Localization in Real World - Virtual World

Submitted by

CHHAVI SHARMA

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2009

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Lawrence WC Wong for the continuous support and guidance during my research and study at National University of Singapore. His perpetual energy and enthusiasm in research has motivated all his students, including me.

Besides my supervisor, I would like to thank Dr Soh Wee Seng for his valuable inputs and insightful comments without which I would not have been able to finish the second part of my thesis.

I would also like to thank Dr Tan Ben Kiang, Truc and Daniel from the Department of Architecture, for their help during the creation of 3D models.

I warmly thank my lab officers Mr Goh Thiam Pheng and Mr Song Xianlin for their support which made my research smooth and stay enjoyable at Communications Lab.

I would like to thank all my labmates for their advice, criticisms, ideas and support. My sincere thanks goes to the folks from the Wonderland Forums and Wiki, for helping me out when I got stuck.

My deepest gratitude goes to my family and friends for being so supportive throughout the course of my study. Without their encouragement and understanding, it would have been impossible for me to finish this work.

Summary

The objective of this thesis is to set-up a 3D mirror world where the objects and the events that are happening in the real physical world can be visualized. The focus and the scope of this thesis is to visualize the tracking in the mirror world for the equivalent objects and people in the real world.

This thesis can be divided into two parts. The first part deals in creating a platform for the visualization of a real world environment, i.e. Communications Lab in NUS Campus. Sun Microsystem Project Wonderland(WL), an open source toolkit, is used as a basic 3D building environment. A centralized web-server is set up such that the mirror world can be accessed using a web browser. A virtual 3D model of the Communications Lab in the Department of Electrical and Computer Engineering is successfully created and imported into the WL. The WL source code (WL is open source) is modified to create a communication channel between the imported mirror world and the real-world applications. The channel is also used to exchange location information between the real world and the virtual world. The real world location information is obtained from a Wi-fi based indoor localization system, Ekahau. A new interface is developed in Java which integrates the Ekahau localization system with the WL.

The second part of the thesis deals in proposing a new approach for the place-

ment of access points for Wi-fi based Fingerprinting localization such as Ekahau. The main idea behind the AP placement algorithm is to minimize the total number of similar fingerprints over the entire array of receiver locations, by varying the access points' location. To solve the optimization problem, a heuristic optimization algorithm *Simulated Annealing* has been implemented as simple brute force search method would be highly computationally expensive and inefficient. Numerical results are obtained for both the brute force search and the Simulated Annealing optimization approach. The proposed algorithm's output, i.e. the access points' location has been applied to a k Nearest Neighbor based localization system and location error has been analyzed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine a scenario where you are at home, sick with fever but you need and want to attend an important meeting at your office. You are prescribed complete bed-rest by the doctor but missing the meeting means big loss to the company and also to your next promotion. Is it possible for you not to move from your bed and still attend the meeting, interact with the speaker and other audiences at the seminar room? Consider another scenario where, you want to play tennis with your friend at your favorite university playground while both of you are at different places. Is it possible for both of you to still play a realistic game without actually being at the same place (A realistic game implies that it is driven by gestures rather than a keyboard or a mouse). Looks like the scenes from Hollywood Sci-Fi blockbusters? Not really, this is the future of Internet aka '3D Internet' and this project is a stepping stone towards it. This project creates an 'Interactive Mirror World' which offers endless possibilities for communication and information transfer from the real world to a mirror world and vice versa.

To realize the scenarios mentioned above, the first and the foremost requirement is the existence of a Mirror world, i.e. a virtual world which is a close replica of the real world. However, no 3D virtual world, as close to real as possible, can be called a mirror world until it is constantly updated with the real world events. This requires a communication channel for the seamless exchange of information between the two worlds. There are many existing 3D virtual worlds like, SecondLife(SL) [1] , HabboHotel(HH) [2] where people can chat, meet and even create their own virtual surroundings but they are more like a fantasized world.

To set-up a visualization platform, which not only creates 3D contents efficiently but also supports a communication channel for seamless exchange of information between the real world and the mirror world is one of the main objectives of this thesis. Sun Microsystem's 'Project Wonderland' (WL) [3] , an open source toolkit for building 3D worlds, has been used as the visualization platform for this project. Though SL is a much more advanced platform for creating virtual worlds but since it is not open source, modifying it according to our specific requirements is difficult.

Though the communication channel must be able to communicate any type of information both ways but the scope of this thesis is limited to the transfer of location information. Ekahau [4], a Wi-fi based Fingerprinting localization system, has been deployed and integrated with the WL to obtain the real world location information for users and devices. Since the Ekahau system is based on Wi-fi signal strength, its accuracy depends upon the deployment of APs. After extensive literature search, it has been observed that the placement of APs such that the location estimation error can be minimized is not a very well studied problem, though the problem has been studied widely from a communication point of view.

Many heuristic and intuitive methods suggest some guidelines to place APs for fingerprinting based localization but a scientific approach to the problem is required. Hence we proposed a novel optimization algorithm for placing the APs for fingerprinting type of localization.

The next section gives an overview of the thesis contributions, followed by an outline of this thesis.

1.1 Thesis Contributions

This thesis can be divided into two parts and the main contributions of this thesis are as follows:

- A visualization platform has been set up to mirror a laboratory in NUS campus using WL. A method has been proposed and implemented to create a communication channel through which location information can be transferred from the real world to the mirror world. Ekahau location sensing system has been deployed to receive the real world location information and then integrated with the WL to transfer the real world location to the mirror world using the communication channel.

- A novel optimization model has been proposed for the placement of APs for Wi-Fi based Fingerprinting type of localization. This model takes the floor map of the building and the total number of APs as input and suggests locations for the placement of APs such that the location estimate error can be minimized. This model is independent of the implementation of the underlying localization system, however, the localization system should be based upon a fingerprinting approach.

1.2 Thesis Outline

This thesis is divided into two parts. The first part deals with the set up of a visualization platform and spans from chapter 2 to chapter 3. Chapter 2 provides the background information of WL which includes a brief overview followed by the description of the underlying technologies upon which WL is built. It also provides the information regarding virtual world software architecture and communication infrastructure in the WL. In addition, this chapter also describes fingerprinting based localization and the Ekahau system. Chapter 3 presents the implementation details for WL centralized server set-up, it also describes the process and the limitations of creating and importing the new content, i.e. own virtual world in the WL. The methodology to integrate the Ekahau system with WL is also presented in detail in this chapter. The changes required in the WL software architecture for creating a communication channel and transferring location information provided by the Ekahau system is presented in this chapter along with the pseudo-codes.

The second part of the thesis focuses upon the placement algorithm for APs from a fingerprinting localization point of view and spans from chapter 4 to chapter 6. Chapter 4 presents the current state-of-the-art methods for the placement of APs. Though a generic approach is adopted to present the previous work done, the emphasis is given to the work which has been done from the view point of location dependent services. Chapter 5 presents the new algorithm for the placement of APs. It starts with a brief overview of the indoor propagation model used for this algorithm and later presents and develops the optimization model. The last section of this chapter presents an overview of the Simulated Annealing(SA) Optimization Algorithm along with its implementation details in context

of APs placement. Chapter 6 presents the simulation platform and the results for the method proposed. It also presents the performance analysis of the proposed method against the manual heuristic approach and an existing scientific approach. The proposed algorithm has also been applied to a k Nearest Neighbor(kNN) based indoor localization method and location error performance has been analyzed.

Finally chapter 7 concludes the thesis and presents the scope for future work and directions.

Chapter 2

Background

This chapter provides the background on WL, an open source toolkit to create 3D virtual worlds, used to build the visualization platform for a laboratory in NUS campus. Ekahau has been used as the location sensing system and since it is a Wi-Fi based fingerprinting localization system, a brief introduction is provided for the fingerprinting localization method followed by the Ekahau system overview.

2.1 WL Overview

WL is a Java based, open source toolkit for creating collaborative 3D virtual worlds. WL allows creation and import of customized 3D contents, i.e. the virtual worlds. Within these virtual worlds, users can communicate with high-fidelity, immersive audio and can share live applications such as web browsers, OpenOffice documents and games. *Project Darkstar*, *Java 3D* and *jVoiceBridge* are the foundation technologies, upon which WL is built.

Project Darkstar [5] is a Java based server software infrastructure which provides event-driven programming model features and easy-to-use services for managing communications, tasks and data access. These services shield the complexities of multi-threaded and distributed systems programming. It also provides APIs for task scheduling, threading, distribution, contention management, load balancing and transaction management. All of the coarse grained behaviour of the communication between a client and the server can also be handled easily using the API.

Java3D [6] is a low level 3D scene-graph based graphics programming API for the Java language. It provides routines for creation of 3D geometries in a scene-graph structure that is independent of the underlying hardware implementation for realtime programming. The API provides scenegraph compilation and other optimization techniques.

jVoiceBridge [7] is a software audio mixer written in the Java Programming Language. It handles Voice over IP (VoIP) audio communication and mixing for the tasks such as conference calls, voice chat, speech detection, and audio for 3D virtual environments. It provides real time immersive stereo audio with distance attenuation in WL.

Subsequent sections will provide the details regarding virtual world architecture and communication infrastructure in WL.

2.1.1 Virtual World Architecture in WL

In WL, the virtual world has its own coordinate system. From a viewer's perspective, +X axis is towards the right, +Y axis is gravitationally up and +Z axis is towards the user. The entire virtual world is a collection of different 'cells'

with each cell representing a 3D volume in the virtual world and has position and bounds in the virtual world. Bounds represent the physical extent of the cell.

Cell Coordinate System In the virtual world coordinate system, a cell's origin is marked with respect to its center, however, a cell also has its local coordinate system where the center of the cell is at (0,0,0). The cell structure is hierarchical and a cell may contain zero or more child cells within itself.

Cell Architecture There are two types of cells: *Stationary* and *Moveable*, which is defined at the time of cell's creation. For example, Avatar is defined as a moveable cell and the virtual world contents like buildings, furnitures etc. are defined as stationary cells. If an object is defined as a stationary cell, it cannot be moved from one location to another at later stages. Each cell has a unique ID and a unique name; the cell name is formed as "CELL_" plus cell ID. A communication channel is also attached with each cell which is used to send and receive messages to/from the cell. Figure 2.1 *(courtesy: WL Official Website)* provides sample cell structures in WL.

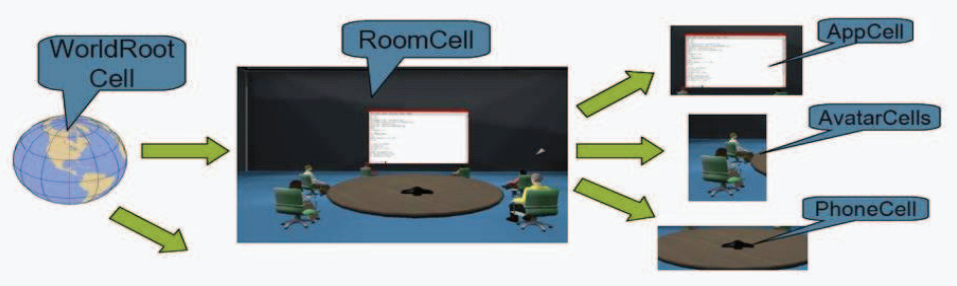


Fig. 2.1: Cell Structure at WL

Cell Representation at WL Cells are defined as XML files in the WL File System (WFS). Each XML file contains information about cell type, i.e. stationary or moveable, cell coordinates with respect to the virtual world, cell bounds and the disk location where actual 3D geometry for the cell is stored. Whenever a new 3D content is created or imported into WL, a new XML file is written into WFS along with the creation of 3D geometry.

2.1.2 Communication Infrastructure in WL

WL is built upon the Darkstar server framework, hence it primarily provides a 'Client-Server' mode of communication. But to avoid overloading the server for minor client level changes, it also provides a direct 'Peer to Peer communication' between the clients. For example, when a client first connects to the server, the client-server communication mode is used and if two clients want to share some application or chat with each other, the direct communication mode is used. The direct 'Peer to Peer' mode of communication is also called 'Channel Communication'. WL mainly uses three different communication mechanisms and protocols. Figure 2.2 *(courtesy: WL Official Website)* presents the different communication protocols used by WL.

1. Darkstar communication is the most frequently used communication mode. Updates such as login event for the client, positioning the avatar, sending and receiving updates to/from clients, logging off event, etc. all use Darkstar communication

2. Standard Session Initiation Protocol(SIP) and Real-Time Transport Protocol(RTP) are used by the jVoiceBridge module of WL for sending the voice

data from the server to the client

3. Peer-to-peer protocol(PPP) is used by WL for application sharing among the clients. Applications such as Firefox web-browser, terminal, chat messages, etc. are shared using PPP.

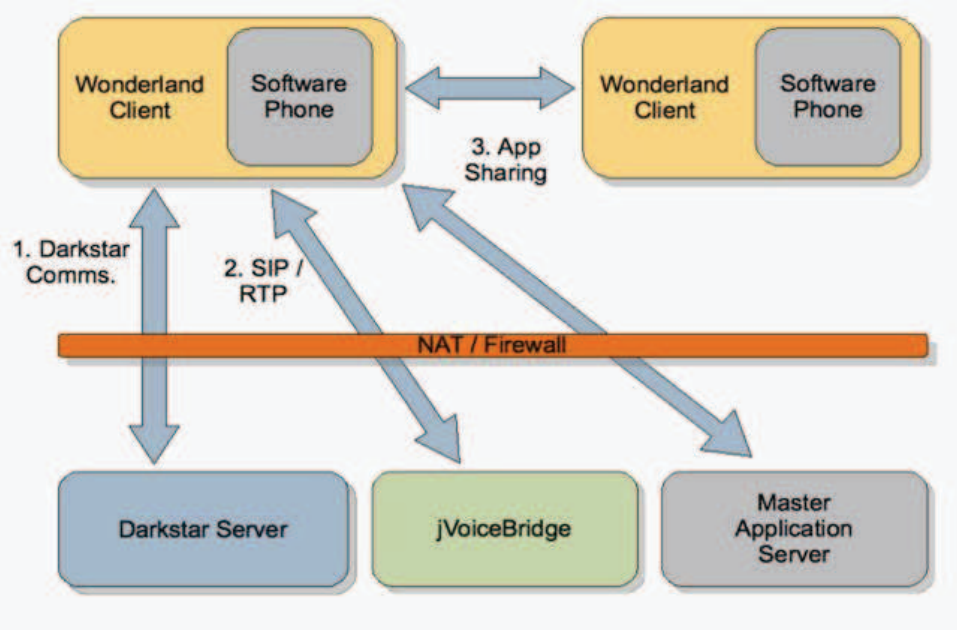


Fig. 2.2: Communication Protocols at WL

2.1.3 Hardware Specification for WL Implementation

WL does not require specific hardware; minimum specifications required to implement a WL baseline system are as follows:

- PC (1.5Ghz+, 1GB RAM) with hardware-accelerated OpenGL drivers installed. For Solaris and Linux, nVidia cards/drivers are recommended

- Accelerated Graphics card with 128MB is the minimum requirement, however a graphic card with 256MB video memory is recommended for the best performance.
- Java SE 6 (JDK 6)
- WL supports Linux, Windows, Solaris and MAC operating systems but application sharing is supported only for X11 applications and hence available on Linux and Solaris platforms only. Shared X11 applications must be launched from a Linux or Solaris client but can be viewed and controlled from any other client (including Windows and Mac OS)

2.1.4 WL and Other Virtual World Toolkits

The objective of this project is to build a mirror world, and unlike most of the virtual worlds, one of the main requirements is the real time information transfer to and from mirror world to the physical world. Unlike SL, WL is open source and hence infinitely flexible to any changes within the system design constraints. Therefore, it can be modified and utilized to cater to our project specific needs. SL is more advanced to WL in terms of visual experience and content creation whereas the main focus in WL is on various infrastructure demands like application sharing, integrating with authentication services using LDAP protocol, voice communication and telephony integration. WL is written in Java and hence has potential to run on hand held devices without much code change, unlike SL.

2.2 Fingerprinting Localization and Ekahau

2.2.1 Localization Algorithm

Fingerprinting is a Wi-Fi based localization technique [8] for indoor localization, which utilizes the relationship between a particular location and its corresponding received signal strength (RSS) value. This type of location sensing system does not require any dedicated hardware other than a network system with existing wireless interfaces. Hence it is much easier to install compared to the other systems. The algorithm can be classified as a pattern-matching algorithm and works in the following two phases:

2.2.1.1 Calibration (Off-line)Phase

This phase is equivalent to the training phase in a pattern-matching algorithm. During this phase, a database of RSS patterns, which is called a *radiomap*, is created. The radiomap consists several *location - RSS vector* pairs and every RSS vector is called a *fingerprint* or *radio-signature* for the corresponding location. The RSS vector, i.e. the fingerprint depends on the number of APs *heard* at a particular location. If n APs can be *heard* at the i^{th} location, the radio map database entry at the i^{th} row can be presented as

$$(x_i, y_i, z_i; \quad RSS_1 \quad RSS_2 \quad ... \quad RSS_n) \tag{2.1}$$

where (x_i, y_i, z_i) represents the i^{th} location and RSS_k represents the RSS value received from the k^{th} AP. Location fingerprints are collected by measuring the RSS value from multiple APs at predefined locations using a mobile unit. The process

of collecting RSS values at certain known locations is also called *Site-Survey*.

2.2.1.2 On-line Phase

During the online phase, a RSS vector is measured at the test location, i.e. the location which needs to be estimated. The measured RSS vector is then compared with all the fingerprints collected during the calibration phase, using an appropriate matching algorithm, to estimate the location. The matching algorithm calculates the *Euclidean Distance* between the measured RSS vector and all the fingerprints and then returns the corresponding location for the one which is at the minimum distance from the measured RSS vector. *kNN* method is generally used as a matching algorithm in which k best matches for the measured RSS vector are identified and then their corresponding locations are averaged out to obtain the test location estimate.

The next section will provide a system overview of the Ekahau Real Time Location Tracking System (RTLS) which is used as the location sensing system for this visualization project.

2.2.2 Ekahau RTLS

The Ekahau RTLS is a fingerprinting based, Wi-Fi location tracking solution that can operate over 802.11 a/b/g/n generations of a Wi-Fi network. It officially claims to achieve an accuracy of 1 to 3 m, with a site survey and calibration process that requires up to 1 h/1200 m area. The Ekahau RTLS architecture is shown in Figure 2.3 *(courtesy: Ekahau Official Website)*. The main components of the Ekahau RTLS are as follows:

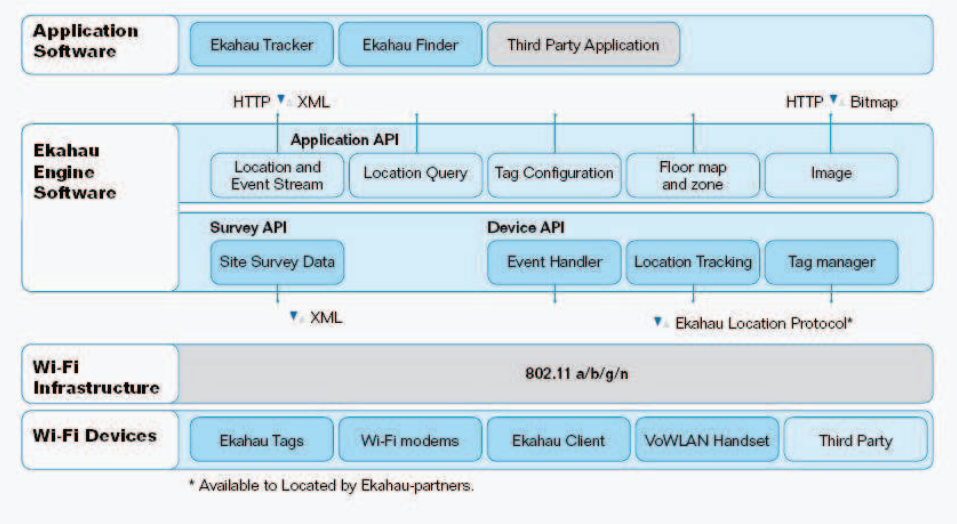


Fig. 2.3: Ekahau RTLS Architecture

2.2.2.1 Ekahau Site Survey

The Ekahau Site Survey(ESS) is a calibration tool which is used to create a radio map for the first phase of fingerprinting localization. The ESS software takes the floor map or map image as input to create the positioning model. Map scale is required to be set to the appropriate value when the map is imported for the first time. Then the user can draw rails on the map which are the possible travel paths between rooms, corridors, floors, and other locations. Once the rails are drawn, the user has to move along with the laptop and an Ekahau compatible Wi-Fi adapter or Ekahau tag, along the rails, to record the RSS samples in the entire area. RSS samples are collected by stopping at every 2-3 meters (recommended) and clicking on the map at the corresponding location. The ESS also provides visualizations in form of the heatmaps for calibration quality, network health, number of APs heard at every location, signal to noise ratio, etc. Figures 2.4 and 2.5 provides some snapshots of the ESS visualizations. Once the entire area has been calibrated, the

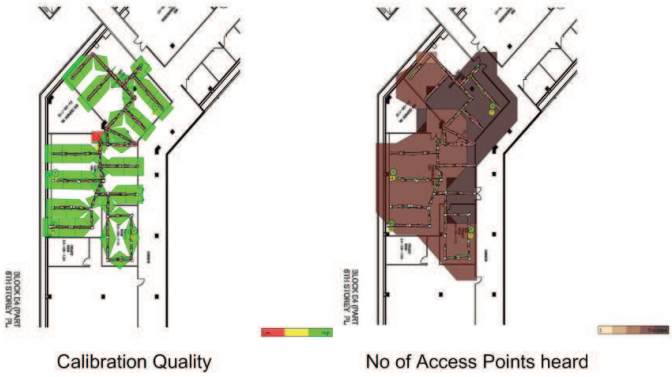


Fig. 2.4: ESS Visualization I

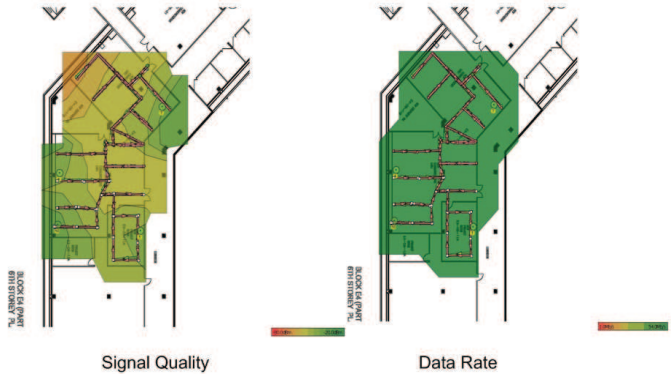


Fig. 2.5: ESS Visualization II

positioning model is uploaded to the Ekahau Positioning Engine database.

2.2.2.2 Ekahau Positioning Engine

The Ekahau Positioning Engine(EPE) is the software implementation of the localization algorithm which runs on a dedicated Windows Server platform. The EPE receives the RSS values reported by the tags and compares the measurements with an existing reference data, i.e. the radio map created and uploaded by the ESS during the calibration phase. The EPE then, calculates the location estimates

for the tags. The EPE polls the tags periodically (configurable period) to receive the respective RSS values. Alternatively, tags can also update the RSS values at the EPE without being polled and this is done by triggering an event, i.e. by pressing the buttons provided on them. The EPE also provides APIs along with a Java based Software Development Kit (SDK) to integrate the Ekahau system with other applications.

2.2.2.3 Ekahau Tags

The Ekahau tags are small battery-operated devices that are attached to the objects or carried by the people required to be tracked. Tags measure signal strengths from APs and transmit the measurements through an 802.11 network to the Ekahau Engine in real-time. They provide two-way communication and can be used during the calibration phase to record the fingerprints along with the ESS software. It is recommended to use similar tags for both the calibration and the tracking process as it improves the overall accuracy of the location estimates.

Chapter 3

3D Visualization using WL

The objective of this project is to set up a 3D visualization platform for a laboratory in NUS campus such that real time information can be transferred into a mirror world and vice versa, seamlessly. As the first step, a fully functional centralized server for WL has been set up on a Ubutu platform. A virtual 'Communication Lab', which is a mirror of physical 'Communication Lab' located in the Department of Electrical and Computer Engineering, has been created and imported into WL. This virtual Communication Lab was created with the help of colleagues from the Department of Architecture. Finally WL has been integrated with a location sensing system *Ekahau* to transfer the real time location information into WL. We now have the virtual 3D platform in which a user's avatar operates from the real-time location inputs obtained from the location sensing system instead of keyboard or mouse inputs. The above steps are described in detail in the subsequent sections.

3.1 WL Centralized Server Setup

WL is both binary (pre-compiled version) and source code distributed. Since the source code requires modifications; it (source code) is checked out from the CVS versioning system and built into Netbeans 6.0. Both the Server IP address and the WL File System location are changed for this build. To set up a web server for the WL, a web application archive file (war) needs to be built. The process of building the war file involves two steps 1.) creating a keystore file and 2.) running the command

ant -Dwonderland.useLocalArt=true pkg-war-combined

from a Ubuntu Terminal. Once the war file is built successfully, it is deployed into a web container so that users can access the web server through a web browser. Here 'Glassfish Application Server' is used as a web container to deploy the war file. The entire process is explained in a flowchart in Figure 3.1.

Users launch the WL based mirror world from the web browser. The Darkstar module prompts the user for login information and on successful login, Java webstart downloads and caches the '3D mirror world' at the user's local machine according to his/her visual range. Caching of the '3D mirror world' at a local machine decreases response time for any future logins. However, the cache can be cleared at user's discretion.

This server supports additional features like voice over IP calls, text chat, application sharing such as Firefox browser and terminal in the mirror world.

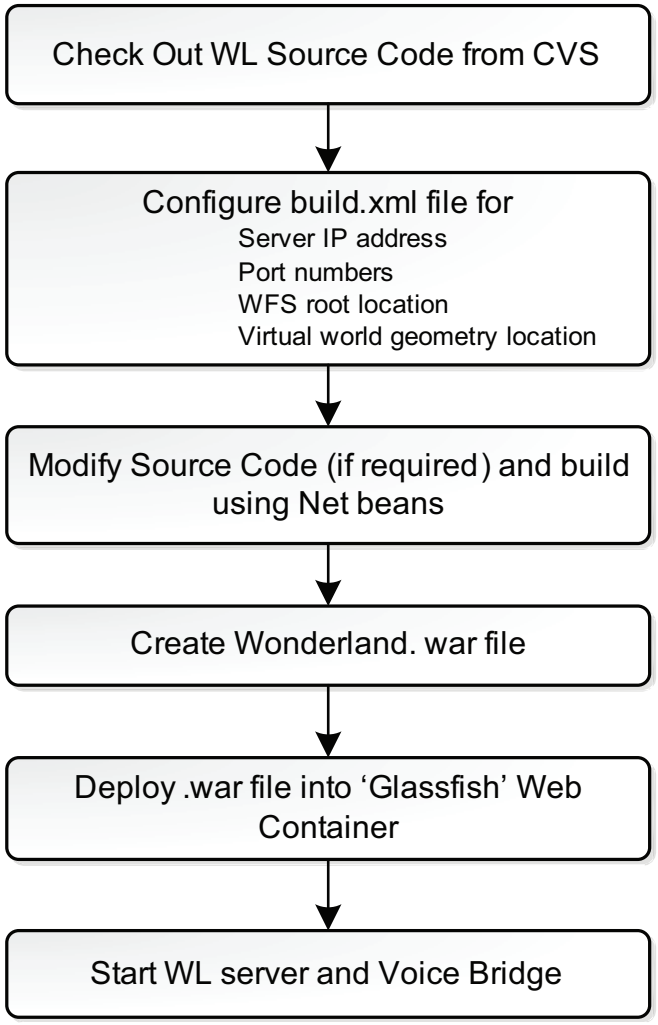


Fig. 3.1: WL Centralized Server set-up

3.2 Importing 3D Models into WL

As a unitary step towards creating a mirror NUS campus, we start with creating a mirror Communication lab. WL supports creation of 3D models using an inbuilt tool called 'World Builder' but this tool is very primitive. It supports very limited geometry and textures and hence not suitable for creating high end, sophisticated 3D contents. Another method to create the mirror world within WL is to import the 3D models created by another 3D modeling tools, e.g. 3ds max, Maya, Blender, etc. But the import process also has limitations on the input format of the 3D model; only 3D models in .x3d format can be imported. We have created 3D models for the Communication Lab using 3ds max and converted it to .x3d format using an another independent converter.

Whenever a new model is imported into WL, a new cell is created at the WL server for the same. WL accepts .x3d model as input file and generates an XML and a 3d geometry file(.j3s format) as the output at the server. The XML file defines the cell type, location coordinates and bounds information for the new 3D model. Bounds define the physical extent of 3D model depending upon the size of the model. The XML file also contains a pointer to the disk location where the actual 3D geometry, .j3s file, is stored. The files describing various Textures are copied manually at the WL server. Figure 3.2 summarizes the model-import process at WL. Figure 3.3 presents a snapshot of virtual Communication Lab in WL along with the real Lab.

We have observed the following limitations and issues for artwork import at WL:

- WL only accepts .x3d models.

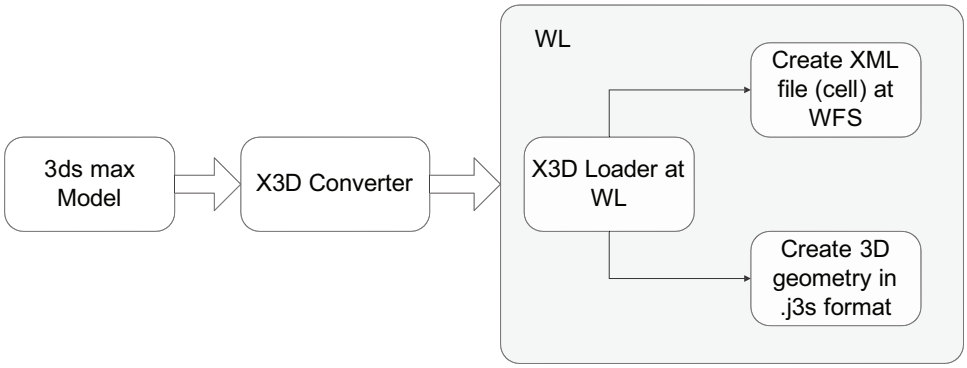
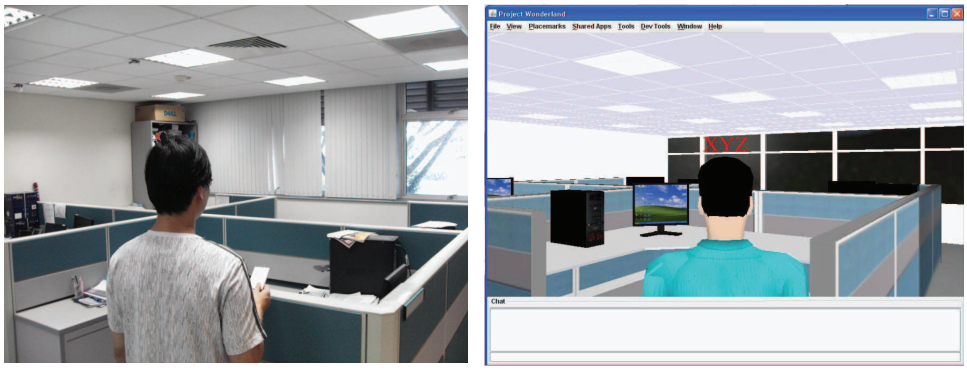


Fig. 3.2: Art Import Process at WL



Real Communication Lab with a user carrying Ekahau Tag

Mirror Communication Lab in WL

Fig. 3.3: Real and Virtual Communication Lab

- WL does not accept the 3D models with built-in texture files. For example, if a 3D model is created such that the texture is directly applied on the model in the form of a colour, the model cannot be imported into WL. This is a severe limitation since having separate texture files increase the 3D model size by manyfold which in turn degrades the performance of WL

- While importing 3D models, only virtual world location can be specified. One cannot scale the model on the fly, this is required many times as the units used by the modeling software and the WL virtual world are different.

- WL throws exceptions if 3D geometry file is very complex or texture file is too big

- WL does not support real time rendering for shading effects, hence the 3D model appears as a flat 2D picture in WL. The only workaround to this limitation is to paste the shading effects as the textures on the 3D models.

3.3 Real Time Location Update

With the exception of audio communication, keyboard and mouse inputs, WL does not support any real time data input from any sensor or device. A user's avatar movement is controlled by keyboard and mouse and location change event is initiated from a client machine. Once the avatar moves to a new location, the client sends an update to the server which in turn updates all other avatars, which are in the visual range of the moved avatar, about this event. The requirement of this project is to transfer the real time location information of the user to the virtual world and make the avatar move accordingly - hence requires integration of

a location sensing system with WL. The WL architecture also needs to be modified such that a location change event can be initiated from the server instead of a client.

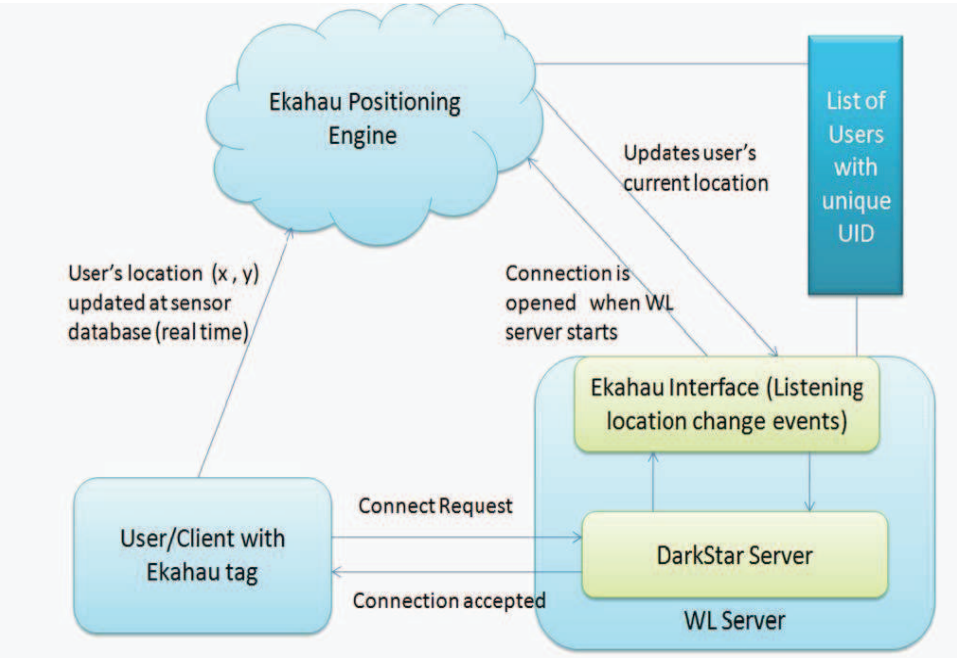


Fig. 3.4: Framework for WL and Ekahau Interface

3.3.1 WL and RTLS Interface

We have used the Ekahau real time location sensing (RTLS) system to track the users in the real world. The WL and RTLS servers run on two different machines and have to be integrated together via a new interface. Let's call it WL-RTLS interface. A user, assigned with a unique User ID (UID), carries an Ekahau tag having a unique tag ID (TID) in the real world so that it can be tracked by the RTLS server. This creates a unique UID-TID one-to-one mapping shared by both the WL and the Ekahau servers. Figure 3.4 describes the framework for WL and RTLS integration. The WL-RTLS interface, which binds WL and RTLS together,

performs the following tasks:

- Location coordinates provided by the Ekahau system are in terms of pixels; the WL-RTLS interface converts these pixel values to real world (x, y) coordinate values. Mapping from the pixel values to the actual coordinates is done beforehand and remains the same thereafter.

- When the WL server starts for the first time, the WL-RTLS interface sends a streaming connection request to the Ekahau RTLS server. If the connection request is successful, a session is created and remains open until the WL server is up and running.

- The WL-RTLS interface also initiates a session of Ekahau Location Listener(ELL) which listens to the location-change events. The location-change event can be triggered by pressing a button on an Ekahau tag. Alternatively, the RTLS server also polls every tag once every 10 seconds which is considered as a periodic location-change event. The ELL always remains updated with the most recent position for all the users.

The pseudo-code is presented in Figure 3.5.

3.3.2 Changes at WL Server

The foundation of the WL server is Darkstar, hence the Darkstar infrastructure has been utilized to incorporate real time transfer of location data. A periodic task is initiated by the 'TaskManager' at the start of the WL server which runs at a predefined 'Update Interval'. The 'Update Interval' is a variable, currently set to 1 second. This task retrieves the list of users, currently logged in the mirror world

```
---------------------  At WonderlandBoot Class  -----------------
void startEkahau ()
{
        SEND Connection Request to Positioning Engine
        IF 'Connection Accepted ' THEN
                ADD Location Listener
                IF 'Success' THEN
                        SEND command to start tracking
                ELSE
                        PRINT Error Message
                ENDIF
        ELSE
                PRINT Error Message
        ENDIF
}

-------------------------------------------------------------------------
--------------------------  At Location Listener  ----------------------

WHILE Connection is OPEN
        IF 'new event occurs '
                CONVERT the event description to string
                EXTRACT tagid and respective location
                TRANSLATE pixel values to  <x, y> cord
                UPDATE corresponding User  's location
        ENDIF
END WHILE

-------------------------------------------------------------------------
```

Fig. 3.5: Pseudo code : WL-Ekahau Interface

from the 'UserManager' and reads the location for every user from the ELL. If this task notices any change in the user's location, it communicates it to the respective user.

To communicate the location change information from the server to the client, Darkstar's 'Channel Mechanism' is utilized. In WL, every object including an Avatar is a 'cell' and every cell has a unique channel associated with it for communication purposes. But, the information transferred over this channel is in the form of a message. Hence for every 'location information transfer', a new message type 'LOC-CHANGE' is created and transferred over the 'AvatarP2PChannel' channel. The pseudo-code is presented in Figure 3.6.

3.3.3 Changes at WL Client

The client-code is also modified so that a client can understand the 'LOC-CHANGE' message and its avatar can move to the new location according to the information contained in the message. To recognize the LOC-CHANGE message, 'AvatarP2PChannelListener' is modified so that it can receive and decode the location 'LOC-CHANGE' message. A new 'stimulus' is added in the class responsible for the Avatar movement and animation to make it move to the new location. Once the listener receives a new 'LOC-CHANGE' message, it decodes it and then triggers the stimulus which in turn makes the Avatar walk to the new location. The pseudo-code is presented in Figure 3.7.

```
----------------------------------------------  At WonderlandBoot Class  ----------------------------------------

public void initialize ()
{
        GET DarkStar's TaskManager Reference
        CREATE new Task using Darkstar  's TaskManager
        SCHEDULE the task to run at specified interval
}

-------------------------------------------------------------------------------------------------------------------
----------------------------------------------------  At New Task  ------------------------------------------

public void run ()
{
        GET DarkStar's UserManager Reference
        FOR each logged in user obtained from UserManager Reference
                GET unique User ID
                READ the location for this User ID from LocationListener
                IF location for this user ID has been changed
                        CALL sendLocationChange  (userName, newLocation ) method
                ENDIF
        END LOOP
}

void sendLocChangedMessage  (String userName , Point3f newLocation )
{
        IF user if still logged in
                GET DarkStar's ChannelManager Reference
                GET User's Avatar's 'AvatarP2PChannel' using ChannelManager Reference
                CREATE a new location change message
                CONVERT the message to raw bytes
                SEND the message using  'AvatarP2PChannel'
        ENDIF
}

-------------------------------------------------------------------------------------------------------------------
```

Fig. 3.6: Pseudo code : Changes at WL server

```
----------------------   At AvatarP2PChannelListener Class  -------------------------

public void receivedMessage (ClientChannel clientChannel , SessionId sessionId ,
byte[] b)
{
        ADD new case to receive LOC _CHANGE message
        GET an instance of WalkBehavior class
        CALL locationChangedEvent (location) method using WalkBehavior instance
}

-------------------------------------------------------------------------------------
--------------------------------   At WalkBehavior class  ---------------------------------

ADD a new flag  'new message'

public void locationChangedEvent  (location)
{
        SET 'new message' TRUE
        UPDATE newLocation for the user
}

public void updatedata ()
{
        IF 'new message' is TRUE
                SET locDifference  = newLocation  - currentLocation
                CALCULATE number of steps avatar needs to walk
        END IF
        WHILE number of steps is not zero
                CALL updateState () method which makes avatar walk
        END WHILe
}

-------------------------------------------------------------------------------------
```

Fig. 3.7: Pseudo code : Changes at WL client

Chapter 4

Review of Access Point Placement Techniques

This chapter presents the current state-of-the-art in placing the APs. Our focus is to emphasize the work done from the location dependent services point of view and more specifically, for Fingerprinting based localization.

4.1 AP Placement - Communication Systems

The problem of placing APs has been studied widely by researchers and many solutions have been presented. But the problem has been analyzed from a communication point of view, i.e. the placement problem has been formulated such that it maximizes the signal coverage, data rate and SNR, and minimizes the bit error rate. In [9], a grid-based approximation algorithm has been proposed to place the APs in a manner which minimizes the number of APs required while ensuring

that along with the coverage, the received SNR at each location is sufficient to meet the offered load at that location. [10], [11] have also proposed algorithms for AP placement by maximizing the coverage at all receiver locations. In [12], a cost function for AP placement has been formulated such that it minimizes the Bit Error rate (BER) for a WLAN scenario. A non-linear optimization method 'Simulated Annealing' has been applied to solve the optimization problem. A variation of Nelder-Mead simplex method has been applied in [13], which implements a pattern search algorithm for minimization of the cost function, i.e. the ratio of covered points in a mesh. This algorithm also focusses upon coverage maximization.

4.2 AP Placement - Localization Systems

To optimize the location of APs for location dependent services, such that the location estimate error can be minimized, is still a very challenging and difficult problem. Wi-Fi based localization methods depend upon the RSS value received from multiple APs and hence, intuitively, the number and placement of APs should be an important factor in designing Wi-fi based localization algorithms. There are several papers in the literature which studies and analyzes the above intuition experimentally. In [14], the authors have developed an analysis framework and suggested to deploy at least four APs for fingerprinting based localization. In [15], a testbed has been developed for indoor localization using the Ekahau Fingerprinting system and the experimental results have been presented which shows that for the same number of APs, their placement can affect the location estimation error significantly. Their analysis even shows that the best placement strategy for APs from the coverage point of view does not yield good location accuracy and hence

not suitable for the localization systems. [16] also presents experimental results suggesting that increasing the number of APs decreases the location estimation error.

However, there are very few papers which actually proposes the AP placement algorithms for location dependent services. The following are a few approaches taken by the researchers in this area:

4.2.1 Euclidean Distance Maximization

In [17], the authors have proposed a method for AP location optimization in which the Euclidean distance (ED) of the received signal array among all the sampling points (receiver locations) has been maximized in order to increase the diversity of the RSS array. By increasing the diversity among RSS samples at all the receiver locations, the location estimation accuracy should also improve as higher diversity will in turn increase the chances of having unique 'fingerprints' at all the receiver locations. However, if we examine this method in detail for different scenarios, there seems to be a fallacy in this method.

Consider a simple scenario: Let R_1, R_2 and R_3 be three location fingerprints at three sampling points. In the first case, assume $R_1 >> R_2$ and R_3 ; R_2 and R_3 are very similar. In the second case, assume R_1, R_2 and R_3 are moderately different from each other. Using the ED Maximization method, chances of selecting the former case as the optimum output are much higher than that of the latter. However, the latter case seems to provide higher number of unique fingerprints. This argument can be very well supported by the following numeric example:
Case I:

$$R_1 = -60dbm, R_2 = -20dbm, R_3 = -15dbm \tag{4.1}$$

$$Euclidean \quad Distance \quad among \quad three \quad sampling \quad points = 90 \tag{4.2}$$

Case II:

$$R_1 = -20dbm, R_2 = -30dbm, R_3 = -40dbm \tag{4.3}$$

$$Euclidean \quad Distance \quad among \quad three \quad sampling \quad points = 40 \tag{4.4}$$

The Euclidean distance is much higher in case I than case II and hence this method will select case I as the optimal output. However, the localization algorithm will not be able to distinguish between receiver locations 2 and 3 in case I since both RSS vectors are more or less the same while in case II, all three receiver locations will be distinguished easily.

Though the first case seems to be unlikely for an empty room or outdoor scenarios but in real world indoor scenario, there are high chances of obtaining abruptly varying RSS patterns caused by various obstructions, furnitures, walls, doors, etc. The authors have not considered the effects of walls, doors and other obstructions in their propagation model.

4.2.2 Direct Location Error Minimization Techniques

In [18], the authors have proposed a method to place the APs optimally by directly minimizing the localization error. For this method, an estimation error model has been formulated first for the fingerprinting type of localization. This error model is based upon the likelihood of estimating the user's position as $\hat{x}$, given the user's actual position as x. Hence to make use of this error model, one needs to know the likelihood function for the underlying localization algorithm. Though the authors have assumed the likelihood function as a Guassian function and applied

it generally for all the localization algorithms, but this is a big assumption and not always true for the realistic scenarios. Hence to use this method accurately, an accurate specific likelihood function is required which may not be obtained easily for every localization algorithm.

[19] proposes linear and multiple regression methods to model the signal strength in indoor environments and then analyzes the relation between the received signal strength detection error and the localization error. It further suggests that the number of APs and their placement can effect the localization error significantly. The authors have also established a relationship between the standard deviation of predicted signal strength and the location estimation error and some experimental results have also been presented for a few AP combinations. However, the analysis assumes that the underlying localization system is triangulation or least square estimation based and hence it will not be valid for fingerprinting localization

A greedy AP placement method is proposed in [20] for a triangulation based localization algorithm where APs are placed such that, at every receiver location, the total number of APs heard is above a certain threshold degree. Here the underlying assumption is that in indoor environments, due to several obstacles, all APs may not be heard at every receiver location. However, this assumption is only valid if the APs are not placed at sufficient height. If they are deployed at ceiling heights, then this assumption can be overcome easily for most of the receiver locations. This method also requires the threshold degree to be determined experimentally beforehand which can be a tedious task.

4.3 Conclusion

A lot of research is undergoing in the area of various probabilistic and deterministic pattern matching algorithms in order to increase the accuracy of fingerprinting localization method but the AP placement problem is still in its infancy. To the best of our knowledge, only [17] provides a solution for placing APs optimally for fingerprinting localization. Hence, a new algorithm for the problem has been presented in the next chapter.

Chapter 5

Access Point Placement Algorithm

This chapter describes the new AP placement(APP) algorithm in detail. It starts with an overview of the Indoor Radio Propagation model that is used for simulating the environment, followed by the detailed description of optimization model for the APP problem. The last section provides the overview and implementation details of an optimization algorithm *Simulated Annealing* which has been used to solve the APP problem.

5.1 Indoor Radio Propagation Model

In an indoor environment, the RSS value strongly depends upon the layout of the building, since in addition to the free space loss, electromagnetic waves are also scattered at obstacles, attenuated by penetrated walls or windows and diffracted

at edges, making the indoor propagation channel a multipath channel. Therefore, a site-specific RPS's 'Ray Tracing' propagation model is used which incorporates the characteristics of an indoor environment. In this method, a finite number of rays are launched from a transmitter position. At obstacles, these rays are divided in a reflected and possibly a penetrating part. Each ray is traced, until a given maximum of the pathloss is exceeded. Since a 'Ray Tracing' propagation model takes into account the actual nature of indoor environment, propagation results are reasonably accurate.

5.2 APP - Optimization Model

Fingerprinting localization is a pattern matching algorithm which can be divided into two phases: *Calibration(Off-line)* and *On-line*. During the calibration phase location fingerprints, i.e. RSS vectors, are collected at pre-defined locations. During the on-line phase, a measured RSS vector is compared with all the fingerprints to find the closest match. A detailed description for the fingerprinting localization is mentioned in Chapter 2.

The methodology used by fingerprinting localization suggests that it will perform well when all the fingerprints are unique. If more than one location have same fingerprint, then the localization algorithm will not be able to distinguish among those locations. Hence a suitable methodology is desired to maximize the total number of unique fingerprints so that maximum location accuracy can be achieved. However, RSS vector or fingerprint depends on both the number and the placement of APs. The RSS vector length increases with the increase in number of APs which in turn increases the probability of obtaining unique fingerprints. But

if more and more APs are deployed without considering their placement, it may not help in increasing accuracy since APs may provide symmetrical RSS values and make fingerprints ambiguous for more one than locations. Figure 5.1 presents two scenarios, using two APs. In the first scenario, both the APs are placed such that the RSS values obtained at two different locations $L1$ and $L2$ are the same, and since both locations have same fingerprints, the localization algorithm cannot decide between $L1$ and $L2$. In the second scenario, APs are placed asymmetrically and hence provide unique fingerprints at locations $L1$ and $L2$.

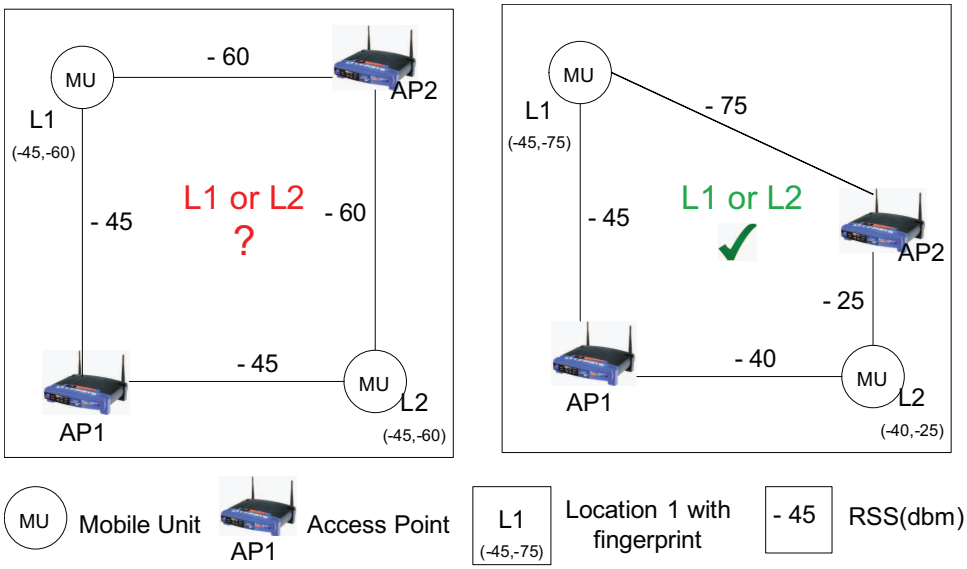


Fig. 5.1: Effect of AP Placement

Therefore, for a given number of APs, the placement should be such that the total number of unique RSS vectors, i.e. 'Location Fingerprints' are maximum. In other words, we need to minimize the total similar fingerprints over the entire RSS array. Let us define the term *Similar Fingerprint* (SF) formally, as follows.

Let $R_i = [r_{i1} \quad r_{i2} \quad .... \quad r_{iN}]$ denote a RSS vector for the i^{th} receiver location

in a 2-Dimensional space, N is the number of total APs deployed, r_{ik} is the RSS value obtained at the i^{th} receiver location from the k^{th} AP and T_r is the number of total number of receivers or sampling points. Any two 'RSS vectors' or 'fingerprints' are said to be similar if

$$R_i - R_j < T_{th} \tag{5.1}$$

where

$$T_{th} = threshold \quad value; \qquad i \in 1, 2, ...T_r; \qquad j \in 1, 2, ...T_r; \qquad i \neq j \tag{5.2}$$

The above inequality is equivalent to

$$\wedge_{k=1}^{N} (r_{ik} - r_{jk}) < T_{th} \tag{5.3}$$

where k denotes the k^{th} AP. It implies that iff all the corresponding components of two RSS vectors are within a certain threshold distance, T_{th}, with each other, then this pair of RSS vectors is called *Similar Fingerprints*. As discussed earlier, the localization algorithm will not be able to distinguish between the two receiver locations successfully if their corresponding fingerprints are not unique. Hence, we define our optimization model such that it minimizes the total number of *Similar Fingerprints*. The objective function for the optimization problem will be the total number of SFs that need to be minimized over the entire array of RSS vectors by varying the AP location. If there are L possible locations where the APs can be placed, then the total number of combinations possible for APs placement is

$$C_t = \binom{L}{N} \tag{5.4}$$

where N is the number of APs given.

Hence the optimization problem can be formulated as follows:

Minimize :

$$f = \sum_{i=1}^{T_r} \sum_{j=(i+1)}^{T_r} numSimilarFP \tag{5.5}$$

where $numSimilarFP$ = Total number of Similar Fingerprints

Subject to:

$$i \in \{1, 2, ...T_r\}; \qquad j \in \{2, 3...T_r\}; \qquad i \neq j \tag{5.6}$$

Above optimization problem is a highly computationally expensive *Combinatorial Problem* and we have used ray tracing indoor propagation model, which itself is very computationally expensive; this makes the problem even worse. The solution time grows exponentially if we increase the number of APs N and/or the total number of places L where APs can be placed. Simple Brute Force Search approach requires a very large solution time, rendering the solution practically impossible, hence, we considered a heuristic optimization algorithm *Simulated Annealing*(SA) to solve it. SA is chosen over a *Gradient Search* method as the former avoids being stuck in a local minima unlike the latter. The next section describes the SA algorithm in detail along with its application to the APP problem.

5.3 Simulated Annealing Optimization

SA [21] is a heuristic optimization technique that mirrors the annealing process in metallurgy mathematically. During the metallurgical annealing process, a substance is first heated till its melting point and then cooled down slowly in a con-

trolled manner until it solidifies again. The heating process allows the substance's atoms to move from their initial position, which may be at a local minimum energy state, and to wander randomly. Controlled cooling thereafter allows the substance to solidify into the optimum state of minimum energy.

By analogy with the physical process, the SA algorithm starts with an initial state of high temperature; every state having an associated cost, i.e. the value of objective function. From this starting point, random changes are made in the system parameters which provide a new state. These random moves are allowed over a certain region which depends on the current temperature and shrinks as the temperature decreases. The acceptance of this new state depends on the current temperature and the difference of cost values of the new state and the current state. If the new state cost is less than the current state cost, i.e the difference of the costs is negative, then the new move is always accepted and is called a *downhill* move. But if the new cost is higher than the current cost and the difference is positive, then the new state is accepted with a probability

$$P_a = min[1, exp(-k\Delta C/T)] \tag{5.7}$$

where $\Delta C = (newCost - currentCost)$, k is a constant whose value depends on the standard deviation of ΔC and T is the temperature. This acceptance criteria is called the *Metropolis Algorithm* and the move is called an *uphill* move.

The temperature decreases gradually as the process progresses until it reaches to zero or a pre-defined threshold value. (5.7) implies that the probability of accepting an uphill move is directly dependent on the 'temperature' and inversely dependent on the 'cost value difference'. Consequently, at high temperature, the

algorithm may wander wildly and accept the uphill moves even when the cost value difference is high but as the temperature decreases, uphill moves are accepted only if the cost value difference is low. Hence, SA avoids being stuck into a local minima because of these occasional uphill moves. Since the probability of accepting an uphill move is higher at high temperature, SA jumps out of local minima easily at initial states.

5.3.1 SA Implementation for APP Problem

To implement SA for any problem, the following parameters need to be defined:

- Objective function which needs to be minimized
- Initial Temperature for the process
- Cooling schedule to lower the temperature as process progresses
- Equilibrium condition at every temperature level
- Stopping criteria

The objective function for an APP problem is the number of 'Similar Fingerprints' that needs to be minimized. We define the system state as 'APs location', and the temperature as 'Grid Size or Area' in which the random moves are allowed. At the start of the algorithm, APs are allowed to move randomly in a large area which decreases as the algorithm progresses. The cooling schedule is shown in Figure 5.2. Five temperature levels [T_0 T_1 T_2 T_3 T_4] are defined which corresponds to five different radii. Temperature T_0 is the highest and T_4 is the lowest. Since a rectangular grid is used, temperature levels correspond to the varying num-

ber of neighbours. As shown in Figure 5.2, there are 24, 20, 12, 8 and 4 possible neighbours for an AP for temperature T_0, T_1, T_2, T_3 *and* T_4 respectively.

A fixed equilibrium criteria is used which implies that at every temperature level, a fixed number of iterations are allowed. The optimization algorithm stops when the total number of fixed iterations are completed at the lowest temperature. The exact values of these parameters will be provided in Chapter 6. A complete flowchart for SA optimization for the APP problem is shown in Figure 5.3.

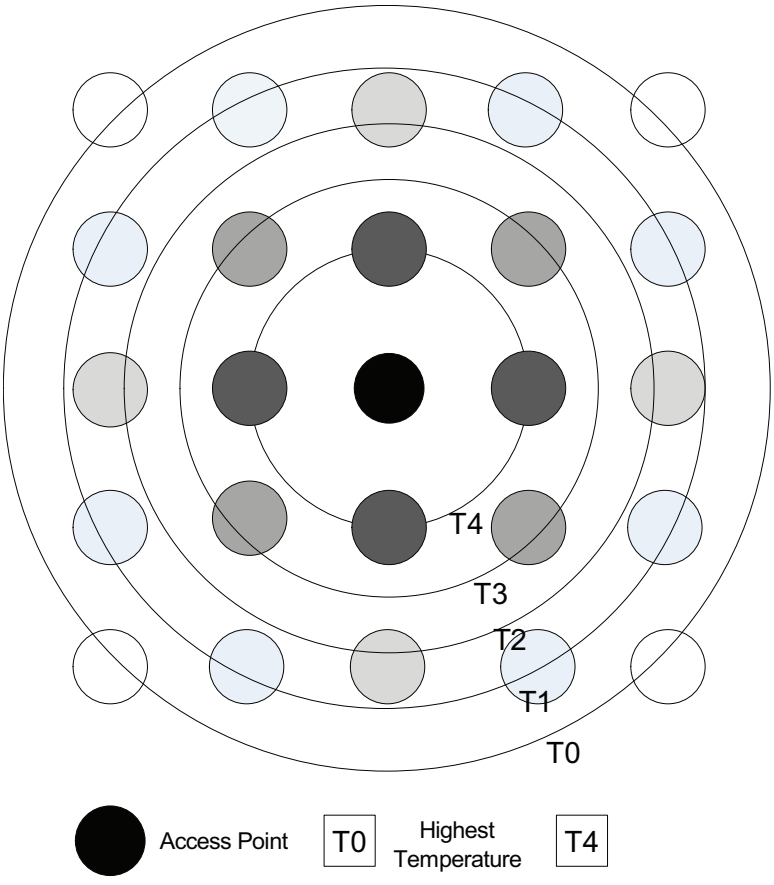


Fig. 5.2: Cooling Schedule

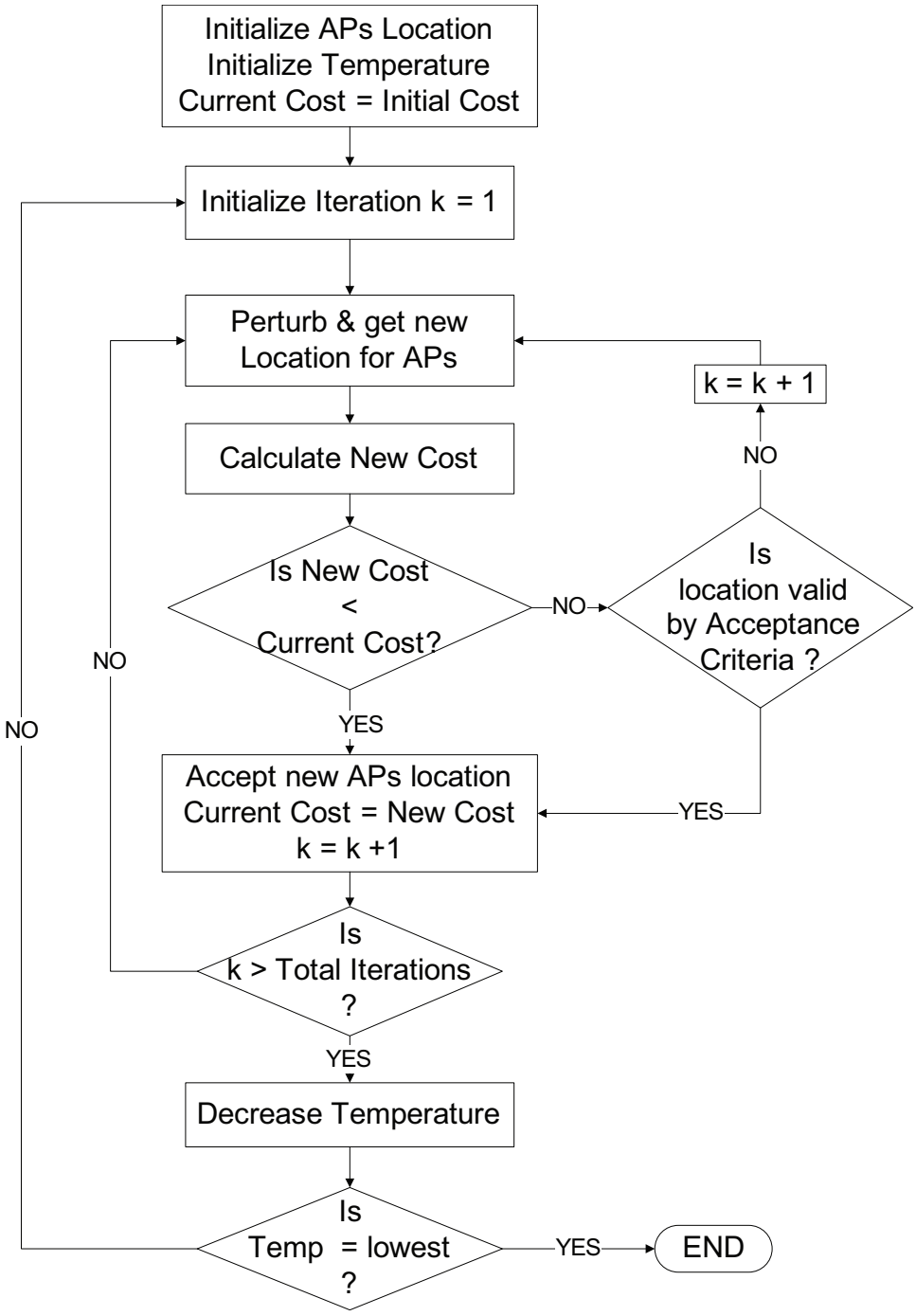


Fig. 5.3: Simulated Annealing

Chapter 6

Simulation Results

This chapter describes the simulation platform and the tools that have been used to quantify the performance of the proposed algorithm for AP placement. The first section of the chapter focuses on the simulation platform and the methodology that has been used, while in the second section, simulation results are presented in detail. Finally APs are placed as per the output of the proposed algorithm and error performance of a kNN [22] based indoor localization method is analyzed. To compare the performance of the proposed algorithm, we have also placed APs using 1.) Manual heuristic placement and 2.) Euclidean distance maximization algorithm and measured the error performance in each case for the same localization algorithm.

6.1 Simulation Setup

Simulation experiments have been conducted for the Communications Lab at NUS campus and the proposed algorithm has been tested using the following two

approaches:

- Brute force search approach, where all the combinations of APs have been tested to find the one which provides the minimum cost function, i.e. the one having minimum number of Similar Fingerprints.
- Using Simulated Annealing optimization which requires less computation time but does not guarantee to provide the optimal solution.

The following three tools are used to set up the simulation platform:

1. The Actix's RadioWave Propagation Simulator (RPS) [23] has been used for the simulation of a Ray tracing propagation model. The RPS takes a floor map as input; a 3D model for a building can be created using the integrated 'Environment Editor'. A 3D building can also be created without a floormap but it is recommended to use a map for accurate measurements and ease of drawing. Once the 3D environment is ready, simulation parameters can be set for the transmitter, receiver and propagation model. This tool provides the output, i.e Channel Impulse Response in both textual and visual form.
2. MATLAB scripting is used to analyze the simulation results. The proposed algorithm is implemented in the MATLAB to find the AP combination which provides the minimum number of Similar Fingerprints.
3. A free scripting language tool 'AutoIt' [24] has been used to automate the entire process. AutoIt has a BASIC like structure and can imitate mouse and keyboard actions for a Windows Desktop environment.

The area of the Communications Lab is 195 m^2 which is partitioned in a grid of 3 m x 3.5 m for placing the APs(transmitters), hence, there are total of 18 possible

locations where an AP can be placed. For the receiver, the grid size is maintained at 0.5 m x 0.5 m. A comprehensive list for the simulation parameters is provided in Tables 6.1, 6.2 and 6.3. The simulation parameters are kept the same for both the brute force search and the simulated annealing optimization approaches. All the experiments are conducted on a Windows Vista machine with 2.0 GHz, Intel Centrino Processor with 2GB RAM.

Table 6.1: Simulation Parameters for Access Points

Parameter	Value
Transmit Power	20 dbm
Carrier Frequency	2.4 GHz
Antenna Type	Isotropic
Antenna Height	1.5 m
Transmitter Grid Size	$3m$ x $3.5m$
Total Possible Locations for AP(L)	18

Table 6.2: Simulation Parameters for Receiver

Parameter	Value
Antenna Type	Isotropic
Receiver Grid Size	$0.5m$ x $0.5m$
Total No of Receivers	770

Table 6.3: Simulation Parameters for Propagation Model

Propagation Phenomena	Limit on No of rays
No of Reflections	Unlimited
No of Penetrations	Unlimited
No of Diffractions	Unlimited

The subsequent sections will provide the simulation workflow for the two approaches.

6.1.1 Simulation Setup - Brute Force Search

A general overview of the simulation platform using the brute force approach is presented in Figure 6.1. For this approach, the MATLAB scripting module runs independently of AutoIT. The AutoIT script performs the following tasks:

Step 1: Launch the RPS application and open the 3D building model

Step 2: Set the AP location with the RPS tool using Transmitter Settings dialogue box

Step 3: Run the simulation and wait until it finishes

Step 4: Save the 'Received Power' file on local disk for the current locations of APs

Step 5: Repeat from step 2 until all the combinations of AP locations have been tested

Once the simulation process is finished and all the result files are saved on a local disk, the MATLAB script is used to analyze the simulation results. The proposed algorithm is run to find the combination of AP locations which provides the minimum number of Similar Fingerprints.

6.1.2 Simulation Setup - Simulated Annealing Optimization

A general overview of the simulation platform for SA Optimization approach is presented in Figure 6.2. For SA optimization, the cost function(number of Similar

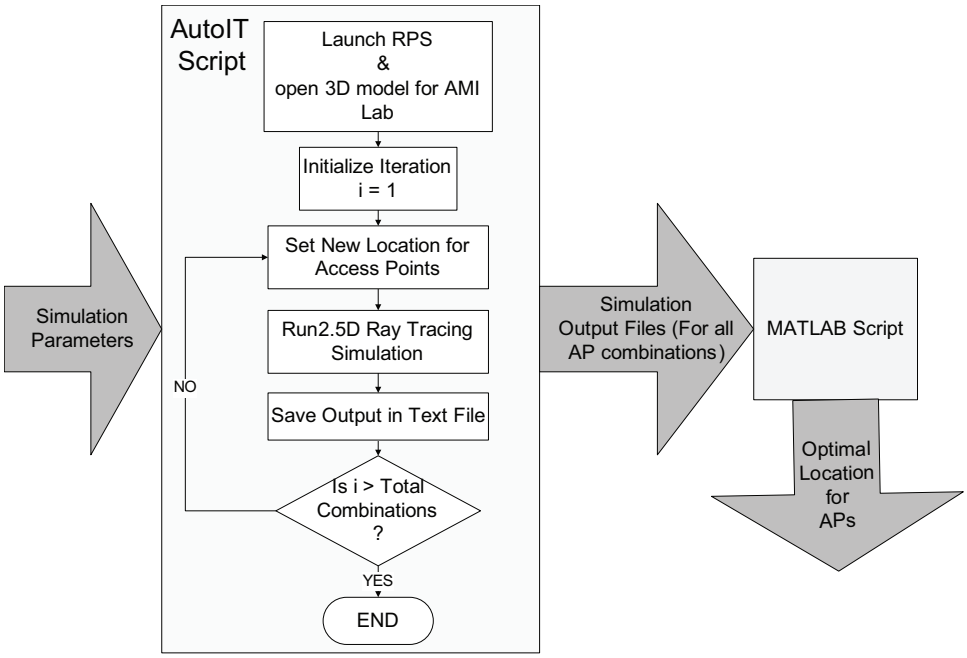


Fig. 6.1: Simulation Workflow: Brute Force Search

Fingerprints) needs to be calculated at every iteration, i.e for every combination of AP locations. After obtaining the cost function value, the procedure is repeated with a new set of AP locations. Hence it is required to integrate the MATLAB module along with the RPS using an AutoIT script. Here the AutoIT script performs the following tasks:

Step 1: Launch the RPS application and open the 3D building model

Step 2: Set the initial AP locations with the RPS tool using Transmitter Settings dialogue box

Step 3: Run the simulation and wait until it finishes

Step 4: Save the 'Received Power' file on local disk for the current locations of APs

Step 5: Run the MATLAB script to calculate the number of Similar Fingerprints for the above file

Step 6: Get the next set of AP locations from the SA algorithm module; go to Step 2 if stopping criteria for the SA algorithm has not been met, else terminate the program

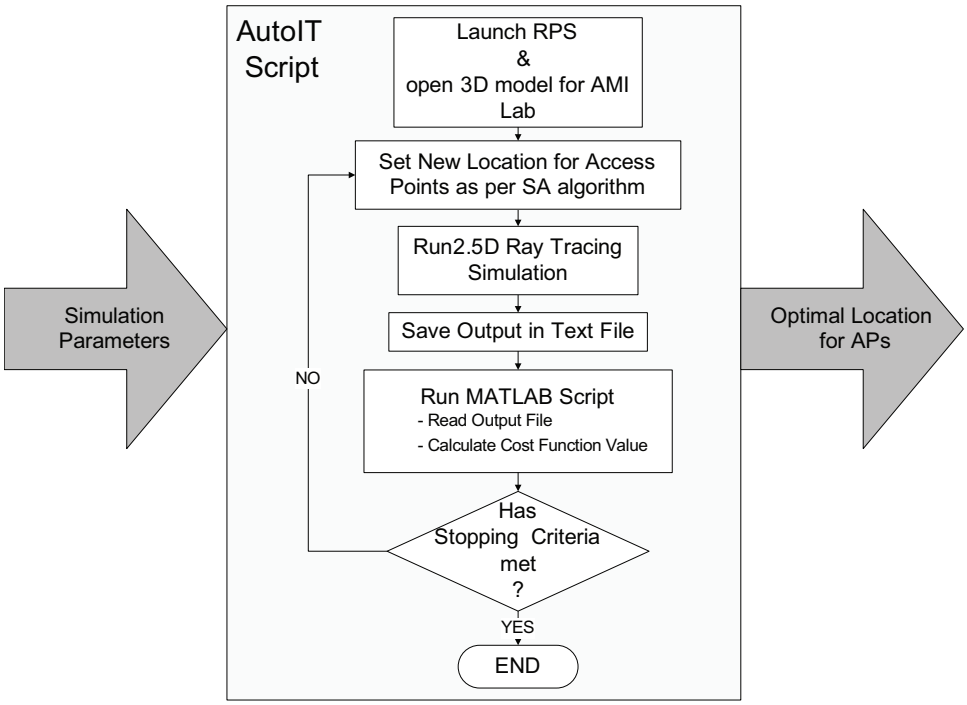


Fig. 6.2: Simulation Workflow: SA Optimization

6.2 Simulation Results and Analysis

This section presents the results obtained from the three approaches : 1.) Proposed algorithm (SF method), 2.) Manual Heuristic (MH method), and 3.) Euclidean Distance maximization algorithm (ED method) presented in [17]. The

Table 6.4: Total possible configurations for AP, given $L = 18$

No of Access Points(N)	Possible Combinations(C_t)
1	18
2	153
3	816
4	3060
5	8568

MH approach tries to place the APs at the centroid of the room. When placing more than one AP, the room is divided into sub-sections and the APs are placed at their respective centroids.

6.2.1 Brute Force Search Approach

Numeric results are computed for sets of 1, 2, 3, 4 and 5 APs. As mentioned earlier, for the simulations, a total of 18 possible locations are identified where the APs can be placed; the total possible combinations in which the APs can be placed is explained in Table 6.4. The total combinations represent the number of iterations required to obtain the optimal AP locations. As it is evident from the table, the number of iterations required to achieve the output increases exponentially with the number of APs.

Figures 6.3 to 6.7 show the AP locations obtained from the three approaches. The figure suggests that AP placement is quite unpredictable and counter-intuitive for location dependent services.

The output, i.e. AP locations, obtained from the three approaches are applied to a kNN based indoor localization method and the average error values are presented in Table 6.5. The curves of the cumulative distribution function of er-

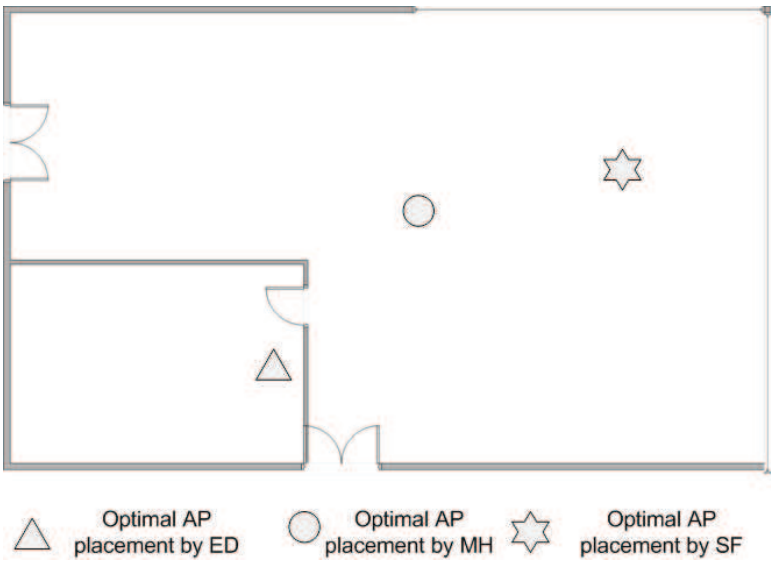


Fig. 6.3: Placement for 1 AP

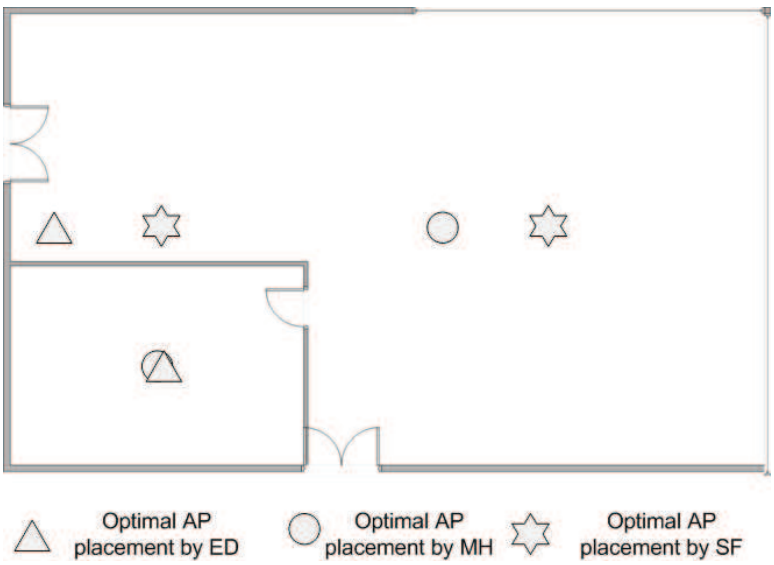


Fig. 6.4: Placement for 2 AP

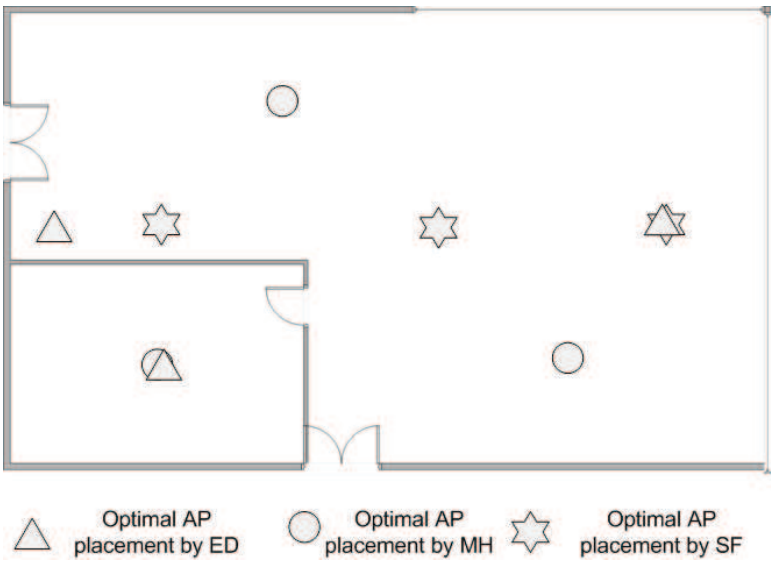


Fig. 6.5: Placement for 3 AP

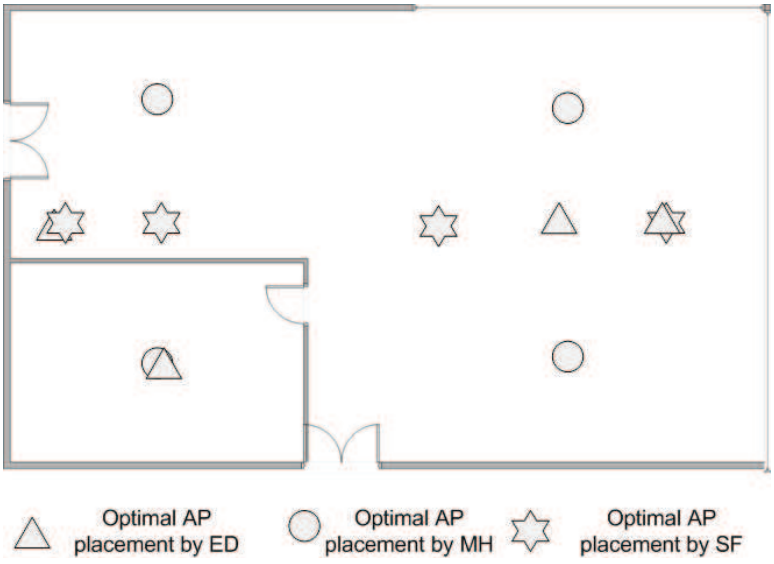


Fig. 6.6: Placement for 4 AP

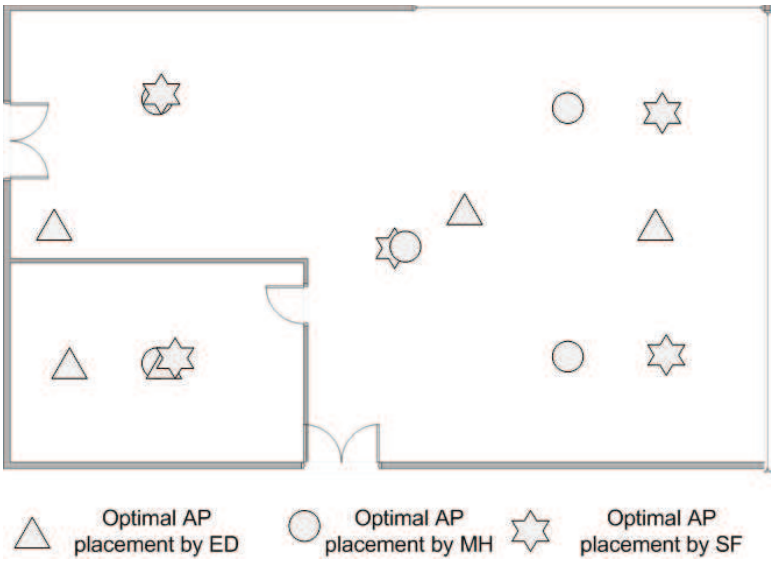


Fig. 6.7: Placement for 5 AP

ror are presented in Figures 6.8 to 6.12 for varying number of APs. The results show that the MH approach, based on centroid position placement, may work well for the communication services but it is not suitable for location dependent services. The proposed approach, i.e. the SF method also performs better than the ED maximization approach. The difference in the error performance for the three approaches is quite significant if the number of APs is less than 3.

Table 6.5: Average Localization Error

No of Access Points	MH method	ED method	SF method
1	4.46	4.08	3.57
2	3.66	2.73	2.70
3	2.48	2.30	2.18
4	2.38	2.14	1.98
5	2.24	2.37	2.00

It is also evident that increasing the number of APs does not always help

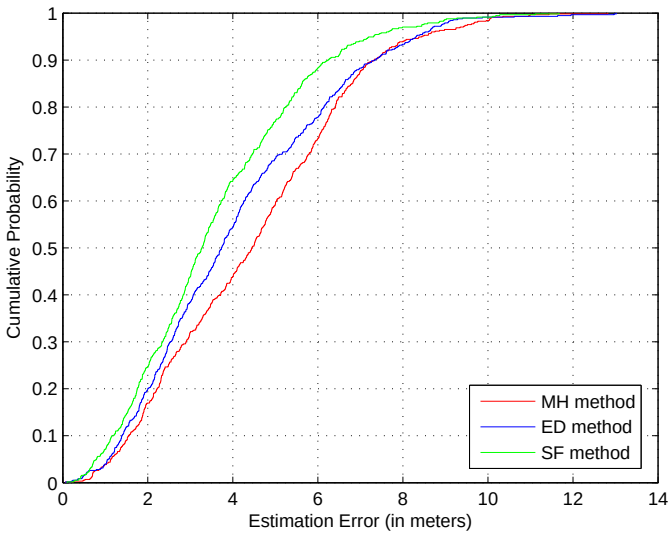


Fig. 6.8: Location estimate error using 1 AP

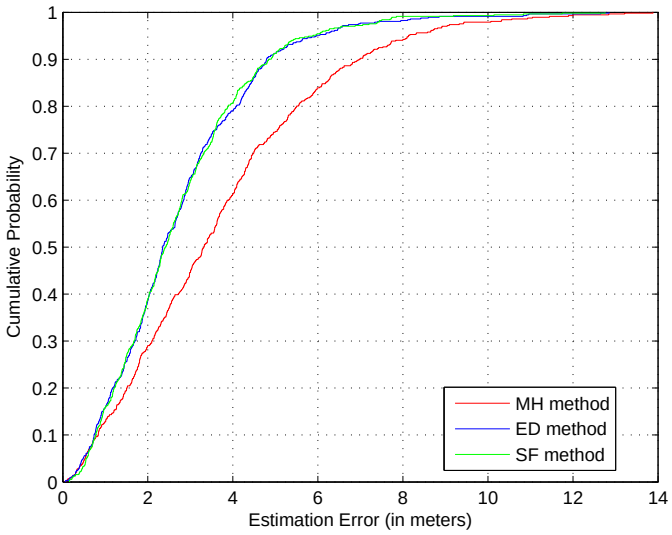


Fig. 6.9: Location estimate error using 2 APs

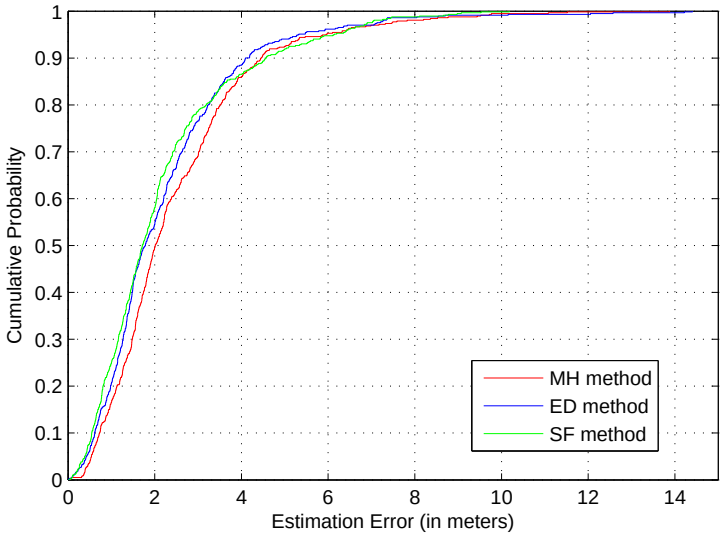


Fig. 6.10: Location estimate error using 3 APs

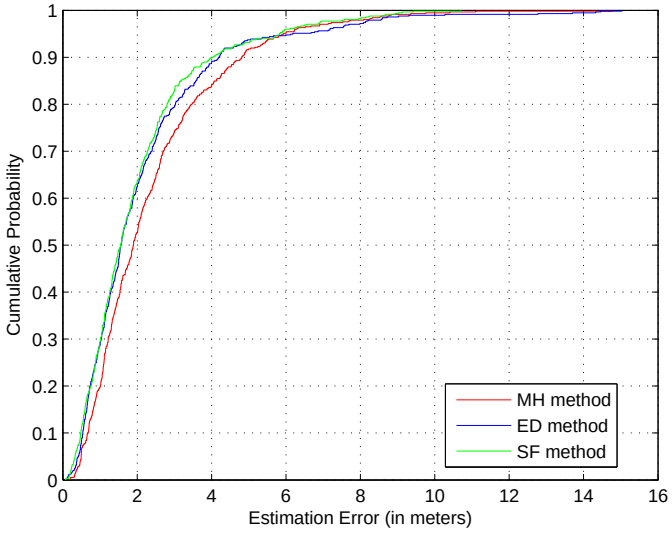


Fig. 6.11: Location estimate error using 4 APs

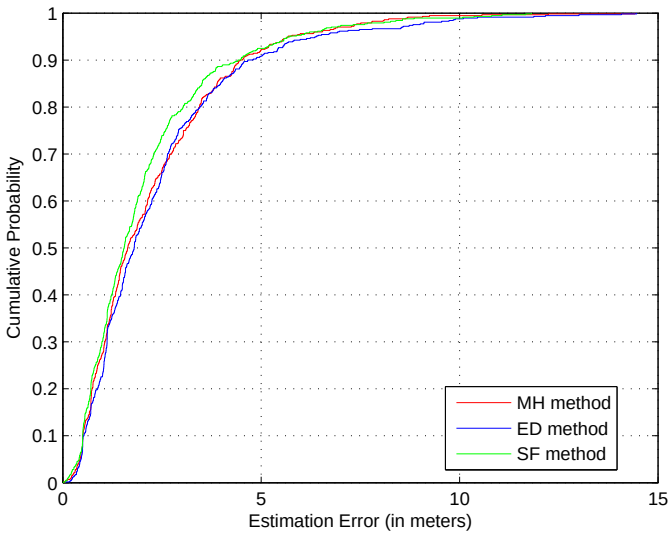


Fig. 6.12: Location estimate error using 5 APs

to improve the accuracy if their placements are not appropriate. The SF method provides better accuracy using 1 AP than the MH approach using 2 APs. However, even with the appropriate placements, if the number of APs is increased beyond a certain limit, it does not decrease the location estimate error. This fact can be further verified from Figure 6.13 which compares the three approaches on the basis of $90th$ percentile of error. The improvement in the location accuracy is very significant when the number of APs is increased from 1 to 3. The accuracy improves further for MH and SF based methods when a fourth AP is added, however, beyond that it either remains same or gets worse. This is because at any receiver location, the RSS values does not remain constant over the time, even if all the parameters for the transmitters and the receivers remain the same. If at a particular receiver location, the RSS vector heard during the calibration phase is significantly different from the one heard during the online phase, then the localization algorithm will

not be able to predict the correct location. Hence there exists a lower bound on the performance of Wi-Fi based localization methods.

However, the performance of fingerprinting localization depends on several other factors as well, e.g. Number of Training Points, Number of Testing Points, Grid Spacing used to collect the training samples, Number of RSS samples collected at each testing and training point, etc. By reducing the grid spacing and increasing the number of testing points, the lower bound on the error performance can be lowered further.

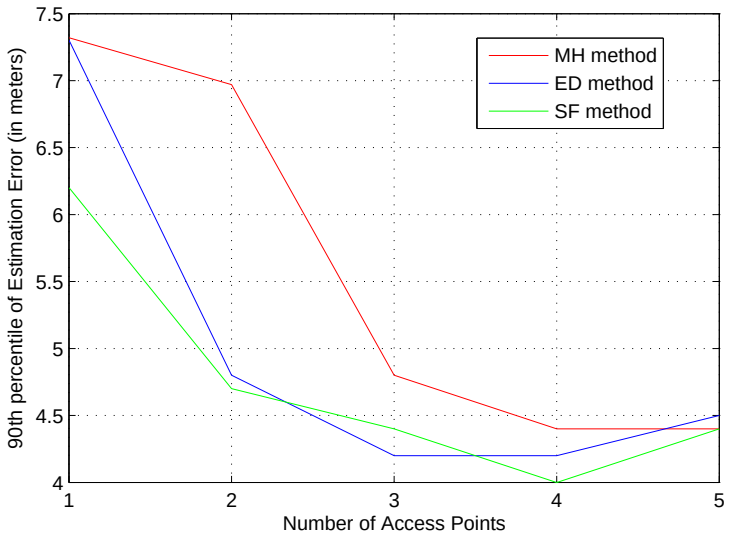


Fig. 6.13: 90th percentile of error

Figure 6.14 compares the cumulative probability functions for the three approaches when location estimation error is under 3 meters. This graph also suggests that adding more than four APs does not improve the accuracy any further.

We have also compared the computation time required to run the SF algorithm and the ED maximization approach. The computation time includes only

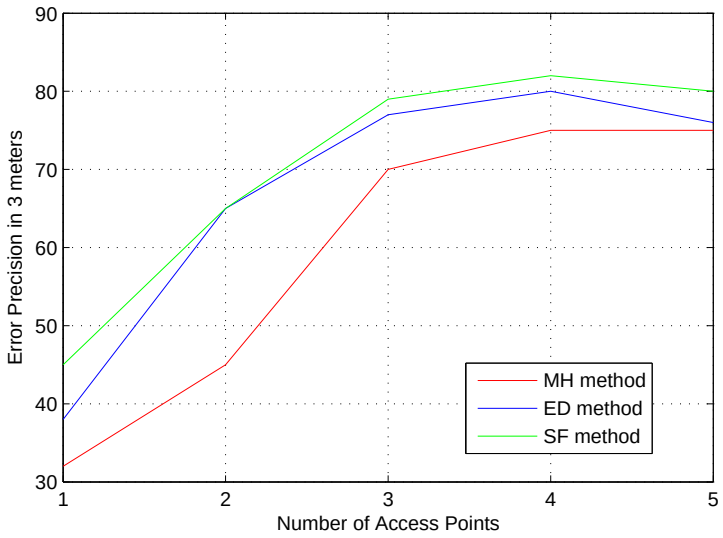


Fig. 6.14: Error precision within 3 meters for varying number of access points

the MATLAB analysis time, for the same number of AP combinations the RPS simulation time will be the same. The results are shown in Figure 6.15 which shows that if the number of APs has increased beyond 3, the ED method requires much higher CPU time than the SF method. The computation complexity of the ED method is higher as it requires to calculate the Euclidean distance which involves several multiplication and squaring operations. On the contrary, the SF method requires only a comparison and an AND operation.

6.2.2 Simulated Annealing Optimization

The SA algorithm has been applied to minimize the optimization function, i.e. the number of Similar Fingerprints for sets of 2, 3, 4 and 5 APs. The initial temperature is obtained for a grid size of 9.3 m which provides 24 possible AP

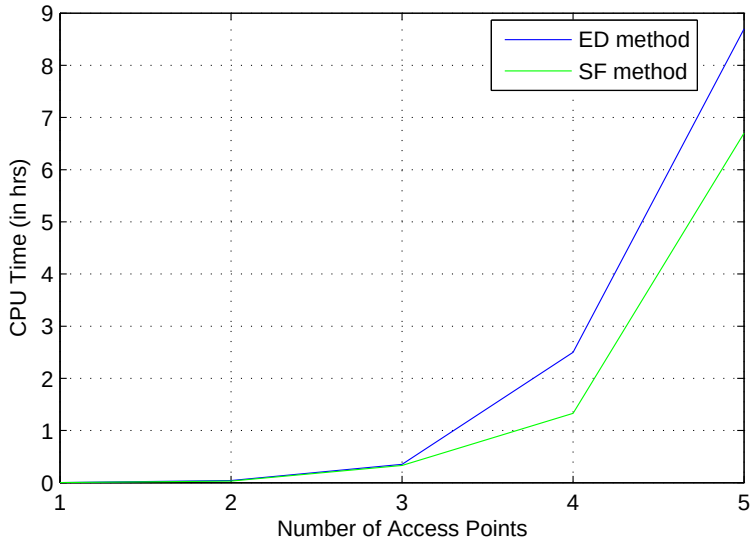


Fig. 6.15: Comparison of CPU time required for ED and SF methods

locations. The cooling schedule is set as

$$T_k = 0.8T_{k-1} \tag{6.1}$$

The total temperature levels are set to five. The number of iterations allowed at each temperature depends on the number of APs and mentioned in Table 6.6. These values are obtained after conducting several initial experiments.

Table 6.6: Parameters for Simulated Annealing

No of AP	Iterations at each Temp
2	10
3	50
4	100
5	200

The SA algorithm also accepts a new uphill move with a probability given as

$$P_a = min[1, exp(-k\Delta C/T)] \tag{6.2}$$

where $\Delta C = (newCost - currentCost)$, k is a constant whose value depends on the standard deviation of ΔC and T is the temperature.

To estimate the value of constant k, the SA algorithm is run for 500 iterations and the standard deviation for the cost value difference(ΔC) is calculated. Based on that, the value of k is set to 0.5.

The output value obtained from the SA algorithm depends on the initial placements; hence we performed each experiment for 5 times for the same number of APs but with different initial placements. The algorithm does not converge to the global minima everytime, but nevertheless, the minima obtained from the SA algorithm is very close to the global minima. Figure 6.16 shows the global minimum value for various number of APs. It also displays the standard deviation for the minimum values as error bars, obtained using the SA algorithm.

Figures 6.17 to 6.20 represent the convergence of the cost value to the global minima for various number of APs. The figures suggests that the SA algorithm does not get stuck in a local minima due to the occasional uphill moves. However, the parameter k should be set very carefully; if it is too high, then the algorithm rejects most of the uphill moves and gets stuck in a local minima and if it is too low, it keeps on wandering randomly.

Table 6.7 compares the number of iterations required to achieve global minima using the Brute Force search and the SA algorithm. The SA algorithm reduces the number of iterations quite significantly and hence, this is very useful especially

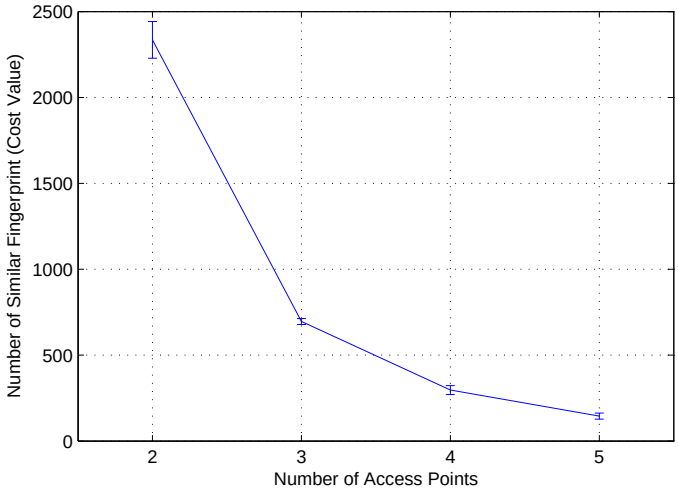


Fig. 6.16: Minimum achieved using SA

when the number of APs is high.

Table 6.7: Comparison between Brute Force Search and SA algorithm

No of APs	Iterations-Brute Force	Iterations-SA
2	153	50
3	816	250
4	3060	500
5	8568	1000

The above results suggest that the SA algorithm can be applied to the APP problem successfully; the number of iterations required to achieve the global minima reduces quite significantly without much compromise on the optimal value.

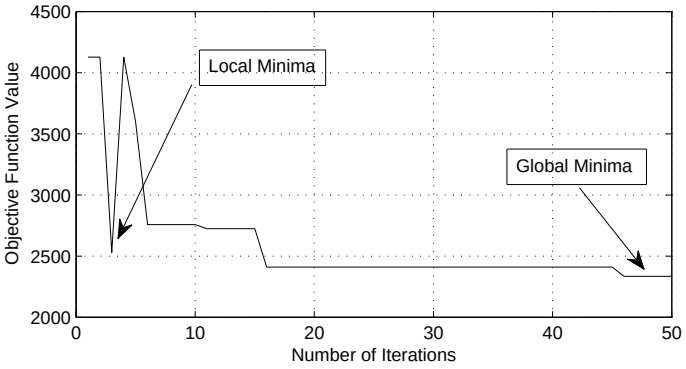


Fig. 6.17: Convergence to Global Minima with 2 AP

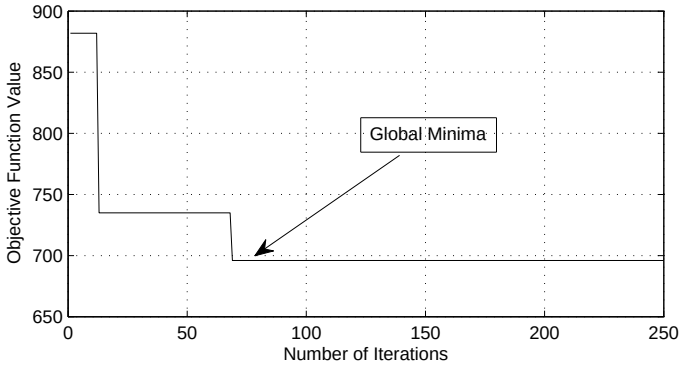


Fig. 6.18: Convergence to Global Minima with 3 AP

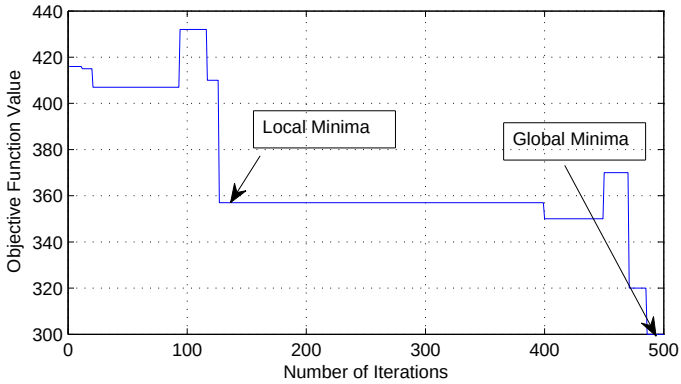


Fig. 6.19: Convergence to Global Minima with 4 AP

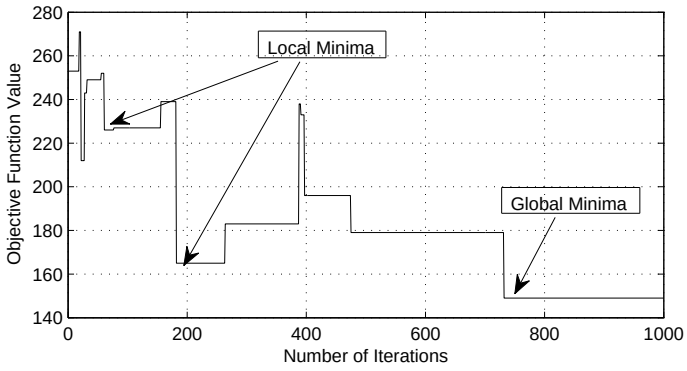


Fig. 6.20: Convergence to Global Minima with 5 AP

Chapter 7

Conclusion and Future Directions

In the first part of this thesis, a visualization platform for the Communications Lab in NUS has been set up using the WL platform. A new interface has been created on WL which integrates it with the Ekahau localization system such that the real world location information is transferred into the mirror world. Avatars are also controlled by the inputs provided by the Ekahau system.

Though WL is an open-source toolkit and hence useful for the ease of development, but its inability to support sophisticated and high end 3D graphics is a severe hindrance. Art import and creation process is still very primitive; WL does not support real time rendering of shading effects. However, the new release, i.e. WL - 0.4 supports a new input format 'Collada' which may improve the visual efficiency of WL.

In the mirror world, instead of walking smoothly, avatars abruptly move from one location to another. This is due to the limitations of the Ekahau system. The Ekahau system may be suitable for locating static objects but it is not able to

track pedestrian movement accurately. After conducting various experiments, we have realized that it takes at least 5 seconds to update a new location, accurately. The delay of 5 sec is not acceptable for pedestrian tracking as the average walking speed of humans is around 1 m/s. Hence, even the prediction algorithm cannot make a reasonable guess for the new location.

In the second part, a novel optimization model for the APP problem has been developed which minimizes the total number of similar fingerprints; its accuracy has been verified by applying its output to a kNN based localization algorithm and the error has been analyzed. The performance of the proposed SF method has also been compared with an existing ED approach and a manual MH approach. The results have shown significant improvement over the MH approach. Though in terms of location accuracy, the SF method has only marginal advantage over the ED method; but it terms of computational complexity, it is much simpler than the ED method, especially when the number of APs increases beyond 3. The SA algorithm has been applied to narrow down the number of iterations required to achieve the optimal output. Results have shown that the SA algorithm reduces the number of iterations drastically. For example, for 5 APs, the SA algorithm requires only 1000 iterations to achieve global minima while the Brute Force search requires 8568 iterations. It has been shown that the SA algorithm has the capability to come out of a local minima easily, however, the simulation parameters need to be defined appropriately and according to the problem at hand. There is no general rule for defining all the parameters.

As the number of APs are increased, location accuracy should always improve as more number of APs will provide more number of unique fingerprints. However, it has been observed that if the number of APs is increased beyond a certain limit,

location accuracy does not improve further. This is because at any receiver location, the RSS values does not remain constant over the time, even if all the parameters for transmitters and receivers remain the same. If at a particular receiver location, the RSS vector heard during the calibration phase is significantly different from the one heard during the online phase, then the localization algorithm will not be able to predict the correct location. Hence there exists a lower bound on the performance of Wi-Fi based localization methods. However, this lower bound can be lowered further by increasing the number of training samples or by reducing the grid spacing.

7.1 Future Directions

We have not considered the effect of furniture and human beings in the simulation model of the Communications Lab, though the effect of walls, floor, ceiling and windows have been incorporated. The RPS simulator allows to create the furniture and persons in the simulation model, but the simulation for a ray-tracing propagation model, with various obstacles placed inside the building, requires a very long time. The simulation experiments need to be run on the machines having very high processing power with parallel processors so that the results can be obtained in a reasonable amount of time after incorporating the effects of furniture and persons.

For the proposed APP algorithm, we have assumed the RSS values as constant, i.e. the dynamic nature of the indoor environment has not been incorporated. In real scenarios, the indoor environment changes constantly because of the moving persons, opening or closing of doors, etc. and hence, even if the transmitters and

the receivers parameters and their locations remain the same, RSS values keep on changing. The fluctuations in the RSS values is also the main reason due to which the location error does not decrease beyond the lower bound even after increasing the number of APs. Hence, this work can be extended to include the time-varying statistical nature of RSS values. More precisely, the distribution of RSS values over a certain time-window can be analyzed to understand the fluctuating pattern of RSS values.

The SA algorithm has been applied to reduce the number of iterations required to achieve the final AP locations. However, the SA algorithm does not have memory and it keeps on revisiting the bad states and hence, it requires large number of iterations to find the output. To avoid this problem, another optimization technique 'Tabu Search' can be used which uses memory structures to avoid revisiting the bad states by making them 'Tabu'. However, both the SA algorithm and the Tabu search requires the fine tuning of simulation parameters and there is no general method available to set the search parameters. 'Reactive Search Optimization' can provide a solution to this problem. It includes the parameter tuning mechanism within the search algorithm itself; the parameters are adjusted by an automated feedback loop that acts according to the quality of the solutions found, the past search history and other criteria specific to the problem at hand. In future, these optimization algorithm can be applied to the APP problem to reduce the solution time further.

Due to the Ekahau system limitations, Avatars move abruptly from one location to the another in the mirror world. We can devise a smoothening algorithm considering the delay as 5 sec and accuracy as 3m for the Ekahau system. This algorithm should predict the next location for the avatar before it is available from

the Ekahau system and once the new location is available from the Ekahau system, predicted value should get updated accordingly. However, since the delay is very large for the pedestrian movement, we may need to adopt the learning approaches, i.e. the artificial intelligence approaches for the smoothening algorithm to make it work.

Bibliography

[1] SecondLife, *http://secondlife.com/*

[2] Habbo Hotel, *http://www.habbo.co.uk/*

[3] Project Wonderland, *https://lg3d-wonderland.dev.java.net/*

[4] Ekahau System *http://www.ekahau.com/*

[5] Project Darkstar, *www.projectdarkstar.com/*

[6] Java3D, *https://java3d.dev.java.net/*

[7] jVoiceBridge, *https://jvoicebridge.dev.java.net/*

[8] P. Bahl and V.N. Padmanabhan,"RADAR: an in-building RF-based user location and tracking system ,"in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE,* vol. 2, 2000, 775-784 vol.2.

[9] B. Rengarajan and G. de Veciana, "Optimizing wireless networks for heterogeneous spatial loads," in *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on,* 2008, 686-691.

[10] J.K.L. Wong et al., "Base station placement in indoor wireless systems using binary integer programming," *Communications, IEE Proceedings-* 153, no. 5 (2006): 771-778.

[11] Jane-Hwa Huang, Li-Chun Wang, and Chung-Ju Chang, "Deployment strategies of access points for outdoor wireless local area networks," in *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st,* vol. 5, 2005, 2949-2953 Vol. 5.

[12] T. Jiang and G. Zhu, "Uniform design simulated annealing for optimal access point placement of high data rate indoor wireless LAN using OFDM," in *Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on,* vol. 3, 2003, 2302-2306 vol.3.

[13] M.H. Wright, "Optimization methods for base station placement in wireless applications," in *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE,* vol. 1, 1998, 387-391 vol.1.

[14] K. Kaemarungsi, "Efficient design of indoor positioning systems based on location fingerprinting," in *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on,* vol. 1, 2005, 181-186 vol.1.

[15] M. Heidari and K. Pahlavan, "Performance evaluation of indoor geolocation systems using PROPSim hardware and ray tracing software," in *Wireless Ad-Hoc Networks, 2004 International Workshop on,* 2004, 351-355.

[16] J. Hightower, B. Schiele, and T. Strang (Eds.): LoCA 2007, LNCS 4718, pp. 1734, 2007. Springer-Verlag Berlin Heidelberg 2007

[17] Yongxiang Zhao, Huaibei Zhou, and Meifang Li, "Indoor Access Points Location Optimization Using Differential Evolution," in *Computer Science and Software Engineering, 2008 International Conference on,* vol. 1, 2008, 382-385.

[18] Roberto Battiti, Mauro Brunato, and Andrea Delai, "Optimal Wireless Access Point Placement for Location-Dependent Services," Departmental Technical Report, January 1, 2003, *http://eprints.biblio.unitn.it/archive/00000489/*

[19] Yongguang Chen and H. Kobayashi, "Signal strength based indoor geolocation," in *Communications, 2002. ICC 2002. IEEE International Conference on,* vol. 1, 2002, 436-439.

[20] V.S. Rani and S.V. Raghavan, "A greedy approach to beacon placement for localization," in *Wireless Days, 2008. WD '08. 1st IFIP,* 2008, 1-5.

[21] S. Kirkpatrick et al., "Optimization by Simulated Annealing," *SCIENCE 220* (1983): 671–680.

[22] Teemu Roos, Petri Myllymaki, Henry Tirri, Pauli Misikangas, and Juha S., "A Probabilistic Approach to WLAN User Location Estimation," in *International Journal of Wireless Information Networks,* Vol. 9, No. 3, July 2002

[23] RadioWave Propagation Simulator, *http://www.actix.com/*

[24] Scripting Tool: AutoIT, *http://www.autoitscript.com/*