Technical Report

Scalable Data Acquisition for Densely Instrumented Cyber-Physical Systems

Aida Ehyaei

Eduardo Tovar

Nuno Pereira and Björn Andersson

HURRAY-TR-110111

Version:

Date: 01-06-2011

Scalable Data Acquisition for Densely Instrumented Cyber-Physical Systems

Aida Ehyaei, Eduardo Tovar, Nuno Pereira and Björn Andersson

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.hurray.isep.ipp.pt

Abstract

Consider the problem of designing an algorithm for acquiring sensor readings. Consider specifically the problem of obtaining an approximate representation of sensor readings where (i) sensor readings originate from different sensor nodes, (ii) the number of sensor nodes is very large, (iii) all sensor nodes are deployed in a small area (dense network) and (iv) all sensor nodes communicate over a communication medium where at most one node can transmit at a time (a single broadcast domain). We present an efficient algorithm for this problem, and our novel algorithm has two desired properties: (i) it obtains an interpolation based on all sensor readings and (ii) it is scalable, that is, its time-complexity is independent of the number of sensor nodes. Achieving these two properties is possible thanks to the close interlinking of the information processing algorithm, the communication system and a model of the physical world.

Scalable Data Acquisition for Densely Instrumented Cyber-Physical Systems

Aida Ehyaei, Eduardo Tovar, Nuno Pereira and Björn Andersson
IPP-HURRAY Research Group
CISTER/ISEP, Polytechnic Institute of Porto
Porto, Portugal
{aaei,emt,nap,baa}@isep.ipp.pt

***Abstract*— Consider the problem of designing an algorithm for acquiring sensor readings. Consider specifically the problem of obtaining an approximate representation of sensor readings where (i) sensor readings originate from different sensor nodes, (ii) the number of sensor nodes is very large, (iii) all sensor nodes are deployed in a small area (dense network) and (iv) all sensor nodes communicate over a communication medium where at most one node can transmit at a time (a single broadcast domain). We present an efficient algorithm for this problem, and our novel algorithm has two desired properties: (i) it obtains an interpolation based on all sensor readings and (ii) it is scalable, that is, its time-complexity is independent of the number of sensor nodes. Achieving these two properties is possible thanks to the close interlinking of the information processing algorithm, the communication system and a model of the physical world.**

Keywords: CPS Foundations; Real-Time Systems; Sensor Networks.

I. INTRODUCTION

Although the information technology transformation of the 20th century appeared revolutionary, a bigger change is on the horizon. The term Cyber-Physical Systems (CPS) has come to describe the research and technological efforts that will ultimately allow the interlinking of the real-world physical objects and the cyberspace efficiently [1-2].

The integration of physical processes and computing is not new. Embedded systems have been in place for a long time and these systems often combine physical processes with computing. The revolution will come from massively deploying networked embedded computing devices allowing instrumenting the physical world with pervasive networks of sensor-rich embedded computation [2]. As Moore's law continues, the cost of a single embedded computer equipped with sensing, processing and communication capabilities drops toward zero. This makes it economically feasible to densely deploy networks with very large quantities of such nodes. Accordingly, it is possible to take a very large number of sensor readings from the physical world, compute quantities and take decisions out of them. Very dense networks offer a better resolution of the physical world and therefore a better capability of detecting the occurrence of an event; this is of paramount importance for a number of foreseeable applications.

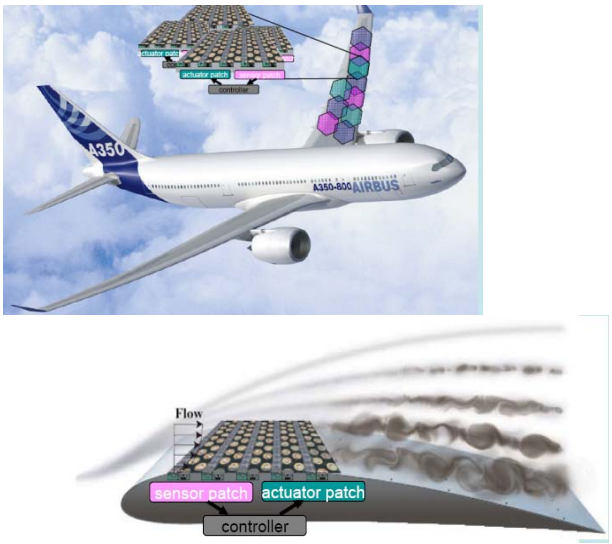


Figure 1. Example system of sensors/actuators embedded across an aircraft wing to perform active flow control (from the WICAS [6] project website).

Structural health monitoring (SHM) of physical infrastructures (bridges, aircrafts, etc.) is an example of CPS applications where high-spatial resolution sensing is required [3]. Other examples are the ongoing efforts within the aircraft industry to respond to environmental concerns by developing technologies that will allow sustained air travel growth while minimizing overall carbon footprint [4, 10]. Active flow control, achieved through local modulation of aircraft skin surfaces (see Figure 1), is one such, yet to be seen, technological development with the potential to offer significant reduction of drag and related fuel consumption and emissions [5]. Significant drag reduction can be obtained by performing active flow control using the aircraft skin surfaces. One approach to achieve that flow control is to perform local adjustments of the skin surfaces using a very dense deployment of sensor/controller/actuator nodes embedded in the aircraft wings and fuselage [6]. Implementing such an active flow control system requires thousands of sensors, controllers and actuator systems (smart skin patches) to be embedded across the aircraft wings and fuselage to create an active aircraft [6]. However, the scale of such a system poses huge challenges in terms of interconnectivity and timely data processing. In this

paper we will look at efficient scalable data acquisition methods for such densely instrumented cyber-physical systems.

CPS with high-spatial resolution sensing must typically fulfill the following requirements (R1 to R4):

R1. Computation (for estimating the state of the physical world) must be based on sensor readings from many sensor nodes, preferably all sensor nodes. The rational for this requirement is that if the computation would be based only on sensor readings from a single pre-determined sensor node (or a pre-defined small subset of sensor nodes) then we derive no benefit from the large number of sensor nodes available.

R2. Sensor nodes must be able to communicate. The rationale for R2 follows from R1.

R3. Broadcast media (such as a shared wired bus or a wireless channel) must be used for communication. The rationale for R3 follows from the fact that a point-to-point communications network would be too expensive as the number of sensor nodes becomes large.

R4. The computation (for estimating the state of the physical world) must be performed within low (and bounded) delay. The rationale for R4 follows from the fact that control algorithms must obtain an estimate of the physical world that is not too old.

Fulfilling all these four requirements is challenging. Consider m sensor nodes in a single broadcast domain. One may believe (based on R3) that any computation which fulfills R1 will have time-complexity which depends on the number of sensor nodes ($O(m)$) because in a single broadcast domain, no two nodes may transmit simultaneously. Therefore, even a simple quantity such as the minimum of the sensor readings of all sensor nodes in a single broadcast domain, would be difficult to compute with a low time-complexity.

However, recent research has made available, for dense networks, algorithms that can compute certain aggregate quantities (such as MIN, MAX or COUNT) with a time-complexity that is independent of the number of nodes (as we will explain in Section II). These algorithms are based on *dominance protocols* (also called *binary countdown protocols*) [12], which are used in the CAN bus [13] and in WiDom [14]. It may be possible to use these algorithms as basic building blocks so that other quantities can be computed as well. This brings a hope that all the above mentioned requirements can be fulfilled.

Although extremely scalable (time-complexity independent of number of nodes) methods for computing MIN (or MAX) are useful in some applications, it would be even more useful if there was a data acquisition algorithm that made it possible to produce an approximate representation of all sensor readings so that an application could compute any desired quantity based on this approximate representation. In fact, sensor readings are often spatially and temporally correlated with only a few abrupt changes (in time or space). Therefore, an interpolation function would be a suitable representation for approximating all sensor readings.

Previous research work [8, 9] has proposed algorithms for obtaining an interpolation of sensor readings from different sensor nodes, and these algorithms are based on dominance protocols, presenting therefore excellent scalability properties for dense instrumented CPS. We believe however that it is possible to further improve the performance of those algorithms not only by taking advantage of knowledge of often-occurring spatial correlation of sensor readings but also by embedding the dynamics of the physical phenomenon in the algorithm which computes the interpolation.

Therefore, in this paper, we develop a novel algorithm for data acquisition of sensor readings in a densely instrumented cyber-physical system. Our new algorithm produces an interpolation of all sensor readings. It fulfills the four requirements stated above and in order to improve its accuracy, it has a model of the dynamics of the physical world embedded into it.

Contribution of this paper: Previous works tackling densely deployed networked sensor systems have proposed algorithms that obtain an interpolation of sensor readings by using the dominance principle. The novelty of the algorithms in this paper is the embedding of a physical model in these algorithms. We also show that our new physically-based interpolation scheme can be implemented on commodity sensor platforms. Based on our experimentation, we find important trade-offs for designers: some physical models are very simple and cause very low run-time overhead (and hence allow a small sampling period), whereas other physical models are more accurate but cause a larger run-time overhead.

The remainder of this paper is structured as follows. Section II provides a background on previous related work, including how to use dominance protocols for efficiently computing certain aggregate quantities in dense networked sensor systems. Section III presents a previously known algorithm for acquiring an interpolation in such systems. In Section IV we define a framework to embed a model of the dynamics of the physical environment into the interpolation scheme. Section V presents a novel algorithm based on this framework while Section VI addresses the implementation of the new algorithm and its experimental evaluation. Finally, in Section VII, conclusions are drawn.

II. Related Work

The wireless sensor network research community has proposed several solutions for processing sensor data. In multi-hop networks, nodes may self-organize into a converge-cast tree with a base station at the root. Techniques for computing useful aggregated quantities (such as MIN) that offer good performance have been proposed previously [7]. Such techniques achieve good performance as a result of exploiting the opportunities for parallel transmission and of en-route aggregation of data. In densely instrumented systems where even a very small area may contain several hundreds of sensor nodes, the performance of the data aggregation techniques as those surveyed in [7] is limited by the fact that nodes in the same broadcast domain cannot transmit in parallel. Thus, the time-complexity of those approaches also heavily depends on the number of sensor nodes. This results in long delays for collecting the information of all nodes and obtaining the

required set of measurements. Therefore, these techniques may not be suitable for feedback control systems that require a short and bounded delay from sampling to actuation.

To tackle this problem, a family of novel distributed algorithms has been recently introduced [8, 9]. In those algorithms, communications and computations are tightly coupled with the physical environment. Notably, they can compute certain aggregate quantities (MIN, MAX or COUNT) with a time-complexity that is independent of the number of sensor nodes. The approach in those works is based on the intelligent exploitation of Medium Access Control (MAC) mechanisms that are inspired on dominance protocols [12]. In the following sub-sections, dominance-based MAC protocols and aggregation methods which use them are described briefly.

Work on the construction of contour maps based on sensor readings share some constraints with our work, and justifies a brief discussion here. A contour map is used to visualize sensor fields by constructing a map having contour lines through points of equal attribute values (e.g. temperature, elevation). For example, a contour map of the temperature of a sensor field will display regions covered by sensors having the same temperature reading under the same contour line. Some works have approached this problem [19-21] by aggregating similar values at intermediate nodes to reduce the number of packets that need to be exchanged to construct the contour map. Interestingly, authors working on this problem have noted that traditional data aggregation methods cannot further improve the scalability of the network, based on the observation that the complexity of such methods is always dependent on the number of sensor nodes [22]. For this reason, the approach taken in [22] was to intelligently select a small portion of the nodes to contribute to the construction of the contour map. Although in a different context, our approach for interpolation employs the same basic idea of selecting a set of sensor nodes to contribute to the interpolation.

A. Basic Principles of Dominance-based MAC Protocols

Dominance protocols [12] have important characteristics for the approaches described in this paper. These protocols have good properties for supporting timeliness in systems with event-triggered messages. Moreover, they are capable of simultaneous "non-destructive" transmission of information in the same broadcast domain.

In the wired implementation of this protocol, the Controller Area Network (CAN) bus [13], all messages have a unique contention field which could be their priority. When a node has a request to transmit, after waiting a predetermined time until the channel becomes idle, it starts a conflict resolution phase (arbitration phase). In this phase, the nodes send their contention field, bit-by-bit, starting with the most significant bit (see Figure 2(a)). The medium is devised in such a way that nodes can hear a recessive bit (a logical '1') only if no other node sends a dominant bit (a logical '0'); the bus behaves as a logical wired-AND. The nodes which hear a dominant bit while themselves send a recessive bit, refrain from arbitration. At last the only one node that reaches the end of arbitration without hearing a dominant bit (unless it was sending it as well), proceeds with transmitting the data.

The wireless implementation of a dominance MAC was dubbed WiDom [14]. During the conflict resolution phase, which is called tournament in WiDom, a node with a recessive bit should listen to the medium to assess whether any dominant bit is being transmitted or not. But, wireless transceivers can hardly be transmitting and receiving at the same time. Thus, when the transmitted bit is dominant there is no need to sense the medium, whereas, when the bit to transmit is recessive, nothing has to be effectively sent, and only the medium state has to be sensed.

Various interesting features of dominance-based protocols (CAN and WiDom are examples) can be exploited to obtain aggregate quantities in large scale dense networks, with a time-complexity that is very low and independent of the number of nodes. Such mechanisms are being used as a key building block in densely instrumented Cyber-Physical Systems as is discussed in the next subsection.

B. Quantity Aggregation

By associating the priorities of messages to physical quantities (such as temperature or acceleration), several high-performance algorithms for data processing can be devised in which the time-complexity is independent of the number of nodes. For instance, if each node uses the value of its sensor reading instead of an arbitrary priority, the node winning the contention for the medium will be the one with the minimum (MIN) of the sensed values [8, 9]. In [8], it is demonstrated that CAN-enabled platforms can be used to compute various aggregate quantities, such as MIN (or MAX). In [9] the authors show the same, but for wireless systems using a sensor-platform optimized for such scalable data aggregation.

In order to understand how a dominance-based MAC protocol can be used to efficiently compute an aggregate quantity, consider how to compute the minimum of all sensor readings. Figure 2(b)-(d) shows this. One naïve approach would be to use a time-division multiple-access (TDMA) scheme and assign one timeslot to each sensor node and let a sensor node transmit its sensor reading in its slot. Figure 2(b) shows this. After one TDMA cycle, a node knows all sensor readings and the minimum can be computed but, unfortunately, the time-complexity is $O(m)$, where m is the number of sensor nodes. The same type of naïve scheme can, of course, be implemented with a prioritized MAC protocol. Figure 2(c) shows this. But with a prioritized MAC protocol, a sensor node can use its sensed data as a priority and hence the MAC protocol will grant medium access to the sensor node with the minimum sensor reading — see Figure 2(d). Since a dominance-based prioritized MAC protocol makes all sensor nodes know the priority of the sensor node which was granted the medium, it holds that all sensor nodes will know the minimum of the sensor reading. This makes it possible to compute MIN.

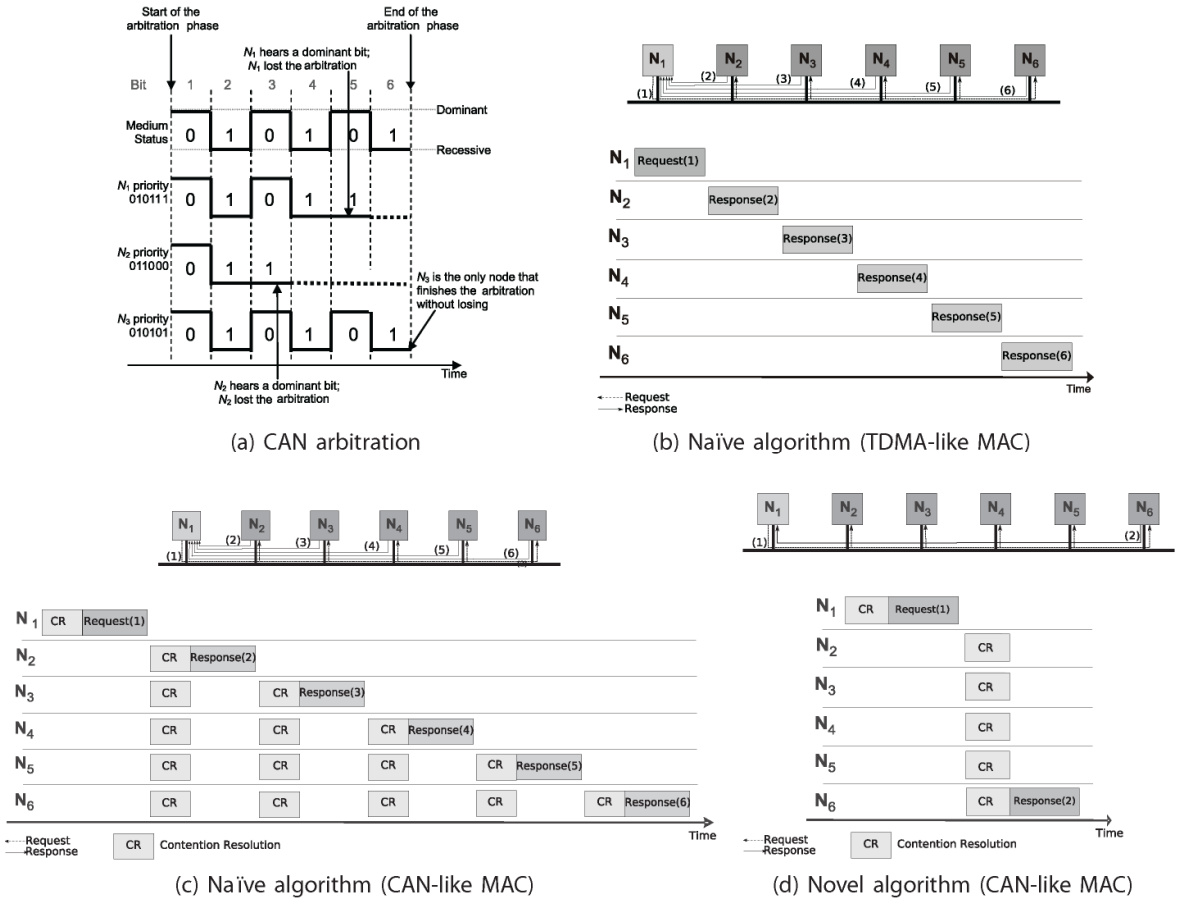


Figure 2. Dominance/Binary-Countdown arbitration motivating examples. (a) Example of bitwise arbitration; (b) example application where $N1$ needs to know the minimum (MIN) temperature reading among its neighbors ($N2$ to $N6$); (c) possible solution for the example application using a CAN-like MAC, using fixed priorities for the messages; (d) possible solution for the example application exploiting the properties of a CAN-like MAC, where priorities are assigned at runtime according to the sensed values.

The importance of the above method is that its computation time is independent of the number of nodes in the broadcast domain. With this method, the minimum value (MIN) and the maximum value (MAX) can be obtained with a time-complexity of $O(npriobits)$, where *npriobits* is the number of bits used to represent the data.

III. INTERPOLATION

Many CPS applications behave as follows:

1. do forever:
2. Each sensor node takes a new sensor reading.
3. Sensor nodes form a (potentially approximate) representation of all sensor readings.
4. The representation of sensor readings is used, for example, to compute an actuation command.

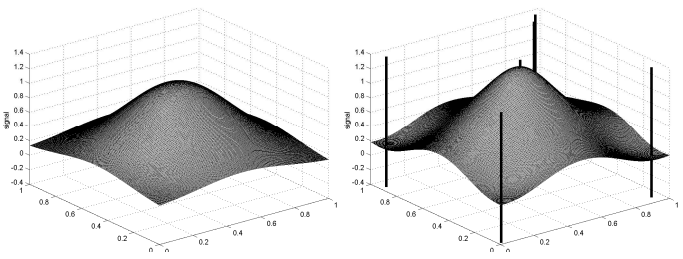
The execution of line 3 in the pseudo code above requires an algorithm which acquires a (potentially approximate) representation of all sensor readings. Ideally, this algorithm should, when line 4 executes, offer a small deviation of the representation of sensor reading as compared to the physical world. In this section, we will discuss how to achieve this for the special case that the physical world does not change during the execution of line 3 and 4; later sections will discuss how to deal with the more general (and realistic) case in which changes in the physical world can occur at any time.

Previous work [8] proposed an algorithm for obtaining an interpolation. The interpolation is a function $f(x, y)$ where x and y are space coordinates and the function $f(x, y)$ approximates sensor readings throughout the area of interest. The function $f(x, y)$ is represented by a set of control points, denoted S, where each control point $q_k \in S$ has three attributes x_k, y_k and s_k, with the meaning that evaluating the interpolation at the location (x_k, y_k) should give the value s_k. On locations where no control point exists, the function $f(x, y)$ is defined as a weighted average of control points; this is called *weighted-average interpolation* (WAI). Formally, the function $f(x, y)$ is defined as:

$$f(x,y) = \begin{cases} 0 & if\ S = \emptyset \\ s_k \quad if\ \exists q_k \in S : x_k = x \wedge y_k = y & \\ \frac{\sum_{k \in S} s_k \cdot w_k(x,y)}{\sum_{k \in S} w_k(x,y)} & otherwise \end{cases} \quad (1)$$

where weights, $w_k(x, y)$, are given by:

$$w_k(x,y) = \frac{1}{(x_k - x)^2 + (y_k - y)^2} \quad (2)$$



(a) Original Signal (b) Interpolated Signal

Figure 3. Interpolation example [8]

Let N_i denote a sensor node. Let (x_i,y_i) denote the location of this sensor node and let s_i denote the sensor reading of this sensor node. We let e_i denote the error of the interpolation at sensor node N_i and we let e denote the maximum error over all sensor nodes. Formally, we express this as:

$$e_i = |s_i - f(x_i, y_i)| \quad (3)$$

and

$$e = \max_{i=1..m} e_i \quad (4)$$

where m is the number of nodes.

An algorithm for efficiently constructing S is proposed in [8] (we refer to it as *Basic Interpolation Algorithm)*. The main idea is that initially the interpolation is zero on each location (this is represented by setting S to the empty set). Then, each sensor node evaluates the interpolation at its location and compares it with its sensor reading and the sensor node with the maximum error is granted the medium for transmitting its location and sensor reading, and this information is added to S. This is repeated k times (where the value of k is selected by the designer). Pseudo code for this algorithm is shown below (each sensor node executes the algorithm and a sensor node can read the variable i to obtain its identifier):

```
1: S ← ∅
2: for j ← 1 to k do
3:     calculate the interpolation function f(x_i,y_i) based on S
4:     calculate e_j.
5:     select a sensor node N_k with the maximum e_k, that is e_k = e. This can
           be achieved using the MAX computation mentioned in Section II.
6:     the location and the sensor reading of N_k forms a control points;
           add this control point to S
7: end for
```

Figure 3 illustrates the operation of the interpolation scheme. Figure 3(a) shows how a physical quantity varies as a function of space coordinates x and y. Figure 3(b) shows an interpolation which is an approximate representation of this physical quantity (represented in Figure 3(a)); the lines indicate the location of control points in S.

IV. A Framework for Using a Physical Model in Interpolation

The interpolation algorithm in the previous section assumes that the physical quantity does not change while the interpolation algorithm executes. In fact, the physical world may change while the interpolation algorithm executes, and this can cause the interpolation represented by S to diverge significantly from the physical quantity being measured. To tackle this problem, we will embed in the interpolation algorithm a model about how the physical world changes. After adding a new control point to S (line 6 in the previous pseudo-code), the control points in S can be updated based on a model about the dynamics of the physical environment. We are interested in a simple framework to describe how this update should be performed. This simple framework should (i) be sufficiently expressive so that many real physical dynamics can be modeled in the framework and (ii) it should be possible to execute efficiently. We achieve this by performing a linear transformation on each element in S. The pseudo-code for such a framework can be obtained by modifying (changes are shaded in gray) the basic interpolation algorithm as follows:

```
1: S ← ∅
2: for j ← 1 to k do
3:     calculate the interpolation function f(x_i,y_i) based on S
4:     calculate e_j.
5:     select a sensor node N_k with the maximum e_k, that is e_k = e. This can
           be achieved using the MAX computation mentioned in Section II.
6:     the location and the sensor reading of N_k forms a control point;
           add this control point to S.
7:     for each element (x_i,y_i, s_i) in S do
8:         xnew_i ← A_1,1*x_i + A_1,2*y_i + A_1,3*s_i + A_1,4
9:         ynew_i ← A_2,1*x_i + A_2,2*y_i + A_2,3*s_i + A_2,4
10:        snew_i ← A_3,1*x_i + A_3,2*y_i + A_3,3*s_i + A_3,4
11:        replace the element (x_i,y_i, s_i) in S by (xnew_i,ynew_i, snew_i)
12:    end for
13: end for
```

With this framework, an application designer must assign values to A. We can see that this framework allows different operations to the signals, such as scaling, translation and rotation. Given our particular interest in active flow control (refer to application illustrated in Figure 1), let us consider that we have a set S which offers an interpolation of the air pressure on top of a wing and the aircraft is moving in the direction towards the x-axis. Suppose that overall, the air moves at a speed of v m/s in the direction of x-axis (v is assumed to be a constant). We can obtain a new set S for an interpolation 0.1 ms later by performing the following operation (lines 7-12):

```
:  xnew_i ← 1*x_i + 0*y_i +  0*s_i + v*0.0001
:  ynew_i ← 0*x_i + 1*y_i +  0*s_i + 0
:  snew_i ← 0*x_i + 0*y_i +  1*s_i + 0
```

It is obvious that the better the model of the dynamics of the physical world is, the lower the error of our interpolation. It should be stressed that our interpolation scheme will offer an interpolation anyway, even if the physical model is simply incorrect; this is because the algorithm revises itself in each iteration based on measured errors.

An application designer clearly must assign values to A. This can be done either by assigning static values at design time, or by finding suitable values for A based on sensor readings. We will see an example of a very simple interpolation scheme using the latter in the next section.

V. New Algorithm For Estimating The Overall Image Of the Physical Process

Tracking methods usually take some information from the subject to predict its behavior in future. To react fast to changes in the physical process, knowing the type of change that affects the signal could be of great help. To illustrate this, assume monitoring the temperature of a field with a fixed (not moving) source of heat which has different degrees of intensity. Here we know the changes in the signal (temperature degree across the field) could be increasing or decreasing with different amount of changes with respect to the location of a sensor and with respect to time.

The idea behind the novel proposed interpolation algorithm is using the system features to recognize the type of changes in the physical quantities. This leads to design a system-dependent interpolation algorithm which can cope better with fast changing physical signals. The basics of our approach is that, by knowing the type of change in the physical quantity, the amount of change in the interpolation control points can be measured and applied in future steps of the interpolation.

In the rest of the paper the algorithm, analysis and also simulation and implementation results are presented for the case that the signal level is increased or decreased with respect to time. The physical model can be described simply as follows:

$$: \; xnew_i \leftarrow 1*x_i + 0*y_i + 0*s_i + 0$$
$$: \; ynew_i \leftarrow 0*x_i + 1*y_i + 0*s_i + 0$$
$$: \; snew_i \leftarrow 0*x_i + 0*y_i + 1*s_i + g_i$$

where g_i is the differential of i^{th} interpolation point. Algorithm 1 describes the proposed approach for this case. Compared to the previous pseudo-code, this algorithm presents some details about the implementation of the interpolation algorithm (details such as how to compute e_j, how to encode the priority for the MAX computation mentioned in Section II) and employs the system call **send_and_rcv** which causes the sensor node to compete for the medium with priority **prio** to transmit a packet with data payload **snd_pack**. Regardless of whether the sensor node transmitted or not, this call returns the winning priority and the packet that was transmitted on the channel. The maximum value that the MAC protocol can use for a priority is denoted as ***MAXP***; for example if 29 priority bits are used in the priority field, then $MAXP = 2^{29}-1$.

All nodes execute Algorithm 1 where, at each iteration of the interpolation (except for the first one), after receiving the information of a new control point, the node that sent the previous control point sends its value again (line 16). Then, it is possible for all the nodes to measure the approximate differential, g, in that control point (line 17). This information will be applied in the next iterations to obtain the interpolation as Equation 5 shows:

$$f(x,y) = \begin{cases} 0 & if\ S = \emptyset \\ s_i\,(j) & if\ \exists q_i \in S;\ x_i = x, y_i = y \\ \dfrac{\sum_{i=1}^{j} s_i\,(j) \cdot w_i(x,y)}{\sum_{i \in S} w_i(x,y)} & otherwise \end{cases} \quad (5)$$

Algorithm 1 Differential Interpolation Algorithm

Require: All nodes start Algorithm 1 simultaneously.
Require: k denotes the desired number of interpolation points.
Require: A node N_i knows x_i,y_i and s_i.
Require: The code below is executed by every node. A node can read the variable i and obtain its node index.

```
1: function find nodes() return a set of packets
2:   S ← ∅
3:   for j ← 1 to k do
4:     g_j ← 0
5:   end for
6:   for j ← 1 to k do
7:     Calculate f(x_i,y_i) in Equation 5 and assign it to the variable
         "myinterpolatedvalue"
8:     s_i ← read sensor
9:     error ← abs( s_i - to integer(myinterpolatedvalue) )
10:    temp_prio ← error × (MAXNNODES + 1) + i
11:    prio ← (MAXP+1) – temp_prio
12:    snd_pack ←< s_i,x_i,y_i>
13:    <winning_prio, rcv_pack> ← send_and_rcv( prio, snd_pack)
14:    S ← S ∪ { rcv_pack }
15:    if j ≠ 1 then
16:      the new sensed data of (j-1)th control point is received.
17:      g_j ← the change in value of the control point
18:    end if
19:  end for
20:  return S
21: end function
```

where:

$$s_i(j) = s_i + (j - i) \cdot g_i \quad (6)$$

and g_i is the differential of i^{th} interpolation point, $s_i(j)$ is the value of the i^{th} interpolation point in the j^{th} iteration, and s_i is the value of the i^{th} interpolation point when it is added to S. The other parameters are as described previously for Equation 1. Because the differential of points in the calculation of the interpolation is used, this algorithm is called *Differential Interpolation*. Later in the paper, it will be useful to simplify the numerator and denominator of Equation 5 as follows:

$$num_j = num_{j-1} + s_j \cdot w_j(x,y) + \sum_{i=1}^{j-1} g_i \cdot w_i(x,y) \quad (7)$$

$$denom_j = denom_{j-1} + w_j(x,y) \quad (8)$$

VI. Implementation Of Differential Algorithm

The mechanisms here proposed can have significant practical impairments. Namely, the computations required for each iteration of the interpolation can make the execution of such mechanisms impractical in current low-power wireless sensor network platforms. In this section, we will present the details of the implementation of Algorithm 1. This implementation considers the platform's limitations regarding floating point operations.

Some further assumptions about the behavior of the signal can be incorporated in the algorithm to significantly decrease the time required to perform computations. Thus, before presenting the details of the implementation, we will discuss different variations of the algorithm, considering small changes in the assumptions about the signal changes.

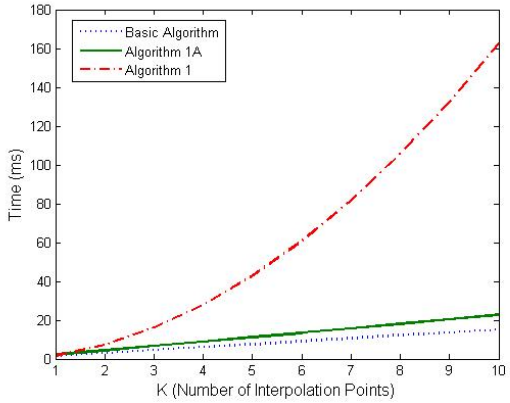


Figure 4. Execution time as a function of k for Basic and Differential algorithms in a real-world platform.

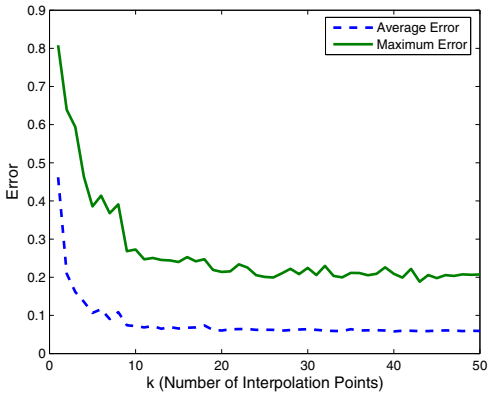


Figure 5. Interpolation error for a static signal

1) Variations of the New Algorithm

Having more information of the signal may make the implementation simpler and exhibit a lower execution time. The following subsections discuss some assumptions that can be introduced to reduce the computation time of the proposed algorithm.

2) Constant Differentials (Algorithm 1A)

According to Algorithm 1, the value of $f(x, y)$ has to be recomputed (line 7 in Algorithm 1) in all iterations, thus also the sums in Equation 5 must be recomputed. If the change in the signal is monotonous in each point, the differentials will be constant and this allows us to consider that the differentials will not change after insertion in the set S. Thus, it is not required to compute all terms of Equation 5 in each iteration. It is possible to reduce the computation time of the algorithm by maintaining the partial sums in the numerator and denominator of Equation 5. This is reflected in Algorithm 1A. The implementation assumes that the differential will not change after it is received.

The following functions are used in Algorithm 1A: **abs**, **read_sensor**, **send_and_rcv**, **snd_value, rcv_value**. Function call **abs** returns the absolute value of the value given as argument; **read_sensor** is a function call used for getting the sensor value from the analog-to-digital converter; **send_and_rcv** is a function that will cause a tournament to be performed and returns the wining priority and the packet received; **snd_value** and **rcv_value** are function calls that can be used to respectively broadcast or receive a value to to/from other nodes.

We performed a brief analysis of the time required to compute Basic Algorithm and the new Differential Algorithm in real-world sensor network platforms. This was done by implementing the algorithms in the MicaZ sensor network platform [23]. The code was compiled with no compiler optimizations and the execution time was measured using one of the microcontroller's real-time clocks. The results are presented in Figure 4. For Algorithm 1, we considered that each node had to compute the differential and also computed the interpolated value at all iterations, which is the worst-case computation scenario. As we can see, the execution time of Algorithm 1 increases much faster. This is because it needs to re-compute Equation 1 at each iteration.

Considering the changes in the control points with respect to time, makes the interpolation more accurate. On the other hand, maintaining the partial sums reduces the execution time of the algorithm, at the cost of the (less) generality of the algorithm.

3) Similar and Constant Differentials for All Iterations (Algorithm 2)

Algorithm 1A allows reducing the execution time of the proposed algorithm. Its time complexity is $O(k^2)$. On the other hand, the complexity of the Basic Algorithm is still lower (it is $O(k)$).

When the changes in the signal are equal for all the points in equal periods of time, the time complexity of the algorithm can be reduced to $O(k)$. In this case, it is only required to calculate the gradient once and use it in all next iterations for all the points. We name this algorithm as Algorithm 2.

We do not present the full algorithm, as Algorithm 2 is similar to Algorithm 1A. In Algorithm 2, a variable *offset* is introduced to keep the sum of changes per iteration, and the loop in lines 68-70 of Algorithm 1A is removed. Now, it is enough for each node to add the offset to the values of the control points when it wants to calculate its interpolated value. The interpolation function is similar to the one given by Equation 5 while Equations 6 and 7 are updated as follows, respectively:

$$s_i(j) = s_i + (j - i)g \tag{9}$$

and

$$num_j = num_{j-1} + s_j \cdot w_j(x, y) + g \cdot offset_j \tag{10}$$

where:

$$offset_j = offset_{j-1} + w_j(x, y) \tag{11}$$

The denominator is the same as Equation 8.

B. Study of the Error

Using simulation experiments, we will now study the error of interpolation schemes. We will first (in Section VI.B.1) see how different parameters affect the error of the interpolation

Algorithm 1A Implementation of Improved Interpolation Algorithm

```
1.  Require: (MAXS+1) × (MAXNNODES+1) + MAXNNODES ≤ MAXP.
2.  Require: (MAXS+1) < 2 (16-1) (due to size of gradient variable).
3.  Require: x_i, y_i denotes the node's coordinates.
4.  Require: The code below is executed by every node. A node can read the variable i and obtain its node index.
5.  type interpolation pack = record
6.          value: uint16
7.          x : uint16
8.          y : uint16
9.          differential : uint16;
10. end record
11. prio, winning prio, temp prio, temp prio MAC order: uint32
12. snd pack, rcv pack, prv_rcv_pack: interpolation pack
13. S : set
14. t : time
15. max error, error, myinterpolatedvalue : uint16
16. dx, dy, sq : uint32
17. num, denom, temp : uint64
18. update_myinterpolation : Boolean
19. differential: int16;
20. function find nodes() return a set of packets
21.         myinterpolatedvalue ←0
22.         num←denom←differential←0
23.         differential_sum←0
24.         S←{};
25.         update_myinterpolation←TRUE
26.         compute_diff←FALSE;
27.         for j←1 to k do
28.             my_sensor_value←read_sensor (); // get sensor value
29.             error←abs(my_sensor_value - myinterpolatedvalue ) // compute error
30.             temp_prio← ((uint32) error) × (MAXNNODES + 1) + i
31.             temp_prio_MAC_order← ( (1 << 27) - 1) – temp_prio
32.             prio← (1 << 27) + temp_prio_MAC_ // encode value to send as a priority
33.             snd_pack   < my_sensor_value, x_i, y_i >
34.             <winning_prio, rcv_pack> ←send_and_rcv(prio, snd_pack) // perform tournament
35.              if (j>1) then
36.                 if (compute_diff = TRUE) then
37.                     differential ← read_sensor() - (myinterpolatedvalue)
38.                        // compute differential
39.                     snd_value(differential) // send differential; note only previously winning node will do this
40.                     compute_diff←FALSE
41.                     mydifferential← differential
42.                 else
43.                     differential ←rcv_value();// receive differential; note only all nodes will do this, except the previously winning node
44.                 end if
45.             end if
46.             prv_rcv_pack.differential←differential
47.             dx← x_i –rcv pack.x
48.             dy← y_i - rcv pack.y
49.             sq←dx*dx + dy*dy  // w^-1
50.             differential_sum ← differential_sum + differential / prv_sq
51.             temp← ((uint64) rcv_pack.value) << 32
52.             num←num + differential_sum + temp div sq // note value is assigned to temp.  variable
53.             temp← ((uint64) 1) << 32
54.             denom←denom + temp div sq // note value is assigned to temp. variable
55.             prv_sq←sq
56.             if (winning_prio == prio) then
57.                 update_myinterpolation ←FALSE  // update_myinterpolation is only  FALSE   when node is the winner
58.                  myinterpolatedvalue ← my_sensor_value
59.                 compute_diff← TRUE
60.                 in_set←TRUE
61.             else
62.                 if (in_set = FALSE) then
63.                     myinterpolatedvalue← num div denom
64.                 else
65.                     myinterpolatedvalue←myinterpolatedvalue+mydifferential
66.                 end if
67.             end if
68.             for each <interpolation_point> in S
69.                  interpolation_point.value← interpolation_point.value+interpolation_point.differential
70.             end for
71.             S←S U {rcv_packet}
72.         end for
73.         return (S)
74. end function
```

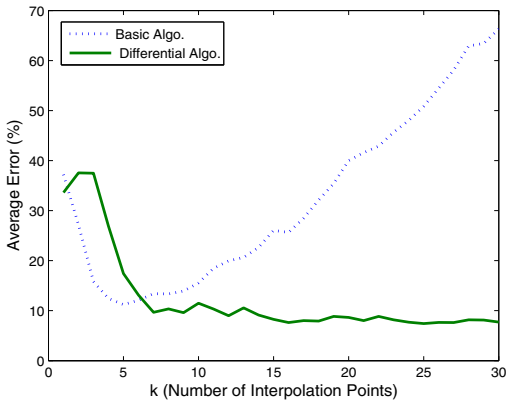


Figure 6. Average Error of Basic and Differential algorithms with constant (4%) change in signal per interpolation round

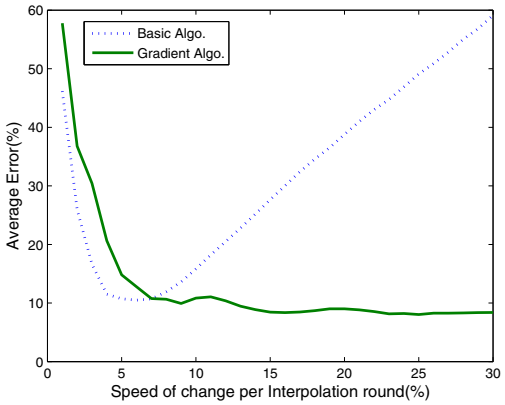


Figure 7. Average Error of Basic and Differential algorithms with random (up to 4%) change in signal per interpolation round

respective algorithms. Concepts and experimental setup can be found in Appendix A.

1) Impact of parameters

Figure 5 shows how the interpolation error varies as a function of *k* for static signals. Simulation results show that the Basic Algorithm works well for smooth signals that change slowly in the time.

Figures 6 and 7 show how the interpolation error varies with k when signal changes occur during interpolation. The rate of change in the signal is fixed or random (up to 4% of maximum amplitude of the signal) in each interpolation round. As we can see, the Basic Algorithm has a poor result and cannot follow the changes in the signal. The average error of the Basic interpolation scheme is high and keeps rising as the algorithm continues execution and time progresses. The explanation for this behavior is intuitive as well. When a control point is added to the set of the interpolation points, S, the one that was added previously may be already measuring a very different value.

Simulation results show a great improvement in interpolation of the signal when the Differential Algorithm is used instead of the Basic Algorithm. The Differential Algorithm has less than 10% average error in interpolating the signal. When the rate of changes is constant, the Differential Algorithm has slightly better results compared to random (but limited) changes, while the Basic algorithm offers even worse results.

Note however, that each iteration of the Interpolation in the Differential Algorithm is longer than in the Basic Algorithm, since there is a re-sending of data in each iteration. If the arbitration takes x time units and sending data takes y time units, the communication time of each iteration in Differential Algorithm last (x + 2 × y) time units compared to (x + y) in Basic Algorithm.

2) Comparison between algorithms

The comparison of the previously presented algorithms, in terms of average error, is shown in Table 1 and 2 for different changes in signal and for two different values of k.

The tables show that for small scaling changes, all the algorithms have acceptable results. But, when the changes in signal are not slow the new dynamic algorithms (1A and 2) have more accurate interpolation results. The tables also illustrate the inefficiency of the Basic Algorithm when the changes in the signal are larger and faster.

TABLE I. PERCENTAGE OF AVERAGE ERROR FOR *K*=10

Algorithm	**Type of change in signal per Interpolation round**		
	Different Increase (up to 4%)	***4% Increase***	***1% Scaling***
Basic Algorithm	9.23	15.82	7.10
Algorithm 1A	9.75	10.96	7.91
Algorithm 2	8.49	10.36	7.94

TABLE II. PERCENTAGE OF AVERAGE ERROR FOR *K*=20

Algorithm	**Type of change in signal per Interpolation round**		
	Different Increase (up to 4%)	***4% Increase***	***1% Scaling***
Basic Algorithm	18.23	38.78	4.75
Algorithm 1A	6.19	8.99	4.96
Algorithm 2	5.70	9.01	4.74

VII. CONCLUSION

We have shown a very simple way to use a model of the physical world as a means to speed up the process of obtaining an interpolation of sensor readings and this gives a distributed computer system the ability to detect changes in the physical world very quickly. This is important for Cyber-Physical Systems. Ongoing work involves synthesizing such an algorithm from a model of the physical world (analogous to the way coefficients in a PID controller are selected based on knowing the transfer function of plant dynamics and given certain performance goals the controller should meet).

Target applications (as described in the introduction) include installations in, for example, aircrafts and cars. In such

installations, one possible communication media is the structure of the aircraft/car itself. The metal plates of the structure can be used for communication (for example, as wave guides). These and other similar possibilities are being investigated (in a joint research effort with specialized companies) as they might present themselves as appealing practical solutions in such settings.

ACKNOWLEDGEMENT

This work was supported by Portuguese Science and Technology Foundation (Fundação para Ciência e Tecnologia - FCT) CISTER Research Unit - FCT UI 608 and the Cooperating Objects Network of Excellence (CONET), a EU-funded project under ICT, Framework 7.

REFERENCES

[1] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar. "Opportunities and obligations for physical computing systems," IEEE Computer, 38(11), pages 23-31, November 2005.

[2] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. "Connecting the physical world with pervasive networks," IEEE Pervasive Computing, pages 59-69, January-March 2002.

[3] J. Caffrey, R. Govindan, E. Johnson, B. Krishnamachari, S. Masri, G. Sukhatme, K. Chintalapudi, K. Dantu, S. Rangwala, A. Sridharan, N. Xu, and M. Zuniga, "Networked Sensing for Structural Health Monitoring," In: Proceedings of the 4th International Workshop on Structural Control, Columbia University, NY, June 2004.

[4] ACARE, Strategic Research Agenda 1 - Volume 2, Section 3. "The Challenge of the Environment," ACARE Report, October 2002. Available online: http://www.acare4europe.org/docs/es volume1-2/volume2-03-environment.pdf

[5] J. Reneaux, "Overview on Drag Reduction Technologies for Civil Transport Aircraft," European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS04), Jyvskyl, July 2004.

[6] Wireless Interconnectivity and Control of Active Systems (WICAS), http://www.shef.ac.uk/systemsutc/projects/wicas

[7] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," Wireless Communications, IEEE , vol.14, no.2, pp.70-87, April 2007.

[8] B. Andersson, N. Pereira, W. Elmenreich, E. Tovar, F. Pacheco, and N. Cruz. "A scalable and efficient approach to obtain measurements in CAN-based control systems," In IEEE Trans. Industrial Informatics 4(2): 80-91 (2008).

[9] N. Pereira, R. Gomes, B. Andersson, and E. Tovar. "Efficient aggregate computations in large-scale dense WSN," In 15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'09), pages 317-326, San Francisco, California, USA, 2009.

[10] C. Dickey, "The Flying Prius", Newsweek, July 2010.

[11] Fly-by-Wireless. : A Revolution in Aerospace. Architectures for Instrumentation and Control. NASA/CANEUS Workshop. NASA/JSC/ES6/George Studor. 3/27/2007

[12] A.K. Mok and S. A. Ward, "Distributed Broadcast Channel Access," Comput. Networks, Vol. 3, November 1979.

[13] Bosch GmbH, Stuttgart, Germany. CAN Specification, ver. 2.0, 1991.

[14] B. Andersson, N. Pereira, and E. Tovar. "Widom: A dominance protocol for wireless medium access," IEEE Transactions on Industrial Informatics, vol. 3(2), May 2007.

[15] N. Pereira, B. Andersson, E. Tovar, and A. Rowe. "Static priority scheduling over wireless networks with multiple broadcast domains," In Proceedings of the 28th Real Time Systems Symposium (RTSS07), Tucson, U.S.A., December 2007.

[16] A. Ehyaei, E. Tovar, and N. Pereira. "Scalable and Efficient Data Processing in Networked Control Systems," HURRAY-TR-101004. http://www.cister.isep.ipp.pt/docs/

[17] E. Tovar, B. Andersson, N. Pereira, M. Alves, S. Prabh and F. Pacheco, "Highly Scalable Aggregate Computations in Cyber-Physical Systems: Physical Environment Meets Communication Protocols," Proceedings of the 7th International Workshop on Real-Time Networks (RTN'08), Prague, Czech Republic, July 1, 2008.

[18] B. Andersson, N. Pereira, E. Tovar, R. Gomes, "Using a prioritized medium access control protocol for incrementally obtaining an interpolation of sensor readings," Seventh Workshop on Intelligent solutions in Embedded Systems,, pp. 29 – 36, Ancona, June 2009.

[19] J. M. Hellerstein, W. Hong, S. Madden and K. Stanek, "Beyond Average: Toward Sophisticated Sensing with Queries," in Proceedings of IPSN, 2003.

[20] X. Meng, T. Nandagopal, L. Li and S. Lu, "Contour Maps: Monitoring and Diagnosis in Sensor Networks," Computer Networks, 2006.

[21] W. Xue, Q. Luo, L. Chen and Y. Liu, "Contour Map Matching For Event Detection in Sensor Networks," in Proceedings of ACM SIGMOD, 2006.

[22] Mo Li, and Yunhao Liu, "Iso-Map: Energy-Efficient Contour Mapping in Wireless Sensor Networks", IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol 22, No. 5, May 2010, Pages 699-710.

[23] http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148%3Amicaz

APPENDIX A. EXPERIMENTAL SETUP

The signal which is used in the simulation experiment is a smooth signal with three peaks as shown in Figure 8, and is described by the following Equation:

$$p(x,y) = 0.6\left(e^{-20((x-0.2)^2+(y-0.8)^2)} + e^{-20((x-0.5)^2+(y-0.5)^2)} + e^{-20((x-0.8)^2+(y-0.2)^2)}\right) + 0.1 \quad (12)$$

The maximum amplitude is 1. It is considered that the all the points of signal increase 4% of the maximum amplitude (or 0.04) per interpolation round for the experiments and results described in Figure 6 and randomly but up to 4% of the maximum amplitude for the experiments and results described in Figure 7. For the Tables 1 and 2, 1% scaling means multiplying the amplitude of each point by 1.01.

For evaluating results of the interpolation algorithms, Average Interpolation Error (*AIE*) and Maximum Interpolation Error (*MIE*) are defined as follows:

$$AIE = \frac{\sum_{i=1}^{m} |VS_i - IS_i|}{m} \quad (13)$$

$$MIE = \max_{i=1..m} |VS_i - IS_i| \quad (14)$$

where VS_i is the measured value of sensor i, IS_i is the calculated value in the geographical position of sensor node i by the interpolation method and m is the number of sensor nodes. These two parameters are calculated at the end of Interpolation to see how close the interpolation is to the signal.

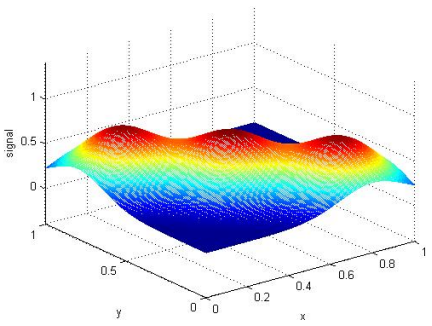


Figure 8: Signal with three peaks