

# Surface reconstruction: hardware requirements of a SOM implementation

James Shuttleworth  
Cogent Computing  
Coventry University  
Coventry, United Kingdom  
csx239@coventry.ac.uk

Elena Gaura  
Cogent Computing  
Coventry University  
Coventry, United Kingdom  
csx216@coventry.ac.uk

Robert M. Newman  
Cogent Computing  
Coventry University  
Coventry, United Kingdom  
r.m.newman@coventry.ac.uk

## ABSTRACT

Surface reconstruction from wireless sensor network node locations is a desirable but computationally expensive process, with requirements typical of any non-trivial processing. We examine an algorithm for reconstructing surface models. The implementation is used as a case study to discuss the issues associated with moving processing onto the sensor network. The study is motivated by the authors' view that large networks (more than 1000 nodes) should be autonomous distributed systems capable of "in network" processing at all levels, including the application level. The two competing paradigms are considered: small motes with limited capabilities using a distributed implementation and powerful motes using localised processing. The state of the art in hardware design and implementation for motes is considered and a case is built for the need, from the perspective of processing capability, for powerful motes.

## Categories and Subject Descriptors

C.3 [Computer Systems Organization]: Special-Purpose And Application-Based Systems—*Real-time and embedded systems*; C.2.4 [Computer Systems Organization]: Computer Communication Networks—*Distributed Systems*

## General Terms

Algorithms, measurement, performance, experimentation

## Keywords

Surface reconstruction, self organising maps, wireless sensors, motes, field sensing

## 1. INTRODUCTION

This paper discusses a possible means of reconstructing the 3D surface on which a large wireless sensor network (WSN) has been randomly deployed. The approach is based on position data from the sensor nodes and the application of a surface reconstruction algorithm (self organising maps) to merge the data into a continuous and efficient surface representation. Implementation issues related to the application (processing, storage and communication requirements) are discussed.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

REALWSN '06 Uppsala, Sweden

Copyright 2006 ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

## 2. SURFACE RECONSTRUCTION

Constructing a surface approximation from sensor data is a special case of the unorganised points problem, in which a continuous surface is required to be reconstructed from a cloud of points. In the domain of WSNs, the problem is to turn a collection of surface coordinates received from sensor nodes into a geometric model of the surface in the form of a mesh. A mesh is simply a collection of connected vertices (points). Self organising maps are one of the algorithms that can be used for surface reconstruction [6]. As an approximation algorithm, this method uses a fixed number of vertices. In its most simple form, the algorithm can be described as the repeated application of the following steps:

1. A mesh surface is constructed
2. A point from the cloud is selected and the vertex from the surface that is closest to the selected point is found
3. The vertex and its neighbours are moved toward the cloud point. The movement of the neighbours is less than the selected vertex, proportionate with distance from the selected vertex.
4. Repeat from step 2

In our implementation described here, a neighbourhood of  $4 \times 4$  vertices around the selected vertex are moved. The weighting for vertex,  $i$ , in the neighbourhood is calculated as  $w_i = f^{distance(T,i)}$  Where  $f$  is a constant controlling fall-off,  $T$  is the neighbourhood centre and  $distance$  is the L2 distance between two points. The value of  $f$  was selected as 0.55 through experimentation. Experimental data are based on the Grand Canyon terrain model from the Georgia Institute of Technology Large Geometric Models Archive [5]. Our area of interest spans a square region 108 kilometres to a side, or 11664Km<sup>2</sup>. The source data is in the form of a height map. The point cloud is populated by randomly selecting  $x$  and  $z$  values and reading the corresponding  $y$  value from the height map. The  $xz$  plane is assumed to be horizontal. That is, the height at a given point,  $(x, z)$  is obtained from the height map,  $H(x, z)$  The number of vertices and nodes required to represent a given surface depends on a number of factors, such as the detail required, the size of the area to be modelled or acceptable error for the application. A useful trait of self organising maps is that they make quite efficient use of a fixed number of vertices.

## 3. IMPLEMENTATION CONSIDERATIONS

Surface reconstruction is useful for visualisation of surface geometry, but an even more useful application would be to make the surface representation available to the sensor nodes themselves. There are very good arguments for doing as much processing in-network as possible [4][2]. A

**Table 1: Typical processor characteristics and performance evaluation.**  $I_d$  = instructions to calculate point-to-point distance,  $I_c$  = instructions to find the nearest vertex,  $I_{total}$  = total instructions for a 4000 vertex, 3200 sensor model.

|                          | Atmel Mega 128 | ARM 7      | ARM 9      | Blackfin      | x86 (PIII) |
|--------------------------|----------------|------------|------------|---------------|------------|
| <b>Clock speed (S)</b>   | 16Mhz          | 20MHz      | 120MHz     | 600MHz        | 3000Mhz    |
| <b>Word width</b>        | 8b             | 32b        | 32b        | 32b           | 32b        |
| <b>RAM</b>               | 4KB            | 64KB       | 128KB      | 52KB ... 1Mb+ | 1GB        |
| $I_d$                    | 143            | 33         | 33         | 42            | 26         |
| $I_c$                    | 473            | 153        | 153        | 249           | 291        |
| $I_{total}$              | 9159568000     | 2114448000 | 2114448000 | 2691984000    | 1668656000 |
| $T(\frac{I_{total}}{S})$ | 572s           | 106s       | 18s        | 4s            | 0.6s       |

terrain model is extremely useful in allowing motes to further analyse and condense data before passing it on. The self organising map method of surface reconstruction is simple to understand and relatively easy to implement on desktop hardware. Yet implementing the algorithm within the wireless sensor network is not straightforward. At the outset, we have two options: distribute processing across the network or collate data at a single node for processing. The apparent benefits of maximally distributed processing are tempting. However, parallelisation tends to require more communication between processors and, since communication can easily outstrip processing in terms of power usage [4], it might not be a good solution. As Estrin *et al.* state, “the seemingly optimal way of solving a problem often results in algorithms whose communication energy costs exceed their benefits” [1]. The “big mote” [4] or “active sensor” [2] camp, on the other hand, advocate more processing power per mote, giving them the ability to manage more complex tasks single-handedly. The arguments are compelling: a richer instruction set can *improve* efficiency by requiring fewer operations; a 32 bit channel to primary storage makes more memory available, simplifying algorithms; 32 bit registers reduce the complexity of performing operations on larger numbers; and so on. It is not unreasonable to assume a requirement of 3200 sensor nodes distributed over an area of  $11000Km^2$  to create a mesh of 4000 vertices. Assuming vertices and nodes are represented as triplets of 32-bit floats, the data alone requires 103KB of memory. With only 64KB of working memory, motes such as the Mica2 using the Atmel Mega 128 processor are unable to process terrain models of this scale. 128KB variations of the ARM processor do have sufficient primary storage for the data. However, since the quoted requirements are a bare minimum, overheads and other memory allocations for the application are likely to leave very little free storage for other tasks. ARM flavours with greater on-chip memory, or the Blackfin processors would make a more sensible choice for memory intensive algorithms such as surface reconstruction. To assess the benefits of larger motes with faster and more instruction-rich processors, we have implemented core parts of the algorithm in C. These functions were compiled for each of the processors in the study (Atmel 128 Mega, ARM7/9, Blackfin) and the number of instructions counted. If  $I_d$  is the number of instructions required to find a single distance and  $I_c$  the number required to find the closest vertex to a node, the total number of instructions for a given set of nodes and vertices is  $I_{total} = (|M|I_d + I_c)|N|$ . Further, knowing the clock speed of the processors allows us to make a simple estimate of execution time. The instruction set of the Atmel processor requires the greatest number of operations to execute the algorithm. Combined with the slowest clock speed, execution time is estimated at 572 seconds, or 9.5 minutes.

The ARM7 takes 106s. The Blackfin processor requires more operations than the ARM processors, but its much higher clock rate gives it the best performance out of the embedded processors, taking just 4s to compute a result. Of course, what is defined as an acceptable execution time is dependant on the application. Generally, however, we can assume that over 9 minutes delay in a field sensing network could be problematic.

#### 4. CONCLUSION

Processing power, communication and storage issues have been examined and found to favour more powerful motes. Although surface reconstruction is not a general requirement of wireless sensor networks, it does represent quite well other problems requiring computationally expensive solutions. Finally, we must consider cost. Even with many benefits, a higher cost might be prohibitive. However, it is very difficult to make a confident assertion that an 8 or 16 bit processor is lower in cost than a 32 bit one. For instance, the current price of the MSP430 used in the Telos style motes is currently \$5.60 in thousand-off quantities. An ARM processor to a similar specification can be purchased in similar quantities for less than four dollars simply because they are produced in massive quantities - the ARM architecture accounts for the majority of the 32 bit computers in the world today, and the parts are made for high volume applications such as mobile phones, PDAs and network switches. It is likely that the ‘piece part’ cost of a ‘big mote’ is of the same order as that of a ‘little mote’.

#### 5. REFERENCES

- [1] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, 2001.
- [2] P. Levis, D. Gay, and D. Culle. Active sensor networks. In *Proceedings of the 2nd USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2005.
- [3] Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak1, and Mani B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM’01*, 2001.
- [4] R. Newman and E. Gaura. Size does matter: the case for big motes. In *To appear in the Proceedings of Nanotech 2006*, Boston, MA., May 2006.
- [5] USGS and Chad McCabe. Grand canyon terrain. Georgia Institute of Technology Large Geometric Models Archive. [http://www-static.cc.gatech.edu/projects/large\\_models/index.html](http://www-static.cc.gatech.edu/projects/large_models/index.html), 1998.
- [6] Yizhou Yu. Surface reconstruction from unorganized points using self-organizing neural networks. In *Proceedings of IEEE Visualization*, pages 61–64, 1999.