Efficient Accelerometer-based Event Detector in Wireless Sensor Networks

József Smidla
University of Pannonia
Department of Computer Science and Systems Technology
Veszprém, Hungary
Email: smidla@dcs.uni-pannon.hu

Gyula Simon
University of Pannonia
Department of Computer Science and Systems Technology
Veszprém, Hungary
Email: simon@dcs.uni-pannon.hu

***Abstract*—In this paper an autocovariance-based event detector algorithm is proposed. The algorithm is able to detect events even if the measurements have poor signal-to-noise ratio, and its performance is independent of the characteristic of the input signal. An efficient implementation of the algorithm is also proposed, which allows the utilization of the algorithm on low-end devices, e.g. in wireless sensor networking nodes. The performance of the algorithm has been tested and compared to a conventional filter-based approach, in a vehicle detector application.**

I. Introduction

Accelerometers are successfully utilized in event detection systems, e.g. fall detection [1], movement detection and analysis [2], [3]. Seismic vibrations, caused by various sources, can also be successfully detected by accelerometers, e.g. in footstep detection and vehicle detection [4]. Such detectors are often used in remote, hostile environments, where sensors need to operate autonomously for a long time. The cooperation of multiple sensors is able to provide better performance, e.g. in [5]. In this paper a sensor networking event detector system is proposed, using inexpensive accelerometers. The novelties of the proposed system are the following:

- The proposed signal processing algorithm has high performance, it is able to detect events from measurement with low SNR, thus inexpensive sensors can be utilized.
- The proposed algorithm is independent of the actual signal characteristics (as opposed to e.g. filter based solutions), thus it provides high performance for a wide range for input signals.
- The proposed signal processing algorithm has extremely low computational needs, thus it can be implemented on devices with scarce resources, e.g. typical sensor networking nodes. The implementation details are presented in this paper.
- Real-time measurement results from a vehicle detection application are presented to support the performance of the proposed solution.

In Section II related work is reviewed. Section III introduces the proposed accelerometer-based solution. In Section IV the system is evaluated in a vehicle detection application. Section V concludes.

II. Related work

For monitoring traffic in urban environment two approaches exist: intrusive and non-intrusive sensors. For intrusive sensors stripping of roads is needed, while this is unnecessary for non-intrusive sensors. Intrusive sensors can be inductive loops, magnetometers, microloop probes, pneumatic road tubes, piezoelectric cables and other weigh-in-motion sensors, while the non-intrusive sensors include video image processing, microwave radar, laser radar, passive infrared, ultrasonic, and passive acoustic arrays. However, most of these solutions are energy demanding and expensive, the deployment is cumbersome, and only a few of them can be used in a concealed application [6]. The proposed solution utilizes inexpensive technology, and is easily deployable and concealable.

III. Proposed solution

Our goal was to detect events based on low-cost accelerometer-based measurements. The application of the system is very versatile, e.g. detection of vehicles passing by, detection of intrusion or illegal woodcut in a forest, just to name a few.

A typical recording from an inexpensive accelerometer (BMA-180 [7]) is shown at the bottom of Figure 6, where the signal, caused by a car passing by 10 meters from the sensor, is buried in noise. Obviously some filtering is required to enhance signal quality in order to provide reliable detection. Bandpass filters are often used to enhance SNR (e.g. [4]) but this solution has multiple drawbacks. For optimal result the parameters of the filter must be designed by taking into consideration the spectral distribution of the signal, thus the detector can have high performance only for a narrow class of input signals. In addition this, the implementation of a digital filter (either FIR or IIR) requires substantial amount of computational power, which may not be available on low-end devices. Our proposed solution is based on an alternative approach and uses the autocovariance of signal.

In general, when x and y are random variables, the covariance between them is defined as follows:

978-1-4673-4623-8/13/$31.00 ©2013 IEEE

$$C_{x,y} = E[(x - E[x])(y - E[y])] \tag{1}$$

where $E[.]$ is the expected value operator. $C_{x,y}$ can be simplified as follows:

$$C_{x,y} = E[xy] - E[x]E[y] \tag{2}$$

If the shifted version of x is x_Δ, such that $x_\Delta(k) = x(k - \Delta)$ then the autocovariance of x is defined as follows:

$$C_{x,x}(\Delta) = E[(x - E[x])(x_\Delta - E[x_\Delta)] \tag{3}$$

The alternative way of computing $C_{x,x}(\Delta)$ is the following, similarly to (2):

$$C_{x,x}(\Delta) = E[xx_\Delta] - E[x]E[x_\Delta] \tag{4}$$

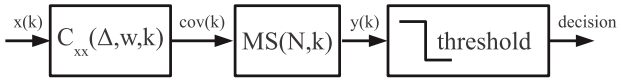

Fig. 1. Flow chart of the algorithm

When x is white background noise then $C_{x,x}(\Delta)$ is low for $\forall\Delta \neq 0$, but when an event is present in x, the autocovariance is high for a wide range of Δ values. Based on this assumption the detector can be built, as shown in Figure 1. First the autocovariance $cov(k)$ of signal $x(k)$ is computed, then the mean square of the autocovariance ($y(k)$) is estimated, and finally a simple thresholding produces the decision. The detailed operation is the following:

Let $\mathbf{x}$ contain successive values of x and let $\mathbf{x}[a:b]$ denote the subvector containing elements from $x(a)$ to $x(b)$, where $a < b$. From such a vector an estimate of $E[\mathbf{x}]$ can be computed as follows:

$$\hat{E}[\mathbf{x}[a:b]] = \frac{1}{b-a+1}\sum_{k=a}^{b} x(k) \tag{5}$$

Furthermore, $E[xx_\Delta]$ can be approximated by $\hat{E}[\mathbf{x}[a:b]^T\mathbf{x}_\Delta[a:b]]$ as follows:

$$\hat{E}[\mathbf{x}[a:b]^T\mathbf{x}_\Delta[a:b]] = \frac{\sum_{k=a}^{b} x(k)x_\Delta(k)}{b-a+1} \tag{6}$$

In our approach (4) can be approximated using (5) and (6) as follows:

$$C_{x,x}(\Delta, w, k) = \hat{E}[\mathbf{x}_1^T\mathbf{x}_2] - \hat{E}[\mathbf{x}_1]\hat{E}[\mathbf{x}_2], \tag{7}$$

where w is an appropriate window size, and $\mathbf{x}_1 = \mathbf{x}[k-w+1:k]$ and $\mathbf{x}_2 = \mathbf{x}_\Delta[k-w+1:k]$.

Finally, the $\mathbf{y}(k)$ output of the algorithm for each input is the mean square value of the last N autocovariance values:

$$\mathbf{y}(k) = \frac{1}{N}\sum_{j=k-N+1}^{k} C_{x,x}(\Delta, w, j)^2 \tag{8}$$

A. Efficient implementation

Computing (7) can be performed using auxiliary variables, which store sums for formulas (5) and (6). The auxiliary variables are the following:

$$\text{sumX} = \sum_{j=k-w+1}^{k} x(j) \tag{9}$$

$$\text{sumY} = \sum_{j=k-w+1}^{k} x_\Delta(j) \tag{10}$$

$$\text{sumProdXY} = \sum_{j=k-w+1}^{k} x(j)x_\Delta(j) \tag{11}$$

A circular buffer called inputBuffer with length $w+\Delta$ is used for the quick update of sumX, sumY and sumProdXY. The buffer includes both $\mathbf{x}[k-w+1:k]$ and $\mathbf{x}_\Delta[k-w+1:k]$, and let us suppose $\Delta \leq w$. Figure 2 shows an example, where variable lastIndex indicates the position of last entry in the buffer. The position of last element of $\mathbf{x}_\Delta[k-w+1:k]$ is (lastIndex + 1) mod $(w+\Delta)$, stored in variable oldestXIndex. Similarly, variable oldestYIndex contains the index of last element of $\mathbf{x}[k-w+1:k]$, where oldestYIndex = (index + 1 + Δ) mod $(w+\Delta)$. When a new sample $x(k+1)$ arrives, the variables sumX, sumY, and sumProdXY are updated using entries indexed by oldestXIndex and oldestYIndex, and also $x(k+1)$ and $x(k+1-\Delta)$. Finally, the algorithm updates the indices and stores the new value in the buffer. Figure 3 shows the pseudo code of the algorithm.

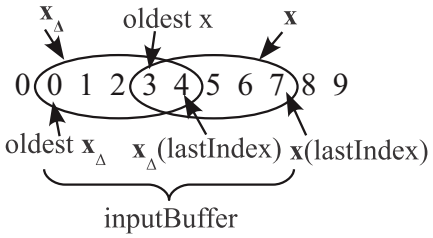

Fig. 2. Utilization of the input buffer, where $w = 5$ and $\Delta = 3$

The mean-square of the last N autocovariance values can be efficiently computed by using another circular buffer with length N (see bufferC in Figure 3). Based on this output a simple thresholding mechanism can decide whether an event is present or not. It can be seen that execution time of the introduced algorithm is independent of the quality and length of the input, and it needs very few and simple operations.

IV. Evaluation

The implemented algorithm uses integer and fixed point variables. As it can be seen in Figure 3, the proposed algorithm needs 6 subtractions, 8 additions, 5 multiplications, 4 divisions, and 4 conditions per sample. However, when w and N are power of 2, then divisions can be substituted with bit-shifts.

procedure init
  sumX := $\sum_{i=0}^{w-1} x(i+\Delta)$
  sumY := $\sum_{i=0}^{w-1} x(i)$
  sumProdXY := $\sum_{i=0}^{w-1} x(i)x(i+\Delta)$
  sumCS := 0
  lastIndex := $w+\Delta-1$
  oldestXIndex := Δ
  oldestYIndex := 0
  indexCov := 0
  for i := 0 **to** $\Delta-1$ **do**
    inputBuffer[i] := 0
  end
  for i := Δ **to** $w+\Delta-1$ **do**
    inputBuffer[i] := $x(i)$
  end
end

function $y(k)$:= detect($x(k)$)
  oldX := inputBuffer[oldestXIndex]
  oldY := inputBuffer[oldestYIndex]
  sumX := sumX - oldX
  sumY := sumY - oldY
  sumProdXY := sumProdXY - oldX * oldY
  lastIndex := (lastIndex + 1) mod $(w+\Delta)$
  $\mathbf{x}_\Delta(k)$:= inputBuffer[(lastIndex - Δ) mod $(w+\Delta)$]
  inputBuffer[lastIndex] := $x(k)$
  sumX := sumX + $x(k)$
  sumY := sumY + $\mathbf{x}_\Delta(k)$
  sumProdXY := sumProdXY + $x(k)\mathbf{x}_\Delta(k)$
  cov := $\frac{\text{sumProdXY}}{w} - \frac{\text{sumX*sumY}}{w^2}$
  sumCS := sumCS - bufferC[indexCov]^2
  sumCS := sumCS + cov^2
  bufferC[indexCov] := cov
  oldestXIndex := (oldestXIndex + 1) mod $(w+\Delta)$
  oldestYIndex := (oldestYIndex + 1) mod $(w+\Delta)$
  indexCov := (indexCov + 1) mod N
  $y(k) := \frac{\text{sumCS}}{N}$
end

Fig. 3. Pseudo code of the algorithm

A. Application

The test device includes the BMA 180 accelerometer and a 8-bit ATMega128RFA1 processor, running at 16MHz with 16 kByte of RAM and 128 kByte of flash memory. It also has an internal EEPROM with size of 4 kByte to store configuration data. The internal radio of the ATMega128RFA1 chip is used to send measurement/detection data in a wireless manner. Figure 4 shows the deployed device. The algorithm described in Figure 3 was implemented in nesC under TinyOS [8].

B. Validating assumptions

The main assumption of the proposed algorithm was that the autocovariance of the noise is low with respect to the autocovariance of the signal. To check the properties of the noise, measurement data was collected without events. From the collected noise data the autocorrelation function was calculated, shown in Figure 5. The results clearly show that the measurement noise is really white.

Fig. 4. The sensor node

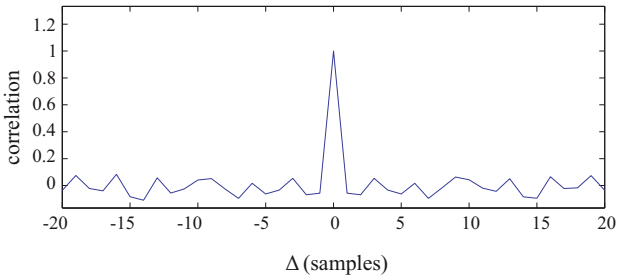

Fig. 5. Autocorrelation of the measurement noise

Figure 6 shows a measurement where two vehicles within 15 seconds passed by the sensor. The row input signal is shown in Figure 6(a), while Figure 6(b) represents the estimated autocovariances of the signal, calculated with a window of size 512. It is clearly visible that the autocovariance of the noise is significantly lower than that of the vibration caused by the vehicles, thus our initial assumptions are supported by the measurements.

C. Effects of parameters

The effect of different w window sizes and Δ values has been tested. In the experiment measurement records for one vehicle, and the measurement noise was used. Figure 7 shows a signal-to-noise-ratio-like quantity, which was calculating in the following way: $y(k)$ was calculated and averaged for the signal when the vehicle was present and also $y(k)$ was calculated and averaged when only noise was present. The ratio of the two quantities is shown in Figure 7. It is clearly visible that with increasing Δ the SNR is changing periodically, showing the periodic nature of the vibration, caused by the vehicle. The SNR is highest at the fundamental frequency, corresponding to $\Delta \approx 18$, but at $\Delta = 1$ a somewhat lower, but still high peak is present. Since the signal frequency depends on the actual event, the optimal Δ depends on the input

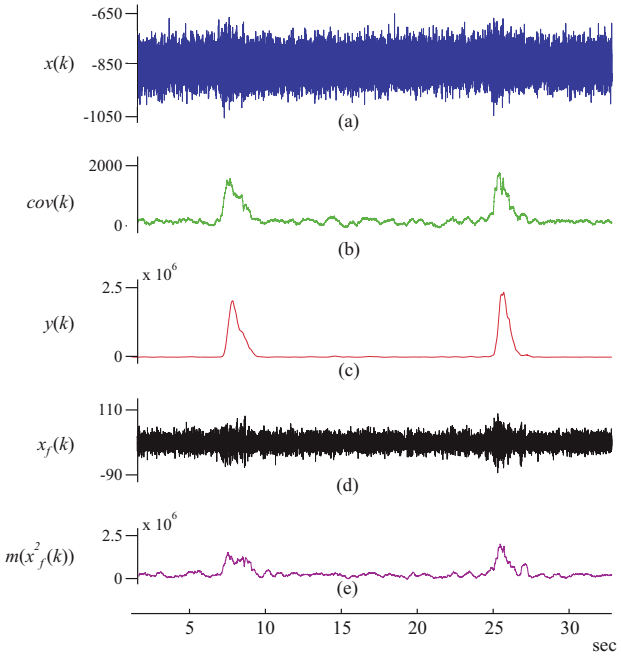

Fig. 6. Signal processing of a measurement with two vehicles passing by (a) output of the sensor $x(k)$, using sampling frequency of 1200 Hz, (b) autocovariance $cov(k)$ of $x(k)$, calculated with $w = 512$, (c) the proposed detector output $y(k)$, with $N = 512$ (d) bandpass filtered version $x_f(k)$ of $x(k)$, (e) mean-square of $x_f(k)$, calculated with running window of length $N = 512$.

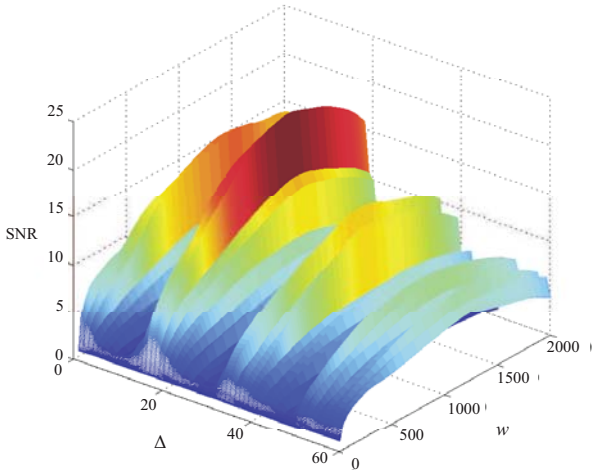

Fig. 7. Signal-noise ratio as a function of parameters w and Δ

signal's characteristics. To avoid this dependence we opted to use $\Delta = 1$, which is suboptimal but provides good SNR, independently of the signal characteristics.

The effect of window size w is also visible in Figure 7: increasing w increases the SNR as it is intuitively expected. However, when w is comparable to the length of the vibration caused by the vehicle ($w \approx 1500$), the SNR saturates, and the further increase of w decreases the SNR. This effect is also expected, since in long windows not only the useful signal but also a significant amount is noise is also present, thus decreasing the obtainable SNR. Although the optimum would be around $w = 1500$, in order to decrease memory requirements and provide fast response, a compromise is proposed with $w = 256$, where the SNR is already high enough, according to Figure 7.

As Figure 8(b) shows, the output varies in a very large range, but even the smallest peaks can safely be distinguished from the noise, as the zoomed portion of Figure 8(b) shows. The threshold can be set to e.g. three times the RMS of the measured noise. Alternatively, an adaptive threshold could be used. Using different thresholds, heavy vehicles, like trucks can easily be distinguished from cars.

D. Comparing with power based detectors

The proposed algorithm is compared with a power-based approach, where the input signal is passed through a bandpass filter, and the power of the filter output is used for detection [4]. In the tests, frequencies from 80-100 Hz were used, where most of the input signal power was present. Figure 6(d) shows the output of the filter, and in Figure 6(e) the mean-square of the filter output is shown. It can be seen that the output of the power-based solution has much lower SNR than that of the proposed solution, shown in Figure 6(c). In the example a 128-tap FIR filter was used, which requires 128 multiplications and additions per sample, while the proposed approach requires only 5 multiplications and 8 additions.

E. Performance test

As discussed in section IV-C, parameters $w = 256$ and $\Delta = 1$ were used, and the sampling the frequency was 300 Hz. The full detector was implemented in the device, but for testing purposes the sensor output was also downloaded. The device was placed near to a surfaced road at a distance of 5.6 meters. Figure 8(a) shows the raw signal $x(k)$ of the accelerometer, where the signal, caused by different moving cars, is heavily buried in noise. In Figure 8(b) the output $y(k)$ is shown. It can be seen that even slow and small cars were successfully detected from a safe distance, using a weak signal.

F. Advantages and disadvantages

The introduced autocovariance-based algorithm needs much less operations than conventional power-based methods, thus processing is feasible in low-end devices. However, the implementation of the proposed algorithm requires 2 circular buffers, so the sensor device needs 2-3 kbytes of RAM for this purpose.

V. Summary

In this paper a novel method was proposed to provide accelerometer-based event detection. An autocovariance-based signal processing algorithm was utilized to allow robust event detection in case of poor signal to noise ratio. The proposed

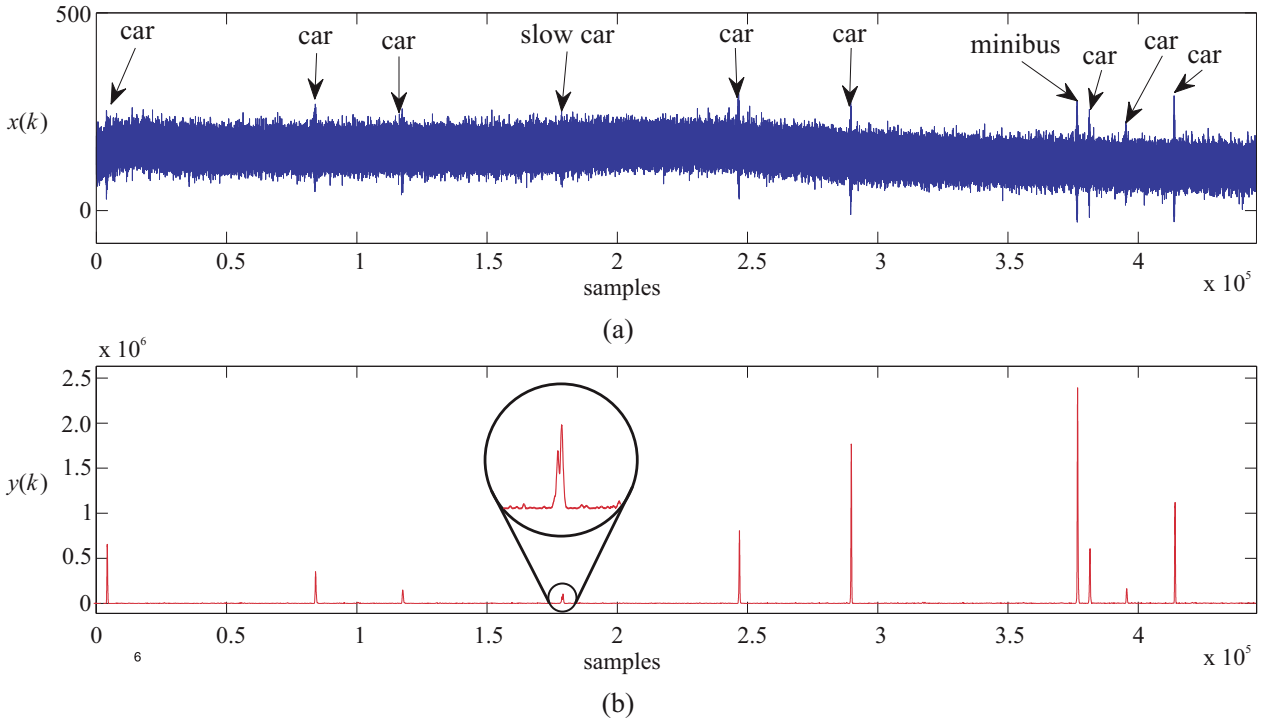

Fig. 8. Operation of the vehicle detection algorithm (a) raw signal data, (b) output $y(k)$ of the signal processing algorithm

solution allows a highly efficient implementation with very low computational needs (8 additions 5 multiplications, 4 shift operations per sample). Thus the proposed algorithm can be implemented on low-end devices: in the paper an 8-bit microcontroller implementation was presented and tested. As opposed to conventional power-based approaches, the proposed solution does not not need a priori signal model, and thus can be used for excitation signals of quite different origins, with the same efficiency. The performance of the algorithm was illustrated in a vehicle detection application.

ACKNOWLEDGMENT

We acknowledge the financial support of the Hungarian State and the European Union under projects TAMOP-4.2.2.A-11/1/KONV-2012-0072, TAMOP-4.2.2/B-10/1-2010-0025, and GOP-1.1.1-11-2011-0070.

REFERENCES

[1] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, Washington, DC, USA, 2009, pp. 138–143.

[2] K. Lorincz, B. Chen, G. Challen, A. Chowdhury, S. Patel, P. Bonato, and M. Welsh, "Mercury: A wearable sensor network platform for high-fidelity motion analysis," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, Berkeley, CA, USA, November 2009, pp. 183–196.

[3] M. Mathie, A. Coster, N. Lovell, and B. Celler, "Detection of daily physical activities using a triaxial accelerometer," *Medical and Biological Engineering and Computing*, vol. 41, pp. 296–301, 2003.

[4] R. Hostettler and W. Birk, "Analysis of the adaptive threshold vehicle detection algorithm applied to traffic vibrations," *Sort*, vol. 20, no. 50, p. 100, 2011.

[5] R. Bajwa, R. Rajagopal, P. Varaiya, and R. Kavaler, "In-pavement wireless sensor network for vehicle classification," in *IPSN'11*, 2011, pp. 85–96.

[6] L. Mimbela and L. Klein, *Summary of Vehicle Detection and Surveillance Technologies Used in Intelligent Transportation Systems*. The Clearinghouse, 2003.

[7] *BMA180 Digital, triaxial acceleration sensor*. [Online]. Available: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Accelerometers/BST-BMA180-DS000-07_2.pdf

[8] P. Levis and D. Gay, *TinyOS Programming*, 1st ed. New York, NY, USA: Cambridge University Press, 2009.