Finding Periodic Discrete Events in Noisy Streams

Abhirup Ghosh, Christopher Lucas, Rik Sarkar
School of Informatics, University of Edinburgh.
abhirup.ghosh@ed.ac.uk, c.lucas@ed.ac.uk, rsarkar@inf.ed.ac.uk

ABSTRACT

Periodic phenomena are ubiquitous, but detecting and predicting periodic events can be difficult in noisy environments. We describe a model of periodic events that covers both idealized and realistic scenarios characterized by multiple kinds of noise. The model incorporates false-positive events and the possibility that the underlying period and phase of the events change over time. We then describe a particle filter that can efficiently and accurately estimate the parameters of the process generating periodic events intermingled with independent noise events. The system has a small memory footprint, and, unlike alternative methods, its computational complexity is constant in the number of events that have been observed. As a result, it can be applied in low-resource settings that require real-time performance over long periods of time. In experiments on real and simulated data we find that it outperforms existing methods in accuracy and can track changes in periodicity and other characteristics in dynamic event streams.

CCS CONCEPTS

•**Information systems** → **Data stream mining;**

KEYWORDS

Periodicity, Particle Filter, Temporal sequence mining

ACM Reference format:
Abhirup Ghosh, Christopher Lucas, Rik Sarkar. 2017. Finding Periodic Discrete Events in Noisy Streams. In *Proceedings of CIKM'17 , Singapore, Singapore, November 6–10, 2017,* 10 pages.
DOI: 10.1145/3132847.3132981

1 INTRODUCTION

We will study discrete event streams that exhibit *approximate* periodicity. These sequences contain noise events interleaved with the periodic events, together with variations in the times that events are observed, and variations in the period itself.

The presence of noise events, or false positives, is common in sensor data and other data streams that are subject to inaccuracies and calibration issues. An example is shown in Figure 1, where we use the output of an accelerometer on a phone to detect footsteps as a person walks. The steps are indicated by large spikes in acceleration, but for any threshold that reliably detects actual steps, extraneous spikes in acceleration create false detections – or noise events. The challenge here is to detect the periodic events against the backdrop of these false positives.

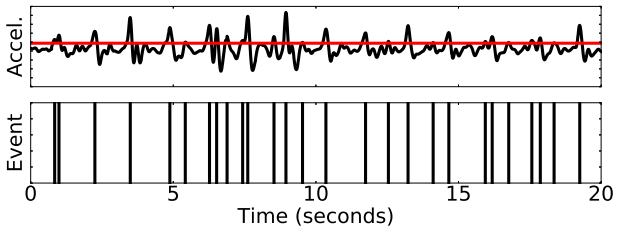


Figure 1: Noise events in footstep detection. Top plot: Accelerometer signal (Euclidean norm of acceleration given by 3-axis accelerometer on mobile phone), with event detection threshold in red. Bottom plot: Detected events. Some of the detections are noise events, produced by inter-footstep accelerations that exceeded the threshold.

Another common property of approximately periodic sequences is that even for events originating in the periodic process, the event times can be subject to noise or drift. While footsteps are periodic, they do not follow a perfect lock-step pattern – a person may pause, speed up, or slow down. These temporal offsets can accumulate, causing the periodic signal to depart from its original phase and/or period. Figure 2 shows an example of this behavior. A change in phase can throw off classical methods such as Fourier transform, which has been used to detect periodicity in discrete event streams in previous works [12].

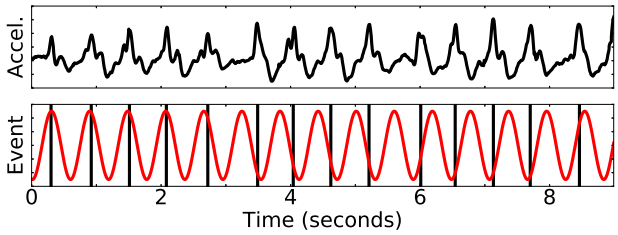


Figure 2: Top plot: Acceleration signal during walking. Bottom plot: Black lines denote local maxima in acceleration above a threshold, corresponding to individual footsteps; red line shows a sinusoid that is matched to the initial period of footsteps, demonstrating phase drift (or noise in inter-event delays) over time.

We thus need methods to find noisy patterns that are functions of time itself from an event stream. Some methods have been developed in the past to solve the problem of detecting discrete periodic events (e.g., [11, 18, 19]) but these methods cannot handle

CIKM'17 , Singapore, Singapore

DOI: 10.1145/3132847.3132981

sequences with variability in inter-event time and consequent phase changes. We will discuss flexible designs to accommodate this variability and as well as noise events, while being computationally efficient for long streams of events.

Motivations and applications. There are several scientific and technological applications for systems that can discover long term patterns and periodicities. With the emergence of low-cost sensor technology, ubiquitous and mobile sensing has become common for health care [17], personalized services [14, 16, 22] and other applications. Detection of regular, periodic behavior plays a role in customizing services to individual users based on their schedules. Steps or gait data is valuable in monitoring for neuromotor diseases like Parkinson's [3, 15] that are characterized by irregular gait.

In systems and engineering domains, e.g., datacenter networks, periodicity can occur at smaller time scales [4] of microseconds to milliseconds, and is relevant in detecting DoS attacks [6]. Complex networks are known to generate periodic patterns, such as emergent traffic behavior in [12], or periodic surge of infectious diseases in social networks [8].

Our contribution. We define a model of discrete signals with approximately periodic sequences embedded in them. The model is described in Section 2, and includes a periodic component with probabilistic inter-event time, and a Poisson process modeling the aperiodic false positive noise events. It generalizes previous systems [10, 11].

Section 3 describes the design of a particle filter system based on this model, which can identify the periodic events from the sequence in an online fashion. In addition to predicting events, it infers parameters of the underlying process such as the local period, the rate at which the period changes, and the rate at which noise events tend to occur. The particle filter system works with a collection of *particles* or *hypotheses*. Each hypothesis includes guesses about the parameters of the model, such as the period. With each new event, the system evaluates the current set of particles and assigns a weight to each depending on how well it has predicted the latest events. Based on these weights, the model occasionally rejuvenates particles to retain a diverse and plausible set of hypotheses and improve long-run accuracy.

To handle noise events, each hypothesis in the particle filter identifies the most recently observed event as either a periodic event or a noise event, in addition to tracking the overall rate of noise events. In systems where almost all events are due to noise, identification of periodic events is costly and error-prone, while the predictability of periodic events loses much of its utility as noise events overwhelm periodic events. In order to achieve high performance and a low memory footprint in contexts where periodic event detection is possible and useful, our particle filter incorporates the assumption that tractable noise rates are more likely.

The system works in an online or incremental manner, in that it only requires knowledge of events up to a point in time, without the need to know the entire event stream. Thus it can be used to track events in real time – even as the period changes. We show it can operate accurately with relatively few particles and thus can be used at a low computational cost to analyze large volumes of data in a single pass. The particle filter processes events as they occur, and incurs no cost for the quiet periods between them. Thus the cost of the method depends only on the number of events, and is independent of the total duration of the event sequence or the sampling rate.

Experiments in Section 4 show that the system can accurately estimate the period of an event-generating process on both synthetic and real-world data. Existing methods are not designed to handle shifts in phase (Figure 2) and noise, as a result, they perform relatively poorly in these cases. Each hypothesis includes expectations about when future periodic events will occur, so the system can be used to predict upcoming events. Our experiments show that these predictions are accurate and robust to noise and changing periods. Moreover, the experiments show that our system works well with sparse data, converging to accurate estimates after observing only a few events. Section 5 discusses previous relevant works.

2 MODEL AND PROBLEM DESCRIPTION

In this section, we will develop a model that allows noise in periodic event sequence – both in the period itself and in occurrence or observation of individual events. Then we discuss how this model relates to different real scenarios.

Periodic events often arise because some recurring process takes a roughly uniform amount of time to finish and yield an observable event. Examples include heartbeats, volcanic geysers or steps taken by a walking person. In these systems an event may sometimes be delayed due to irregularities of the intervening process, resulting in a shift of phase as shown in Figure 2.

If such a system is perfectly regular, then every event $i+1$ occurs at time x_{i+1}, exactly T time units after event i, that is, $x_{i+1} = x_i + T$. The effect of the irregularity is then modeled using a Gaussian of variance σ^2 and zero mean added to T, giving us $x_{i+1} \sim \mathcal{N}(x_i + T, \sigma^2)$. Here T is the fundamental *period* or inter-event interval parameter, while σ can be seen as the "noisiness" of the period over time.

Other than the intrinsic irregularity of the process, the data can reflect noise events. Noise events may either come from the observation mechanism affecting both record of true events as we have seen in Figure 1, or it may represent a different or spurious source of events.

We model this noise sequence as a Poisson process, i.e., a sequence of events $z_1, z_2, \ldots$ where $z_{i+1} = z_i + \delta$, and δ follows an exponential distribution with rate parameter λ; a higher lambda implies a greater rate, or shorter expected durations between false positive events.

Thus in our model, the overall sequence $y = y_1, y_2, \ldots$ can be partitioned into two subsequences[1]:

- A sequence x of periodic events, where $x_{i+1} \sim \mathcal{N}(x_i + T, \sigma^2)$.
- A sequence z of noise events given by $z_{i+1} = z_i + \delta$, where δ follows an exponential distribution with an expectation of $(\lambda)^{-1}$, for a rate parameter $\lambda > 0$.

Problem statement. Given a sequence of event times $y_{1:n} = y_1, \ldots, y_n$, we wish to infer the period T, which is the fundamental parameter that describes the behavior of the system. We would

[1] A subsequence of a sequence is a subset of the events, not necessarily contiguous, in the same order as in the original sequence.

also like to know the parameters σ and λ that determine the extent and nature of the noise in the system.

As an additional feature, we want the algorithm to be *online*, meaning that it processes the sequence one item at a time, updating its estimates with each input without requiring that the system store or process the full event history. The ideal algorithm will make inferences with $O(1)$ time and memory complexity after each event, and can thus remain responsive even after processing arbitrarily large numbers of events.

Generality of the model and local fit with input. This model incorporates quite general scenarios. It does not make any assumption about there being a set of "true" signal events to detect as opposed to observational noise. It assumes that all events may be generated by the same underlying process, and aims to detect the presence of a periodic subsequence. The Gaussian distribution of inter-event time represents a sum of unknown variables that may cause the local variations in the period.

As shown in Figure 1, the threshold for an "event" determines the noise rate. It is possible to set a high threshold to ensure very few noise events, at the risk of missing some fraction of periodic events. However, lower thresholds are more desirable – they may include more noise, but will not miss the periodic events. Our model is designed with this setup in mind – that periodic event information is preserved, even at the cost of more noise events. The parameter λ as part of the hypothesis learns this noise rate. Poisson processes are commonly used for such models, as they only assume that the noise occurs at a certain uniform rate in any temporal locality.

Note that the system operates in terms of a set of evolving hypothesis and can adapt to changing model parameters online as the system behavior changes. Thus, it is not required that the model fits the global data over long periods of time, only that it fits the system approximately, over any short period of time. Experiments show that in fact this model successfully adapts to periodic events and tracks them closely with changes in the system. It's performance remains consistent when the noise and inter-even variability are generated using distributions other than the ones assumed in the model.

Discussion and variations of the model. There are various common scenarios where periodic events are generated by processes that are special cases or variations on the model above. The simplest is what we will call *strict periodicity*, where successive events are separated by an exact time interval T, such as clock ticks, or events tied to clock-like sequences. Formally, $x_i = \phi + iT$, if all events are due to a strictly periodic process. We get this behavior in our model above by setting $\sigma = \lambda = 0$.

Strict periodicities are easy to capture, by simply observing the interval between successive events. A variant of this is *partial periodicity*, where a strict periodic function is mixed with noise. This was considered in [10].

Certain sequences like heartbeats or volcanic geysers satisfy $\lambda = 0; \sigma > 0$ – they are not tied to a clock, but depend on a build up in the intervening process that can have some variability, and change in phase or even dynamic change in the period itself.

There are systems where noise in the periodicity exists, that is $\sigma > 0$, but phase does not not change. Example would be daily activities of a person – such as checking the news in the morning – which may not have a strict inter-event time, but still around a particular time in the morning. These can be modeled by adding *observational noise* to strict periodicity: $x_i \sim \mathcal{N}(\phi + iT, \hat{\sigma}^2)$. The variation in inter-event time does not affect the phase ϕ here. This type of sequences have been addressed in a recent work [18]. The approach there is to consider the histogram of events modulo a number W, which produces a pronounced peak at the correct value of $W = T$. We discuss this and other related methods in Section 5.

3 PARTICLE FILTER DESIGN

In this section, we describe a particle filter based system to detect periodicity in noisy and approximately periodic event sequences. The next subsection briefly summarizes particle filters, which form the foundation of our approach, and lists our notations. Following this, we describe the details of our design to adapt particle filters to the current problem.

3.1 Particle filters and importance sampling

Particle filters, or sequential Monte Carlo methods, are a popular family of online methods for making inferences about latent variables in noisy environments. They have been applied to diverse problems, including locating and tracking objects using sensor data (e.g., [23]), machine vision (e.g., [20]), and natural language processing (e.g., [5]).

The essential idea is that each *particle* is a point sample in the space of possibilities of all the parameters in question. We interchangeably use the term *hypothesis* to mean that each particle is one of our guesses of the true configuration of the system.

Symbol	Description
$f(x; \mathcal{D})$	Probability of random variable $R \sim \mathcal{D}$ taking value in small neighborhood of x.
$F(x; \mathcal{D})$	Probability of random variable $R \sim \mathcal{D}$ taking value $\leq x$.
$\exp(\lambda)$	Exponential distribution with rate λ.
y_i	i^{th} Event timestamp.
$h_i^{(j)}$	j^{th} hypothesis (also called as particle) after observing i^{th} event timestamp.
$\mathcal{L}_i^{(j)}$	Likelihood weight of $h_i^{(j)}$.
	Hypothesis parameters of $h_i^{(j)}$
$T_i^{(j)}$	Period parameter value.
$\sigma_i^{(j)}$	Standard deviation for Gaussian distribution of period variability.
$\lambda_i^{(j)}$	Rate of false positive noise events.
$\hat{z}_i^{(j)}$	Latest event timestamp marked as noise event
$\hat{x}_i^{(j)}$	Latest event timestamp marked as periodic event

Table 1: Definition of useful symbols

Our goal is to determine the period T of an approximately periodic sequence. In probabilistic terms, we want to obtain a posterior distribution over possible values of T_n: the period after the n^{th} event, given a vector of observed event timestamps $y_{1:n} = \{y_1, ..., y_n\}$ up to that point. The period is not the only salient feature of the

event-generating process, so we will use h_n to denote the ensemble of features, which includes the period, variance σ^2, rate of noise events λ and whether an event is periodic (from the subsequence x) or is noise (from subsequence z).

With a posterior distribution over h_n, we can draw probabilistic conclusions about any of these variables. The probability of h_n cannot be computed exactly because it requires solving intractable integrals, but we can efficiently approximate its distribution in constant time per incoming event, by using a simple recursive Bayesian algorithm known as a *bootstrap filter* or *sequential importance sampler*. This idea is closely related to conditional density estimation in machine vision. We refer the reader to [7, 24] for a detailed explanation, restricting our attention to a brief summary in Algorithm 1, along with the distributional assumptions that allow us to apply it to the current problem.

In this algorithm, we need to keep track of our estimates of various parameters after each event. Thus we use subscript i to represent value of variables after event i. Further, each hypothesis or particle has its own estimate of variables, thus we use superscript (j) to represent the estimate of the variables in particle (j). Table 1 summarizes all the symbols used in this section, while Algorithm 1 presents the basic operation of a bootstrap filter.

Algorithm 1: Sequential Monte Carlo with resampling

Particle filter *(Event timestamps $y_{1:n}$)*
begin
Initialize k hypotheses $h_0^{(1)}, \ldots, h_0^{(k)}$ by sampling from a prior distribution over hypothesis parameters.
for *(time step i in $1, \ldots, n$)* **do**
for *(particle j in $1, \ldots, k$)* **do**
sample $h_i^{(j)}$ from $p(h_i|h_{i-1}^{(j)})$
Likelihood weight $\mathcal{L}_i^{(j)} = p(y_i|h_i^{(j)})$
end
Normalize likelihood weights: $\tilde{\mathcal{L}}_i^{(j)} = \frac{\mathcal{L}_i^{(j)}}{\sum_{j=1}^{k} \mathcal{L}_i^{(j)}}$
Resample k hypotheses with replacement according to normalized weights $\tilde{\mathcal{L}}_i^{(j)}$.
end
end

Algorithm 1 processes each event as follows. It initializes several hypotheses from the distribution over features for each particle. Then at every event, it checks how likely the new event is for each of these hypotheses, and uses that as a "score", $\mathcal{L}^{(j)}$, for each particle. Next, it normalizes the scores to turn them into probabilities, and resamples the set of hypotheses according to these to get the set of particles for the next round, effectively favoring neighborhoods of particles that have matched better with recent events. At any time, the estimate for a parameter is the expectation of the parameter over all particles weighted by their current scores.

3.2 Particle filter for discovering periodicity

A hypothesis h_n contains the estimated characteristics of the periodic signal after n events. The periodic characteristics are specified using the period T_n, and standard deviation σ_n for Gaussian distribution of period variability. Our hypothesis h_n also includes the last periodic event $\hat{x}_n$ in the sequence. We have included the rate of background noise events λ_n, and the timestamp of the last event that was attributed to noise, $\hat{z}_n$ in our definition of h_n. We will use i to index event timestamp, y_i, where $1 \le i \le n$.

Under this model, each event y_i originates from either the periodic signal or false positive noise. We introduce $r_i \in \{0, 1\}$ to track event provenance, where $r_i = 1$ represents the case where the i^{th} event comes from the periodic process. This auxiliary variable facilitates efficient inference: using r_i, we obtain a closed-form solution for the density of periodic events conditional on the last periodic event, and noise density conditional on the last event. Calculating $P(r_i|y_{1:i}, h_{0:i-1}, r_{1:i-1})$ exactly is expensive, as the number of states involved increases exponentially as we observe more events, but we can augment our sampling procedure to reflect the posterior distribution of r_i. For simplicity we omit (j) superscripts where they are clear from context.

The bootstrap particle filter algorithm requires us to specify three probability distributions:

- A prior over hypotheses for initialization of $h_0^{(j)}$.
- A likelihood function adjusting the weight of a hypothesis given observed events.
- A distribution defining how hypotheses change between events.

We specify the distributions as follows:

Priors. For each hypothesis $h^{(j)}$ we sample the initial period $T_0^{(j)}$ and noise event rate $\lambda_0^{(j)}$ from exponential distributions, and sample the standard deviation $\sigma_0^{(j)}$ uniformly from the interval $[0, T_0^{(j)}]$. At the initialization stage, we know nothing about the periodicity value or noise event rate other than that they are positive, thus we use exponential distribution as a maximum entropy guess for the distribution. Both $\hat{x}_0^{(j)}$ and $\hat{z}_0^{(j)}$ are assumed to start at zero.

Likelihood weighting. Given a hypothesis h_i, the likelihood of event i occurring at time y_i is equal to $\mathcal{L} = \sum_{r_i \in 0,1} p(y_i, r_i|h_i)$. The periodic signal's contribution to this sum is $\mathcal{L}_{per} = p(y_i, r_i = 1|h_i)$, and is the product of two terms:

- Density function from the previous periodic event, assuming that $\hat{x}_{i-1}$ is known. Since the time of a periodic event is subject to additive Gaussian noise, we represent this density as

$$p(y_i|h_i^{(j)}) = \mathcal{N}(T_i^{(j)} + \hat{x}_{i-1}^{(j)}, \sigma_i^{(j)}). \tag{1}$$

- A term based on the observation that $r_i = 0$ if and only if there are zero noise events in the time between $\hat{z}_{i-1}$ and y_i, inclusive. If a Poisson process generates noise events, then that probability is $e^{-\lambda_i(y_i - \hat{z}_{i-1})}$.

The false positive noise's contribution to the total likelihood is $\mathcal{L}_{bg} = p(y_i, r_i = 0|h_i)$. Here, we have a product of two terms:

- An exponential distribution, with its origin at $\hat{z}_i$ and a rate equal to the noise rate.
- The cumulative probability that no periodic event occurred between $\hat{x}_{i-1}$ and y_i. We approximate this probability by

the complementary cumulative density of periodic events between y_i and $\hat{x}_{i-1}$ by evaluating the complementary cumulative density function for the distribution in Equation 1.

A pseudo code of of the likelihood weighting is presented in Algorithm 2. The components of the likelihood function are illustrated in Figure 3.

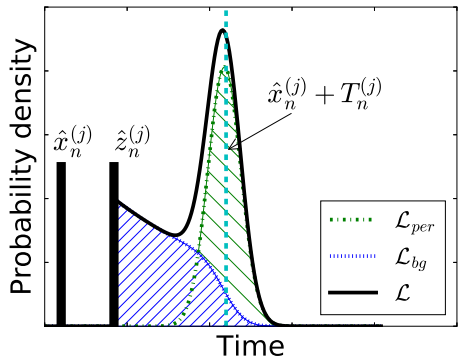


Figure 3: Periodic and false positive noise's contribution to the likelihood of hypothesis j. The periodic contribution $\mathcal{L}_{per}$ is centered around $T_n + \hat{x}_n$ where $\hat{x}_n$ denotes the last periodic event and the noise contribution $\mathcal{L}_{bg}$ starts from the last noise event $\hat{z}_n$.

Algorithm 2: Likelihood Weighting of j^{th} hypothesis $h_i^{(j)}$ on observing i^{th} event timestamp y_i.

Likelihood Weight $(h_i^{(j)}, y_i)$
begin
$\mathcal{L}_{per} = f\left(y_i; \mathcal{N}(\hat{x}_{i-1} + T_i, \sigma_i)\right)$
$\times\left(1 - F\left(y_i - \hat{z}_{i-1}; \exp(\lambda_i)\right)\right)$
$\mathcal{L}_{bg} = f\left(y_i - \hat{z}_{i-1}; \exp(\lambda_i)\right)$
$\times\left(1 - F\left(y_i; \mathcal{N}(\hat{x}_{i-1} + T_i, \sigma_i)\right)\right)$
$\mathcal{L} = \mathcal{L}_{per} + \mathcal{L}_{bg}$
return $\mathcal{L}$
end

Hypothesis updates. With each iteration, the hypothesis distribution is updated according to the prior over state changes. We obtain $h_i^{(j)}$ by sampling from $p(h_i|h_{i-1}^{(j)})$, where $p(h_i|h_{i-1}^{(j)})$ implements for each parameter in $h_i^{(j)}$, a sampling from a Cauchy distribution centered at the previous value of the parameter. For example, $T_i \sim \text{Cauchy}(T_{i-1}, b)$ where b is a tunable scale parameter. The period (T), periodic deviation (σ), and noise rate (λ) (truncated at zero) are updated according to the said Cauchy distribution.

Note that these updates make it possible to iteratively refine parameter estimates, in addition to tracking the dynamics of the system, e.g., when the period or phase changes over successive events.

The $\hat{z}_{i-1}$ and $\hat{x}_{i-1}$ terms are updated differently depending on whether the provenance of the latest event is the periodic or noise process. Given that y_i is observed, we can use $p(y_i, r_i|h_i)$ to obtain a conditional distribution for r_i: $p(r_i = 1|y_i, h_i) = \frac{p(y_i, r_i=1|h_i)}{p(y_i, r_i=1|h_i)+p(y_i, r_i=0|h_i)}$. We can then sample r_i from this Bernoulli distribution, and by extension $\hat{x}_{i-1}$ and $\hat{z}_{i-1}$: if $r_i = 1$ then, $\hat{x}_i = y_i$ and $\hat{z}_i = \hat{z}_{i-1}$. Otherwise, $\hat{x}_i = \hat{x}_{i-1}$ and $\hat{z}_i = y_i$.

Output: period estimate. After seeing n events, the expected period of the event sequence is $\mathbb{E}[T_n] \approx \sum_{i=1}^{k} \tilde{\mathcal{L}}_n^{(j)} T^{(j)}$, and represents an estimate of the period. More generally, we can take the expectation of arbitrary functions $f(.)$ of hypotheses by replacing $T_n^{(j)}$ with $f(h_n^{(j)})$, to obtain estimates of other parameters such as noise rate and period variability. In our experiments, we use the weighted median as a more robust estimate of parameters.

3.3 Practical optimizations for noisy environments

The algorithm presented in the previous section can detect the period of a noisy sequence of events, but can require a large number of hypotheses to do so reliably; the particle filter needs a considerable number of hypotheses to ensure that several occupy high-probability regions of the parameter space. If we use a small number of hypotheses, then the particle filter will occasionally provide poor or high-variance estimates, especially when the period or dynamics are extreme or a priori unlikely. In this section, we present a modification our basic algorithm that preserves accuracy while using fewer hypotheses.

A pragmatic observation from an end-user's point of view is that if the number of noise events in a sequence is much greater than the number of periodic events, then the period value of the signal is not useful for predicting future events, and periodic event detection may become infeasible. Motivated by this observation, we can modify the likelihood weighting strategy for the previous algorithm to capture the assumption that we are facing an estimation problem for which there exists a *useful* solution. In so doing, we improve our ability to obtain useful solutions if they exist, and incur negligible costs if they do not.

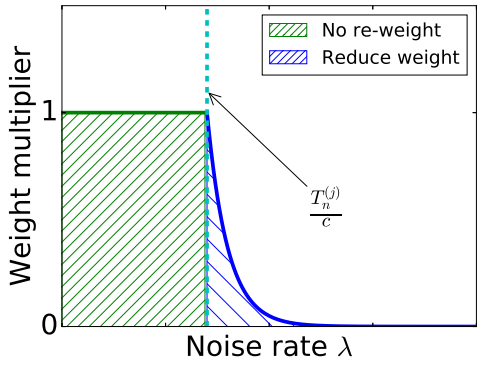


Figure 4: Re-weighting function.

We fix an application specific input parameter c to our algorithm, expressing a ratio of the number of noise events to the number of periodic or regular events, above which useful inferences are likely

to be elusive. Parameter c can be tuned for specific applications, but in practice the value 2 works well across the settings for all of our experiments. We assign lower probabilities to hypotheses that have higher noise event rates than c, reducing the weights of the hypotheses for which $\lambda_i^{(j)} \times T_i^{(j)} < c$ according to an exponential decay function. The start location of the exponential decay function is set to be $\frac{c}{T_i^{(j)}}$, and the rate depends on the importance of the assumption of limiting noise rate. We multiply the likelihood of a hypothesis derived as described in the previous section by a value drawn from a function as represented in Figure 4. With this strategy, we ensure low weights for the hypotheses which have large periods and high noise event rates that would lead to poor predictive accuracy.

4 EXPERIMENTAL EVALUATION

We tested the performance of our particle filter and compared it with existing ideas on real human gait data and human pulse data. We also created several sets of artificial data with different values of intrinsic variability, noise event rate, and period to precisely characterize the detection accuracy. Overall, we found that:

- In presence of noise events and phase drifts the particle filter method outperforms other techniques in accuracy.
- With higher noise event rates and phase drift, only the particle filter produces reasonably accurate and reliable results.
- The particle filter method can operate with a reasonable and constant number of about 256 particles to give good results.
- The particle filter runs more efficiently than competing methods, and thus is suitable for operation in an online environment to produce increasingly accurate results.
- On data with rapidly changing periodicity, the method can track the period closely. We show this by applying the technique to human step data to determine changing gait rate.
- The method applied to real data such as human pulse and gait data accurately detects periodicity.

The following subsections first describe the competing techniques for periodicity detection, and then the detailed experimental results.

4.1 Comparison algorithms

The following popular methods for detecting periodicity are compared with the particle filter algorithm.

Fourier Transform: The frequency with highest spectral energy in Fast Fourier Transform algorithm is a reasonable candidate for the true frequency of the system. This technique has been used in the past as a practical method for detecting periodicity [12].

Autocorrelation: The cross-correlation of the input signal with the signal itself with different amounts of offsets, thus building the autocorrelogram can be used to find period of a signal. The delay of the signal where the cross-correlation value reaches its maximum value can then be treated as the period of the signal [25]. A linear search on period values is required to find this maximum.

Segmentation based algorithm: The method described in [18] detects periodicity in face of approximate periodicity and noise, as described in Section 2. This algorithm operates by checking all possible periods by computing the histogram of events modulo the period, and looking for a "peak" in this histogram. This checking of all possible periods makes it inefficient on large datasets.

4.2 Sensitivity to noise and model parameters

To gain understanding of the basic operation of the system in noisy environments with variability in period and presence of noise events requires data with precisely known values of these parameters. We thus simulated artificial data based on the models described in Section 2.

4.2.1 Generation of synthetic data. With fixed values for parameters: period T, deviation of periodic events σ, and noise event rate λ, we generated two separate sequences: the periodic events and the noise events, and merged them in sorted order.

Periodic event sequence (x). The first periodic event x_0 is set to 0. Each subsequent periodic event i is designated to occur at time x_i given by $x_i = x_{i-1} + \delta_{per}$. The distance between two consecutive periodic events, δ_{per} is drawn form the Gaussian distribution $\mathcal{N}(T, \sigma^2)$. While drawing the samples δ_{per}, we make sure that the $(i-1)^{\text{th}}$ periodic event always precedes i^{th} periodic event, i.e. $x_{i-1} < x_i, i > 0$, by re-sampling δ_{per} from the Gaussian distribution.

False positive noise event sequence (z). Similar to periodic event sequence, the first noise event, z_0 is assumed to occur at time 0. Each subsequent noise event time z_i is constructed using recursive function $z_i = z_{i-1} + \delta_{bg}$. The inter-event distance between two consecutive noise events, δ_{bg} is drawn from an exponential distribution with rate λ.

The final event sequence y is a merge of the two event sequences in sorted order.

4.2.2 Performance evaluation of accuracy. The evaluations are based on the signal characteristic parameters: period T, the relative variance of periods ($\frac{\sigma}{T}$), and ratio of the frequency of noise events to frequency of periodic events in the signal.

For a signal with period T, let us denote the period predicted by an algorithm in consideration by $\hat{T}$. We define the relative error ξ as:

$$\xi = \left| \ln \frac{\hat{T}}{T} \right| \tag{2}$$

This measure is symmetric with respect to multiples of T. That is, an estimate of αT is considered equally erroneous as an estimate T/α.

For all the experiments, unless mentioned otherwise, we use a signal with 40 periodic events; the total number of observed events varies according to the noise event rate. In all experiments, the particle filter assumes that the rate of noise events rarely reaches twice the rate of periodic events. This assumption is not critical in practice as we show in our results in Figure 6(b). All experiments are repeated 16 times with independently generated synthetic data.

Effect of number of particles. Particle filters involve a tradeoff between efficiency and accuracy. Increasing the number of particles

increases accuracy, but the computational cost increases linearly with number of particles. As expected, larger particle counts lead to lower average error (Figure 5). The error decreases steadily with increasing number of particles and we find that the error is quite low at about 256 particles, beyond which the gains are small. Thus, in the rest of the experiments, we use 256 particles. Note that computational cost for using 256 particles is nominal in comparison to other algorithms as discussed later in this section.

All the box plots in this paper have bottom and top boundaries as 0.25 (Q_1) and 0.75 (Q_3) quartiles of the relative error ξ. The middle line in the boxes represent the median values (Q_2) of the relative error. The whiskers are represented as 0.15 times the inter quantile range, e.g. $Q_3 + 1.5 \times (Q_3 - Q_1)$. Values appearing outside these ranges are considered as outliers and represented as individual points.

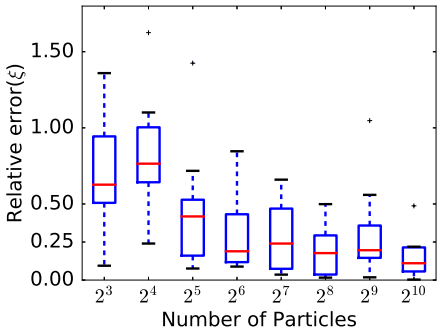


Figure 5: Error in period detection decreases with number of particles. 256 particles is a reasonable tradeoff for practical use where error is low and number of particles is small. Signal parameters: period T=10, relative deviation ($\frac{\sigma}{T}$) = 0.6, and average number of noise events per periodic event = 1.5.

Effect of relative deviation of periodic events ($\frac{\sigma}{T}$). The deviation that the periodic signal can have, given by the standard deviation σ of the normal distribution, determines how fast the phase can drift away from the original configuration. Figure 6(a) shows that as σ increases relative to T, other methods perform poorly for even small values of σ, and do increasingly worse. The particle filter achieves reliable results even at high variance. This is in part due to the fact that unlike other methods, the particle filter is designed to be adaptive to change in phase and period and does not attempt to fit a single global model, including phase, to the entire signal.

Effect of noise events. Figure 6(b) shows that the particle filter based method has the lowest period detection error among all other methods for all noise event rates starting from no noise event up to a rate of 2.5 times periodic events. In this case, other methods perform well for few noise events, but with increase of noise events, their relative errors rise rapidly. Note that our parameters are set to assume that noise rates of over twice the periodic event rates are unlikely. However, at a rate of 2.5 times the errors for particle filter are still lower than other methods.

Effect of period (T) The period T itself is not expected to have any significant effect on the relative error. Our experiments show in Figure 6(c) that while all methods have a general increase in variance, particle filter is by far the most accurate across the spectrum.

Effect of number of observations. Figure 7(a) shows that particle filter based method converges with minimum number of observations in comparison to other methods. In this experiment, Fourier transform and autocorrelation algorithm never converge to low error. The fast convergence of particle filter implies that it can work with smaller quantities of data and does not require long observation sequences.

Tracking rapidly changing periods. In this experiment, we fix the noise parameters while changing the periodicity of the signal. In Figure 7(b), we observe that the period estimate of the particle filter closely follows the true period changing curve, whereas other algorithms perform badly with changes in the period value. Particle filter algorithm also converges quickly to the true period after the true period changes which enables it to catch rapid dynamics of a system. We note that similar behavior is noticed in our experiments with human steps as walking speed changes in section 4.3.1.

We also tested the system on data generated under different models – with different distributions (normal, Laplace, gamma, uniform etc with different parameters) for noise rate and inter-event time, and found that the system detects periodicity accurately. As explained in Section 2, this is due to the fact that for the sake of adaptability, the system prioritises local features over global ones, and thus is not greatly affected by differences in global models. We omit the detailed plots here.

4.2.3 Execution time comparison. We measured the time taken to run the particle filter on datasets of various sizes, and found that computing based on 256 particles is quite efficient and takes less time to process a large event stream compared to Fourier Transform algorithm and other algorithms.

The particle filter operates with constant amount of computation per event, thus its execution time increases linearly with the number of events, while Fast Fourier Transform takes $O(n \log n)$ time to process a signal of length n. Thus, the FFT execution time increases at a superlinear rate. Figure 7(c) shows that as expected, the particle filter requires less time to process large event stream compared to Fast Fourier Transform. The Segmentation based algorithm and autocorrelation check for all possible period values and require substantially larger execution times, and are omitted in this plot. The plot is based on a period value to 10, relative deviation of periodic events to 0.2, and relative amount of noise events to the periodic events to 0.05. We used particle filter based method implemented using Java 1.7 in Linux and Fast Fourier Transform functionality available in Apache math library [1]. We run these two algorithms in a typical desktop machine with 8GB primary memory and Intel *i*5 processor.

One of the advantages of the particle filter is that it processes events only as they happen, and do not perform computations in their absence. In contrast, the complexity of methods like Fourier

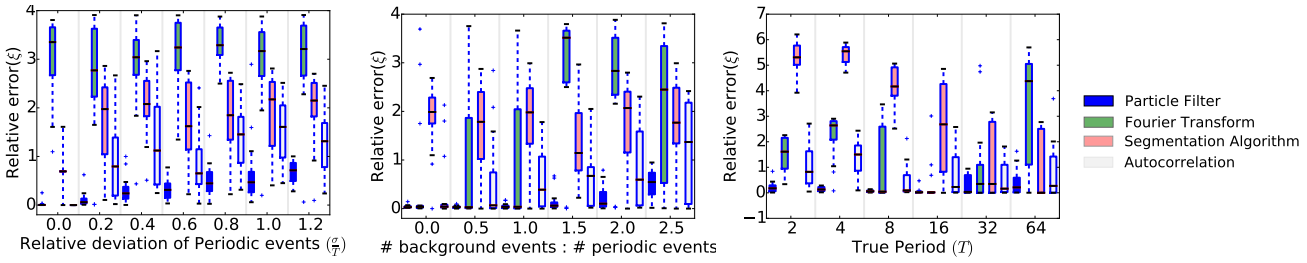


Figure 6: Base signal parameters: period T=10. Average number of noise events per periodic event = 0.05. Relative deviation of periodic events 0.1. (a) Relative error in period detection for different deviations of periodic events (σ/T). Noise rate is low to focus on the effect of σ. Even for large σ, particle filter shows small error and high reliability. (b) Period detection error with increasing rate of noise events relative to periodic events. Particle filter performs significantly better for reasonably large error sizes. (c) Particle filter has lowest period detection error over varying periods.

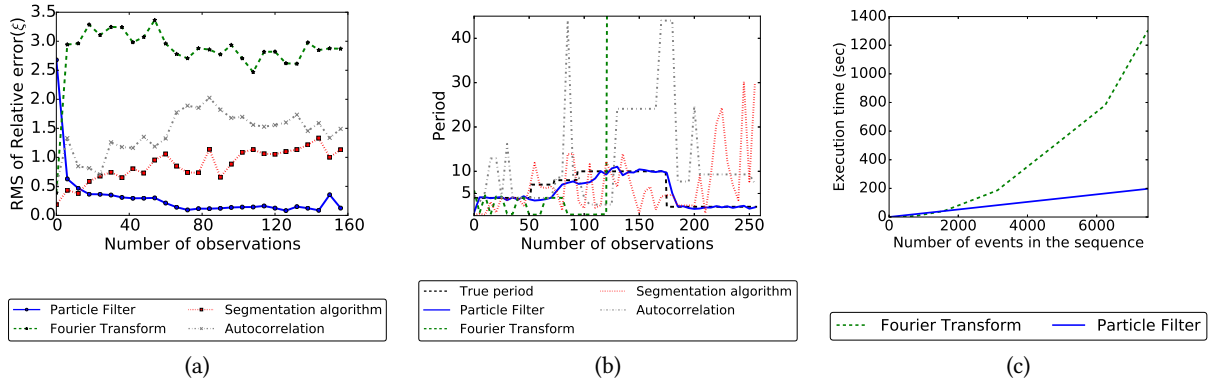


Figure 7: Base signal parameters for (a) and (b): relative deviation of periodic events = 0.2, and relative amount of noise events to the periodic events = 0.01 (a) Periodicity of the signal is set to 10.0. Experiment plots RMS error from 16 iterations. Particle Filter based method converges faster than other algorithms. (b) Particle filter based method successfully tracks changing periodicity of the signal. Period estimate for Fourier Transform algorithm is not shown completely as it is too large beyond certain point. (c) Particle filter based method needs less time to process large event stream compared to Fourier Transform. (c) Particle filter based method execution time increases linearly with number of events.

transform and autocorrelation scale with the duration of the sequence, as opposed to number of events. Thus in systems with sparse events, or with very high sampling rates, these methods incur high cost for the quiet regions, while the cost of particle filters are independent of these parameters and depend only on the number of events. We believe that the particle filter can be made more efficient than this prototype implementation with suitable code optimizations.

4.3 Experiments on real datasets

We found that the particle filter can closely track changing periodicity in human gait data in comparison to other algorithms and accurately predict next human pulse event with low error.

4.3.1 Periodicity in human step data . We compared the performance of the particle filter based algorithm with other algorithms described in subsection 4.1 for tracking changing periodicity in human steps. We collected accelerometer signals using mobile phones while users walk at their own pace, varying their walking speed between walking segments. An accelerometer measures acceleration of a user giving a three dimensional signal S^x, S^y, and S^z. The Euclidean norm represents the magnitude of the signal at time t: $S_t = \sqrt{(S_t^x)^2 + (S_t^y)^2 + (S_t^z)^2}$, followed by smoothing using a Gaussian filter and normalizing around its mean. All peaks in the normalized signal above a predefined threshold are considered as discrete step events. Taking different values for threshold results in different rates of noise events as we have seen in Figure 1. We calculated the ground truth of step periodicity as the average time taken per step over a segment.

Tracking a single walking trace. Figure 8(a) shows that the estimated period from the particle filter closely tracks the changing curve of ground truth step periodicity as it changes between segments. Here we used the threshold as the half of the standard deviation of the smoothed magnitude signal to detect step events.

Figure 8(b) shows that due to its local adaptive nature, the particle filter has the lowest periodicity estimation error among all algorithms. Figure 8(c) shows the CDF of error over twelve traces collected from six different people.

The particle filter method is robust to different choices of thresholds for extracting step events from the accelerometer signal. This is shown in Figure 9 for different choices of threshold.

4.3.2 Pulse prediction. The particle filter can reliably predict upcoming events in a noisy sequence. For this experiment we collected videos of users' fingertips, and used the red color component as a signal for pulse events. We applied similar preprocessing on the pulse signal as we did for accelerometer signal.

The estimate of period by the particle filter gives an estimate of the next event time. We measured the next pulse event prediction error as the absolute difference in predicted and actual event times. Figure 10 shows that the particle filter quickly converges to reliably predict the pulse event and consistently performs better than other algorithms. The larger errors (e.g. at around 55 second mark) represent sudden changes in heart rate due to change in user's activity.

5 DISCUSSION OF RELATED WORK

Time series analysis has traditionally considered real-valued functions. Fourier transform is a natural tool for detecting periodic patterns in real-valued signals, and has also been applied to detect periodicity in binary event streams [12]. However, as observed in [18], it does not perform well in data with noise, missing signals and other errors. It also does not lend itself to online inference, which is one of our main desiderata. Further, Fourier transform aims to fit a global phase and periodicity to the data. In our model, where phase can drift, the signal can easily go out of phase with the basic sine wave (Figure 2).

A *sketch* based approach in [11] uses random linear projection over windows of time series data to compress them for easier comparison. This makes comparison between windows easier, but does not easily extend to detecting periodic events at arbitrarily large scales. More recently, Gaussian processes with periodic and Fourier kernels have been used to detect such periodic patterns [9, 21]. These methods can detect periodic relationships in real-valued signals, but it is unclear how they might be extended to accommodate point events or online inference; the cost of inference scales at least linearly with the number of data points being considered (and is $O(N^3)$ in naive implementations), falling short of our goal of constant-time inference per new event.

More relevant to our topic, Li et al. [18] and its extension [19] have recently considered detecting periodicity in binary event sequences with noise. This approach is based on the intuition that the histogram modulo the correct time period will show peaks corresponding to true periodic events. This trial for various candidate periods, however, makes the method computationally expensive and difficult to adapt to streaming data. This method also is based on the model of a system with a fixed phase and thus fails when applied to more dynamic models.

Periodicity in sequences of symbols or strings has been considered in various other contexts. A filtering method based on the "apriori property", is described in [10]. It aims to find frequent periodic sequences in strings, which is computationally expensive. Other methods for periodicity in symbol sequences including gene sequences are considered in [13, 26].

None of these existing methods can handle input with intrinsic variability which can cause the phase of the periodicity to change. As a result, as we saw in our experiments, these methods are not suitable for online operation or tracking rapidly changing periods of signals.

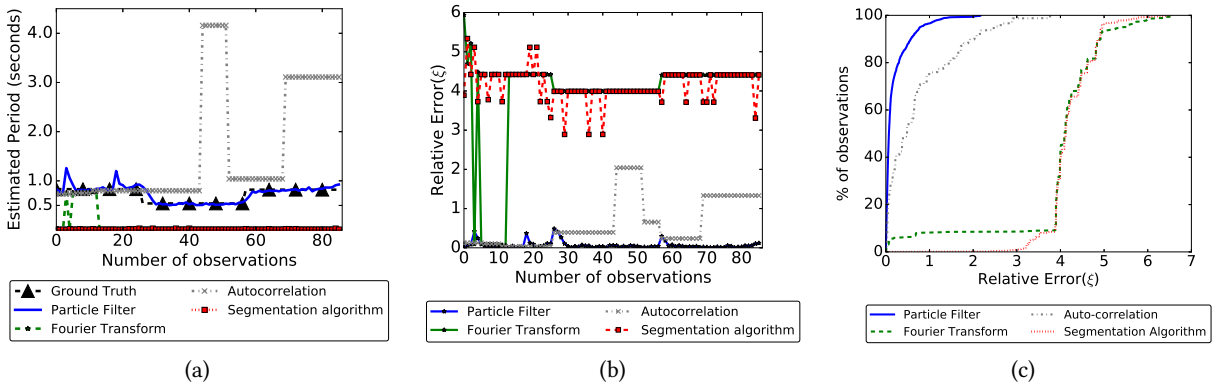
6 CONCLUSION

We have demonstrated the use of a sequential Monte Carlo method to detect and track the periodicity in discrete event streams. Unlike other methods, this technique does not rely on the underlying process sticking to a constant phase. As a result, it adapts to noise and changes in period very quickly. Experiments show this method to be more accurate and efficient than existing methods for detecting periodicity in noisy event streams.

Our basic approach is quite general, and can be adapted to detect and characterize more complex temporal patterns. Doing so will require a more flexible class of event-generating functions, which we expect will be a fruitful area for future research.

REFERENCES

[1] Apache commons math 3.6.1. Binary, 2016.
[2] Dataset. text files, 2017.
[3] Jens Barth, Jochen Klucken, Patrick Kugler, Thomas Kammerer, Ralph Steidl, Jürgen Winkler, Joachim Hornegger, and Björn Eskofier. Biometric and mobile gait analysis for early diagnosis and therapy monitoring in parkinson's disease. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 868–871. IEEE, 2011.
[4] Theophilus Benson, Aditya Akella, and David A Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280. ACM, 2010.
[5] Kevin R Canini, Lei Shi, and Thomas L Griffiths. Online inference of topics with latent dirichlet allocation. In *International conference on artificial intelligence and statistics*, pages 65–72, 2009.
[6] Chen-Mou Cheng, HT Kung, and Koan-Sin Tan. Use of spectral analysis in defense against dos attacks. In *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, volume 3, pages 2143–2148. IEEE, 2002.
[7] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.
[8] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.
[9] Nooshin Haji Ghassemi and M. Diesenroth. Analytic long term forecasting with periodic gaussian processes. In *AISTATS*, pages 303–311, 2014.
[10] Jiawei Han, Guozhu Dong, and Yiwen Yin. Efficient mining of partial periodic patterns in time series database. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 106–115. IEEE, 1999.
[11] Piotr Indyk, Nick Koudas, and S Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *VLDB*, pages 363–372, 2000.
[12] Tanvi Jindal, Prasanna Giridhar, Lu-An Tang, Jun Li, and Jiawei Han. Spatiotemporal periodical pattern mining in traffic data. In *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing*, page 11. ACM, 2013.
[13] Ivan Junier, Joan Hérisson, and François Képès. Periodic pattern detection in sparse boolean sequences. *Algorithms for Molecular Biology*, 5(1):1, 2010.
[14] Panagiota Katsikouli, Rik Sarkar, and Jie Gao. Persistence based online signal and trajectory simplification for mobile devices. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 371–380. ACM, 2014.
[15] Jochen Klucken, Jens Barth, Patrick Kugler, Johannes Schlachetzki, Thore Henze, Franz Marxreiter, Zacharias Kohl, Ralph Steidl, Joachim Hornegger, Bjoern Eskofier, et al. Unbiased and mobile gait analysis detects motor impairment in parkinson's disease. *PloS one*, 8(2):e56956, 2013.
[16] Jennifer R Kwapisz, Gary M Weiss, and Samuel A Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.



(a) (b) (c)

Figure 8: (a) Periodicity estimation on human step data. Particle filter based method closely follows changing periodicity. (b) Error in period estimation in human gait data. Particle filter based method has lowest periodicity detection error. (c) Particle filter based algorithm has small periodicity estimation error for all the samples.

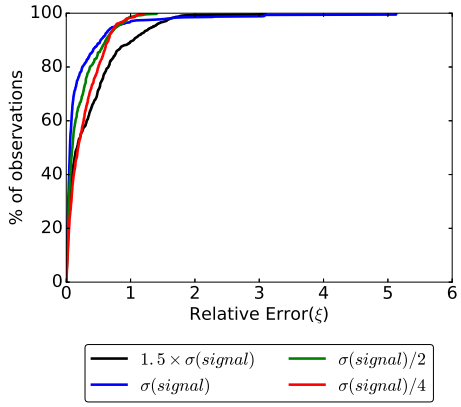


Figure 9: CDF of particle filter error, where event detection thresholds are multiples of the standard deviation of a pre-processed accelerometer signal, showing low estimation error in all cases.

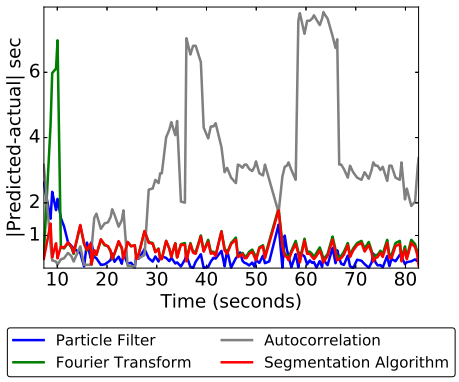


Figure 10: After a small number of events, the next pulse event prediction error (|predicted event time – actual event time|) is generally lower for the particle filter compared to other algorithms.

[17] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9), 2010.

[18] Zhenhui Li, Jingjing Wang, and Jiawei Han. Mining event periodicity from incomplete observations. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 444–452. ACM, 2012.

[19] Zhenhui Li, Jingjing Wang, and Jiawei Han. eperiodicity: Mining event periodicity from incomplete observations. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1219–1232, 2015.

[20] Katja Nummiaro, Esther Koller-Meier, and Luc Van Gool. An adaptive color-based particle filter. *Image and vision computing*, 21(1):99–110, 2003.

[21] Michael A Osborne, Stephen J Roberts, Alex Rogers, Sarvapali D Ramchurn, and Nicholas R Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 109–120. IEEE Computer Society, 2008.

[22] Valentin Radu, Panagiota Katsikouli, Rik Sarkar, and Mahesh K Marina. A semi-supervised learning approach for robust indoor-outdoor detection with smartphones. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, pages 280–294. ACM, 2014.

[23] Leland Stewart and Perry McCarty Jr. Use of bayesian belief networks to fuse continuous and discrete information for target recognition, tracking, and situation assessment. pages 177–185, 1992.

[24] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[25] Michail Vlachos, S Yu Philip, and Vittorio Castelli. On periodicity detection and structural periodic similarity. In *SDM*, volume 5, pages 449–460. SIAM, 2005.

[26] Jiong Yang, Wei Wang, and Philip S. Yu. Mining asynchronous periodic patterns in time series data. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):613–628, 2003.