EFFECT OF 3D TOPOGRAPHICAL SURFACES FOR THE PERFORMANCE EVALUATION OF WIRELESS SENSOR NETWORKS

by

Müzeyyen Gökçen Arslan

B.S., Computer Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2006

EFFECT OF 3D TOPOGRAPHICAL SURFACES FOR THE PERFORMANCE EVALUATION OF WIRELESS SENSOR NETWORKS

APPROVED BY:

Prof. Cem Ersoy .....
(Thesis Supervisor)

Prof. Emin Anarım .....

Dr. Ali Vahit Şahiner .....

DATE OF APPROVAL: 14.June.2006

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Professor Cem Ersoy for his guidance and support during my study. His encouragement helped me to realize this thesis.

My thanks must go to my project leader Murat Apohan and work friends at the National Institute of Electronics and Cryptology (TÜBİTAK UEKAE). Without their support, I could not achieve my graduate study.

I must also thank my friends for their technical support and continuous belief in me.

My very special thanks is for my parents, for their love, trust, help and care throughout my life. I also would like to thank my sister Ece İrem, for cheering me up all the time.

ABSTRACT

EFFECT OF 3D TOPOGRAPHICAL SURFACES FOR THE PERFORMANCE EVALUATION OF WIRELESS SENSOR NETWORKS

A *wireless sensor network* (WSN) is a self-organizing network, consisting of tiny wireless nodes which carry out a sensing and transferring task in collaboration. WSNs are getting popular because of their ease of deployment, self-organizing capability, low cost and their wide range of applications. This wide spectrum of applications raises most of the time application specific research problems in WSN protocol stack and algorithm design. However, the performance of the most proposed models in literature, have been evaluated on planar surfaces, assuming a distance based sensing and 2D freespace communication model, while assuming a random deployment scheme which commonly takes place in 3D inaccessible terrains.

In this thesis, we investigated the problem of incorporating a realistic modeling environment into sensor networks. Our motivation has been the non-realistic and contradictive assumptions in WSN performance evaluations, where the formations of the topographic surface that would normally block the communication and sensing task are not taken into account. We incorporated a 3D terrain model into the performance evaluation of a collaborative target tracking application. The evaluation is done on various artificially generated but realistic terrains, on the performance metrics of *mean error* which is a measure of tracking accuracy, *number of communicating sensor pairs* and *number of detecting sensors*. Our simulations show that the performance predictions could be misleading on the paper design, due to non-realistic assumptions with regards to the WSN deployment region.

ÖZET

3 BOYUTLU TOPOGRAFİK YÜZEYLERİN TELSİZ ALGILAYICI AĞLARIN PERFORMANS DEĞERLENDİRMESİNE ETKİSİ

Telsiz algılayıcı ağlar, veri aktarımı ve algılama gibi işlemleri işbirliği içinde gerçekleştiren küçük algılayıcı düğümlerinden oluşur. Algılayıcı ağlar, kolay konumlandırılmaları , kendi kendine organize olabilmeleri, düşük maliyetleri ve geniş uygulama alanlarından dolayı yaygınlaşmaktadır. Bahsedilen geniş uygulama alanları, algılayıcı ağların tasarımı sırasında uygulamalara özel bir takım sorunları da beraberinde getirmektedir. Ancak literatürde var olan çoğu çözüm, bir yandan düzlemsel alanlarda gerçekleşen mesafe tabanlı algılama ve iki boyutlu bir haberleşme modelini temel alırken, bir yandan da normalde erişilmez arazilerde gerçekleşen rastgele konumlandırma senaryosunu temel almaktadırlar.

Bu çalışma algılayıcı ağlarına gerçekçi bir modelleme ortamı dahil edilmesini içerir. Motivasyonumuz, algılayıcı ağların performans değerlendirmeleri sırasında, topografik düzlemlerin etkilerinin hesaba katılmaması gibi gerçek dışı varsayımlardır. İşbirliği içinde gerçekleşen bir hedef izleme uygulamasına üç boyutlu bir arazi modeli dahil edilmiştir. Algılayıcı ağların yapay olarak yaratılmış arazilerdeki performans değerlendirmesinde üç metrik hesaba katılmıştır; hedef takibindeki ortalama hata, haberleşen algılayıcı çifti ve hedefi sezen algılayıcı sayısı. Benzetimlerimiz, algılayıcı ağlarının konumlandırılma alanlarına yönelik gerçek dışı varsayımlardan dolayı, kağıt üstündeki tasarımların ve performans öngörülerinin yanıltıcı olduğunu göstermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$Cr_{t_y}^{t_x}$	Ratio of communication rate of terrain x to terrain y
$Dr_{t_y}^{t_x}$	Ratio of detection rate of terrain x to terrain y
r_m	target's measured range
R_m	average variance or covariance
R_t	true variance or covariance
R_m^{11}	covariance of observation error in x coordinate
R_m^{22}	covariance of observation error in y coordinate
R_m^{12}	variance of observation errors in x and y coordinates
s_N^2	sample variance
s_{N-1}^2	bias corrected sample variance
β	target's true bearing
ε_m	target's observed elevation angle
ε	target's true elevation angle
η_m^c	observed y coordinate of the target
φ^c	target mean state observed in cartesian coordinates
λ_β	bias compensation factor
μ	mean of distribution
ν_r	error in range
ν_β	error in bearing
ν_ε	error in elevation angle
r	target's true range
σ_r	standard deviation of range
σ_β	standard deviation of bearing
σ_ε	standard deviation of elevation angle
ξ_m^c	observed x coordinate of the target
2D	2 Dimensions
3D	3 Dimensions

ALTW	Altitude in 16-bit Words
cnode	collector node
C4ISR	Command Control Communications Computer Intelligence Surveillance Reconnaissance
CRAD	Curvature Radius
EKF	Extended Kalman Filter
EOF	End of File
ICTP	Information Controlled Transmission Power adjustment
LOS	Line of Sight
MEMS	Micro-Electromechanical Systems
MDS	Multidimensional Scaling
SCAL	Scale
SWSN	Surveillance Wireless Sensor Network
WSN	Wireless Sensor Network
XPTS	Points in X direction
YPTS	Points in Y direction

1. INTRODUCTION

1.1. Motivation and Purpose

Advances in wireless communications, digital electronics and embedded systems have led to the development of a new generation of ad hoc networks, namely wireless sensor networks, (WSN). These devices are low cost, low power and capable of communicating wirelessly in short distances. A WSN consists of a number of tiny nodes deployed inside the phenomenon, which carry out a sensing and transferring task in collaboration [1]. Recently WSNs have received much attention due to their wide range of applications including security and surveillance, control of complex systems, and monitoring of environment. These wide spectrum of applications raise most of the time application specific research problems in WSN protocol stack and algorithm design. Some of these problems include *power efficient self organization techniques*, *sleep scheduling*, *minimum energy routing*, *sensing coverage*, *deployment* and *target tracking*. These research problems need application-specific approaches for deriving an accurate analysis on the WSN system. However, the performance of the most proposed models in literature, have been evaluated on planar surfaces assuming a 2D freespace communication and a distance–based sensing model, with no obstacles or blocking among nodes and, nodes and the target. In real life, random deployment of a WSN, which most models assume, takes place in 3D terrains, where blockage occurs and affects line of sight (LOS) conditions, resulting in a degradation on the communication and sensing task. Therefore, a more realistic model which considers the 3D terrain effects on both sensing and communication tasks, needs to be built into sensor network performance evaluation.

This thesis covers the problem of incorporating a 3D terrain model into the performance evaluation of a collaborative target tracking application. The evaluation is done on various artificially generated but realistic terrains.

Collaborative target tracking is important in terms of considering the 3D terrain

effects on both communication and sensing tasks of the nodes. The goal of this thesis is to show how the performance metrics of a WSN system, e.g., accuracy in a target tracking application, in a realistic terrain simulation could deviate from those on a planar surface simulation. Our simulations show that the performance predictions could be misleading on the paper design, due to non-realistic assumptions with regards to the WSN deployment region.

The rest of the thesis is organized as follows: Chapter 2 studies WSN foundations in terms of WSN architecture and applications as well as covering the current research problems. In Chapter 3, basics for 3D topography generation have been given. Here, terrain data format, terrain processing, terrain generation methods and generation tools have been explained. Chapter 4 gives the background on target tracking application where the 2D and 3D WSN evaluation has been performed. In Chapter 5, the performance results of simulations have been analyzed. We finally conclude and define the future work in Chapter 6.

2. WIRELESS SENSOR NETWORK FOUNDATIONS

2.1. Background

The rapid progress of wireless communication and embedded MEMS technologies have made *wireless sensor networks* possible. These wireless nodes are capable of storing, collecting, and processing ambient information and communicating with the neighbouring nodes. In the past, sensors were connected with wire lines [2]. Today wireless environment is combined with ad–hoc networking technology to facilitate communication among sensor nodes.

In WSNs, 100s to several 1,000s of sensor nodes are densely deployed throughout the sensor field, where each sensor independently senses the environment but collaboratively achieves complex information gathering and dissemination tasks like *intrusion detection*, *target tracking*, *localization*, *environmental monitoring*, *health monitoring* and *remote sensing* [3]. The nodes are often randomly deployed via scattering of nodes in the field as shown in Figure 2.1.

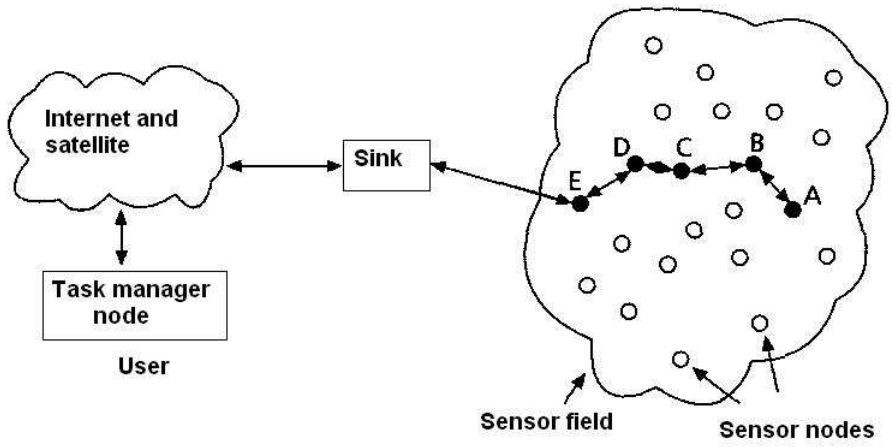

Figure 2.1. Sensor nodes scattered in a field

These sensor nodes collect data and route data back to the sink by multihop infrastructureless architecture. The sink conveys the gathered data to the task manager

via internet or satellite channel. The sink node, which is also referred to as a collector node *cnode*, is often more capable than the other nodes in terms of computation, storage and energy resources. Multiple WSNs can be integrated into a larger network through internet or interface of *cnode*s, as seen in Figure 2.2.

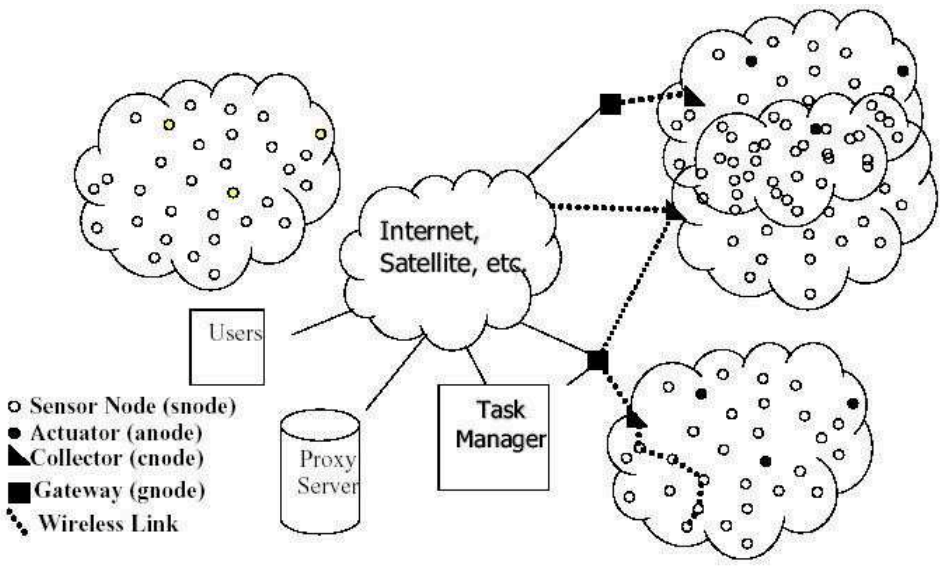

Figure 2.2. WSN topology

The realization of a sensor network architecture is affected by many design issues, which have not been addressed by previous ad hoc networking techniques. To satisfy the goals of these new generation ad hoc networks, several design factors should be carefully analyzed, such as *fault tolerance*, *mobility*, *scalability*, *production costs*, *operating environment*, *sensor network topology*, *hardware constraints*, *transmission media*, *data throughput*, *message latency*, *power consumption*, and *security*. For further information on these issues, [4, 5] could be taken as references.

Sensor networks have wide range of applications [5]. These applications include:

- *Industrial Control and Monitoring*

  WSNs could be used in industrial safety applications for detection and identification of dangerous materials, poisons, etc. Monitoring and control of moving or rotating machinery is another suitable area for WSNs. With their low energy requirements, sensor networks are effective means in monitoring the temperature, lubrication flow and vibration of the components.

- *Home Automation and Consumer Electronic*

  In smart home applications, one can use a PDA type device and remotely control the household devices, like television, DVD player, stereo, lights, and locks that are equipped with a wireless sensor network connection. Other examples are controlling the temperature of the house, combining multiple events such that, when a call is received, home entertainment system is turned off and recording a person's weight without manual intervention. House security system is another home application area of WSNs. The WSN integrated security system could be used to detect unlocked doors, a broken window and in turning off indoor lights to save energy.

- *Security and Military Sensing*

  WSN security applications can be extended to several military applications. Their size, camouflage property, distributed control and routing structure (i.e., without single point of failure), and use of spread spectrum technique in communication, make them difficult to be detected and destroyed by enemies.

  A WSN could be typically used for replacing soldiers or mines in security-sensitive areas such as borders, tactical fields, etc. WSNs could also be instrumental in identifying targets of potential attack and locating friendly forces. Seismic vibration sensors, magnetic sensors, ultrawideband radar sensors could be used for these applications.

- *Asset Tracking and Security Chain Management*

  An example of asset tracking application is tracking the records of tens of thousands of containers in a large port.

  A related application in supply chain management is the identification of location of items in large warehouses. The result of a lost item is that, it is unavailable for sale.

- *Intelligent Architecture and Environment Sensing*

  Environmental application areas of WSNs include rain gauge, soil moisture measurements, automation of farming equipment, location determination of animals within the ranch and sensing of environmental contaminants.

- *Health monitoring*

  "Health monitoring is defined as the monitoring of non-life-critical health infor-

mation to differentiate it from medical telemetry", [5]. An application of WSNs in health monitoring is monitoring one's pulse and respiration via wearable computers which is then sent to personal computers for analysis. Others include tracking daily blood sugar, chronic disorders, etc.

2.2. Performance Evaluation of WSNs

As discussed in previous sections, the performance evaluation of WSN research problems are mostly modeled on planar surfaces. This section aims to give a background on the proposed techniques for these WSN research problems. All of the works defined below are evaluated on 2D surfaces.

2.2.1. Coverage and Deployment

Sensor network layer design, e.g., physical layers [6, 7, 8], media access layer [9], network layer [10, 11, 12], has been a popular research problem in recent years. Although these issues have been largely investigated, application layer protocols remain still an unexplored area. One should consider application-specific issues during the algorithm design of a WSN protocol, e.g., a coverage degree is required for every point in the monitoring task while for target tracking only certain regions should be covered with a certain degree.

Coverage and deployment are among the primary research challenges in sensor network design. The coverage problem is defined as, *how well the sensing field is monitored by sensors* while the deployment problem is *to decide on a deployment strategy to meet the coverage requirements of the sensing field* [2].

Deployment problems are discussed in [13, 14]. In [13], the model for WSN evaluation assumes all sensors are identical and able to communicate with the base station, while the deployment algorithm tries to maximize the coverage area. In [14], the probability of detecting a target passing a predefined region determines how well the deployment is done.

Coverage problems have also been widely discussed and formulated in various ways in literature. Circle covering and art gallery problem are computational geometry approaches for evaluating the coverage problem in WSN. Circle covering is the problem of arranging a fixed number of circles that fully cover the plane while minimizing the radius of circles. Art gallery problem [15] is the problem of deciding on the number of guards such that every point in the art gallery is monitored by at least one guard.

Some works consider energy efficiency while monitoring the coverage as in [16, 17, 18]. In [19], authors study placement of sensors with a full coverage and minimal energy consumption. The heuristic authors developed, divides the sensor nodes into mutually exclusive sets such that a set provides a complete coverage. Only one set is active and consumes power at any moment. In [20], implications to network planning of performance of the sensor networks have been analyzed, depending on three coverage measures. *Area coverage* is the fraction of area covered by sensors, *node coverage* is the fraction of sensors that can be removed without reducing the covered area, and *detectability* is the capability of sensor network to detect an object moving in the network.

In [21], sensing coverage and breach path is determined by applying the Neyman-Pearson detection model. The authors analyze the breach probability of the weakest breach path, besides the deployment strategy in a *surveillance wireless sensor network* (SWSN). Here, the security level of an SWSN is associated with the *breach probability* which is defined as "the miss probability of an unauthorized target passing through the field". For simplification, the area to be monitored is modeled as a 2D connected grid. At each node of the grid, detection probability is calculated and the weakest path is derived through calculating the miss probability of the path using Dijkstra's shortest path algorithm. This probability is then used to estimate the required number of sensors for a given area and a false alarm rate.

2.2.2. WSN Terrain Modeling

In the literature, only a few researchers have considered terrain effects when attacking the problems of WSN, such as routing, localization and sensing coverage. In [22], the authors optimize the number of sensors for a given area and determine their placements for a target detection system. The aim is to minimize the cost by reducing number of sensors, yet provide sufficient sensor coverage by taking into account the environmental effects (obstacles, such as buildings, trees, etc.) and redundancy due to failures.

An inter-related issue to the target detection is the localization problem. In many senses, localization is a fundamental problem in WSN. First, when collecting data from sensors, their location must be stamped in order to output meaningful results. Second, in target detection systems, the exact position of the sensor which communicated the data associated with the sensed target, must be known. Besides, most of the communication protocols rely on the knowledge of sensor positions, where the location of the sensors is not predetermined since they are mostly dropped from an airplane in a remote terrain. Work done in [22], assume that the localization of the sensors has been successfully computed in order to realize a target detection scenario. Instead of utilizing a binary detection model which depends on the range of sensors, probabilistic detection is used with the function $e^{\alpha d}$ where d is the distance and α is the detection quality of the signal. The deployment area is modeled as a grid where sensor at a grid point i is supposed to detect a target at a grid point j, whose probability is p_{ij}. Without a terrain p_{ij} and p_{ji} would be equal, but in the presence of terrain they are not equal, since "a sensor at a lower elevation is unlikely to detect a target at a higher elevation, but a sensor at a higher elevation can detect a target at a lower elevation", [22]. Here, obstacles are modeled by altering detection probabilities of grid pairs, occlusion either leads to zero detection probability or reduces it. Line equations are used in this work in order to determine the LOS. However, terrain structure is modeled only through random obstacles.

In [23], WSN is used to determine a sniper's location and bullet's trajectory by

measuring the arrival of acoustic events in a terrain environment. According to a classification, the sensors eliminate multipath effects, and only consider LOS signals. A WSN is also utilized in a snow monitoring application for avalanche probability estimation [24]. The project is currently developing models for mountainous terrains and running simulations on the terrain maps to determine the optimal node placement. In [25], environmental effects such as obstruction of sensor signals from the terrain is taken into account when performing localization in WSN. In order to avoid miscalculation of distance of acoustic signals, when LOS is obstructed, cameras are placed on beacon nodes to detect which sensor pairs have LOS and which do not, and thus determining which signals must be taken into account when performing localization. Another localization related work has been done in [26]. Here localization is performed by taking into account the anisotropic network topology and complex terrain properties. The Multidimensional Scaling (MDS) technique which is a dissimilarity analyzer, can determine spatial structures in the data. MDS analyzes the dissimilarity of distance information between sensors and deduce their coordinates.

2.2.3. WSN Modeling in 3D

Many existing results on WSN algorithms in Section 2.2.1 are evaluated on planar networks, however in real life, random placement of nodes take place in inaccessible terrains. Therefore, three dimensional settings must be considered to reflect real life situations, [27].

In [2], a solution for 3D coverage problem is tackled at polynomial time. A field is said to be α-covered if every point in the sensing field is at least covered by α sensors assuming that the sensing range of a sensor is modeled by a 3D ball. α-coverage is related with how many neighbouring balls intersect a sensor's ball. In [3], the deployment problem in 3D is analyzed. For energy efficiency, minimum number of sensors serve as an optimality measure for WSN. An optimality measure is the average number of spheres that contain a point in the field. Work in [28] also tries to solve the coverage and *coverage hole* problem in 3D. The goal is to minimize the number of sensors and guarantee no *coverage hole* in the target area which means that every

point in the area is covered by at least one sensor. 3D space coverage for tactical underwater sensor networks is analyzed in [29]. Sensors are initially deployed in water via buoys. They can be lowered to various depths to maximize 3D space coverage. The optimization problem is defined as maximizing the space covered by each node and average distance between node pairs.

2.3. Target Tracking

Target tracking, which is defined as the processing of measurements obtained from a target to get an estimate of its current state is another research challenge in WSNs. Target tracking has its applications in Command, Control Communications, Computer Intelligence, Surveillance and Reconnaissance (C4ISR) in military, [30]. The spatial coverage, and multiplicity in the sensing task, which comes from the distributed nature of WSNs, facilitate the tracking of targets in coordination to improve the target localization accuracies. Targets to be tracked could be either vehicles or people traversing across the range of many sensors in the phenomenon. The tasks associated with tracking, namely the *detection*, *classification*, and *tracking* of moving, low-observable events require non-local collaboration among sensors [31], which in turn brings new challenges with it such as, *sensor information fusion*, *communication*, *sensor management* and *decision making*.

Data fusion, in other words, aggregation of network data in WSNs, reduces tracking errors. Sensor collaboration based on sensor selection saves bandwidth by preventing redundant packets and data requests wandering in the network, thus prevents link failures and increases the network lifetime. In data fusion, raw signals that are sampled at individual nodes, are not directly communicated to other nodes. Instead the summary of statistics are stored locally and these statistics which are small in size are relayed to other nodes upon request.

The main problem in the collaborative target tracking is to answer, *who should sense at any given time*, *what has to be sensed* and *who the information must be transmitted to*. An answer to these problems, in the observation of non-local moving

events, would to wake up the sensors in the region where the event has been predicted to move into, or where there has been considerable physical change. For low-observable events, the collaboration can include dynamic selection of sensors' data, in order to make data fusion and thus improve detection accuracy. These issues are also important in terms of network efficiency. At a given time many sensors make observations and flood the network with information packets. This necessitates smart decisions on who should sense at any given time.

In collaborative target tracking, not all sensors provide useful information as to target state estimation. In the process of incrementally updating the target estimate by adding in the measurements of neighbouring sensors, a decision has to be made as to selecting an optimal subset and optimal way of adding in these measurements into the current belief. The criteria for deciding on the optimal subset mostly uses the sensor position, sensing modality information without communicating data regarding the measurements at sensors [31].

As discussed before, the task of tracking brings the energy optimization problem with it. Scarce energy resources, e.g., limited battery leads to low detection probability and false alarms, resulting in over–flow data wandering in the network. Therefore a target tracking application should take energy efficiency as the primary concern in the algorithm design. One approach to conserving energy-efficiency is to perform the signal processing tasks only that are relevant to the current query. When there is no query the node stays in the stand-by mode. Also, the node does not publish the processed information unless it is asked for it.

In [31], the target tracking problem is presented as an information optimization problem. This approach has been applied to sensor querying and data routing. This enables a sensor to make a decision on sensing and communication based on the information gain and cost. The information gain is represented in Figure 2.3.

The uncertainty in estimation is modeled by a Gaussian distribution, in the ellipsoids. The dashed ellipsoids represent the updated beliefs after incorporating mea-

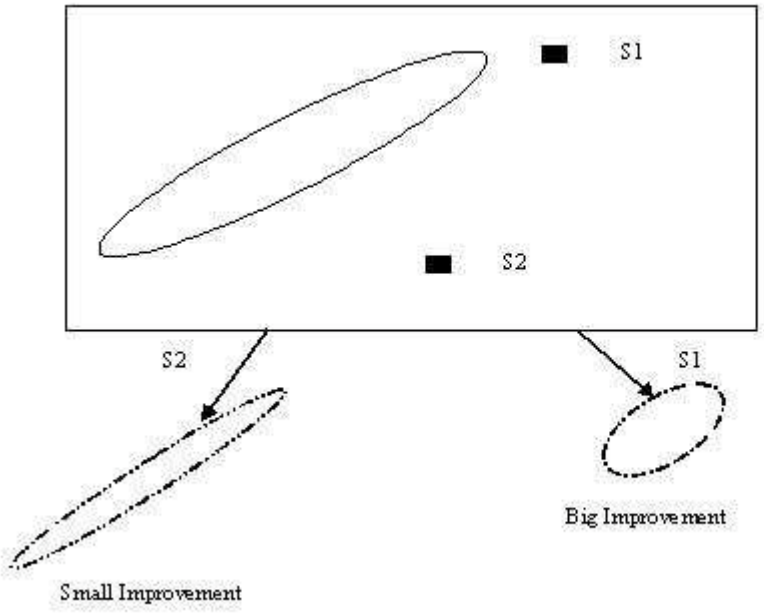

Figure 2.3. Sensor selection based on information gain

surements either from S_1 and S_2, while the solid ellipsoid represents the current belief. Although the area of uncertainty is reduced by the same amount in both of the cases, S_2 has the longer principal axis in the uncertainty distribution, therefore S_1 is selected over S_2.

In [32], a *location-aware* data routing approach that limits the scope of collaborative signal processing to relevant subset of nodes has been adapted. Location aware routing means, routing of information in a WSN should be based on geographies rather than the nodes. In other words, the geographic location of the nodes determine the routing rather than arbitrary node identities. The relevant nodes are selected from those which conserve networks resources, e.g., energy and bandwidth. The phenomenon of interest is divided dynamically into spatial cells. Within each cell, the cell manager coordinates the tasks. The tracking task consists of five steps which is explained in Algorithm 1. The location-aware routing restricts data distribution to regions, directly affected by the data.

In [33] authors proposed similar solutions to energy-efficient target tracking. Moreover, they also handled the multiple target tracking problem. Similarly, they

Algorithm 1 Target tracking algorithm [32]

1: *Cells that are near to the potential target are put on alert.*
2: *If the target is detected, the cell becomes active.*
3: *Tracking is done on the target of the desired type, which includes target location, speed and direction estimation for future predictions.*
4: *Target related data is sent to other cells based on the predictions.*
5: *The alerted cells, where the target is detected, begin tracking task and the process repeats.*

divided the region into geographic locations, named cells, and predicted target motion from one cell to another. The cells which the target is predicted to move into are activated for detecting potential targets. The nodes in the activated cells run the five-step algorithm in Algorithm 2.

Algorithm 2 Target tracking algorithm [33]

1: *Perhaps all the nodes in the activated cells detect the target, and store the timing of detection together with the detection data. The nodes who detected the target report their energy detector outputs to the manager node.*
2: *At the following time instants, the manager node tries to localize the target based on the energy outputs of the nodes. The simplest algorithm for localization, is to base the location on the highest signal. There are more sophisticated algorithms for localization at the cost of higher complexity.*
3: *The manager node uses information of target positions from the signal outputs, to predict the future target positions.*
4: *The predictions form the cells the target is most likely to move into. These cells are activated for potential target motion.*
5: *One of the potential cells is activated since now the target is in that cell, others go into stand-by mode for energy conservation.*

Authors also considered multiple target tracking problem, which is divided into two cases: 1) The targets occupy different cells in time and space. In this case, Algorithm 2 is applied for each track. 2) The restriction of targets in time and space is too restrictive. In such a case, classification algorithms are needed to operate on spatial-temporal target signatures. This necessitates priori statistical knowledge about

types of targets.

In [34], authors proposed a hierarchical multiple target tracking algorithm that is robust against false alarm rates and low detection probability. The target tracking algorithm is capable of initiating and terminating the tracking task. The observations in nodes are first fused locally and then transmitted to super nodes, thus reducing the network communication. The sensor network model is based on a 2D surveillance region. Putting some of the nodes into dormant state by sleep schedule [35] is another trend for energy efficient target tracking applications. In [36] nodes that are far away from targets go to sleep and save energy while guaranteeing accurate tracking and timely delivery.

In [30], the performance of a collaborative target tracking based on maximum mutual information-based sensor selection has been evaluated with regards to Kalman and Information filter. Mutual information between the target state and the observation measures how much the current observation tells about the current target state. The maximum mutual information based sensor selection algorithm for the distributed data fusion architecture is depicted in Figure 2.4. The algorithm defined in Figure 4.1 is a more refined form of this one.

The authors defined a mutual information gain, J. If J is sufficiently large, then the sensor shares its information with the neighbouring nodes, otherwise it does not transmit. To decide whether its own mutual information is sufficiently high, a sensor should also know the neighbouring sensors' mutual information. This can be estimated from the neighbouring sensors' characteristics (e.g., standard deviation of target range observations, standard deviation of target bearing observations, communication transmission power) and position information. The work also adapts an Information Controlled Transmission Power adjustment (ICTP) scheme in which the communication transmission power of a sensor is regulated according to its information content. Upon the simulations, authors showed that the proposed method gives better results than Kalman and Information filter, in terms of energy usage for the desired target localization accuracies. They also compared the performance of the sensor selection algorithm

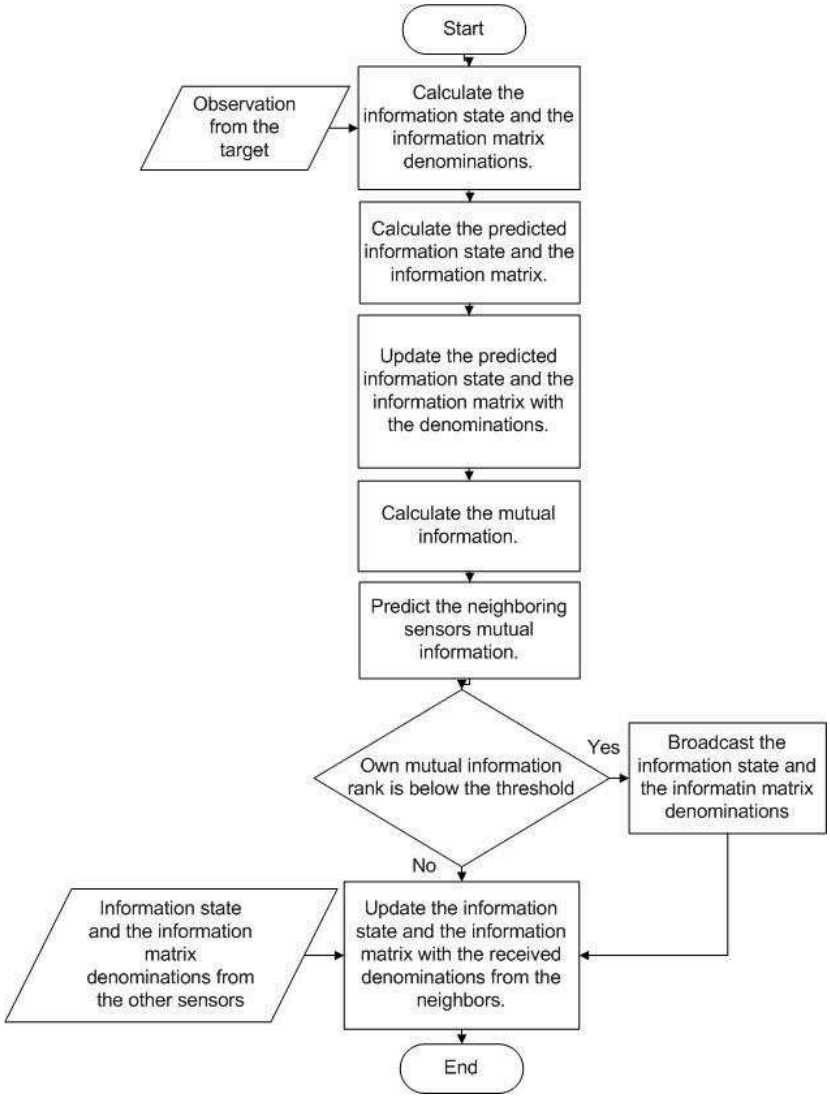

Figure 2.4. Target tracking algorithm according to maximum mutual information based sensor selection

with regards to minimum Mahalanobis distance based selection and random selection and showed the proposed method has achieved better performance in terms of mean error at a given channel capacity and energy exhausted at a given mean error.

This thesis uses a collaborative target tracking application in the performance evaluation of a WSN. It makes a comparative performance study between various terrain types and the 2D target tracking model that we take as reference [30]. Our study takes the collaborative data fusion part (excluding the maximum mutual information based sensor selection and ICTP scheme) of this work. The basics of the referenced

model has been explained in detail in Section 4.2.

3. 3D TOPOGRAPHY GENERATION

3D terrain generation is achieved with Perlin Noise in 2D in Geofrac environment, [37]. The resulting heightmap is exported to Terragen terrain format [38], for visual purposes, e.g., rendering, and further processing by the sensor network simulator program. This chapter aims to give a background information on 3D topography generation, i.e., background on heightmaps, heightmap processing, terrain generation and rendering tools that we use, and finally terrain generation methods.

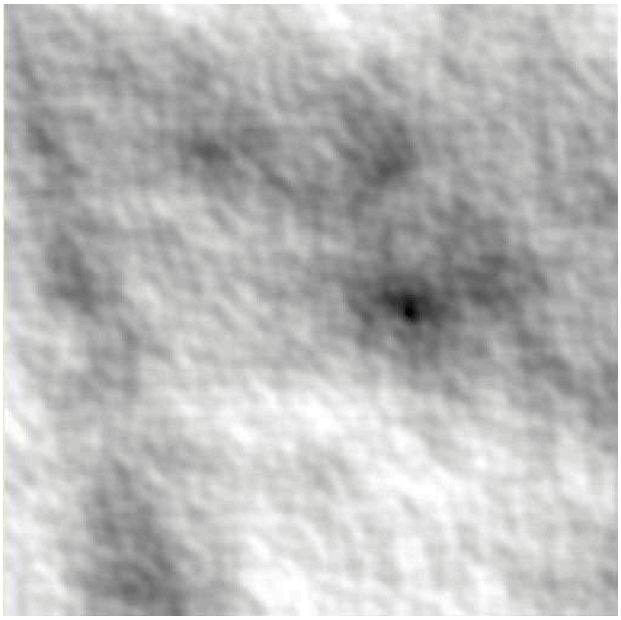

Figure 3.1. A heightmap

3.1. Heightmaps

Heightmaps are 2D representation of terrains in a grid form, consisting of two-dimensional array of height values. The x and y coordinates are associated with z values, which represent the terrain height. To simulate a random continuous terrain at rendering time, points between grid points are interpolated. During processing of the heightmap, three points with coordinates (x, y, z) define a triangle in 3D space. Heightmap processing is explained in the next section.

By assigning a color to each height value, a heightmap could be displayed as an

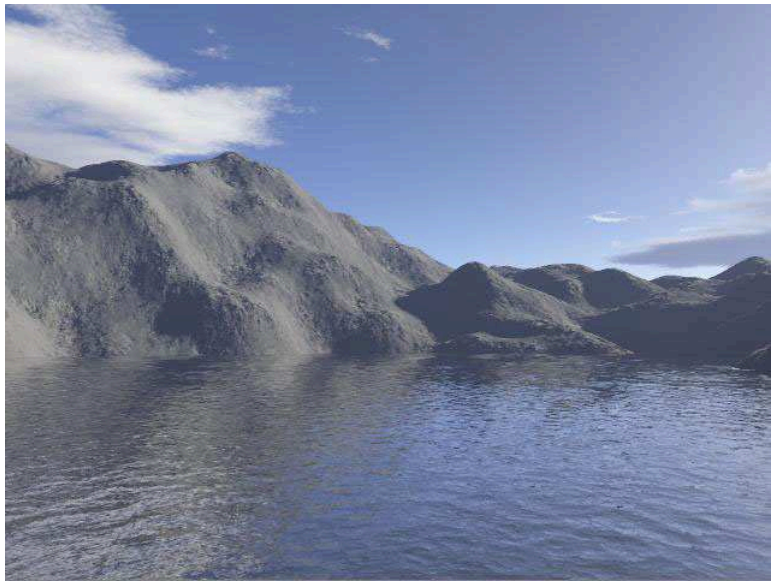

Figure 3.2. Rendered form of heightmap in Figure 3.1 (rendered in Terragen)

image. Higher points in the terrain (large values) are represented by white, and lower points (small values) are represented by black as seen in Figure 3.1. The rendered form of the heightmap can be seen in Figure 3.2.

3.2. Heightmap Processing

The terrain consists of grid points with height values. Each grid square is divided into two triangles as seen in Figure 3.3. Figure 3.3 denotes the 2D top view of the heightmap and the triangles.

The LOS between two points is calculated according to the ray-triangle intersection algorithm in [39, 40]. The function takes 5 vectors as inputs: *orig*, *dir*, *vert0*, *vert1* and *vert2*. The *orig* is the starting point vector and *dir* is the unit direction vector from the start vector to the end vector. Each vector is defined by three coordinates in 3D space. The x and y, are the grid coordinates with z being equal to the terrain height plus sensor height (10 cm) or target height (150 cm). The ray-triangle intersection algorithm returns whether the ray intersects the triangle defined with the vertices vectors, *vert0*, *vert1*, and *vert2*. The algorithm transforms the origin of the vector and returns the vector $(t, u, v)^T$ where t is the distance to the intersection point and (u, v) are the coordinates of the intersection.

The ray-triangle intersection algorithm is applied to every two triangles in each grid of the rectangle between the source and destination point, e.g., in Figure 3.3 at most 12 triangles are processed. If the algorithm intersects one of the triangles before reaching the final triangle, it does not proceed further. An optimization could be developed in terms of processing less number of triangles, however this has not been studied in the scope of this thesis.

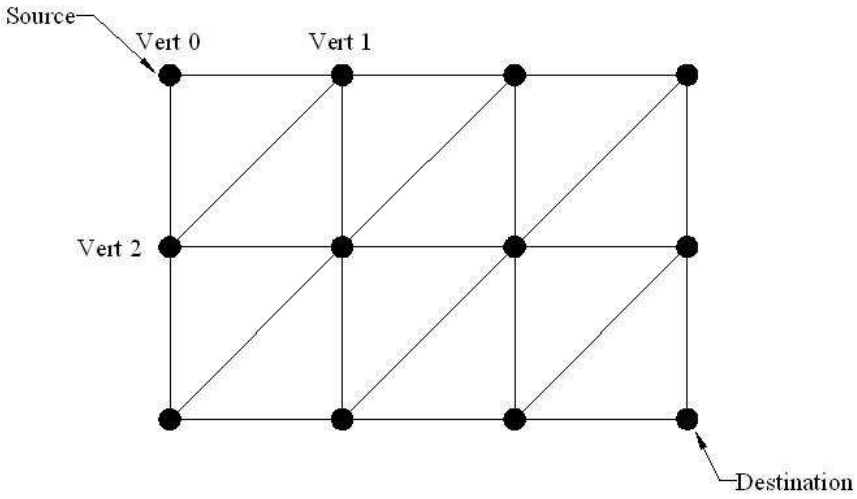

Figure 3.3. Heightmap grid

3.3. 3D Terrain Generation Tools

- GeoFrac

  Geofrac2000 [37], is a shareware program for generating heightmaps of realistic terrain data and outputting texture maps, 3D and Polygon Meshes. As well as generating a random terrain, it has various tools for manipulating 2D heightfields such as sculpting, digging, smoothing, and elevating terrains. These methods are applied either using sophisticated techniques or manual modifications via grayscale painting. Random terrains are generated with various methods, only two of them are discussed here. Perlin Noise (Section 3.4.1) and Midpoint Displacement (Section 3.4.2).

- Terragen

  Terragen [38], is a work-in-progress scenery generator that creates near-photorealistic

landscape images and animations. It is also capable of creating special effects, art and recreation on landscape images.

Besides its terrain generation capabilities, Terragen is a rendering program. It has also export/import options between various formats, e.g., RAW, SGI and LIGHTWAVE 3D objects. In our study, we only used the file format in heightmap processing and rendering capability of the program, in order to present the visual image of the terrain where the target tracking applications run. Previously, we used Terragen to export the terrain into RAW file format, in order to be processed by the Matlab simulation program. However we have seen that the terrain structure degenerate when the heightmap values are converted to 8 bit unsigned values. We have used Matlab script [41] for processing our terrains in the Terragen format. The terrain file format is given in Appendix A.

3.4. Terrain Generation Methods

3.4.1. Perlin Noise

In nature many phenomena exhibit fractal behaviour: *large* and *small variations*, [42]. The examples could be seen in the distribution of patchy grass on a field, waves in the sea, the movements of an ant, the movement of branches of a tree, patterns in marble, and in winds. Also they have various levels of detail, when carefully noticed. A common example is the pattern of a mountain range. When it contains large variations in height, it becomes a *mountain*, medium variations come out as *hills*, small variations come out as boulders and tiny variations come out as *stones*. The Perlin Noise generates this pattern by simply adding up noise functions at different scales.

However, random number generators are not enough to create a Perlin Noise, although they are common in computer programs for generating unpredictability. Random number generators are generally used for making objects move and behave more naturally. However, sometimes could output too sharp results [42], so that the resulting patterns do not look natural.

In this thesis the landscapes (terrains) are generated with 2D Perlin Noise, since the landscapes are perfect application areas for 2D Perlin Noise. In the generation phase, the landscape does not need to be stored anywhere in memory, and the height of any point can be calculated easily, unlike the subdivision method, (Section 3.4.2.2).

Perlin noise is also suitable for generating textures, e.g., marble, wood, swirly patterns. A 3D texture could also be defined. The procedure can be thought as a solid block of material, from which an object can be 'carved'. This allows us to produce textures which can be applied to any shaped object without distortion.

3.4.1.1. Noise Functions. A noise function is a seeded random number generator. An integer is the input as a parameter to the generator, and a random number based on that parameter is the output. It must produce the same output if input the same parameter. The discrete noise function in Figure 3.4 is obtained by assigning a random value to every point in the X axis. In Figure 3.5 a continuous function can be defined that takes a non-integer as a parameter, by smoothly interpolating between the values.

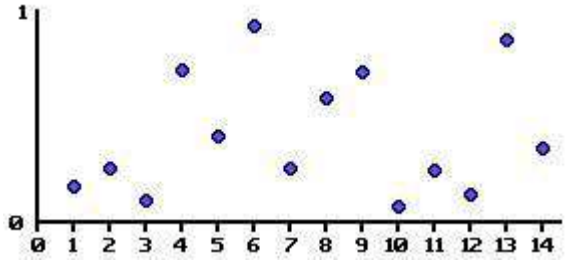

Figure 3.4. Discrete noise

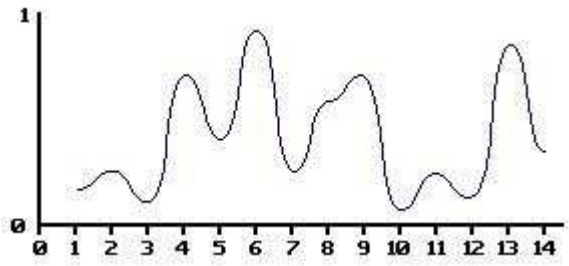

Figure 3.5. Interpolated noise function

3.4.1.2. Background on Noise Functions. There are two basic parameters associated with a sin wave.

- The wavelength of a sinusoidal wave is defined as the distance from one peak to another, whereas the frequency is defined to be $\frac{1}{wavelength}$.
- The amplitude is the height of the wave.

In the noise function of Figure 3.6, the spots are the random values along the dimension of the function. Here, the amplitude is the difference between the minimum and maximum values the function could have. The wavelength is the distance from one spot to the next and the frequency is defined as $\frac{1}{wavelength}$.

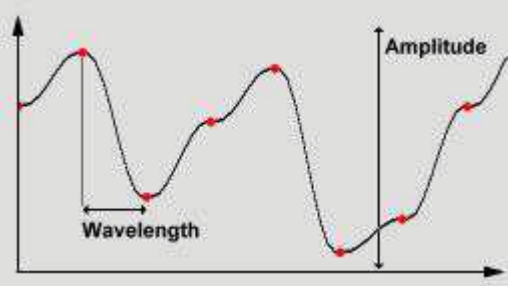

Figure 3.6. Noise Wave

3.4.1.3. Generating Perlin Noise. If lots of such smoothed functions with various frequencies and amplitudes are added up, a noise function is created. This is the Perlin Noise Function as in Figure 3.7. It is obvious that this function has quite variations of type large, medium and small.

The same technique could be applied in 2D and generate a 2D Perlin Noise as seen in Figures 3.8 and 3.9. Many computer generated landscapes are made using this 2D noise functions. With 2 dimensional noise, the numbers input are the coordinates. The noise function defines every (x, y) point. The most common use of this is textures.

The one dimensional example in Figure 3.7 uses twice the frequency and half the amplitude for each successive noise function added, which is quite common. However, Perlin Noise functions can also be created with different characteristics by using other frequencies and amplitudes at each step.

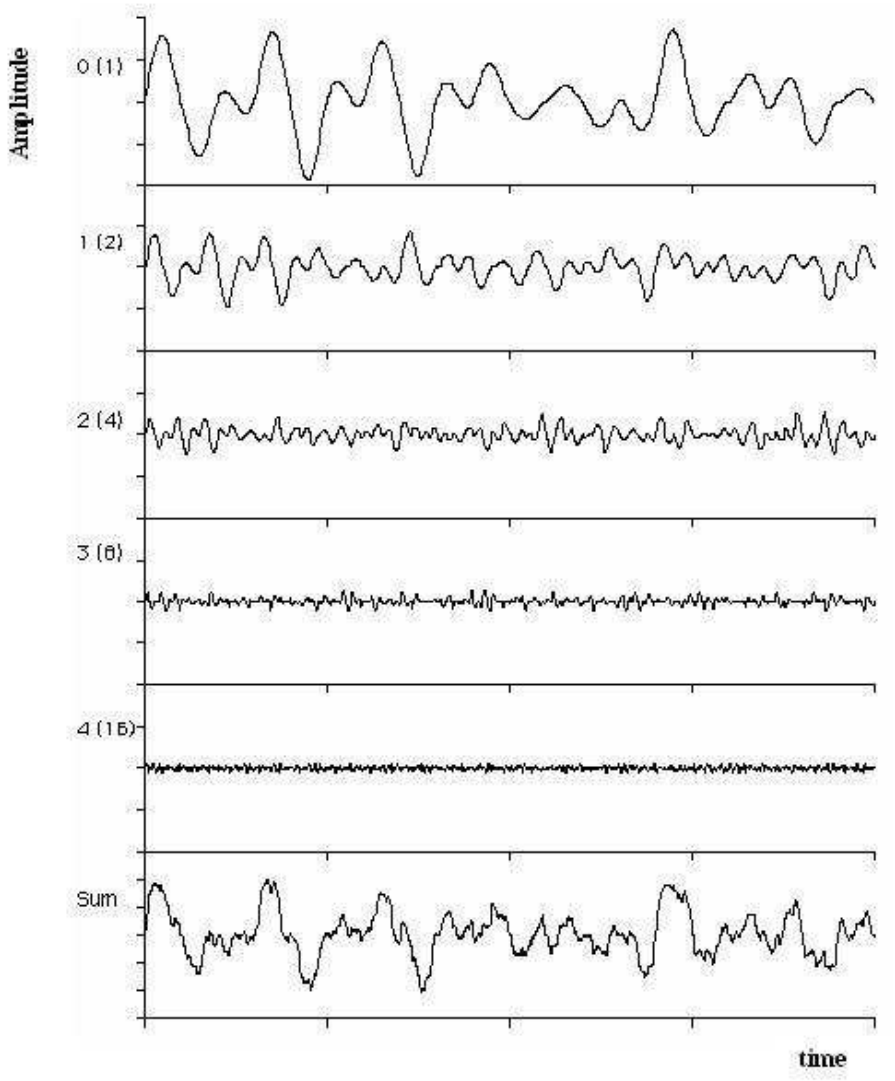

Figure 3.7. Perlin Noise Function

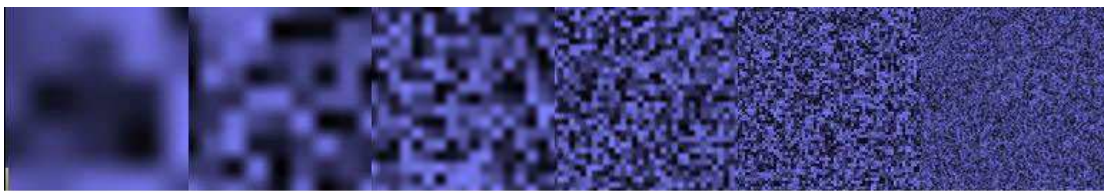

Figure 3.8. 2D Noise Functions

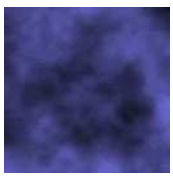

Figure 3.9. 2D Perlin Noise Function

3.4.1.4. Interpolation. The values in the noise function needs to be smoothed out, in order to look like a realistic object – in our case the landscape. A standard interpolation function takes three inputs; a and b, the values to be interpolated between, and x which takes a value between 0 and 1. Based on the value of x, the interpolation function returns a value between a and b. It returns a when x equals 0, and it returns b when x is 1. It returns some value between a and b, when x is between 0 and 1. There are three interpolation types commonly used, [42]:

Linear Interpolation

Although it is a simple algorithm, it might not give good results.

```
function Linear Interpolate(a, b, x)
    return a*(1-x) + b*x
end of function
```

Cosine Interpolation

This method gives a much smoother curve than Linear Interpolation, although it might result in a slight loss of speed.

```
function Cosine_Interpolate(a, b, x)
    ft = x * 3.1415927
    f = (1 - cos(ft)) * .5
    return a*(1-f) + b*f
end of function
```

Cubic Interpolation

This method gives very smooth results, but much slower than the other two methods described. The interpolation functionS described above took three inputs, the cubic interpolation takes five. Instead of a and b, we now need $v0$, $v1$, $v2$ and $v3$, along with x as before. These are:

```
function Cubic_Interpolate(v0, v1, v2, v3,x)
    P = (v3 - v2) - (v0 - v1)
    Q = (v0 - v1) - P
    R = v2 - v0
    S = v1
    return Px3 + Qx2 + Rx + S
end of function
```

3.4.1.5. Smoothed Noise. Aside from interpolation, the output of the noise function can also be smoothed to make it less random looking, and also less square in the 2D version, [42].

Rather than taking the value of the noise function at a single coordinate, the average of the value, and it's neighbouring values could be taken.

In Figure 3.10, the diagram illustrates the difference between smoothed noise, and the same noise function without smoothing. It is clear that the smooth noise is flatter, without the extremes of unsmoothed noise, and the frequency is nearly the half. Unlike in one-dimension, smoothing is more useful in 2D, where its effect is to reduce the squareness of the noise (Figure 3.11).

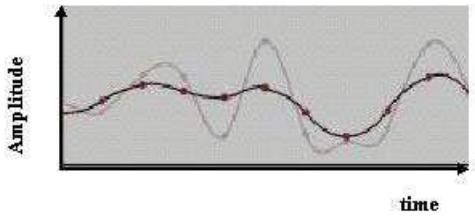

Figure 3.10. Smoothed and unsmoothed noise

3.4.2. Midpoint Displacement

3.4.2.1. Midpoint Displacement in One Dimension. One-dimensional midpoint displacement is a suitable algorithm for drawing the ridgelines, since mountains appear on a distant horizon, [44]. The pseudocode is shown in Algorithm 3.

Figures 3.12 through 3.14 illustrate generating a fractal through Midpoint Dis-

Figure 3.11. Smoothed and unsmoothed noise in 2D

Algorithm 3 Midpoint displacement algorithm

```
1: Start with a single horizontal line segment
2: for all i sufficiently large do
3:    for all line segment in the scene do
4:       Find the midpoint of the line segment
5:       Displace the midpoint in Y by a random amount
6:       Reduce the range for random numbers
7:    end for
8: end for
```

placement. In Figure 3.12, the random number range is $[-1.0, 1.0]$, so the midpoint of the line is displaced by a random amount between $[-1.0, 1.0]$. In Figure 3.13, the range is reduced to half and this time, it is $[-0.5, 0.5]$. The two midpoints are displaced by the random amounts in this range. Finally in Figure 3.14, the range is $[-0.25, 0.25]$ and the four midpoints are displaced by four random numbers in this range.

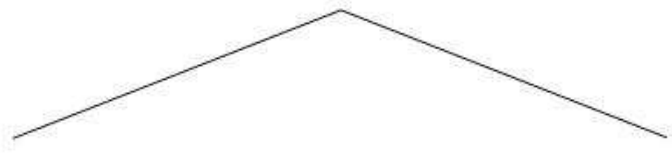

Figure 3.12. Midpoint displacement: first iteration

The roughness of the fractal depends on how much the random number range is reduced. If it is reduced more during each pass of the loop, the ridgeline will look smoother. Otherwise, ridgeline will be jagged.

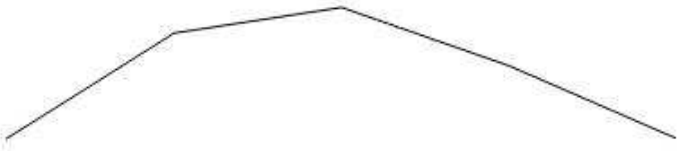

Figure 3.13. Midpoint displacement: second iteration

Figure 3.14. Midpoint displacement: third iteration

3.4.2.2. Midpoint Displacement in Two Dimensions. We start with an empty 2D array of points. Array must be square and the dimension should be a power of two, plus one (e.g., 33x33, 65x65, 129x129). The four corner points are set to the same height value. Here, we used a 5x5 array as an example, (Figure 3.15). The four corner "seed" values are highlighted in black.

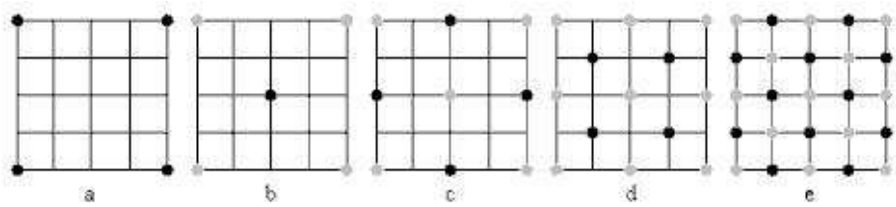

Figure 3.15. Midpoint displacement algorithm in 2D

The iterative subdivision routine has two steps, [44]:

The diamond step: Taking a square of four points, we generate a random value at the square midpoint, where the two diagonals meet. The midpoint value is calculated by averaging the four corner values, plus a random amount. This gives the diamonds when multiple squares are arranged in a grid.

The square step: Taking each diamond of four points, we generate a random value at the center of the diamond. We calculate the midpoint value by averaging the corner values, plus a random amount generated in the same range as used for the diamond step. This gives the squares again.

In the first pass, at *diamond step*, a value at the center of the array based on the height of the four corner values is generated. The four corner values are averaged and added a random value from the range -1.0 to 1.0. This new value is shown in black, and the existing corner values are shown in gray in Figure 3.15 b.

At *square step*, the same range is used for generating the random values. There are four diamonds at this stage; which meet in the center of the array. Four diamond centers are calculated this time. The corners of the diamonds are averaged to find the base for the new values. Figure 3.15 c shows the new values in black and existing values in gray.

In the second pass, we again start with the *diamond step*. The second pass has four squares instead of one, unlike the first pass. This time, four square centers are calculated. Also, the range for generating random numbers is reduced. For example, the random number range is reduced from (-1.0, 1.0) to (-0.5, 0.5). In Figure 3.15 d, the four square center values that are calculated at this step are shown in black.

Finally, the *square step* in the second pass is calculated. With 12 diamond centers, 12 new values are calculated, which are shown black in Figure 3.15 e.

3.5. Terrain Generation with Geofrac

This section demonstrates various terrains under different terrain generation parameters. Terrains are generated using Perlin Noise function. The *harmonics* and the *frequency* determines the turbulence of the terrain while the *amplitude* determines the vertical scale in the heightmap grid points.

Terrains generated with Perlin Noise, are stored as heightmaps in the Geofrac and exported to Terragen for rendering. Both in the terrain generation and export phase, the vertical scale (*amplitude*) can be adjusted, which also affects the topography. Figures 3.16 through 3.23 are rendered in Terragen for demonstrative purposes.

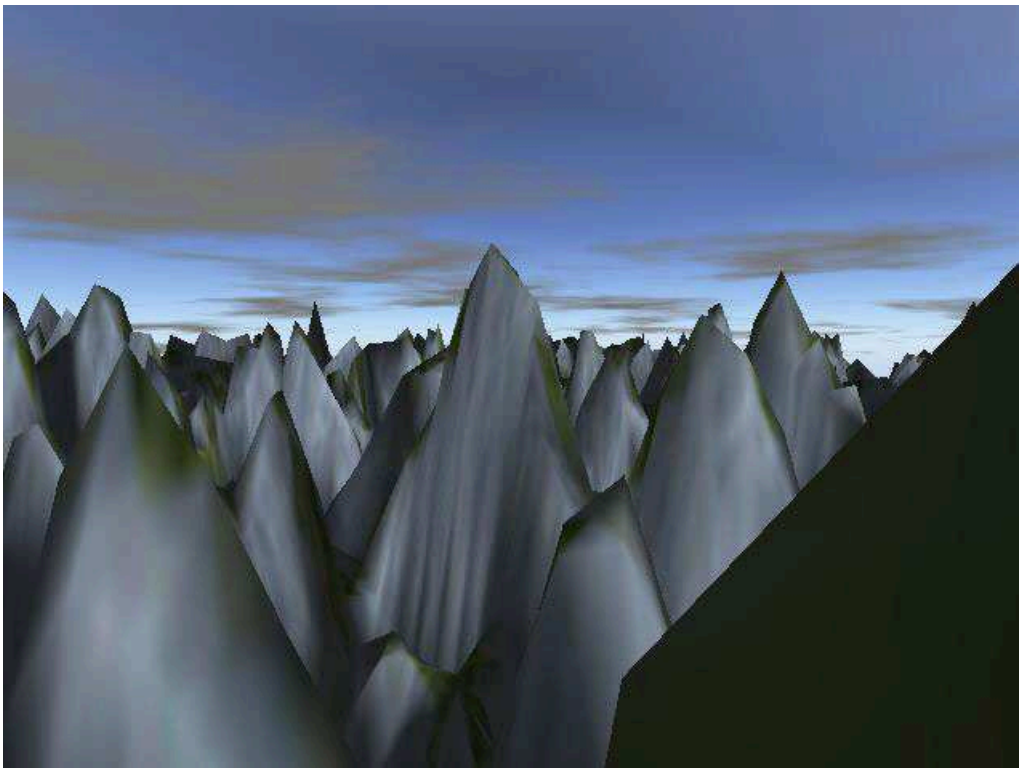

Figure 3.16. Terrain generated with Perlin Noise: *harmonics*=1, *frequency*=0.15, *amplitude*=50

Either decreasing the *harmonics* or increasing the *frequency* causes a wiggliness, sharp peaks in the terrain. The effect of *harmonics* on the topography could be seen more clearly on Figures 3.16 through 3.19. Here, while the *frequency* is set to 0.15, *harmonics* increase. As the *harmonics* increase, the terrain gets smoother, as in Figure 3.19. At a fixed *harmonics* value, terrain gets peaky as the *frequency* is increased as demonstrated in Figures 3.19 through 3.22.

The effect of the amplitude on the topography of the terrains can be seen in Figures 3.21 and 3.23. Both of the terrains have been created by the same harmonics and frequency values. However, the former looks much smoother than the latter. It comes from the fact that they have different vertical scales, although the turbulence is the same. It is clear that Figure 3.23 is much more natural looking than Figure 3.21. The high frequency values when combined with a small vertical scale, resulted in small ledges on the surface in Figure 3.23, whereas the outcome of the high frequency when combined with a big vertical scale resulted in shaper peaks in Figure 3.21. Terrain 2 (Figures 4.4 and 4.5) is similar to the terrain in Figure 3.23. Although they have the

Figure 3.17. Terrain generated with Perlin Noise: *harmonics*=2, *frequency*=0.15, *amplitude*=50

same parameters, they are not exactly the same, since the program creates the terrain randomly at each time.

We can conclude that high frequency values mostly result in unrealistic terrains, with sharp peaks, unless they are combined with small amplitude values, which convert the sharp peaks into small ledges on the surface.

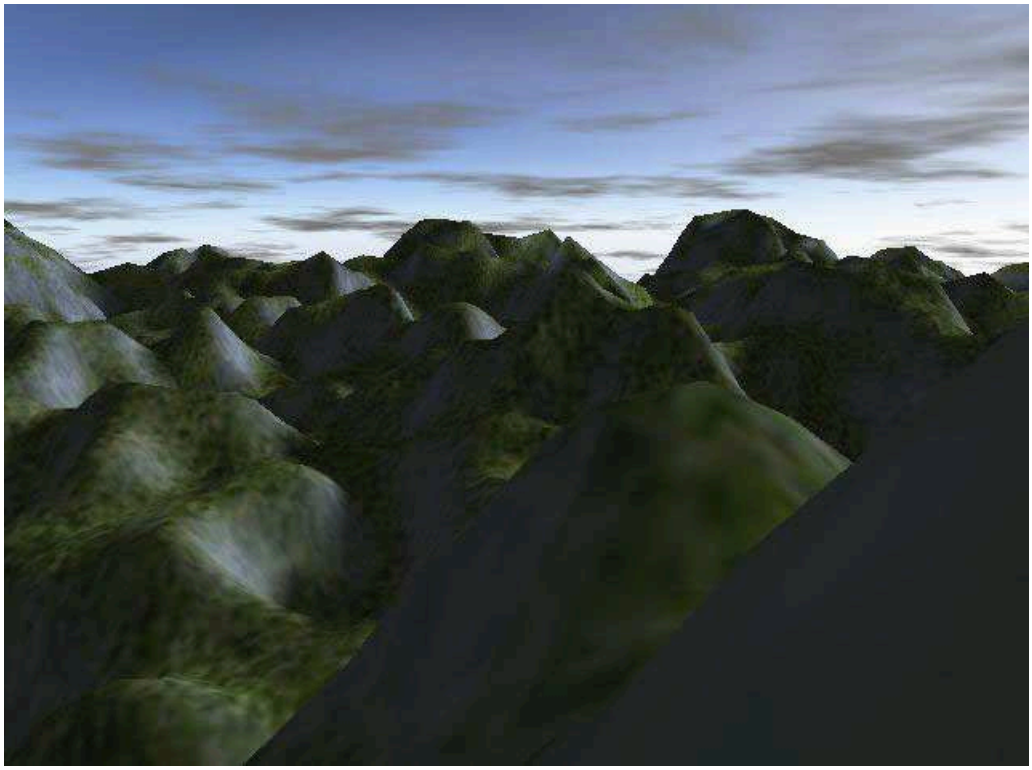

Figure 3.18. Terrain generated with Perlin Noise: *harmonics*=4, *frequency*=0.15, *amplitude*=50

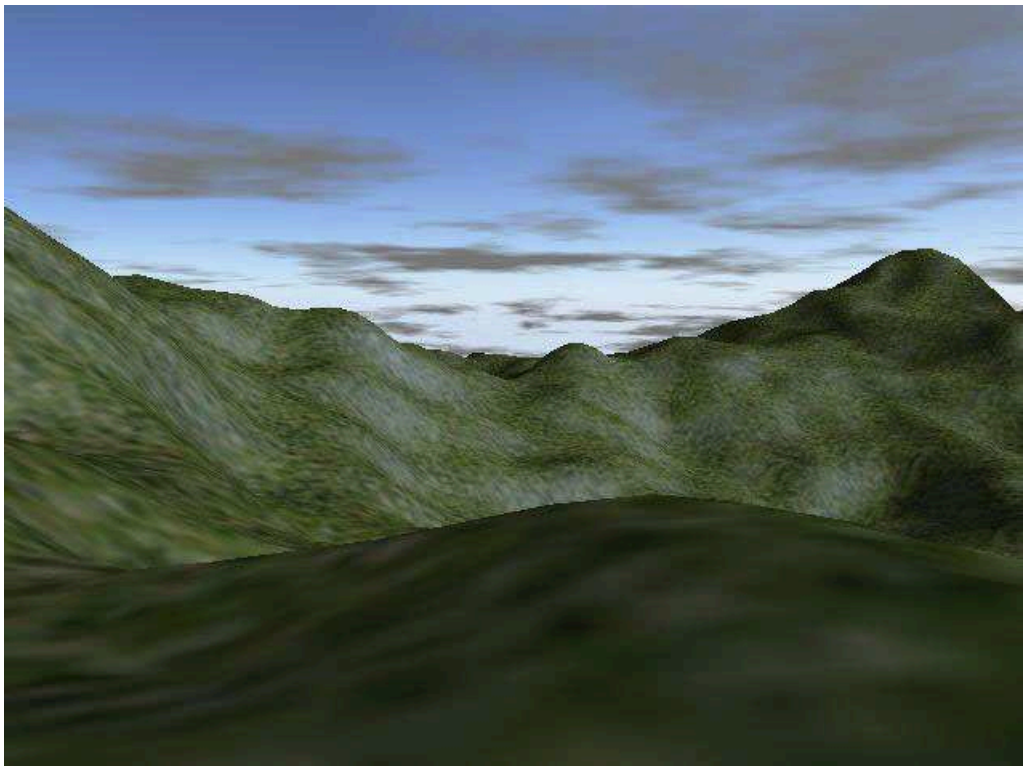

Figure 3.19. Terrain generated with Perlin Noise: *harmonics*=6, *frequency*=0.15, *amplitude*=50

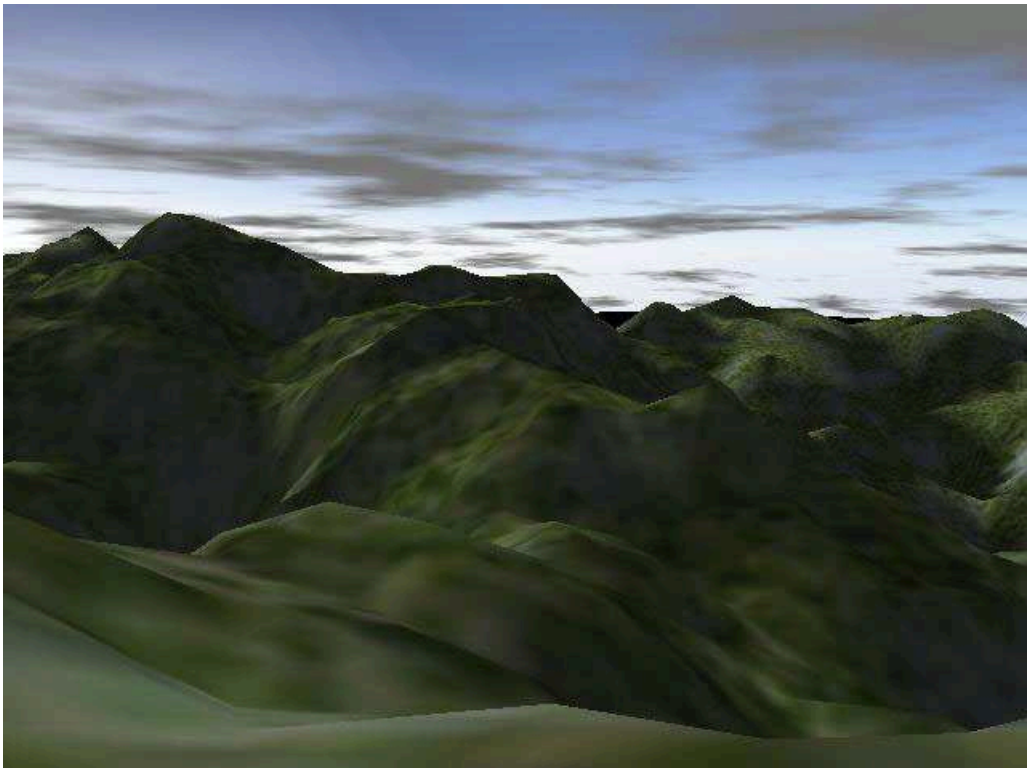

Figure 3.20. Terrain generated with Perlin Noise: *harmonics*=6, *frequency*=0.3, *amplitude*=50

Figure 3.21. Terrain generated with Perlin Noise: *harmonics*=6, *frequency*=1, *amplitude*=50

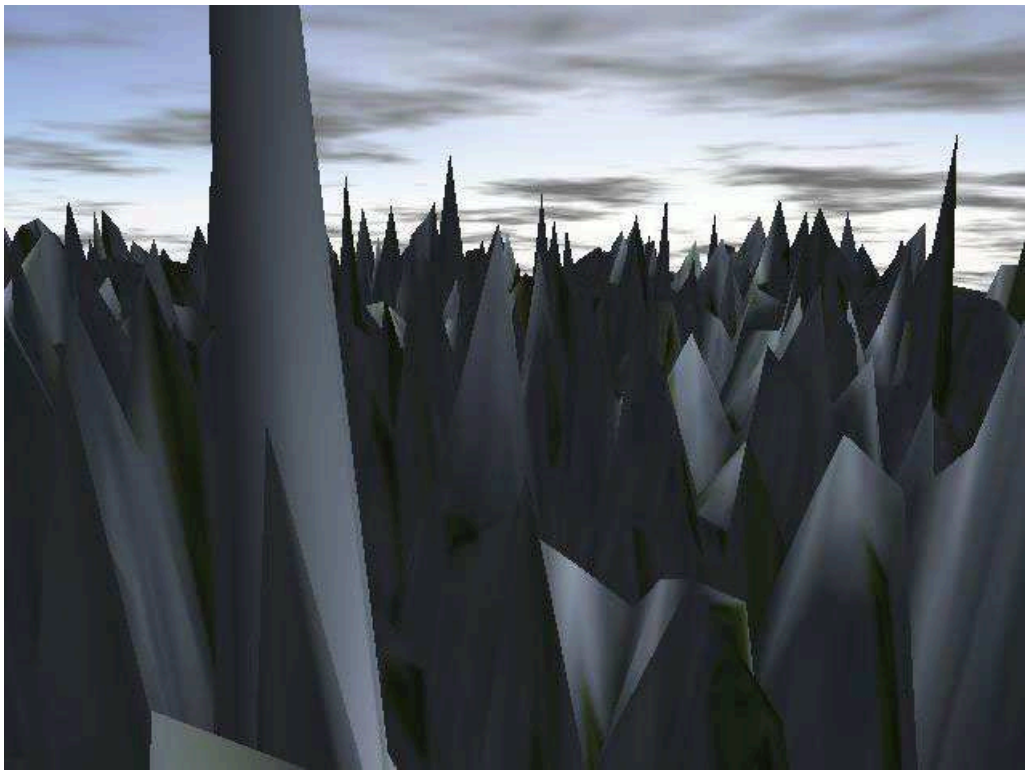

Figure 3.22. Terrain generated with Perlin Noise: *harmonics*=6, *frequency*=10, *amplitude*=50

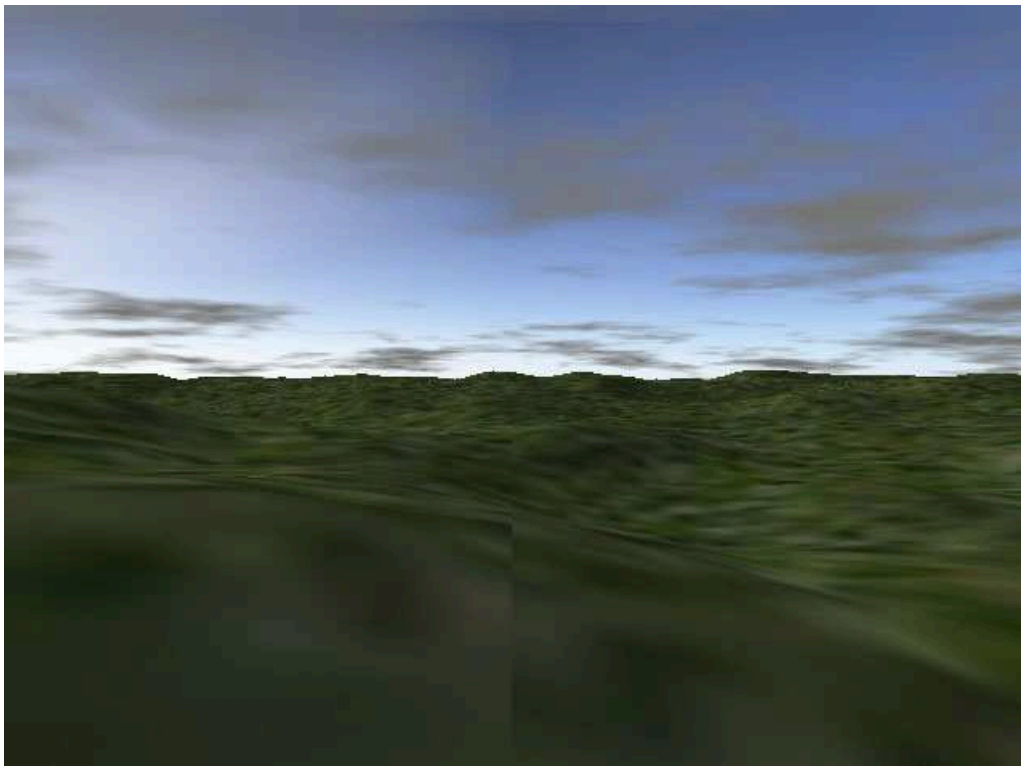

Figure 3.23. Terrain generated with Perlin Noise: *harmonics*=6, *frequency*=1, *amplitude*=5

4. BASICS FOR 3D TOPOGRAPHIC WSN EVALUATION IN TARGET TRACKING

Many existing performance evaluations on WSN take place on planar networks, which do not reflect the real life situations. These evaluations assume a random deployment scheme in evaluations. However, sensor nodes are randomly deployed in *inaccessible terrains*, where the terrain properties, e.g., ledges, rocks may block line of sight conditions, between sensor nodes and the sensor node and the target. This work aims to show the performance of a target tracking application under a realistic scenario and depicts whether the 2D results are realistic enough since the communication and sensing capabilities change in 3D terrain. The evaluations are done on various terrains, including hilly, smooth, ledgy terrains and aim how different terrains affect communication and sensing capabilities and resultantly the performance of the target tracking application.

This chapter aims to provide the background on the target tracking model we take as a reference and the terrain environment where the sensors are deployed. In Section 4.1 the need for unbiased converted measurements in target tracking application is explained. Section 4.2 aims to provide the necessary background on target tracking. In Sections 4.2.2 and 4.2.3 the observation model parameters in 2D/3D are derived and also 3D to 2D mapping in the terrain environment is explained, and finally in 4.2.4, the data fusion architecture is defined.

Section 4.3 explains sample surfaces; the terrains where the WSN target tracking application is evaluated, together with their generation parameters. The terrains are shown in their rendered forms.

4.1. Unbiased Target Tracking

In target tracking applications, cartesian coordinates are the most suitable form of representing target motions. However, in most systems, measurements of the target position are reported in polar coordinates (range, bearing, elevation angle) with respect to the target position. The measurement errors have considerable affects on the performance of the target tracking system. There are two implementation alternatives to remedy this problem. The first method is to use a linear Kalman filter with measurements converted to a Cartesian frame of reference. Here the errors of the converted measurements are correlated. The second method is to use an extended Kalman filter (EKF) that uses a mixed coordinate filter by incorporating the original measurements in a nonlinear fashion into target position estimation, [45].

The classical conversion from polar to cartesian coordinates results in biased and inconsistent estimates for cross-range measurement error. The main idea for unbiased conversion for 2D and the derivations for the Gaussian noise case is given in Section 4.2.2, then continued for the 3D case in Section 4.2.3.

4.2. Target Tracking Data Processing Architecture

In this section, the process and observation models for target tracking are defined. Then the foundations of the distributed data fusion architecture are presented.

4.2.1. Process Model

The target process is a four dimensional vector that consists of the two dimensional position of the target, (ξ, η), and the velocity of the target, $(\dot{\xi}, \dot{\eta})$, at each of these dimensions. The target process state vector is defined by

$$\mathbf{x} = [\xi \, \eta \, \dot{\xi} \, \dot{\eta}]^T, \tag{4.1}$$

and it evolves in time according to

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{v}(k) \tag{4.2}$$

in [30]. Here, $\mathbf{x}(k)$ is the real target state vector at time k as given in (4.1), $\mathbf{F}$ is the process transition matrix, and $\mathbf{v}$ is the process transition noise.

4.2.2. Observation Model in 2D

Sensors can only observe the first two dimensions of the process. The velocity of the target is not observable by the sensors. Furthermore, sensors collect range and bearing data, but they cannot observe the coordinates of the target directly. Because sensors observe the target state in polar coordinates, linear filtering formulations do not help as discussed in Section 4.1.

The measured range and bearing are defined with respect to the true range r and bearing β as

$$\begin{aligned} r_m &= r + \nu_r \\ \beta_m &= \beta + \nu_\beta \end{aligned} \tag{4.3}$$

in [46]. The errors in range ν_r and bearing ν_β are assumed to be independent with zero mean and standard deviations σ_r and σ_β, respectively.

The classical conversion from polar to cartesian coordinates is given by

$$\begin{aligned} x_m &= r_m \cos\beta_m \\ y_m &= r_m \sin\beta_m. \end{aligned} \tag{4.4}$$

Due to the nonlinear transformation of the noisy bearing, this conversion can give biased estimates. A compensation factor can be introduced based on the knowledge

about the bearing noise, ν_β. From the symmetric probability density function pdf of $f(\nu)$, it is clear that

$$E\left[\sin \nu_\beta\right] = 0. \tag{4.5}$$

Then the expectation values become

$$\begin{aligned} E\left[x_m\right] &= \lambda_\beta r \cos\beta \\ E\left[y_m\right] &= \lambda_\beta r \sin\beta. \end{aligned} \tag{4.6}$$

where $\lambda_\beta = E\left[\cos \nu_\beta\right]$ is the bias compensation factor. The classical conversion 4.4 is biased if $\lambda_\beta \neq 1$. Then the unbiased conversion for uniform distribution can be given as:

$$\begin{aligned} x_m^u &= \lambda_\beta^{-1} r_m \cos\beta_m \\ y_m^u &= \lambda_\beta^{-1} r_m \sin\beta_m. \end{aligned} \tag{4.7}$$

The target mean state observed after the unbiased polar-to-Cartesian conversion in 2D is given by

$$\varphi^c = \begin{bmatrix} \xi_m^c \\ \eta_m^c \end{bmatrix} = \begin{bmatrix} \lambda_\beta^{-1} r_m \cos\beta_m \\ \lambda_\beta^{-1} r_m \sin\beta_m \end{bmatrix} \tag{4.8}$$

in [30].The elements of covariance and variance of the observation errors based on the observations r_m and β_m in 2D are, [46]

$$\begin{aligned} R_m^{11} &= var(x_m^u | r_m, \beta_m) \\ &= (\lambda_\beta^{-2} - 2) r_m^2 \cos^2\beta_m + \frac{1}{2}(r_m^2 + \sigma_r^2)(1 + \lambda_\beta^{'} \cos 2\beta_m) \\ R_m^{22} &= var(y_m^u | r_m, \beta_m) \\ &= (\lambda_\beta^{-2} - 2) r_m^2 \sin^2\beta_m + \frac{1}{2}(r_m^2 + \sigma_r^2)(1 - \lambda_\beta^{'} \cos 2\beta_m) \\ R_m^{12} &= cov(x_m^u, y_m^u | r_m, \beta_m) \end{aligned}$$

$$= \quad (\lambda_\beta^{-2} - 2) r_m^2 \cos\beta_m \sin\beta_m + \frac{1}{2}(r_m^2 + \sigma_r^2)\lambda_\beta^{'} \sin 2\beta_m \tag{4.9}$$

where

$$\begin{aligned} \lambda_\beta &= E\left[\cos\nu_\beta\right] = e^{-\sigma_\beta^2/2}, \\ \lambda_\beta^{'} &= E\left[\cos 2\nu_\beta\right] = e^{-2\sigma_\beta^2} = \lambda_\beta^4 \end{aligned} \tag{4.10}$$

These factors are derived from the distribution of the bearing (and elevation) noise, [46]. The bearing measurement error has the zero-mean Gaussian distribution, which is denoted as $\nu_\beta \sim N(0, \sigma_\beta)$.

In [46], the elements of the true covariance and variance R_t have also been derived.

$$\begin{aligned} R_m^{11} &= var(x_m^u | r, \beta) \\ R_m^{22} &= var(y_m^u | r, \beta) \\ R_m^{12} &= cov(x_m^u, y_m^u | r, \beta) \end{aligned} \tag{4.11}$$

However,the true covariance is unavailable due to the fact that the true range and bearing must be known. Thus covariance values conditioned on the measured range r_m and measured bearing β_m have been derived. In [47], covariance conditioned on r_m and β_m have been derived by conditioning the covariance R_t on r_m and β_m, $R_m = E\left[R_t | r_m, \beta_m\right]$. In [46] a direct procedure is followed by deriving x_m^u and y_m^u based on the r_m and β_m.

4.2.3. Observation Model in 3D

The observations for 3D case are, [46]:

$$\begin{aligned} r_m &= r + \nu_r \\ \beta_m &= \beta + \nu_\beta \end{aligned}$$

$$\varepsilon_m = \varepsilon + \nu_\varepsilon \tag{4.12}$$

where r is the true range, β is the bearing angle, ε is the elevation angle and ν_r, ν_β, ν_ε are independent zero-mean noise with standard deviation σ_r, σ_β and σ_ε respectively. Then the unbiased conversions in this case:

$$\begin{aligned} x_m^u &= \lambda_\beta^{-1}\lambda_\varepsilon^{-1} r_m \cos\beta_m \cos\varepsilon_m \\ y_m^u &= \lambda_\beta^{-1}\lambda_\varepsilon^{-1} r_m \sin\beta_m \cos\varepsilon_m \end{aligned} \tag{4.13}$$

while the covariances of the observation errors in 3D are

$$\begin{aligned} R_m^{11} &= \frac{1}{4}\lambda_\beta^{-2}\varepsilon_\beta^{-2}(r_m^2 + 2\sigma_r^2)(1 + (\lambda_\beta^{'})^2\cos 2\beta_m) \times (1 + (\lambda_\varepsilon^{'})^2\cos 2\varepsilon_m) \\ &- \frac{1}{4}(r_m^2 + \sigma_r^2)(1 + \lambda_\beta^{'}\cos 2\beta_m(1 + \lambda_\varepsilon^{'}\cos 2\varepsilon_m) \\ R_m^{22} &= \frac{1}{4}\lambda_\beta^{-2}\varepsilon_\beta^{-2}(r_m^2 + 2\sigma_r^2)(1 - (\lambda_\beta^{'})^2\cos 2\beta_m) \times (1 + (\lambda_\varepsilon^{'})^2\cos 2\varepsilon_m) \\ &- \frac{1}{4}(r_m^2 + \sigma_r^2)(1 - \lambda_\beta^{'}\cos 2\beta_m(1 + \lambda_\varepsilon^{'}\cos 2\varepsilon_m) \\ R_m^{12} &= \frac{1}{4}\lambda_\beta^{-2}\varepsilon_\beta^{-2}(\lambda_\beta^{'})^{-2}(r_m^2 + 2\sigma_r^2)\sin 2\beta_m(1 + (\lambda_\varepsilon^{'})^2\cos 2\varepsilon_m) \\ &- \frac{1}{4}\lambda_\beta^{'}(r_m^2 + \sigma_r^2)\sin 2\beta_m(1 + \lambda_\varepsilon^{'}\cos 2\varepsilon_m) \end{aligned} \tag{4.14}$$

where

$$\begin{aligned} \lambda_\beta &= E\left[\cos\nu_\beta\right] = e^{-\sigma_\beta^2/2}, \\ \lambda_\beta^{'} &= E\left[\cos 2\nu_\beta\right] = e^{-2\sigma_\beta^2} = \lambda_\beta^4 \\ \lambda_\varepsilon &= E\left[\cos\nu_\varepsilon\right] = e^{-\sigma_\varepsilon^2/2}, \\ \lambda_\varepsilon^{'} &= E\left[\cos 2\nu_\varepsilon\right] = e^{-2\sigma_\varepsilon^2} = \lambda_\varepsilon^4 \end{aligned} \tag{4.15}$$

It should be noted that the covariance matrix for 3D is a three-by-three matrix, consisting of elements R_m^{11}, R_m^{22}, R_m^{33}, R_m^{12}, R_m^{13}, R_m^{23}. However, in this study the only the coordinates (x, y) are of importance, not the z coordinate. The target tracking process continues in 2D, since the z coordinate is bound to the terrain height in the

corresponding (x, y) coordinate and need not be observed. We can imagine a 2D mapping of the 3D terrain, where (x, y) coordinates do not change, only the height values are eliminated. The parameters observed $(r_m, \beta_m, \varepsilon_\beta)$ are mapped to 2D coordinates. As a result, we only need the R_m^{11}, R_m^{22}, R_m^{12} elements of the covariance matrix.

4.2.4. Distributed Data Fusion Architecture

Information state $\mathbf{y}$ and the information matrix $\mathbf{Y}$ associated with an observation estimate $\hat{\mathbf{x}}$, and the covariance of the observation estimate $\mathbf{P}$ at time instant k are given by [48]

$$\hat{\mathbf{y}}(k) = \mathbf{P}^{-1}(k)\hat{\mathbf{x}}(k), \tag{4.16}$$

$$\mathbf{Y}(k) = \mathbf{P}^{-1}(k). \tag{4.17}$$

In [48], it is shown that by means of sufficient statistics, an observation φ contributes $\mathbf{i}(k)$ to the information state $\mathbf{y}$ and $\mathbf{I}(k)$ to the information matrix $\mathbf{Y}$ where

$$\mathbf{i}(k) = \mathbf{H}^T\mathbf{R}^{-1}(k)\varphi(k), \tag{4.18}$$

$$\mathbf{I}(k) = \mathbf{H}^T\mathbf{R}^{-1}(k)\mathbf{H} \tag{4.19}$$

and $\mathbf{H}$ is the observation matrix of the sensor.

Instead of sharing the measurements related to the target state among the collaborating sensors, sharing the information form of the observations results in a simple additive fusion framework that can be run on each of the tiny sensing devices. The distributed data fusion equations are

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{i}_i(k), \tag{4.20}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \sum_{i=1}^{N} \mathbf{I}_i(k) \tag{4.21}$$

in [30]. Here, N is the total number of sensors participating in the fusion process and $\hat{\mathbf{y}}(k \mid k-1)$ represents the information state estimate at time k given the observations up to and including time $k-1$.

Just before the data at time k are collected, if we are given the observations up to the time $k-1$, the predicted information state and the information matrix at time k can be calculated from, [30]

$$\hat{\mathbf{y}}(k \mid k-1) = \mathbf{Y}(k \mid k-1)\mathbf{F}\mathbf{Y}^{-1}(k-1 \mid k-1)\hat{\mathbf{y}}(k-1 \mid k-1), \tag{4.22}$$

$$\mathbf{Y}(k \mid k-1) = [\mathbf{F}\mathbf{Y}^{-1}(k-1 \mid k-1)\mathbf{F}^T + \mathbf{Q}]^{-1} \tag{4.23}$$

where $\mathbf{Q}$ is the state transition covariance.

State estimate of the target at any time k can be found from, [30]

$$\hat{\mathbf{x}}(k \mid k) = \mathbf{Y}^{-1}(k \mid k)\hat{\mathbf{y}}(k \mid k). \tag{4.24}$$

4.2.5. Collaborative Target Tracking Algorithm

The algorithm employed by a sensor for tracking targets in a collaborative manner within the distributed data fusion framework is depicted in Algorithm 4, [30] and explained as a flowchart in Figure 4.1. The information state and the information matrix denominations associated with the current observation are defined by (4.18). The predicted information state and the information matrix are computed by (4.22).

$$\hat{\mathbf{y}}(k \mid k) = \hat{\mathbf{y}}(k \mid k-1) + \mathbf{i}(k), \tag{4.25}$$

$$\mathbf{Y}(k \mid k) = \mathbf{Y}(k \mid k-1) + \mathbf{I}(k). \tag{4.26}$$

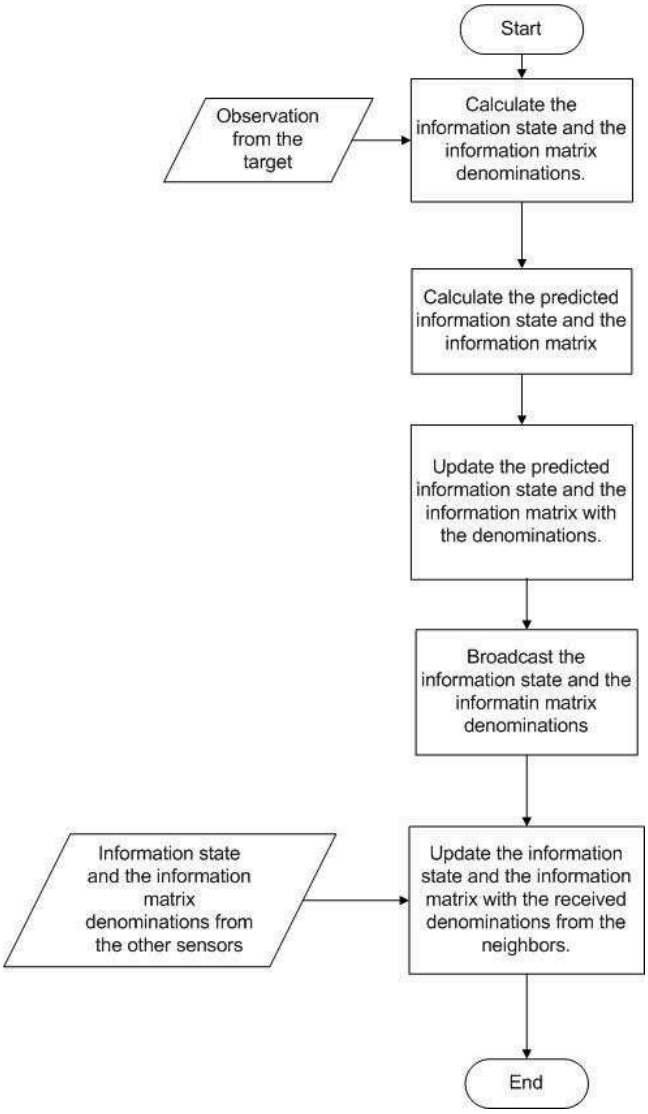

Figure 4.1. Target Tracking algorithm employed by each sensor, modified from the ICTP adjustment algorithm

4.3. Sample 3D Surfaces

3D terrains, where the results are evaluated, are generated via Perlin Noise method in GeoFrac2000, with the parameters in Table 4.3. The rendered form of terrains are in shown in Figures 4.2 through 4.9. Each of the terrain has more than one views, rendered with different camera position and orientations. Terrains are rendered with Terragen.

Algorithm 4 Pseudocode of the collaborative target tracking algorithm run on a sensor. Modified from [30].

Require: *Observations* $r_m(k)$, $\theta_m(k)$ *from the target at time instant k, information state* $\boldsymbol{i}_i(k)$ *and information matrix* $\boldsymbol{I}_i(k)$ *at time instant k from the neighboring sensors i, system variables* $\boldsymbol{F}$, $\boldsymbol{Q}$, $\boldsymbol{H}$, $\boldsymbol{R}$.

Ensure: *Target position estimate* $\hat{\boldsymbol{x}}(k \mid k)$ *at time instant k.*

1: $\varphi(k) \Leftarrow f(r_m, \theta_m)$

2: $\boldsymbol{i}(k) \Leftarrow \boldsymbol{H}^T \boldsymbol{R}^{-1}(k)\varphi(k)$

3: $\boldsymbol{I}(k) \Leftarrow \boldsymbol{H}^T \boldsymbol{R}^{-1}(k)\boldsymbol{H}$

4: $\hat{\boldsymbol{y}}(k \mid k-1) \Leftarrow \boldsymbol{Y}(k \mid k-1)\boldsymbol{F}\boldsymbol{Y}^{-1}(k-1 \mid k-1)\hat{\boldsymbol{y}}(k-1 \mid k-1)$

5: $\boldsymbol{Y}(k \mid k-1) \Leftarrow [\boldsymbol{F}\boldsymbol{Y}^{-1}(k-1 \mid k-1)\boldsymbol{F}^T + \boldsymbol{Q}]^{-1}$

6: $\hat{\boldsymbol{y}}(k \mid k) \Leftarrow \hat{\boldsymbol{y}}(k \mid k-1) + \boldsymbol{i}(k)$

7: $\boldsymbol{Y}(k \mid k) \Leftarrow \boldsymbol{Y}(k \mid k-1) + \boldsymbol{I}(k)$

8: $\hat{\boldsymbol{y}}(k \mid k) = \hat{\boldsymbol{y}}(k \mid k-1) + \sum_{i=1}^{N} \boldsymbol{i}_i(k)$

9: $\boldsymbol{Y}(k \mid k) = \boldsymbol{Y}(k \mid k-1) + \sum_{i=1}^{N} \boldsymbol{I}_i(k)$

10: $\hat{\boldsymbol{x}}(k \mid k) = \boldsymbol{Y}^{-1}(k \mid k)\hat{\boldsymbol{y}}(k \mid k)$

Table 4.1. Perlin Noise Parameters

Terrain Number	Harmonics	Frequency	Amplitude
Terrain 1	6	0.15	50
Terrain 2	6	1	5
Terrain 3	6	0.3	20
Terrain 4	7	0.1	20

As seen figures, Terrain 1 and Terrain 3 are hilly terrains which have peaks, although Terrain 1 is of more mountainous type in the sense that the surface in Terrain 1 is flatter than Terrain 3, which also means Terrain 3 has a more ledgy surface than Terrain 1. The reason for this, is that Terrain 3 is generated with a higher frequency for the same harmonics values as seen in Table 4.3. Higher values of amplitude results in a mountainous surface in Terrain 1. Terrain 2 is not a hilly (slopy) terrain but has small-sized ledges. The reason behind that is also explained in Section 3.5. Although generated with a higher frequency, Terrain 2 is not hilly because of its small vertical

scale. High frequency, that would otherwise result in peaks, has caused small ledges in Terrain 2 when the vertical scale is relatively small. Terrain 4 is not ledgy-surfaced, but relatively slopy in the sense that it has more wide-scale slopes than Terrain 2 but has less number of ledges which makes it flatter. The reason behind this is clear when Table 4.3 is referenced, since Terrain 4 is generated with the smallest frequency and a relatively smaller amplitude. The outcome of the topography of the terrains in terms of communication and detection capability of the sensor nodes is explained in Sections 5.4 and 5.5.

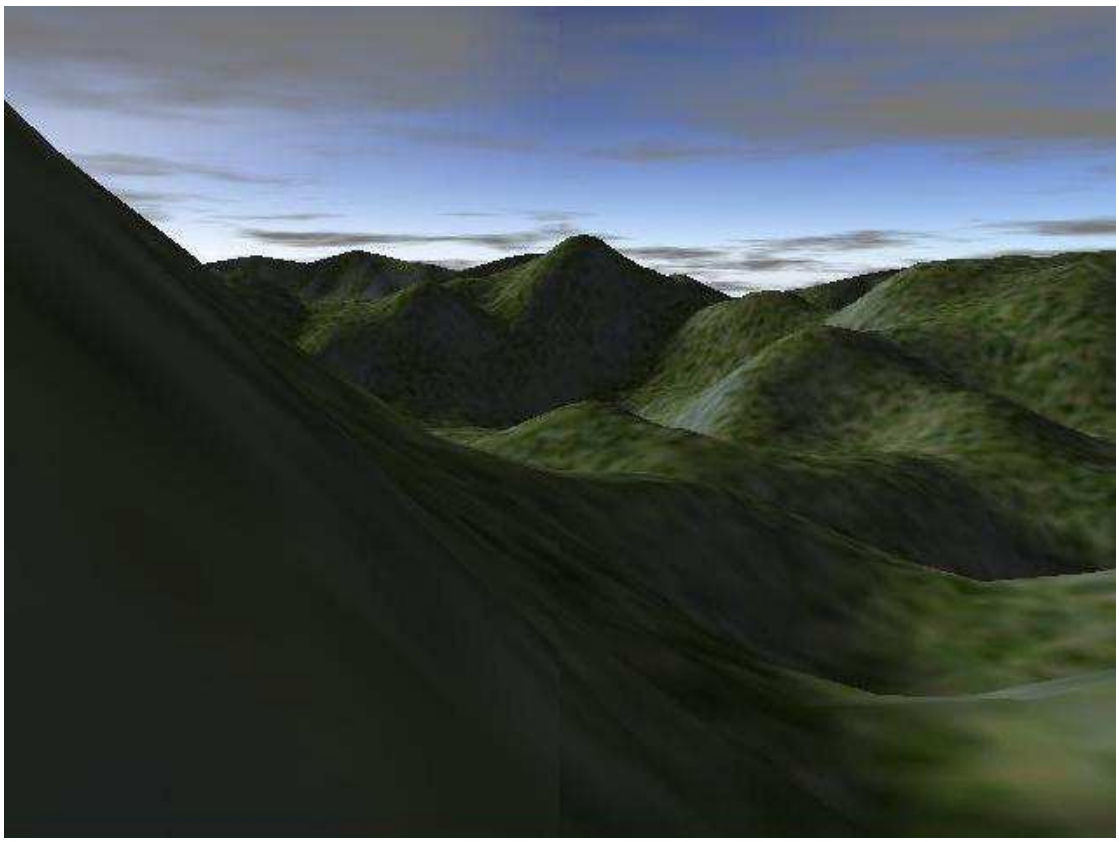

Figure 4.2. Terrain 1, part (a)

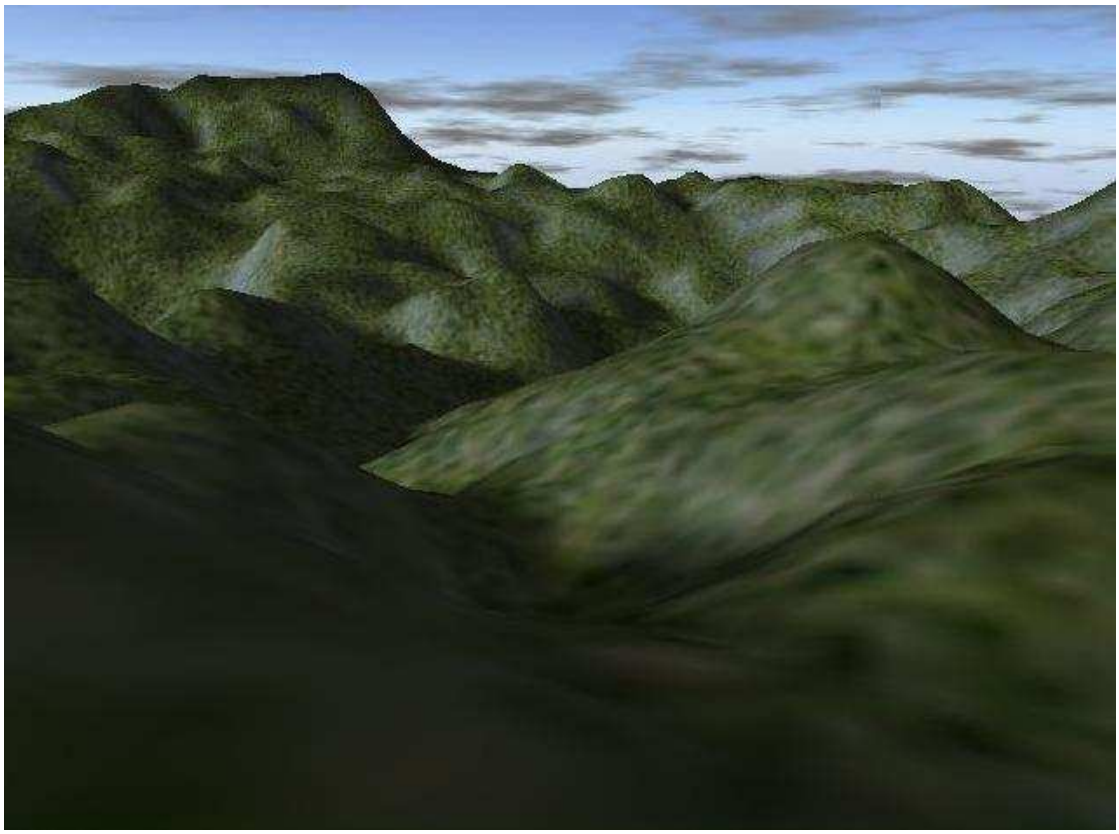

Figure 4.3. Terrain 1, part (b)

Figure 4.4. Terrain 2, part (a)

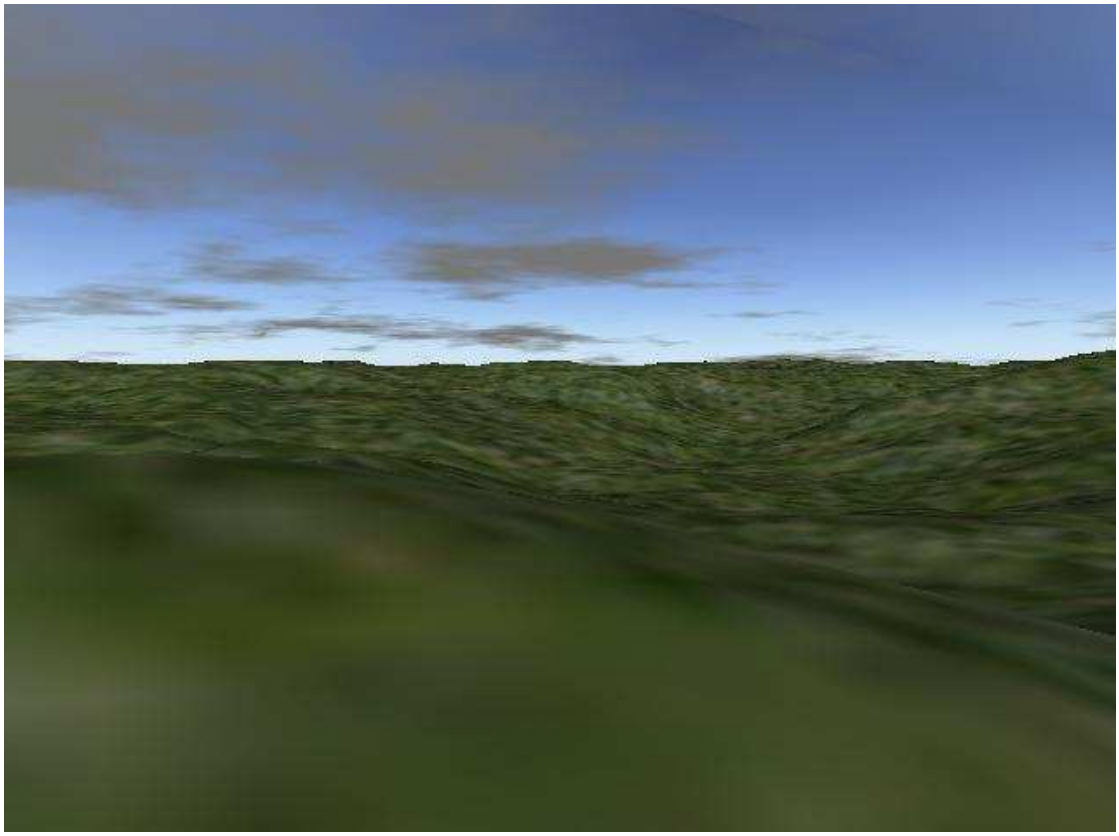

Figure 4.5. Terrain 2, part (b)

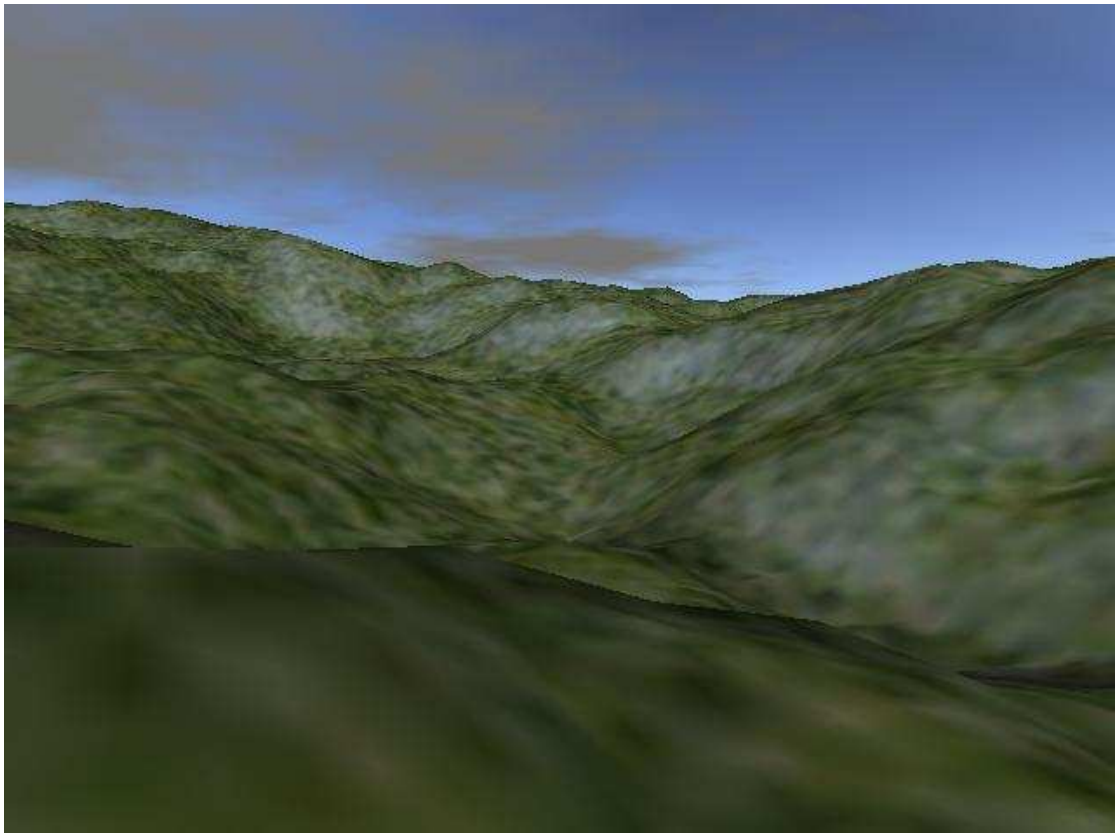

Figure 4.6. Terrain 3, part (a)

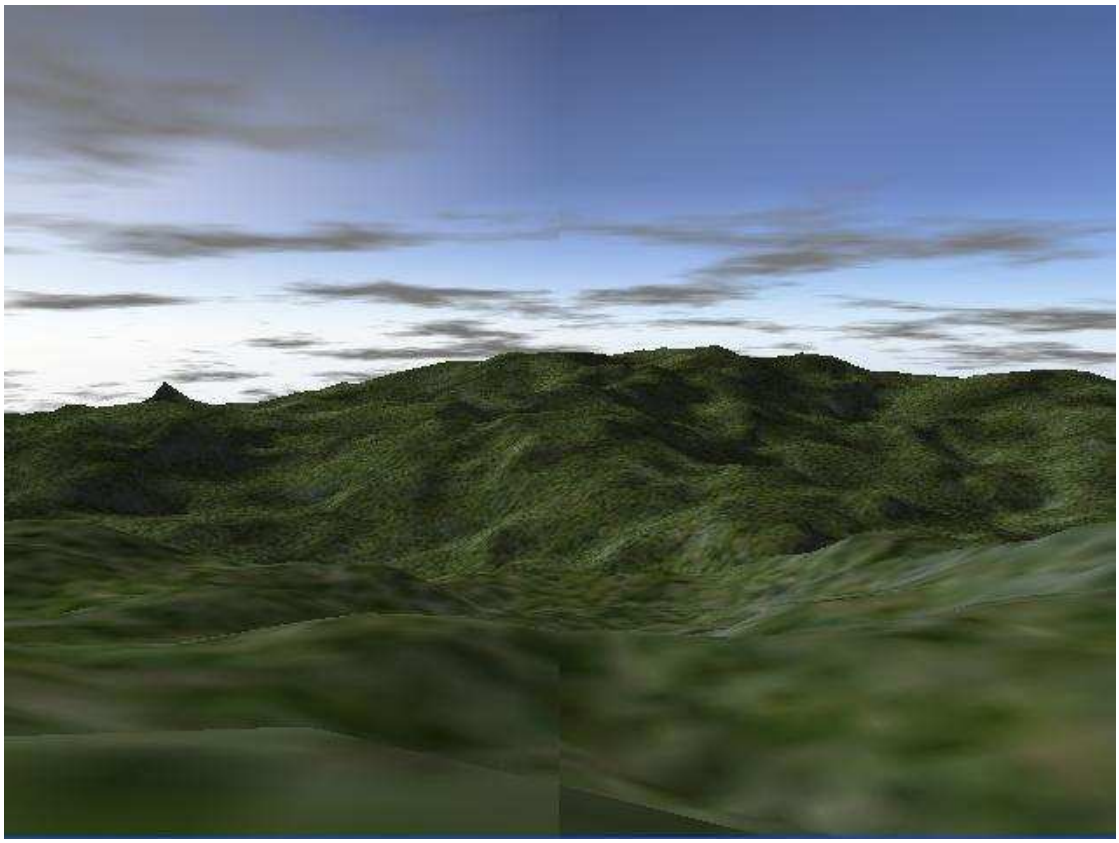

Figure 4.7. Terrain 3, part (b)

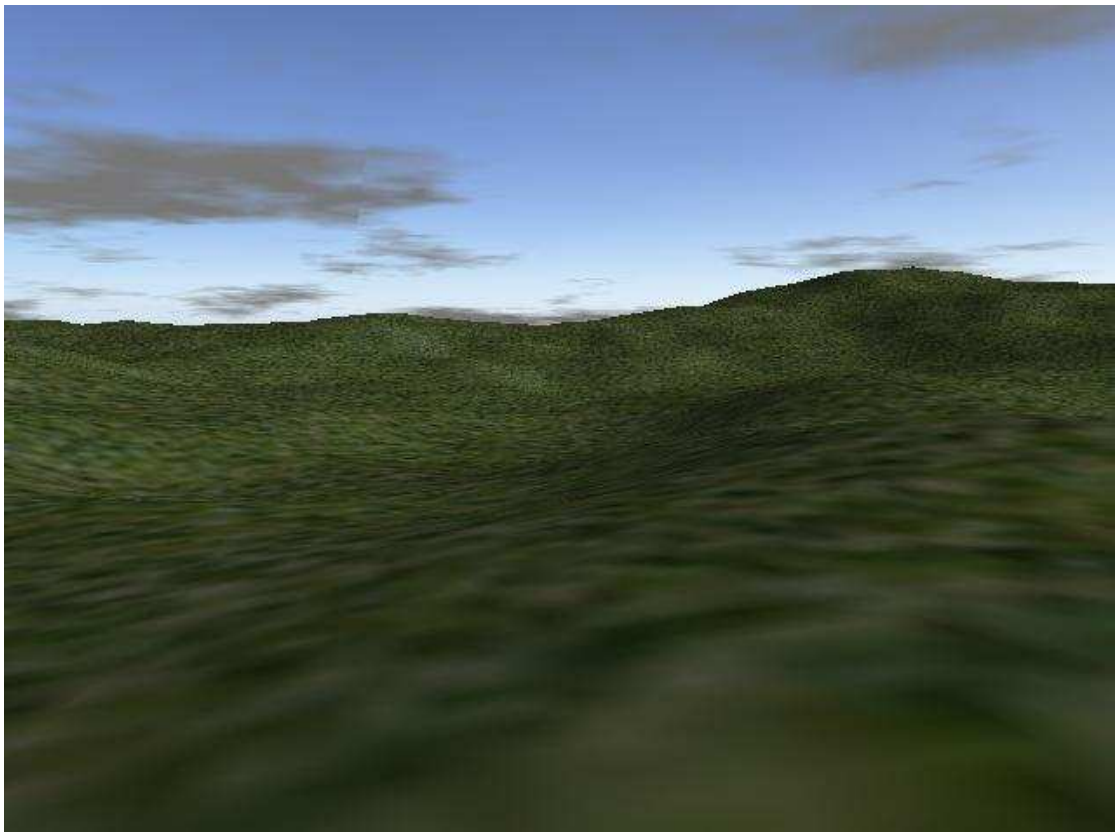

Figure 4.8. Terrain 4, part (a)

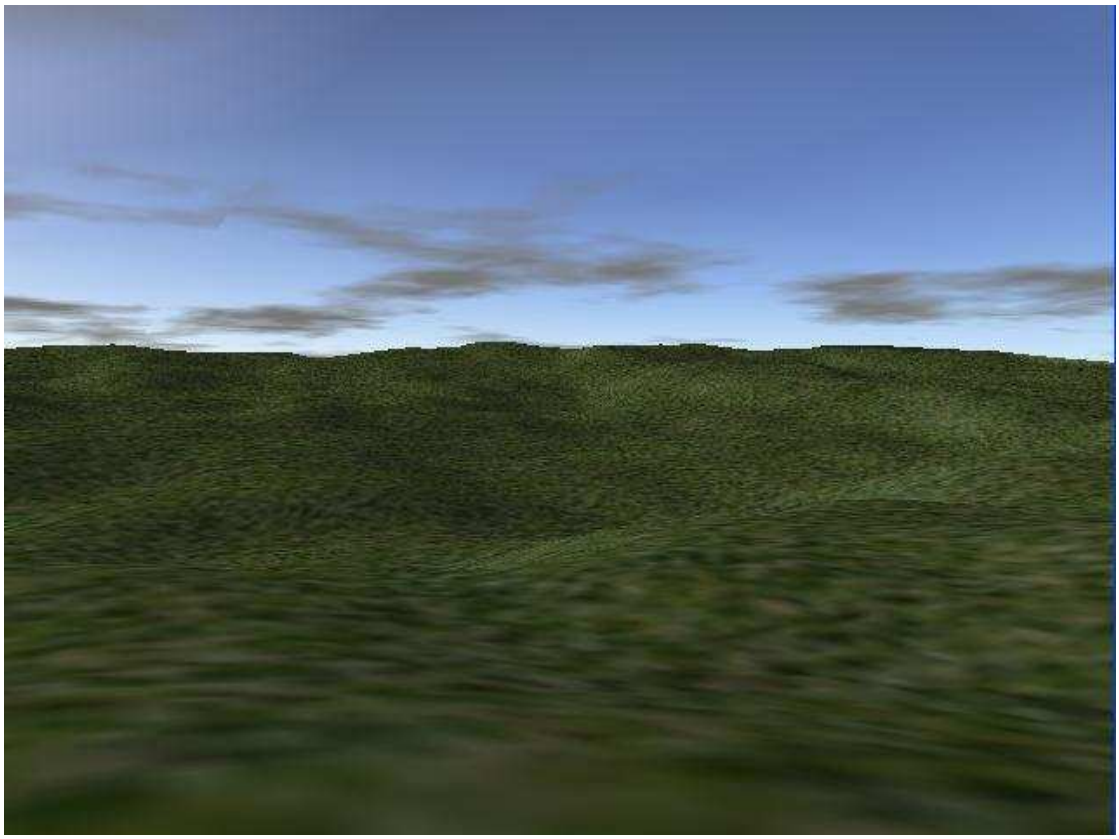

Figure 4.9. Terrain 4, part (b)

5. 3D TOPOGRAPHIC PERFORMANCE EVALUATION OF WSN

The aim of this chapter is to analyze the effect of incorporating 3D terrains into a WSN target tracking simulation environment. Following the information on the simulation environment and experiments, we comment on the simulation results, with regards to the topography of the terrains and other simulation parameters.

5.1. Simulation Environment & Methodology

We have done our simulations with the heightmaps of the terrains shown in Section 4.3. The heightmaps are processed with the techniques explained in Section 3.2. The heightmap processing is done in terms of two WSN tasks: communication and sensing.

For each terrain, ten different sensor deployments are done. The results in Sections 5.3, 5.5 and 5.6 are derived from the same target motion behaviour. Associated with each deployment in a terrain, the communication matrix and detection array is derived. The communication matrix defines how many sensor pairs out of all sensor pairs could communicate in a channel via broadcast, while the detection array defines how many sensors could detect the target at each time slot of the complete simulation duration. The communication matrix is the output of Visual C++ program, since it runs faster than the Matlab code, while the detection array is the output of a Matlab program. The simulation program runs in Matlab, outputs the *real target motion*, *observation error*, *cooperative information filter error*, *total packets received*, *total packets sent* in the network, *total detecting sensors* and *total sending sensors*.

Table 5.1. Shadow fading communication model parameters.

Carrier frequency	1.8 GHz
Path loss exponent	2
TX & RX antenna height	0.1 m
Shadow fading standard deviation	4
Sensor transmission power	-30 dB

5.2. Experiments

We run Monte Carlo simulations to examine the performance of the WSN target tracking application based on the distributed data fusion architecture, taking the methodology and the work in [30]. We examine the mean error in four terrains and with four different scenarios each having different number of sensors. Each of these 16 simulations are run with ten random deployments in a 200 m $\times$ 200 m area. All data points in the graphs represent the means of ten runs. A target moves in the area according to the process model described in Section 4.2.3. We utilize TWR-ISM-002-I radar [49] detection model with pseudorandom signaling, whose typical range is 18 meters, [30].

In collaborative information fusion, if a sensor is able to communicate with other sensors to share its belief about the target state, it broadcasts its information state and the information matrix. Communication may not be possible due to the non-LOS condition between nodes in terrain. Sensors update their belief about the target state with these received data as described in Section 4.2.4. In our simulations, we use the same shadow-fading radio propagation model for the communication signals as in [30], whose parameters are given in Table 5.1, [50].

5.3. Analysis of the Mean Error for Collaborative Target Tracking

In Figures 5.1 and 5.2, the mean error comparison of cooperative information filter has been done for various terrain types and the 2D surface. It is seen that Terrain 2 has the highest mean error, although it has relatively less slopes than Terrain 1 and

Terrain 3. However, Terrain 2 has a lot of ledges on its surface, which means, although the peaks are small in size, they are enough to block the communication among sensors. Terrain 1 and Terrain 3 have similar mean errors, since they have both hilly surfaces. With being more hilly Terrain 1 has a higher mean error. Among the evaluations in terrains, Terrain 4 has the lowest error rate, since it does not have a hilly surface nor any ledges. In Figure 5.2, the error comparison of Terrain 4 and the 2D surface has been shown. Even in the most dense scenario and in comparison with the flattest terrain, there is a high deviation between the mean errors; the error in the 2D scenario is 0.34 m whereas it is 3.46 m in Terrain 4. In other terrains, the mean error for the 300 sensors scenario is approximately 6.1 m, which makes the targeting application impractical.

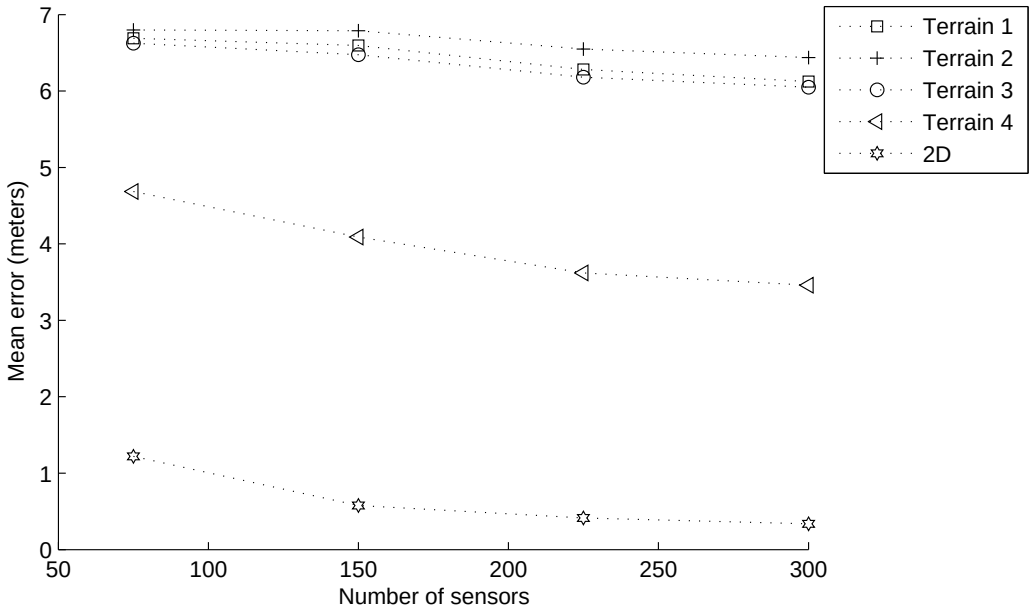

Figure 5.1. Mean error comparison for cooperative information filter

Figure 5.3, 5.4 and 5.5 compares the observation errors between 2D and respectively Terrain 1, Terrain 2, Terrain 4, from the viewpoint of the sensor that is marked as a star at the coordinates $(191, 141)$. The circles indicate other sensors in the field. As the target moves away, the observation errors for the sensor increase. The results for Terrain 3 is not given here, since it gives similar results with Terrain 1. The observation errors are calculated according to 2D and 3D distances, they do not consider whether the target is detected or not. Observation errors give similar but not the same

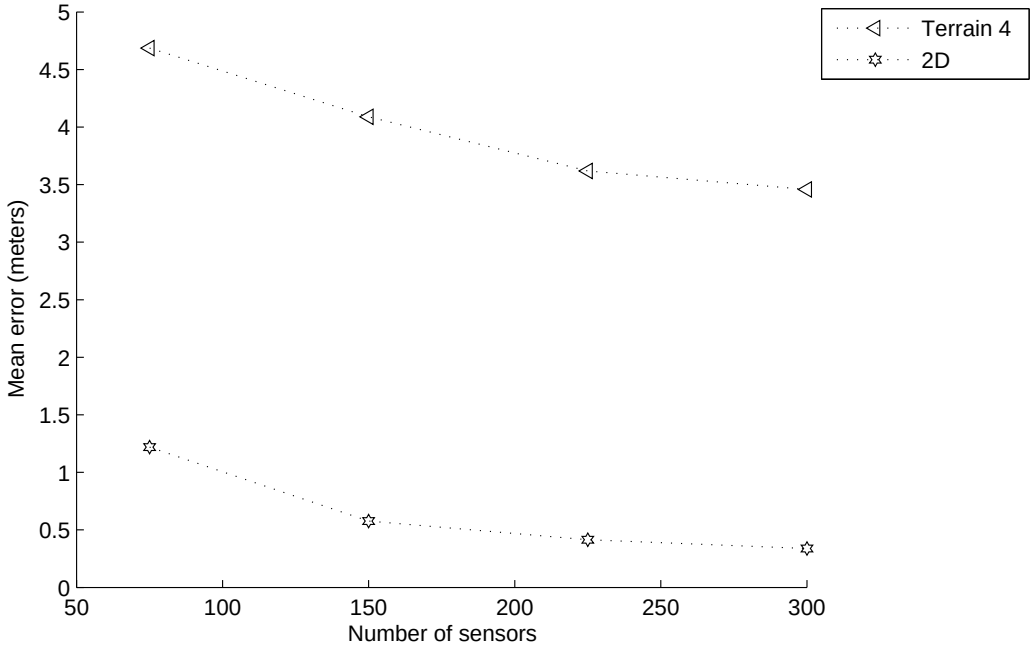

Figure 5.2. Mean error comparison of Terrain 4 and 2D surface for cooperative information filter

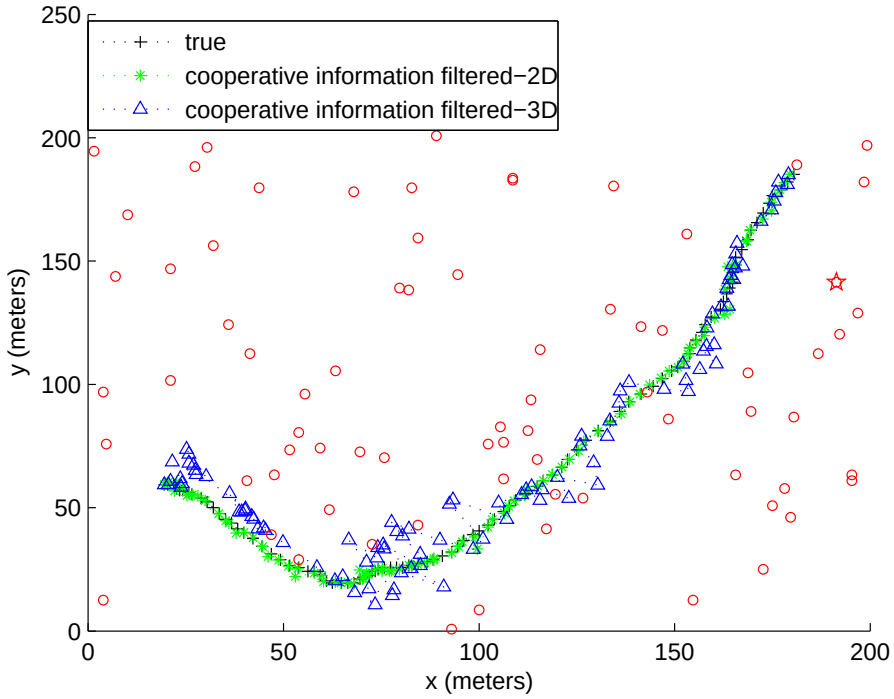

Figure 5.3. 2D and 3D observation errors of a sensor in Terrain 1, in 75 sensor scenario

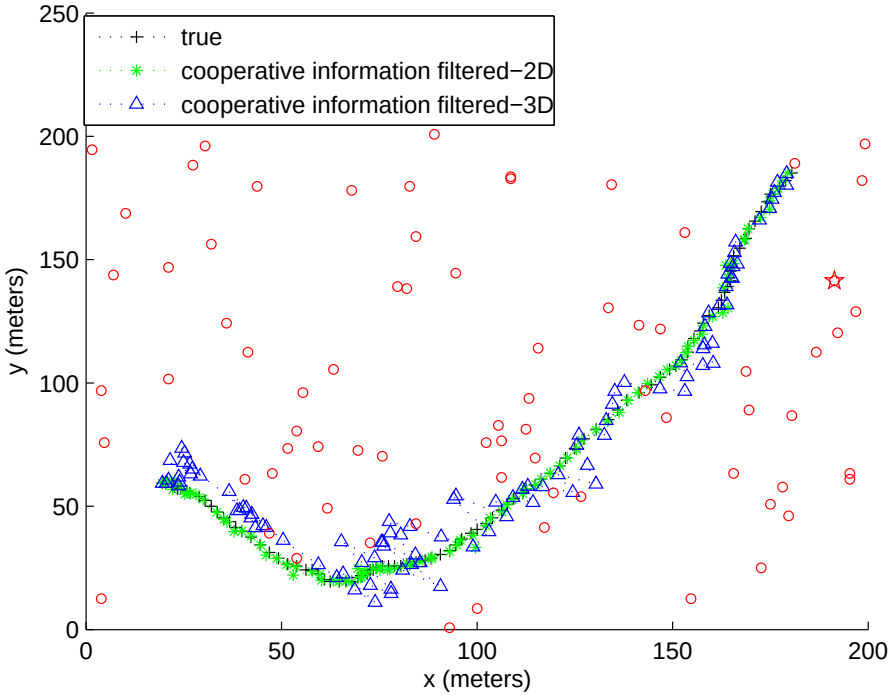

Figure 5.4. 2D and 3D observation errors of a sensor in Terrain 2, in 75 sensor scenario

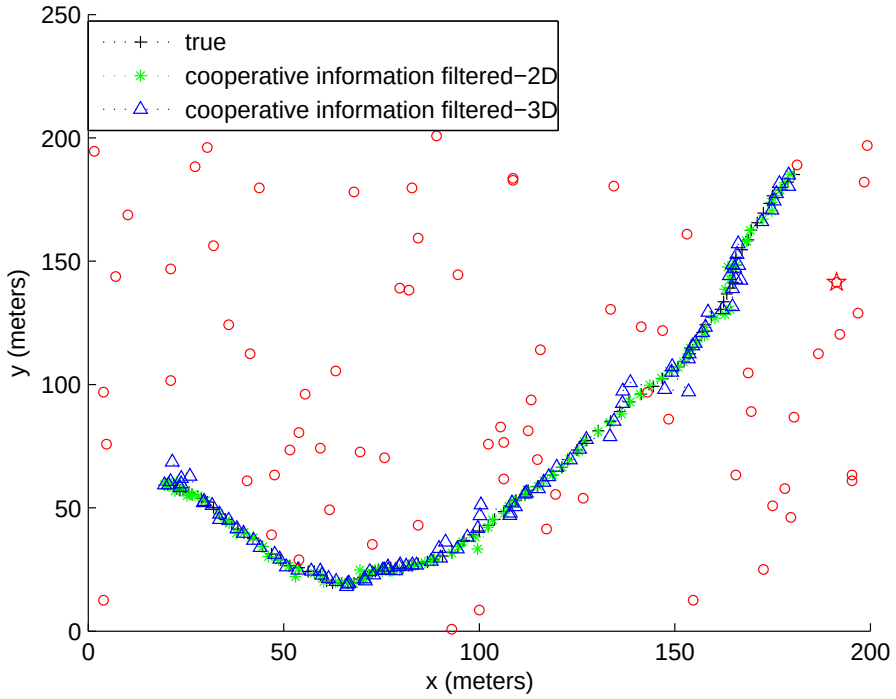

Figure 5.5. 2D and 3D observation errors of a sensor in Terrain 4, in 75 sensor scenario

results for the three of the terrains, since each terrain has a different topology, and therefore has different distance, bearing degree and elevation degree parameters.

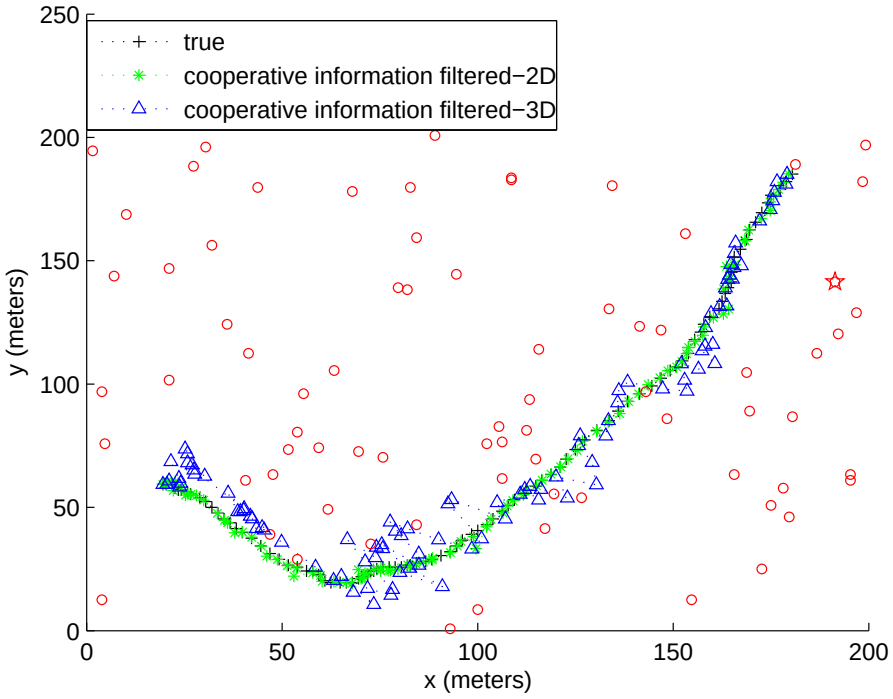

Figure 5.6. 2D and 3D cooperative information errors of a sensor in Terrain 1, in 75 sensor scenario

The cooperative information errors of Terrain 1, Terrain 2 and Terrain 4 have been compared with that of 2D in Figures 5.6, 5.7 and 5.8. Here, again the error in Terrain 3 is not illustrated, since each three terrains (Terrain 1, Terrain 2 and Terrain 3) have similar mean errors. The results are evaluated according to the viewpoint of the sensor marked as a star. As seen also in the mean error graph (Figure 5.1), mean errors of Terrain 1 and Terrain 2 are very close to each other, resulting in similar target location estimates. On the other hand Terrain 4 achieves a better mean error, and the target estimate is close the original state and cooperative information filtered estimate.

The differences in the target location estimates are more clear in Figure 5.9, which focuses on a more refined segment of the target's route.

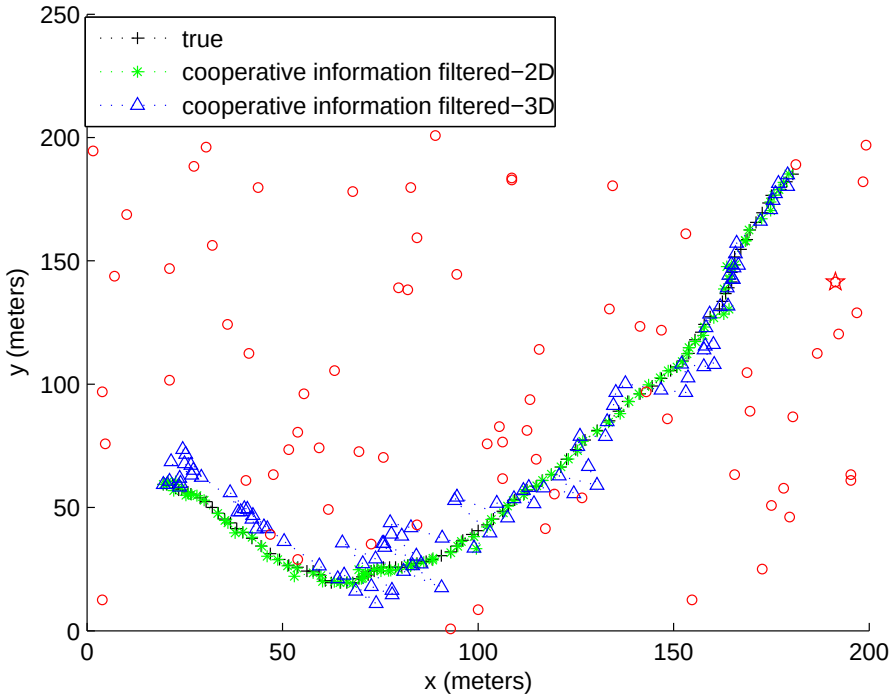

Figure 5.7. 2D and 3D cooperative information errors of a sensor in Terrain 2, in 75 sensor scenario

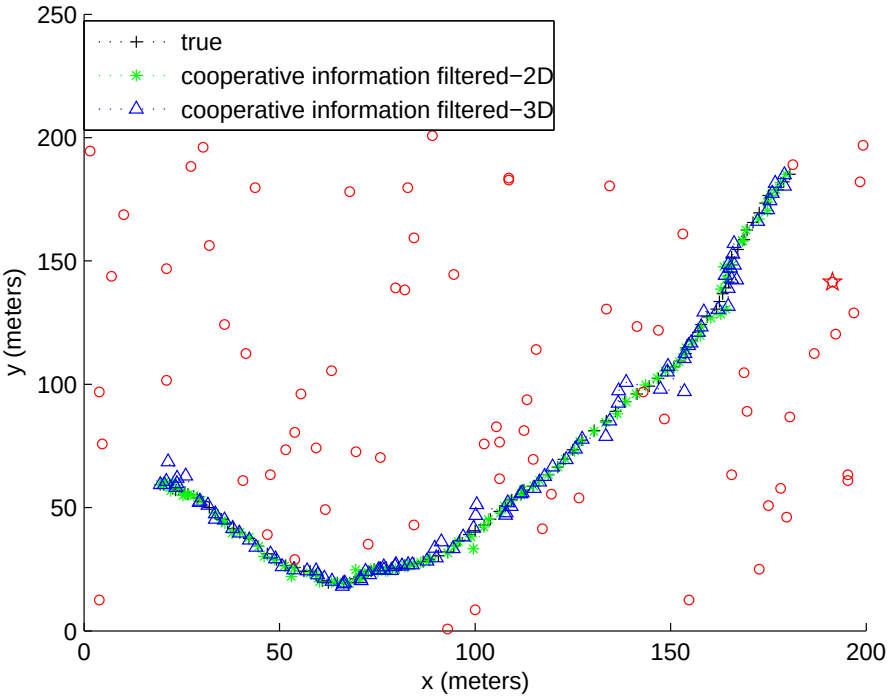

Figure 5.8. 2D and 3D cooperative information errors of a sensor in Terrain 4, in 75 sensor scenario

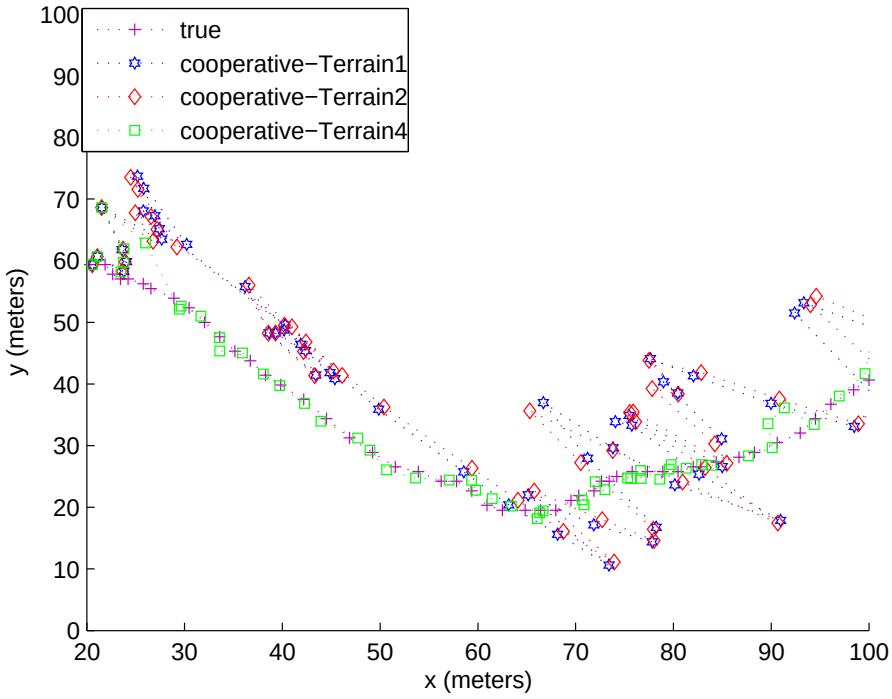

Figure 5.9. Cooperative information errors of Terrain 1, Terrain 2 and Terrain 4

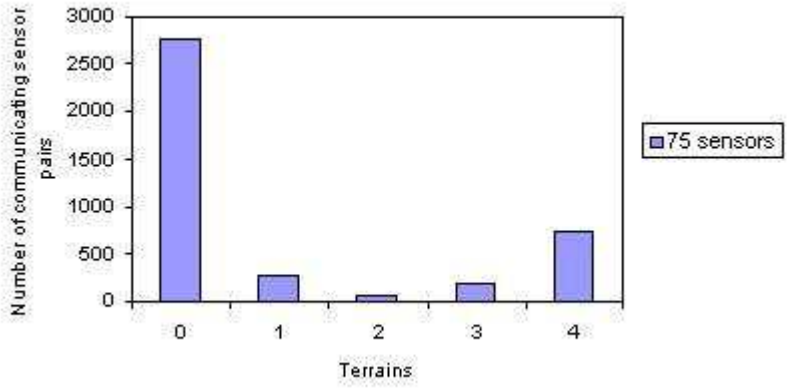

Figure 5.10. Comparison of sensor communication rate in 3D terrain (75 sensors)

5.4. Analysis of the Communication Rate Among Sensors

In this section the communication among sensors in a terrain environment have been measured. Figures 5.10 through 5.13 show the results. The values are obtained from one of the upper or lower triangles of the communication matrix, as discussed in Section 5.1. The y-axis shows the number of communicating sensor pairs, while the x-axis shows the terrain types. Terrain 0 represents the 2D surface, while other

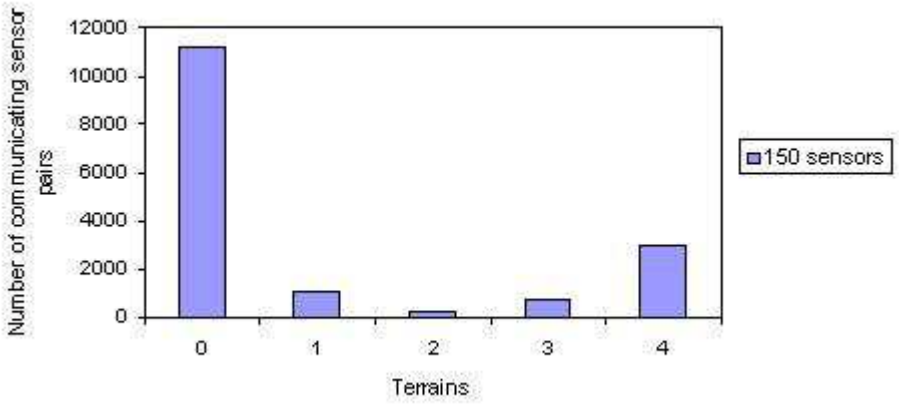

Figure 5.11. Comparison of sensor communication rate in 3D terrain (150 sensors)

terrains follow the previous indexing. The communication comparison has been done for all scenarios, 75 sensors, 150 sensors, 225 sensors and 300 sensors. All graphs have nearly the same results, therefore they will not be discussed separately.

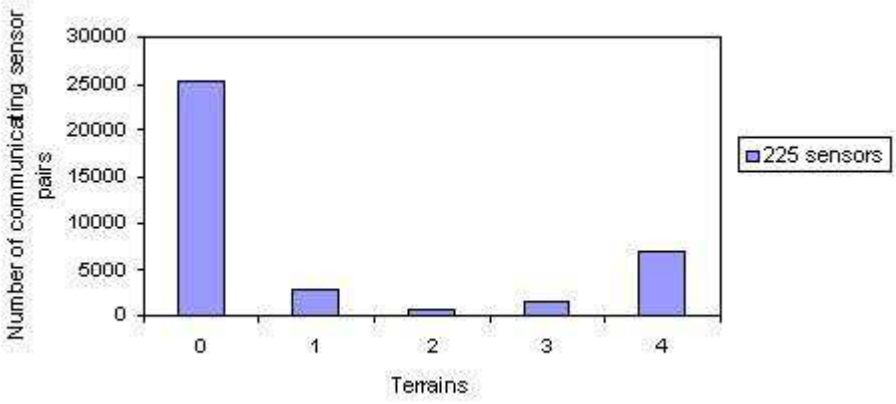

Figure 5.12. Comparison of sensor communication rate in 3D terrain (225 sensors)

It is clear that Terrain 4 has the highest number of communicating sensor pairs among other terrains, since it has a flatter surface than other terrains, in the sense that it has less slopes and less number of ledges. Even in this condition, it has one fourth communication capability of the 2D surface. Terrain 1 has the second highest number of communicating sensors among all the terrains. It should be noted that Terrain 1 is not the second best in the mean error comparison among terrains, Figure 5.1. This comes from the fact that the mean error is not a function of only the communication capability, but also the detection capability. Since the peaks in Terrain 1 occur in a

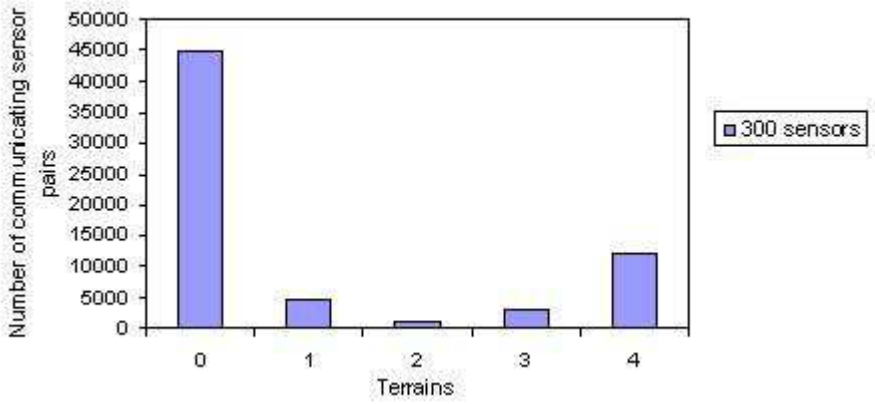

Figure 5.13. Comparison of sensor communication rate in 3D terrain (300 sensors)

large area, they do not block communication as ledges would normally do, just in the case of Terrain 2. Terrain 3, having a similar topography with Terrain 1 is the third best and Terrain 2 has the worst communication capability. This comes from the fact that, although Terrain 2 has no peaks, it has many ledges (small sized peaks, rocks, etc.) which block the communication between sensors.

5.5. Analysis of the Detection Rate of Sensors

In this section, the total target detection by sensors, on the 2D surface and in the 3D terrain environments have been measured. Figures 5.14 through 5.17 show the results. The values are the averaged values of the total detection in the simulation duration obtained from the detection array, as discussed in Section 5.1. The y-axis shows the number of detecting sensors at a time, while the x-axis shows the terrain types. The detection comparison has been done for all scenarios, 75 sensors, 150 sensors, 225 sensors, and 300 sensors. All graphs have the same relative results, therefore they will not be discussed separately.

It can be seen from charts that Terrain 4 having a smooth, not ledgy surface and less number of peaks, has the highest number of detecting sensors, coming after the 2D surface. It can be seen that, the 2D surface and Terrain 4 have similar results, unlike the results in the communication rate. It has two reasons. First the sensors have a sensing range of 18 meters, which means that even in the case of complete LOS, there

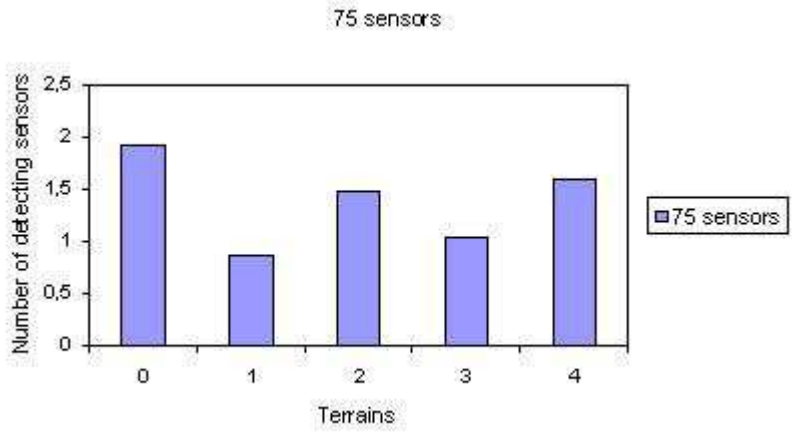

Figure 5.14. Comparison of sensor detection rate in 3D terrain (75 sensors)

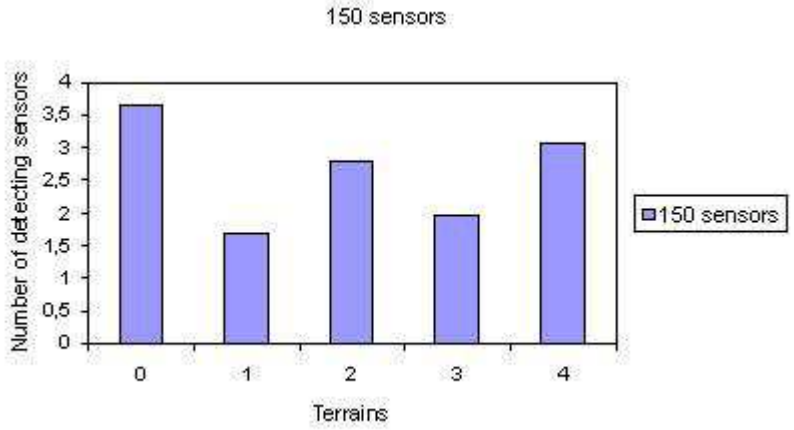

Figure 5.15. Comparison of sensor detection rate in 3D terrain (150 sensors)

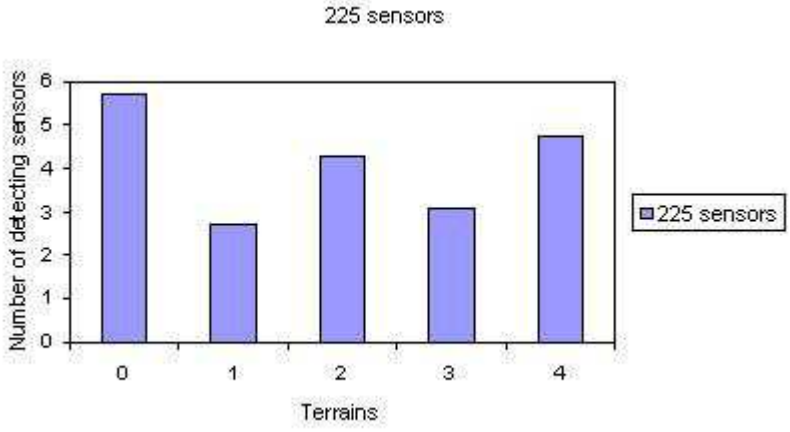

Figure 5.16. Comparison of sensor detection rate in 3D terrain (225 sensors)

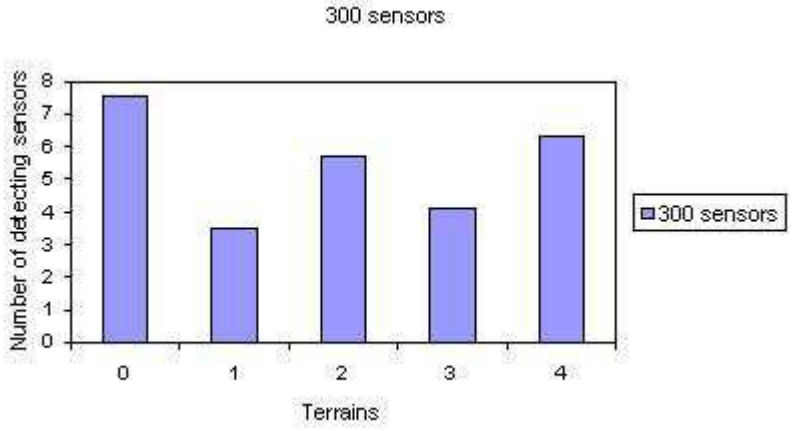

Figure 5.17. Comparison of sensor detection rate in 3D terrain (300 sensors)

will be a low detection rate due to this range. Second, the angle of the 3D line segment between the target and the sensor is higher than the angle of the line segment between sensors, and thus it is less likely to be obstructed by the terrain formations.

Terrain 2 is the second best in terms of detection, after Terrain 4. Although Terrain 2 has a small number of peaks (hills), it has a ledgy surface. It should be noted that ledges, which are sources of obstruction in the communication, do not affect detection as would the peaks do. Since the target and sensor height is assumed to be 150 cm and 10 cm respectively, the 3D line between the two, would not be blocked by the small ledges on the surface, contrary to the 3D line between a sensor pair in the communication task. Terrain 3 comes as the third best, since it has less number of peaks than Terrain 1, although it has a more ledgy surface.

Here, a discussion about the dominancy of the communication and detection, on mean error could be made. It is obvious that Terrain 1 and Terrain 3 give out similar results on communication and detection rates, with Terrain 1 giving out a better result for communication and Terrain 3 giving out better result for the detection rate. The communication and detection ratios of the Terrain 1 and Terrain 3 are $Cr_{t_3}^{t_1} = 1.53$ and $Dr_{t_1}^{t_3} = 1.17$, respectively. Although Terrain 1 has a higher communication ratio than Terrain 3's detection ratio, Terrain 3 gives out a better performance on the mean error. This implies that detection dominates in the mean error, when the topographies

give out similar results on detection and communication.

With regards to the topography of the terrains, we can also make such an assumption: A terrain having a hilly surface with a lot of peaks results in a low performance on detection, since the angle of the 3D line segment between the target and the sensor would be higher and be more likely to be obstructed by the large peaks, instead of ledges. On the other hand, a terrain having a ledgy surface results in a low performance on communication between the sensors, since the angle of the 3D line segment between sensor pairs would be smaller and be more likely to be obstructed by the peaks.

5.6. Analysis of Mean Error for Kalman and Individual Information Filters

Mean error comparisons for Kalman and individual Information filter has been depicted in Figures 5.18 and 5.19, respectively. Since both of these filters require no collaboration, they depend on the detection rate in the terrain environment, not the communication rate. The charts compare mean errors in each of the terrains and on the 2D surface, in the 75 sensor scenario. The mean error charts are almost parallel with the detection rate charts but in an inverse way, since as the detection rate increases, the mean error decreases.

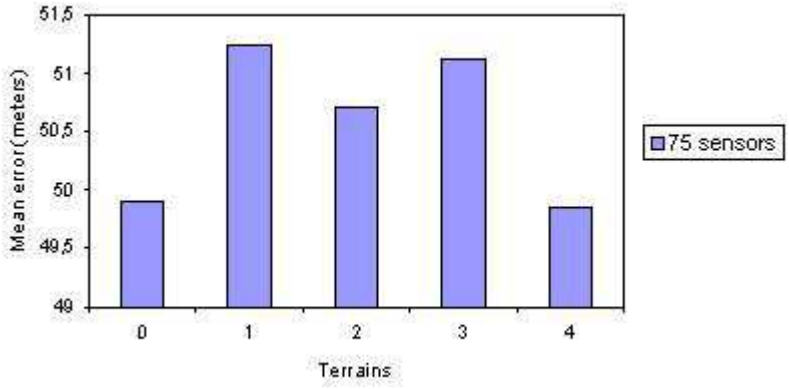

Figure 5.18. Mean error comparisons for Kalman filter

Here, an exception is the mean error on the 2D surface and in Terrain 4. It

can be seen that the 2D surface has slightly higher error rate than Terrain 4. This result comes from different covariance matrices in 2D and 3D, e.g., the elements in the covariance matrix of 2D has higher values than those in 3D.

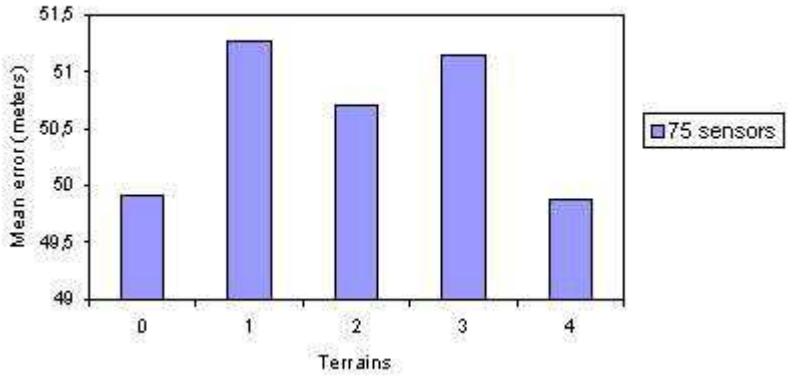

Figure 5.19. Mean error comparisons for Individual Information filter

The Kalman filter errors of Terrain 1, Terrain 2 and Terrain 4 have been compared with that of 2D in Figures 5.20, 5.21 and 5.22 from the viewpoint of the sensor marked as a star. The Kalman filter outputs of Terrain 1 and Terrain 2 do not output similar results this time, as opposed to the target location graphs in collaborative tracking (Figures 5.6 and 5.7). Since Terrain 2 has a higher detection rate, it achieves a better target location estimation. As also seen in Figure 5.21, the target location estimation in Terrain 2 is very close to that of the 2D surface. Among the three terrains, Terrain 4 achieves the best performance. It even gives a better estimation than that obtained by the 2D surface, which comes from the differences in the covariance matrix derivations in 2D and 3D, as discussed before.

5.7. Analysis of a Different Target Motion

In Figure 5.23, a comparison has been made in terms of detection rate of the sensors with a different target motion scenario, *motion 2*. Motion 2 can be seen in Figures 5.24 through 5.26. It is obvious that, although the target's new motion is wigglier than the previous one, it has a lower detection rate in each terrain, except in

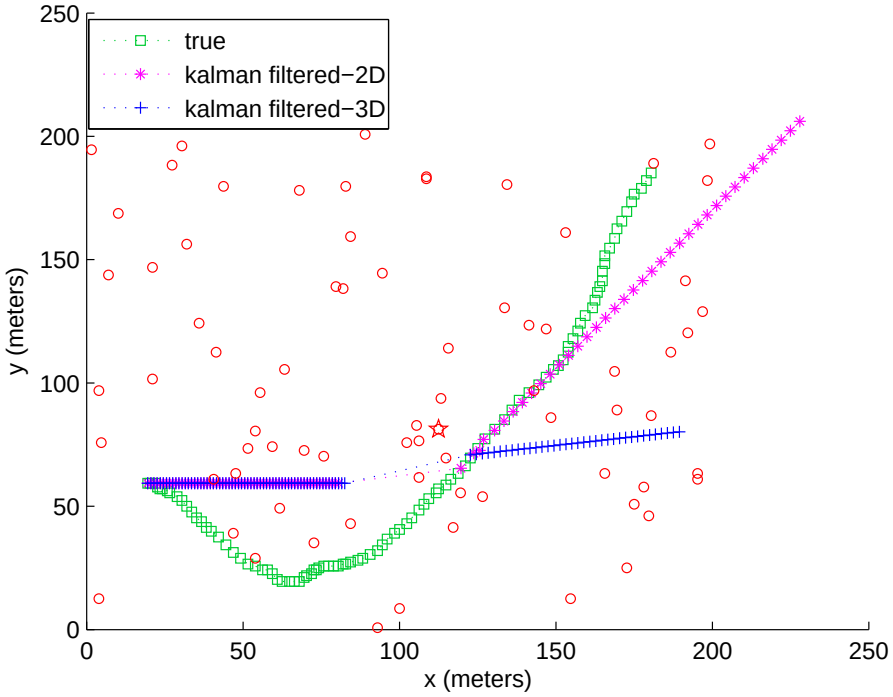

Figure 5.20. 2D and 3D Kalman filter errors of a sensor in Terrain 1, in 75 sensor scenario

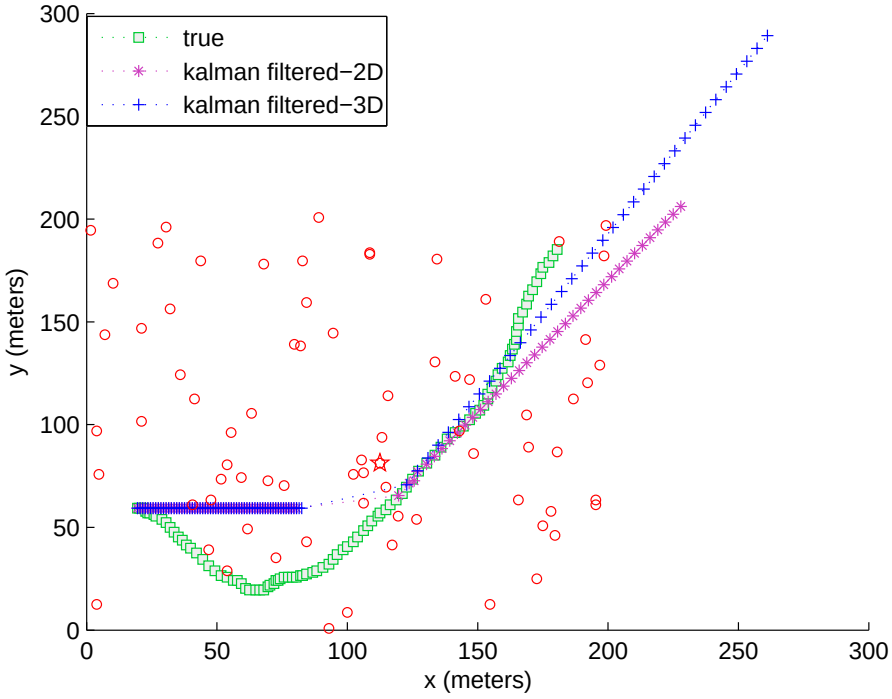

Figure 5.21. 2D and 3D Kalman filter errors of a sensor in Terrain 2, in 75 sensor scenario

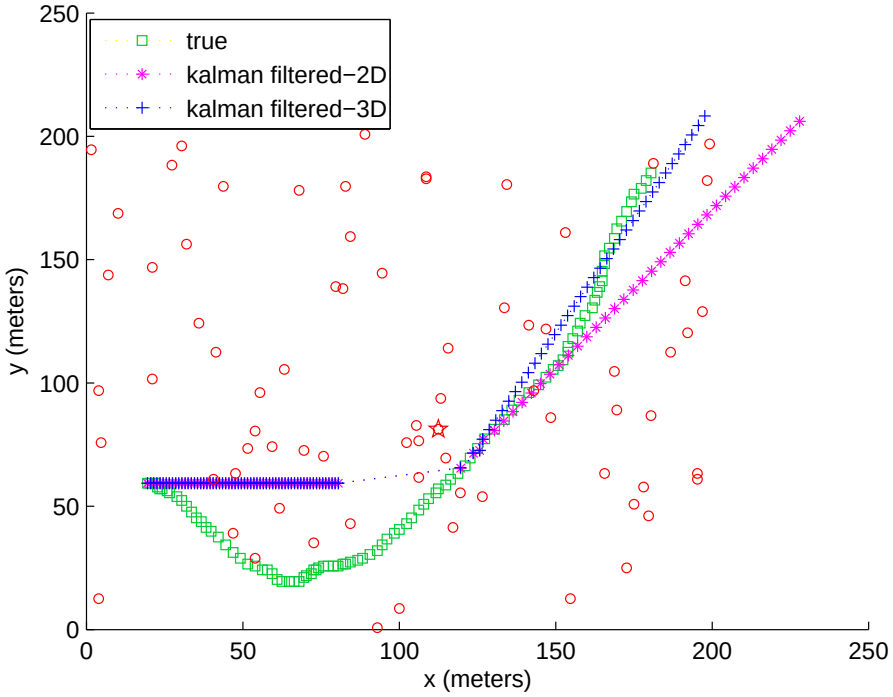

Figure 5.22. 2D and 3D Kalman filter errors of a sensor in Terrain 4, in 75 sensor scenario

Terrain 1. The reason behind this is the speed of the target. The target moves faster in this new motion, as seen from the distance between the markers in Figures 5.24 through 5.26. Since the LOS conditons vary continuously, the target's fast movement degrades the detection rate, and since the sensing range is 18 m the curly movement does not increase the performance.

Figures 5.24 through 5.26 show the performance of the collaborative information filter with this new target motion, in terms of the target's observed state from the point of the sensor marked as a star. Terrain 1 and Terrain 2 gives out similar results, and Terrain 4 achieves a better performance than both of the two terrains, as discussed in Section 5.3. It should be noted that, in the regions where there are not enough sensors, the observed state of the target deviates a lot from the real state in the case of Terrain 1 and Terrain 2. Terrain 4, with its smooth surface has a lower error rate, even in these regions.

This new target motion has been investigated with different sampling rates. Since

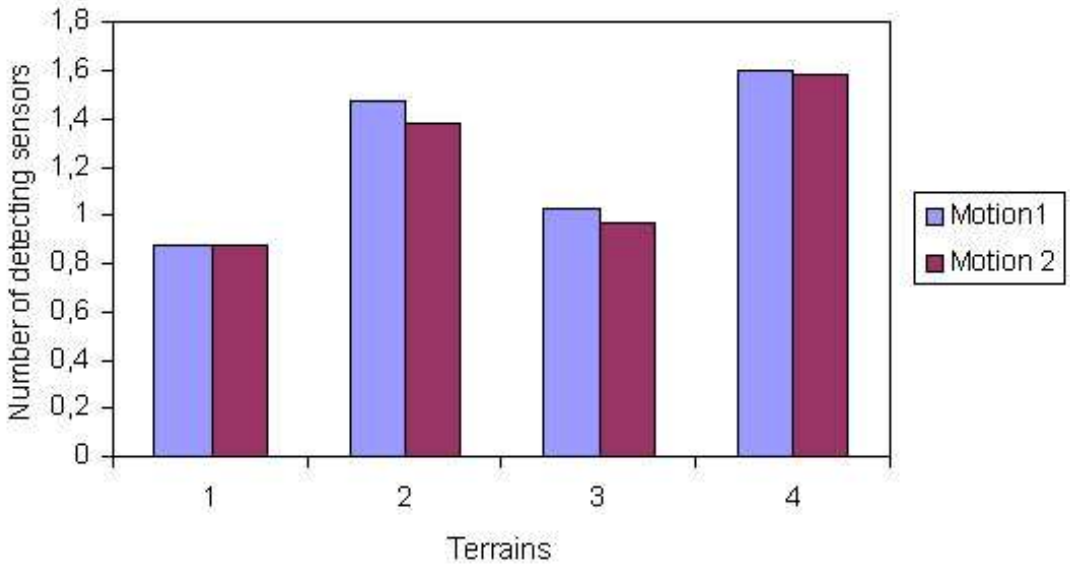

Figure 5.23. Comparison of target detection rate of motion 1 and motion 2 for 75 sensors scenario

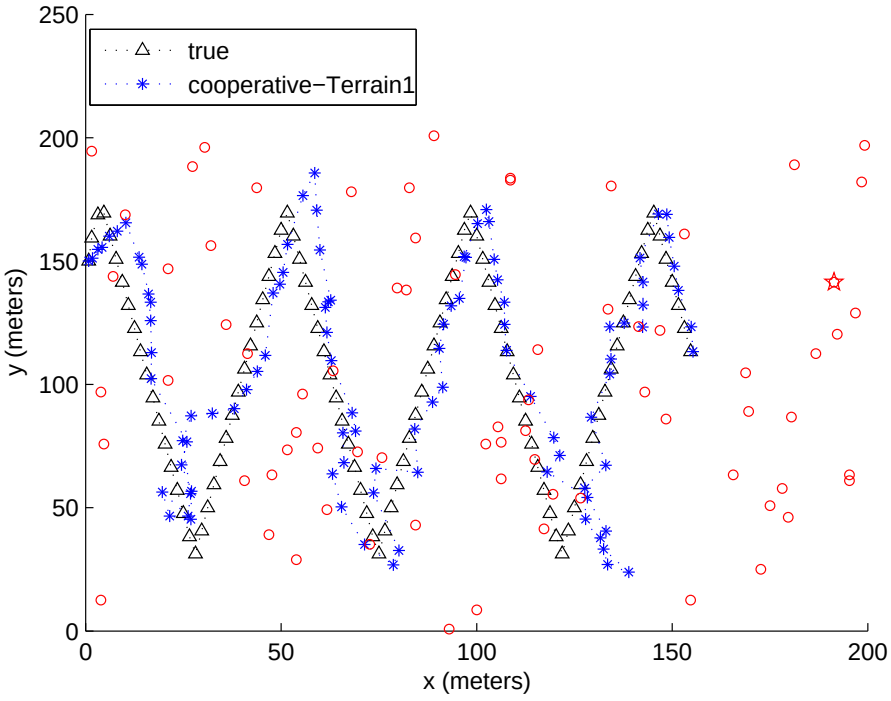

Figure 5.24. 2D and 3D Collaborative information filter errors of a sensor in Terrain 1, in 75 sensor scenario

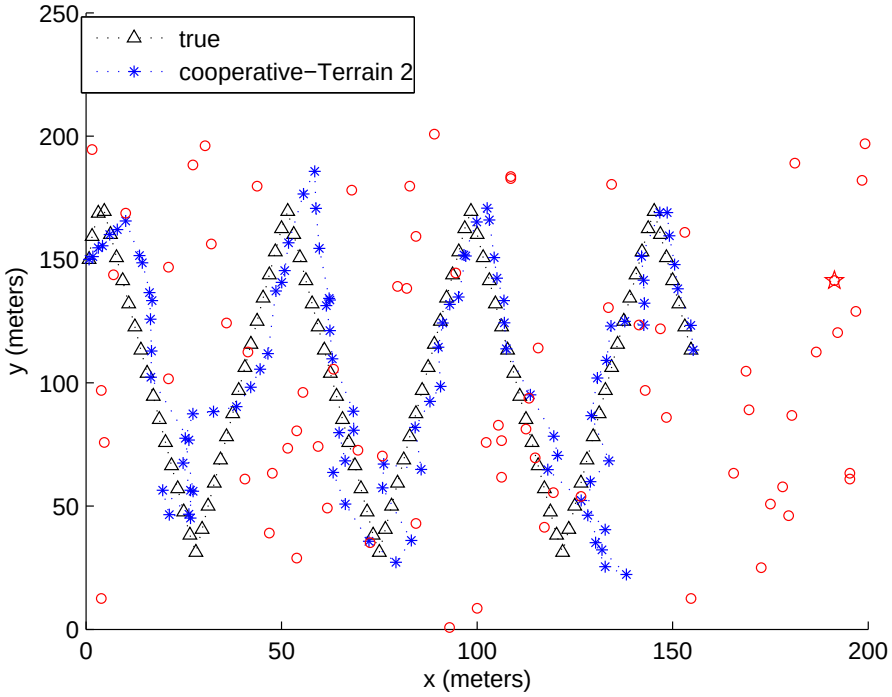

Figure 5.25. 2D and 3D Collaborative information filter errors of a sensor in Terrain 2. in 75 sensor scenario

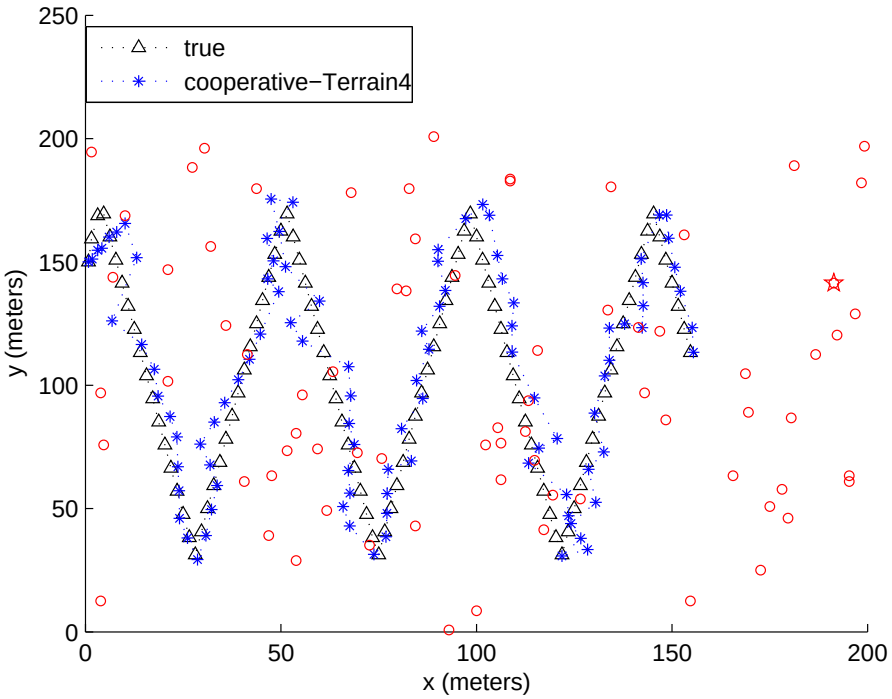

Figure 5.26. 2D and 3D Collaborative information filter errors of a sensor in Terrain 4, in 75 sensor scenario

the LOS conditions vary continuously the aim is to capture the target position as frequent as possible. The mean error of this new target motion has been compared with that of the previous target motion at sampling rates of 5, 2.5, 1.25 and 0.62 times the sampling rate of the previous motion respectively, denoted by line. It could be seen in Figure 5.27, that the mean error has been improved in the first scenario and degrades as the sampling rate is reduced.

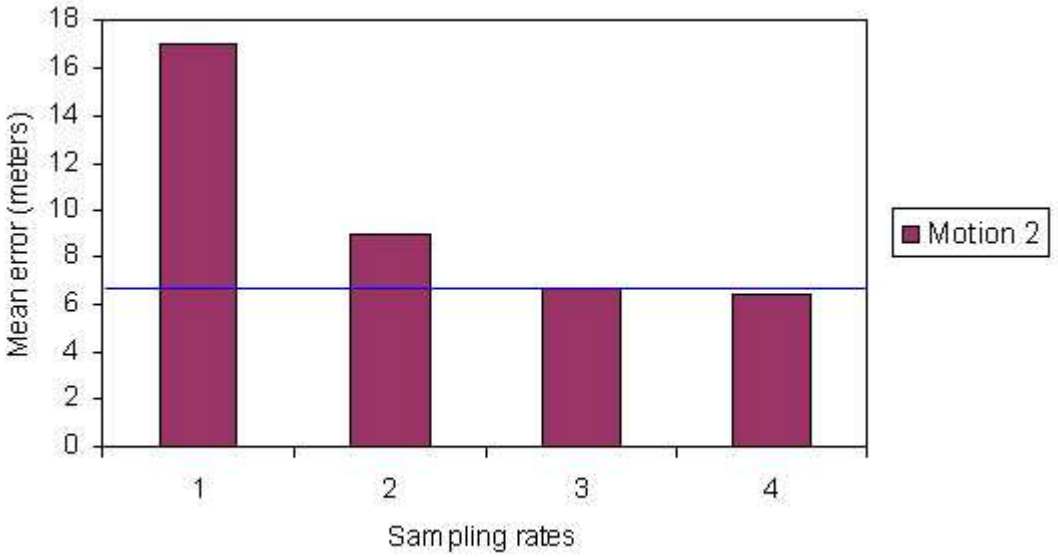

Figure 5.27. Mean error on different sampling rates, in 75 sensor scenario

5.8. Analysis of Terrain Generation Parameters

The mean error of the target tracking application has been analyzed in terms of different terrain generation parameters for the 75 sensors scenario. The effect of amplitude, frequency and harmonics of the Perlin Noise, can be seen in Figures 5.28, 5.29 and 5.30.

As seen in Figure 5.28, the mean error increases as the amplitude of the terrain is increased, at fixed harmonics and frequency values. The amplitude denotes the distance between the lowest and highest point in the terrain, namely the vertical scale. The behaviour of the mean error comes from the height of the peaks on the terrain surface. With a small amplitude value, the formations on the surface become ledges and with a high amplitude the formations become peaks. The amplitude factor both

affects communication and detection tasks, resulting in a degradation in the tracking accuracy.

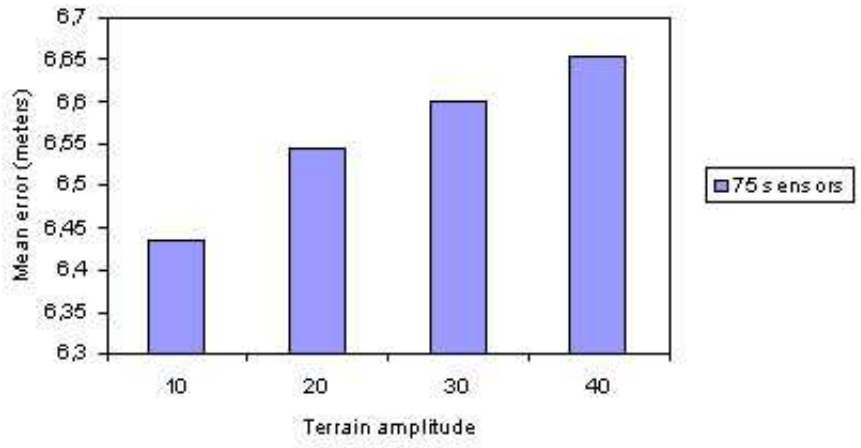

Figure 5.28. Effect of terrain amplitude on mean error, with *harmonics*=5 and *frequency*=0.3

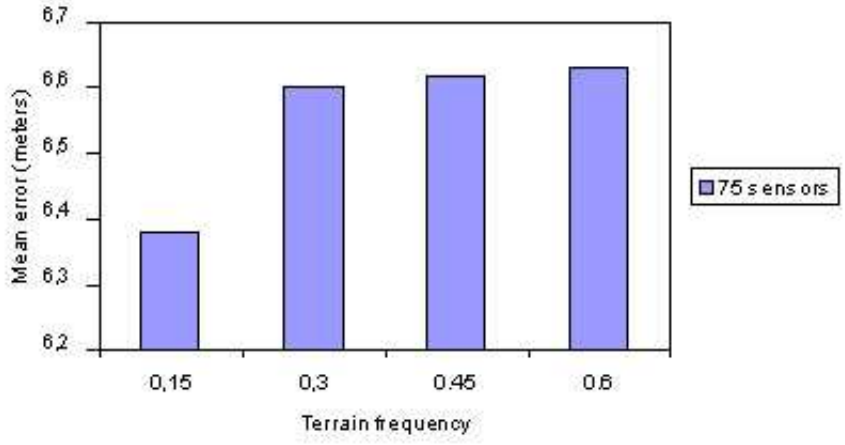

Figure 5.29. Effect of terrain frequency on mean error, with *harmonics*=5 and *amplitude*=20

Figure 5.29 depicts the effect of frequency on the mean error at fixed harmonics and amplitude values. As the frequency increases, the turbulence of the terrain increases resulting in a higher mean error. It should be noted that the degradation the performance is higher when the frequency is small. When the frequency increases, incremental mean error decreases. This comes from the different scale noise functions in Perlin Noise. Here, the frequency parameter denotes the highest frequency with the

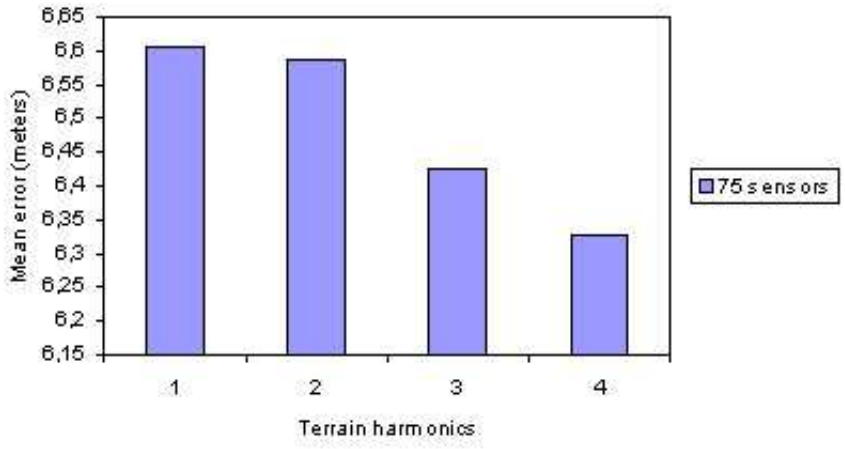

Figure 5.30. Effect of terrain harmonics on mean error, with *frequency*=0.12 and *amplitude*=10

lowest vertical scale in the terrain, namely the last noise wave with the smallest amplitude. Since at fixed amplitude and harmonics, the frequency of the last noise function would form ledges, the detection task would not be affected as would be effected in lower frequencies.

Figure 5.30 denotes the effect of number of harmonics on the mean error at fixed frequency and amplitude values. The harmonics is the number of noise functions used in the Perlin Noise. The frequency input to the terrain generation, is the frequency of the last noise function with the smallest vertical scale. As the number of harmonics increases, smaller frequencies are used in the terrain generation which means the terrain gets smoother, and eventually the mean error is decreased.

5.9. Analysis of Target Height

In this section, the effect of target height on the mean error is investigated in Terrain 1, Terrain 2 and Terrain 3 for the 75 sensors scenario. It can be seen in Figure 5.31 that the target tracking accuracy increases as the target height is increased. As discussed in previous sections, the targets could be either people or vehicles, e.g., cars, trucks, whose heights could range from 100 cm to 250 cm.

It should be noted that the accuracy is most improved in Terrain 3. This comes from the fact that Terrain 3 has a peaky surface, which blocks the LOS condition between the target and the sensor node. On the other hand, less improvement is achieved in Terrain 2 due to its ledgy surface, which does not block the LOS between the target and the sensor most of the time. Terrain 1 comes after Terrain 3 in terms of error improvement, due its relatively smooth surface.

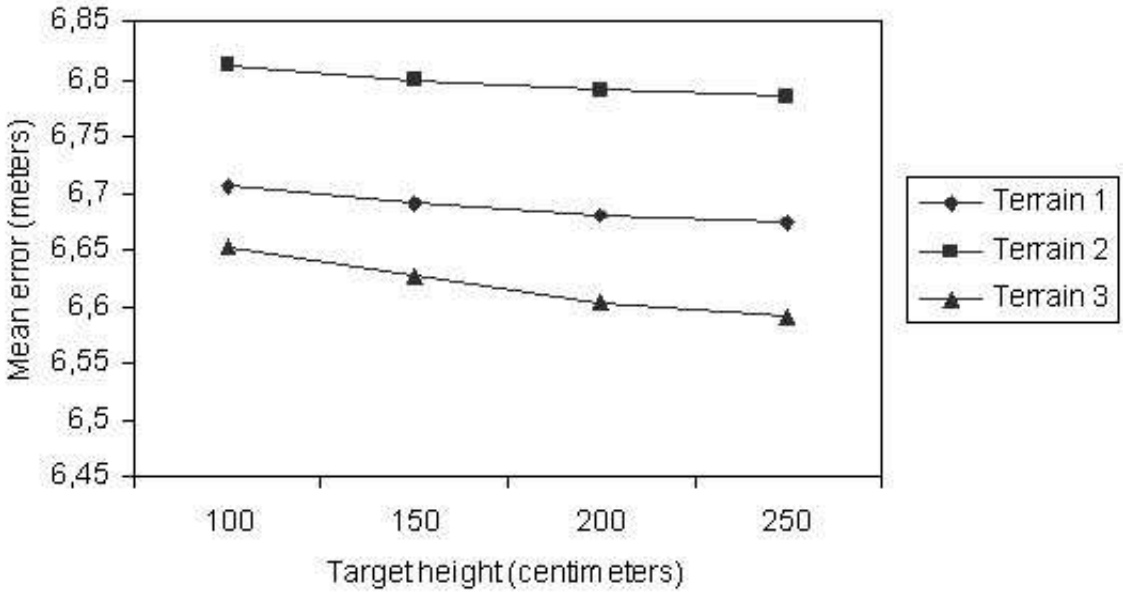

Figure 5.31. Effect of target height on mean error

6. CONCLUSION

In this thesis, we explored the problem of WSN performance evaluation on 3D topographic surfaces and made a performance comparison for a collaborative target tracking application scenario.

Most of the recent modeling and studies regarding a WSN research problem has been done on 2D planar surfaces assuming a distance based sensing and 2D freespace communication model. These works assume a random deployment scheme which commonly takes place in 3D inaccessible terrains, by randomly deploying thousands or sensors from a plane. A contradiction between the assumptions on the research problem and the modeling environment occurs as a result of this.

In this thesis we investigated the problem of incorporating a realistic modeling environment into sensor networks. Our motivation has been the non-realistic and contradictive assumptions in WSN performance evaluations, where the formations of the topographic surface such as 3D line of sight, rocks, ledges, that would normally block the communication and sensing tasks are not taken into account. In the real life, the outcome of this blockage is a considerable degradation on the communication and sensing task.

We attacked the problem by generating artificial but realistic terrains, via terrain generation tools. We changed the generation parameters, and obtained various terrain representations, *heightmaps*. The results became more clear, when the heightmaps have been rendered into 3D terrains with a rendering software. We used four terrains as the testbeds for our simulations: Hilly terrain, ledgy terrain, mild-hilly terrain and smooth terrain. Our goal was to experiment with a collaborative target tracking application in different types of terrains and see the topographic effects on the *tracking accuracy*, which is a combined task of communication and detection in the network.

The outcome of our simulations, are the *mean error* which is a measure of track-

ing accuracy, the *number of communicating sensor pairs* and the *number of detecting sensors*.

In the mean error comparison of four terrains and a 2D surface, the ledgy surfaced terrain gave out the highest mean error. In a ledgy terrain, although the peaks are small in size, they are enough to block the communication among sensors. The two hilly terrains that have similar mean errors, come after the ledgy-surfaced terrain, in the mean error. The smooth terrain gave out the lowest mean error. It is seen that even in the most dense scenario and in comparison with the flattest terrain, there is a high deviation between the mean errors of the 2D surface and the smoothest terrain, which makes the targeting application impractical in a realistic scenario.

We also evaluated the detection and communication capabilities of the WSN in the terrain environment. For the communication capability, we computed the number of communicating sensor pairs. Considering non-LOS conditions between the sensors in the terrain, we derived a communication matrix.

The result of the communication matrix turned out to be the flattest terrain having the highest number of communicating sensor pairs, due to its smooth topographic surface. Then comes hilly and mild-hilly terrains, giving out similar results, although the terrain having wide scaled peaks, has a higher communication rate. The ledgy surfaced terrain, gives out the worst performance on communication

As for the detection, the flattest terrain still achieves the best performance, due to its smooth surface. This time ledgy-surfaced gave the second best result in detection, although it gave out the worst performance on communication. This came from the fact that, having a ledgy surface is of little importance in the detection task, since the angle of the 3D line segment between the sensor and the target is high due to the height of the target (150 cm), and not affected by ledges. Mild-hilly with its less number of wide scaled peaks, has a higher detection rate than the hilly terrain. We also evaluated the performance of Kalman and Individual Information filter in terms of the mean error, using the terrains. It is shown that the filters give approximate results

with the detection rate of the terrains.

As a result, we have seen that the *target tracking accuracy* in a WSN is both dominated by communication and detection task which are also dominated by the topography of the terrains. It has been shown that the communication rate degrades in a ledgy surfaced terrain, although the terrain has little number of slopes. Large peaks, on the other hand are sources of obstruction in the detection phase.

We finally investigated the effect of terrain generation parameters and target height on mean error. We showed that when the amplitude and the frequency are increased, the mean error also increases. The frequency affects the turbulence of the terrain, creating peaks or ledges depending on the amplitude, thus decreasing the LOS conditions. The amplitude affects the height of the peaks or ledges, which again affects the LOS conditions and eventually the mean error. The harmonics is the number of noise functions used in the terrain generation. At fixed frequency and amplitude values, we showed that, larger number of harmonics generates smoother terrains and improves the tracking accuracy. Regarding different objects moving in the region, the target height could vary from 100 cm to 250 cm. We showed that as the target height increases, the detection capability and resultingly the tracking accuracy increases. The improvement in the mean error depends on the structure of the underlying terrain.

In this thesis, we mainly contributed to a new realistic performance evaluation scheme for WSNs. We have developed a terrain simulation environment, where the sensors are assumed to be deployed. Upon the high deviation of the results from those of 2D planar surfaces, we showed that the performance predictions could be misleading on the paper design, due to non-realistic assumptions on the WSN deployment region.

6.1. Future Work

Several scenarios could be added to make our simulations more realistic. Natural formations, e.g., river, sea, ponds, can be incorporated to our terrains. We can model that the sensors randomly deployed onto these regions, do not function any more.

We can also model the target's mobility according to the topography of the terrains. By this we can achieve a more realistic terrain simulation environment and evaluate our results under these assumptions.

The terrain impact on the communication other than clear LOS, might be another future study of this thesis. The fresnel zone information could be useful in calculating the signal attenuation between two sensors even if the path has a clear LOS. Here, since we have triangles rather than the height values between each sensor pair, the problem would converge to fresnel zone and triangle intersection problem.

APPENDIX A: TERRAGEN FILE FORMAT

A.1. File Structure

A terragen terrain file consists of a 16-byte identifier and a number of chunks. The positions of the various chunks are flexible, however, it should be noted that some chunks must appear before others.

The chunks contain its 4 byte identifier, e.g., 'ALTW' and then the chunk data. All chunks are aligned to the nearest 4 bytes.

A terrain file must contain the following:

1. At the beginning of the file, an 8-byte 'TERRAGEN' string.
2. After the 'TERRAGEN' string, an 8-byte 'TERRAIN' string.
3. A 'SIZE' chunk.
4. 'XPTS' and 'YPTS' chunks are required, if the terrain is not square.
5. A 4-byte 'EOF' string at the end of the file which is necessary for old versions of Terragen.

The actual elevation data is described after these chunks. Currently the elevation data is stored in the 'ALTW' chunk. The elevation and heightmap coordinate units (X, Y and Z) are defined in Terrain units, and not meters. Conversion to meters, can be done from Terragen heightmap settings; the default scale is 30 meters to one terrain unit. The actual scale value for a particular terrain is stored in the file's 'SCAL' chunk.

A.2. Chunks

- 'XPTS' 4-byte identifier.

  Appears after the 'SIZE' identifier and before any altitude data. This identifier is followed by a 2-byte integer value xpts, and 2 bytes of padding. xpts is equal

to the number of grid points in the x-direction.

- 'YPTS' 4-byte identifier.

  Appears after the 'SIZE' identifier and before any altitude data. This identifier is followed by a 2-byte integer value ypts, and 2 bytes of padding. ypts is equal to the number of grid points in the y-direction.

- 'SIZE' 4-byte identifier, necessary.

  Appears before any altitude data. This identifier is followed by a 2-byte value equal to n - 1, and 2 bytes of padding. In square terrains, n is the number of grid points in one direction. In non-square terrains, n is equal to the number of grid points in the shorter direction. For example, in a terrain with a heightmap of 400 grid points in the x-direction and 500 grid points in the y-direction, the size would have the value of 399.

- 'SCAL' 4-byte identifier, optional.

  Appears before any altitude data. This identifier is followed by three 4-byte floating point values (x,y,z). It represents the scale of the terrain in metres per terrain unit. Currently, Terragen allows only uniform scaling, so x, y and z scaling is equal.

- 'CRAD' 4-byte identifier, optional.

  Appears before any altitude data. This identifier is followed by one intel-ordered 4-byte floating point value, which is radius (in kilometers) of the planet being rendered. The default value is the approximate radius of the Earth which is 6370.

- 'CRVM' 4-byte identifier, optional.

  Appears before any altitude data. This identifier is followed by one unsigned integer. Mode 0 means the terrain is rendered on a flat surface assuming no earth curvature. Mode 1 means the terrain is stretched over a sphere with the radius $CRAD * 1000/zscale$, centred at $(midx, midy, -CRAD * 1000/zscale)$, where $midx = XSIZE/2$ and $midy = YSIZE/2$. In mode 1, the map will look normal when viewed from above, but geographic distances will be stretched towards the edge of the map if there is a lot of curvature.

- 'ALTW' 4-byte identifier.

  Appears after the 'SIZE' identifier and must appear after the 'XPTS' and 'YPTS'

identifiers. ‘ALTW’ stands for ’Altitude in 16-bit Words’. After the ‘ALTW’ identifier, the following values must appear in order:

- HeightScale, a 2-byte signed integer value.
- BaseHeight, a 2-byte signed integer value.
- Elevations, a sequence of 2-byte signed integers.

There are $(xpts * ypts)$ elevation values. xpts and ypts either have been set in the ‘SIZE’ chunk or the ‘XPTS’ and ‘YPTS’ chunks. The values in elevations are not absolute altitudes. The absolute altitude of a particular point (in the same scale as x and y) is equal to $BaseHeight + Elevation * HeightScale/65536$.

REFERENCES

1. Akyildiz, I. F., W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey", *Computer Networks*, Vol. 38, No. 4, pp. 393-422, 2002.

2. Huang,C. F., Y. C. Tseng, and Li-Chu Lo, "The Coverage Problem in Three-Dimensional Wireless Sensor Networks", *Proceedings of IEEE GLOBECOM*, Vol. 5, pp. 3182-3186, 2004.

3. Watfa, M. K. and S. Commuri, "Optimality Measures for Coverage in 3D Wireless Sensor Networks", *1st International Symposium on Wireless Pervasive Computing*, pp. 1-6, January 2006.

4. Akyildiz, I. F., S. Weilian, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor Networks", *IEEE Communications Magazine*, Vol. 40, No. 8, pp. 102-114, August 2002.

5. Callaway, E. H., *Wireless Sensor Networks, Architectures and Protocols*, Aucherbach Publications, 2004.

6. Shih, E., S. H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks", *Proceedings of ACM International Conference on Mobile Computing and Networking*, pp. 272-287, July 2001.

7. Woo, A. and D. E. Culler, "A transmission control scheme for media access in sensor networks", *Proceedings of ACM International Conference on Mobile Computing and Networking*, pp. 272-287, July 2001.

8. Ye, W., J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks", *Proceedings of IEEE INFOCOM*, pp. 1567-1576, 2002.

9. Demirkol, I., C. Ersoy, and F. Alagoz, "MAC Protocols for Wireless Sensor Networks: a Survey," *IEEE Communications Magazine*, Vol. 44, No. 4, pp. 115-121, April 2006.

10. Braginsky, D. and D. Estrin, "Rumor routing algorithm for sensor networks", *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 22-31, October 2002.

11. Ganesan, D., R. Govindan, S. Shenker, and D. Estrin, "Highly resilient, energy efficient multipath routing in wireless sensor networks", *ACM Mobile Computing and Communication Review*, Vol. 5, No. 4, pp. 11-25, October 2001.

12. Heinzelman, W. R., A. Chandrakasan, and H. Balakrishnan, "Energy- efficient communication protocols for wireless microsensor networks", *Proceedings of 33rd Annual Hawaii International Conference on Systems Science*, pp. 3005-3014, January 2000.

13. Clouqueur, T., V. Phipatanasuphorn, P. Ramanathan, and K. Saluja, "Sensor Deployment Strategy for Detection of Targets Traversing a Region", *Mobile Networks and Applications*, Vol. 8, No. 4, pp. 453-461, August 2003.

14. Howard, A., M. J. Mataric and G. S. Sukhatme, "An Incremental Self-Deployment Algorithm for Mobile Sensor Networks", *Autonomous Robots*, Vol. 13, No. 2, pp. 113-126, September 2002.

15. O'Rourke, J., "Computational geometry", *International Journal of Computational Geometry and Applications*, Vol. 2, No. 2, pp. 215-217, 1992.

16. Tian, D. and N. D. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks", *Proceedings of ACM International Workshop on Wireless Sensor Networks and Applications*, October 2002.

17. Wang, X., G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage

and connectivity configuration in wireless sensor networks", *Proceedings of ACM International Conference on Embedded Networked Sensor Systems*, pp. 28-39, November 2003.

18. Yan, T., T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks", *Proceedings of ACM International Conference on Embedded Networked Sensor Systems*, pp. 51-62, November 2003.

19. Slijepcevic, M. P., "Power efficient organization of wireless sensor networks", *Proceedings of IEEE International Conference on Communications*, Vol. 2, pp 472-476, June 2001.

20. Liu, B., and D. Towsley, "A Study of the Coverage of Large-scale Sensor Networks," *Proceedings of the First IEEE International Conference on Mobile Adhoc and Sensor Systems*, 2004

21. Onur, E., C. Ersoy and H. Delic, "Finding Sensing Coverage and Breach Paths in Surveillance Wireless Sensor Networks", *Proceedings of the IEEE PIMRC*, September 2004.

22. Dhillon, S. S., K. Chakrabarty, "Sensor placement for effective coverage and surveillance in distributed sensor networks", *IEEE Wireless Communications and Networking*, Vol.3, pp. 1609-1614, March 2003.

23. Maroti, M., G. Simon, A. Ledeczi and J. Sztipanovits, "Shooter localization in urban terrain", *IEEE Computer*, Vol. 37, No. 8, pp. 60-61, August 2004.

24. Henderson, T. C., E. Grant, K. Luthy and J. Cintron, "Snow monitoring with sensor networks", *Proceedings of 29th Annual IEEE International Conference on Computer Networks*, pp. 558-559, November 2004.

25. Bulusu, N., D. Estrin and J. Heidemann, "Scalable Coordination for Wireless Sensor Networks: Self-Configuring Localization Systems", *Proceedings of the 6th In-*

ternational Symposium on Communication Theory and Applications, July 2001.

26. Ji, X. and H. Zha, "Sensor Positioning in Wireless Ad-hoc Sensor Networks Using Multidimensional Scaling", *Proceedings of IEEE Conference on Computer Communications*, Vol. 23, No. 1, pp. 2652-2661, March 2004.

27. Ravelomanana, V., "Extremal properties of three-dimensional sensor networks with applications", *IEEE Transactions on Mobile Computing*, Vol. 3, No. 3, pp. 246- 25, July-August 2004.

28. Watfa, M. K., and S. Commuri, "A Coverage Algorithm in 3D Wireless Sensor Networks", *Proceedings of the First International Symposium on Wireless Pervasive Computing*, pp. 1-6, January 2006.

29. Tezcan, H., E. Cayirci and V. Coskun, "A distributed scheme for 3D space coverage in tactical underwater sensor networks", *Proceedings of IEEE Military Communications Conference*, Vol.2, pp. 697-703, October-November 2004.

30. Onel,T., C. Ersoy and H. Delic, "An Information Controlled Transmission Power Adjustment Scheme for Collaborative Target Tracking", *Proceedings of IEEE Symposium on Computers and Communications*, June 2006.

31. Zhao, F., J. Shin and J. Reich, "Information-driven dynamic sensor collaboration", *IEEE Signal Processing Magazine*, Vol. 19, No. 2, pp 61-72, March 2002.

32. Brooks, R.R., P. Ramanathan and A. M. Sayeed, "Distributed target classification and tracking in sensor networks", *Proceedings of IEEE*, Vol. 91, No. 8, pp. 1163-1171, August 2003.

33. Li, D., K. Wong, Y. Hu, and A. Sayeed, "Detection, classification and tracking of targets in distributed sensor networks," *IEEE Signal Processing Magazine*, Vol. 19, No. 2, March 2002.

34. Oh, S., S. Sastry, and L. Schenato, "A hierarchical multiple-target tracking al-

gorithm for Sensor Networks", *Proceedings of IEEE Interantional Conference on Robotics and Automation*, April 2005.

35. Yan, T., T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks", *Proceedings of ACM International Conference on Embedded Networked Sensor Systems*, November 2003.

36. Du, X. and F. Lin, "Efficient energy management protocol for target tracking sensor networks", *Proceedings of 9th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 45-58, May 2005.

37. LaSor, J. W., *GeoFrac 2000*, http://www.geofrac2000.com/, 2005.

38. Planetsite Software, *Terragen photorealistic rendering software*, http://www.planetside.co.uk/terragen/productmain.shtml, 2005.

39. Moller, T. and B. Trumbore, *Fast, Minimum Storage Ray/Triangle Intersection*, http://www.graphics.cornell.edu/pubs/1997/MT97.html, 2002.

40. Moller, T. and B. Trumbore, "Fast, Minimum Storage Ray/ Triangle Intersection", *Journal of Graphics Tools*, Vol. 2, No. 1, pp. 21-28, 1997.

41. O'Malley, S., *Matlab script for Terragen*, http://www2.cs.uh.edu/∼ somalley/, 2006.

42. Elias, H., *Perlin Noise*, http://freespace.virgin.net/hugo.elias/models/m_perlin.htm, 2006.

43. Bourke, P., *Perlin Noise and Turbulence*, http://astronomy.swin.edu.au/∼ pbourke/texture/perlin/, 2000.

44. Martz, P., *Generating Random Fractal Terrain*, http://www. gameprogrammer.com/fractal.html, 2006.

45. Bar-Shalom, Y., X. R. Li and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley & Sons, Inc., 2001.

46. Mo, L., X. Song, Z. Sun, and Y. Bar-Shalom, "Unbiased converted measurements for tracking", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 3, pp. 1023-1026, 1998.

47. Lerro, D. and Y. Bar-Shalom, "Tracking with debiased consistent converted measurements versus ekf", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 29, No. 3, pp. 1015-1022, July 1993.

48. Grocholsky, B., A. Makarenko, and H. Durrant-Whyte, "Information-theoretic coordinated control of multiple sensor platforms", *Proceedings of the IEEE International Conference on Robotics & Automation*, pp. 1521-1526, September 2003.

49. Advantaca, http://www.advantaca.com, 2006.

50. Kotz, D., C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions", *Proceedings of the ACM/IEEE Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 78-82, October 2004.