A Distributed Triangulation Algorithm for Wireless Sensor Networks on 2D and 3D Surface

Hongyu Zhou, Hongyi Wu, Su Xia, Miao Jin, Ning Ding

The Center for Advanced Computer Studies(CACS)
University of Louisiana at Lafayette

LOUISIANA
Lafayette

Outline

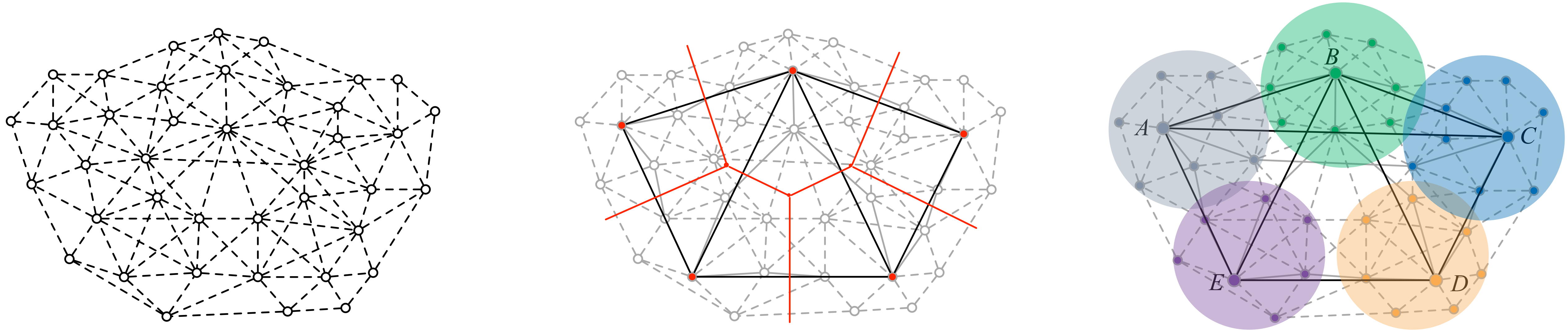
- Introduction
- Triangulation Algorithm
- Extension to 3D Surface
- Application and Simulation Results
- Conclusion

INTRODUCTION

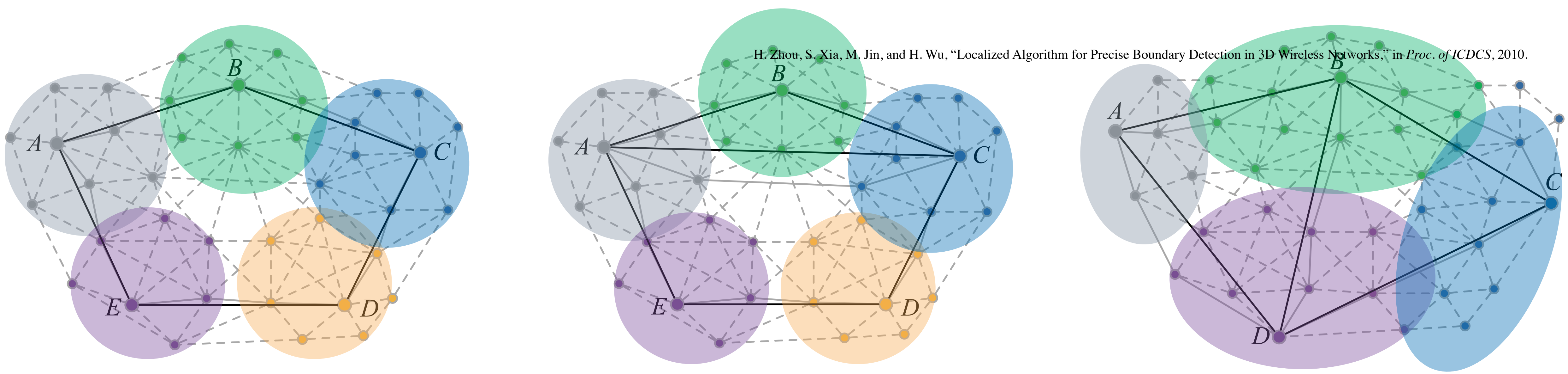
- Wireless sensor network exhibits randomness
- Wireless sensor network topology is a graph
- Triangulation is a subgraph of the topology
- Triangulation mesh is very important for many applications of sensor networks:
  geometry-based routing, localization, coverage, segmentation, data segmentation

Introduction

- Related work



(a) A 2D network graph. (b) Triangulation under ideal CDG. (c) CDG (k=1): not planar.



(d) CDM (k=1): planar but not triangulated. (e) Failed triangulation. (f) CDG (k=2): triangulated but coarse.

[2] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu, "Greedy Routing with Guaranteed Delivery Using Ricci Flows," in *Proc. of IPSN*, 2009.

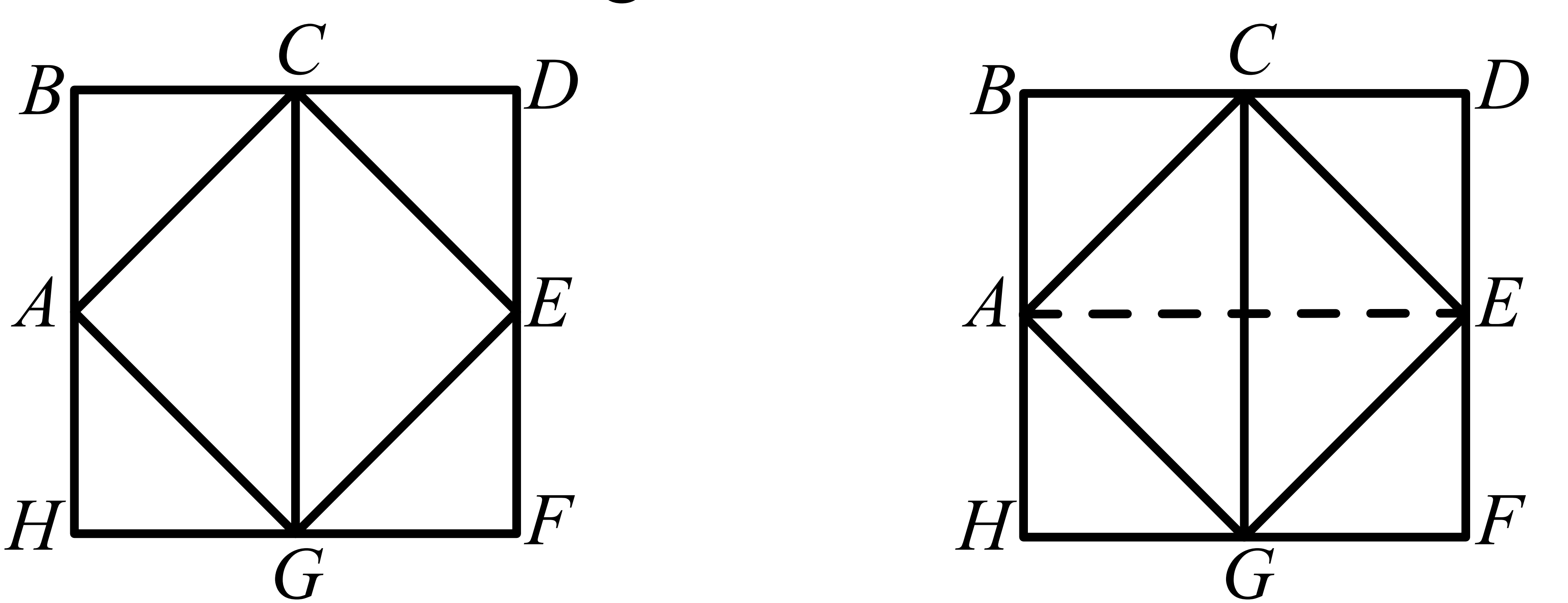
[3] H. Zhou, S. Xia, M. Jin, and H. Wu, "Localized Algorithm for Precise Boundary Detection in 3D Wireless Networks," in Proc. of ICDCS, 2010.

INTRODUCTION

- We proposed a distributed triangulation algorithm
  - works for any arbitrary 2D sensor networks without communication model constraint, and we prove the correctness of the algorithm in 2D sensor networks
  - tolerates some measurement errors
  - also works for 3D open or closed surfaces

Triangulation Algorithm

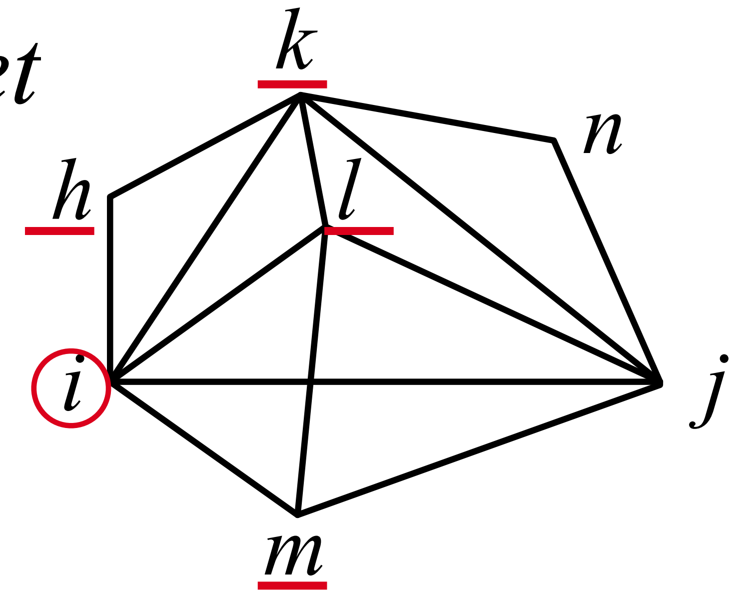
- Basic idea
  - Each triangulation mesh edge will associate with **two** triangles, except boundary edge
  - If we add extra edges into triangulation mesh, they will change this association
  - The association can be used to identify these extra edges.



LOUISIANA Lafayette

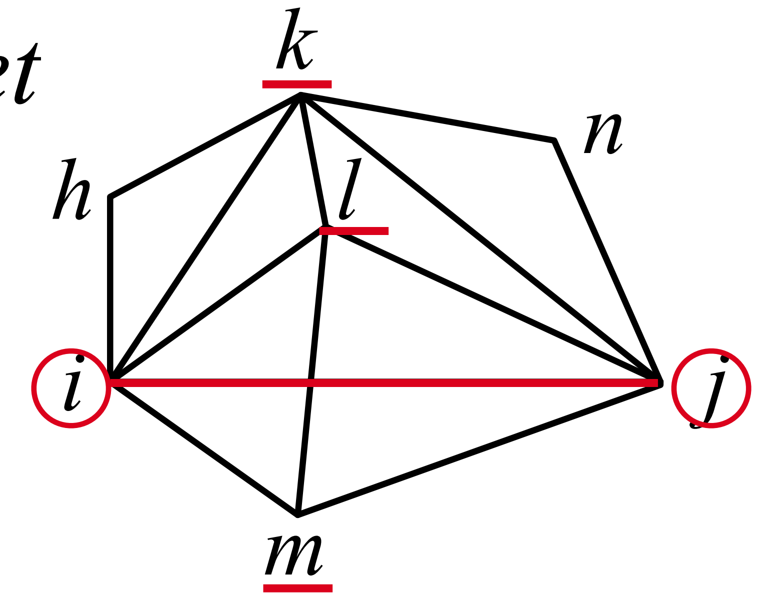
Triangulation Algorithm

- Definitions
  - ***node neighbor set* (NNS),** $N_v(i)=\{h,k,l,m\}$
  - *edge neighbor set* (ENS), $N_e(e_{ij})=N_v(i)\cap N_v(j)=\{k,l,m\}$
  - *refined edge neighbor set* (RNS), $R_e(e_{ij})=\{l,m\}$
  - *edge weight*, $W(e_{ij})=2$, $W(e_{hk})=2$
  - *associated edge neighbor set* (AENS), $A(e_{ij})=\{e_{il}, e_{im}, e_{jl}, e_{jm}\}$
  - *equivalent edges*, e_{ij}, e_{lm}
  - *critical edge*, e_{kj}



TRIANGULATION ALGORITHM

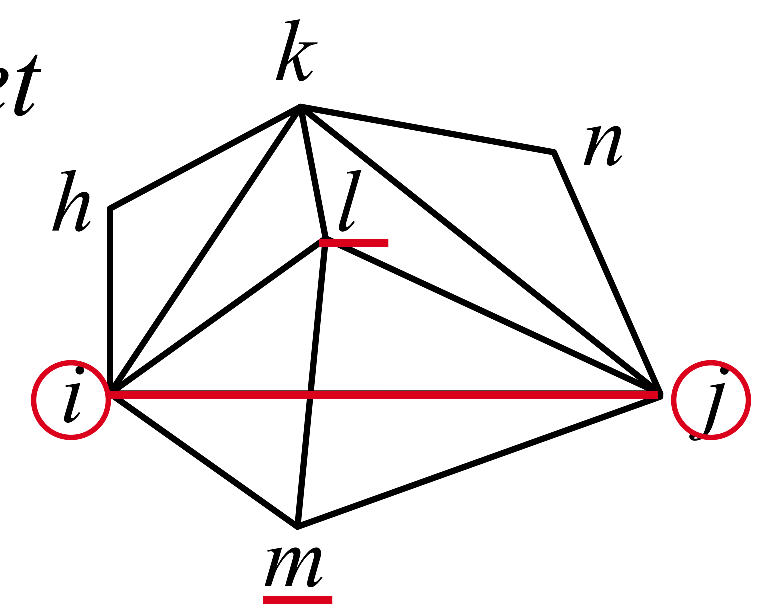
- Definitions
  - *node neighbor set* (NNS), $N_v(i)=\{h,k,l,m\}$
  - ***edge neighbor set* (ENS),** $\boldsymbol{N_e(e_{ij})=N_v(i)\cap N_v(j)=\{k,l,m\}}$
  - *refined edge neighbor set* (RNS), $R_e(e_{ij})=\{l,m\}$
  - *edge weight*, $W(e_{ij})=2$, $W(e_{hk})=2$
  - *associated edge neighbor set* (AENS), $A(e_{ij})=\{e_{il}, e_{im}, e_{jl}, e_{jm}\}$
  - *equivalent edges*, e_{ij}, e_{lm}
  - *critical edge*, e_{kj}



LOUISIANA
Lafayette

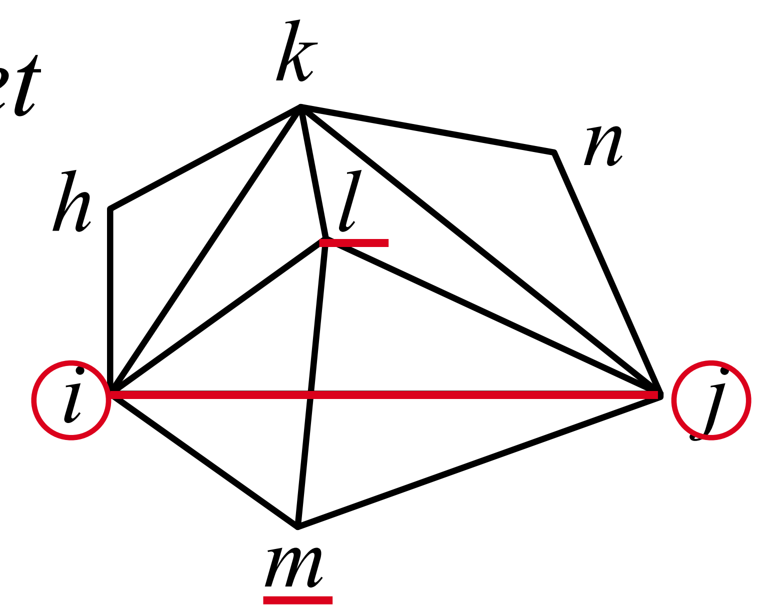
Triangulation Algorithm

- Definitions
  - *node neighbor set* (NNS), $N_v(i)=\{h,k,l,m\}$
  - *edge neighbor set* (ENS), $N_e(e_{ij})=N_v(i)\cap N_v(j)=\{k,l,m\}$
  - ***refined edge neighbor set* (RNS),** $\boldsymbol{R_e(e_{ij})=\{l,m\}}$
  - *edge weight*, $W(e_{ij})=2$, $W(e_{hk})=2$
  - *associated edge neighbor set* (AENS), $A(e_{ij})=\{e_{il}, e_{im}, e_{jl}, e_{jm}\}$
  - *equivalent edges*, e_{ij}, e_{lm}
  - *critical edge*, e_{kj}



TRIANGULATION ALGORITHM

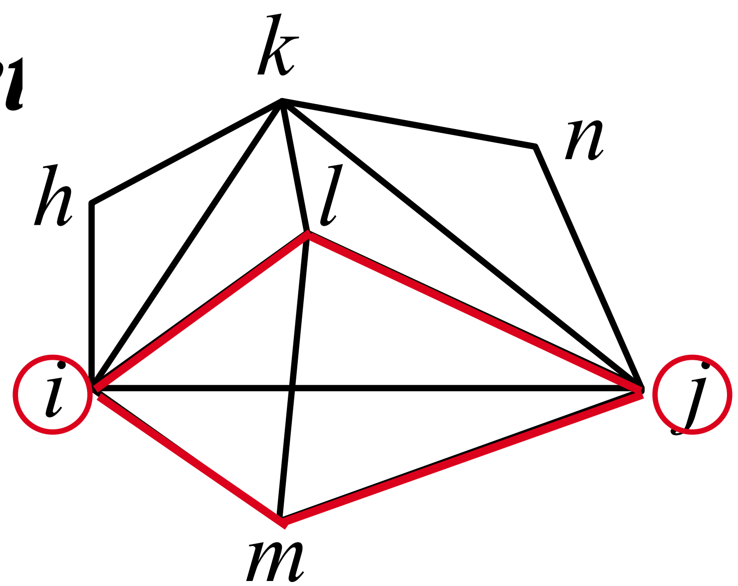
- Definitions
  - *node neighbor set* (NNS), $N_v(i)=\{h,k,l,m\}$
  - *edge neighbor set* (ENS), $N_e(e_{ij})=N_v(i)\cap N_v(j)=\{k,l,m\}$
  - *refined edge neighbor set* (RNS), $R_e(e_{ij})=\{l,m\}$
  - ***edge weight***, $\boldsymbol{W(e_{ij})=2}$, $\boldsymbol{W(e_{hk})=2}$
  - *associated edge neighbor set* (AENS), $A(e_{ij})=\{e_{il}, e_{im}, e_{jl}, e_{jm}\}$
  - *equivalent edges*, e_{ij}, e_{lm}
  - *critical edge*, e_{kj}



LOUISIANA
Lafayette

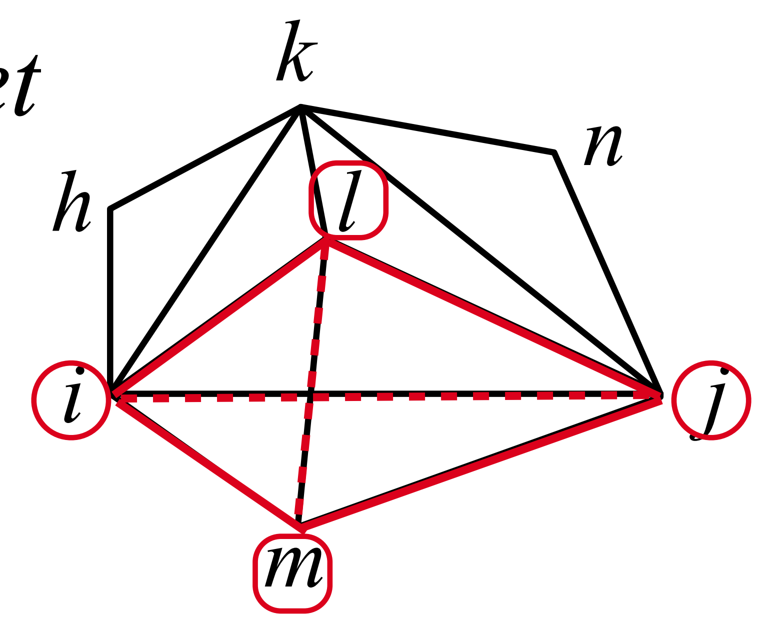
TRIANGULATION ALGORITHM

- Definitions
  - *node neighbor set* (NNS), $N_v(i)=\{h,k,l,m\}$
  - *edge neighbor set* (ENS), $N_e(e_{ij})=N_v(i)\cap N_v(j)=\{k,l,m\}$
  - *refined edge neighbor set* (RNS), $R_e(e_{ij})=\{l,m\}$
  - *edge weight*, $W(e_{ij})=2$, $W(e_{hk})=2$
  - ***associated edge neighbor set* (AENS),** $\mathbf{A(e_{ij})=\{e_{il}, e_{im}, e_{jl}, e_{jm}\}}$
  - *equivalent edges*, e_{ij}, e_{lm}
  - *critical edge*, e_{kj}



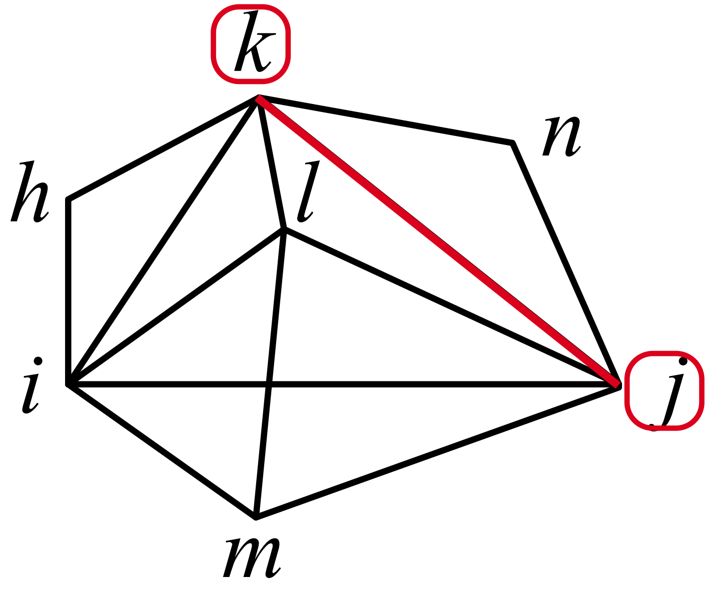
TRIANGULATION ALGORITHM

- Definitions
  - *node neighbor set* (NNS), $N_v(i)=\{h,k,l,m\}$
  - *edge neighbor set* (ENS), $N_e(e_{ij})=N_v(i)\cap N_v(j)=\{k,l,m\}$
  - *refined edge neighbor set* (RNS), $R_e(e_{ij})=\{l,m\}$
  - *edge weight*, $W(e_{ij})=2$, $W(e_{hk})=2$
  - *associated edge neighbor set* (AENS), $A(e_{ij})=\{e_{il}, e_{im}, e_{jl}, e_{jm}\}$
  - ***equivalent edges***, e_{ij}, e_{lm}
  - *critical edge*, e_{kj}



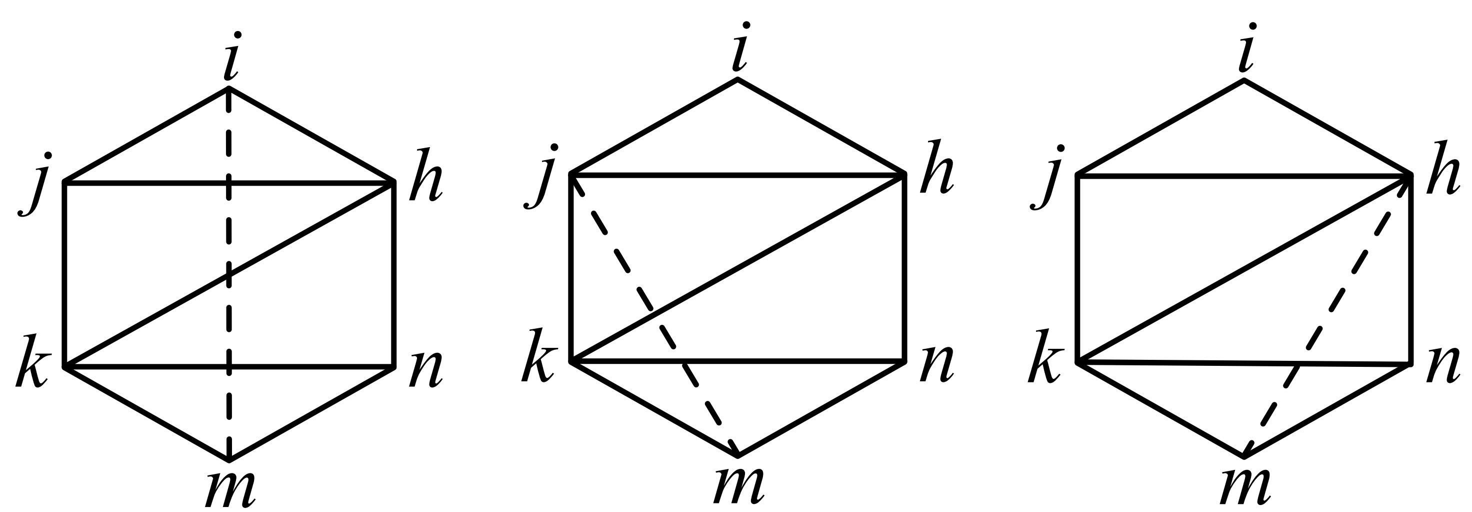
Triangulation Algorithm

- Definitions
  - *node neighbor set* (NNS), $N_v(i)=\{h,k,l,m\}$
  - *edge neighbor set* (ENS), $N_e(e_{ij})=N_v(i)\cap N_v(j)=\{k,l,m\}$
  - *refined edge neighbor set* (RNS), $R_e(e_{ij})=\{l,m\}$
  - *edge weight*, $W(e_{ij})=2$, $W(e_{hk})=2$
  - *associated edge neighbor set* (AENS), $A(e_{ij})=\{e_{il}, e_{im}, e_{jl}, e_{jm}\}$
  - *equivalent edges*, e_{ij}, e_{lm}
  - ***critical edge***, e_{kj}



TRIANGULATION ALGORITHM

- **Lemma 1**: A subgraph of ***G*** is triangulated *if and only if* every edge of the subgraph has a weight of **two**.
- Given a graph G and a *triangulation subgraph T, extra edges are G-T*
- **Objective:** remove all extra edges
- There are three type of *extra edges, e_0, e_1, e_2*

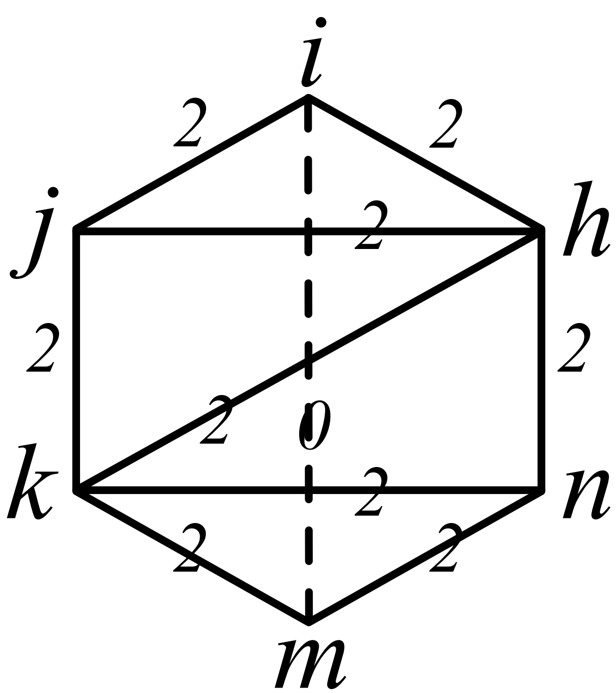


(a) e_0 (b) e_1 (c) e_2

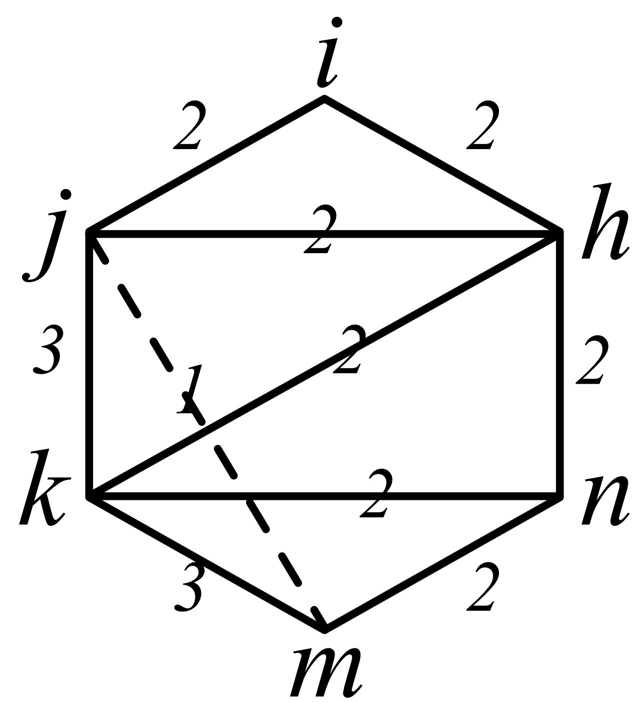
LOUISIANA
Lafayette

TRIANGULATION ALGORITHM

- **Theorem 1**: An edge with a weight *less than* **two** must be removed in order to produce a triangulated subgraph.



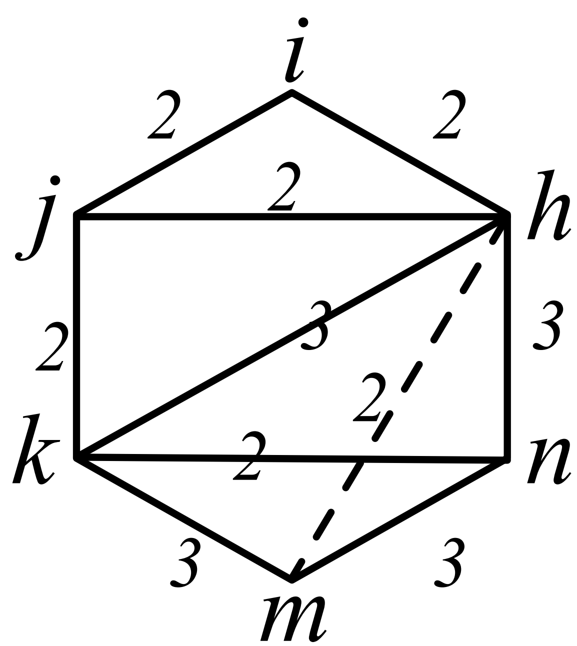
(a) e_0



(b) e_1

TRIANGULATION ALGORITHM

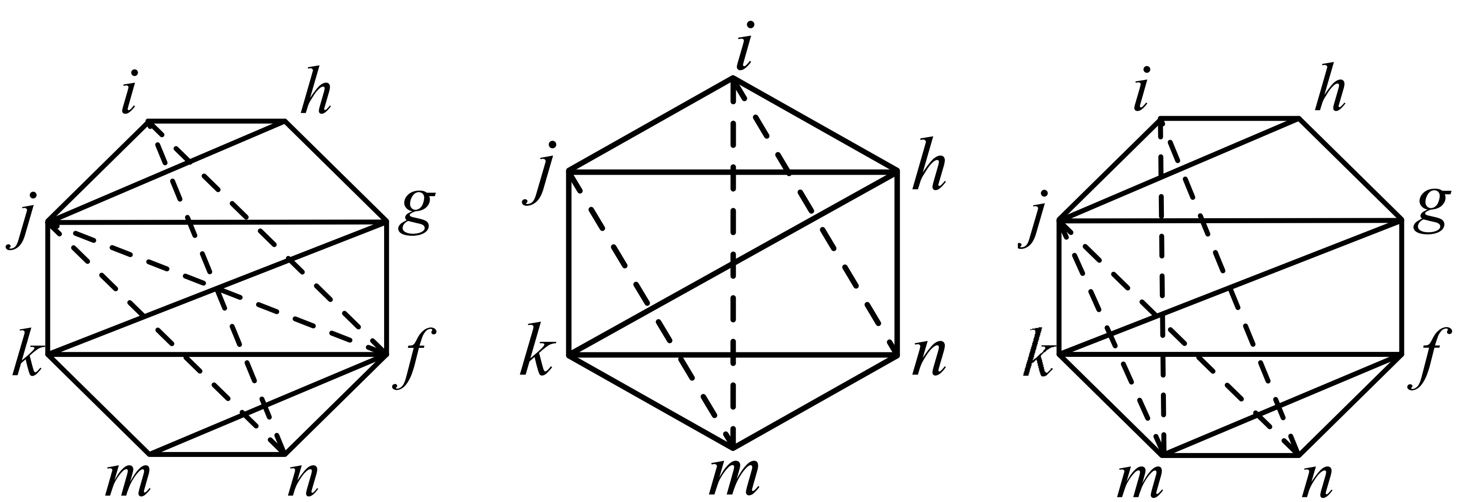
- **Theorem 2**:An *non-critical edge* can be recognized as an e_2 edge and safely removed if all edges in its *AENS* have their weight greater than **two**.



(c) e_2

TRIANGULATION ALGORITHM

- **Lemma 2**: An *e_0 extra edge* can exist in G'', only if it depends on at least two other *extra edges*.
- **Lemma 3**: An e_1 extra edge can exist in G'', only if it at least depends on another extra edge.



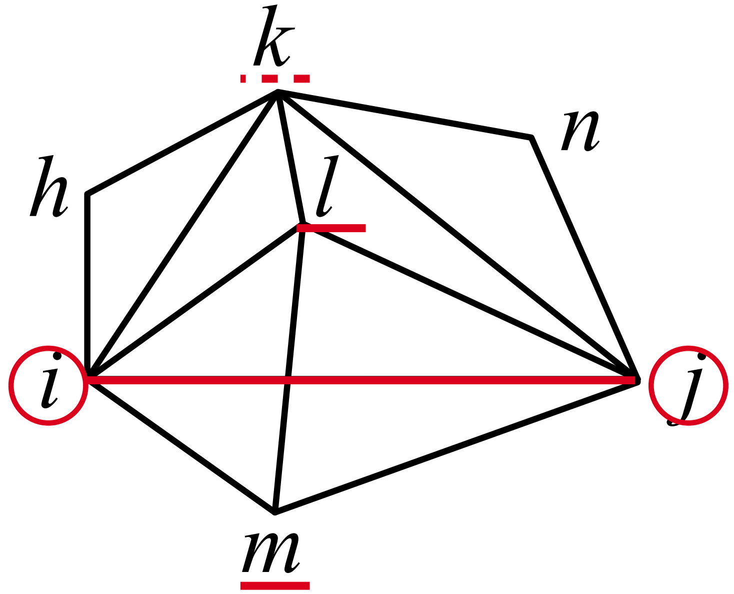
(a) Loop chain. (b) Non-loop chain. (c) Same head and tail.

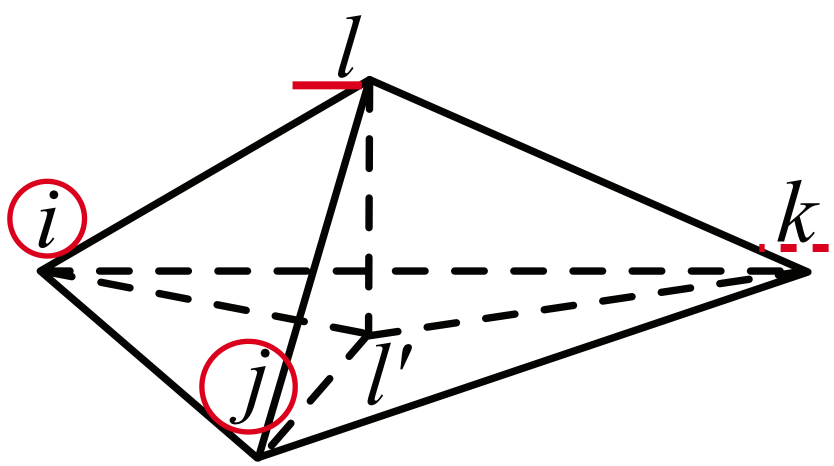
TRIANGULATION ALGORITHM

- Step1: Initialization.
  each edge calculate *NNS, ENS, RENS, AENS*, and *edge weight*;
- Step2: Iterative edge removal according to **Theorems** 1 and 2;
- Step3: Removal of e_0 and e_1 chains.

EXTENSION TO 3D SURFACE

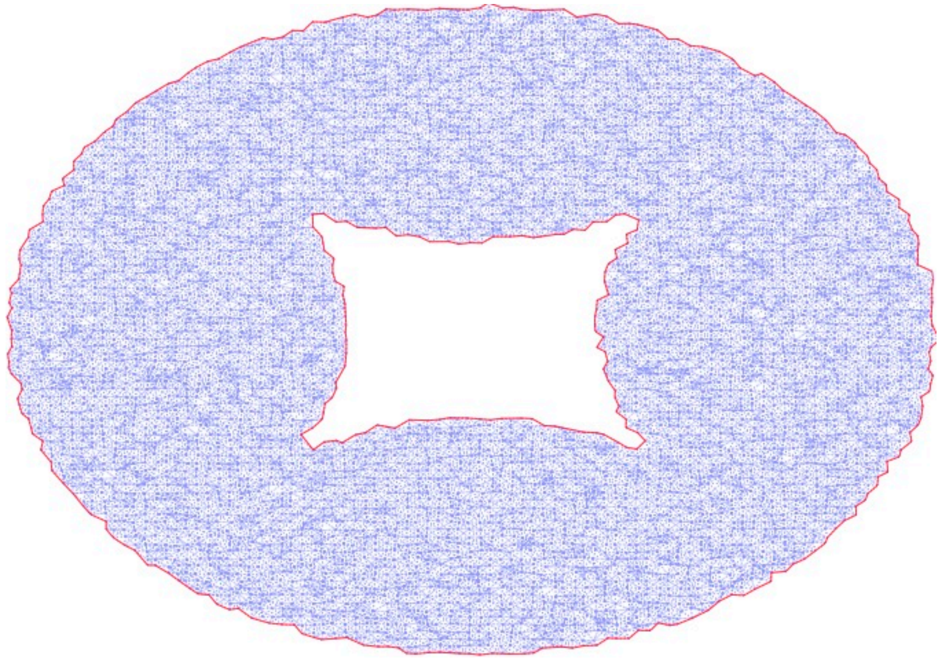
- Similar algorithm can be applied in a sensor network on 3D surface with minor modification in determining *RNS* and *edge weight*



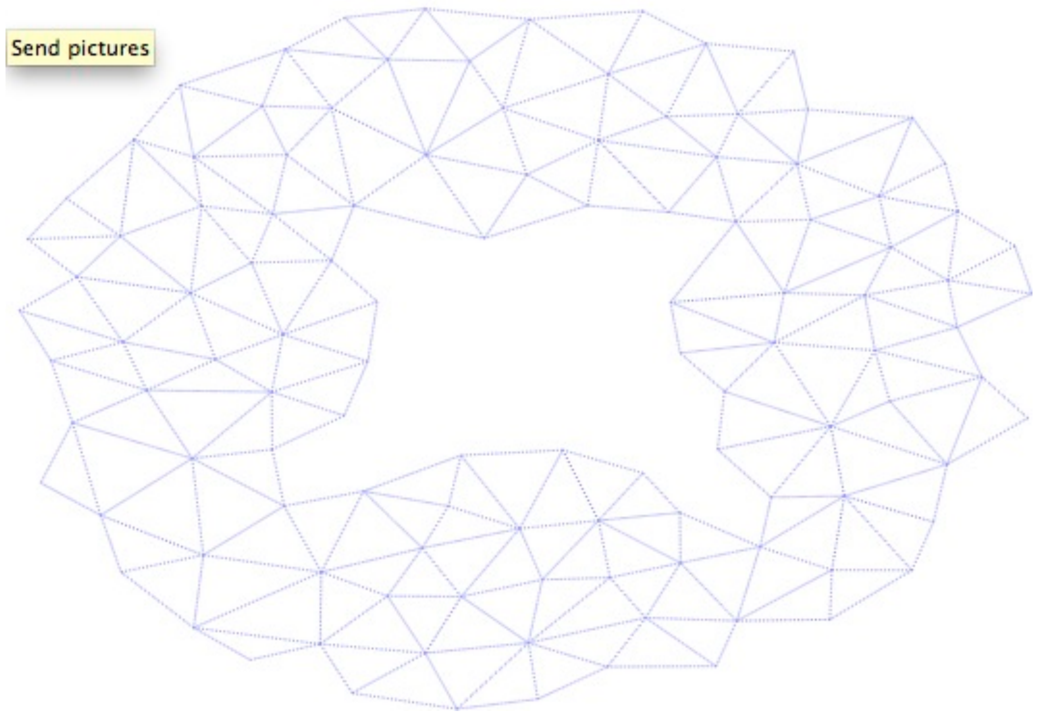


APPLICATION AND SIMULATION RESULTS

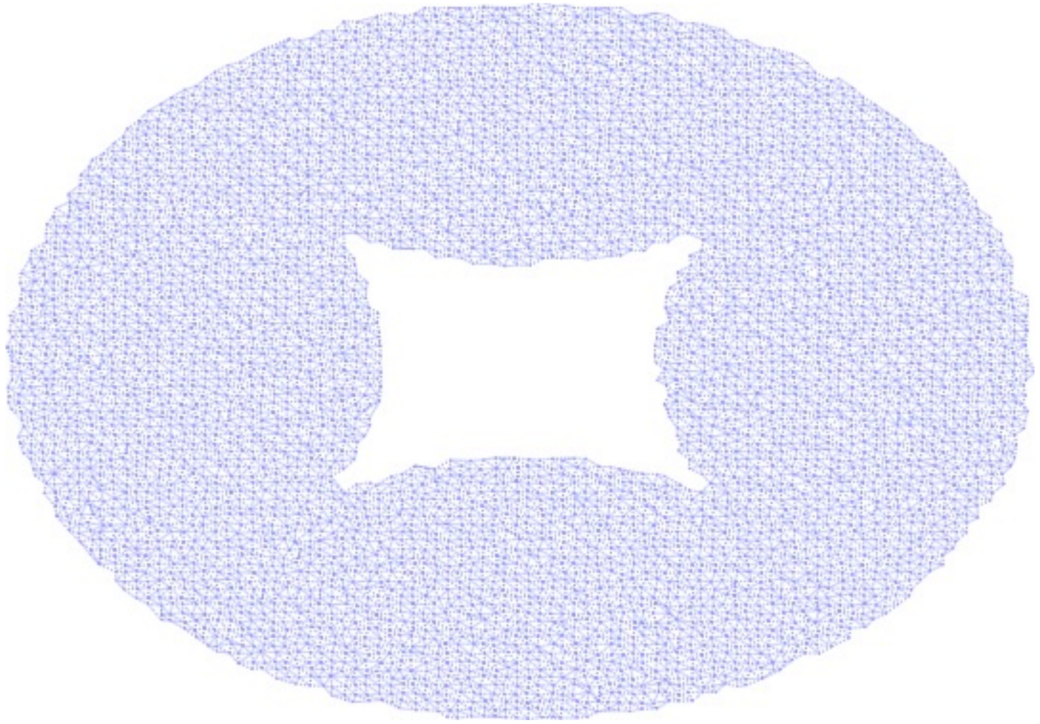
2D network



(a) Sensor network topology



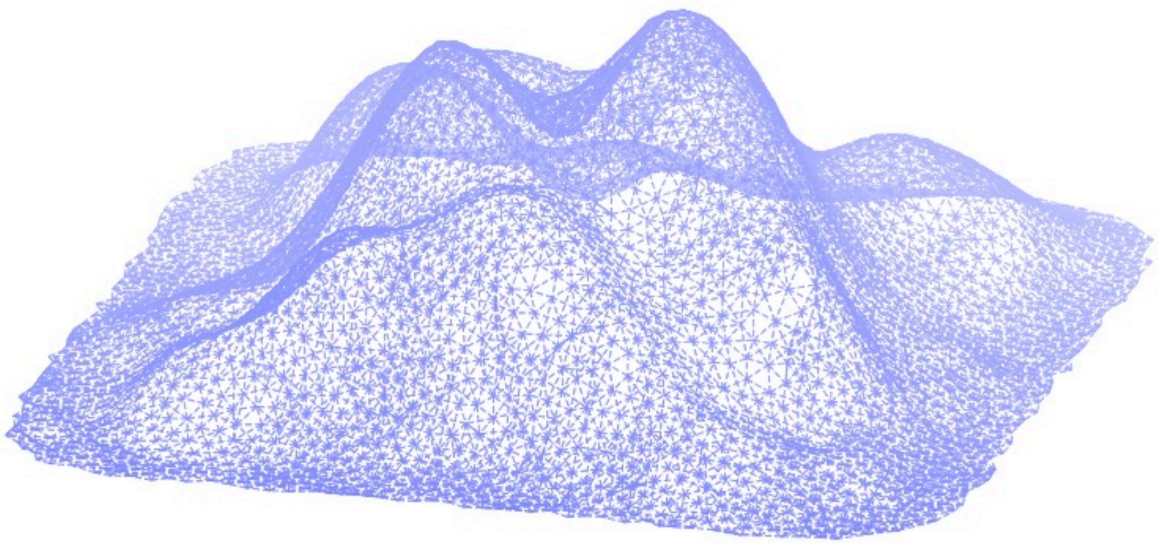
(b) Finest triangulation by [2]



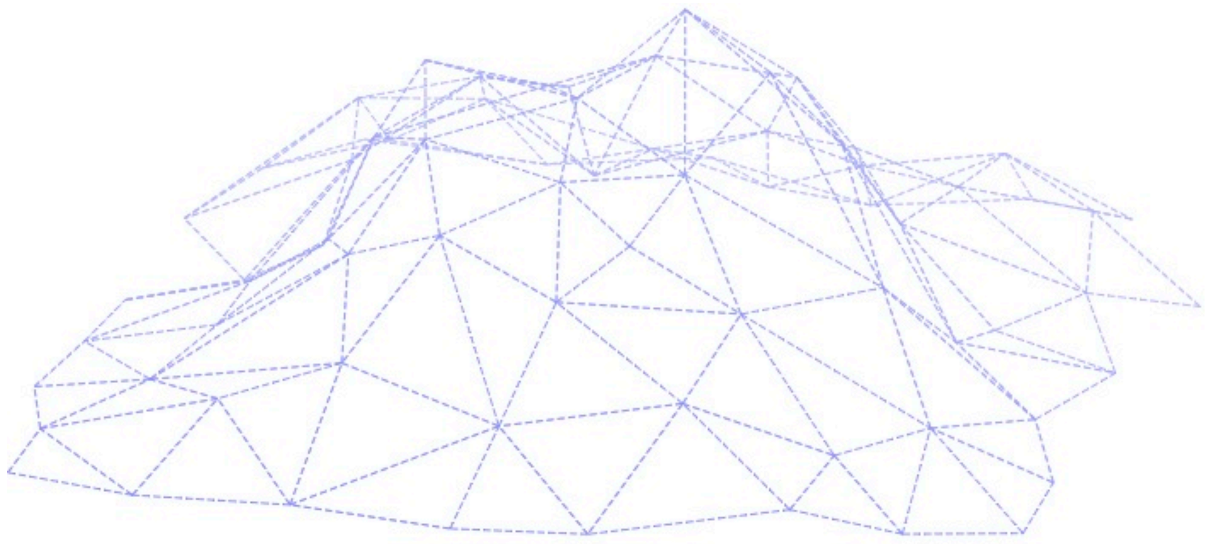
(c) Proposed triangulation

Application and Simulation Results

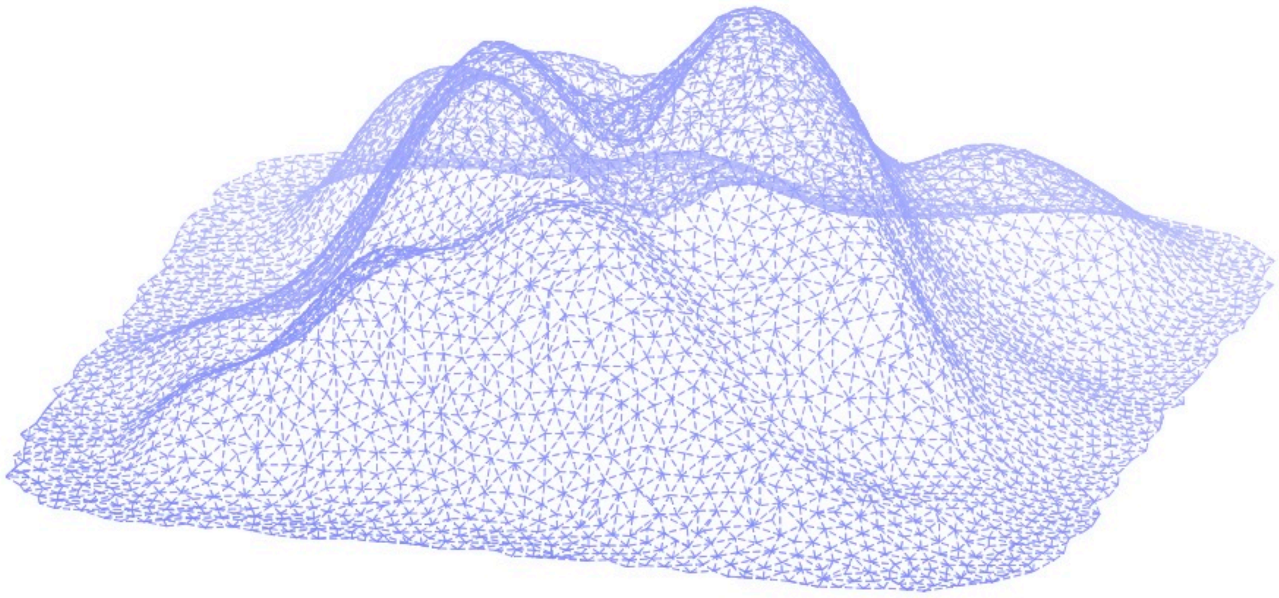
3D network with open surface



(a) Sensor network topology



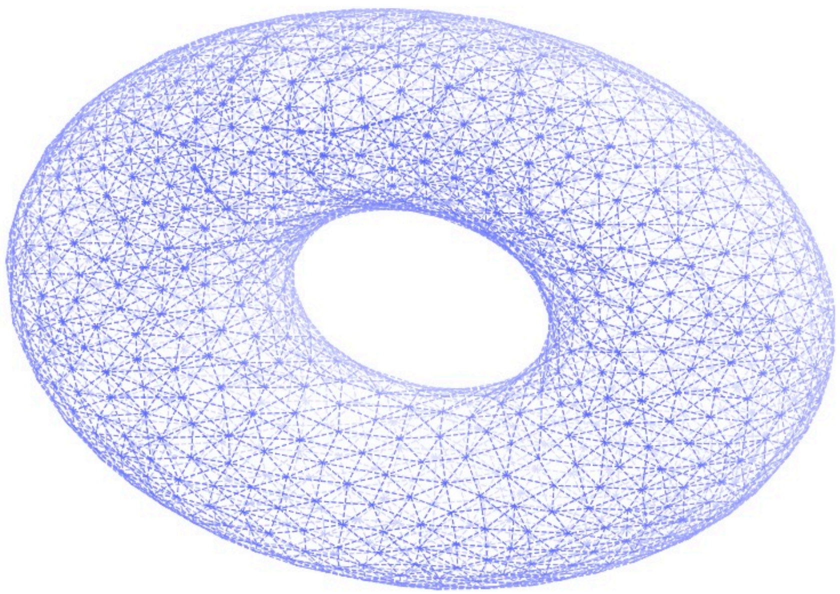
(b) Finest triangulation by [3]



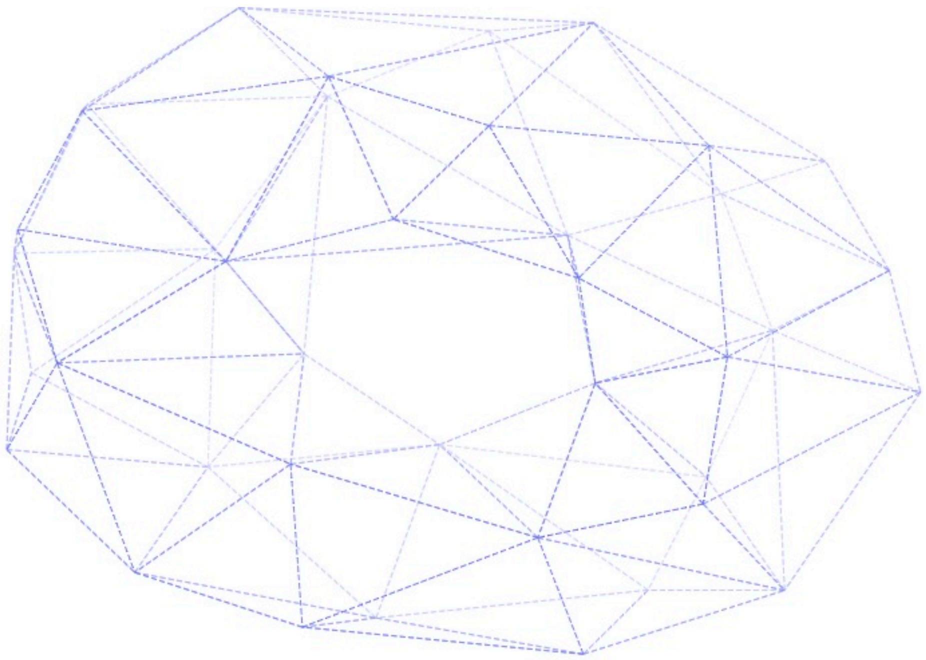
(c) Proposed triangulation

APPLICATION AND SIMULATION RESULTS

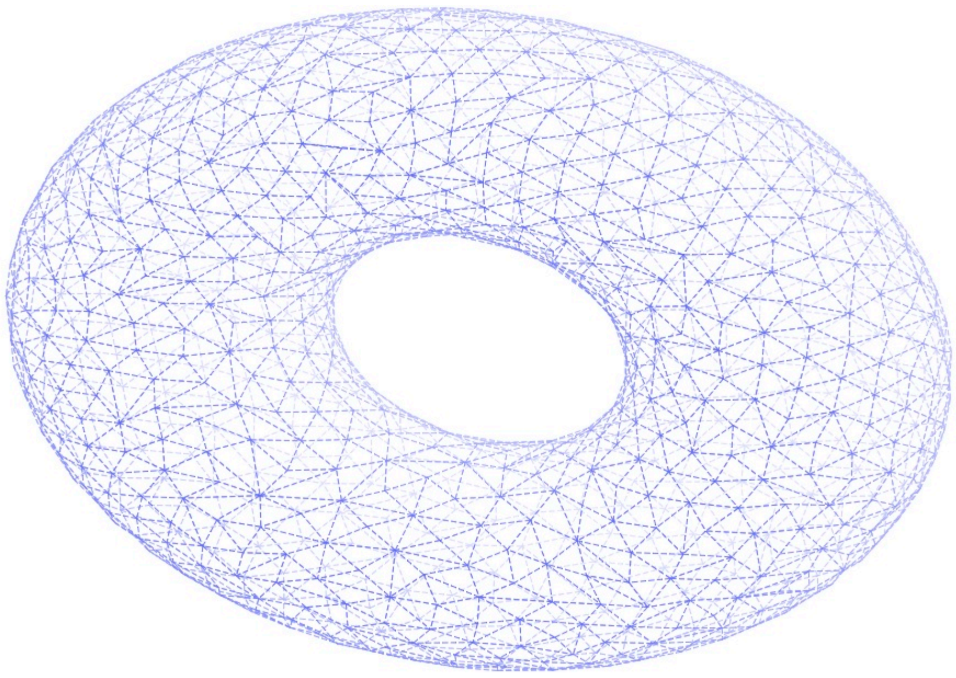
3D network with
closed surface



(a) Sensor network topology



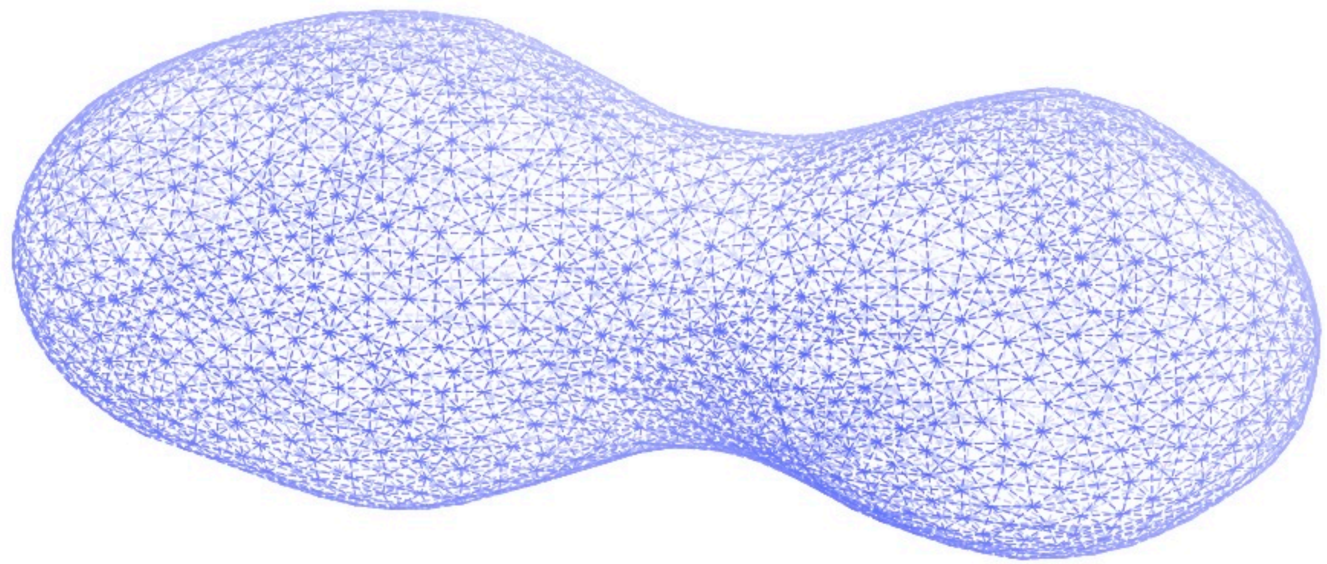
(b) Finest triangulation by [3]



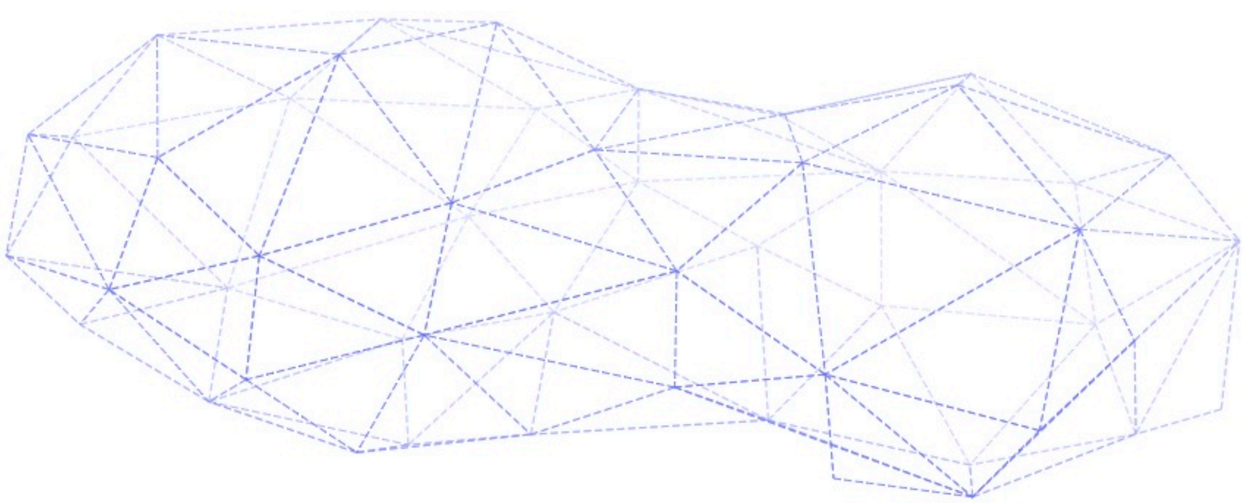
(c) Proposed triangulation

Application and Simulation Results

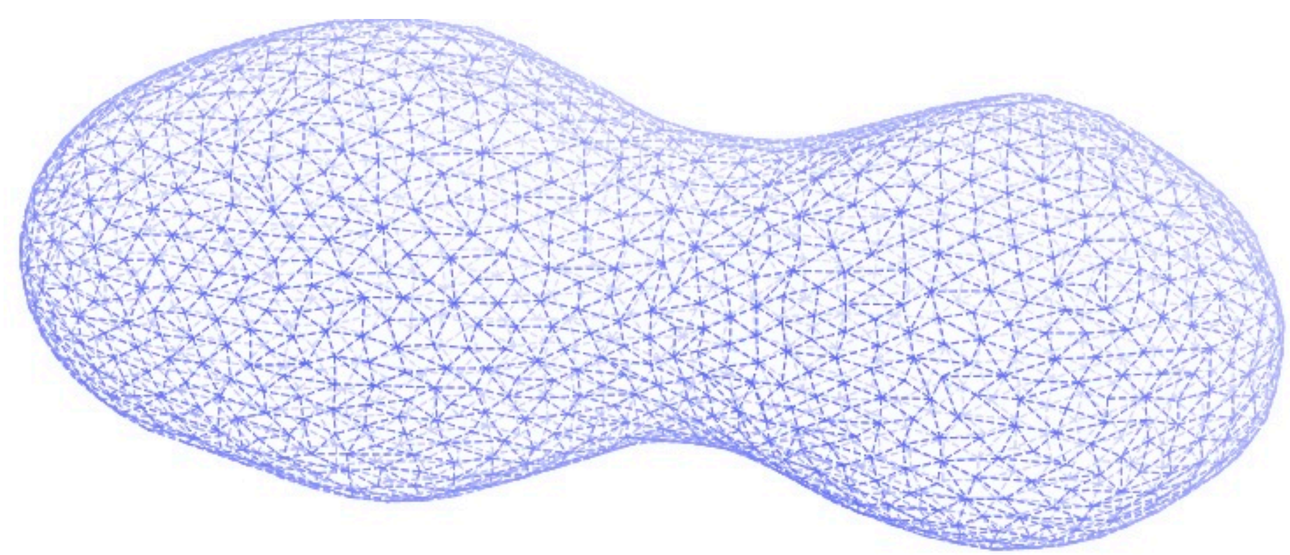
3D network with
closed surface



(a) Sensor network topology



(b) Finest triangulation by [3]

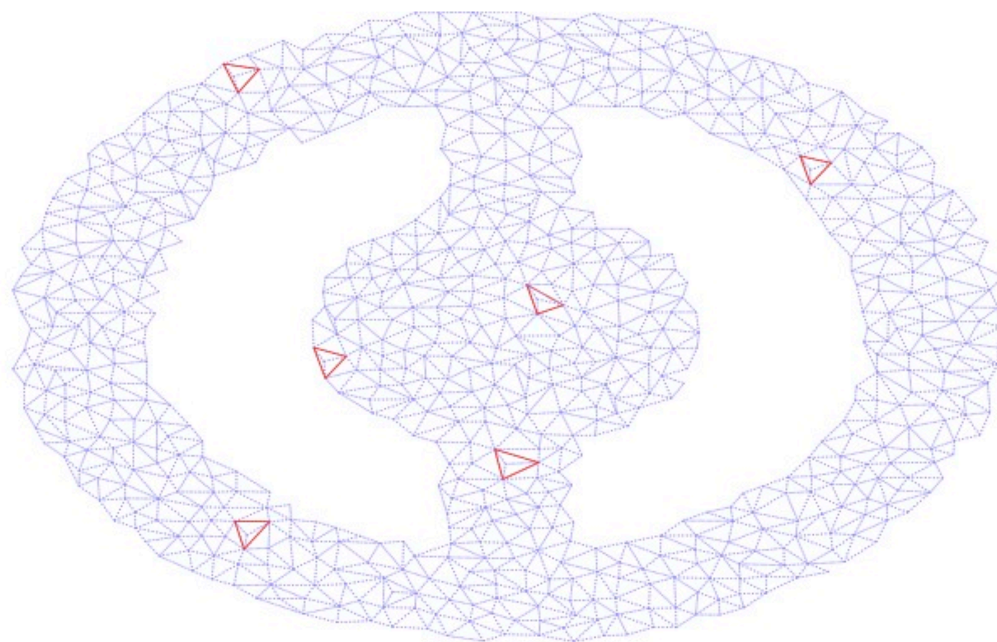


(c) Proposed triangulation

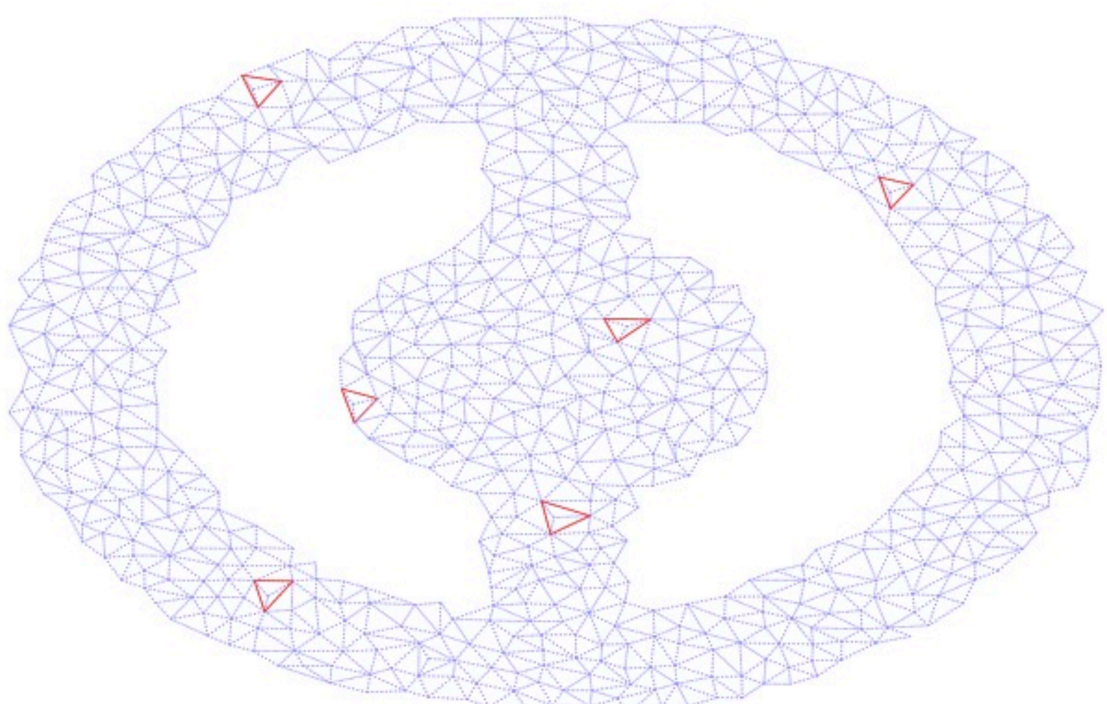
LOUISIANA
Lafayette

APPLICATION AND SIMULATION RESULTS

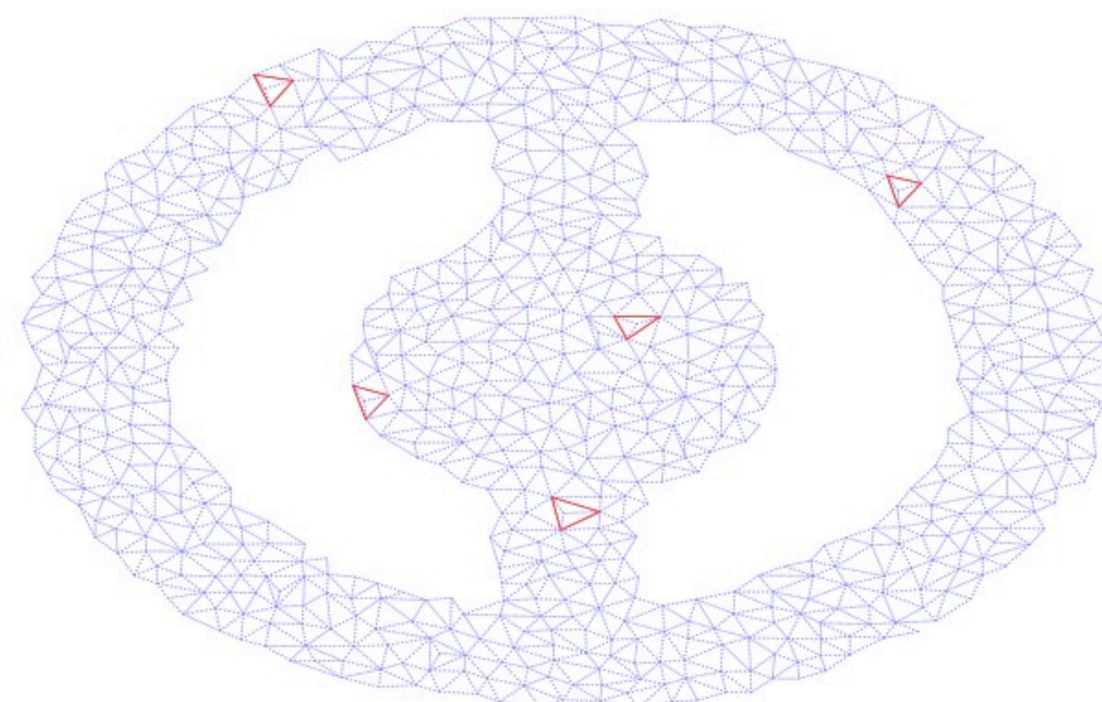
2D network with distance errors



(a) Triangulation under 10% distance errors.



(b) Triangulation under 20% distance errors.

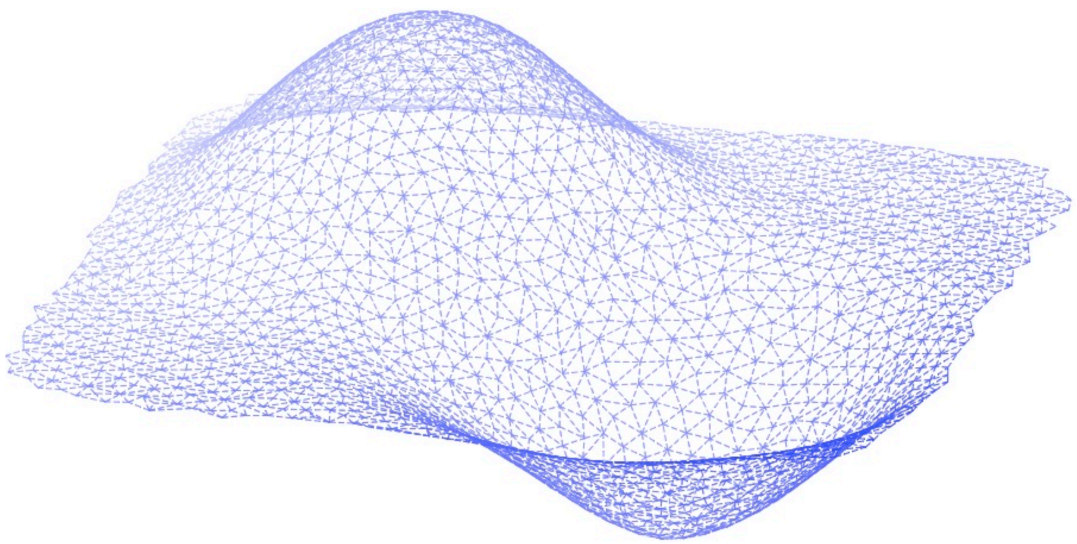


(c) Triangulation under 30% distance errors.

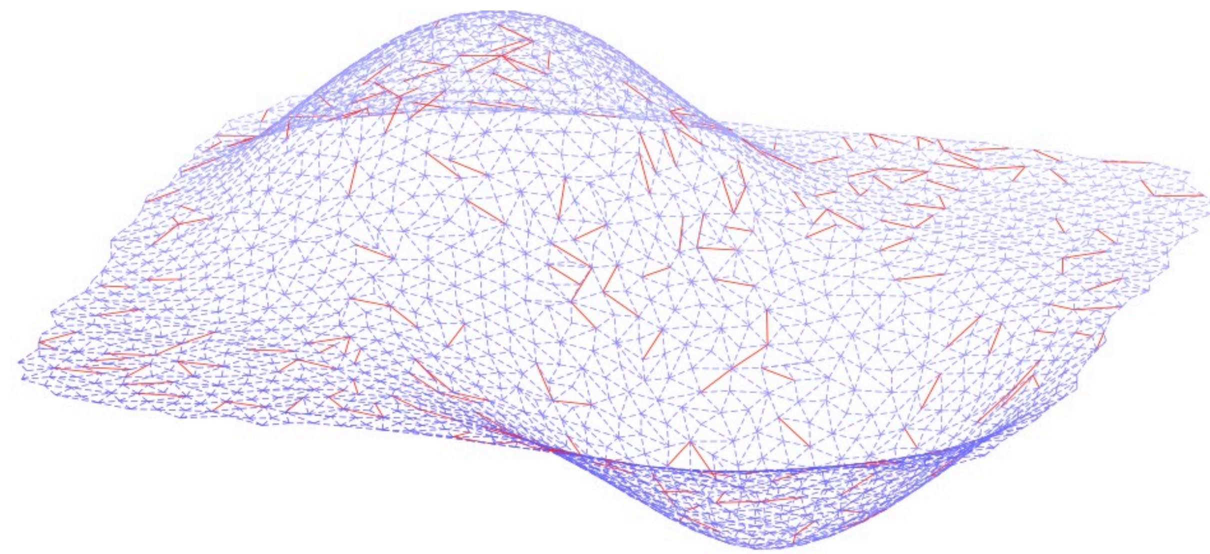
LOUISIANA Lafayette

Application and Simulation Results

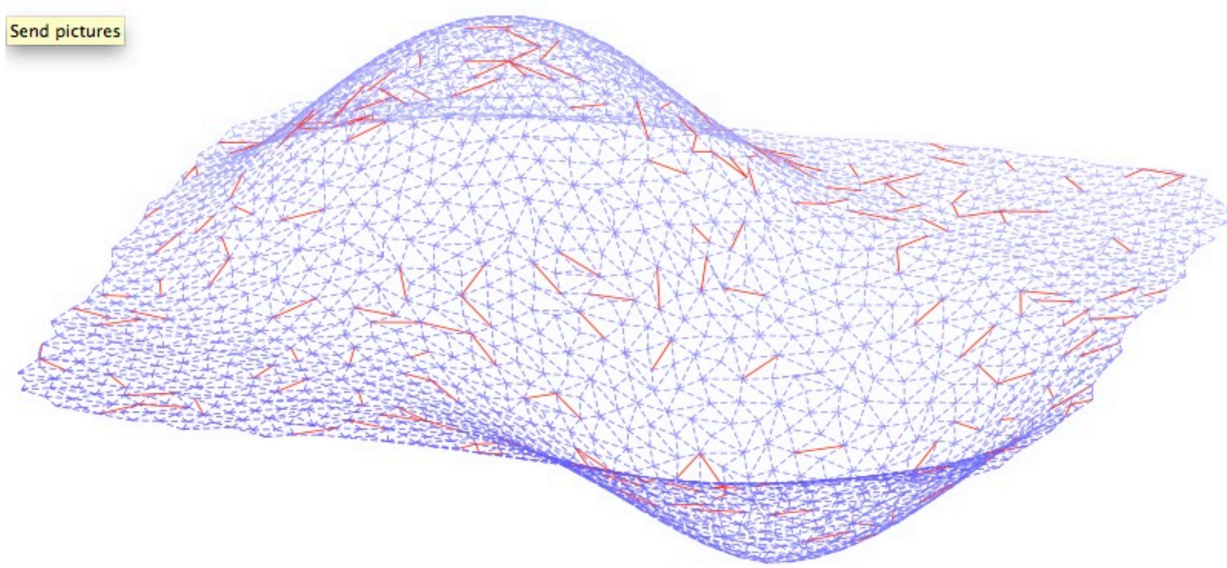
Difference Communication Models



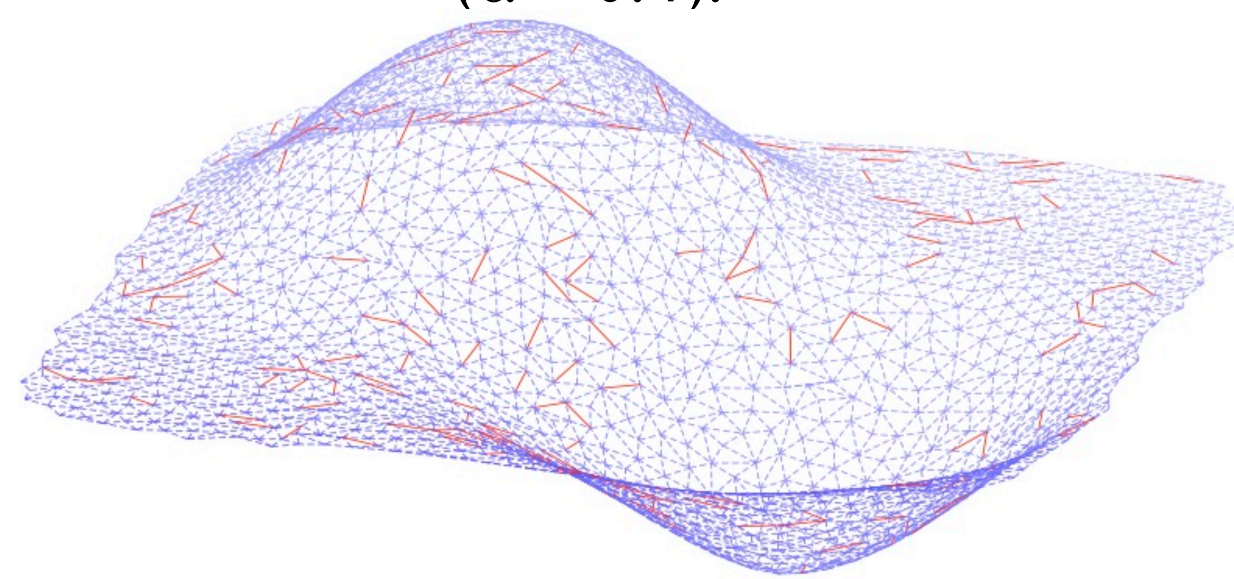
(a) UDG



(b) Triangulation under Quasi-UDG ($\alpha = 0.4$).



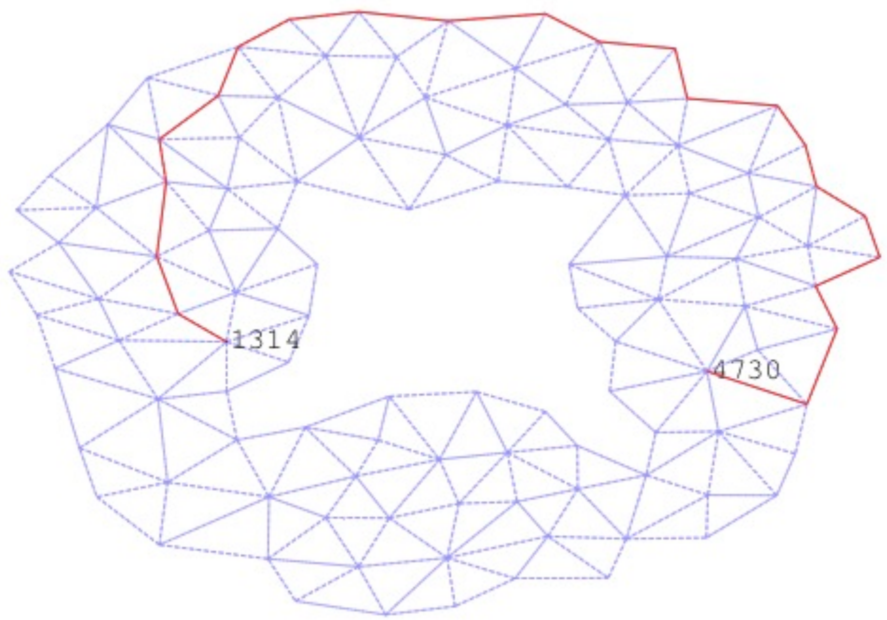
(c) Triangulation under Quasi-UDG ($\alpha = 0.6$)



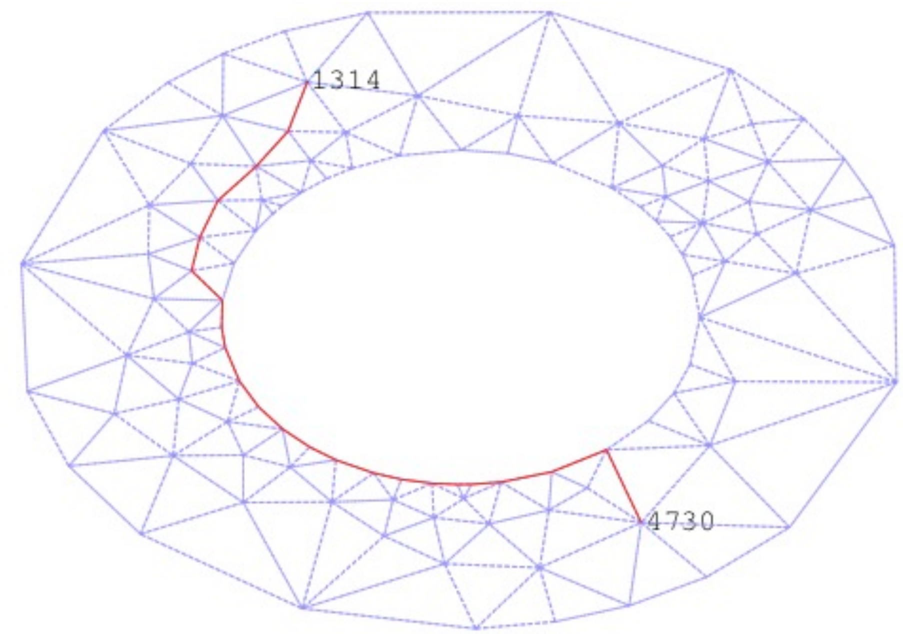
(d) Triangulation under Log-normal.

APPLICATION AND SIMULATION RESULTS

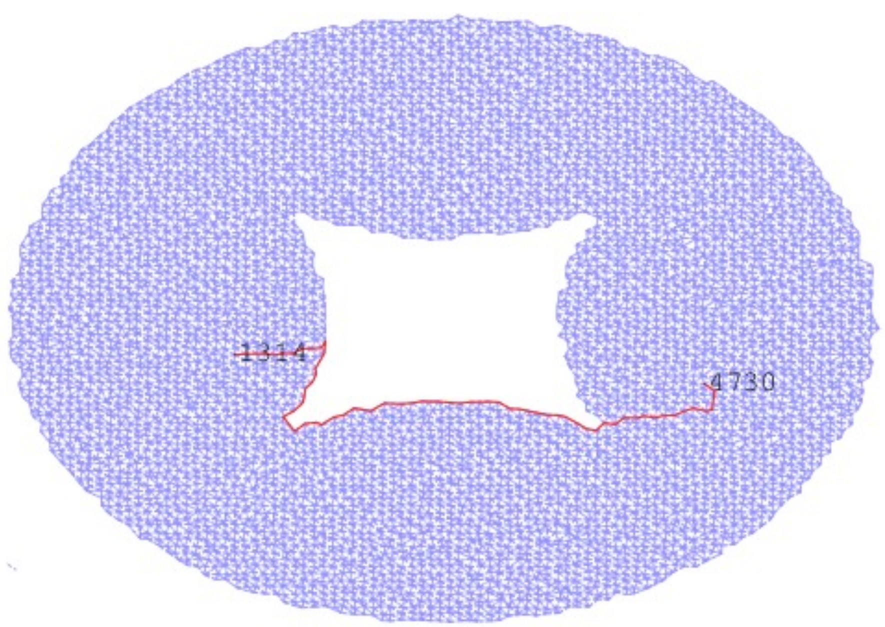
Greedy routing based on Ricci flow in 2D networks



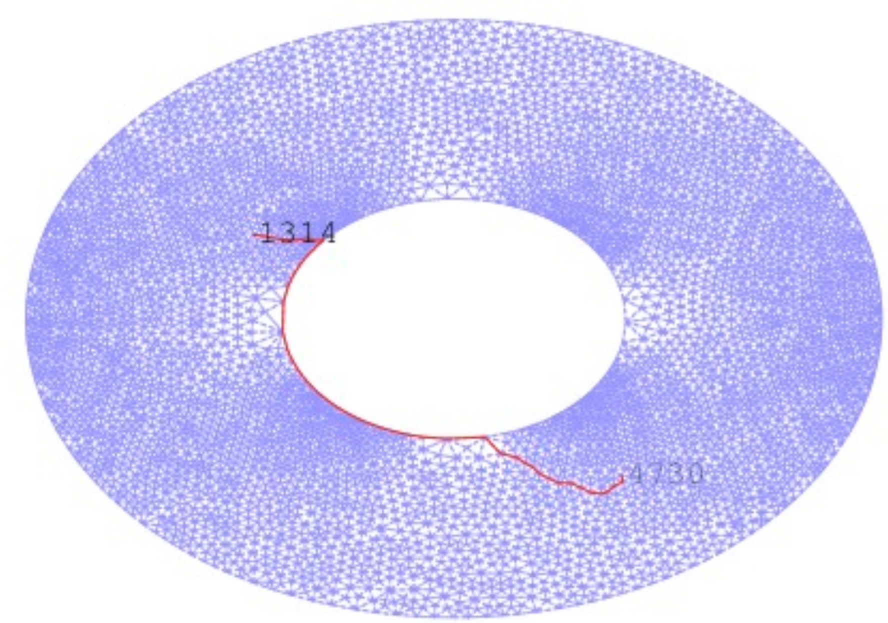
(a) Finest triangulation by [2]



(b) Ricci embedding



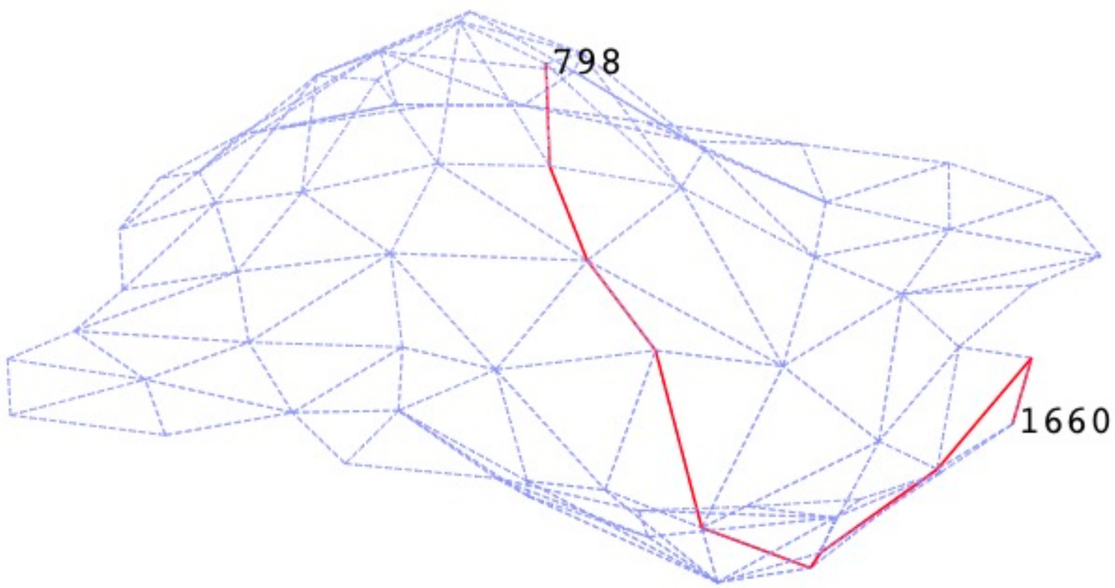
(c) Proposed triangulation



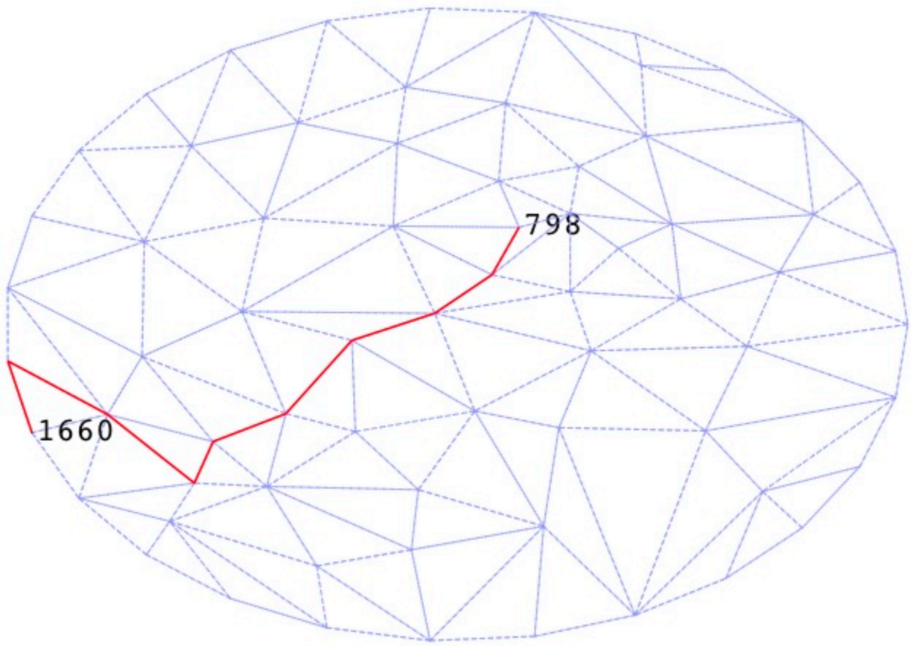
(d) Ricci embedding

APPLICATION AND SIMULATION RESULTS

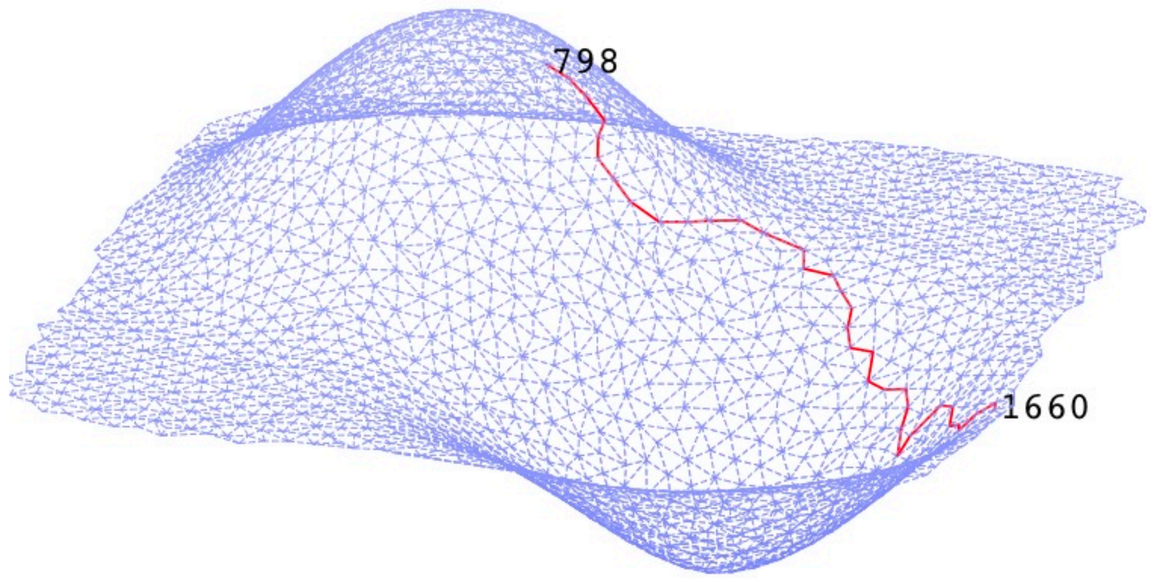
Greedy routing based on Ricci flow in 3D networks



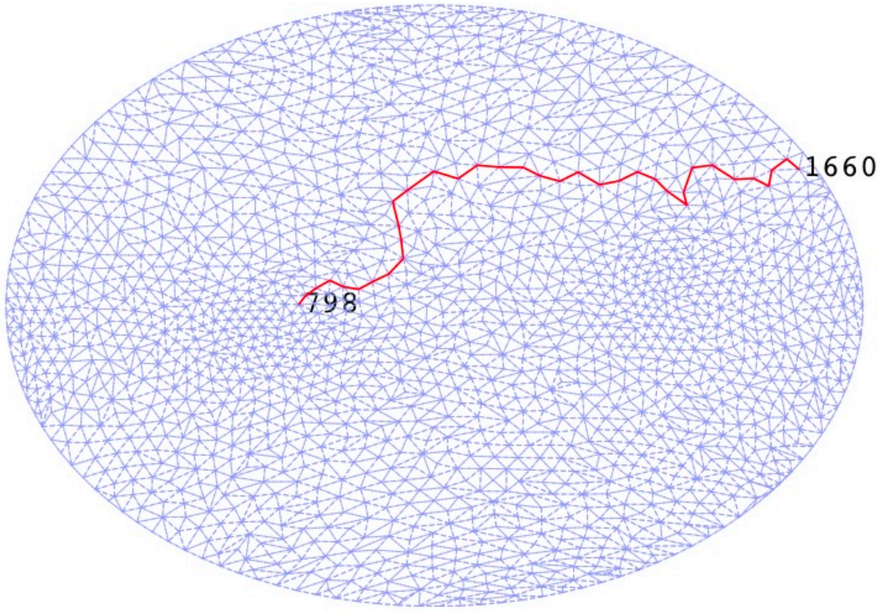
(a) Finest triangulation by [3]



(b) Ricci embedding



(c) Proposed triangulation



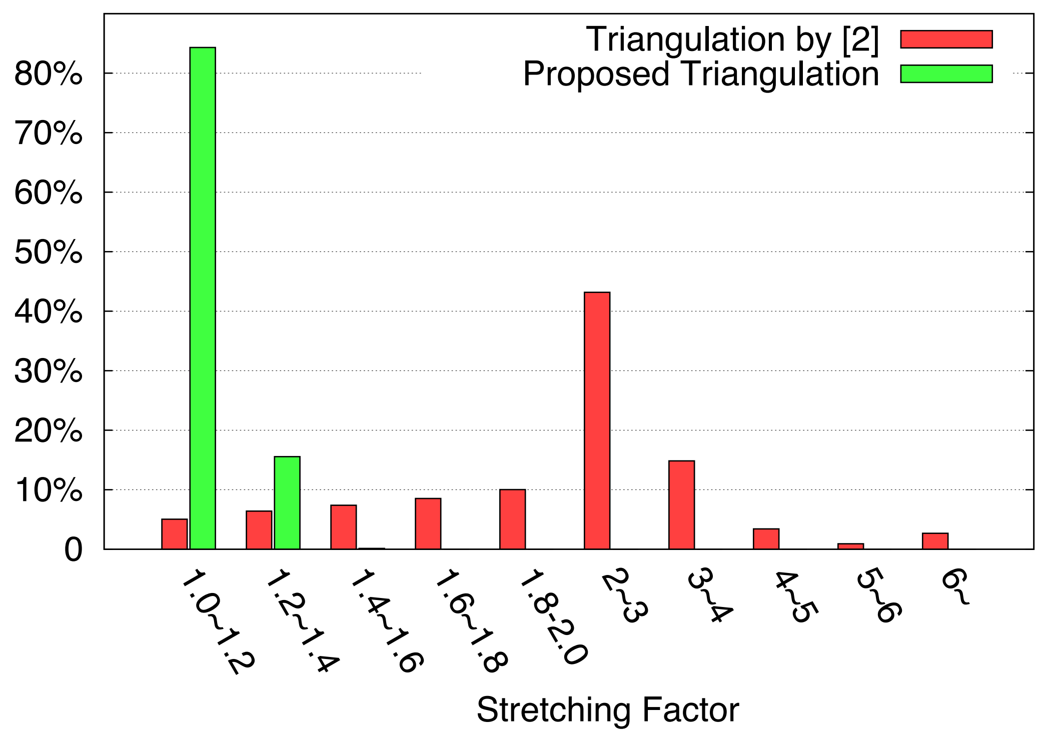
(d) Ricci embedding

Application and Simulation Results

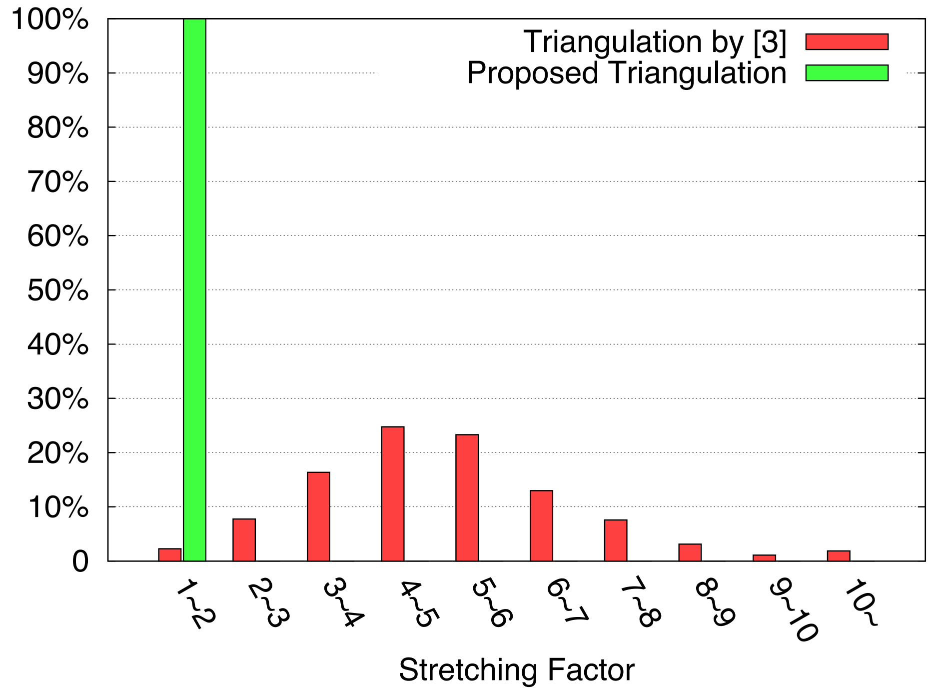
Comparison of stretch factor in greedy routing.

	2D networks	3D networks
Triangulation by [2], [3]	2.065	5.417
Proposed Triangulation	1.214	1.091

Distribution of stretch factor in greedy routing



(a) 2D networks



(b) 3D networks

CONCLUSION

- In summary, we have proposed a distributed algorithm that can triangulate an arbitrary sensor network without position information.
- The algorithm can achieve the finest triangulation and tolerate distance measurement errors.
- We have proven its correctness in 2D and extended it to 3D surface.
- A range of geometry-based network algorithms can benefit from the proposed triangulation.