large, rectangular regions with low values. As can be seen from the activity markings, these regions correspond to the passive activities (computer work, standing and talking while standing), where the series is flat and lacks distinctive patterns. Looking in detail, we see how the activity markings almost perfectly line up with the transitions in the CMP, which is major difference with the CMPs without noise elimination. We do see some line artefacts in the final activity of the dataset for all channels, near index 190 and 280, which correspond to increases in the signal on all three channels. The questions whether or not there is an activity change occurring there is in a way debatable and may simply be a question of tweaking the noise parameter or refining the activity labels.

Of course, the CMP visualization can provide more insights than simply the difference between passive and active activities. It can also be used to differentiate between different activities, provided these activities have different underlying patterns. As an example, we can see that for the CMP of the first channel, the activities "walking" and "walking while talking" have better matches than "walking" and "stairs". This is not the case for channels two and three. When looking at the closeup data of Figure 9, we can in fact see a more distinct pattern for channel one for the "stairs" activity. We do not quantify the ability to discern various activities as it is not in scope of this paper, our goal was simply to demonstrate the added benefit of noise elimination when visualizing data with the CMP.

To conclude, we demonstrated the effect of the noise elimination technique for data visualization using the CMP on accelerometer data for an activity dataset. Flat signals, such as those from recording passive activities, will result in high values in the CMP and might make it difficult to see the patterns of the underlying data. If we apply the noise elimination technique while calculating the CMP, the passive activities become easily discernible as regions of low values, giving the user better insight in the underlying structure of the data and allowing the user to focus more on the more salient parts. Importantly, the regions with active activities are unaffected by the noise elimination, meaning we can apply noise elimination without risk.

8 Conclusion

In this paper we explained the unintuitive behavior of the z-normalized Euclidean distance when comparing sequences that are flat and noisy, and demonstrated how this negatively affects the Matrix Profile and techniques using the Matrix Profile as building block. We discussed several properties of the z-normalized Euclidean distance, including an estimation of the effect of noise, which we use to eliminate this effect altogether.

We applied our noise elimination technique on three different use cases involving real-world data from open data sets. For anomaly detection on the Yahoo! Webscope anomaly dataset, we were able to automatically guess twice as many anomalies while utilizing less than one third of attempts when using our technique. When used for semantic time series segmentation, we showed an improved accuracy for detecting the transition between two passive activities. Finally, in our visualization use case, we showed a major change in the visualization of activity data using the Contextual Matrix Profile, allowing us to separate the underlying activities that were previously indistinguishable.

Since our technique is conceptually simple, users should be able to reason whether or not their use case will benefit from our technique. Our technique is straightforward to implement and incurs only a constant factor overhead, so it can be used by everyone using Matrix Profile related techniques working with data containing flat and noisy subsequences.

Future remains on more robust noise estimations and dealing with series where noise characteristics change over time.

References

1. Akbarinia, R., Cloez, B.: Efficient Matrix Profile Computation Using Different Distance Functions (2019), https://arxiv.org/abs/1901.05708
2. Casale, P., Pujol, O., Radeva, P.: Personalization and user verification in wearable systems using biometric walking patterns. Personal and Ubiquitous Computing **16**, 1–18 (06 2012)
3. Dau, H.A., Keogh, E.: Matrix Profile V: A Generic Technique to Incorporate Domain Knowledge into Motif Discovery. In: Proc. 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 125–134 (2017)
4. De Paepe, D., Janssens, O., Van Hoecke, S.: Eliminating noise in the matrix profile. In: Proc. 8th International Conference on Pattern Recognition Applications and Methods. pp. 83–93 (feb 2019)
5. De Paepe, D., Vanden Hautte, S., Steenwinckel, B., De Turck, F., Ongenae, F., Janssens, O., Van Hoecke, S.: Generalizing the Matrix Profile. Engineering Applications of Artificial Intelligence Submitted 2019-06-06
6. Furtado Silva, D., E. Batista, G.: Elastic Time Series Motifs and Discords. In: 17th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 237–242 (2018)
7. Gharghabi, S., Ding, Y., Yeh, C.C.M., Kamgar, K., Ulanova, L., Keogh, E.: Matrix Profile VIII: Domain Agnostic Online Semantic Segmentation at Superhuman Performance Levels. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 117–126. IEEE (nov 2017)
8. Gharghabi, S., Imani, S., Bagnall, A., Darvishzadeh, A., Keogh, E.: MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios. In: IEEE Int. Conf. on Data Mining (ICDM). pp. 965–970 (2018)
9. Imani, S., Madrid, F., Ding, W., Crouter, S., Keogh, E.: Time Series Snippets: A New Primitive for Time Series Data Mining. In: IEEE Int. Conf. on Big Knowledge (ICBK). pp. 382–389. IEEE (2018)
10. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks. Proc. 8th ACM SIGKDD international conference on Knowledge discovery and data mining p. 102 (2002)

11. Lavin, A., Ahmad, S.: Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark. In: IEEE 14th Int. Conf. on Machine Learning and Applications (ICMLA). pp. 38–44 (dec 2015)
12. Linardi, M., Zhu, Y., Palpanas, T., Keogh, E.: Matrix Profile X. In: Proc. 2018 Int. Conf. on Management of Data (SIGMOD). pp. 1053–1066 (2018)
13. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, B.: Exact Discovery of Time Series Motifs. In: Proc. SIAM International Conference on Data Mining (2009)
14. Mueen, A., Viswanathan, K., Gupta, C., Keogh, E.: The fastest similarity search algorithm for time series subsequences under euclidean distance (2015)
15. Papadimitriou, S., Faloutsos, C.: Streaming Pattern Discovery in Multiple Time-Series. International Conference on Very Large Data Bases (VLDB) pp. 697–708 (2005)
16. Rafiei, D.: On similarity-based queries for time series data. In: Proceedings 15th International Conference on Data Engineering. pp. 410–417. IEEE (1999)
17. Reiss, A., Stricker, D.: Introducing a new benchmarked dataset for activity monitoring. In: International Symposium on Wearable Computers. pp. 108–109 (2012)
18. Vitter, J.S.: Random sampling with a reservoir. ACM Transactions on Mathematical Software (TOMS) **11**(1), 37–57 (1985)
19. Wang, X., Lin, J., Patel, N., Braun, M.: A Self-Learning and Online Algorithm for Time Series Anomaly Detection, with Application in CPU Manufacturing. In: Proc. 25th ACM International on Conference on Information and Knowledge Management - CIKM '16. pp. 1823–1832. ACM Press, New York, New York, USA (2016)
20. Yeh, C.C.M., Kavantzas, N., Keogh, E.: Meaningful Multidimensional Motif Discovery. In: IEEE Int. Conf. on Data Mining (ICDM). pp. 565–574. IEEE (2017)
21. Yeh, C.C.M., Kavantzas, N., Keogh, E.: Using Weakly Labeled Time Series to Predict Outcomes. Proc. of the VLDB Endowment **10**(12), 1802–1812 (2017)
22. Yeh, C.C.M., Van Herle, H., Keogh, E.: The matrix profile allows visualization of salient subsequences in massive time series. Proc. IEEE Int. Conf. on Data Mining, ICDM pp. 579–588 (2017)
23. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A., Keogh, E.: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In: IEEE 16th Int. Conf. on Data Mining (ICDM). pp. 1317–1322 (2016)
24. Zhu, Y., Imamura, M., Nikovski, D., Keogh, E.: Time Series Chains: A New Primitive for Time Series Data Mining. In: IEEE Int. Conf. on Data Mining (ICDM). pp. 695–704 (2017)
25. Zhu, Y., Mueen, A., Keogh, E.: Admissible Time Series Motif Discovery with Missing Data (2018), https://arxiv.org/pdf/1802.05472.pdf
26. Zhu, Y., Yeh, C.C.M., Zimmerman, Z., Kamgar, K., Keogh, E.: SCRIMP++: Time Series Motif Discovery at Interactive Speeds. In: IEEE Int. Conf. on Data Mining (ICDM). pp. 837–846 (2018)
27. Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.C.M., Funning, G., Brisk, P., Keogh, E.: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. 2016 IEEE 16th Int. Conf. on Data Mining (ICDM) pp. 739–748 (2016)