An Implementation of Surface Reconstruction from Point Set

RUI YANG, Simon Fraser University
ZHUOLI JIANG, Simon Fraser University

Abstract goes here.

Additional Key Words and Phrases: MST, point set, PCA, Marching Cube

1. INTRODUCTION

In computing graphic, an important research area is the surface reconstruction from an unorganized 3D point set. The input point set has no connectivity, geometry and normal informations. Those points are on or near a surface M that we want to derive a surface $M\prime$ to approximate. If we can recover the structure, topology and interconnectivity from an unknown point set, we can not only better understand the original characteristic of the point set and use the output for further processing, but also contribute solving this general problem that appears in other area such as MRI, laser range scanner, radar and seismic surveys and mathematical models.

In early 90s, Hoppe et al[1] proposed an algorithm to infer the topology, the presence of boundaries and also geometry of surface M from set of points $x1, ..., xnR^3$. This algorithm is considered as a classic technique to solve surface reconstruction problem. Nowadays, several techniques are raised towards this problem or subset of this problem. For example, Poisson [2] is highly resilient to data noise but it assumes the orientation of points. Hoppe's algorithm doesn't assume any information of the input point set. Based on that the algorithm can deal with a more general problem, which can be (or partly) used to solve any specific instance. Therefore we implemented Hoppe's algorithm in this project in order to better study on this classic algorithm and research area thus we can gain a solid knowledge of surface reconstruction.

2. ALGORITHMS

The input of Hoppe's algorithm is an unorganized 3D point set $x1, ..., xn \in R^3$. We first compute the set $Nbhd(x_n)$ Describe Input here.
What does each algorithms do
how these algorithms are connected. (What do they contribute to the whole implementation.)

2.1. K Nearest Neighbours

2.2. Principle Component Analysis

2.3. K Euclidean Mininal Spanning Tree

What is k-EMST, how it is different from normal EMST

2.4. Normal Propagation

2.5. Marching Cubes

what each algorithm is doing.

ALGORITHM 1: Marching Cubes Algorithms

Input: Iso-level function, initial bounding cube, and maximum sub-divide level
Output: Result triangulated mesh
$CubeStack = \emptyset$; $Level = 0$; $Mesh = \emptyset$;
$InitialBoundingBox.Level = 0$;
$CubeStack$.push($InitialBoundingBox$);
repeat
    $CurrentCube = CubeStack$.pop();
    Compute iso-level on each endpoints of $CurrentCube$;
    if *(iso-level on endpoints are all positive)* or
      (iso-level on endpoints are all negative) **then**
        Continue;
    end

until *CubeStack is empty*;
Rnd_α = Random(ID_α, $index$);
$Found = TRUE$;
for *each node* β *in* α*'s two communication hops* **do**
    Rnd_β = Random(ID_β, $index$);
    if *(*$Rnd_\alpha < Rnd_\beta$*)* or *(*$Rnd_\alpha == Rnd_\beta$ and $ID_\alpha < ID_\beta$*);*
    then
        $Found = FALSE$; break;
    end
end
if $Found$ **then**
    $FreNum_\alpha = index$;
else
    $index$ ++;
end

3. PERFORMANCE

How our algorithms are doing.

3.1. Output Quality

3.2. Time Complexity

3.3. Space Complexity

4. DISCUSSION

Why some of the output are bad, and why they cannot be solved for now.

4.1. Variety of Input

explain why horse.smf cannot be used as an input
how the result can be

4.2. Iteration Methods of Marching Cubes

divide and conquer vs. normal iteration (pros and cons)
explain possible holes in out implementation

4.3. Size of Marching Cubes

5. FURTHER IMPROVEMENTS

What we have omitted. What can be done better

5.1. Spacial Sensitive Structure

How can our work be improved by using a better datastructure

5.2. Refinement of Output Mesh

The result mesh of our current implementation is composed of standalone triangles, i.e. Each vertex is included in only one triangle. This problem limits further processing of the output mesh. It can be solved by indexing all the vertices, and avoid generating a new vertex if one has existed at the same position.

5.3. Cube Ambiguity Problem

[Lorensen et al. 1987]

6. INTRODUCTION

As a new technology, Wireless Sensor Networks (WSNs) has a wide range of applications [**?**; **?**; **?**], including environment monitoring, smart buildings, medical care, industrial and military applications. Among them, a recent trend is to develop commercial sensor networks that require pervasive sensing of both environment and human beings, for example, assisted living [**?**; **?**; **?**] and smart homes [**?**; **?**; **?**].

> "For these applications, sensor devices are incorporated into human cloths [**?**; **?**; **?**; **?**] for monitoring health related information like EKG readings, fall detection, and voice recognition".

While collecting all these multimedia information [**?**] requires a high network throughput, off-the-shelf sensor devices only provide very limited bandwidth in a single channel: 19.2Kbps in MICA2 [**?**] and 250Kbps in MICAz.

In this article, we propose MMSN, abbreviation for Multifrequency Media access control for wireless Sensor Networks. The main contributions of this work can be summarized as follows.

— To the best of our knowledge, the MMSN protocol is the first multifrequency MAC protocol especially designed for WSNs, in which each device is equipped with a single radio transceiver and the MAC layer packet size is very small.
— Instead of using pairwise RTS/CTS frequency negotiation [**?**; **?**; **?**; **?**], we propose lightweight frequency assignments, which are good choices for many deployed comparatively static WSNs.
— We develop new toggle transmission and snooping techniques to enable a single radio transceiver in a sensor device to achieve scalable performance, avoiding the nonscalable "one control channel + multiple data channels" design [**?**].

7. MMSN PROTOCOL

7.1. Frequency Assignment

We propose a suboptimal distribution to be used by each node, which is easy to compute and does not depend on the number of competing nodes. A natural candidate is an increasing geometric sequence, in which

$$P(t) = \frac{b^{\frac{t+1}{T+1}} - b^{\frac{t}{T+1}}}{b-1}, \tag{1}$$

where $t = 0, \ldots, T$, and b is a number greater than 1.

In our algorithm, we use the suboptimal approach for simplicity and generality. We need to make the distribution of the selected back-off time slice at each node conform to what is shown in Equation (1). It is implemented as follows: First, a random variable α with a uniform distribution within the interval $(0, 1)$ is generated on each node, then time slice i is selected according to the following equation:

$$i = \lfloor (T+1) \log_b[\alpha(b-1)+1] \rfloor.$$

It can be easily proven that the distribution of i conforms to Equation (1).

So protocols [**?**; **?**; **?**; **?**; **?**; **?**; **?**] that use RTS/CTS controls[1] for frequency negotiation and reservation are not suitable for WSN applications, even though they exhibit good performance in general wireless ad hoc networks.

[1]RTS/CTS controls are required to be implemented by 802.11-compliant devices. They can be used as an optional mechanism to avoid Hidden Terminal Problems in the 802.11 standard and protocols based on those similar to **?**] and **?**].

ALGORITHM 2: Frequency Number Computation

Input: Node α's ID (ID_α), and node α's neighbors' IDs within two communication hops.
Output: The frequency number ($FreNum_\alpha$) node α gets assigned.

```
index = 0; FreNum_α = -1;
repeat
    Rnd_α = Random(ID_α, index);
    Found = TRUE;
    for each node β in α's two communication hops do
        Rnd_β = Random(ID_β, index);
        if (Rnd_α < Rnd_β) or (Rnd_α == Rnd_β and ID_α < ID_β);
        then
            Found = FALSE; break;
        end
    end
    if Found then
        FreNum_α = index;
    else
        index ++;
    end
until FreNum_α > −1;
```

7.1.1. Exclusive Frequency Assignment. In exclusive frequency assignment, nodes first exchange their IDs among two communication hops so that each node knows its two-hop neighbors' IDs. In the second broadcast, each node beacons all neighbors' IDs it has collected during the first broadcast period.

Eavesdropping. Even though the even selection scheme leads to even sharing of available frequencies among any two-hop neighborhood, it involves a number of two-hop broadcasts. To reduce the communication cost, we propose a lightweight eavesdropping scheme.

7.2. Basic Notations

As Algorithm 2 states, for each frequency number, each node calculates a random number (Rnd_α) for itself and a random number (Rnd_β) for each of its two-hop neighbors with the same pseudorandom number generator.

Bus masters are divided into two disjoint sets, $\mathcal{M}_{RT}$ and $\mathcal{M}_{NRT}$.

RT Masters. $\mathcal{M}_{RT} = \{\vec{m}_1, \ldots, \vec{m}_n\}$ denotes the n RT masters issuing real-time constrained requests. To model the current request issued by an $\vec{m}_i$ in $\mathcal{M}_{RT}$, three parameters—the recurrence time (r_i), the service cycle (c_i), and the relative deadline (d_i)—are used, with their relationships.

NRT Masters. $\mathcal{M}_{NRT} = \{\vec{m}_{n+1}, \ldots, \vec{m}_{n+m}\}$ is a set of m masters issuing nonreal-time constrained requests. In our model, each $\vec{m}_j$ in $\mathcal{M}_{NRT}$ needs only one parameter, the service cycle, to model the current request it issues.

Here, a question may arise, since each node has a global ID. Why don't we just map nodes' IDs within two hops into a group of frequency numbers and assign those numbers to all nodes within two hops?

8. SIMULATOR

If the model checker requests successors of a state which are not created yet, the state space uses the simulator to create the successors on-the-fly. To create successor states the simulator conducts the following steps.

(1) Load state into microcontroller model.

Fig. 1. Code before preprocessing.

(2) Determine assignments needed for resolving nondeterminism.
(3) For each assignment.
    (a) either call interrupt handler or simulate effect of next instruction, or
    (b) evaluate truth values of atomic propositions.
(4) Return resulting states.

Figure 1 shows a typical microcontroller C program that controls an automotive power window lift. The program is one of the programs used in the case study described in Section 8. At first sight, the programs looks like an ANSI C program. It contains function calls, assignments, if clauses, and while loops.

8.1. Problem Formulation

The objective of variable coalescence-based offset assignment is to find both the coalescence scheme and the MWPC on the coalesced graph. We start with a few definitions and lemmas for variable coalescence.

Definition 8.1 (*Coalesced Node (C-Node)*). A C-node is a set of live ranges (webs) in the AG or IG that are coalesced. Nodes within the same C-node cannot interfere with each other on the IG. Before any coalescing is done, each live range is a C-node by itself.

Definition 8.2 (*C-AG (Coalesced Access Graph)*). The C-AG is the access graph after node coalescence, which is composed of all C-nodes and C-edges.

Lemma 8.3. *The C-MWPC problem is NP-complete.*

Proof. C-MWPC can be easily reduced to the MWPC problem assuming a coalescence graph without any edge or a fully connected interference graph. Therefore, each C-node is an uncoalesced live range after value separation and C-PC is equivalent to PC. A fully connected interference graph is made possible when all live ranges interfere with each other. Thus, the C-MWPC problem is NP-complete. □

Lemma 8.4 (Lemma Subhead). *The solution to the C-MWPC problem is no worse than the solution to the MWPC.*

Proof. Simply, any solution to the MWPC is also a solution to the C-MWPC. But some solutions to C-MWPC may not apply to the MWPC (if any coalescing were made). □

9. PERFORMANCE EVALUATION

During all the experiments, the Geographic Forwarding (GF) [**?**] routing protocol is used. GF exploits geographic information of nodes and conducts local data-forwarding to achieve end-to-end routing. Our simulation is configured according to the settings in Table I. Each run lasts for 2 minutes and repeated 100 times. For each data value we present in the results, we also give its 90% confidence interval.

10. CONCLUSIONS

Conclusions goes here.

REFERENCES

Lorensen, W., and Cline, H. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, 21(4):163-9, 1987.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. 1992. Surface reconstruction from unorganized points. Computer Graphics (SIGGRAPH 92 Proceedings), 26(2):7178, July 1992.

Table I. Simulation Configuration

TERRAIN[a]	(200m×200m) Square
Node Number	289
Node Placement	Uniform
Application	Many-to-Many/Gossip CBR Streams
Payload Size	32 bytes
Routing Layer	GF
MAC Layer	CSMA/MMSN
Radio Layer	RADIO-ACCNOISE
Radio Bandwidth	250Kbps
Radio Range	20m–45m

Source: This is a table sourcenote. This is a table sourcenote. This is a table sourcenote.

Note: This is a table footnote.
[a]This is a table footnote. This is a table footnote. This is a table footnote.