

**EXPLOITING COPLANAR CLUSTERS TO
ENHANCE 3D LOCALIZATION IN WIRELESS
SENSOR NETWORKS**

ÇAĞIRICI, ONUR

JANUARY 2015

EXPLOITING COPLANAR CLUSTERS TO ENHANCE 3D LOCALIZATION IN WIRELESS SENSOR NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF
NATURAL AND APPLIED SCIENCES OF
IZMIR UNIVERSITY OF ECONOMICS

BY
ÇAĞIRICI, ONUR


IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE
IN THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

JANUARY 2015

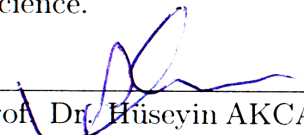
Approval of the Graduate School of Natural and Applied Sciences

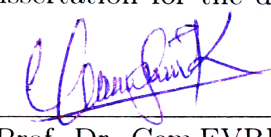

Prof. Dr. Cüneyt GÜZELİŞ
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.


Prof. Dr. Turhan TUNALI
Head of Department

We have read the dissertation entitled **Exploiting Coplanar Clusters to Enhance 3D Localization in Wireless Sensor Networks** completed by **Onur ÇAĞIRICI** under supervision of **Assoc. Prof. Dr. Cem EVRENDİLEK** and **Assoc. Prof. Dr. Hüseyin AKCAN** and we certify that in our opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Master of Science.


Assoc. Prof. Dr. Hüseyin AKCAN
Co-Supervisor


Assoc. Prof. Dr. Cem EVRENDİLEK
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Hüseyin AKCAN
Dept. of Software Engineering, IUE

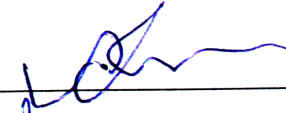
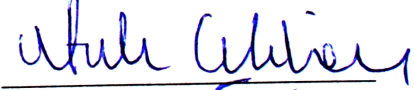

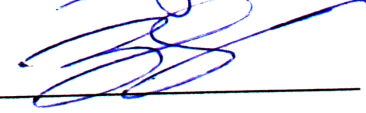
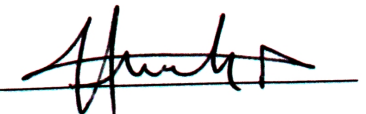
Asst. Prof. Dr. Ufuk ÇELİKKAN
Dept. of Software Engineering, IUE

Assoc. Prof. Dr. Cem EVRENDİLEK
Dept. of Computer Engineering, IUE

Asst. Prof. Dr. Burkay GENÇ
Inst. Of Population Studies, Hacettepe Uni.

Prof. Dr. Brahim HNICI
Dept. of Computer Engineering, IUE

Date: 23. 01. 2015

“Research is what I’m doing when I don’t know what I’m doing. ”

—*Wernher Von Braun*

Acknowledgments

During the preparation process of my thesis, I have had the privilege of working with a number of people who have made my time at the Izmir University of Economics enjoyable and rewarding.

First of all, I would like to thank to my advisor, Assoc. Prof. Dr. Cem Evrendilek and my co-advisor, Assoc. Prof. Dr. Hüseyin Akcan. They put their trusts in me and accepted me as a researcher for the research project that they investigate, which is supported by The Scientific and Technological Research Council of Turkey (TUBITAK) Career Grant no: 112E099. Without them, I could have not found such an interesting research topic for my thesis.

I am extremely grateful to my advisor, Dr. Evrendilek who has accepted me as his thesis student and gave me the chance to study under the supervision of a great person and outstanding computer scientist. He has patiently mentored me through the preparation of this dissertation. Under his guidance, I was able to learn the fundamental lessons of being a researcher. I owe every second to him that he spent on revising my draft copies.

I am very thankful for the time from Asst. Prof. Dr. Gazihan Alankuş, who helped me to develop one of the main parts of the algorithms suggested in this thesis.

I am also very thankful to Dr. Rasmus Fonseca, who has developed a great Java library, ProGAL, which I have used to implement the algorithms in this thesis. Even though we have never met in person, he has generously spent time on answering my e-mails about my endless questions about the tool he has developed.

I would like to thank to my colleagues Berkehan Akçay and Serhat Uzunbayır, who have never hesitated to help me in during my hard times.

Finally, I would like to express my greatest thanks to my father Ufuk, my mother Aynur and my beloved Deniz Ağaoğlu for their great support. They have endured me during the time I was the most unbearable and always stood by me regardless of my grumpinesses. Without their endless love and support, I would have never completed this thesis.

ABSTRACT

EXPLOITING COPLANAR CLUSTERS TO ENHANCE 3D LOCALIZATION IN WIRELESS SENSOR NETWORKS

Çağırıcı, Onur

M.Sc. in Intelligent Systems Engineering
Graduate School of Natural and Applied Sciences

Supervisor: Assoc. Prof. Dr. Cem Evrendilek
Co-Supervisor: Assoc. Prof. Dr. Hüseyin Akcan
January 2015, 61 pages.

This thesis studies range-based WSN localization problem in 3D environments that induce coplanarity. In most real-world applications, even though the environment is 3D, the grounded sensor nodes are usually deployed on 2D planar surfaces. Examples of these surfaces include structures seen in both indoor (*e.g.* floors, doors, walls, tables etc.) and outdoor (*e.g.* mountains, valleys, hills etc.) environments. In such environments, sensor nodes typically appear as coplanar node clusters. We refer to this type of a deployment as a *planar deployment*. When there is a planar deployment, the coplanarity causes difficulties to the traditional range-based multilateration algorithms because a node cannot be unambiguously localized if the distance measurements to that node are from coplanar nodes. Thus, many already localized groups of nodes are rendered ineffective in the process just because they are coplanar. We, therefore propose an algorithm called Coplanarity Based Localization (CBL) that can be used as an extension of any localization algorithm to avoid most flips caused by coplanarity. CBL first performs a 2D localization among the nodes that are clustered on the same surface, and then finds the positions of these clusters in 3D. We carry out experiments using trilateration for 2D localization, and quadrilateration for 3D localization, algorithm and experimentally verified that exploiting the clustering information leads to a more precise localization than mere quadrilateration. We also propose a heuristic to extract the clustering information in case it is not available, which is yet to be improved in the future.

Keywords: Range-based localization, wireless sensor network, WSN localization in 3D, NP-Hardness

ÖZ

3B'DE KABLOSUZ ALGILAYICI AĞ KONUMLAMASININ İYİLEŞTİRİLMESİ İÇİN EŞDÜZLEMSEL KÜMELERİN KULLANILMASI

Çağırıcı, Onur

Akıllı Mühendislik Sistemleri, Yüksek Lisans
Fen Bilimleri Enstitüsü

Tez Yöneticisi: Doç. Dr. Cem Evrendilek
İkinci Tez Yöneticisi: Doç. Dr. Hüseyin Akcan
Ocak 2015, 61 sayfa

Bu tez, 3B'de mesafe ölçümüne dayalı kablosuz algılayıcı ağları (KSA) konumlama problemini, eşdüzlemselliği tetikleyen ortamlarda inceliyor. Gerçek hayat uygulamalarının çoğunda, ortam 3B olmasına rağmen, uçmayan algılayıcılar 2B düzlemsel yüzeyler üzerinde dizilirler. Bu yüzeyler iç mekan yüzeyleri (katlar, kapılar, duvarlar, masalar *vb.*) olabileceği gibi, dış mekan yüzeyleri (dağlar, vadiler, bayırlar *vb.*) de olabilir. Bu tür ortamlarda algılayıcılar tipik olarak eşdüzlemsel kümeler halinde görünürler. Bu tip dizilime *düzlemsel dizilim* adını veriyoruz. Düzlemsel dizilimin bulunduğu ortamlarda, eşdüzlemsellik geleneksel mesafe ölçümüne dayalı konumlama algoritmaları için zorluklar oluşturur çünkü bir düğüm, eşdüzlemsel düğümlerden elde edilen uzaklık ölçümleriyle muğlak olmayan bir şekilde konumlanamaz. Böylece, hali hazırda konumlanmış birçok düğüm grupları, eşdüzlemsel oldukları için etkisiz hale gelirler. Bu nedenle, düzlemsel konuşlanma olduğunu bildiğimiz durumlarda, bu güçlükten başa çıkmak için, Coplanarity Based Localization (CBL), Türkçe adıyla Eşdüzlemsellik Tabanlı Konumlama (ETK) adında bir algoritma sunuyoruz. Sunduğumuz algoritma herhangi bir konumlama algoritmasının uzantısı olarak kullanılabilir. ETK, ilk olarak eşdüzlemsel yüzeylerde bulunan düğüm kümelerini, aynı kümedeki diğer düğümlere göre pozisyonlarını bulmak için bir 2B konumlama algoritması kullanır ve daha sonra kümelerin 3B'de yerlerini bulur. 2B konumlama algoritması olarak trilateration'u ve 3B konumlama algoritması olarak quadrilateration'u kullanarak yürüttüğümüz deneylerde de gördüğümüz üzere, kümelenme bilgisini kullanmak, salt quadrilateration'dan daha doğru sonuç veren bir konumlamaya yol açıyor. Kümelenme bilgisinin gelmediği durumda ise, eşdüzlemsel kümeleri keşfetmeye yönelik ve geliştirilmeye açık bir sezgisel de sunuyoruz.

Anahtar Kelimeler: Uzaklık-tabanlı konumlama, kablosuz algılayıcı ağları, 3B'de KAA konumlama, NP-Zorluk.

Contents

Acknowledgments	vi
Abstract	vii
Öz	viii
List of Figures	x
1 Introduction	2
1.1 Motivation	3
1.2 Contributions	7
1.3 Organization of the Thesis	8
2 Background and Terminology	9
3 Coplanarity Based Localization	17
3.1 Assumptions and Preliminaries	17
3.2 Trilateration and Quadrilateration	24
3.2.1 Experimental Evaluation of Trilateration and Quadrilateration	27
3.3 Localizing the Coplanar Node Clusters	28
3.3.1 Problem Definition	28
3.3.2 The CBL Algorithm	29
3.3.3 Experimental Evaluation of CBL	35
3.4 Extracting the Coplanar Clusters	40
3.4.1 Problem Definition	40
Setting the Volume Threshold κ	41
Setting the Hop Distance θ	44
3.4.2 A Heuristic to Extract Coplanar Clusters	48
3.4.3 Experimental Evaluation of Planar Clustering	51
4 Conclusion	53
4.1 Related Work	53
4.2 Discussion	54
4.2.1 Summary of Contributions	55
4.2.2 Future Research Directions	56
5 Bibliography	57

List of Figures

1.1	A wireless sensor network	2
1.2	Sensor nodes deployed onto a valley (a), onto a mountain (b), and inside a multi-storey building (c).	4
1.3	A node being localized by three localized nodes in 2D	6
1.4	A node being localized by four localized nodes in 3D	6
1.5	The unlocalized node E cannot be localized unambiguously because of coplanarity.	7
2.1	An example edge set of a graph	10
2.2	Point formation of the graph with the edge set given in 2.1	10
2.3	A point formation of a graph with the edge set given in Figure 2.1	10
2.4	A graph with trilateration ordering	11
2.5	A complete (on the left), and an incomplete (on the right) tetrahedron	12
2.6	Seven topological regions that a node can be with respect to three localized nodes	13
2.7	Three possible positions for the node in three different topological regions	13
2.8	Planar deployment	15
2.9	Random deployment	16
3.1	Assigning positions to the seed nodes	19
3.2	The projection of three spheres with centers a , b , c and d to the $z = 0$ plane.	21
3.3	Picking a position among two candidate points	23
3.4	Trilateration (a) and quadrilateration (b) algorithms	26
3.5	The recall percentages of trilateration (a) and quadrilateration (b) with respect to increasing node connectivity	27
3.6	The average offsets of trilateration (a) and quadrilateration (b) with respect to increasing error magnitude when the average node connectivity is 15	28
3.7	Transformation of a coplanar cluster	30
3.8	Semi-localization (a), and rigid-localization (b) of an unlocalized cluster	33
3.9	CBL algorithm	34
3.10	Slicing a cube with perpendicular planes	37
3.11	Recall percentage of CBL with noiseless distance measurements.	37

3.12	The change in average offsets (a), and recall percentages (b) of CBL and quadrilateration with respect to increasing error magnitude and various planarity factors when the sensing range is 40 units	38
3.13	A closer view of the plots in Figure 3.12a	39
3.14	A coplanar group of nodes (a), a non-coplanar group of nodes, interpreted as coplanar (b), a non-coplanar group of nodes, not interpreted as coplanar (c)	42
3.15	An example of inappropriate expansion from two angles of view	43
3.16	Two nodes from a coplanar cluster, and their neighbors	44
3.17	Filtering edges with respect to changing hop distance	46
3.18	Planar clustering of a coplanar node set	47
3.19	Heuristic to extract coplanar clusters	49
3.20	Average offsets (a) and recall percentages(b) of CBL and quadrilateration when the information on coplanar clusters is not available	52
4.1	This thesis in the state-of-art	55

Chapter 1

Introduction

Recent developments in technology pose a need for wireless sensor network (WSN) technologies to be used broadly [1]. A WSN consists of wireless sensor nodes that communicate among each other in order to complete a given task. There are several application areas that use WSNs. Zhong made a classification of these application areas in his thesis [2] namely, military tasks, industrial process monitoring and control, habitat and environment monitoring, health-care applications, home automation, and vehicle networks and intelligent transportation systems. The sensor nodes usually communicate with each other to pass their gathered data to a destination. In Figure 1.1, we see a wireless sensor network. The rectangles represent wireless sensor nodes and the dashed lines between them represent their communication links.

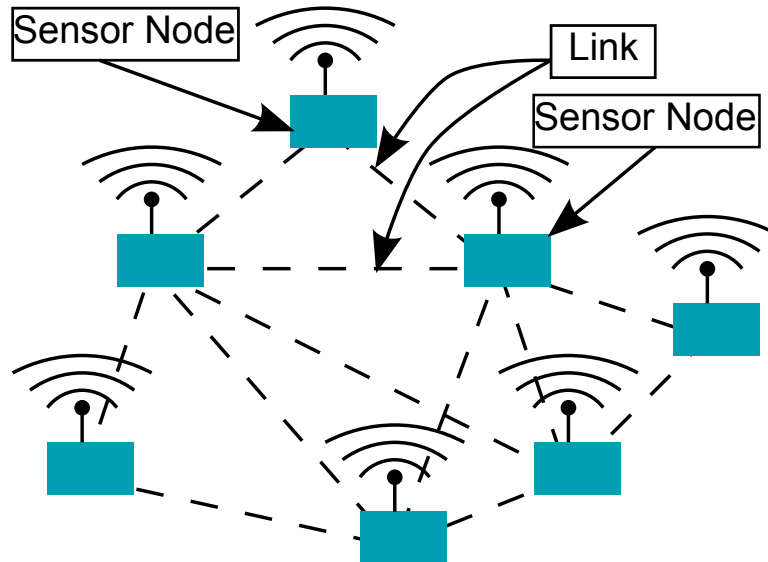


Figure 1.1: A wireless sensor network

Independent of the application area, the location information of wireless sensors in a WSN is crucial to improve the quality of service. For all types of applications, the information of position can be appended to the data that is gathered from the sensor nodes, easing the traceability of the sensors, as in WSNs that

works in geographical routing. The information of positions is able to increase the quality of the applications such as geographical based queries. When equipped with proper devices, a sensor node can measure distance to another node in the network using the communication links shown in Figure 1.1. Finding the positions of these sensors is called *WSN localization*. If the sensor nodes are positioned only by using the distance measurements, this process is called *range-based localization*. Even though there are lots of studies on range-based WSN localization in 2D, [2, 3, 4, 5, 6, 7, 8], the volume of studies for 3D WSN localization is relatively small [9].

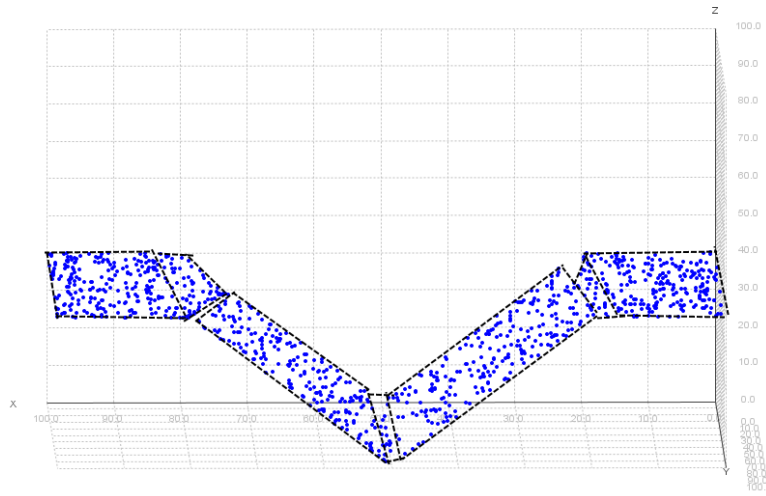
The physical world we live in, however, is a 3D environment. Therefore, many applications require localization in 3D. Most of the scenarios in real-world WSN applications that need localization have usually deployments where the sensor nodes sit on planar regions to form sets of planar clusters as seen in Figure 1.2. In Figure 1.2a, there are 1000 sensor nodes deployed onto a valley. The same number of sensors are deployed onto a mountain in Figure 1.2b, and inside a multi-storey building in Figure 1.2c. It is hence observed that a method for exploiting the information of structural information in 3D is very much needed in order to improve the quality of localization achieved. In this thesis, we study the range-based WSN localization problem in 3D environments in which sensors are deployed onto planar surfaces.

1.1 Motivation

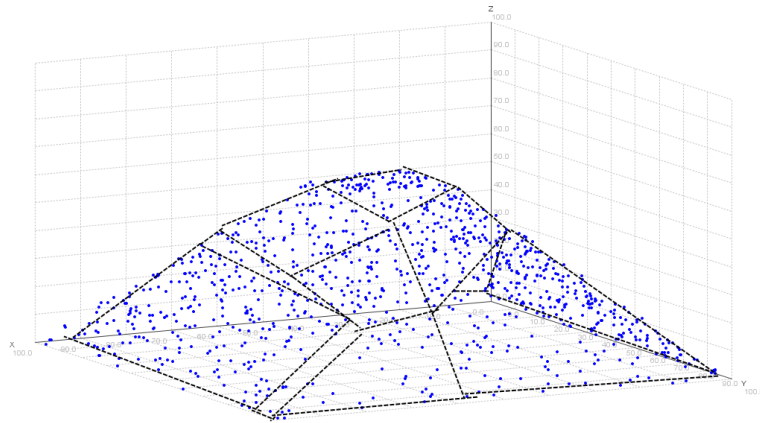
Localization, in general, is defined as finding the relative coordinates of an object, with respect to a certain reference system. The corresponding coordinates can be the relative coordinates in a room, coordinates in a building or global coordinates on the earth. In order to find the position of an object, several techniques were developed. A well-known method to localize an object on the earth globe is called Global Positioning System (GPS) [10]. GPS uses four or more orbiting satellites to localize an object on the earth. In 2000, Benefon released the first cellular phone which uses an integrated GPS device [11]. With this development, GPS has become a widely-used system throughout the world. However, there are major drawbacks of GPS, which can be listed as:

- The accuracy is relatively low for critical usage (up to 15 meters deviation).
- The system has incremental costs.
- The energy consumption of the system is excessive.
- The system is nonfunctional in indoor environments.

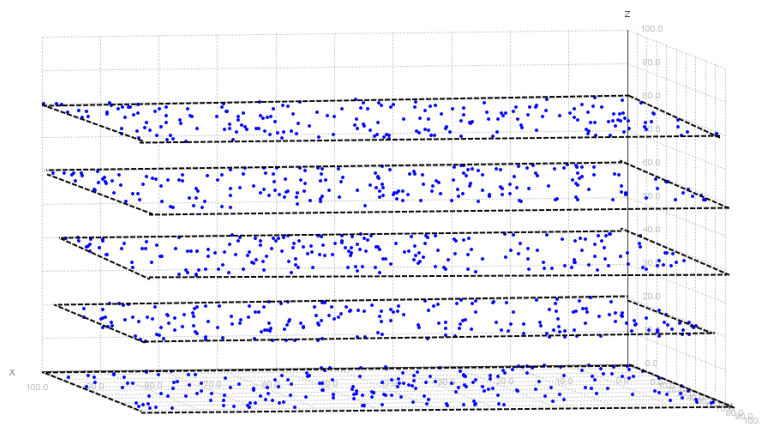
These reasons have paved the way for a localization method that works independent of GPS. A considerable GPS-free localization system was designed by Bulusu et al. in 2000 [3]. They use a frequency-based approach to localize



(a) Sensor nodes deployed onto a valley



(b) Sensor nodes deployed onto a mountain



(c) Sensor nodes deployed inside a building

Figure 1.2: Sensor nodes deployed onto a valley (a), onto a mountain (b), and inside a multi-storey building (c).

ultra-wideband (UWB) radio nodes [12]. UWB radio nodes are able to measure distances among themselves using ultra-wide band signals [13]. They can be used both for static localization when the nodes are immobile [14], and dynamic localization when nodes are able to move [7]. This method was studied extensively, and more research is still being conducted on GPS-free localization [5, 4, 8, 15, 16, 17].

In 2001, Bahl and Padmanabhan developed a very popular indoor positioning system (IPS) [18]. The system is called RADAR (not to be confused with Radio Detection And Ranging [19]) and its working principle is similar to GPS but was designed for one floor in a building. After RADAR was developed, several studies were made in this area [20, 21]. The system uses three *base stations* in order to localize objects and track their movements. Hence, the installment of the system is expensive for a multi-storey building. Besides, power consumption of the base stations increases with the number of the objects being tracked. Thus, developing an energy efficient solution becomes critical for the successful operation of IPSs.

A well-known decentralized system, called a *wireless sensor network* (WSN) [22] has been used in order to achieve a scalable, efficient and low-cost localization for indoor environments. WSN is a type of *wireless ad hoc network* [6, 23] and uses low-cost devices called *wireless sensor nodes*. Localization can be done using the wireless sensor nodes either by utilizing the pairwise distances among the sensor nodes, called range-based localization [4, 18, 17, 7] or just by using the connectivity information of the sensor nodes, called range-free localization [2, 3, 24].

Range-based WSN localization is a method to obtain the relative positions of the sensor nodes with respect to a number of nodes with known positions by using the available pairwise distances among the sensor nodes. While measuring distances, sensor nodes use methods such as time of arrival (TOA) [6], time difference of arrival (TDOA) [25], or received signal strength (RSS) [26]. Although range-based WSN localization is proven to be an NP-Hard problem [27], Eren *et al.* [28] showed that, if certain conditions are satisfied, range-based localization can be done in polynomial time, using trilateration which uses three distance measurements to localize a node. However, when there are errors in distance measurements which is the case in real life, localization of a network by trilateration becomes intractable [29]. Despite being intractable, trilateration is still used frequently [9, 4, 14].

In Figure 1.3, we see a node D , being localized by three nodes A , B and C . The circles in the figure represent the distance between the unlocalized node and the localized nodes. The coordinates of D can be determined by computing the intersection point of these circles.

In 3D, quadrilateration can be used to localize WSN nodes. Analogous to trilateration in 2D, quadrilateration uses four distance measurements from four already localized non-coplanar nodes to localize a fifth node. In Figure 1.4, we see the localization of an unlocalized node E , using the distance measurements

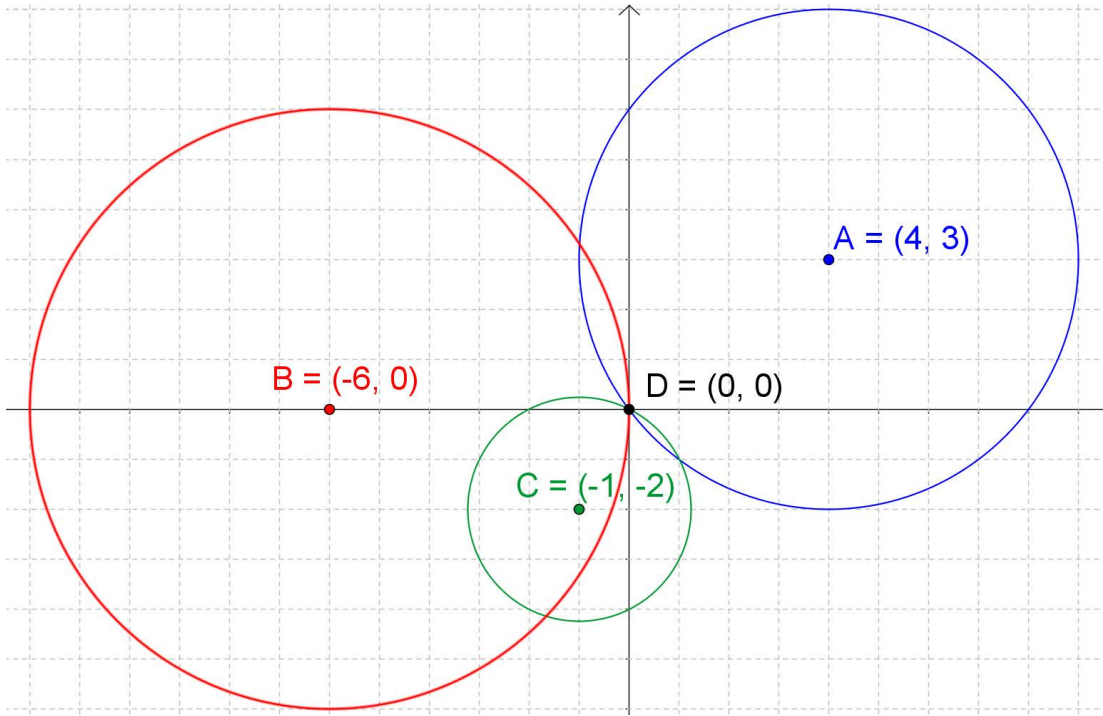


Figure 1.3: A node being localized by three localized nodes in 2D

from four localized nodes A , B , C and D .

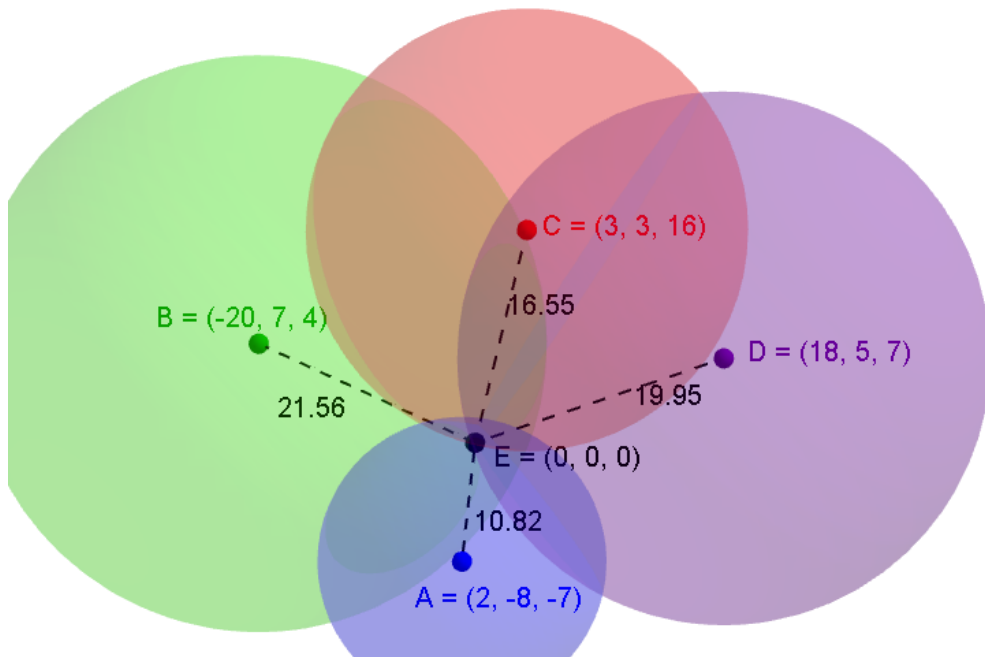


Figure 1.4: A node being localized by four localized nodes in 3D

When the localized nodes are on the same plane, *i.e.* coplanar, however, no matter how many distance measurements are available, a node cannot be unambiguously located unless it is also on the same plane as the others. In Figure 1.5, there are many distance measurements from many localized nodes to E . Since the localized nodes sit on the same plane, there are always two possible

positions for the unlocalized node. One is E , and the other is the reflection of E about the plane that the others sit on, E' . Since E is not coplanar with the localized nodes, there is no way of telling E' from E .

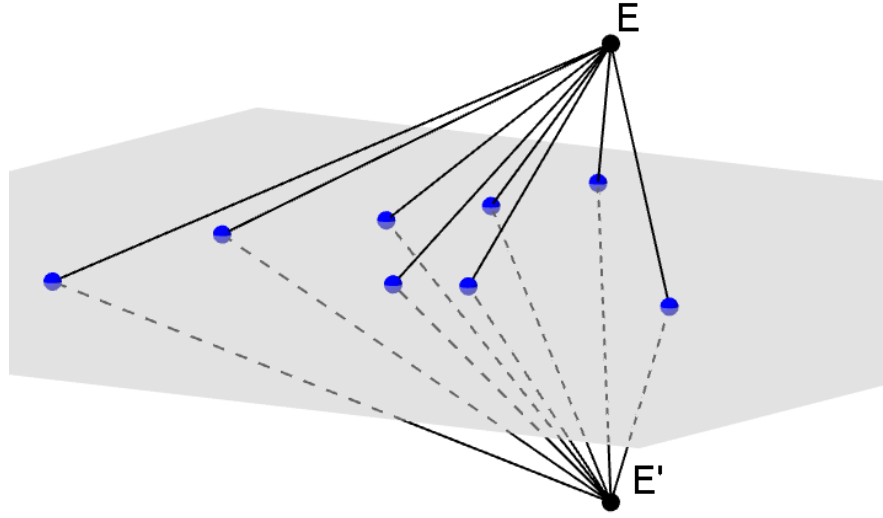


Figure 1.5: The unlocalized node E cannot be localized unambiguously because of coplanarity.

In most real-world applications, sensor nodes are deployed on planar surfaces. Examples of these surfaces include structures seen in both indoor (*e.g.* floors, doors, walls, tables etc.), and outdoor (*e.g.* mountains, valleys, hills etc. recall Figure 1.2) environments. In such environments, sensor nodes typically appear as coplanar node clusters. We call this type of a deployment as a *planar deployment*. We try to extract and utilize such information in order to achieve a better range-based WSN localization than the traditional range-based localization methods, such as trilateration [28], quadrilateration [15], and multilateration [25] when the environment is particularly known to follow a planar deployment.

1.2 Contributions

Our main contributions in this thesis are:

1. We show that when the nodes are deployed on planar surfaces in 3D, as is mostly the case in real-world applications, quadrilateration does not work as expected due to a massive amount of coplanar sensors.
2. We propose an algorithm to utilize the coplanarity, if the sensor nodes are known to follow a planar deployment pattern. We call this algorithm Coplanarity Based Localization (CBL).
3. We show that CBL performs a more precise localization than mere quadrilateration when the information of coplanar node clusters are known apriori.

4. When the clustering information is unavailable, we propose a heuristic algorithm that tries to extract the coplanar clusters. Although our first attempt to extract clustering information is yet to be improved, we also clearly highlight the need for an algorithm to extract such information accurately in 3D localization.

1.3 Organization of the Thesis

The organization of this thesis is as follows: In Chapter 1, we present the introduction. In Chapter 2, we give the basic background and knowledge for range-based wireless sensor network localization. In Chapter 3, we present CBL, and a heuristic to extract coplanar clusters. Finally in Chapter 4, we conclude this thesis.

Chapter 2

Background and Terminology

In this chapter, we present basic background and terminology for WSN localization.

A WSN graph is a graph $G = (V, E)$, such that each vertex $v \in V$ corresponds to a sensor node. We assume that each sensor node has the same sensing range, denoted by R , and an edge $(v, w) \in E$ exists if and only if $d(v, w) \leq R$ where $d(v, w)$ denotes the Euclidean distance between v and w . This type of a graph is called a *unit-disk graph* in 2D, and a *unit-ball graph* in 3D.

In our problem, a WSN graph can contain three types of nodes; namely seed, localized, and unlocalized nodes. The coordinates of a *seed node* is known apriori. The network is usually localized based upon the positions of such seed nodes. If a node is not a seed node, but localized subsequently based onto the positions of the seed nodes, it is called a *localized* node. We say that a node is *unlocalized* if its position is unknown.

Localizing a WSN graph G means assigning a position to each vertex $v \in V$ in $d \in 2, 3$ dimensions, so that the distances given by the edge weights are all satisfied. This operation is referred to as finding a *point formation* of G [30, 31]. The point formation of a graph G is denoted by $G_{\mathbb{F}}$. For instance, let us consider a WSN graph with the vertex set $V = \{v_1, \dots, v_5\}$, and the edge set as given in Figure 2.1.

In Figure 2.2, a possible point formation of the given graph in 2D is shown with respect to the edge set given in Figure 2.1.

In Figure 2.3, 2D coordinates are assigned to the vertices of the graph whose point formation is given in Figure 2.2.

If we fix the positions of any three nodes in Figure 2.3, then unique positions can be assigned to the rest. If there is a unique point formation of a graph G in d dimensions, then G is called *globally rigid* in d dimensions. Global rigidity

$$\begin{aligned}
d(v_1, v_2) &= 2.00000 \\
d(v_1, v_3) &= 3.60555 \\
d(v_1, v_4) &= 3.00000 \\
d(v_2, v_3) &= 2.23606 \\
d(v_2, v_4) &= 3.60555 \\
d(v_2, v_5) &= 5.00000 \\
d(v_3, v_4) &= 3.16227 \\
d(v_3, v_5) &= 3.16227 \\
d(v_4, v_5) &= 2.82842
\end{aligned}$$

Figure 2.1: An example edge set of a graph

$$\mathbb{F}_G = \left([v_1, v_2, v_3, v_4, v_5], \begin{bmatrix} 0, 0 \\ 0, 2 \\ 2, 3 \\ 3, 0 \\ 5, 2 \end{bmatrix} \right)$$

Figure 2.2: Point formation of the graph with the edge set given in 2.1

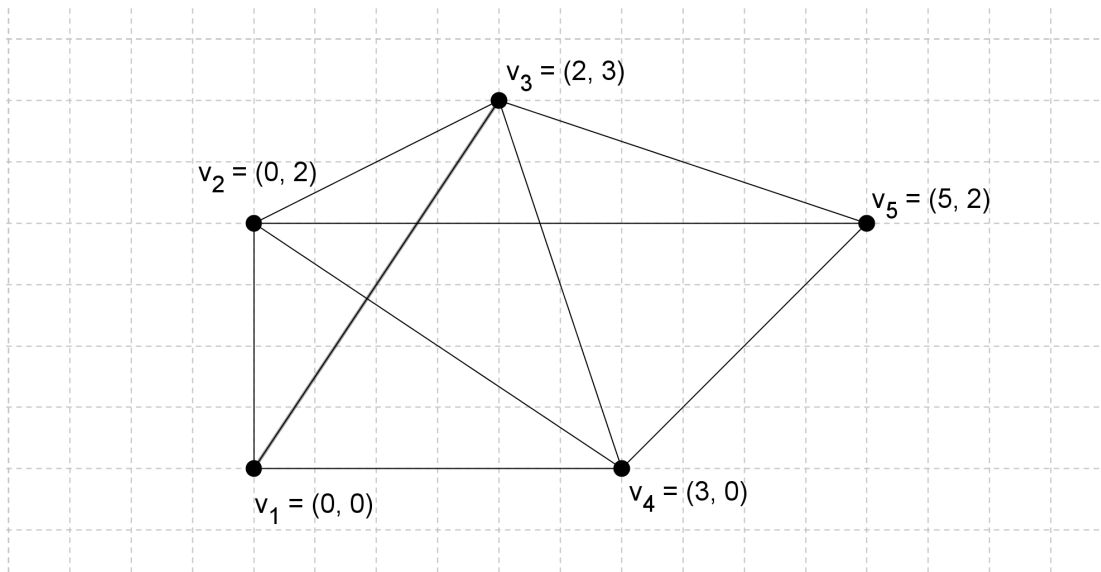


Figure 2.3: A point formation of a graph with the edge set given in Figure 2.1

is both sufficient and necessary for a graph to be localizable in 2D [17], and can be tested in polynomial time [32, 33, 34]. Although global rigidity is a necessary condition in 3D as well [35], the sufficiency for a graph to be localizable in 3D is still an open problem [34]. Saxe [36] proved that localization is an NP-Hard problem for all dimensions even when the network is known to be localizable.

Trilateration [28] is a localization algorithm that uses distance measurements from three non-collinear nodes to localize an unlocalized node in 2D. If a WSN graph can be fully localized using trilateration, we say that this graph has a *trilateration ordering* [28, 17]. In Figure 2.4, we see a graph with a trilateration ordering where the vertices are labeled 1 through 9. The nodes 1, 2 and 3 are picked as seed nodes and marked with squares.

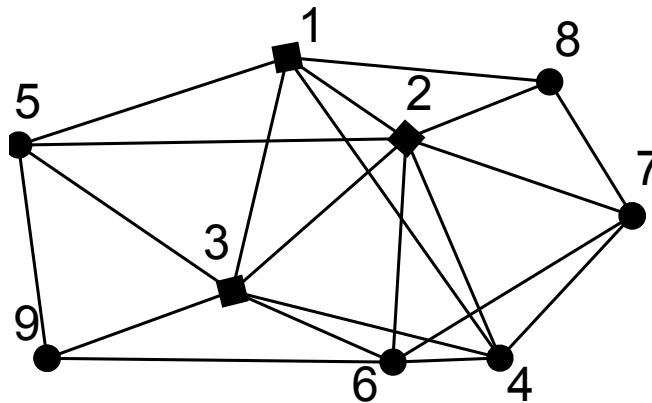


Figure 2.4: A graph with trilateration ordering

The whole graph in Figure 2.4 can be localized in six steps with the following ordering

- 1, 2, 3 \rightarrow 4
- 1, 2, 3 \rightarrow 5
- 2, 3, 4 \rightarrow 6
- 2, 4, 6 \rightarrow 7
- 1, 2, 7 \rightarrow 8
- 3, 5, 6 \rightarrow 9

Notice that we cannot localize the graph if the seed nodes have no common neighbors. For instance, if we pick 3, 5 and 9 as seed nodes, then we cannot localize any nodes since they have no common neighbors. In order to localize the maximum possible number of nodes *i.e.* to reach the maximum recall percentage possible, the algorithm tries every possible node triplet as the seed when a trilateration ordering is not specified. Even though trilateration is a polynomial time algorithm, if the distance measurements are not fully accurate due to environmental noise, localizing a graph with trilateration is NP-Hard [37] even when a trilateration ordering is given as part of the input. In 3D, a similar algorithm,

quadrilateration [9, 15] can be used to localize a WSN that uses distance measurements from four non-coplanar sensors to localize an unlocalized sensor in 3D, which is also a polynomial time algorithm.

Since we need to use non-coplanar nodes to avoid ambiguities, we should detect if four nodes lie on the same plane. In order to detect coplanarity, we can form a tetrahedron using the pairwise distances between four points a , b , c and d . The volume of the tetrahedron can be found by *Cayley-Menger determinant* [38] of the matrix given below.

$$M = \begin{bmatrix} 0 & d(a,b)^2 & d(a,c)^2 & d(a,d)^2 & 1 \\ d(a,b)^2 & 0 & d(b,c)^2 & d(b,d)^2 & 1 \\ d(a,c)^2 & d(b,c)^2 & 0 & d(c,d)^2 & 1 \\ d(a,d)^2 & d(b,d)^2 & d(c,d)^2 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

where $d()$ corresponds to the Euclidean distance between nodes. Hence, the volume of a tetrahedron can be computed as follows.

$$\mathcal{V}_{abcd} = \frac{\sqrt{\det(M)}}{288}$$

where \mathcal{V}_{abcd} denotes the volume of the tetrahedron formed by the points a , b , c , d , and given pairwise distances.

If $\det(M) > 0$, then $\mathcal{V}_{abcd} > 0$, indicating that the points are not coplanar. If $\det(M) = 0$, then $\mathcal{V}_{abcd} = 0$, indicating that the points are coplanar. If $\det(M) < 0$, then $\mathcal{V}_{abcd} \notin \mathbb{R}$, indicating that the tetrahedron is *incomplete*. An incomplete tetrahedron means that one of its corners is "open". In Figure 2.5, we see two tetrahedra, one is complete and the other is incomplete.

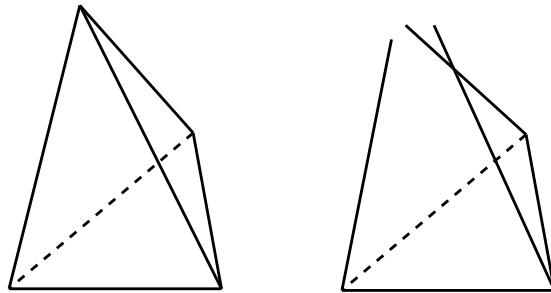


Figure 2.5: A complete (on the left), and an incomplete (on the right) tetrahedron

Let us assume that a node D is to be localized by three localized nodes A , B and C . The node D is said to be *flip* in 2D, if its computed position is in a different topological region than its actual position with respect to the nodes A , B and C . If a node is localized to a different topological area with respect to its three beacons, this situation is referred to as a *flip* [39]. These topological areas can be obtained by drawing three lines passing through each pair of localized nodes, as seen in Figure 2.6.

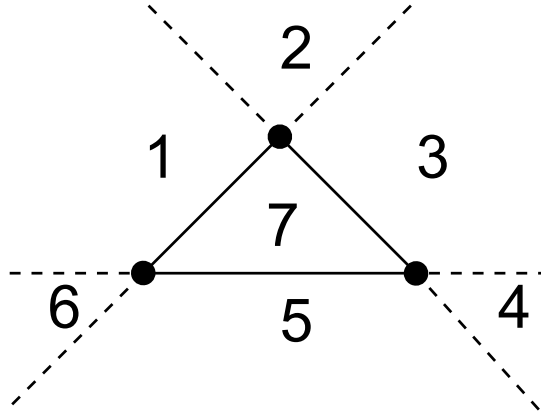


Figure 2.6: Seven topological regions that a node can be with respect to three localized nodes

The drawn lines divide the plane into seven topological areas. In Figure 2.7 a flip ambiguity is demonstrated. A , B and C are the localized nodes. The dashed lines divide the 2D plane into seven regions. Three possible positions D_1 , D_2 and D_3 are inside three different topological regions with respect to A , B and C .

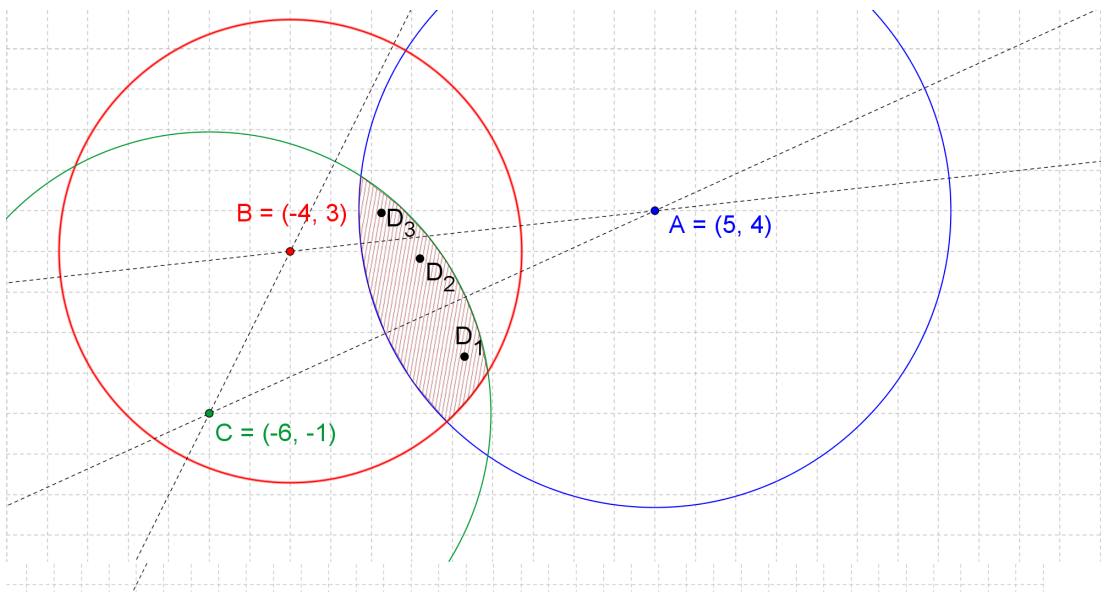


Figure 2.7: Three possible positions for the node in three different topological regions

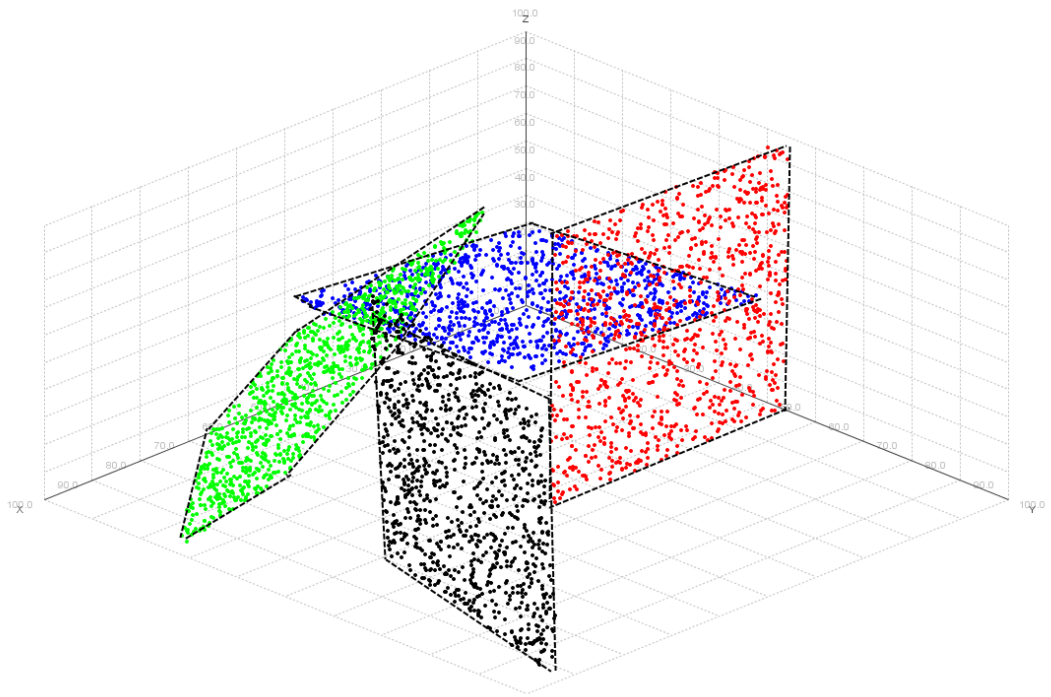
The same concept can be used to define a flip in 3D. Note that each triplet in a localized node quadruplet define a plane, dividing the 3-space into fifteen regions with four planes. We say that a node is flipped in 3D if its estimated position is inside a different region than its original position with respect to the four localized nodes used to localize it.

Figure 2.8 demonstrates a planar deployment of 4000 nodes that are distributed into four coplanar clusters. In Figures 2.8a and 2.8b, we see the clusters from two different angles of views. Each cluster contains 1000 nodes and indicated with a different color. If two neighbor nodes are in the same cluster,

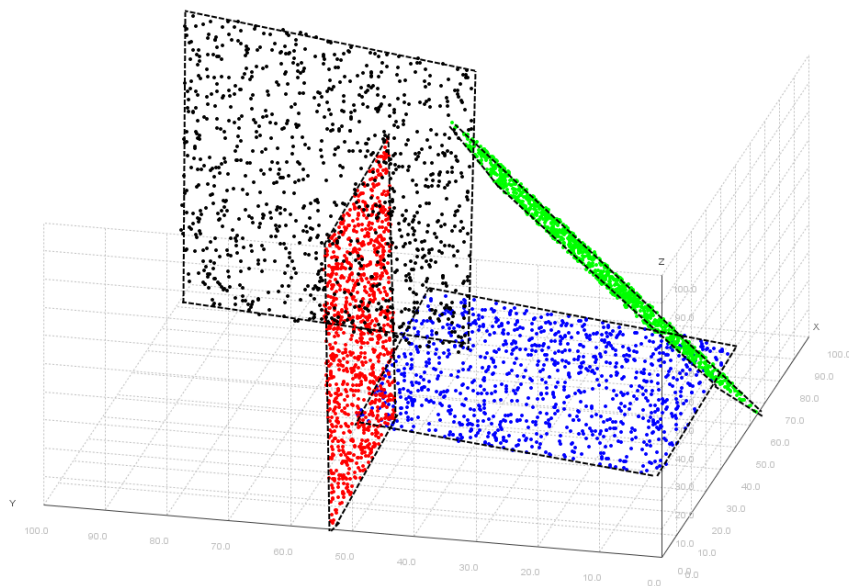
they are called *coplanar neighbors*. Otherwise, we refer to them as *interplanar neighbors*. Correspondingly, an edge between two coplanar neighbors is called a *coplanar edge*, and otherwise, an *interplanar edge*. A group of nodes that are in the same cluster cannot be used to localize one of their common interplanar neighbors unambiguously.

Figure 2.9a and 2.9b demonstrate the corresponding views when an equal of nodes are randomly distributed. The figures show the difference between a planar deployment, and a random deployment in 3D. When there is a random deployment, the nodes are seen as a cloud of points. However, in planar deployment, the planar structures are more distinguishable to the human eye. Moreover, during the localization process, a single node has a relatively more number of coplanar neighbors than its interplanar neighbors.

One can notice that every three nodes define a plane in 3D. However, the planar deployments that we seek to exploit are not particularly these plane formations with so few nodes in each cluster. In order to indicate how planar the deployment is, we define a metric called *planarity factor*, denoted by μ , and calculate it as $1 - \frac{k^2}{n}$, where k is the number of the clusters and n is the number of sensor nodes. It is basically the number of clusters divided by the number of nodes on each cluster and subtracted from 1.

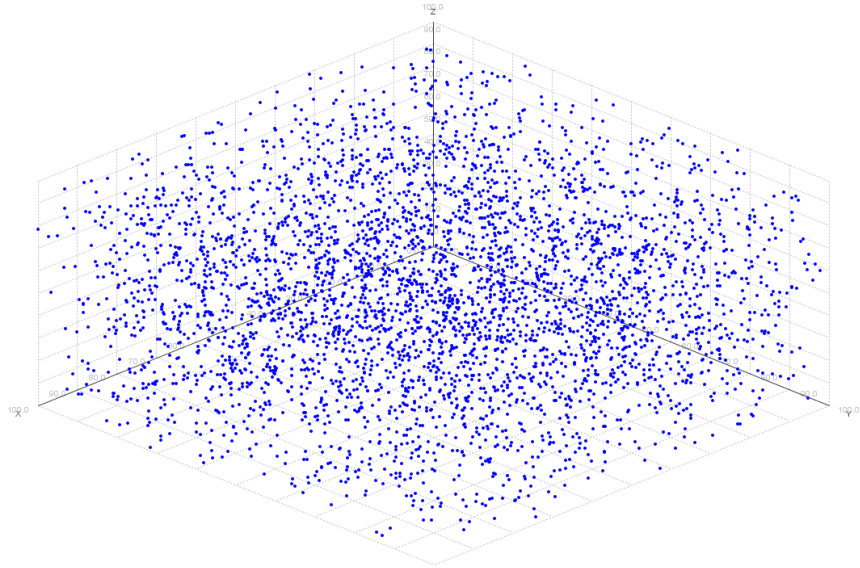


(a) An example of a planar deployment as seen from a particular angle

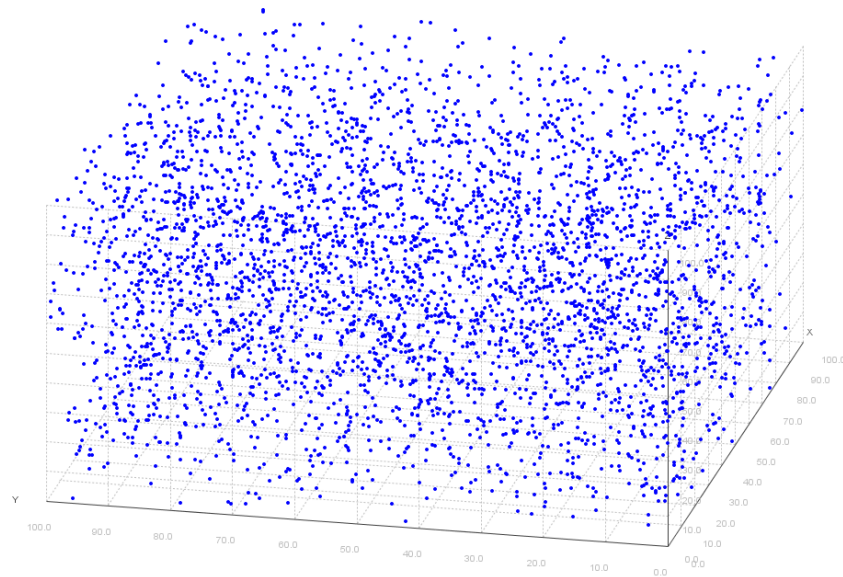


(b) The same planar deployment as seen from another angle.

Figure 2.8: Planar deployment



(a) An example of random deployment with the same number of nodes as seen from the same angle of view as in 2.8a



(b) The same random deployment as seen from the same angle of view as in 2.9b

Figure 2.9: Random deployment

Chapter 3

Coplanarity Based Localization

In this chapter we introduce the localization problem that we deal with, and propose an algorithm to solve it. There are two main problems that we tackle when the sensor nodes are known to follow a planar deployment pattern. The first problem is *localizing the coplanar clusters*, in case we have the clustering information, namely, we know which coplanar cluster does each node belong to. This information can be obtained when airdropping the sensor nodes onto planar surfaces with a known map. If the sensor nodes are dropped in clusters, even though we may not know the exact positions of the sensor nodes, we will know the planar surfaces that the nodes are on.

The second problem that we address arises when the clustering information is either lost or not available. In this case, an additional effort is needed to discover the coplanar node clusters. Only after the clustering information is made available, CBL can exploit it. We refer to this second problem as *extracting the coplanar clusters* or *planar clustering*.

In Section 3.1, we give the assumptions and preliminaries for the algorithms that we present. In Section 3.2, we explain the algorithms that we use for 2D and 3D localization, namely, trilateration and quadrilateration. In Section 3.3, we present a framework for range-based localization of the coplanar node clusters either found previously or given as part of input. If the clustering information is not given, we describe a heuristic algorithm in Section 3.4, to find the coplanar node clusters.

3.1 Assumptions and Preliminaries

We make the following assumptions while presenting the algorithms:

- WSN graph $G = (V, E)$ is a global variable and can be reached by any function.

- Each sensor node n has a *neighbor list* that contains the nodes inside the sensing range of n . Any subsets of the neighbors of n can be accessed by using the notation $n.Neighbors(\{n_i, n_j, n_k, \dots\})$.
- A coplanar cluster $C_i = (V_i, E_i)$ is a subgraph of a WSN graph $G = (V, E)$ where $V_i \subset V$, $E_i \subset E$, and $\forall a \neq b \in V_i, (a, b) \in E \Leftrightarrow (a, b) \in E_i$. The coplanar clusters are stored inside a set called $CSet$, defined as a global variable.
- The cluster of a sensor node n can be accessed by using the notation $ClusterOf(n)$
- Each sensor node n has two positions, namely its *local position* and its *global position* which are denoted by $n.LPos$ and $n.GPos$ respectively. Global position is the coordinates in 3D. Local position is in 2D, and relative to the local positions of its coplanar neighbors. If a node n in a cluster C_i is localized with respect to nodes $\{C_i \setminus n\}$ only, we say that the node is *locally positioned*. Determining the global position of a node means that node is *globally positioned*.
- All sensor nodes except the seed nodes are initially unlocalized in both 2D and 3D.
- After performing 2D localization on a coplanar cluster $C_i = (V_i, E_i)$, we are able to obtain the 2D point formation of the sensors in that cluster.
- Although CBL can be used as an extension of any localization algorithm, we assume that trilateration and quadrilateration are used for 2D and 3D localization respectively. These two algorithms are presented in Section 3.2.
- When the distance measurements are noisy, we assume that a certain value is added to the measured distance values based on the *error magnitude*. The error is modeled as the summation of a high probability small noise and a low probability large noise as experimentally gathered [4, 40, 41]. The small noise is a Gaussian random process with mean $N(f(R), \mathcal{E}/100)$ where R is the sensing range of the sensors, $f(R) = 0.022\ln(1 + R) - 0.038$ and \mathcal{E} is the magnitude of error. The large noise value is selected with a uniform random process between $\pm P\%$ of the wireless range with probability 0.05 where $P \in [0, 10]$. It is assumed that we have the knowledge of the error magnitude.

While running trilateration among a coplanar node cluster, we pick in that cluster three seed nodes with a known distance between any two. Since we do not know any of the coordinates initially, we construct a triangle on the $z = 0$ plane using the pairwise distances among these three nodes. Figure 3.1 demonstrates three nodes a , b and c picked as the seed nodes. The pairwise distances are denoted by r_{ab} , r_{ac} and r_{bc} .

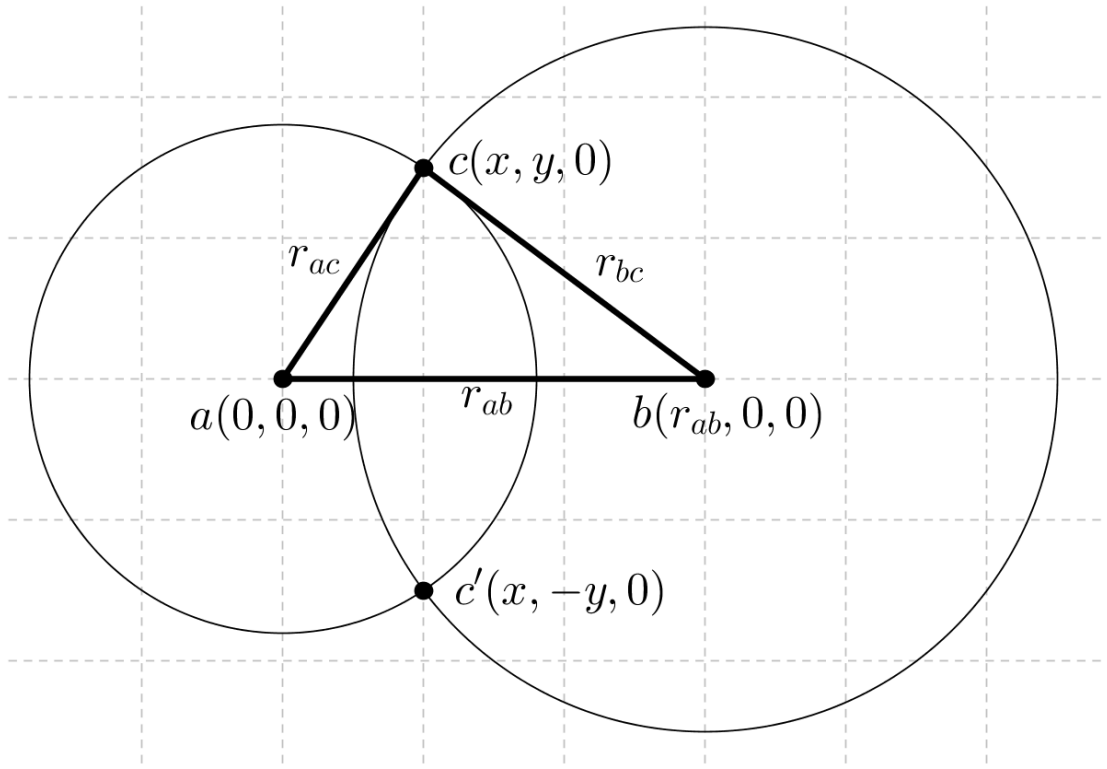


Figure 3.1: Assigning positions to the seed nodes

The first node, a is placed on the origin.

$$a.LPos \leftarrow (0, 0, 0) \quad (1)$$

Then, we place b on the positive side of the x -axis, at distance r_{ab} to a .

$$b.LPos \leftarrow (r_{ab}, 0, 0) \quad (2)$$

In Figure 3.1, we see three nodes a , b and c . As given in (1) and (2), we assign the coordinates to a and b . The coordinates of c is yet to be computed and shown as (x, y) .

After assigning coordinates to a and b , we compute the two intersections as follows.

$$x^2 + y^2 = r_{ac}^2 \quad (3)$$

$$(x - r_{ab})^2 + y^2 = r_{bc}^2 \quad (4)$$

If we combine (3) and (4), then we have

$$x^2 - 2 * r_{ab} * x + r_{ab}^2 - x^2 = r_{bc}^2 - r_{ac}^2 \quad (5)$$

$$x = \frac{r_{ab}^2 - r_{bc}^2 + r_{ac}^2}{2 * r_{ab}} \quad (6)$$

Using (5), we compute x coordinate of the node c as in (6). From (3), we obtain;

$$y^2 = r_{ac}^2 - x^2 \quad (7)$$

$$y = \pm \sqrt{\frac{4 * r_{ab}^2 * r_{ac}^2 - (r_{ab}^2 - r_{bc}^2 + r_{ac}^2)^2}{4 * r_{ab}^2}} \quad (8)$$

Notice that there are two possible values for the y coordinate. We pick the positive value. As a result, given three seeds a , b and c with the pairwise distances, we assign the following coordinates to the seeds:

$$a.LPos \leftarrow (0, 0, 0)$$

$$b.LPos \leftarrow (r_{ab}, 0, 0)$$

$$c.LPos \leftarrow (x, y, 0)$$

where

$$x = \frac{r_{ab}^2 - r_{bc}^2 + r_{ac}^2}{2 * r_{ab}}$$

$$y = \frac{\sqrt{4 * r_{ab}^2 * r_{ac}^2 - (r_{ab}^2 - r_{bc}^2 + r_{ac}^2)^2}}{2 * r_{ab}}$$

Before performing quadrilateration, we pick four seed nodes, a , b , c and d . Instead of forming a triangle, we form a tetrahedron in 3D. Hence, we take the intersection of three spheres. We place a , b and c onto $z = 0$ plane by using the computations (1) to (8). Figure 3.2 demonstrates the projection of the spheres on $z = 0$ plane. The centers of the spheres are a , b and c . The pairwise distances are the radii of the spheres and denoted by r_{ab} , r_{ac} , and r_{ad} .

In order to assign coordinates (x', y', z) to d , we make the following computations.

$$r_{ad}^2 = x'^2 + y'^2 + z^2 \quad (9)$$

$$r_{bd}^2 = (x' - r_{ab})^2 + y'^2 + z^2 \quad (10)$$

$$r_{cd}^2 = (x' - x)^2 + (y' - y)^2 + z^2 \quad (11)$$

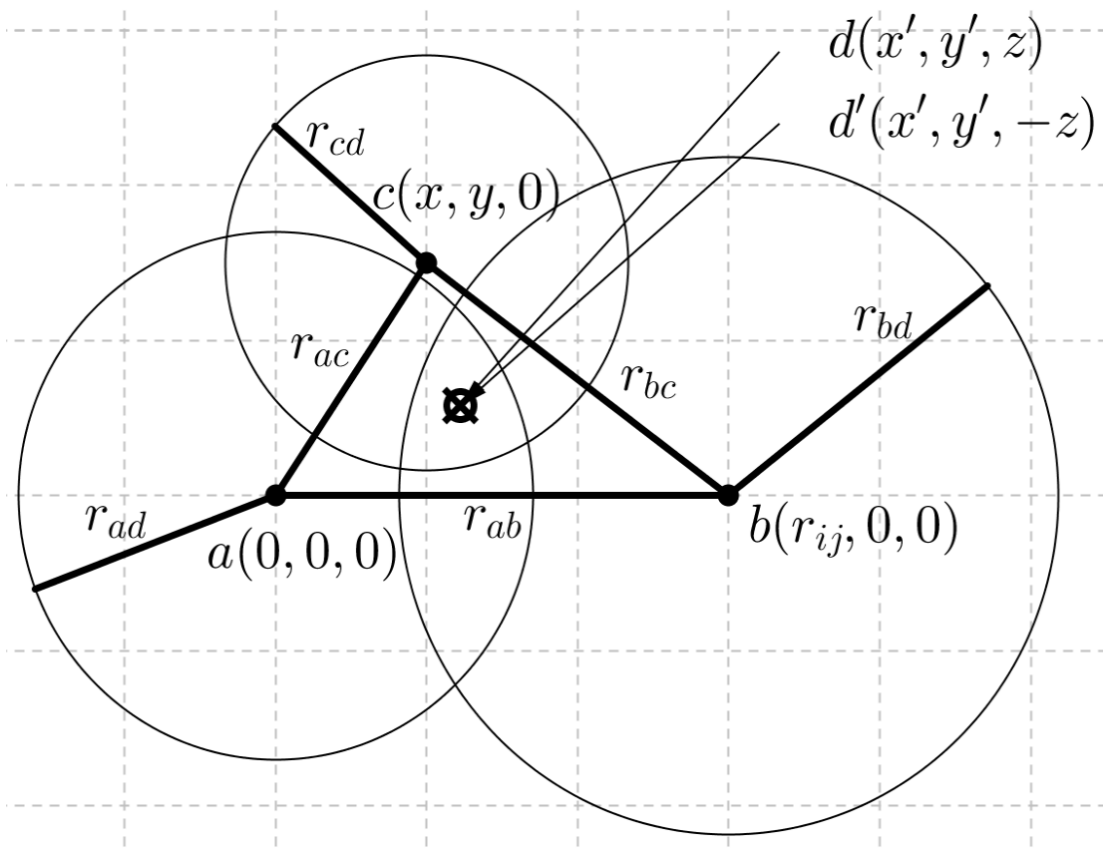


Figure 3.2: The projection of three spheres with centers a , b , c and d to the $z = 0$ plane.

If we subtract two equations (9) - (10), then we have

$$r_{ad}^2 - r_{bd}^2 = (x')^2 - (x' - r_{ab})^2 \quad (12)$$

$$= 2 * x' * r_{ab} - r_{ab}^2 \quad (13)$$

then we can compute x' and y' as follows.

$$x' = \frac{r_{ad}^2 - r_{bd}^2 + r_{ab}^2}{2 * r_{ab}} \quad (14)$$

$$y' = \frac{r_{ad}^2 - r_{cd}^2 - (x')^2 + (x' - x)^2 + y^2}{2 * y} \quad (15)$$

Using the Pythagorean theorem, we can compute z

$$z = \pm \sqrt{r_{ad}^2 - (x')^2 - (y')^2} \quad (16)$$

Similar to 2D case, we pick the positive value among two possible values for z coordinate. Hence, following the steps (1) to (10) we

$$a. GPos \leftarrow (0, 0, 0)$$

$$b. GPos \leftarrow (r_{ab}, 0, 0)$$

$$c. GPos \leftarrow (x, y, 0)$$

$$d. GPos \leftarrow (x', y', z)$$

where

$$x' = \frac{r_{ad}^2 - r_{bd}^2 + r_{ab}^2}{2 * r_{ab}}$$

$$y' = \frac{r_{ad}^2 - r_{cd}^2 - (x')^2 + (x' - x)^2 + y^2}{2 * y}$$

$$z = \sqrt{r_{ad}^2 - (x')^2 - (y')^2}$$

When three spheres intersect, there are two possible values for the z coordinate, leaving us with two symmetrical points with respect to the plane through

a , b and c . If we are localizing the point n in 3D, we need an extra distance measurement from a point which is not coplanar with a , b and c , to choose between two symmetrical points. The process of obtaining two candidate points, and then arbitrarily picking one is referred to as *semi-localization* of a node.

If we use a fourth distance to another localized node d , then we are able to pick the correct point p between p_1 and p_2 , assuming that the distances are accurate. However, if d is coplanar with a , b and c , both of the points will satisfy the distance constraint and we will not be able to tell if p_1 or $p_2 \neq p_1$ is the correct position for n .

$$p = \begin{cases} \text{NULL} & : \|\vec{p_1 d}\| = \|\vec{p_2 d}\| \\ p_1 & : \|\vec{p_1 d}\| = d(n, d) \\ p_2 & : \|\vec{p_2 d}\| = d(n, d) \end{cases} \quad (17)$$

where $d(n, d)$ is the Euclidean distance between the unlocalized node n and the fourth localized node d and $\|\vec{p_1 d}\|$ denotes the magnitude of the vector drawn from p_1 to the global position of d .

The last step assigns a unique value to the global position of unlocalized node n . Thus, we call this process *unique-localization* of a node in 3D.

DEFINITION 1 (Unique-localization of a node) *A node n is said to be uniquely localized with respect to four other localized nodes a , b , c and d such that $a \neq b \neq c \neq d$, if there is only one possible position of n with respect to the global positions of a , b , c and d .*

Assigning a position to a single node is similar in trilateration and quadrilateration. When a node has two candidate positions, which can be obtained by intersecting two circles in 2D and intersecting three spheres in 3D, a point can be picked by using (17). However, if we consider that the distance measurements might be noisy, a point qualifies as a solution if it is inside a certain margin with respect to the distances as shown in Figure 3.3 where p_1 and p_2 are the candidate points, r is the distance to the last node used to eliminate one of the solutions, and \mathcal{E} is the error magnitude.

$$\sigma(p_1, p_2, r) = \begin{cases} \text{NULL}, & \text{if } \left| \|\vec{p_1 d}\| - r \right| \leq r * \mathcal{E} \wedge \left| \|\vec{p_2 d}\| - r \right| \leq r * \mathcal{E} \\ p_1, & \text{if } \left| \|\vec{p_1 d}\| - r \right| \leq r * \mathcal{E} \\ p_2, & \text{if } \left| \|\vec{p_2 d}\| - r \right| \leq r * \mathcal{E} \\ \text{NULL}, & \text{otherwise} \end{cases}$$

Figure 3.3: Picking a position among two candidate points

3.2 Trilateration and Quadrilateration

In this section, we present the algorithms that we choose to perform 2D and 3D localization. We use trilateration to find the local coordinates of the nodes with respect to their coplanar neighbors and quadrilateration to find the global positions of the nodes in a WSN graph.

In Figure 3.4a, we present trilateration algorithm. we first initialize the point formation \mathbb{F}_{best} , that stores the best 2D point formation of the nodes in line 1. Then, we iterate on each fully-connected non-collinear node triplet (a, b, c) from line 2 through line 17. In line 3, we initialize \mathbb{F} that stores the 3D point formation of G with respect to each seed triplet (a, b, c) . In line 4, we determine the local positions of a , b and c by forming a triangle as explained in equations (1) - (8). Then, we add a , b and c into a queue data structure $Q_{\text{localized}}$ that keeps track of the localized nodes. From line 6 through line 14, we obtain the 2D point formation \mathbb{F} of the given graph G with respect to the seed nodes a , b and c . In line 7, we remove the node i from $Q_{\text{localized}}$ and notify its neighbors. Notifying a node in this context means increasing its localized neighbor count by one. In line 10, we check if j has are more than three localized neighbors. Hence, as last two steps, we check if j is collinear with these neighbors in line 11 and try to localize the node in line 12. If so, then we add j into the queue of localized nodes in line 13. When there are no more nodes left to be localized, we check if all the nodes in the graph are localized. If so, then we return the obtained point formation in line 15. In line 16, we check if the obtained point formation \mathbb{F} contains more nodes than the best point formation \mathbb{F}_{best} . If so, then we declare the new best point formation as the current one in line 16. In line 18, we return the best point formation with respect to the number of nodes.

In Figure 3.4b, we present quadrilateration algorithm. we first initialize the point formation \mathbb{F}_{best} , that stores the best 3D point formation of the nodes in line 1. Then, we iterate on each fully-connected and non-coplanar node triplet (a, b, c, d) from line 2 through line 17. In line 3, we initialize \mathbb{F} that stores the 3D point formation of G with respect to each seed quadruplet (a, b, c, d) . In line 4, we determine the global positions of a , b , c and d by forming a tetrahedron as explained in equations (1) - (16). Then, we add a , b , c and d into a queue data structure $Q_{\text{localized}}$ that keeps track of the localized nodes. From line 6 through line 14, we obtain the 3D point formation \mathbb{F} of the given graph G with respect to the seed nodes a , b , c and d . In line 7, we remove the node i from $Q_{\text{localized}}$ and notify its neighbors. Notifying a node in this context means increasing its localized neighbor count by one. In line 10, we check if j has are more than four localized neighbors. Hence, as last two steps, we check if j is coplanar with these neighbors in line 11 and try to localize the node in line 12. If so, then we add j into the queue of localized nodes in line 13. When there are no more nodes left to be localized, we check if all the nodes in the graph are localized. If so, then we return the obtained point formation in line 16. In line 16, we check if the obtained point formation \mathbb{F} contains more nodes than the best point formation \mathbb{F}_{best} . If so, then we declare the new best point formation as the current one in

line 15. In line 18, we return the best point formation with respect to the number of nodes.

REMARK 1 *For a given graph $G = (V, E)$, finding a trilateration ordering consists of $O(|V|^3)$ phases in the worst case. As three distance measurements from three non-collinear nodes are sufficient to localize a node, $O(|E|)$ time is spent in each phase by using a queue data structure keeping the track of localized nodes. This can be accomplished by keeping track of the number of localized neighbors of every node and inserting a node when the count hits a score of three. An effective three is the smallest count ≥ 3 with three non-collinear neighbors. On reaching the count three for the first time check collinearity. From then on, every new increment check only the new with any two previous. Therefore, it terminates after at most after $O(|V|^3 * |E|)$ steps. Quadrilateration, on the other hand, tries every possible node quadruplet as seed in order to achieve its highest possible localization percentage. Therefore, it runs in $O(|V|^4 * |E|)$ time in the worst case.*

We use CBL as the extension of the algorithms presented in Figures 3.4a and 3.4b. In the following section, we conduct simulations to test trilateration and quadrilateration to observe the effect of error on the precision of localizations achieved by both algorithms.

```

input: Graph  $G = (V, E)$ 
output: 2D point formation of  $G$ 
TRILATERATION( $G$ )
1:  $\mathbb{F}_{\text{best}} \leftarrow ([ ], [ ])$  /* Best point formation */
2: for each non-collinear and fully-connected  $\{a, b, c\} \subseteq V_i$  do
3:    $\mathbb{F} \leftarrow ([ ], [ ])$ 
4:   Form triangle using  $a, b, c$  /* Equations (1) - (8) */
5:    $Q_{\text{localized}} \leftarrow [a, b, c]$ 
6:   while  $Q_{\text{localized}}$  is not empty do
7:      $i \leftarrow \text{dequeue}(Q_{\text{localized}})$ 
8:     add  $(i, i.LPos)$  into  $\mathbb{F}$ 
9:     for all neighbors  $j$  of  $i$  do
10:      if  $(++j.Count) \geq 3$  then
11:        if  $j.LocalizedNeighbors(1, 2, last)$  are not collinear then
12:          if  $j$  can be localized in 2D then
13:             $\text{enqueue}(j, Q_{\text{localized}})$ 
14:      end while
15:      if  $|\mathbb{F}^1| = |V|$  then return  $\mathbb{F}$ 
16:      if  $|\mathbb{F}^1| > |\mathbb{F}_{\text{best}}^1|$  then  $\mathbb{F}_{\text{best}} \leftarrow \mathbb{F}$ 
17: end for
18: return  $\mathbb{F}_{\text{best}}$ 

```

(a) Trilateration

```

input: Graph  $G = (V, E)$ 
output: 3D point formation of  $G$ 
QUADRILATERATION( $G$ )
1:  $\mathbb{F}_{\text{best}} \leftarrow ([ ], [ ])$  /* Best point formation */
2: for each non-coplanar and fully-connected  $\{a, b, c, d\} \subseteq V_i$  do
3:    $\mathbb{F} \leftarrow ([ ], [ ])$ 
4:   Form tetrahedron using  $a, b, c, d$  /* Equations (1) - (16) */
5:    $Q_{\text{localized}} \leftarrow [a, b, c, d]$ 
6:   while  $Q_{\text{localized}}$  is not empty do
7:      $i \leftarrow \text{dequeue}(Q_{\text{localized}})$ 
8:     add  $(i, i.GPos)$  into  $\mathbb{F}$ 
9:     for all neighbors  $j$  of  $i$  do
10:      if  $(++j.Count) \geq 4$  then
11:        if  $j.LocalizedNeighbors(1, 2, 3, last)$  are not coplanar then
12:          if  $j$  can be localized in 3D then
13:             $\text{enqueue}(j, Q_{\text{localized}})$ 
14:      end while
15:      if  $|\mathbb{F}^1| = |V|$  then return  $\mathbb{F}$ 
16:      if  $|\mathbb{F}^1| > |\mathbb{F}_{\text{best}}^1|$  then  $\mathbb{F}_{\text{best}} \leftarrow \mathbb{F}$ 
17: end for
18: return  $\mathbb{F}_{\text{best}}$ 

```

(b) Quadrilateration

Figure 3.4: Trilateration (a) and quadrilateration (b) algorithms

3.2.1 Experimental Evaluation of Trilateration and Quadrilateration

In this section, we conduct experiments to test the performance trilateration and quadrilateration when there is a random deployment. For the experiments with trilateration, we deploy 100 sensors randomly in a 100×100 units square with a uniform distribution. For the experiments with quadrilateration, we deploy the same number of sensors randomly inside a $100 \times 100 \times 100$ units cube with a uniform distribution.

We take the average node connectivity *i.e.* average number of neighbors per node as the control value for the tests. We run 1000 tests for each connectivity value, and report the average of the recall percentage. In Figure 3.5, we test both algorithms with noiseless range measurements, and observe the changes in recall percentages with respect to the increasing node connectivity.

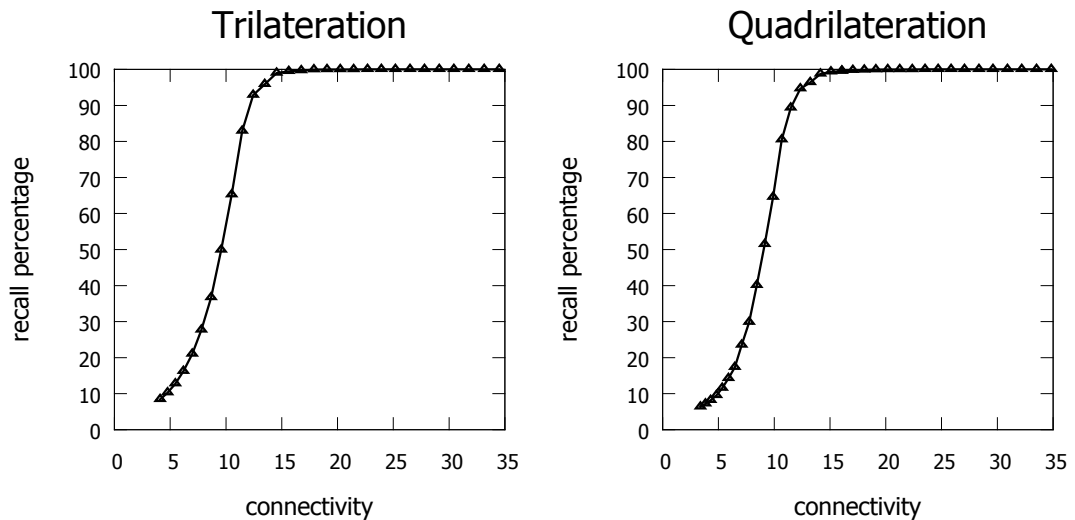


Figure 3.5: The recall percentages of trilateration (a) and quadrilateration (b) with respect to increasing node connectivity

The figure tells us that both trilateration and quadrilateration require an average connectivity of 15 neighbors per node. Now, let us check the average offsets of both algorithms with increasing error magnitude. We use the average connectivity at 15 neighbors per node, which is the value for both algorithms to reach their maximum recall percentage. For trilateration, we use robust triangles [43] to reduce the number of the flips. In Figure 3.6, we see the average offsets for both algorithms with increasing error magnitude.

The figure tells us that the error has more impact in 3D compared to 2D. Therefore, we expect that when we limit the use of quadrilateration by running trilateration to find the local positions of the nodes, the average offset will drop.

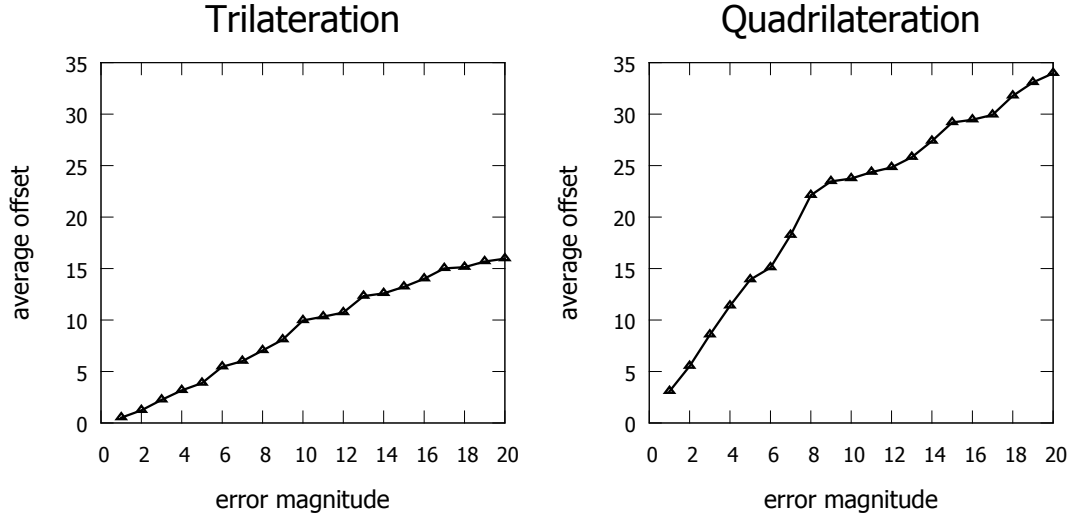


Figure 3.6: The average offsets of trilateration (a) and quadrilateration (b) with respect to increasing error magnitude when the average node connectivity is 15

3.3 Localizing the Coplanar Node Clusters

In this section, we tackle the problem of localizing a WSN in 3D where there is a planar deployment, and the information on the coplanar clusters is given apriori. We first define the problem of localizing the coplanar node clusters, and then propose an algorithm called Coplanarity Based Localization (CBL) to solve the corresponding problem.

3.3.1 Problem Definition

The problem of localizing clusters is finding the 3D point formation of a WSN graph $G = (V, E)$ which is partitioned into k subgraphs within each of which the nodes are coplanar. These partitions are called *coplanar clusters* and stored in a set $CSet = \{C_1 = (V_1, E_1), C_2 = (V_2, E_2), \dots, C_k = (V_k, E_k)\}$ with the following properties.

$$\begin{aligned}
\bigcup_{i \in [1, k]} V_i &= V \\
\bigcup_{i \in [1, k]} E_i &\subset E \\
\bigcap_{i \in [1, k]} V_i &= \emptyset \\
\bigcap_{i \in [1, k]} E_i &= \emptyset \\
\forall a \neq b \in V_i, (a, b) \in E &\Leftrightarrow (a, b) \in E_i
\end{aligned}$$

Two nodes v, w in the vertex set V_i of a coplanar cluster $C_i = (V_i, E_i)$ are called *coplanar neighbors*, and the edge (w, v) is called a *coplanar edge*. In an effort to localize G in 3D, we first localize the nodes in each cluster C_i in 2D with respect to their coplanar neighbors and then find the positions of the coplanar clusters in 3-space. The coplanarity causes difficulties for traditional multilateration methods because of flip ambiguities. Therefore, we propose a localization algorithm with potential to increase the precision of the localization when there is a planar deployment. This algorithm is called Coplanarity Based Localization (CBL). CBL obviously assumes the existence of coplanar node clusters given as the part of the input. Our approach has two main advantages over the known 3D localization techniques:

- i) We break the WSN localization problem of size n in 3D into k 2D localization problems.
- ii) By using the clustering information, we avoid using coplanar nodes to localize another node that may lead to a flip ambiguity.

The algorithm that we propose to find the 3D point formation of a WSN graph G is presented in Section 3.3.2

3.3.2 The CBL Algorithm

In this section, we present the proposed algorithm CBL to localize a WSN in 3D when there is a planar deployment. CBL is presented as an extension of quadrilateration with the ability to exploit the information on coplanar clusters. It should be noted that CBL is orthogonal to the localization algorithm used at its core. In other words, it will work with algorithms other than quadrilateration.

As the clustering information is assumed to be available, we can apply known 2D localization techniques in each cluster. We use trilateration presented in Figure 3.4a as the 2D localization algorithm. Localizing a coplanar cluster means computing the equation of the plane that the cluster is on. Thus, it is enough

to find the global positions of three locally positioned nodes from a cluster to compute the equation of the plane. These three nodes are called *support nodes* of the cluster. Using the global positions of the support nodes, all the remaining locally positioned nodes in a coplanar cluster can be globally positioned by a simple transformation with respect to the support nodes as soon as the support nodes are globally positioned.

In Figure 3.7, we see an example transformation of a coplanar cluster. Both the local and the global positions of the support nodes are shown in red they are marked with arrows. The dashed arrows indicate the transformation from $z = 0$ plane to the actual plane that the cluster is on.

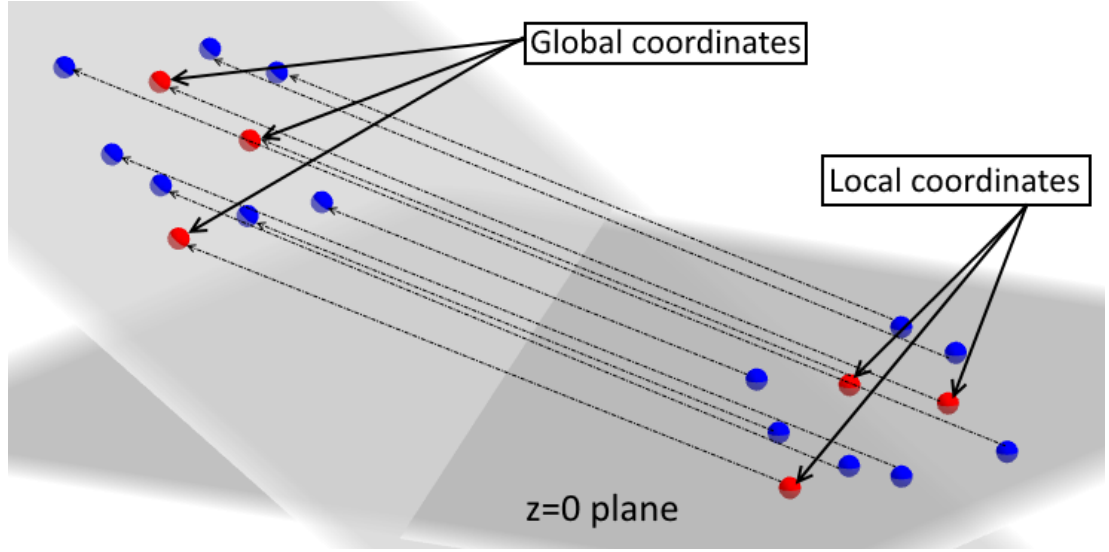


Figure 3.7: Transformation of a coplanar cluster

The transformation process is as follows. Given three support nodes a , b and c , let us denote the local coordinates of these nodes as a_{local} , b_{local} and c_{local} respectively. The global coordinates of these nodes are denoted by a_{global} , b_{global} and c_{global} respectively. In order to build a transformation matrix, we use a common coordinate system. Hence, we pick the centroid of the local coordinates, CL , as the origin of local coordinate system and the centroid of the global coordinates, CG , as the origin of the global coordinate system. The x -axis of the local coordinate system is

$$\vec{x}_{\text{local}} = \frac{\overrightarrow{(CL)(a_{\text{local}})}}{\|\overrightarrow{(CL)(a_{\text{local}})}\|} \quad (18)$$

where $\|\overrightarrow{(CL)(i_{\text{local}})}\|$ denotes the magnitude of the vector drawn from the origin of the local coordinate system to the local coordinates of i . Then, we find a vector that is orthogonal to x -axis by first finding

$$\vec{v} = \frac{\overrightarrow{(CL)(b_{\text{local}})}}{\|\overrightarrow{(CL)(b_{\text{local}})}\|} \quad (19)$$

This vector is used to determine the z -axis of the local coordinate system, which is orthogonal to \vec{x} and \vec{v} .

$$\vec{z}_{\text{local}} = \frac{\overrightarrow{x_{\text{local}}} \times \vec{v}}{\|\overrightarrow{x_{\text{local}}} \times \vec{v}\|} \quad (20)$$

Finally, we compute the y -axis of the local coordinate system, which is certainly orthogonal to x -axis and z -axis.

$$\vec{y}_{\text{local}} = \overrightarrow{x_{\text{local}}} \times \vec{z}_{\text{local}} \quad (21)$$

The matrix of the local coordinate system is called *input matrix* and computed as follows.

$$M_{\text{input}} = \begin{bmatrix} x_{\text{local}}^1 & y_{\text{local}}^1 & z_{\text{local}}^1 \\ x_{\text{local}}^2 & y_{\text{local}}^2 & z_{\text{local}}^2 \\ x_{\text{local}}^3 & y_{\text{local}}^3 & z_{\text{local}}^3 \end{bmatrix} \quad (22)$$

where the columns of the matrix are the transpositions of $\overrightarrow{x_{\text{local}}}$, $\overrightarrow{y_{\text{local}}}$ and $\overrightarrow{z_{\text{local}}}$

The global coordinate system is computed similar to steps (18) - (22).

$$\vec{x}_{\text{global}} = \frac{\overrightarrow{(CG)(a_{\text{global}})}}{\|\overrightarrow{(CG)(a_{\text{global}})}\|} \quad (23)$$

$$\vec{v} = \frac{\overrightarrow{(CG)(b_{\text{global}})}}{\|\overrightarrow{(CG)(b_{\text{global}})}\|} \quad (24)$$

$$\vec{z}_{\text{global}} = \frac{\overrightarrow{x_{\text{global}}} \times \vec{v}}{\|\overrightarrow{x_{\text{global}}} \times \vec{v}\|} \quad (25)$$

$$\vec{y}_{\text{global}} = \overrightarrow{x_{\text{global}}} \times \vec{z}_{\text{global}} \quad (26)$$

$$M_{\text{output}} = \begin{bmatrix} x_{\text{global}}^1 & y_{\text{global}}^1 & z_{\text{global}}^1 \\ x_{\text{global}}^2 & y_{\text{global}}^2 & z_{\text{global}}^2 \\ x_{\text{global}}^3 & y_{\text{global}}^3 & z_{\text{global}}^3 \end{bmatrix} \quad (27)$$

The transformation matrix is computed using M_{input} and M_{output} by making the following matrix multiplication.

$$[M_{\text{transform}}] = [M_{\text{output}}] [M_{\text{input}}]^T \quad (28)$$

where $[M_{\text{input}}]^T$ denotes the transposition of the input matrix.

After we build the transformation matrix, we apply the transformation to each locally positioned node n with local coordinates n_{local} . We compute the global coordinates of node n_{global} by applying the transformation, we use the origins of the local and the global coordinate systems, CL and CG .

$$n_{\text{global}} = [M_{\text{transform}}] \cdot (\overrightarrow{n_{\text{local}}} - \overrightarrow{CL}) + \overrightarrow{CG} \quad (29)$$

where the origins CL and CG are treated as vectors from the point $(0, 0, 0)$.

At the beginning of the localization process, after a cluster is picked as seed, the nodes in that cluster cannot be used to perform a unique localization of a node. Therefore, we are only left with the option of semi-localization of a node. If a cluster is localized based on a semi-localized support node, we refer to that cluster as a *semi-localized cluster*.

DEFINITION 2 (Semi-localization of a cluster) *A coplanar cluster $C_i = (V_i, E_i)$ is said to be semi-localized if one of the support nodes in V_i is semi-localized while the other two are uniquely localized.*

Once we are alone with a seed cluster and a semi-localized cluster, we can localize a third cluster as given by the following definition.

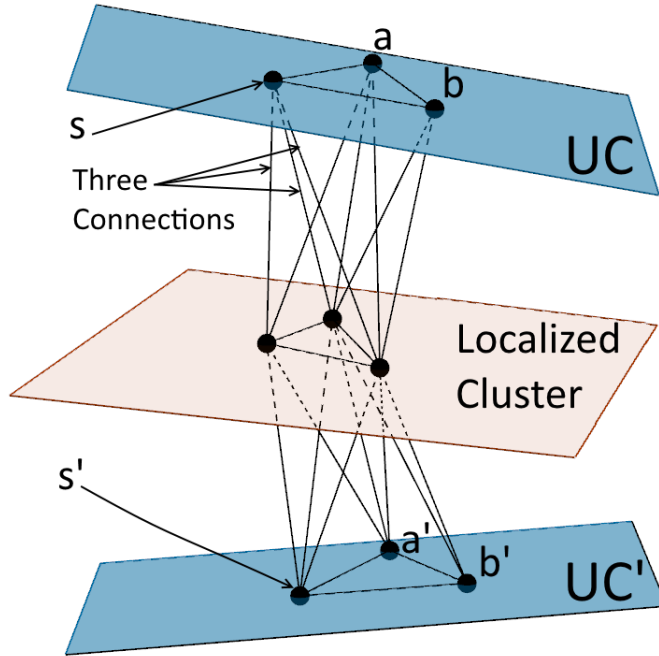
DEFINITION 3 (Rigid-localization of a cluster) *A cluster $C = (V_i, E_i)$ is said to be rigidly localized if all the three support nodes in V_i are uniquely localized.*

In Figure 3.8a, we see the semi-localization of an unlocalized coplanar cluster UC . The support nodes of UC are indicated by letters s , a and b . After picking one of the possible positions for the first support node, denoted by s and s' , the remaining support nodes a and b can be uniquely localized with respect to s and three other localized nodes from localized cluster.

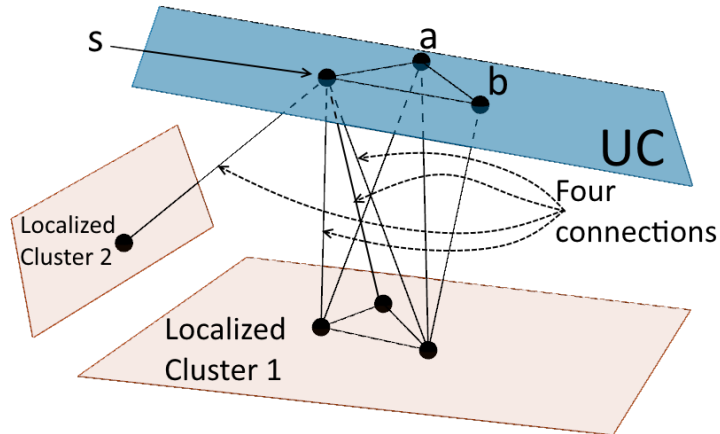
REMARK 2 *The only possible combinations of positions of the support nodes in Figure 3.8a are either (s, a, b) or (s', a', b') , which are reflections around the localized cluster.*

REMARK 3 *The rigid-localization of a cluster cannot be performed unless the number of already localized clusters is $m \geq 2$, and one coplanar cluster is on a different plane than at least one of the remaining $m - 1$.*

In Figure 3.8b, we see that the support node s has one extra localized neighbor which is not coplanar with the rest. Therefore, s can be uniquely localized, determining unique positions for its coplanar neighbors. In order to localize all the nodes, we have to perform rigid-localization of all the remaining clusters.



(a) Semi-localization of a cluster



(b) Rigid-localization of a cluster

Figure 3.8: Semi-localization (a), and rigid-localization (b) of an unlocalized cluster

We present CBL in Figure 3.9. The algorithm takes a WSN graph $G = (V, E)$ as the input and returns the estimated 3D point formation of G . In order to localize the maximum possible number of nodes *i.e.* to reach the maximum recall percentage possible, the algorithm tries every possible coplanar cluster pair as the seed cluster C_{seed} and semi-localized cluster C_{semi} . In line 1, we find the local positions of the nodes with respect to their coplanar neighbors. We initialize the point formation \mathbb{F}_{best} that stores the best 3D point formation of the nodes in line

```

input: WSN graph  $G = (V, E)$ , coplanar cluster set  $CSet = \{C_1, C_2, \dots, C_k\}$ 
output: 3D point formation of  $G$ 
CBL( $G, CSet$ )
1: find local positions of the nodes using trilateration
2:  $\mathbb{F}_{\text{best}} \leftarrow ([ ], [ ])$  /*Best point formation*/
3: for all  $C_{\text{seed}} \in CSet$  do
4:   for each  $C_{\text{semi}} \neq C_{\text{seed}} \in CSet$  do
5:     if  $C_{\text{semi}}$  can be semi-localized with respect to  $C_{\text{seed}}$  then
6:       /*See Figure 3.8a*/
7:        $Q_{\text{localized}} \leftarrow [C_{\text{seed}}, C_{\text{semi}}]$ 
8:        $\mathbb{F} \leftarrow ([ ], [ ])$ 
9:       while  $Q_{\text{localized}}$  is not empty do
10:         $C_i \leftarrow \text{dequeue}(Q_{\text{localized}})$ 
11:        for all nodes  $v \in V_i$  do
12:          if  $v$  is globally positioned then
13:            add  $(v, v.GPos)$  into  $\mathbb{F}$ 
14:            for all neighbors  $w$  of  $v$  do
15:              /* $C_w \leftarrow \text{ClusterOf}(w)$ */
16:              if  $++w.Count$  changes the state of  $C_w$  to rigid then
17:                /* $C_w$  can be rigid-localized (see Figure 3.8b)*/
18:                 $\text{enqueue}(j, Q_{\text{localized}})$ 
19:            end for
20:          end if
21:        end for
22:      end while
23:      if all the coplanar clusters are localized then return  $\mathbb{F}$ 
24:      if  $|\mathbb{F}^1| > |\mathbb{F}_{\text{best}}^1|$  then  $\mathbb{F}_{\text{best}} \leftarrow \mathbb{F}$ 
25:    end if
26:  end for
27: end forreturn  $\mathbb{F}_{\text{best}}$ 

```

Figure 3.9: CBL algorithm

2. Then, we iterate on each coplanar cluster C_{seed} from line 3 through line 23. At each iteration, we pick C_{seed} as the seed cluster. In line 4, we iterate on the coplanar clusters except the seed cluster. If we find a cluster C_{semi} that can be semi-localized with respect to the seed cluster in line 5, then we add C_{seed} and C_{semi} into a queue data structure $Q_{localized}$ that keeps track of the localized nodes in line 6. Then, we initialize \mathbb{F} that stores the 3D point formation of G with respect to each (C_{seed}, C_{semi}) pair. When we find a seed and a semi-localized cluster, we perform rigid-localization for the remaining coplanar clusters, from line 8 through line 19. In line 9, we remove the cluster $C_i = (V_i, E_i)$ from $Q_{localized}$. We iterate on each node v in the vertex set V_i of cluster C_i in line 10. If v is globally positioned, then we add v and its global position into \mathbb{F} . Then, we notify the neighbors w of v from line 13 through line 16. Notifying a node in this context means increasing its localized neighbor count by one. In line 14, we first increase the localized neighbor count of w and then check if the updated value leads to the rigid-localization of the coplanar cluster of w . If so, then we add the cluster of w into $Q_{localized}$ in line 15. When there are no more coplanar clusters left to be localized, we check if all the clusters in the graph are localized. If so, then we return the obtained point formation in line 20. In line 21, we check if the obtained point formation \mathbb{F} contains more nodes than the best point formation \mathbb{F}_{best} . If so, then we declare the new best point formation as the current one. In line 24, we return the best point formation with respect to the number of nodes.

REMARK 4 *By the same reasoning given in Remark 1, given a graph $G = (V, E)$ and k coplanar clusters, CBL works in $O(k^2 * |E|)$ time in the worst case.*

3.3.3 Experimental Evaluation of CBL

In this section, we present the experimental results on the performance of CBL in various environments. We conduct our simulations for the environments that show different types of characteristics in both planar clustering and localization phases. These characteristics are basically modeled by the planarity factor μ , calculated by dividing the number of coplanar clusters by the average number of nodes in each cluster and subtracting the result from 1.

$$\mu = 1 - \frac{k^2}{n}$$

where k is the number of nodes and n is the number of clusters.

We measure the quality of a localization with respect to the changes in recall percentage (*i.e.* the ratio of the localized nodes), and precision error (*i.e.* positional offset per node). The control parameter for the tests is picked as the magnitude of the environmental noise. It is verified experimentally that using the available information on coplanar clusters leads to a more precise localization than mere quadrilateration. While evaluating the experimental results, we refer to the tests where information of clusters is utilized as **CBL**. When the network

is localized by mere quadrilateration, we refer to these type of tests as **quadrilateration**. In mere quadrilateration, the clustering information is not used even though it is available.

We conduct simulations using Java [44] as the programming language and Eclipse IDE [45] as the development tool. We developed an in-house simulator to simulate the environment by placing static nodes with a uniform distribution probability in a $100 \times 100 \times 100$ unit cubic volume.

We use two main types of deployments: the deployments where the clusters are intersecting, and non-intersecting. In these two types of deployments, we use various number nodes, and various number of clusters. A deployment is named after the combination of the number of coplanar clusters, and the number of sensor nodes on each coplanar cluster separated by a dash in between. We also use the letter I or D at the beginning in order to indicate if the clusters are intersecting or disjoint. In Table 3.1, we give the deployments that we use along with the planarity factor of the deployment. We divide the planarity factors of the deployments into three. Namely, *low*, *average* and *high*. We say that the planarity factor is *high* when $\mu \in \{0.988, 0.985, 0.980, 0.960\}$, *average* when $\mu \in \{0.920, 0.900, 0.840\}$ and *low* when $\mu \in \{0.680, 0.360, 0.100, -0.800\}$.

High		Average		Low	
Deployment	μ	Deployment	μ	Deployment	μ
I-3-200	0.988	I-4-100	0.960	D-16-50	0.680
I-3-270	0.985	D-8-100	0.920	D-16-25	0.360
I-4-200	0.980	I-5-50	0.900	D-27-30	0.100
I-5-160	0.968	D-8-50	0.840	D-27-15	-0.800

Table 3.1: The names of the sub-type deployments with respect to the number of clusters and the number of nodes per cluster

The disjoint planar deployment is simulated with respect to the number of coplanar clusters, k . In order to create disjoint clusters, we first divide the original cube into k sub-cubes. In each sub-cube, we generate a random plane using the plane equation $ax + by + cz + d = 0$ and scatter equal number of nodes on these planes.

In Figure 3.10a, we see a cube divided into eight sub-cubes. In Figure 3.10b, the same cube is divided into nine sub-cubes

In our simulations, we use unit ball graph as the sensing model and assume that the network graph is always connected. We denote the sensing range by R . In order to model noisy range measurements, we use empirically gathered noise data [40, 41]. Each edge $(v, w) \in E$ in $G = (V, E)$ is modified with respect to a random number generated using a Gaussian random distribution with $N(f(R), \mathcal{E}/100)$ where \mathcal{E} is the magnitude of the error and R is the sensing range. $f(R)$ is defined

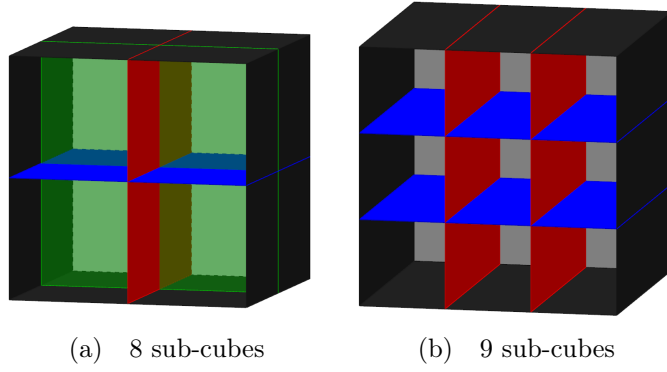


Figure 3.10: Slicing a cube with perpendicular planes

as follows;

$$f(R) = 0.022 \ln(1 + R) - 0.038$$

The large noise value is selected as $\mp P\%$ of the sensing range where P is generated with a uniform random process between 0 and 10 units with probability 0.05.

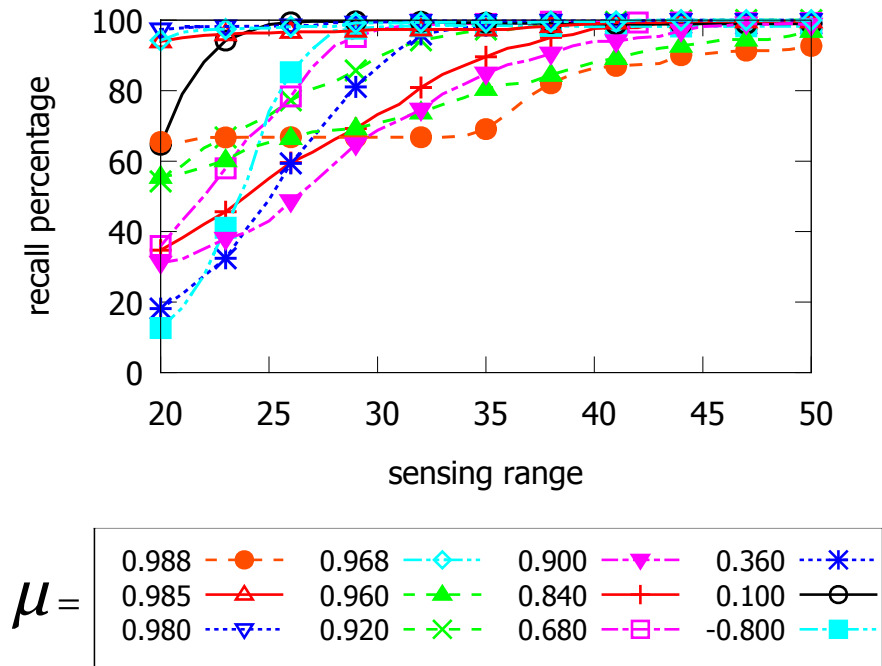
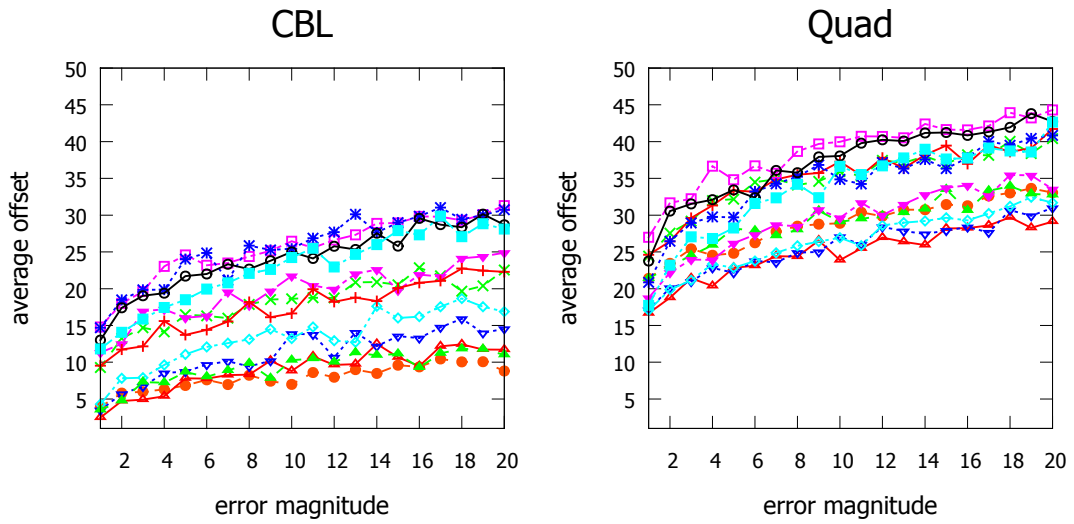
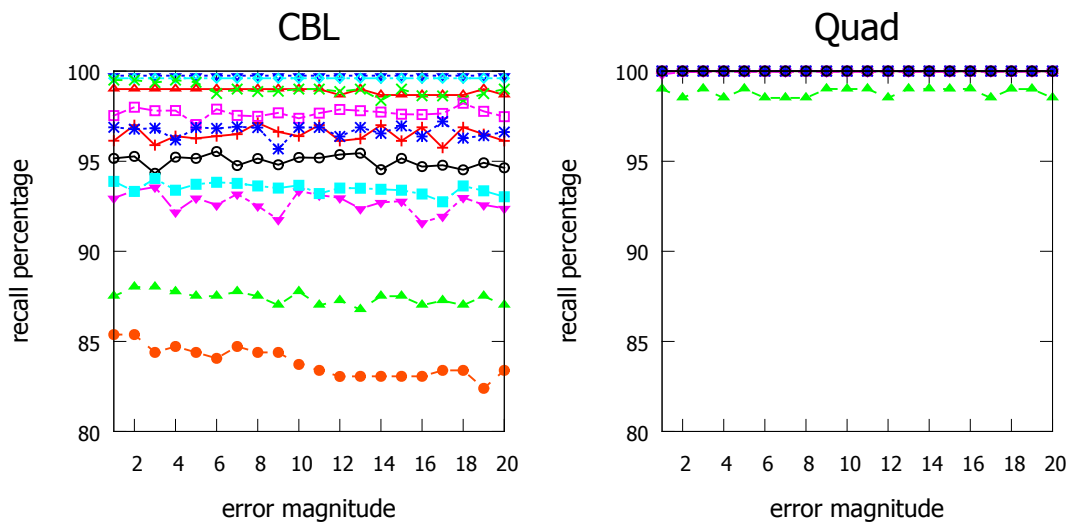


Figure 3.11: Recall percentage of CBL with noiseless distance measurements.

First, let us present the recall percentage of CBL with noiseless distance measurements in Figure 3.11. Based on the recall percentages, we pick 40 units as the sensing range for the tests with noisy range measurements, where over 80% of the nodes are localized for all values of μ .



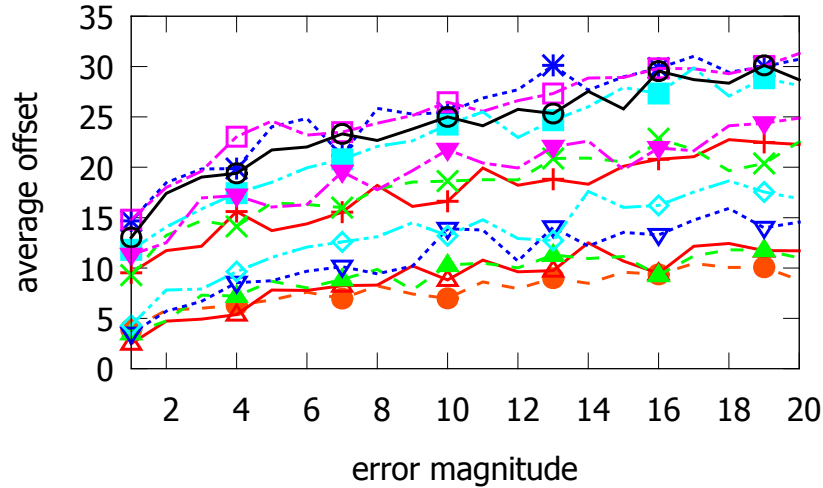
(a) Average offsets



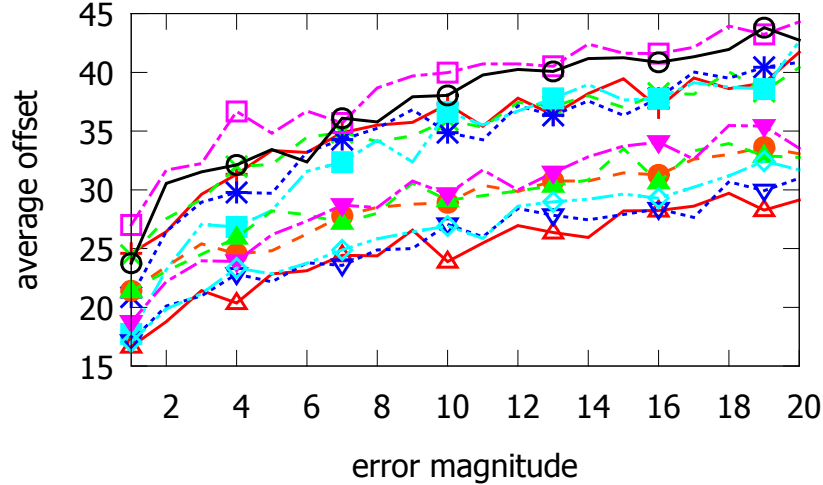
(b) Recall percentages

Figure 3.12: The change in average offsets (a), and recall percentages (b) of CBL and quadrilateration with respect to increasing error magnitude and various planarity factors when the sensing range is 40 units

In Figure 3.12a we see the average offsets and in Figure 3.12b we see the recall percentages when the sensing range is 40 units and the error magnitude takes values in range between 1 and 20. The figure tells us that exploiting the clustering information reduces the average offset of the localization. Even though the recall percentage of CBL is lower than quadrilateration, we can say that instead of localizing a node imprecisely, CBL performs a more precise localization than quadrilateration. Quadrilateration, on the other hand, is able to localize more nodes with more precision errors.



(a) CBL



(b) Quadrilateration

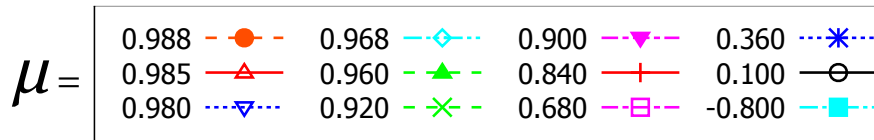


Figure 3.13: A closer view of the plots in Figure 3.12a

For a better comparison, we give the offset plots with a bigger scale in Figure 3.13. In Figure 3.13a, we see the average offset of CBL. In Figure 3.13b, we see the average offset of quadrilateration. We see that ignoring the available clustering

information increases the offset by 10 units per node in general. In Figure 3.13a, we see the average offset of CBL. When the planarity factor is high, CBL causes less offset than the other cases. Even though CBL is effected by the increasing error magnitude, the average offset stays under 20 units per node when $\mu \geq 0.960$. When the planarity factor is average, CBL localizes the network almost as precise as the low planarity factor case. However, when the error magnitude increases, the difference between two cases becomes more obvious. When the planarity factor is low, CBL causes the highest average offset with an offset of around 30 units per node. When the planarity factor is average, the average offset ranges between 15 units per node and 25 units per node.

3.4 Extracting the Coplanar Clusters

In this section, we deal with the case where there is a planar deployment and the information on coplanar clusters are not available in 3D WSN localization problem. First, we define the problem of extracting the coplanar clusters and then we propose a heuristic algorithm to solve the defined problem.

3.4.1 Problem Definition

Given a set of points in 3D with no position information, the problem of extracting the coplanar clusters is to cover these points with k planes based only on the available pairwise distances. Given a WSN graph $G = (V, E)$, the points correspond to the nodes in V and the pairwise distances are the edge set E . We refer to the partitions $\{C_1 = (V_1, E_1), C_2 = (V_2, E_2), \dots, C_k = (V_k, E_k)\}$ as *coplanar clusters*. Each node included into a coplanar cluster is called an *on-plane node* while the rest is called as *off-plane nodes*. In 2D, a variation of this problem is presented as "*Covering a Set of Points with a Minimum Number of Lines*" in [46]. The problem presented assumes that the coordinates of the points are known. Even though with known positions, it is proved to be inapproximable in [47]. Instead of the positions, we are only able to use pairwise distances. Considering that these distances might be inaccurate, it is conjectured that extracting the coplanar clusters is also a difficult problem.

For checking the coplanarity of four nodes, we use Cayley-Menger determinant (for the details on Cayley-Menger determinant usage, see Chapter 2). Trying to extract the coplanar clusters by using noisy pairwise distance measurements have two main challenges.

- i) Even though the points are coplanar, because of the inaccurate distances, Cayley-Menger determinant gives either positive or an imaginary number as its result. Therefore, we set a volume threshold κ to accept the results less than some value. As a result of experimental verification, this value is set as $6 * \ln(\mathcal{E} + 1)$ where \mathcal{E} is the error magnitude.

- ii) When the distance between a group of interplanar neighbors are too close, they might be interpreted as coplanar because of the volume threshold. In order to avoid this, also use a value called *hop distance* denoted by θ . The edges whose values are less than θ are unavailable during the extension process. This value is set to half the average edge weight, and increased by one at each iteration during the extension phase of the coplanar clusters.

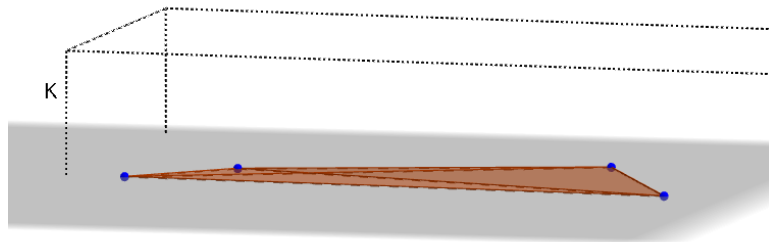
Setting the Volume Threshold κ

As the distance measurements might possibly be inaccurate, searching for a perfect plane, would be a waste of effort. Therefore, we set a threshold value, κ , for the volume of the tetrahedron formed by the six distance measurements among four nodes $\{a, b, c, d\}$. We say that a node subset $\{a, b, c, d\}$ are coplanar if $\mathcal{V}_{abcd} \leq \kappa$, where \mathcal{V}_{abcd} is the volume of the tetrahedron formed by a, b, c and d as given by Cayley-Menger determinant.

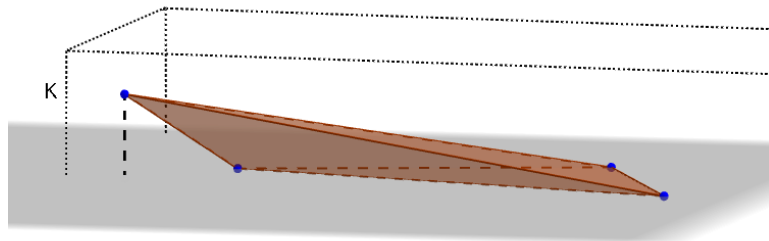
After finding a set of four coplanar nodes $\{a, b, c, d\}$, we create a coplanar cluster $C_i = (\{a, b, c, d\})$. Then we try to add more nodes into C_i by finding other nodes that are also coplanar with three of the nodes inside that cluster. This process is called *extending* that cluster. We add a node n into a cluster C_i if $V_{nabc} \leq \kappa$ such that $\{a, b, c\} \subset C_i$. When there is no node left to be added into C_i , we search for another cluster $C_j \neq C_i$, and then extend C_j as described above. We continue finding and extending clusters until there are no more off-plane nodes left. The whole operation is called *extracting the coplanar clusters* or *planar clustering*.

While extracting the coplanar node clusters, the volume threshold value needs to be set carefully. Otherwise, a coplanar cluster might be expanded to include nodes belonging to a completely different coplanar cluster. This situation arises particularly when coplanar clusters are deployed on many planes are very close to each other. In this case, it becomes a major issue to tell whether the variations in the volume of tetrahedra stem from the errors in the distance measurements or picking nodes in the vicinity of the intersection of actually different planes. In order to set a volume threshold, 10000 tests are run by placing four points a, b, c, d in a unit square. We take the average difference between actual volume $\mathcal{V}_{abcd}^{\text{act}}$ and the volume computed with the noisy distance measurements $\mathcal{V}_{abcd}^{\text{comp}}$. As a result, the volume threshold κ is defined as $6 * \ln(1 + \mathcal{E})$ where \mathcal{E} is the error magnitude.

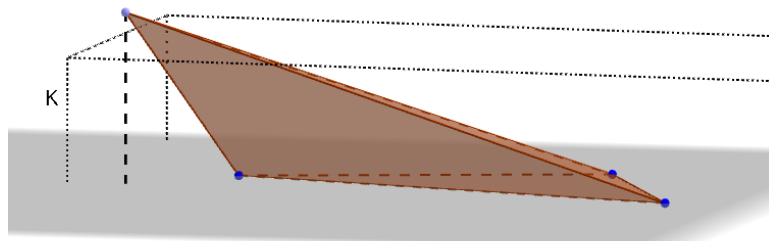
Figure 3.14 demonstrates three examples for the usage of volume threshold. In Figure 3.14a, there are four coplanar nodes. The volume of the tetrahedron formed by these nodes is zero. When the distance measurements are noisy, the same tetrahedron might be formed differently, as seen in Figure 3.14b. Even though the nodes in Figure 3.14b are not coplanar, they are treated as coplanar because the volume of the formed tetrahedron is less than the volume threshold. In Figure 3.14c, the volume of the tetrahedron formed by four nodes is bigger than the threshold. Thus, the four nodes are interpreted as non-coplanar nodes.



(a) A tetrahedron whose volume is zero



(b) A tetrahedron whose volume is between zero and the volume threshold



(c) A tetrahedron whose volume is greater than the volume threshold

Figure 3.14: A coplanar group of nodes (a), a non-coplanar group of nodes, interpreted as coplanar (b), a non-coplanar group of nodes, not interpreted as coplanar (c)

If a group of coplanar nodes are within the communication range of nodes from another coplanar cluster, the pairwise distances between interplanar neighbors might be too small. Thus, the volumes of the formed tetrahedra could be smaller than that specified by the volume threshold, even though the nodes are in different coplanar clusters. In Figure 3.15a, we see a planar deployment where there are 16 coplanar node clusters. Each different color indicates a coplanar cluster found. The cluster that is pointed at by a black arrow consists of nodes from actually three different coplanar node clusters. This kind of clustering leads to very high offsets since all the nodes in the found cluster will be localized imprecisely both in 2D and in 3D. Figure 3.15 demonstrates the improper expansion from two angles of view. In Figure 3.15b, three black arrows with numbers indicate three different clusters. In Figure 3.15c, we see the same deployment from a different angle of view.

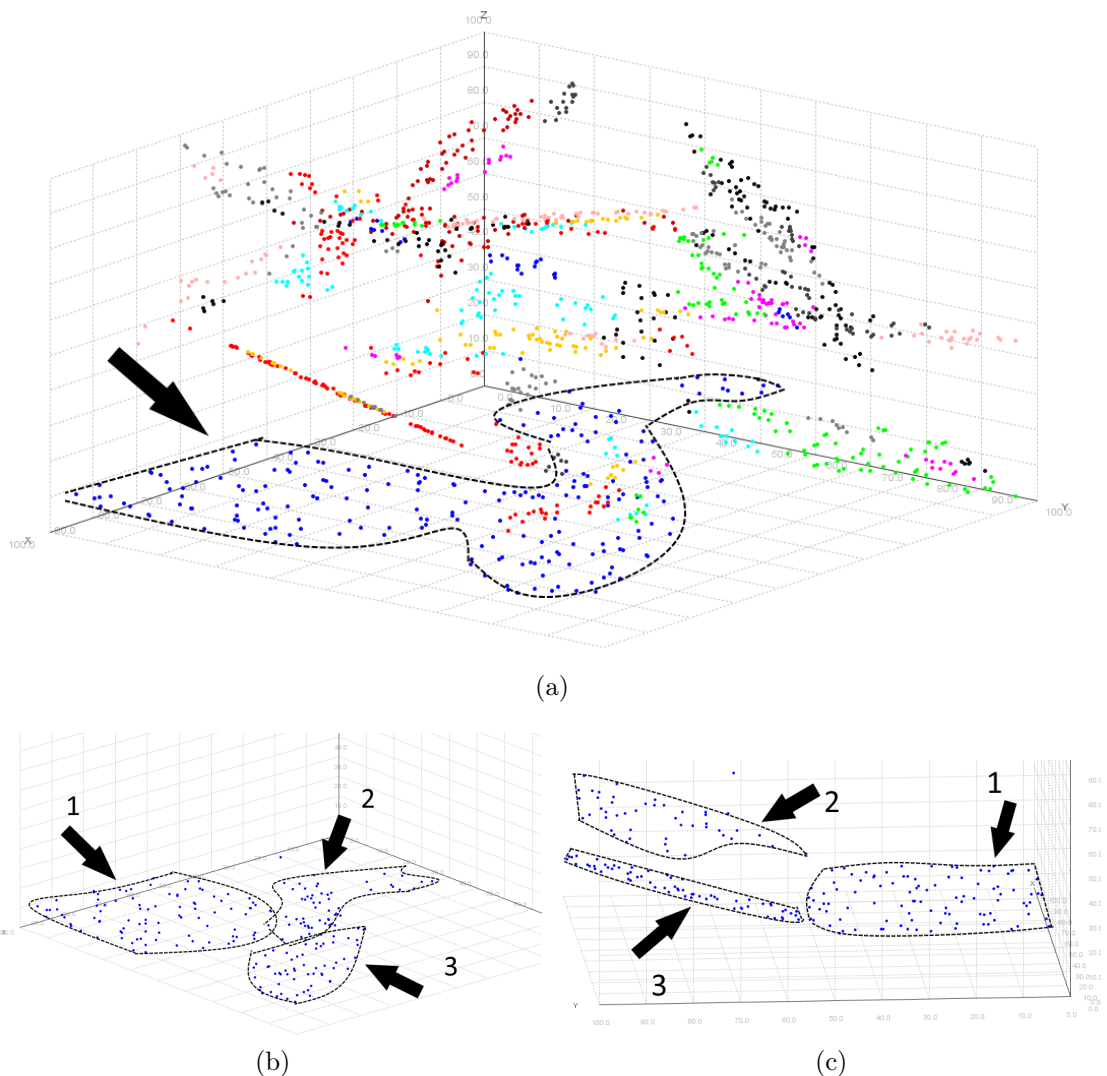


Figure 3.15: An example of inappropriate expansion from two angles of view

This type of extensions occur when the volume threshold is greater than zero and the interplanar edges are used frequently. In order to limit the usage of interplanar edges, we take the following precautions.

- We sort the nodes by their connectivities and seek for a set of four coplanar nodes among the nodes with the lowest connectivity. Picking the nodes with smaller number of neighbors reduces the possibility to use interplanar connections while extending a cluster. As seen in Figure 3.16, the node v has more neighbors since it is closer to another cluster while node w has coplanar neighbors only. The interplanar edges are indicated with dashed lines and coplanar edges are indicated with straight lines.
- We ignore the edges with higher weights than a pre-defined value called *hop distance* and denoted by θ . Notice that in Figure 3.16, the interplanar edges are usually longer than the coplanar edges. Thus, by setting a hop distance, we reduce the possibility of checking the coplanarity with interplanar neighbors.

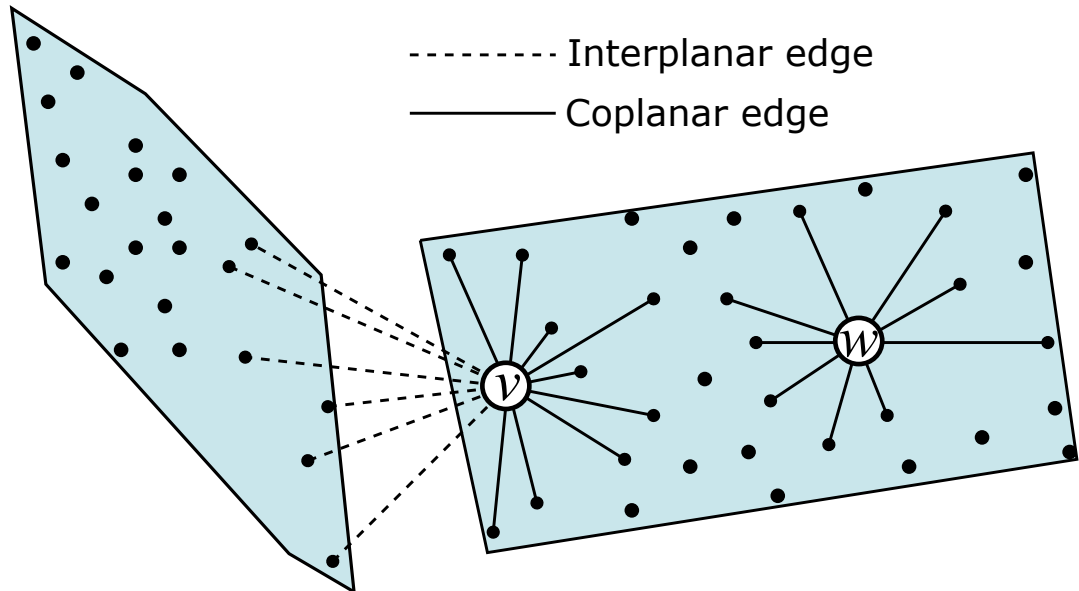


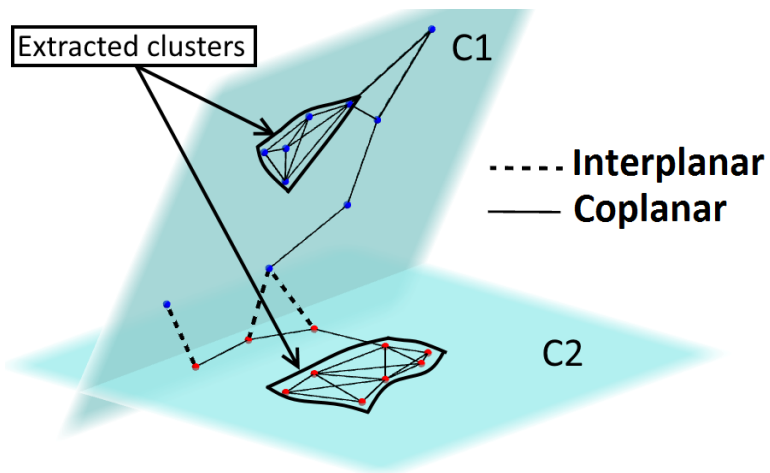
Figure 3.16: Two nodes from a coplanar cluster, and their neighbors

Setting the Hop Distance θ

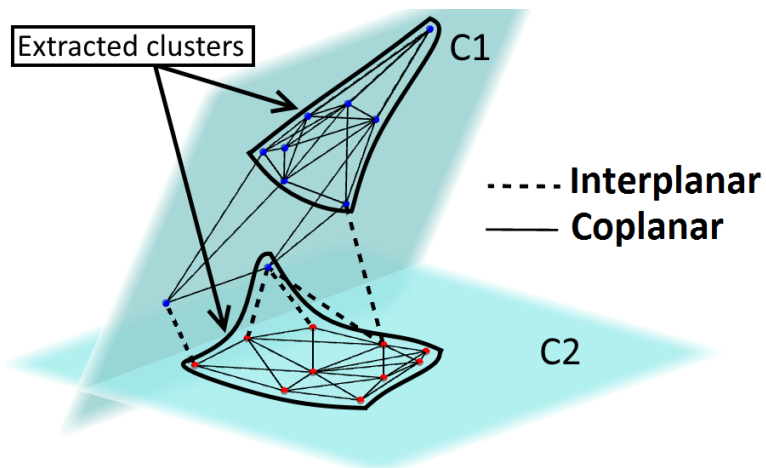
As well as the volume threshold, the hop distance is also very important to achieve an accurate extraction of coplanar clusters. Let us explain the effect of θ with an example. In Figure 3.17, we see two coplanar clusters, denoted by $C1$ and $C2$, with ten nodes on each. The coplanar edges are denoted by straight lines and interplanar edges are denoted by dashed lines. In Figure 3.17a, the hop distance is set as 10 units. Therefore, while extending a cluster, the distance values greater than 10 units are ignored in order to avoid the extension demonstrated in Figure 3.15. However, seven out of twenty nodes are interpreted as off-plane because of insufficient connectivity, even though they are in a coplanar cluster. In Figure 3.17b, the hop distance is set as 20 units. In this case, only two nodes are interpreted as off-plane. Notice that one of the nodes in $C1$ has three interplanar neighbors, and it forms a tetrahedron whose volume is less than the volume

threshold. Hence, it was included into wrong cluster. In Figure 3.17c, the hop distance is set to 30 units. Because of dense interplanar edges, after a small number of nodes from $C1$ is included into $C2$, the extension of $C2$ continues including the nodes in $C1$.

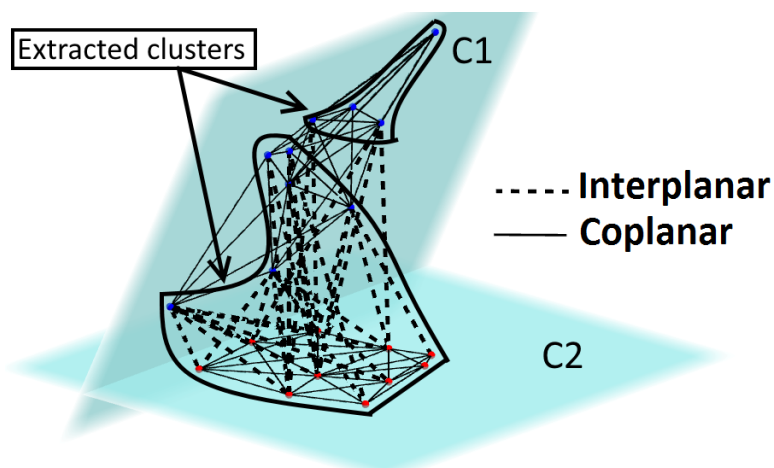
Considering the deployment given in Figure 3.17, we can say that for this specific deployment, 20 units is the best hop distance. However, without any information on the deployment, setting the hop distance as an arbitrary value may end up with extracting $k' \gg k$ coplanar clusters where k' is the number of extracted clusters and k is the number of actual clusters. An example of this case is demonstrated in Figure 3.18 with a coplanar cluster with 1000 nodes. Assume that all the nodes are deployed on a single plane *i.e.* there is only one coplanar cluster, as seen in Figure 3.18a. In Figure 3.18b, we see the 2D view of the coplanar cluster shown in Figure 3.18a. If we know that there is only one coplanar cluster, then we are able to run 2D localization algorithm on that cluster and determine the global positions of all nodes the network. Otherwise, we have to extract the coplanar clusters. Considering the noisy distance measurements, if we set a hop distance very small, then we end up with extracting excessive number of clusters out of one. In Figure 3.18c, we see the extracted clusters out of the original one when the hop distance is 10 units.



(a) $\theta = 10$

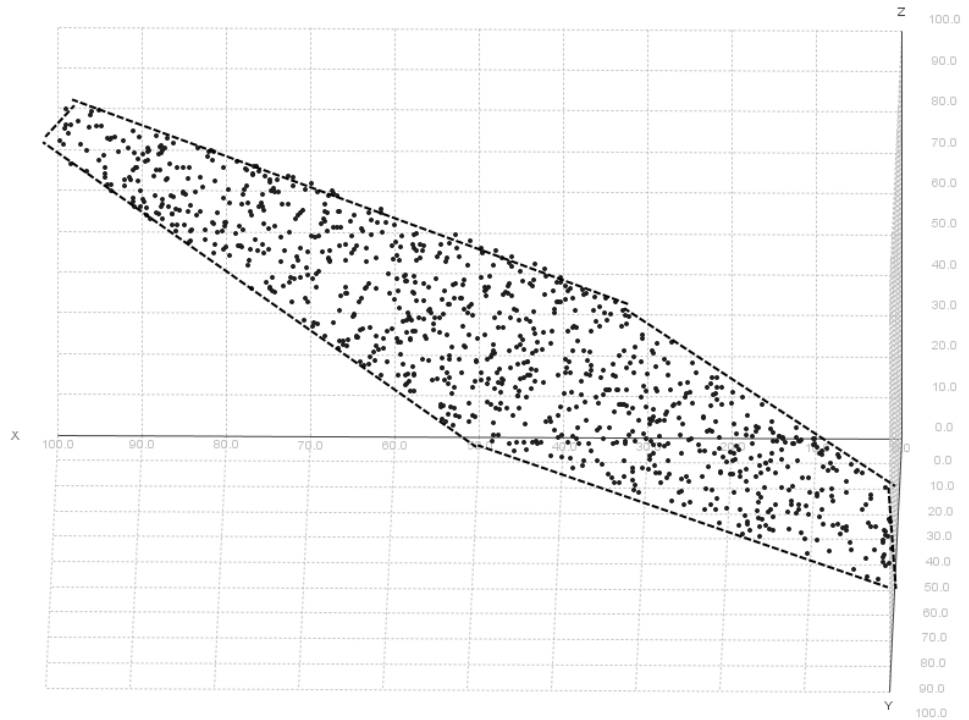


(b) $\theta = 20$

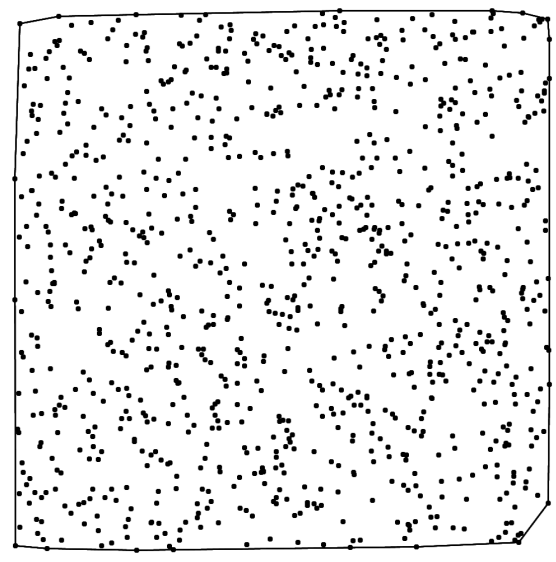


(c) $\theta = 30$

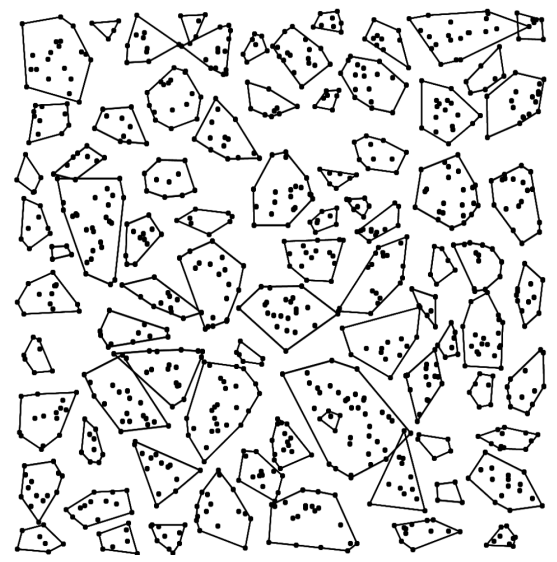
Figure 3.17: Filtering edges with respect to changing hop distance



(a) Sensor nodes deployed on one plane



(b) 2D view of the coplanar cluster



(c) Extracted clusters out of the original one

Figure 3.18: Planar clustering of a coplanar node set

To sum up, there are two types of improper extensions that we have to avoid.

- i) *Over-extension*: The extension of a coplanar cluster to include more than one cluster, as seen in Figure 3.15.
- ii) *Short-extension*: The extension of a coplanar cluster that covers only a tiny part of the actual coplanar cluster, as seen in 3.18.

We use the volume threshold κ and the hop distance θ in order to avoid improper extensions. κ is set once at the beginning and is not changed during the extraction process. Finding the best value for θ leads to the extraction of the actual coplanar clusters, which is very difficult if we do not have information on the deployment. Therefore, we set the hop distance as half the average edge weight in the WSN graph $G = (V, E)$.

$$\theta = \frac{\sum_{(v,w) \in E} d(v,w)}{2 * |E|}$$

where E is the edge set and $|E|$ is the number of edges in the graph.

After one round of extensions, if there are still off-plane nodes, then we increase θ by one to include more nodes into the coplanar clusters. When θ is increased, that means some unavailable edges become available for use while extending clusters. The algorithm that we propose to find the 3D point formation of a WSN graph G is presented in Section 3.4.2.

3.4.2 A Heuristic to Extract Coplanar Clusters

In this section, we give our heuristic algorithm to extract the coplanar node clusters in a given WSN graph. We assume that the number of coplanar clusters is unknown as well as the number of sensors in each cluster. We present our heuristic as a two part algorithm in 3.19. In Figure 3.19a, we give the overall algorithm to extract the coplanar clusters. The algorithm tries to extract coplanar clusters by using the available pairwise distances in WSN graph $G = (V, E)$ and adds the extracted clusters into coplanar cluster set $CSet$. Since the algorithm only modifies the global variables, it does not take any input. The threshold value κ and the hop distance θ are set inside the algorithm. Figure 3.19b shows the extension process of a coplanar cluster.

Remember that a coplanar cluster is denoted by $C_i = (V_i, E_i)$. We present the heuristic by assuming that as a node is placed into a coplanar cluster C_i , it is added into the vertex set V_i , and the coplanar edges of this node are added into the edge set E_i .

The algorithm given in Figure 3.19a uses the WSN graph $G = (V, E)$ and the

```

input: WSN graph  $G = (V, E)$ 
output: Coplanar cluster set  $CSet$ 
1: function EXTRACTCLUSTERS( $G$ )
2:    $\kappa \leftarrow 6 * \ln(\mathcal{E} + 1)$       /*  $\mathcal{E}$  is the error magnitude */
         $\sum_{(v,w) \in E} d(v,w)$ 
3:    $\theta \leftarrow \frac{\sum_{(v,w) \in E} d(v,w)}{2 * |E|}$       /* Average edge weight divided by two */
        /*  $E$  is the edge set, and  $|E|$  is the number of edges in  $G = (V, E)$  */
4:    $OffPlane \leftarrow V$ 
5:   sort the nodes in  $OffPlane$  by connectivity in ascending order
6:    $i \leftarrow 0$ 
7:   for Each fully-connected  $\{a, b, c, d\} \subset OffPlane$  do
8:     if  $\mathcal{V}_{abcd} \leq \kappa$  then
9:        $C_i \leftarrow \{a, b, c, d\}$ 
10:      EXTENDCLUSTER( $C_i, OffPlane, \theta, \kappa$ )
11:      add  $C_i$  into  $CSet$ 
12:       $i \leftarrow i + 1$ 
13:    end if
14:  end for
15:  while ( $OffPlane \neq \emptyset$ ) AND ( $\theta \leq$  sensing range) do
16:     $\theta \leftarrow \theta + 1$ 
17:    for Each  $C_i \in CSet$  do EXTENDCLUSTER( $C_i, OffPlane, \theta, \kappa$ )
18:  end while
19: end function

```

(a) Overall planar clustering algorithm

```

1: function EXTENDCLUSTER( $C_i, OffPlane, \theta, \kappa$ )
2:   repeat
3:      $Newbies \leftarrow \emptyset$ 
4:     for Each  $\{a, b, c\} \subset C_i$  do
5:       for Each off-plane node  $d \in OffPlane$  do
6:         if ( $d(a, d) \leq \theta$ ) AND ( $d(b, d) \leq \theta$ ) AND ( $d(c, d) \leq \theta$ ) then
          /*  $d(a, d)$  denotes the Euclidean distance between  $a$  and  $d$  */
7:         if  $\mathcal{V}_{abcd} \leq \kappa$  then add  $d$  into  $Newbies$ 
8:        $C_i \leftarrow C_i \cup Newbies$ 
9:        $OffPlane \leftarrow OffPlane \setminus Newbies$ 
10:    until  $Newbies = \emptyset$ 
11: end function

```

(b) Extending a coplanar cluster

Figure 3.19: Heuristic to extract coplanar clusters

coplanar cluster set $CSet$ which are defined as global variables. Notice that $CSet$ is empty at the beginning. In line 2, we set the value of the volume threshold. In line 3, the hop distance is set. In line 4, we mark all the nodes in the vertex set of WSN graph as off plane by adding them into the set called $OffPlane$. Then, we sort the nodes in $OffPlane$ with respect to the number of neighbors they have from lowest to highest in line 5. In line 6, we set an integer value i as zero. This value will be used as the indices of the found clusters later on. In order to find a cluster, we iterate on each node quadruplet (a, b, c, d) in the off-plane node set in line 7. We check if these four nodes form a tetrahedron with a volume smaller than the volume threshold in line 8. If so, then we create a cluster in line 9. Then, we call the function to extend a coplanar cluster $EXTENDCLUSTER(C_i, OffPlane, \theta, \kappa)$ where C_i is the cluster created in line 9, $OffPlane$ is the set of off-plane nodes, θ is the hop distance set and κ is the volume threshold. After extension of the cluster is done, we add the cluster into the coplanar cluster set in line 11 and then increase i by one in line 12. After we find a cluster and finish extending it, we search for a new cluster. When all the fully-connected off-plane node quadruplets are iterated and no clusters are found, we move onto the next phase. Because of the limited edges, it is possible that some of the nodes are left out while finding and extending clusters. Therefore, from line 15 through line 18 we keep on extending the found clusters. This second extension phase continues until either the increasing hop distance is equal to the sensing range or there are no more off-plane nodes left. In line 16, we increase the hop distance by one unit. Of course, the increment might be different if the nodes are distributed into a cube with a different volume. After updating the hop distance, we iterate on and extend each coplanar cluster in the cluster set in line 17.

The extension process of a coplanar cluster is presented in Figure 3.19b. The algorithm takes four parameters. Namely, a coplanar cluster to be extended C_i , the set of off-plane nodes $OffPlane$, the hop distance θ and the volume threshold κ . We search for off-plane nodes that is coplanar with any three of the nodes in the coplanar cluster from line 2 through line 10. In line 3, we create a set called $Newbies$ to store the new nodes that will be added into the coplanar cluster. In line 4, we iterate on each node triplet (a, b, c) in the coplanar cluster C_i . In line 5, we iterate on each off-plane node d and we check if d is closer to a, b and c than the hop distance in line 6. We assume that if a pairwise distance is not available *i.e.* two sensor nodes cannot sense each other, the distance value is infinite. If all three pairwise distances between d and a, b, c are less than θ , then we check if the volume \mathcal{V}_{abcd} is less than the allowed volume threshold κ . If all the conditions are satisfied, then we add d into $Newbies$ in line 7. When all the triplets in the cluster are iterated, we add the found coplanar nodes into coplanar cluster C_i in line 8. We also remove them from off-plane node set in line 9. The algorithm runs until there are no new nodes to add into the coplanar cluster that is being extended.

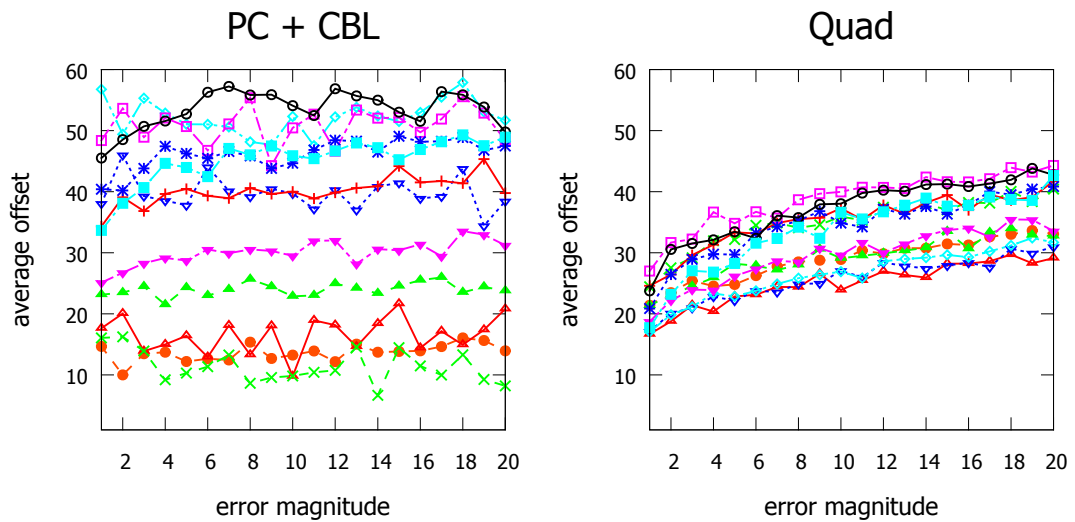
REMARK 5 *In the algorithm presented in Figure 3.19b, it might seem redundant to iterate on each triplet in line 4 at each iteration of extraction. However, the new nodes added into the coplanar cluster in line 8 might have common off-plane neighbors with a node that is already in the cluster. Since we do not want to skip*

that off-plane neighbor, we iterate all the triplets all over again.

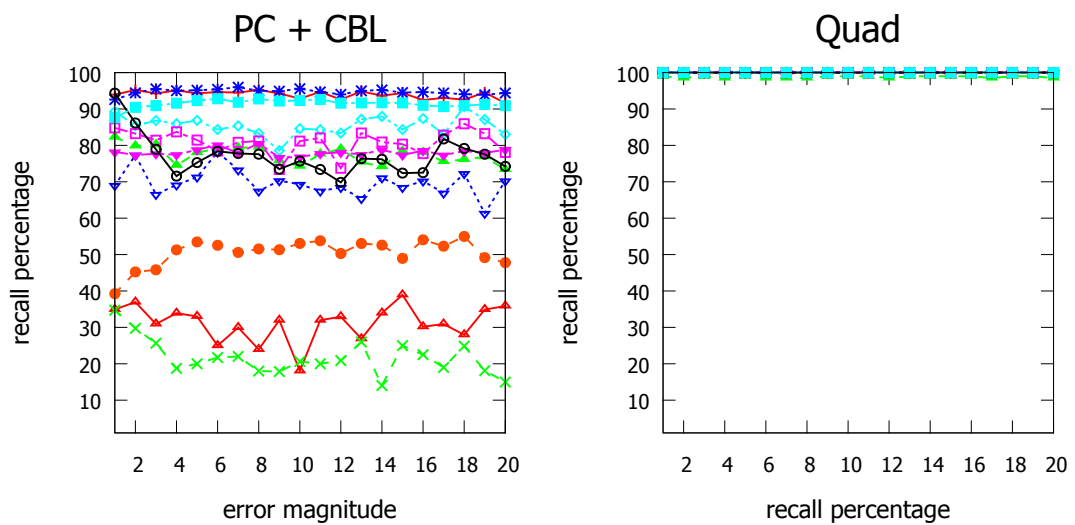
3.4.3 Experimental Evaluation of Planar Clustering

In this section, we present the experimental results on the performance of CBL, run after the heuristic that we propose in Figure 3.19b with the same setup used in Section 3.3.3. We have shown that CBL localizes the given network more precisely in Section 3.3.3. In this section, we assume that the clustering information is not present to test the proposed heuristic to extract coplanar clusters presented in Figure 3.19.

First, planar clustering algorithm is run to extract coplanar clusters and then CBL is used to localize the network. We present the results of experiments that we use planar clustering before CBL with title **PC + CBL**. The charts where only mere quadrilateration is run are entitled as *Quadrilateration*. In Figure 3.20, we see the average offsets and recalls of CBL and quadrilateration when the clustering information is not available. Figure 3.20a presents the average offsets and Figure 3.20b presents the recall percentages of two algorithms. The results show us that when CBL is used with the proposed heuristic, we prefer quadrilateration to localize the network. There is much more room for improvement of this algorithm. We leave finding the hop distance and volume threshold for an accurate extraction of the coplanar clusters as an open problem.



(a) Average offsets



(b) Recall percentages

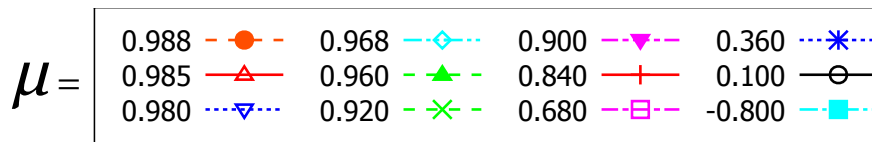


Figure 3.20: Average offsets (a) and recall percentages(b) of CBL and quadrilat-
eration when the information on coplanar clusters is not available

Chapter 4

Conclusion

In this chapter, we give our concluding remarks and future research directions for range-based localization of a 3D WSN that we know to follow a deployment on the planar surfaces. In Section 4.1, we give the related work. In Section 4.2, we discuss our work and give the future research topics that we would like to study on.

4.1 Related Work

Range-based localization is basically a graph embedding problem [48], which has been shown to be strongly NP-Hard for all dimensions [36, 49].

In 2004, Aspnes *et al.* [50] showed that the problem is also NP-Hard for unit-disk graphs. Eren *et al.* [28] proposed trilateration, which is a polynomial time algorithm that can be used to localize a WSN. However, trilateration needs exact range measurements to function precisely, which is not the case in real-world due to device errors or environmental noise. The environmental noise causes the measured distances to be slightly different than the original distances. Evrendilek and Akcan [37] showed that localization through trilateration is NP-Hard when the distance measurements are imprecise.

The imprecise distance measurements cause a node to be localized far from its original position. This positional offset accumulates at the latter stages of the localization. In order to reduce this type of ambiguities, Akcan and Evrendilek [43] used triangles whose angles are less than a certain value, referred to as robust triangles.

Aspnes *et al.* [17] investigated the localization and localizability of a network in 2010. They defined the term *global rigidity* and showed that it is sufficient and necessary condition for a WSN graph to be localized in 2D. Even though global rigidity is defined for all dimensions, the sufficient and necessary conditions for a

WSN to be localized in 3D have not been found yet.

If the sensor nodes estimate their own positions, *i.e.* self localization is performed, the intractability of the problem signals an excessive amount of computation, potentially depleting the batteries of the self-localizing sensors. Therefore, not only localizing the sensors, but also completing the process with less number of computations is essential. Energy efficiency can be achieved either by reducing the energy spent per node [51, 52, 53], or by reducing the total number of computations during the localization process [17, 28, 54]. A review of different approaches of node localization discovery in wireless sensor networks can be found in [9].

We study the WSN localization problem using clustering, which is an efficient way to approach the problem [55]. In 2000, Amis *et. al* [56] present a heuristic to form multi-hop clusters in a dynamic WSN. In 2002, Chatterjee *et al.* [57] proposed a weighted clustering algorithm, that clusters the nodes in a WSN based on the neighbor distances. In 2004, Demirbas *et. al.*[58] proposed a clustering method named solid-disc clustering that works in constant time and similar to [57], it is based on the Euclidean distances. In 2008, Zainalie and Yaghmaee [59] designed an algorithm that clusters the sensor nodes with maximum number of nodes in each cluster. In 2008, Lederer *et. al.* [60], developed an algorithm that first partitions the sensor field into Voronoi cells with respect to the apriori given landmark nodes and then extract the combinatorial Delaunay complex as the dual complex of the landmark Voronoi diagram and embed the combinatorial Delaunay complex as a structural skeleton. In 2009, Yue *et. al.* [61] followed the work done in [60] and improved the localization quality by a landmark node selection algorithm.

In 2012, Yao *et. al.* [62] studied the localization of a WSN in 3D, in which sensors are deployed on surfaces. Different than our work, they use a layered approach, that is based upon the nodal height measurements. Cucuringu *et. al.* [63] developed a non-incremental non-iterative anchor-free algorithm for localizing sensor networks in 2D, which is experimentally shown to be robust to noisy distance measurements.

We use an extra constraint while clustering, based on the observation that the sensor nodes usually deploy on surfaces that can be modeled using planes. In line with the method proposed by Akcan and Evrendilek [4, 43] in 2013 which uses dual wireless radios, our work localizes the formed structures instead of localizing each node one by one.

4.2 Discussion

In this section, we conclude this thesis by discussing our contributions to the topic and pointing at the future research directions.

4.2.1 Summary of Contributions

This thesis addresses range-based wireless sensor network (WSN) localization problem in 3D where the sensor nodes sit on planar surfaces to form rigid 2D structures. Figure 4.1 shows the scope of this thesis in the state-of-art. We work in the intersection of 2D and 3D localization and make a contribution to 3D localization.

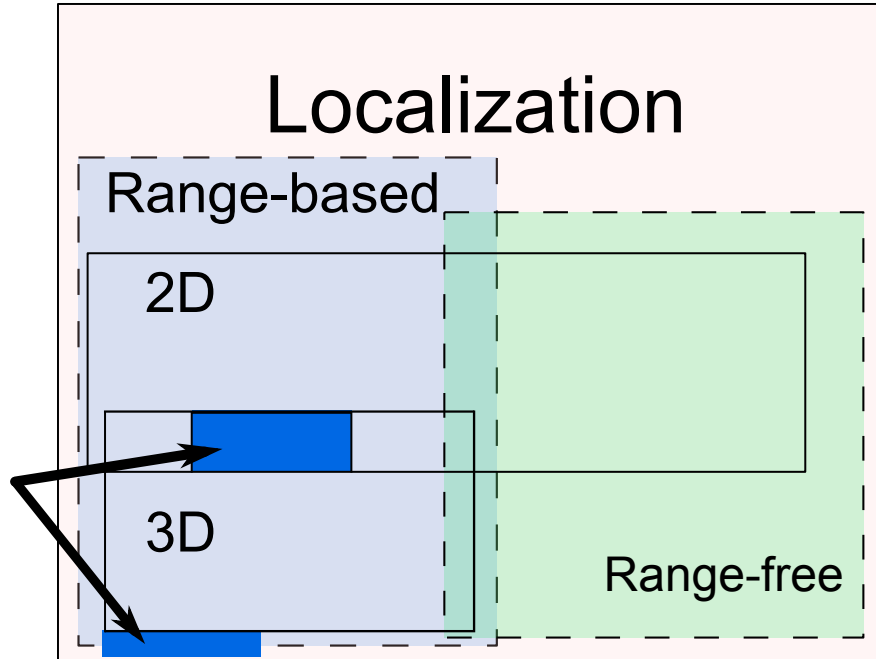


Figure 4.1: This thesis in the state-of-art

The summary of our contributions can be listed as follows.

- Our main contribution is the proposed algorithm for range-based WSN localization in Section 3.3, CBL, that aims to exploit the structural information where the sensor nodes are deployed on planar surfaces in a 3D environment. CBL can be used as an extension of any localization algorithm.
- We have shown that if the information on the coplanar clusters is not present, extracting such information is very much needed.
- Based on the observation above, we have defined the problem of extracting the coplanar clusters and presented a first-attempt heuristic to solve the problem in Section 3.4. Even though our heuristic fails to extract the information accurately, we have pointed out two parameters, the volume threshold κ and the hop distance θ , that need to be set carefully.
- We have defined a metric called planarity factor to indicate how planar is the deployment of the sensor nodes.

4.2.2 Future Research Directions

For our future research, we would like to investigate the following.

- The most obvious research direction is to discover the relationship among the planarity factor μ , the volume threshold κ and the hop distance θ .
- Another research direction is to investigate the theoretical bounds of the coplanar cluster extraction. We would like to answer the following questions about the problem:
 - Is the problem NP-Hard?
 - Is the problem approximable?
 - Does the information about the planarity factor change the difficulty of the problem?
 - Given the pairwise distances, can we find values for the volume threshold κ and the hop distance θ based on the average connectivity and the average edge weight of the WSN graph?
- We would like to use CBL with different range-based algorithms than trilateration and quadrilateration to find out if CBL improves the quality of localization with other algorithms.

Chapter 5

Bibliography

- [1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *Communications Magazine, IEEE*, vol. 40, pp. 102–114, Aug 2002.
- [2] Z. Zhong, *Range-Free Localization and Tracking in Wireless Sensor Networks*. PhD thesis, The Graduate School of the University of Minnesota, Sep 2010.
- [3] N. Bulusu, J. Heidemann, and D. Estrin, “GPS-less low-cost outdoor localization for very small devices,” *Personal Communications, IEEE*, vol. 7, pp. 28–34, Oct 2000.
- [4] H. Akcan and C. Evrendilek, “GPS-free directional localization via dual wireless radios,” *Computer Communications*, vol. 35, no. 9, pp. 1151–1163, 2012.
- [5] H. Akcan, V. Kriakov, H. Bronnimann, and A. Delis, “GPS-Free node localization in mobile wireless sensor networks,” in *Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE’06)*, (Chicago, Illinois, USA), p. 35–42, 2006.
- [6] S. Capkun, M. Hamdi, and J. Hubaux, “GPS-free positioning in mobile ad-hoc networks,” in *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, p. 10, 2001.
- [7] C. Chang, “Localization and Object-Tracking in an Ultrawideband Sensor Network,” Master’s thesis, UC Berkeley, USA, 2004.
- [8] C. Savarese, J. Rabaey, and K. Langendoen, “Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks,” in *Proceedings of the General Track of the Annual Conference on USENIX Annual Technical Conference*, pp. 317–327, 2002.
- [9] A. Pal, “Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges,” *Network Protocols and Algorithms*, vol. 2, no. 1, pp. 45–73, 2010.

- [10] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*. Springer-Verlag, 4 ed., 1997.
- [11] Benefon, “Benefon: 1999 Annual Report.” <http://web.lib.hse.fi/FI/yrityspalvelin/pdf/1999/Ebenefon1999.pdf>, 1999.
- [12] W. Hirt, “Ultra-wideband radio technology: overview and future research,” *Computer Communications*, vol. 26, no. 1, pp. 46–52, 2003.
- [13] J. McCorkle, “A tutorial on ultrawideband technology.” IEEE P802.15 Working Group for Wireless Personal Area Networks (WPANS), 2000.
- [14] S. Gezici, Z. Tian, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor, and Z. Sahinoglu, “Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks,” *Signal Processing Magazine, IEEE*, vol. 22, pp. 70–84, July 2005.
- [15] G. Mao, B. Fidan, and B. Anderson, “Wireless sensor network localization techniques,” *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.
- [16] H. Khoury and V. Kamat, “Evaluation of position tracking technologies for user localization in indoor construction environments,” *Automation in Construction*, vol. 18, no. 4, pp. 444–457, 2009.
- [17] J. Aspnes, T. Eren, D. Goldenberg, A. Morse, W. Whiteley, Y. Yang, B. Anderson, and P. Belhumeur, “A theory of network localization,” in *IEEE Transactions on Mobile Computing*, vol. 5, pp. 1663–1678, 2010.
- [18] P. Bahl and V. Padmanabhan, “RADAR: An In-Building RF-Based User Location and Tracking System,” in *INFOCOM '00*, pp. 775–784, 2000.
- [19] J. Erickson, “Radio Location and the Air Defence Problem: The Design and Development of Soviet RADAR,” *Science Studies*, vol. 2, pp. 241–263, July 1972.
- [20] H. Koyuncu and S. Yang, “Survey of Wireless Indoor Positioning Techniques and Systems,” *International Journal of Computer Science and Network Security*, vol. 10, pp. 121–128, May 2010.
- [21] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of Wireless Indoor Positioning Techniques and Systems,” *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, pp. 1067–1080, Nov 2007.
- [22] W. Dargie and C. Poellabauer, *Fundamentals of wireless sensor networks: theory and practice*. John Wiley and Sons, 2010.
- [23] C. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall PTR, 2002.
- [24] T. He, S. Krishnamurthy, L. Luo, Y. Ting, L. Gu, R. Stoleri, G. Zhou, Q. Cao, P. Vicaire, J. Stankovic, T. Abdelzaher, J. Hui, and B. Krogh,

- “VigilNet: An Integrated Sensor Network System for Energy-efficient Surveillance,” *ACM Trans. Sen. Netw.*, vol. 2, pp. 1–38, Feb 2006.
- [25] A. Savvides, C. Han, and M. Strivastava, “Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors,” in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, MobiCom '01*, pp. 166–179, ACM, 2001.
- [26] J. Stoep, *Design and Implementation of Reliable Localization Algorithms Using Received Signal Strength*. University of Washington, 2009.
- [27] Y. Yechiam, “Some theoretical aspects of position-location problems,” in *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pp. 1–8, 1979.
- [28] T. Eren, O. Goldenberg, W. Whiteley, Y. R. Yang, A. Morse, B. D. O. Anderson, and P. Belhumeur, “Rigidity, computation, and randomization in network localization,” in *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, pp. 2673–2684, 2004.
- [29] H. L. Groginsky, “Position Estimation Using Only Multiple Simultaneous Range Measurements,” *Aeronautical and Navigational Electronics, IRE Transactions on*, vol. 6, pp. 178–187, Sept 1959.
- [30] T. Eren, P. Belhumeur, and A. Morse, “Closing Ranks in Vehicle Formations Based on Rigidity,” in *In Proceedings of the 41st IEEE Conference on Decision and Control*, pp. 2959–2964, 2002.
- [31] T. Eren, P. Belhumeur, B. Anderson, and A. Morse, “Rigidity, computation, and randomization in network localization,” in *15th IFAC World Congress*, pp. 2673–2684, Jul 2002.
- [32] B. Hendrickson, “Conditions For Unique Graph Realizations,” *SIAM J. Comput.*, vol. 21, pp. 65–84, 1992.
- [33] B. Jackson and T. Jordan, “Connected rigidity matroids and unique realizations of graphs,” *Combinatorial Theory*, vol. 94, pp. 1–29, 2005.
- [34] R. Connelly, “On generic global rigidity,” *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, vol. 4, pp. 147–155, 1991.
- [35] G. Laman, “On graphs and rigidity of plane skeletal structures,” *Journal of Engineering Mathematics*, vol. 4, no. 10, pp. 331 – 340, 2002.
- [36] J. Saxe, “Embeddability of weighted graphs in k-space is strongly NP-Hard.,” in *17th Allerton Conference in Communications, Control and Computing on*, pp. 480–489, 1979.
- [37] C. Evrendilek and H. Akcan, “On the complexity of trilateration with noisy range measurements,” *IEEE Communications Letters*, vol. 15, no. 10, pp. 1097 – 1099, 2011.

- [38] D. Sommerville. Bronx, New York: Dover Publications, 1958.
- [39] A. Kannan, B. Fidan, and M. Guoqiang, “Analysis of Flip Ambiguities for Robust Sensor Network Localization,” *Vehicular Technology, IEEE Transactions on*, vol. 59, pp. 2057–2070, May 2010.
- [40] B. Alavi and K. Pahlavan, “Modeling of the TOA-based distance measurement error using UWB indoor radio measurements,” *Communications Letters, IEEE*, vol. 10, pp. 275–277, Apr 2006.
- [41] J. Park, E. Demaine, and S. Teller, “Moving-Baseline Localization,” in *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pp. 15–26, Apr 2008.
- [42] J. Sally and P. Sally, *Roots to research: a vertical development of mathematical problems*. American Mathematical Society Bookstore, 2007.
- [43] H. Akcan and C. Evrendilek, “Reducing The Number Of Flips In Trilateration With Noisy Range Measurements,” in *Proceedings of the 12th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'13)*, (New York, NY, USA), p. 20–27, June 2013.
- [44] “The history of Java technology.” <http://www.oracle.com/technetwork/java/index-jsp-142903.html>. Accessed: 2014-12-31.
- [45] “About the Eclipse foundation.” <http://www.eclipse.org/org/>. Accessed: 2014-12-31.
- [46] M. Grantson and C. Levcopoulos, “Covering a Set of Points with a Minimum Number of Lines,” *Algorithms and Complexity*, vol. 3998, pp. 6–17, 2006.
- [47] U. Feige, “A Threshold of $\ln N$ for Approximating Set Cover,” *J. ACM*, vol. 45, pp. 634–652, Jul 1998.
- [48] C. Thomassen, “The graph genus problem is NP-complete,” *Journal of Algorithms*, vol. 10, no. 4, pp. 568–576, 1989.
- [49] S. Pemmaraju and I. Pirwani, “Good Quality Virtual Realization of Unit Ball Graphs,” in *Proceedings of the 15th Annual European Conference on Algorithms, ESA'07*, pp. 311–322, 2007.
- [50] J. Aspnes, D. Goldenberg, and R. Yang, “On the Computational Complexity of Sensor Network Localization,” in *In Proceedings of First International Workshop on Algorithmic Aspects of Wireless Sensor Networks*, pp. 32–44, 2004.
- [51] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, p. 10, 2000.
- [52] B. Zhang and F. Yu, “An energy efficient localization algorithm for wireless

- sensor networks using a mobile anchor node,” in *Information and Automation, 2008. ICIA 2008. International Conference on*, pp. 215–219, Jun 2008.
- [53] T. Srinath, “Localization in resource constrained sensor networks using a mobile beacon with in-ranging,” in *Wireless and Optical Communications Networks, 2006 IFIP International Conference on*, p. 5, 2006.
- [54] J. Ding, L. Zhang, G. Cheng, Z. Ling, Z. Zhang, and Y. Lei, “Study on DV-Hop Algorithm Based on Modifying Hop Count for Wireless Sensor Networks,” *ARPJ Journal of Systems and Software*, pp. 1452–1456, Oct 2012.
- [55] A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks,” *Computer Communications*, vol. 30, no. 14–15, pp. 2826–2841, 2007.
- [56] A. Amis, R. Prakash, T. Vuong, and D. Huynh, “Max-min d-cluster formation in wireless ad hoc networks,” in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 32–41, 2000.
- [57] M. Chatterjee, S. K. Das, and D. Turgut, “WCA: A Weighted Clustering Algorithm for Mobile Ad hoc Networks,” *Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks)*, vol. 5, pp. 193–204, 2002.
- [58] M. Demirbas, A. Arora, and V. Mittal, “FLOC: A fast local clustering service for wireless sensor networks,” in *Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS/DSN)*, 2004.
- [59] S. Zainalie and M. Yaghmaee, “CFL: A clustering algorithm for localization in Wireless Sensor Networks,” in *Telecommunications, 2008. IST 2008. International Symposium on*, pp. 435–439, Aug 2008.
- [60] S. Lederer, Y. Wang, and J. Gao, “Connectivity-Based Localization of Large Scale Sensor Networks with Complex Shape,” in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, 2008.
- [61] W. Yue, S. Lederer, and G. Jie, “Connectivity-Based Sensor Network Localization with Incremental Delaunay Refinement Method,” in *INFOCOM 2009, IEEE*, pp. 2401–2409, Apr 2009.
- [62] Z. Yao, W. Hongyi, J. Miao, and X. Su, “Localization in 3D surface sensor networks: Challenges and solutions,” in *INFOCOM, 2012 Proceedings IEEE*, pp. 55–63, Mar 2012.
- [63] M. Cucuringu, Y. Lipman, and A. Singer, “Sensor Network Localization by Eigenvector Synchronization over the Euclidean Group,” *ACM Trans. Sen. Netw.*, vol. 8, pp. 1–42, Aug 2012.