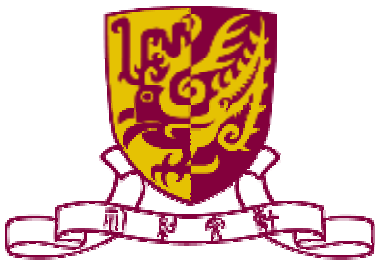


The Chinese University of Hong Kong
Department of Computer Science and Engineering

LYU0301

Location-based Services Using GSM Cell Information over Symbian OS

Supervised by: Prof. Michael Lyu Rung Tsong

Prepared by: Lee Kwok Chau CEG 01709691

Group Members: Mok Ming Fai CEG mfmok1@cse.cuhk.edu.hk
Lee Kwok Chau CEG leekc1@cse.cuhk.edu.hk

Abstract

This report describes the motivation, background information, experiments done and problem encountered by our group in participating in the final year project related to using GSM cell information for location-based services over Symbian OS.

The report would firstly introduce the Symbian OS, the major operating system used in mobile phones nowadays, in the aspects of highlighted features specific to mobile phones. Following is current role of location-based service, LBS in short, which is a highly demanding service in the fields of emergency, navigation and information distribution. The next step is to show different kinds of general positioning methods as well as current LBS technologies. These three components would play important roles in our project.

Due to the complicated network design, expensive hardware and telecom company dependency, general users would not have the initiatives to try out location-based services. This project aims to adopt simple GSM cell information positioning method which can be applied into ordinary Symbian mobile phones without extra cost and hardware upgrade.

The lack of accuracy of region-based GSM cell ID positioning would still be a great problem. The report will include issues of cell IDs collection, enhancement of the method by considering cell change events as well as experiments in both two-dimensional region (the CU campus) and one-dimensional path (railway). The sample application, MTR Traveller, which takes advantage of the proposed method in locating current station, will also be presented with principles involved, problem encountered, implementation concerns.

However, the proposed location estimation method can be applied to different kinds of LBS. Besides, developers have to spend time in investigating the underlying principles of the positioning methods, perform optimization for specific purposes, collect cell data, as well as edit content inside the application. Therefore, there is a great desire to have middleware, including a well-designed application programming interface (API) and a set of handy software tools, to assist in the above actions.

Content

Chapter 1: Introduction

1.1 Motivation

Location-based services become highly essential after the wireless network becomes slightly mature and accepted by majority. Many researchers have already figured out different kinds of issues for location-based services, including positioning methods and location information exchange protocols. It would be a realizable service sooner or later.

However, although advanced methods can meet accuracy requirement and perform quite well, they often require extra cost to existing network, special hardware (e.g. GPS) and telcom company dependent, resulting in the situation that most of the ordinary users, say mobile phone users or PDA users, cannot enjoy the location-based services. At the same time, small developers, who do not have a fund to cooperate with service providers and network operators, cannot develop their applications for location-based services, although they may have some innovative ideas.

Also, not many location-based services do require so accurate positioning, such as mobile games. There should be some suitable technologies that suit the requirement of individual applications.

This generates the idea of using GSM cell information with current data-enabled cell phone operating system which allows programs written by general developers. GSM cell information which can be accessed by all the GSM handset users without upgrading their devices and telco independent; The programmable capability of Symbian OS, one of the major operating system for mobile devices, allow developers making use of various functions in the phone, including GSM information retrieval and taking photo with camera, freely.

1.2 Project Equipment

Besides general PCs, the project involves a Symbian mobile phone, Nokia 7650, which is equipped with Symbian OS 6.1 Series 60. Besides normal telephone functions, it consists of integrated digital camera, MMS and SMS handling, WAP browsing as well as Bluetooth connectivity. These features may be useful for the project in the future. In addition, a USB Bluetooth dongle allows us to transfer the written software for the handset to execute while general PC is responsible for programming and emulation before actually testing on real mobile phone.

Chapter 2: Symbian Operation System

The word Symbian represents a software licensing company jointly owned by mobile phone and device manufacturers, including Nokia, Panasonic, Psion, Samsung Electronics Siemens and Sony Ericsson. Obviously, Symbian OS is the product of this company and it is specially designed for data-enabled 2G, 2.5G and 3G mobile phones. Originally, it was designed for any mobile devices, but it now focuses on mobile phone market because of the new affiliation of various handset manufacturers in late 1990s.

Robust multi-tasking kernel, integrated telephony support (e.g. GSM/EDGE and CDMA), communications protocols (e.g. WAP and Bluetooth), data management, advanced graphics support (support of direct-access and common hardware accelerator) and a variety of application engines enable Symbian OS to become the major operating system for current generation of mobile phones, such as Nokia 7650 / 3650, N-Gage, Sony Ericsson P800 / P900, etc.

In short, the functionalities of Symbian phone are summarized in the following diagram [1].

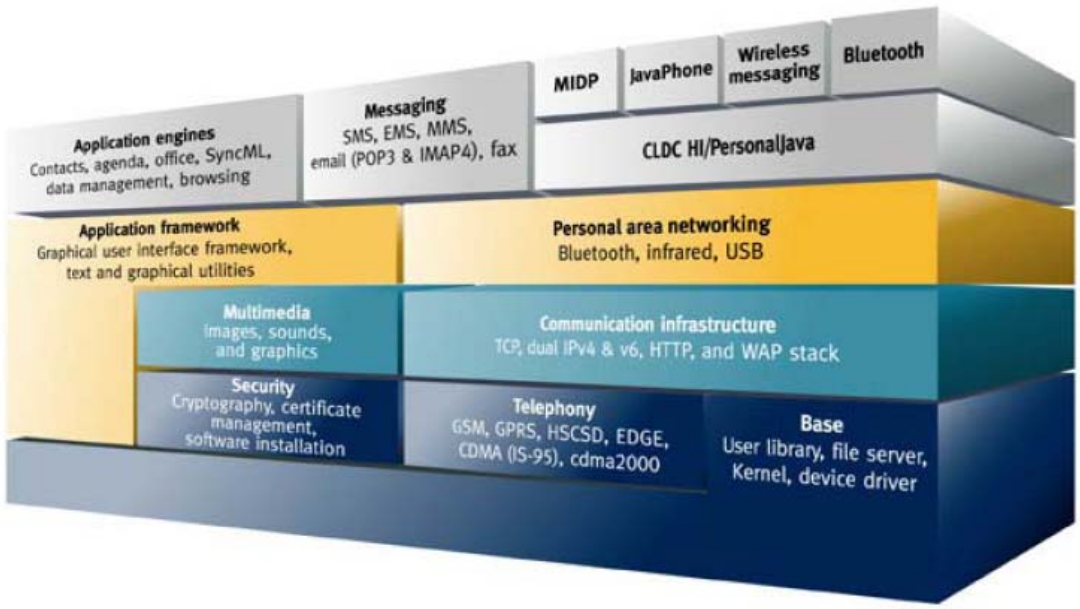


Fig 2.1: Symbian 7.0 architecture

One point should be emphasize here is that the option of Symbian OS introduces new opportunities because of its programming capability, compared to the phones released in the past where self-written programs are generally not accepted. Moreover, Symbian phone would usually contain hardware, such as digital camera, and

communication protocols, such as Bluetooth, other than telephony, so developers can interact with these special features and maximize the functionalities of their applications. This project

The main focus of this chapter is to illustrate how Symbian OS handles the basic weakness of mobile phone, such as limited memory, and how we can write our program for Symbian OS.

2.1 Development Environment

Symbian OS is a C++-based system. Therefore, to support the native OS, C++ becomes the major programming language. Besides, Java (possibly J2ME), which always provides platform-independent solution to developers, can also be a choice. However, later session within this chapter will show why programming in native code (i.e. C++) is preferred, rather than J2ME.

Each mobile phone manufacturers would provide their SDKs, including Symbian OS, emulator, documentation and different supports specific to their phones, for software development. For Nokia 7650, Series 60 platform 1.0 was downloaded for our project for Nokia Developer Home (http://www.forum.nokia.com). Developers are able to fully test and debug their programs under emulation before actually running them in mobile phones (because careless program may even crash the handset).

Generally, the development environment is under Microsoft Visual C++ with application wizard included in the SDKs. Developers may refer to the documentation or reference books to figure out the use of APIs.

However, by personal experience, Nokia SDK does not provide enough explanations to all of the APIs, especially those GUI components specifically for Nokia phone (i.e. they are not Symbian-standard components). For example, a progress bar must be in standard size (i.e. 114 or 120 pixels) with stating in the documentation; otherwise, the program would experience failure when the main dialog is opened. Therefore, the Nokia developer discussion forum is helpful in such cases for figuring out solution during programming.

The following diagram shows the process of developing a Symbian program written in C++ and running them in a real phone.

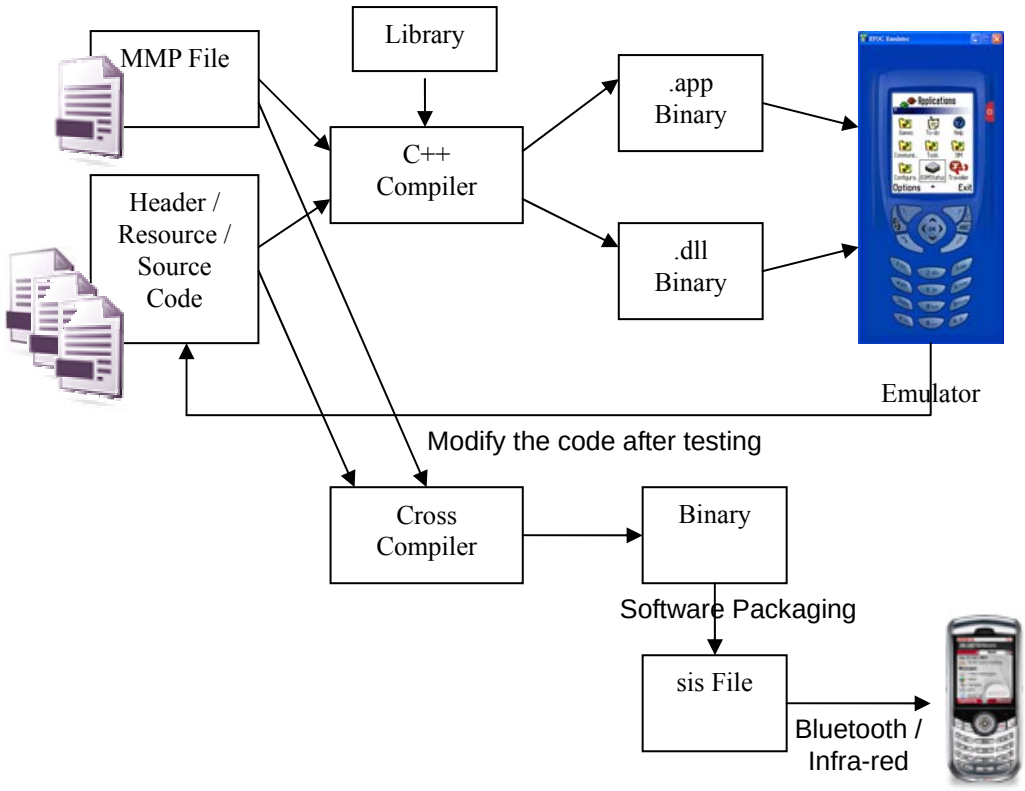


Fig 2.2: Development cycle of Symbian program

Besides source code, MMP file, which is a metadata to describe the source code and resources used (e.g. bitmaps and icons), is also supplied. Through C++ compiler, app binary (for general application) or dll binary (for building library) is then generated. Using emulator, application can be tested. After completing testing, the source code and MMP file are compiled through cross compiler, possibly ARM instruction compiler, would also give binary code. All the necessary files, including bitmaps, images, icons and data file, would be grouped together through software packaging. The resulting sis file should be transferred to actual handset using any communication technologies, like Bluetooth and infra-red.

Some features of emulator are also stated here. The emulator allows developer to know the statistics like memory used, heaps allocated so as to optimize their codes in size and memory usage. Moreover, it has a mode to generate random memory allocation faults, which would easily happen in handset with limited memory, so that resulting application can really take this case into account (compared with writing program for normal PC, it is not necessary to care memory usage).

2.2 Naming Convention in Symbian Program

Symbian OS has its own naming convention which is quite different from other

programming languages or platforms. The below briefly describes them. All variables and class names always have a letter prefix as follows:

T: Type that do not have reference to external objects (e.g. TInt, TChar) which does not have C++ constructor and destructor;
C: Class derived from CBase, where instance of C class is allocated on the heap. Two-phase construction is required (e.g. CCoeControl);
R: Resource class (e.g. client connection) (e.g RArray);
M: Interface only (i.e. purely abstract) (e.g. MDesArray);
K: Constant (e.g KPi = 3.14)
i: Member variable (e.g. iMap)
a: Argument of a method (e.g. aFileName)

The reason why the naming convention is discussed here because it can again show how Symbian handles limited resources. This naming convention would remind developers to use appropriate types whenever necessary and to deallocate unwanted resources (e.g. call `Close()` for R type objects and cleanup C type objects after use).

2.3 Special Programming Features for Mobile Phones

Symbian program has their own error handling framework and memory leak prevention and they are investigated one by one.

2.3.1 Error Handling

Traditional C++ use try-and-catch statement for handling exception. However, in Symbian, there is no concept about exception. Instead, all functions are defined as leaving functions (with L postfix, e.g. `RunL()`) or non-leaving functions (without postfix ,e.g. `Draw()`). When error occurs in leaving function, it will exit from the function and call `User::Leave()` to ensure graceful deallocation of resources. For non-leaving functions, usually library functions, they handle any error internally so that the function caller does not need to take extra consideration about using them.

In addition, instead of conventional “new” keyword to allocate memory, Symbian suggests the use of “new (ELeave)” instead because, by doing so, the system would try to allocate memory with extra error handling to prevent any system crash due to out-of-memory error.

2.3.2 Memory Leak Prevention

Usually, mobile phone would not be turned off or reset, in contrast to computers.

Therefore, if memory leak occurs, the problem would accumulate. Therefore, it is essential to have memory leak prevention. By default, in the emulator, if there is memory leak after leaving the program, it would generate kernel error, telling the developer the occurrence of memory leak. A careful developer should avoid this kernel error from happening before actually deploying the code.

Moreover, cleanup stack and two-phase construction may help in this issue. Once fatal errors occur, memory is still kept in no leakage status.

2.3.2.1 Cleanup Stack

The cleanup stack addresses the problem of cleaning up objects that have been allocated on the heap, where pointer to that piece of heap is an automated variable (i.e. declare inside the function). For example,

```
void CTest::FunctionL()
{
    CMyClass* x = new (ELeave) CMyClass;
    x->DoSomethingL();
    delete x;
}
```

The above code shows good programming practice to do allocation and deallocation in pair. However, error may occur at `DoSomethingL()` and the function `FunctionL()` would leave immediately. Therefore, the delete operation is not executed, and the heap allocated by pointer x is still occupied the memory. The code should be modified as follows to make use of cleanup stack.

```
void CTest::FunctionWithCleanupStackL()
{
    CMyClass* x = new (ELeave) CMyClass;
    CleanupStack::PushL(x);
    x->DoSomethingL();
    CleanupStack::PopAndDestroyL(x);
}
```

The highlighted code shows the use of cleanup stack. It would push the pointer x into the cleanup stack by `PushL()`. Once `DoSomethingL()` encounters failure, the cleanup stack would still hold the pointer of x and clear the heap when the program

exits. If the code runs smoothly, `PopAndDestroyL()`would pop the pointer out from the cleanup stack and deallocate it.

2.3.2.2 Two Phase Construction

In object construction, memory allocation is extremely common. In fact, only cleanup stack may have a potential problem. Consider the code again:

```
void CTest::FunctionWithCleanupStackL()
{
    CMyClass* x = new (ELeave) CMyClass;
    CleanupStack::PushL();
    x->DoSomethingL();
    CleanupStack::PopAndDestroy(x);
}
```

The cleanup stack handle memory leak properly if error occurs at DoSomethingL(), but it x is not pushed into the cleanup stack at the moment of its construction (since `PushL()` is called x's constructor). If this fails, the code would leave without having x in the cleanup stack. As a result, memory leak may still a problem.

Thus, two phase construction is invented to prevent such situation. Besides normal C++ constructors (first-phase constructor), a second-phase constructor function, `ConstructL()`, is written also for extra memory allocation. These two constructions would be wrapped up together as follows:

```
CMyClass* CMyClass::NewL()
{
    CMyClass* self = new (ELeave) CMyClass; // first phase constructor
    CleanupStack::PushL(self);              // push into cleanup stack
    self->ConstructL();                     // second phase constructor
    CleanupStack::Pop(x);                   // pop out from cleanup stack
    return self;
}
```

Usually, first phase constructor would just copying the arguments and do not perform any memory allocations of member variables inside (notice that the memory allocation of "self" has been handled by new (ELeave) statement already). After

pushing “self” into cleanup stack, code for memory allocation of member variables can be located here. It is quite clear that `ConstructL()` is clean-safe.
In addition, the first-phase constructor should be kept “private” in order to avoid developer from using traditional C++ object construction.

The resulting code would be modified like this:

```
void CTest::FunctionWithCleanupStackL()
{
    CMyClass* x = CMyClass::NewL();
    CleanupStack::PushL();
    x->DoSomethingL();
    CleanupStack::PopAndDestroy(x);
}
```

2.4 Symbian Programs Developed

Before investigating into location-base services and location estimation methods, some Symbian programs were developed for us to get familiar with Symbian program development, which greatly differs from traditional Java and C++ programming.

The first one is Robot War, which is an unfinished game around robots. In Robot War, different kinds of GUI components were used, such as progress bar, user menu, container, dialog, text and bitmap drawing, GIF image decoding, etc. This is a great progress for us to reduce development time, as we have experience, for later application.

Another one is Nokia Square. It combines the game logic and object drawing.

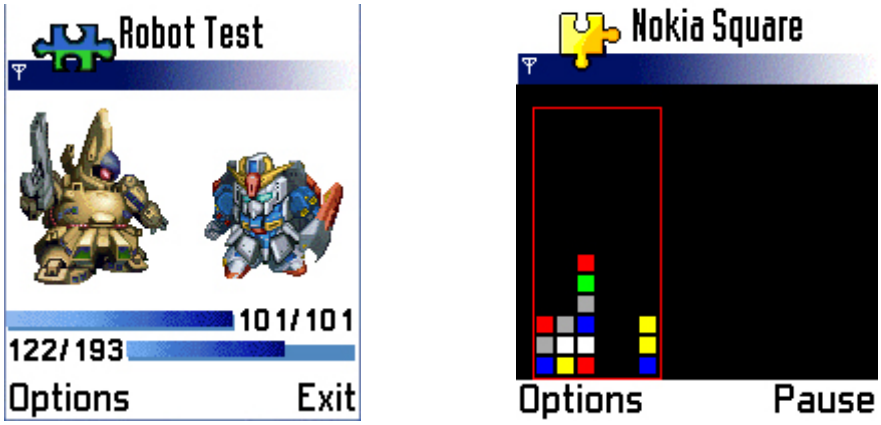


Fig 2.3: The screenshots of Robot War and Nokia Square

2.5 Why Programming in Symbian?

There is another solution, J2ME, which allows a mobile platform independent application. However, the consequence is still the performance problem and cannot access to mobile phone specific functions that J2ME APIs do not cover.

The performance issue would be a great problem for J2ME application, especially program with extensive GUI and screen drawing, such as mobile games.

However, in our project, the major concern is whether phone specific functions are accessible. As shown in the title of the project, we would like to use GSM cell information for location estimation. J2ME does not have related API to retrieve this kind of information (because not all mobile phones allow developers to do so). Although Nokia Symbian SDK hides those APIs also (which will be discussed in later chapter), we have figured out some ways to tackle this issue for GSM cell data retrieval. Therefore, at this stage, J2ME would not be in our consideration.

2.6 Conclusion

This chapter briefly introduced the features of Symbian OS. The measures to tackle limited memory problem are also emphasized here, namely new error handling framework, cleanup stack and two-phase object construction. It also showed the programs that our groups have developed.

The major reason of using Symbian program instead of other solutions is the ability to access phone-specific functions. As a result, other Symbian applications would be developed in the entire project.

The next few chapters would introduce general idea of location-based services and location estimation.

Chapter 3: General Location-Based Services

Location-based services, or LBS in short, refer to the services provided by mobile operators based on location, including user's current location and location of a target. In fact, knowledge of user's location can enable or enhance different applications, such as fleet management, object tracking and routing. For example, application may provide you information corresponding to your current location. However, LBS can be released to ordinary users until all known issues have been solved, including privacy. This chapter would give the overview of location-based services, including the classification, system architecture, service requirement and platform standardization.

3.1 Types of Location-Based Services

There are numerous of location-sensitive applications that people may think of. In technical view, all location-based services can be classified as two types, namely pull-based and push-based. These two types based on client-server model, where clients may be wireless computing devices or mobile phones, while servers refer to network operators or service providers.

Pull-based LBS describes the action that client (or client application) actively asks server for location so that it can further deal with the returned information to achieve certain goals. Positioning is a simple example of pull-based LBS. Users would continuously request for where they are in the map so that they can go for a specific places. Another example is the restaurant finder which people can find the nearest restaurant around them. Pull-based service often involves object tracking.

Push-based LBS works in the opposite way: server side would locate the position of client device through wireless connection so that server (or server application) may use this piece of information to perform some tasks. On typical example is the advertisement from mobile operator. When users travel overseas, say from Hong Kong to Macau, operator would eventually detect this event and send advertisement and information to users' cell phones via SMS, telling users the price of international calling. Usually, push-based service would be useful in location-sensitive content distribution.

3.2 Location-Based Applications

In business or user perspective, location-based applications can be classified as different types in terms of their use. The below shows several fields on which LBS is

applicable.

Emergency: The initial motivation of location-based services is to locate victim's position on emergency call in the United States. Originally, terminals of emergency centre could only show the corresponding address of home or office lines. However, there was no such support on mobile phone calls. Mobile phone positioning was highly demanded.

Information-Oriented: Like the location-sensitive advertising example stated in the "pull-based service" model, client location data is useful for network operators and content providers to deliver suitable information, such as weather of that district, surrounding traffic condition.

Navigation: Navigation is just positioning service with guidance to subscribers what way they should go. One typical example is driving navigation system which provides support to ordinary drivers. With very high accuracy, positioning devices, such as Global Positioning System (GPS), could be used on military and war.

Billing Service: Applications under this field are often adopted by service providers or resource owners. They provide some kinds of service, like car parking or public transport, where charges are different at different locations. Keeping track of user location within duration of service allows automatic calculation the overall charges. Such billing service is usually a pull-based one.

Games and Entertainment: Location information can be inserted into game logic, giving game developers high opportunities to design innovative games, especially multi-player games. Virtual game, where factors of reality are put into the virtual game world, may also be realized, including treasure hunt and role-playing game (RPG).

Others: Besides the above, LBS could be used for the various purposes. For administrative issues, managers may want to know where their employees are. People can also keep track of their valuable belongings, such as vehicle and jewelry box, by putting location-sensitive devices on them. Besides, LBS also helps network operators themselves in terms of cell planning and network optimization. They can know the distribution and usage of their customers so that they can change the base station deployment and tune the configuration, like output signal strength, in a cost-effective way.

Notice that different kinds of applications require different latency and accuracy. For emergency, positioning or navigation purposes, the accuracy and timeliness are the primary concern. On the other hand, only fair accuracy and latency is needed for content providing or entertainment. However, one of tradeoff of getting good performance is the cost. As a result, choosing a suitable positioning method or technology, which will be discussed in later chapter, for a right purpose can make a great deal.

3.3 Parties Involved in Location-Based Service

Different parties in the community play different roles, so do those in LBS world.

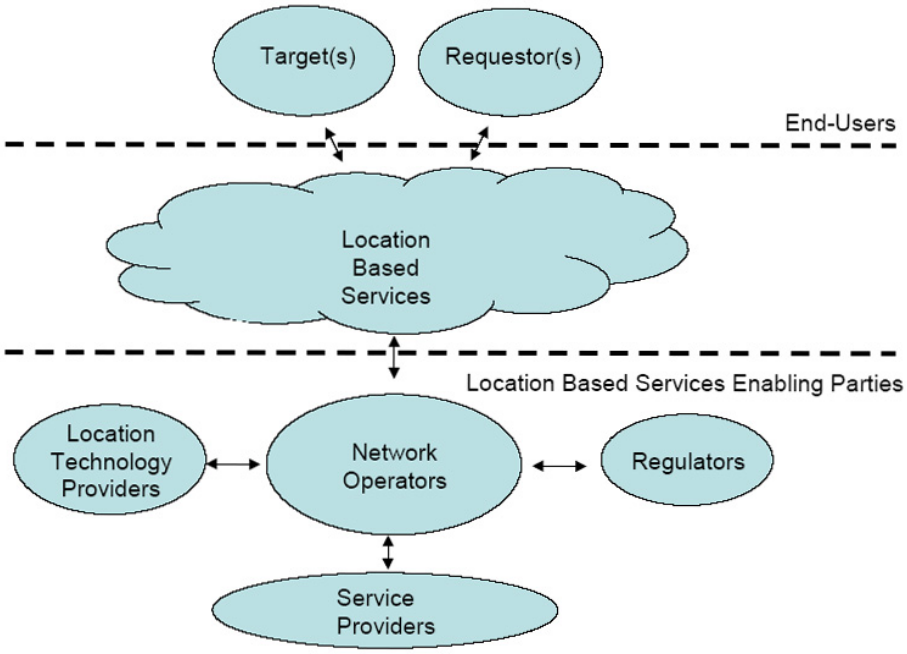


Fig 3.1: Parties involved in location-based service industry

The above diagram shows the parties in the general LBS system. Network operators, with the help of software or hardware from location technology providers, own the subscribers' location information through their base stations and wireless networks. They would be omitted if users themselves could get the position data (e.g. GPS). At the same time, other service providers may have location-related information, such as detailed map and route (spatial content), real-time traffic condition, yellow pages, etc. Therefore, enabling location-based services would require the cooperation between these two parties. Then, application developer can design their applications built on top of operators and service providers.

In the above diagram, target (i.e. the object that applications would like to keep track, such as users and vehicle) and requestor (i.e. the one who wants to obtain target's location information) are the 'clients' in pull-based model and push-based model respectively. Client should be regard all of the underlying structure as a black box and enjoy the service.

Regulator refers the group that inspects how network operators use the location information for privacy purpose.

3.4 Overall Architecture

After understanding the parties involved, the below would further show a detailed LBS system architecture. Network operators would use different measurements (e.g. TOA and E-OTD), which would be discussed shortly, to generate location information. Service providers can be map engines or content servers. Application developers can also join the system by setting up application servers or XML servers for specific purposes. These three parties together form a LBS enabling platform, which is illustrated below.

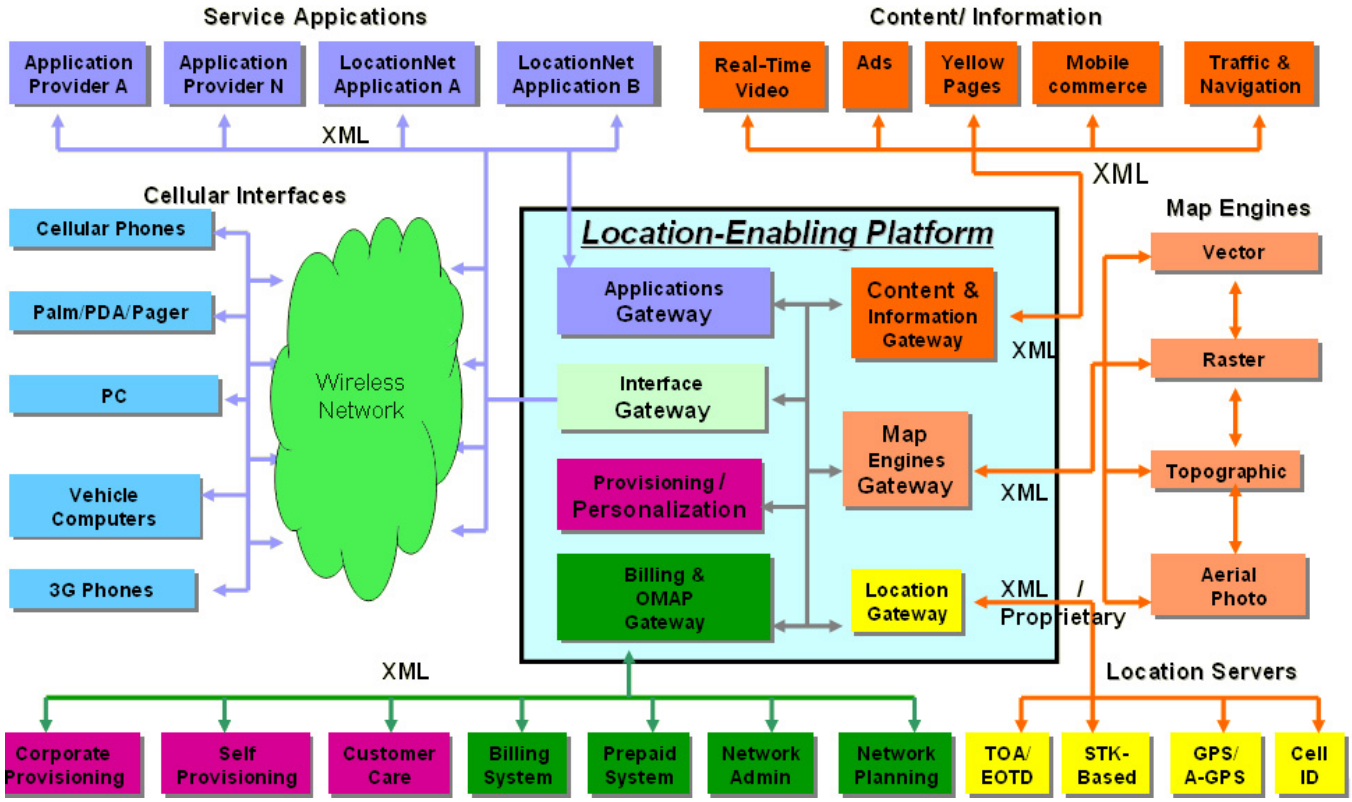


Fig 3.2: Overall architecture of a general LBS system

Users, then, can access the platform using different LBS-enabled devices (e.g.

mobile phones) through different channels (e.g. GSM telephony network).

3.5 Location-Based Service Requirements

Consider the scenario of TCP/IP, where security issue and quality of service (QoS) were not considered into account at the time it was first introduced, request of changes on protocol becomes extremely difficult as it had been already widely deployed in the whole world. Therefore, to make location-based service to be well-prepared for commercial use, the system design should be relied on certain requirements.

Flexibility for All Parties: From the above discussion, one may observe that the LBS enabling platform requires the cooperation between the three different units. Therefore, good design of protocols between parties would be highly desired in order to facilitate the development (e.g. good location API may enhance application development), especially when the system is in large scale.

Quality of Service (QoS): Many people have heard of this term, which describes the need of some guaranteed properties. In querying user's location, requestor should be allowed to specify desired accuracy or delay in order to have a particular service working properly. Emergency tracking is a typical example that requires high accuracy and minimized response time. The Federal Communications Commission (FCC) in the US proposed the rule in which network operators must automatically determine 911 callers' locations with ±125 meters accuracy in 2/3 of all cases [5]. Therefore, QoS feature should be included to fit the desired purposes and even rules.

Platform / Technology Independent: LBS environment involves different platforms, networks and positioning methods. If the information provided all of them require different standards, application developers or content providers have to construct different copies for different platforms / technologies and this would increase the overhead of development. To allow developers and users enjoy different platform, the LBS standard should be made platform / technology independent.

Roaming: Like mobile voice call, roaming should also be the standard of location-based services so that applications can still work in different countries (though the service price may have a difference). The location-based network should have cross-operator as well as cross-platform capability. As a result, service provider can also distribute the content in world-basis instead of limiting only to its country while same application can be applied for different regions.

Privacy, Security and Regulation: Another major concern of LBS is the right to know where a person is. If location information is used unethically, crimes (e.g. theft and kidnapping) would happen more easily. To safeguard the privacy of subscribers, protecting location information is vitally essential. Technically, location data should be highly secured, in terms of their storage and transfer. Besides, authentication policy should be held so that only permitted requestors can access location information. Moreover, regulating authorities should watchdog how network operators handle those data. That is why 'regulator' is also involved in LBS system.

3.6 The Need of Standardization

To achieve the above requirements, one of the solutions is to have standardized protocol between requestors and LBS enabling platform. Open Mobile Alliance (OPM) proposed an application protocol called Mobile Location Protocol (MLP). MLP defines simple and secure access layers to query location information independent of underlying communication standards, wireless network types and positioning methods.

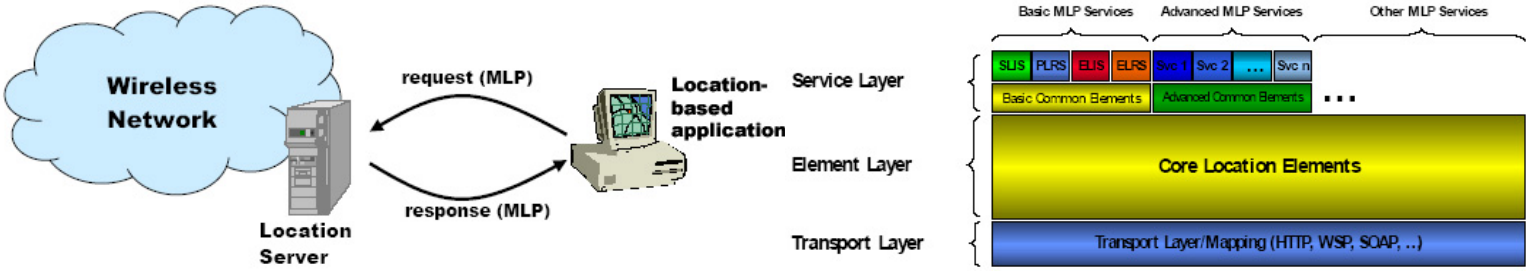


Fig 3.3 MLP request-response operation (left), MLP data format and layer definition

Other standardizations also exist in other parts. For example, JAVA platform can be used for all client devices using location-based service API provided. The communication between service providers and network operators can also be standardized with the use of XML, the format designed to encounter the exchange of a wide variety of data in the scalable internet.

3.7 Conclusion

This chapter presents the current situation of location-based services as well as a general structure of LBS system. A list of requirement of LBS is also proposed in order to meet future challenges. While different LBS applications require different accuracy, the next chapter would have a discussion on various positioning methods and technologies that gives their own advantages and disadvantages.

Chapter 4: General Positioning Methods and Current LBS Technologies

When mobile device moves inside a wireless network, like wireless LAN or GSM network, it would connect to the nearest base stations (or cells) (e.g. wireless LAN access points). The location of these base stations in the device's connection list may somehow indicate the fact that the device is under the intersection of reception regions of those base stations.

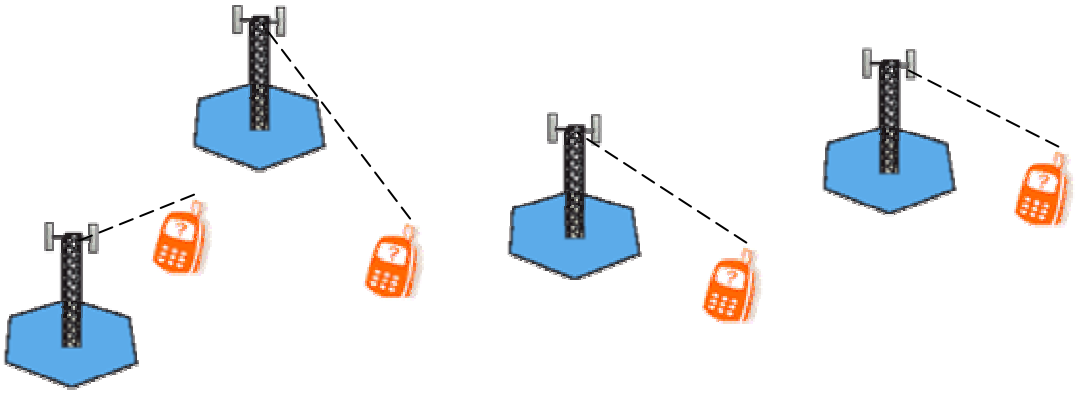


Fig 4.1: Mobile devices inside wireless networks with connections to base stations

Various location estimation methods are based on this idea. Basically, they differ from each others by, but not limited to, accuracy and number of base stations involved. Some general location estimation methods would be highlighted with also state-of-the-art technology for positioning.

4.1 Region Estimation – Pure Cell Information

Some applications may require the location with a region, instead of a point. Such method is easy to be implemented. The following covers one of the instances: using unique base station information.

Each base station has its coverage (i.e. cell size). For a mobile device, the cell that gives the strongest received signal strength would be the registered cell of the device. By knowing the current registered cell, it is sure that the device is inside that cell's coverage.

Such method makes use of the base station coverage which is generally a great circular region, which is not so attractive to application developers. However, this method is extremely simple and only one base station is required. And these are the beauties of this method. In the project, we would slightly modify the method and try

to make it practically in our application.

4.2 Point Estimation

For service that requires higher accuracy, a point is much more preferred. However, practically, it is not achievable because error must occur (that is why the word “estimation” is used). The tradeoff is that additional information is required. The below would illustrate same standard methods for point estimation.

4.2.1 Signal Attenuation

Signal attenuation method makes use of the measured signal strength to estimate the distance between base station and the mobile device. Basically, the received signal strength, p (or the ratio of received signal strength to transmitted signal strength, $\triangle P$), decreases exponentially with the increase in distance from base station, r, as shown.

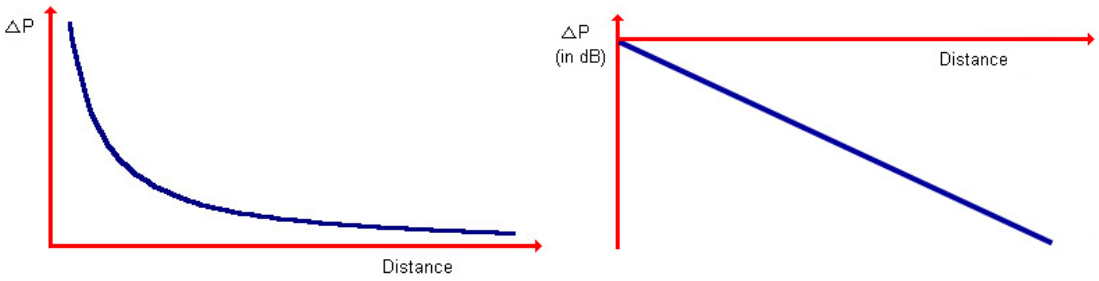


Fig 4.2: $\triangle P$ (without unit) against d (left), $\triangle P$ (in dB) against d (right)

However, signal strength itself would have different problems, like random factor, obstacle attenuation, etc., which lowers the accuracy of such method.

With the above, one may only know the mobile device is at one of the points within a circle with radius r. Therefore, more base stations are put into account in order to obtain a point instead. Such method is well-known as ‘triangulation’, or ‘trilateration’ in formal.

The position can be determined by locating the interaction the three circles generated by three base stations (as shown in Fig 4.3), where r_i is the measured distance from BS_i, and (x_i, y_i) is the position of BS_i. Thus, the position of mobile device, (x, y), can be obtained by the following formula.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} x_2 & y_2 \\ x_3 & y_3 \end{bmatrix}^{-1} \begin{bmatrix} x_2^2+y_2^2+ r_1^2-r_2^2 \\ x_3^2+y_3^2+r_1^2-r_3^2 \end{bmatrix}$$

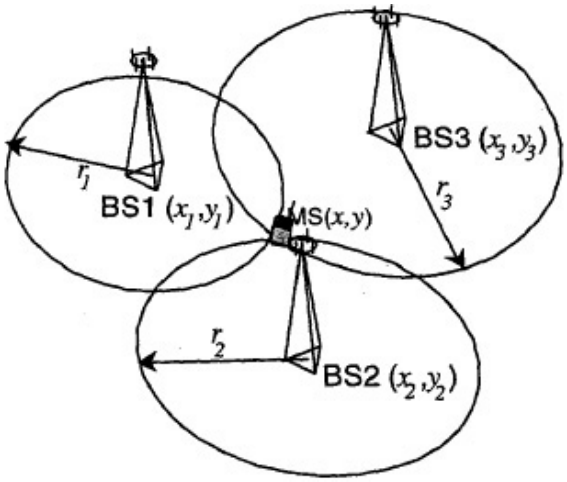


Fig 4.3: Trilateration calculation for signal attenuation and Time of Arrival (TOA)

4.2.2 Time of Arrival (TOA)

The estimation is based the round-trip time (RTT) of signal transmission to base station and back to mobile device or vice versa (i.e. initiated by base station). The distance, *r*, from base station is related to half of the RTT value. With also trilateration, the position (x, y) can be obtained by the above formula.

The advantage of using TOA is that it does not any support network operators. However, as it requires system-wide synchronization for TOA to occur (notice that base stations only listen to signals at allocated time slots), accurate timing cannot easily achieved. TOA clock inaccuracy of 1μs would introduce an error of ±300m [8].

4.2.3 Angle of Arrival (AOA)

In the above two methods, at least 3 base stations are required and more would be needed to increase accuracy. However, in rural area where base station density is small, the performance of those methods is suffered. To tackle this weakness, the position of mobile device is estimated by angle of arrival.

To use such method, base stations have to be equipped with sophisticated antenna arrays. The angle of the incoming signal can be known due to electronic steering of the arrays. Thus, the interaction between at least two lines of bearings (with angle Θ_i) from base stations can then locate the position of mobile device (see Fig. 4.4) The selling points of AOA are that it does not require accurate timing reference and synchronization, and only two base stations' bearings are enough.

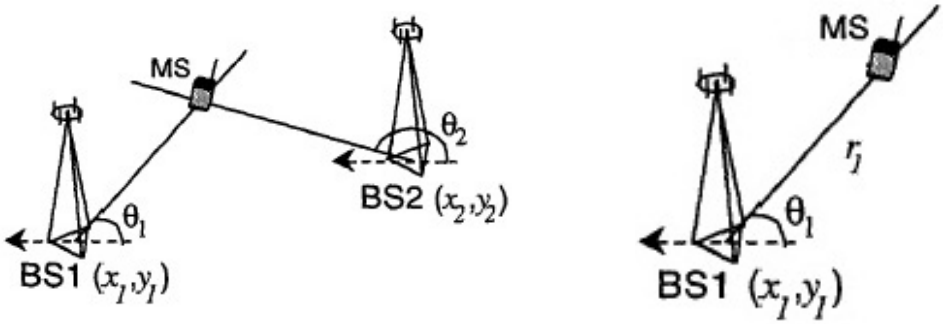


Fig 4.4: Angle of Arrival (AOA) (left), hybrid method (right)

However, in return, the cost of the base station increases because of the antenna arrays. Moreover, a good but complicated AOA estimator algorithm is required in order to have accurate angle measuring.

4.2.4 Hybrid Method

The hybrid method is more advanced. Both TOA and AOA measurements are used, giving the distance r_1, and the angle Θ_1 respectively. Therefore, with exactly one station, the position (x, y) can be calculated. In practice, more base stations would give a better estimation.

4.2.5 Common Sources of Errors for Point Estimation

The most common sources of errors are due to multipath propagation, low cell density and accuracy of base station location itself.

Multipath propagation is the error typical raised in urban area, or even indoor area, where signals have to pass through obstacles like buildings. Those obstacles would invoke additional copies of transmitted signal, which would cause delay and interference. Receiver side does not know which is the correct “line of sight” signal it has to pick up. This would introduce error in both TOA and AOA measurements.

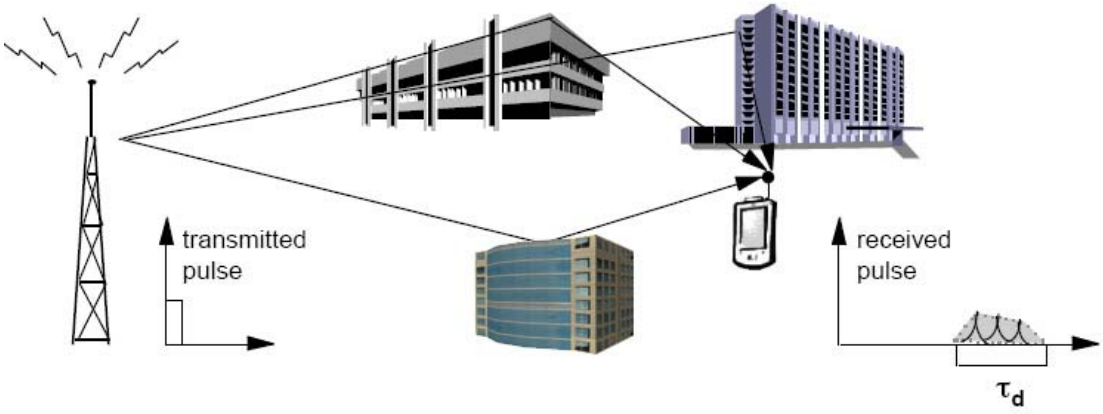


Fig 4.5 Multipath Propagation

At low cell density (e.g. rural area) and the relatively large cell size, the accuracy would significantly drop due to the lack of base station references, which is more serious than the error by multipath propagation.

As all the calculation is based on the location(s) of base station(s) involved in both trilateration and AOA calculation, the accuracy of base station itself is the important key of accuracy.

4.3 State-of-the-Art Technologies

The above are just general methods that apply to all kinds of wireless networks have base stations for reception. However, each wireless network or technology may have its specific features that facilitate positioning, such as E-OTD and A-GPS. Four of them would be introduced in the following, namely Global Positioning System (GPS) as well as Global System for Mobile Communications (GSM).

4.3.1 Global Positioning System (GPS)

4.3.1.1 How GPS works

GPS is a satellite-based positioning and navigation system, which is originally designed for military use. The positioning method of GPS is similar to TOA, while base stations involved are those 24 satellites in the orbit. GPS compares time a signal is transmitted to satellite with the time it is back to the device. By using trilateration, position (latitude, longitude) can be calculated with at least three satellite references. To locate three-dimensional position (i.e. with also altitude), at least four references are required.

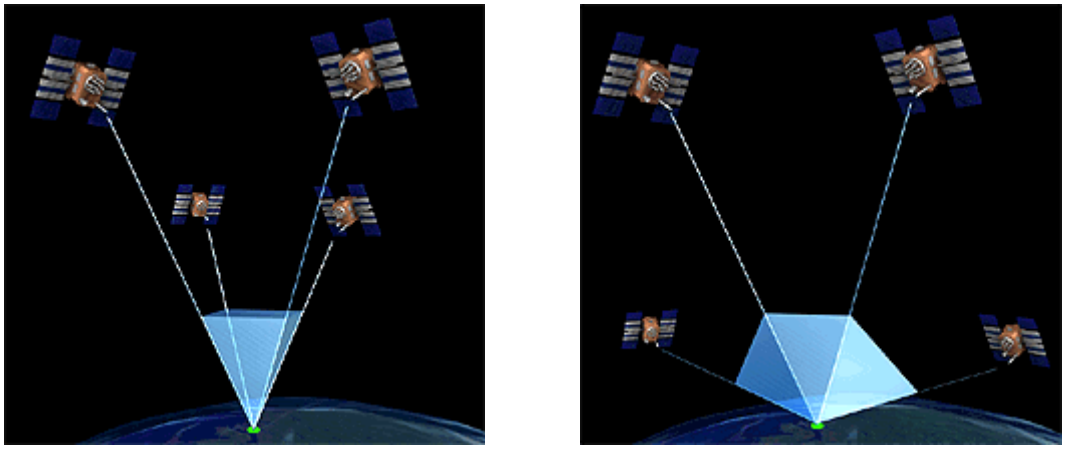


Fig 4.6: Illustration of how clustering of satellites would generate errors. Small tetrahedron formed would enlarge the resulting error in a greater extent for the same measurement error

The error of GPS measurement comes from ionosphere and troposphere delays, multipath propagation, receiver clock inaccuracy, geometric dilution of precision (GDOP) and selective availability (SA). GDOP varies with the geometric location of those satellites in which the closer the satellite references are; the larger the error the measurement would enlarge. Selective availability refers to man-made error the US Department of Defense for safety reasons.

Nowadays, multi-channel of GPS signals is allowed so that GPS device can handle more than 12 parallel satellite signals at a time (recall that more satellites would give higher accuracy). Differential GPS would further raise the accuracy because ground base stations are also involved in GPS measurement to correct the error.

GPS is suitable for applications that require both high precision and velocity estimation. Driving navigation is a typical example.

4.3.1.2 Short-Coming of GPS

The following shows the short-comings of GPS so that GPS cannot be adopted used by ordinary users as LBS solution.

Poor Indoor and Urban-Area Capabilities: GPS works based on satellite signals. However, those signals would be weaken by building, vehicles, etc. when it is used in urban area. Those signals would be nearly blocked in indoor environment. Thus, locating one's position is difficult in these areas.

High Power Consumption: One of the main concerns for mobile devices is limited power supply. Anything designed for mobile device may try to keep power consumption low in order to have long operating time. In contrast, GPS generally consumes high power. As a result, it may not be used critical LBS, such as emergency.

Cost: Although it is free to use the positioning services provided by those 24 satellites, the hardware cost of making a mobile GPS device is still expensive and may not be affordable by mainstream users.

4.3.2 Global System for Mobile Communications (GSM)

GSM is one of cellular radio network that commonly used for mobile telephony. It makes use of time division multiple access and digital technology for voice transmission. Efficient voice encoding and nice data rate to information content radio

make GSM become one of the major standards in the market.

It has several points to note:

1. Most of the people have GSM-capable mobile phones nowadays so that LBS can be available by ordinary mobile phone users.
2. The above techniques, namely pure cell information, signal attenuation, TOA and AOA, still hold in GSM networks, while GSM cell information and GSM TOA are commonly used.
3. The positioning performance varies with cell deployment (e.g. different cell density for urban area and rural area).

4.3.2.1 GSM Cell Information

Each base station contains unique cell identification which is accessible by mobile phone itself. Identification includes location area code (LAC) (or location ID) and cell ID (CI). Such method is simple and does not require the help of telecom company. However, the consequence is that the accuracy is only fair and highly dependent on cell deployment.

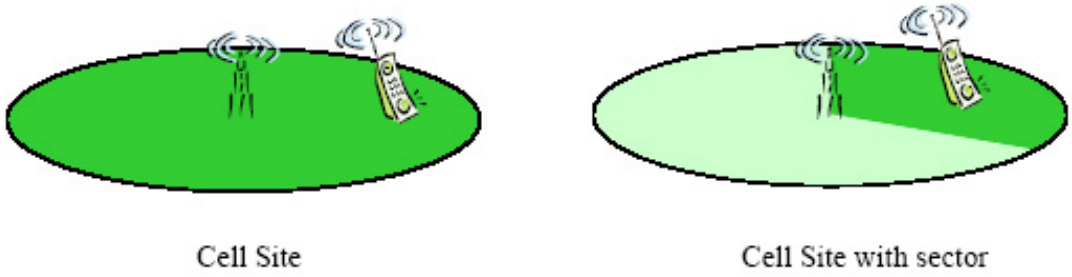


Fig 4.7: Region ensured by using the technique of GSM cell information for normal cell site and cell site with sector (shaded in dark green)

4.3.2.2 GSM Time of Arrival (TOA)

GSM TOA requires mobile phone itself to trigger access bursts to base stations in order to get round-trip time through signal timestamps from surrounding base stations. With trilateration, the location of mobile phone can be restricted to a point or smaller area as shown before. Nevertheless, such technique requires base station synchronization and positioning is only available during calls (i.e. it cannot perform positioning when the phone is idle). Also, if GSM-TOA is used by a large number of users in a particular region, base station work load would increase drastically and this would affect the normal voice communication service. The situation can be made better if high data rate GSM network is allowed (EDGE or 3G).

Moreover, if TOA is initiated by mobile phone, only new mobile phones can do so. Therefore, usually, TOA is started and calculated by base stations to ensure handset independent. As a result, the cost of base station and overhead to base station's normal operation are both increased. Network operators would not prefer this because this requires affect the voice communication quality to their customers.

4.3.2.3 GSM Enhanced Observed Time Difference (E-OTD)

GSM E-OTD methods can highly increase the performance by observing the time difference of signal. From the diagram, one may find that the mobile phone only needs to interact with a serving cell site. Assistance data, including (but not limited to) a list of surrounding base stations to be scanned and frame offsets between these base stations, would be returned to mobile phone. Mobile phone would then scan all the recent transmissions from neighbour base stations and observe the timing differences through built-in position calculation and training function.

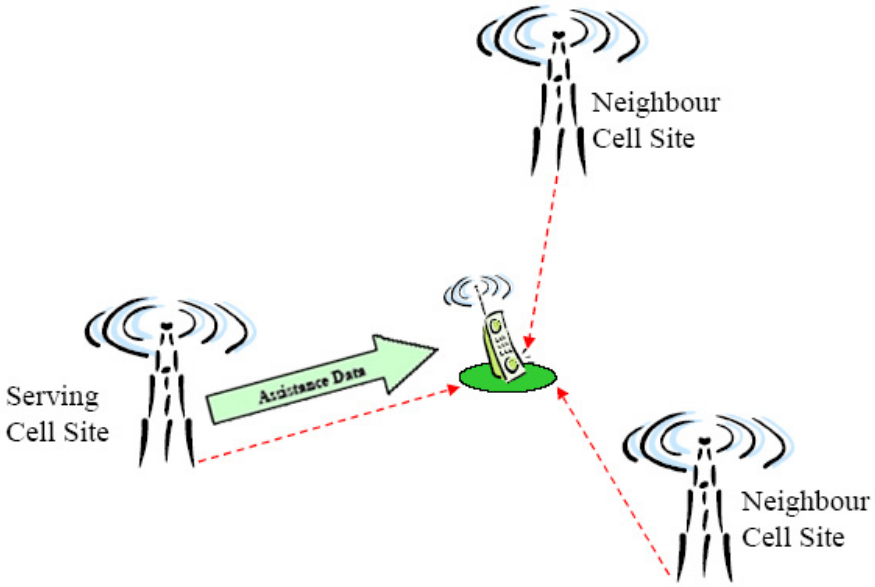


Fig 4.8: GSM E-OTD in operation

As the timing reference is no longer a significant problem (that is why "enhanced" comes), E-OTD would have a distinctive accuracy among the above (see Appendix). Meanwhile, new function should be supported by mobile phone itself and E-OTD phone is expensive. Also, there may be slight impact on current network infrastructure to make E-OTD compatible (e.g. positioning query from mobile phone has to be supported).

4.3.2.4 GSM Assisted GPS (A-GPS)

This method is the hybrid of GPS and GSM technologies. Cell phone is equipped

with GPS device that receives GPS satellites signal, while the information flow is the same as GSM E-OTD. The built-in location calculation functions make use of both trilateration calculation on GPS signals and assistance data from serving cell site.

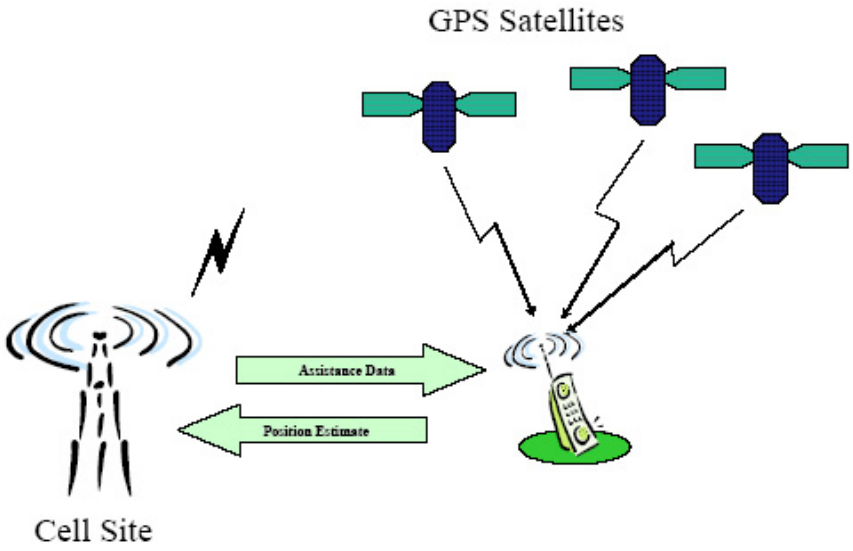


Fig 4.9: GSM A-GPS in operation involving GPS satellites and a serving cell site

The procedures are listed below:

1. The phone obtain signals from three or more satellites;
2. It calculates position via trilateration from GPS received information;
3. It sends the position estimated to a cell site;
4. The serving cell site would return necessary assistance data back to the phone;
5. The phone determines the position by internal E-OTD functions.

Such method improves the former methods a lot:

1. Most of the A-GPS implementations would generate less traffic on GSM network because of less data flow between handsets and base stations (because the GPS estimated position would filter out redundant cell information needed from base stations).
2. Accuracy improves much, usually up to 100 meters for urban areas and 10m for rural areas (notice that GPS well-performs in open area).
3. Such method also provides good velocity estimation.

4.3.3 Phone Support for Advanced GSM Positioning Methods

As mentioned before, some methods, like E-OTD and TOA initiated by the mobile phone itself, require special support on the phone. The below shows some of the ways to let location-based information be transferable to the handset.

4.3.3.1 Modified SIM card

SIM (Subscriber Identity Module) card is a necessary component of GSM to store network registration data (e.g. telephone number registered). Introducing new fields stored in the SIM card, location-based information can be transferred to and from mobile phones.

Those fields are often designed by telcom company, or telco in short, and application development team together so that developers can use the API provided. The problem is that location data is telco dependant.

4.3.3.2 Third Generation GSM (3G / 3GSM)

3GSM represents the third generation mobile communication system based on GSM technology. It aims to link the wireless world with internet and computing by offering high data rates as well as IP-based internet access.

In 3GSM, location information would be included in the standard also. The internal techniques selected may be TOA or other advanced methods. Combining high data capability, more innovative location-based services is allowed, like location-based video streaming. However, a long period of time is required for 3GSM system to replace the current GSM network.

4.4 Why Using Pure GSM Cell ID Method in the Project?

After introducing the strength and weakness of different GSM positioning methods, we decide to pick GSM cell ID method in our project.

Firstly, we choose GSM because most of the phones are GSM-enabled so that ordinary users can enjoy LBS with just their phones. GPS and A-GPS, which are expensive in price, are not suitable for ordinary subscribers.

Secondly, other methods require the support from telco because of the involvement of extra communication between base station and mobile phone other than voice transmission, like AOA and E-OTD. On the other hand, obtaining only cell information does not need help from telco.

Thirdly, although TOA initiated by handset does not require the need of telco also, the up-to-date mobile phone operating system, such as Symbian OS, would prefer to hide the list of surrounding cells in reception and their details, like timestamps. On the other hand, cell ID method requires only current registered cell

information. Moreover, TOA method is only available during calls while cell ID positioning method is always available, provided that the phone is turned on.

However, although using pure cell information for position has the advantages of simple implementation, no extra cost and only current registered cell information required, the accuracy is really suffered, especially with large cell. The entire project would try some ways to improve this weakness.

4.5 Conclusion

General positioning methods, such as TOA, AOA, etc., were introduced. They could be applied in any wireless networks, like WLAN and Bluetooth, with different accuracy and weakness. World-wide positioning technologies, namely GPS and GSM, and advanced positioning methods specific to GSM, including E-OTD and A-GPS, were presented also. The chapter ends up by stating the reasons why GSM cell ID method is selected in our project.

Chapter 5: GSM Cell ID Collection Methods

The next step is to obtain cell information for studies and developing applications. Without the help of telco, we have to gather the cell information, store them and let the application to interpret them within the mobile phone. Two traditional methods and our cell ID collection method are stated here.

5.1 Communicating with GSM Modem

GSM phone consists of a GSM modem an interface for underlying GSM communication. If, by some means, one can communicate with GSM modem, GSM cell information can be retrieved.

GSM modem inside the phone uses a set of 'AT commands' for instruction, similar to current 56k PC modems. The below shows the general structure of GSM AT command flow.

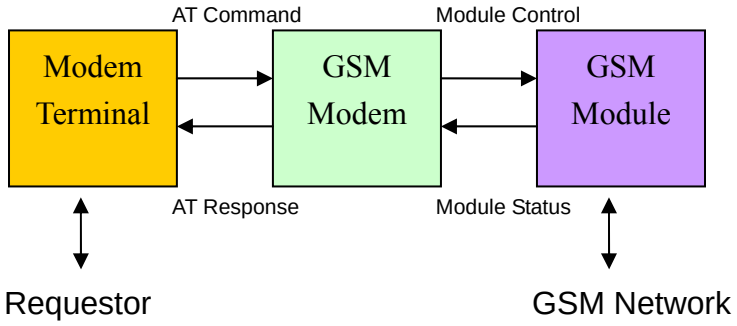


Fig 5.1: General structure of controlling GSM communication through GSM modem

Modem terminal is an external hardware or device, such as PDA, that can communicate with GSM modem. AT commands and responses flow through some communication channel, including serial cable (RS-232) or infra red. Then GSM would translate the request or redirect the response to GSM module, eventually sent to GSM network.

The AT command of retrieving network information varies slightly for different brands of mobile phones. However, the common network registration AT command, '`AT+CREG?`' is based on the common standard GSM 07.07 specification [15] and most of the phone manufacturers follow it. The following shows the scenario of using a Pocket PC as terminal connected with handset through serial cable:

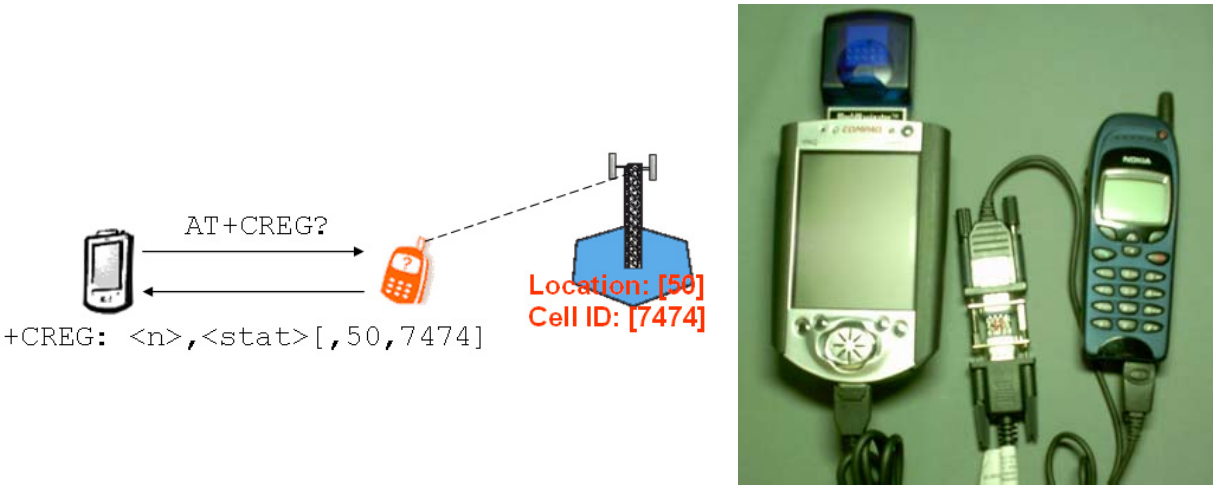


Fig 5.2: Obtaining current cell information using AT command and response (left) (where <*n*> and <*stat*> state the network connectivity status), picture of the setting of communicating with GSM modem inside the phone (right)

The above hardware setting is quite complicated and information collector has to write a program to automatically generate AT commands dumping to the serial cable.

5.2 Using Internal Phone Engineering Mode

Some of the phones may have 'engineering mode' for experienced users. Of course, general users would not have temptation to look into it. The engineering mode would differ from phone to phone, giving different pieces of information.

For example, traditional engineering mode of Nokia phone can be activated through PC software. The below shows the snapshot.

Fig 5.3 Nokia phone engineering mode

Although no other hardware involved (only the phone itself) for collecting cell information, manual recording, say with pen and paper, is needed because there is no way to store those data inside traditional phones.

5.3 Using Symbian API

Symbian phone is a programmable phone so that developer can write their own programs to do whatever the operating system allows. Therefore, if Symbian OS itself does provide some APIs for accessing GSM information, the collection can be hardware-free and the cell IDs recorded can be stored into the phones (no manual recording required).

5.3.1 Problem Encountered and Solution

However, the problem is that current SDK and documentation do not mention such API. As mentioned in earlier chapter, mobile phone operating system, such as Symbian OS, would like to hide all hardware information from both users and developers.

After searching from the web [19], we found that it is possible to do so. It was figured out that the library in the mobile phone actually had such APIs, but the SDK did not have such routine for developers to call. Finally, we obtained a header file, which was from earlier version of SDK, from other experienced Symbian programmers and access to the GSM cell information.

5.3.2 Symbian Application - GSM Status

The below is the application, GSM Status, showing the current registered cell information, including location ID, cell ID and received signal strength.

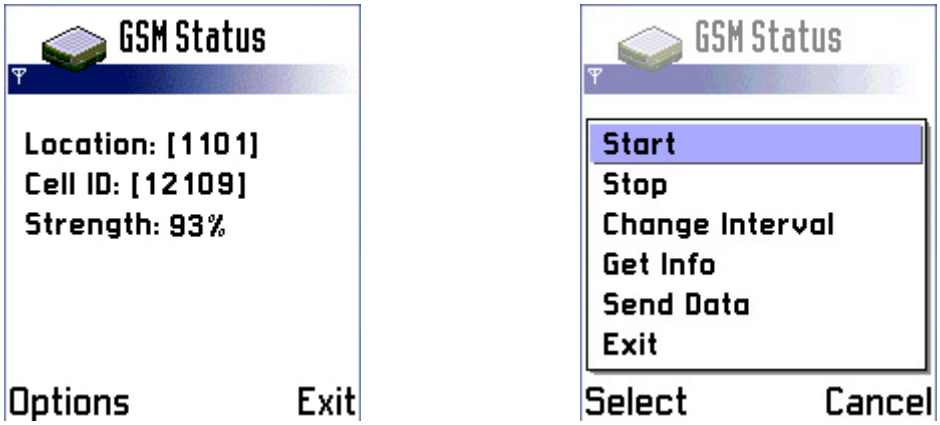


Fig 5.4: The snapshots of a Symbian application, GSM Status. Detecting current registered cell information (left) and menu of GSM Status (right)

GSM Status also requires the use of timers and active objects, a Symbian term to describe thread-like objects, to refresh the cell information periodically and handling the time-consuming signal strength retrieval routine.

5.4 Conclusion

Traditionally, people collect cell information by either sending AT commands to GSM modem inside the phone or invoking phone engineering mode. However, the outcomes are that complicated hardware settings and manual data recording respectively. In our project, a Symbian application, GSM Status, was made for solely cell ID collection after hacking the SDK and getting into hidden APIs.

This part also illustrates a very important feature of Symbian OS: programmable capability. Developer can directly interact with the phone functions (GSM network module in this case). This cannot be done in the past (people have to write programs on other devices, like Pocket PC, and transfer data with the phones) and no user would use bring complicated hardware with phones to enjoy the application.

Chapter 6: Enhancing GSM Cell ID Positioning Method by Using Cell Change Event

In Chapter 4, it was shown that GSM cell ID positioning method has several advantages, including simple implementation, no extra cost needed, no support from telco and only current registered cell information required. These fit our project motivation tightly. However, such method would only telling "which region the device is in", rather than "where the device is" – it is a region-based location estimation, unlike TOA and AOA which are point-based.

While many people may stick on using cell ID purely, we would try to use another view to look into this positioning method. In fact, using cell change event would provide more information.

6.1 Problem of GSM Cell ID Positioning Method

The following would have a discussion on the inadequacies of using this approach, namely limited information and reliability.

6.1.1 Limited Information Provided

This method based on which cell the phone is current connecting to. The major weakness is that information depends on the cell size. With large cell size (and low cell density), the returned information may be even useless.

Imagine there are only three cells (Hong Kong Island + Lautau Island, Kowloon and the New Territories). GSM cell ID positioning is only able to locate the device either in the given three locations. There is no way for people to know more. However, if there are more cells (e.g. each district has a serving base station) information obtained can be constricted to lower level (e.g. district level in this case).

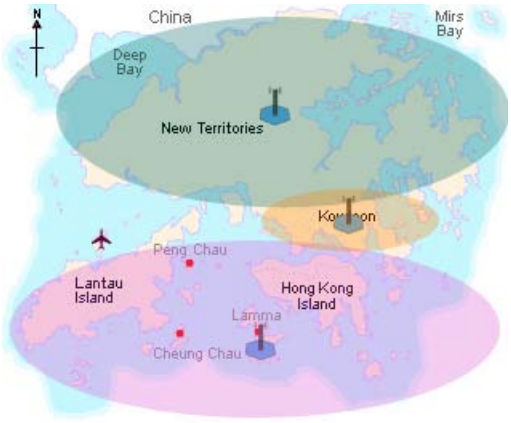


Fig 6.1: Fictitious scenario: Hong Kong with only three cells

The source of the problem is that using pure cell information is a region-based estimation in which information content is limited by the cell size and density.

6.1.2 Low Reliability of Registered Cell

The cell which gives the mobile phone the highest received signal strength would be chosen as current registered cell among all neighbour cells. However, the received signal strength depends on the signal attenuation of surrounding obstacles, so current registered cell is different from time to time. There is no guarantee that, at a particular position, the device must have the same registered cell, especially when coverage regions are highly overlapped to each other.

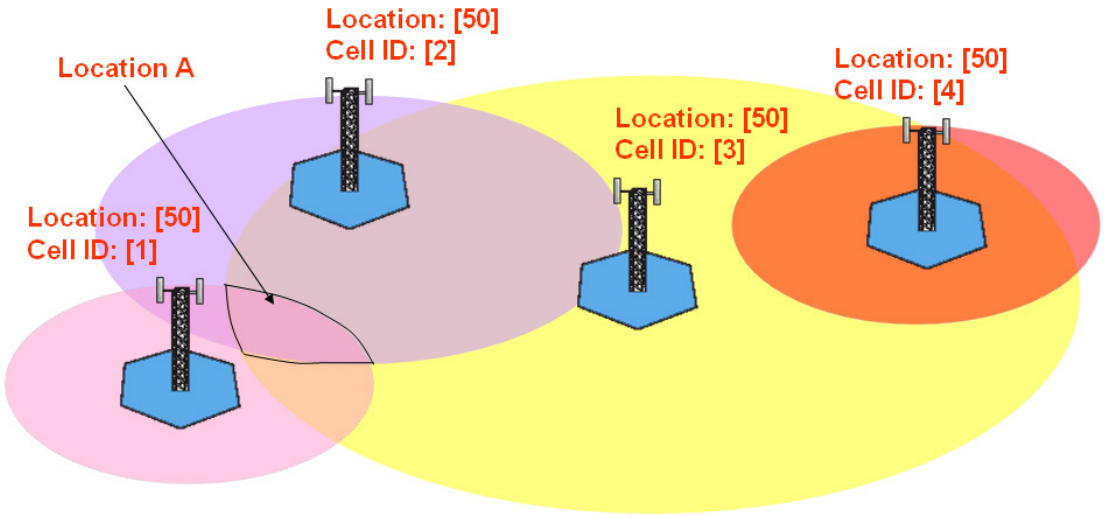


Fig 6.2: Illustration of overlapped cell coverage

From the above figure, the target handset is located under the overlapped coverage of cell 1, 2 and 3. All of them are inside the list of "neighbour cells" in the phone. At different time t, different cells would be registered due to varying received signals. If the internal database mapping states that, for example, cell ID 1 represents location A only, the service would become unreliable because the system would never recognize the device is in location A when registered cell is 2 or 3.

6.2 Cell Change Event

To obtain more information, the idea of cell change event is proposed. Cell change event refers to the moment when there is a change in current registered cell. The cell change event from cell 1 to cell 2 indicates that the received signal strength from cell 2 is higher than cell 1. As a result, more information is offered and, more importantly, a more reliable positioning is provided.

6.2.1 Transition Information Offered

By listening to the cell change event, as a result, two extra pieces of information can be deduced from a particular cell change event.

1. The target device is entering / leaving a "boundary" at that time.
2. The transition information can also be provided (i.e. from 1 region to another region).

The below scenarios show the difference from pure cell ID detection. When two mobiles entering the purple region in different directions, the cell change event is different (pink region → purple region and yellow region → purple region). More information provided can distinguish more discrete "locations". Compared with pure cell ID detection, two scenarios would give the same piece of conclusion: two cell phones are both in the purple region.

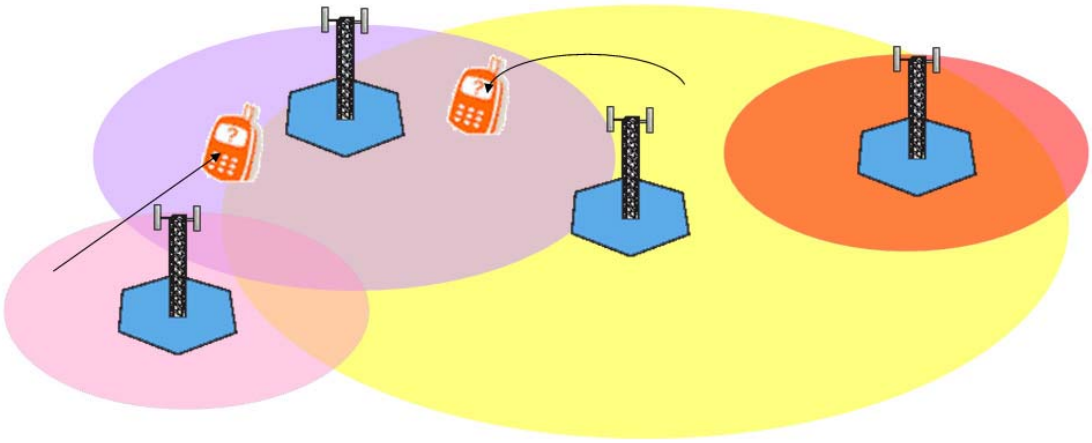


Fig 6.3: Different transitions can be deduced by different cell change events

As a result, the table that stores location-cell ID mapping would be modified into something like the following.

Cell ID	Location
1	A
2	B
3	C
…….	…….

Old Cell ID	New Cell ID	Location
1	2	A1
1	3	A2
2	1	B1
2	3	B2
3	1	C1
3	2	C2
…….	…….	…….

Fig 6.4: Different location-cell ID mappings used by pure cell ID detection (left) and cell change method (right)

6.2.2 Reliability Improvement

However, the main beauty of this approach is not simply the transition information. Instead, the main feature is to figure out the boundary relatively reliably and accurate timing. As mentioned in former section, signal attenuation would lower the reliability of positioning. It is guaranteed that when the original cell cannot offer the device with enough signal strength, it would be replaced by another "better" cell. Therefore, provided that the target device is moving across the boundary, cell changes would probably occur. This piece of information is more consistent than "When the current registered cell is XXX, the mobile phone must at location YYY".

Therefore, the cell change event could figure out the moment that the phone is at the boundary of the new registered cell's coverage as following (In practical, the "boundary" is irregular in shape, but not circular). The estimated position becomes a line (possibly a thick line) rather than a great region.

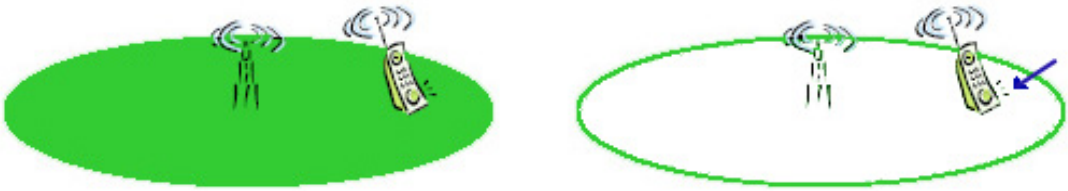


Fig 6.5: The positioning region that two approaches would indicate: pure cell ID detection (left) and cell change method (right)

However, one point should be emphasized here. Cell change method can only perform positioning at the boundaries only. It cannot give any indication when the device is inside the cell area.

Moreover, unfortunately, the boundary would not be fixed forever. Due to the regular adjustment on cell size (i.e. signal strength transmitted by each base station) by network operators to optimize the operating cost, the boundary would be varied from time to time. Therefore, regular update of location-cell ID mappings is necessary.

6.3 Hybrid Method

However, as the improvement using cell change event is only applied at cell boundaries only, both methods stated above can be combined together. In fact, using both of them would not be any conflict since cell ID detection concerns the current information while cell change method considers the moment at which registered cell changes. Therefore, in our MTR Traveller project, which would be introduced in later

chapters, pick both of them for location estimation.

The consequence is that two separate tables are necessary to store all information: one for location-cell ID mapping and another for location-transition mapping as shown in Fig 6.4, which may be a storage overhead when location-base service is applied in large area, like whole Hong Kong.

6.4 Conclusion

The cell change event provides transition information and performs location estimation in a more reliable way. However, as cell change event is applicable at the cell boundary, hybrid method becomes a new consideration. In the next few chapters, there would be experiments on using GSM cell ID positioning.

Chapter 7: Understanding Cell Distribution in Two-Dimensional Space – Experiments in CU Campus

After GSM cell data collection application, GSM Status, had been available, experiments were performed in the CU campus in order to understand more about cell distribution and to clarify our initial expectation on GSM network. By drawing the predicted cell distribution ("predicted" is used because we do not know exactly where the base stations are located), the accuracy of location estimation using proposed method can be known.

7.1 Initiative and Expectation

Without seriously compare with other existing methods available, such as GSM E-OTD and GSM A-GPS, we think that using pure cell information is sufficient to locate which region the device is in the CU campus.

Another initiative is that we would like to collect the cell information in order to find out the possibility of reverse-engineering the actual locations of all base stations in the campus, which would be invisible to ordinary mobile phone users.

To actually prove the above, a series of experiments was done by using GSM Status. Besides, there were some other expectations before performing the experiment:

1. GSM cells inside CU campus were of similar sizes and all of them are micro-cells (100m to 1000m). If so, the error of location estimation can be bounded by some values.

2. Those cells would have small overlapped area with other cells so that error due to multiple registered cells at a particular location could be ignored. This implies that each cell would occupy a distinct area that others are not responsible for and this makes the positioning becomes simpler. Small overlapped area also implies that it should not have the situation that a large cell covers a small cell.

3. There would be in different shapes that cannot be easily controlled. This expectation appears because a simple hexagonal shape to represent the geographical coverage of base station antenna [20].

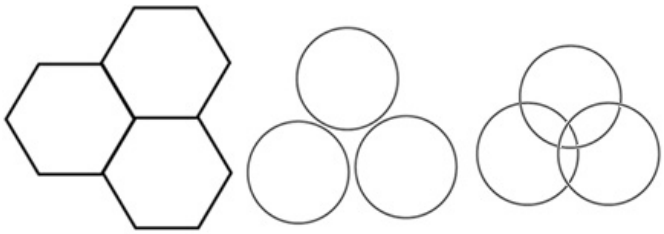


Fig 7.1 Hexagonal cell site is common description of cell coverage because circular shape would introduce gaps or overlapping areas

4. The altitude of a specific location would not affect the result (e.g. the registered cell at a specific location would be the same with different altitudes). Thus, this experiment concentrates on two-dimensional domain only.

7.2 Experiments in CU Campus

Two different measuring methods would be used: static method and cell change method. They are based on two positioning methods (i.e. pure cell information collection and cell change event collection) mentioned in the previous chapter.

7.2.1 Static Method

The so-called static method means that data recording is done at a number of points in the campus. The experimenter has to wait at a particular location for a sufficiently long period of time (e.g. several minutes) to observe the variation of current registered cells and received signal strength. The cell that has the longest time of registration would be marked as representation of that location. Consequently, the distribution of cells can be roughly estimated by those points.

The number of selected points must be sufficient large enough to determine the cell distribution and locate the cell boundaries clearly. Therefore, the key point of doing this experiment is that the selected should be dense enough to cover the whole map. The most ideal case is that all the points (in fact, there are infinitely many) in the map can be measured so that the cell distribution of the campus is able to be figured out eventually. However, it is practically not possible and sometimes not necessary because non-overlapped region would have only one cell registered.

Another disadvantage of doing so is that it would take a long time to collect enough information. If the number of selected points is large, it may take even several days to accomplish the task.

The experiment was done using Nokia 7650 mobile phone, embedded with GSM

Status application, with SmarTone as the network operator. The locations interested in this experiment were the public spaces in the campus, including four college and central campus. To increase the accuracy of measurement, the experiment was performed 5 times in total between September 2003 and October 2003.

The experimental result would be shown at Section 7.2.3 for easy comparsion. For simplicity and easy reading, not all of the points measured are shown in the map.

7.2.2 Cell Change Method

As mentioned in the previous chapter, using GSM cell change method can determine the boundaries of cell coverage. This experiment is based on that idea to capture the cell change events instead of measuring at static locations. The experimenter has to walk around the campus and record all the cell change. By having enough identical cell change events, a cell boundary can be shown, instead of the received signal strength at the specific points. The below shows an example. The given four cell change events are identical (i.e. indicating that the phone moved from the pink cell to the purple cell). Therefore, it can make a reasonable estimation that the blue line is part of the boundary of purple cell's coverage.

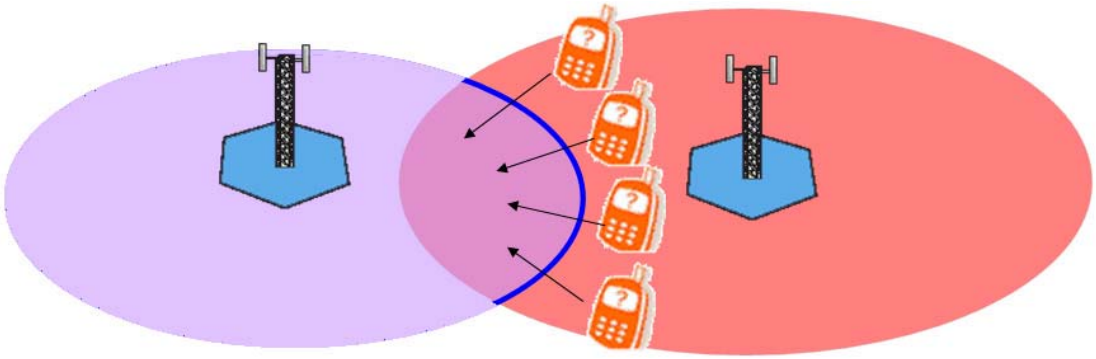


Fig 7.3: Determining a boundary using cell change method in our experiment

The challenge is that the practical possibility of entering a cell for from all directions without the knowledge of actual location of a cell. Also, notice that there are numerous of cells that the experimenter has to enter. Therefore, the path selected in the experiment cannot be preliminarily planned.

This experiment could be accomplished in a smaller amount of time, compared with the static method mentioned previously. With the same equipment, the experiment was held in October 2003 with another network operator in Hong Kong, Peoples. Again, the measurement was repeated for 5 times to obtain more reliable results. The result would be placed at next section for easy comparison.

7.2.3 Experimental Results and Observations

The experimental results give us more understanding on GSM networks and the inadequacies of proposed GSM positioning methods in two-dimensional region. By the cell data recorded, the distribution of cells and cell coverage were roughly estimated (by grouping the marked cells for static method and by showing the boundaries of all the cells involved for cell change method).

Fig 7.2: Experimental result of static method for SmarTone in the campus

Fig 7.3: Experimental result of static method for Peoples in the campus

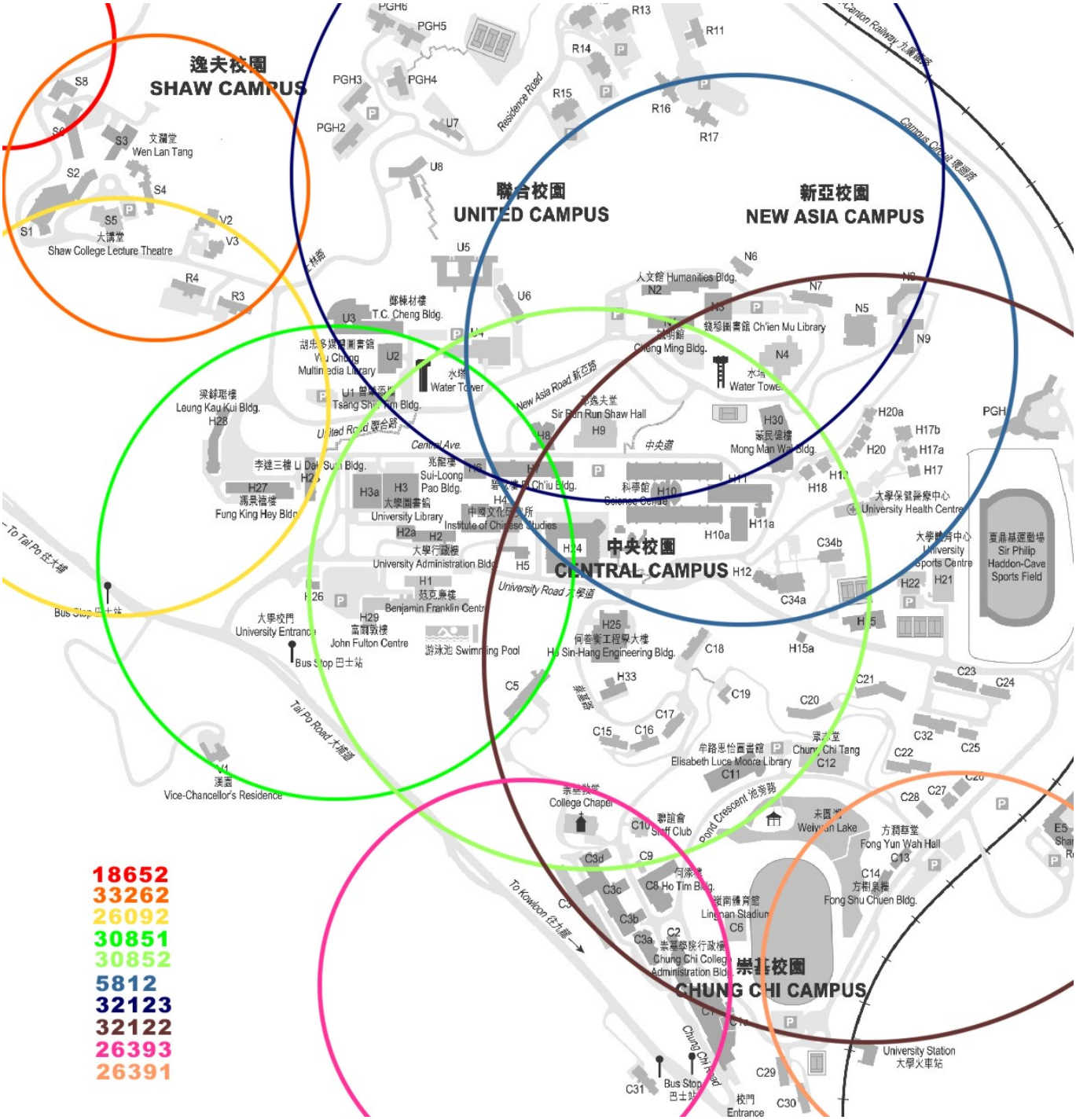


Fig 7.4: Estimation of cell distribution and coverage for SmarTone in the campus

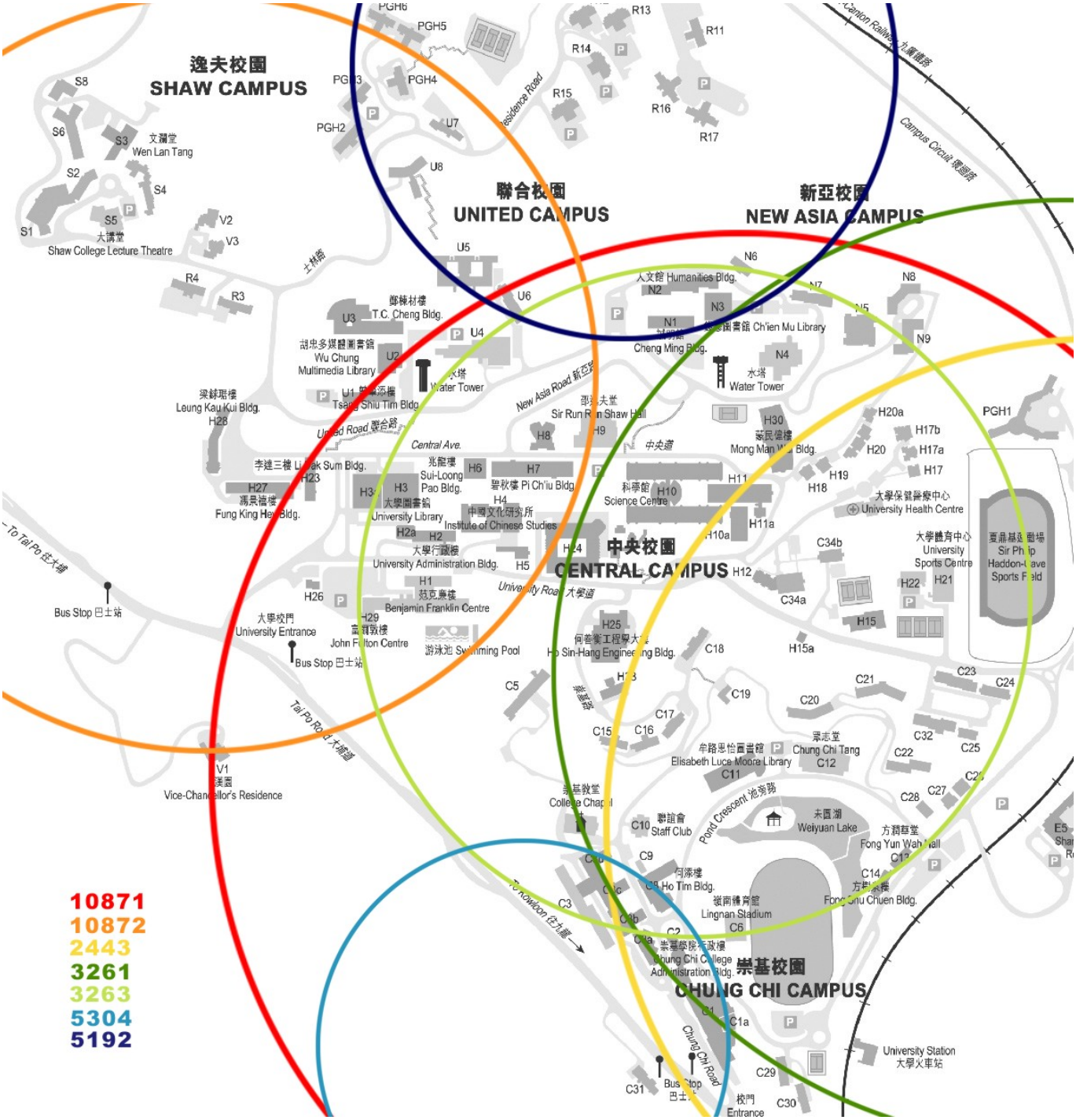


Fig 7.5: Estimation of cell distribution and coverage for Peoples in the campus

Notice that the above estimations only present the cell coverage as circle but, in reality, the cell coverage does not have definite shape.

It is observed that there are number of contradictions with our initial expectations. Firstly, cells vary greatly in size as shown in the diagrams, but most importantly, the extent of overlapping is so high that we cannot expect (there is even a situation that a large macro-cell encapsulating a micro-cell – completely overlapped). With high extent of overlapping, only using cell ID itself for locating is practically unreliable, as stated in Section 4.2.1, while cell change method can only be applied on boundaries only (Section 4.2.2).

Secondly, as the CU campus is not an urban area indeed, network operators would fill the campus with few cells in a loose manner to minimize the cost. As a result, the GSM cell size is quite large for accurate positioning.

Thirdly, the altitude of location also makes a difference and this is out of our expectations. For example, in Ho Sin-Hang Engineering Building, the registered cell ID would be kept at 30852 on 1st floor. Nevertheless, it would be 32122 or 5812 on higher floor, like 9th floor, and cell ID 30852 would never show up. Therefore, cell ID-location mapping may need to store altitude also, generating storage overhead for large area positioning (e.g. Hong Kong).

Another observation is that the cell distribution can somehow determine the positioning performance for different mobile operators. By comparing the given two cell distributions by network operators, the main difference is the number of major cell involved and their cell coverage. SmarTone would keep their cell sizes small and densely deployed over the campus, compared with Peoples. Therefore, it can also be concluded that GSM positioning depend tightly on the cell deployments of different network operators.

The experiment can also indicate the difficulties of reverse-engineering the actual location of base stations. The cell size is large, resulting in enlarging the measurement error. Irregular cell shape and high degree of overlapping would garble the estimation.

7.3 Conclusion

The proposed GSM cell ID location estimation cannot work well with two-dimensional space, especially in the conditions of large cell size, irregular cell

shape and high extent of overlapping. All of these observations make us giving up our initial plan to provide location-based service in the campus and reverse-engineering the locations of base stations. However, another idea can be raised: using GSM cell ID in one-dimensional scenario (i.e. a line or a route). There would be a detailed discussion in the next chapter.

Chapter 8: Using GSM Cell ID Positioning Method on One-Dimensional Path – MTR Traveller

A number of factors, such as cell overlapping, would make the initial plan of applying GSM cell information location estimation in the campus unrealizable. However, the cell change method can really improve the reliability at cell boundaries and offering more information. Finally, it is figured out the key is limiting the region of interest into a one-dimensional area (i.e. a line).

8.1 Initial Observation

While the series of experiments in the CU campus showed the inadequacies of pure cell ID detection and cell change method, another rough investigation was held for MTR and KCR railways. For stations in subway, such as Mong Kok and Prince Edward MTR stations, there was exactly one cell ID change between two stations in subway as shown. This event can tell users that they are on the way from station 1 to station 2.

Fig 8.1: Observation of cell ID change in MTR stations in subway

This raises the idea of using cell change method in one-dimensional path.

8.2 Working Principle

The idea is illustrated in the following figure.

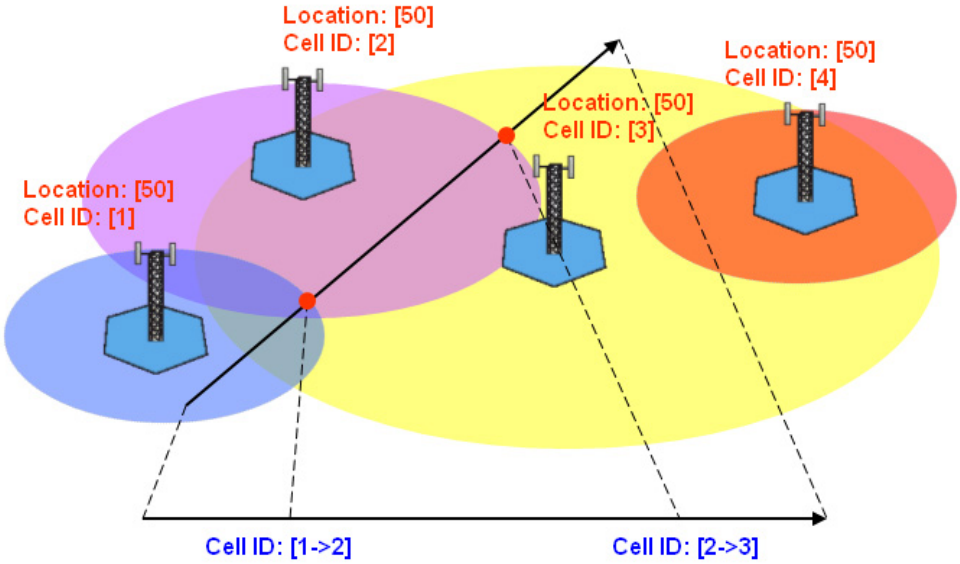


Fig 8.2: Illustration on applying cell change methods on a path

Consider the route from cell 1 via cell 2 and cell 3, as indicated by the black arrow. Recall that cell change event would occur at cell boundary, where signal strength of current registered cell is weaker than that of another. Therefore, cell changes would take place at locations specified by red dots. Therefore, in the journey, the mobile phone would experience two cell changes. The main idea is that by the occurrence of the cell change events 1 → 2 and 2 → 3, it may be possible to identify the mobile is moving in the specified path.

The diagram in Fig 8.3 would also intuitively show why the power of cell change method in handling a path. It was known that cell change method can locate target device at the cell boundary. Usually (but not always), the path cuts the boundary at two points. Therefore, the problem can be resolved into a point estimation of specific cell changes (i.e. the intersections of the path interested and the cell boundary).

Therefore, the below shows how our method evolutes for accuracy improvement.

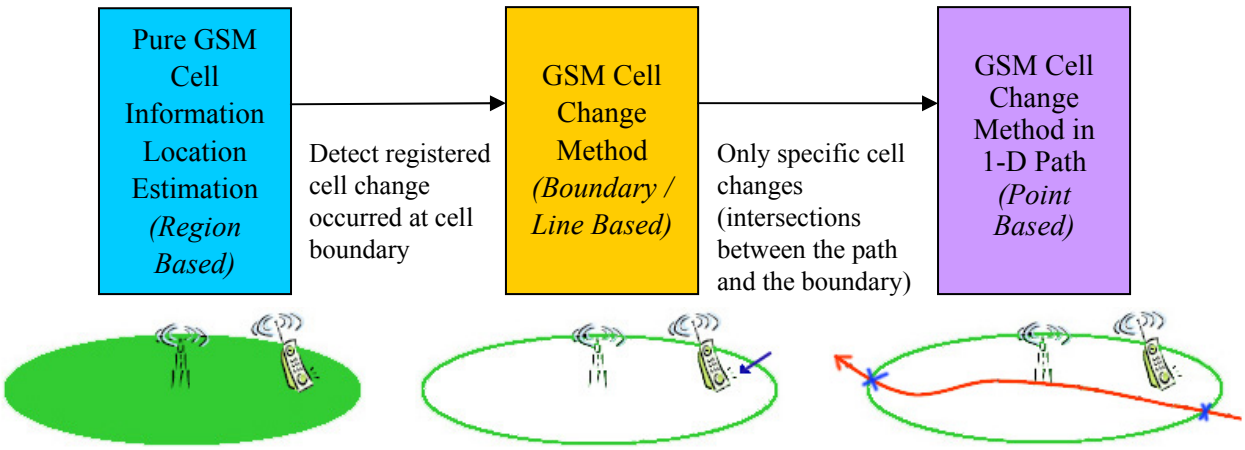


Fig 8.3: Proposed GSM positioning method improvement from using pure GSM cell ID and cell change method to cell change method in 1-D path

8.3 Limitations of the Idea

However, two problems had to be solved. The first one is that there exists more than one path (theoretically, infinitely many paths are possible) giving the same combinations of cell change events (Fig 8.4 left). Another challenge is that multiple cell change events may occur other than those interested (Fig 8.4 right).

The first one somehow points out the limitation of the idea: the path should not have a great variation or even fixed. The problem can be minimized if:

1. The cell size is sufficiently small for the application using this method (it is based on the requirement of application);

2. The overlapping boundary arc between two cells is small enough. It is because the problem is due to having multiple paths having SAME cell change combination, which, in turn, depends on the length of overlapping boundary arc that allows multiple lines. Small arc would have smaller number of lines

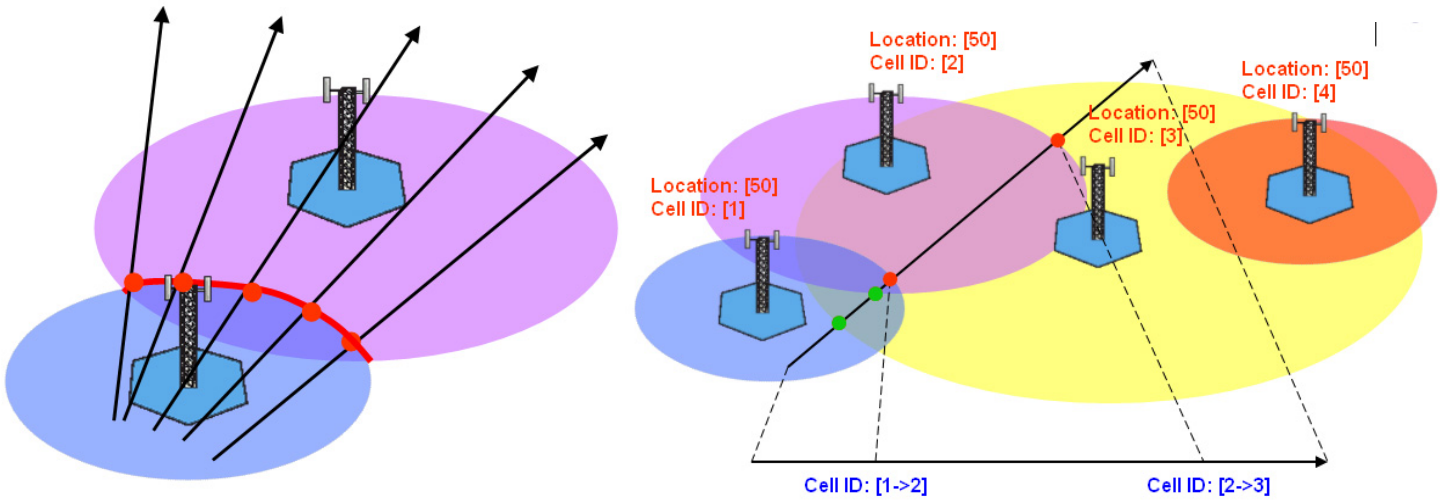


Fig 8.4: Multiple paths having the same cell ID change combination which occur only on the overlapping arc between cell sites involved (left), other cell change, indicated by green dots, may also occur (right) compared with Fig 8.2

The second problem is generally unavoidable, even though the path is fixed and has been already known. There are quite many other cell change events possible, provided that the received signal strength of the old cell is lower than that of the new cell. The ultimate solution is that more data collection trials have to be done to figure out all the cell changes possible in the specific path.

8.4 Symbian Application Using the Proposed Method – MTR Traveller

MTR Traveller is the application that aims to perform station positioning in **BOTH MTR stations and KCR stations** (do not mislead by the name because MTR Traveller is initially for designed for MTR stations only) after the idea of using GSM cell change method on route. The following sections would investigate into the development of MTR Traveller.

8.4.1 Strengths of the Proposed Method on MTR Traveller

It was stated in Section 8.1 that the initial observation of this proposed method is

from MTR stations in subway. While in Section 8.3, we figure out some limitations of the method. Using the method in MTR Traveller can fit the necessary conditions:

1. The path travelled by MTR (i.e. the MTR railway) is fixed in location and it would not change with time. Therefore, the problem of multiple paths can be ignored.

2. As mentioned before, in between two MTR stations in subway, there exists exactly one cell ID change. The case of numerous cell changes scattering along the path would not happen.

   Therefore, the use of the proposed method can suit our application well.

8.4.2 Hybrid Method on Stations in Open Area

For stations in open area, the resulting problem cannot be solved so easily as those stations in subway. It is because many different base stations are involved in between two stations. With large number of cells and high degree of overlapping (even a macro-cell encapsulating a micro-cell), the cell detection can be confused of the device direction.

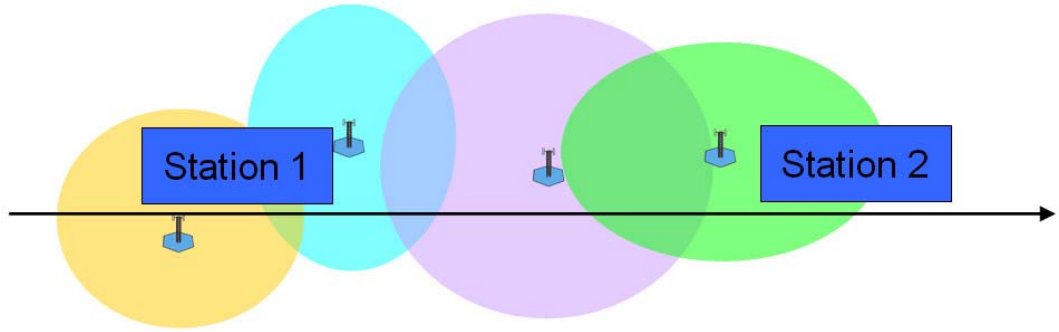


Fig 8.5: Multiple cells between two stations

Moreover, there are multiple cells in station platform also because station platform is long enough to be covered by more than one cell site (e.g. station 1 in Fig 8.5 is covered by both brown and blue cells). Therefore, we cannot simply use a cell ID to map a station.

Therefore, as mentioned in Chapter 4, hybrid method, where both pure cell ID detection and cell change method are combined together, can be applied here. As a result, two types of data would be required, namely station cells (mapping for pure cell ID detection) and transition pairs (mapping for simplified cell change method). Station cell set collects the entire cell IDs possible to involved with the station

platform; Transition pair set stores all cell changes that would occur in between two stations. An example would be accompanied in the following for explanation.

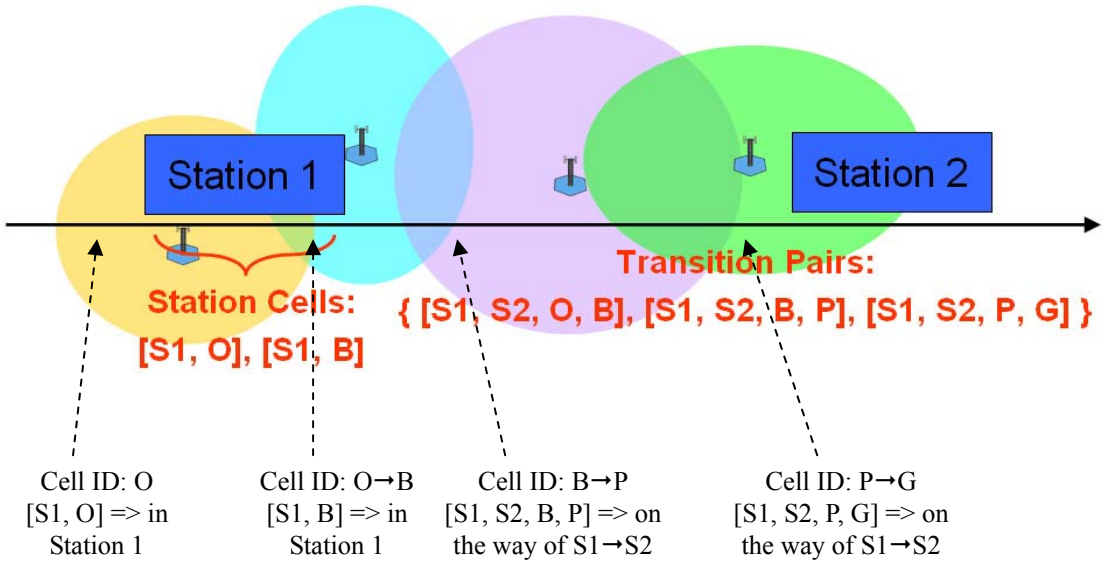


Fig 8.6: MTR Traveller using station cells and transition pairs for station positioning where O = Orange, B = Blue, P = Purple, G = Green

Station cells indicate the fact that it is at station (e.g. current registered cell ID is O and [S1, O] in station cell set imply that the train is at station 1). Transition pairs indicate that the train is on the way between two stations and the sequence of two cells would indicate the direction (e.g. transition pair [S1, S2, P, G] with cell ID B → P imply that the train is on the way from station 1 to station 2; transition pair [S1, S2, P, G] with cell ID P → B imply that the train is on the way from station 2 to station 1).

As a result, station positioning for open area can still be realized. This implies that both MTR and KCR can work under the same principle.

8.4.3 Data Collection in MTR and KCR Stations

Before the implementation of MTR Traveller, the collection of cell information along both MTR and KCR railway is necessary. Again, the task was accomplished using our Symbian application, GSM Status. SmarTone, Peoples and Sunday SIM cards were involved in this data collection between October and November 2003.

The below presents the data in graphical form for convenience. Here the cell data of SmarTone in both MTR and KCR stations are selected for discussion of the results. The remaining cell data is put in the part of Appendix. The data collection aimed to illustrate the idea of the distribution of cells at this stage. Therefore, data was not collected for some of the MTR lines, like MTR Tung Chung Line. Moreover, the cell

IDs were collected for more than 5 times for data reliability reason.

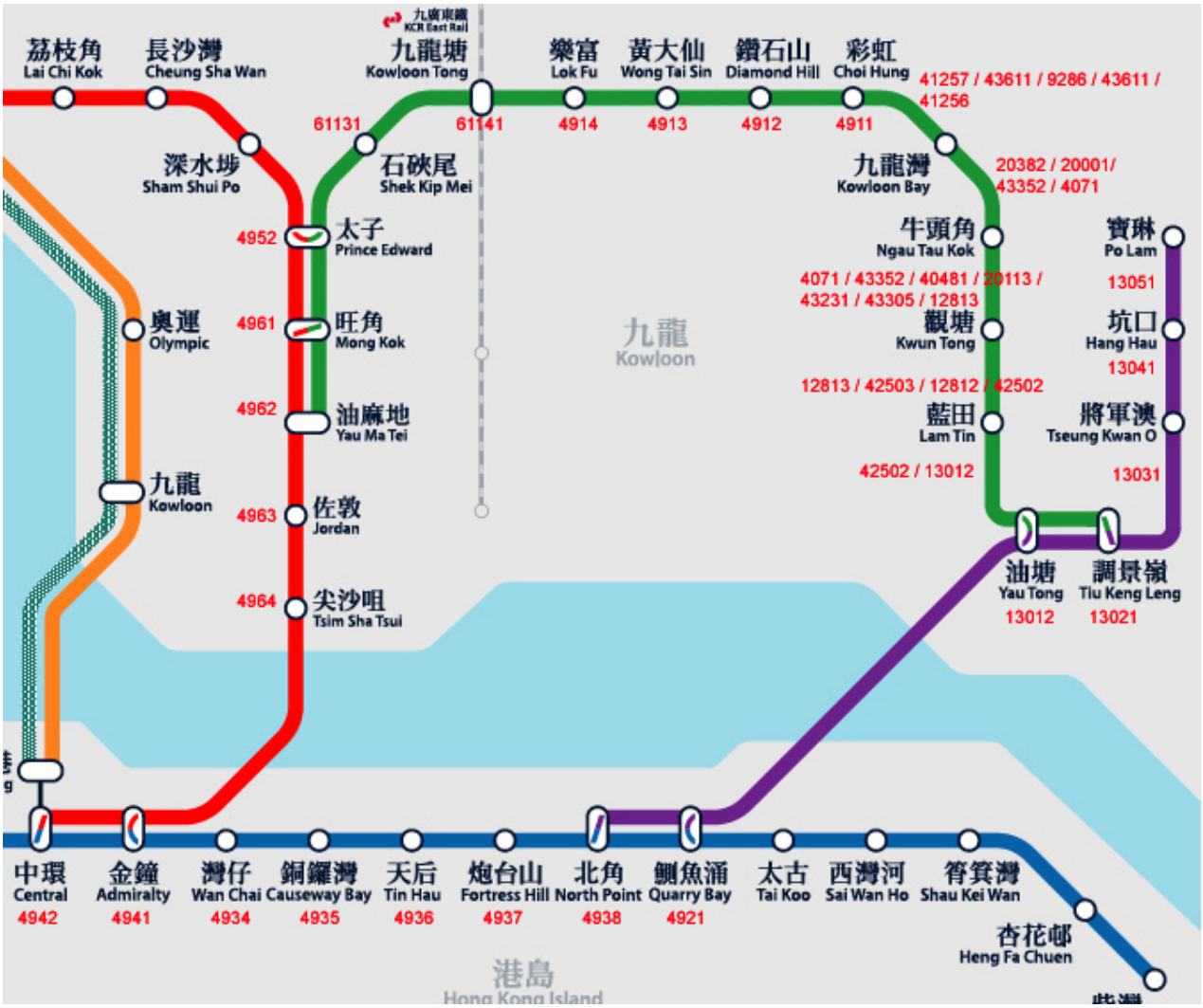


Fig 8.7: Cell IDs of selected MTR stations for SmarTone

Fig 8.8: Cell IDs of KCR stations for SmarTone between Kowloon Tong station and University station

From the above diagrams, it illustrates the difference between stations in subway (e.g. Mong Kok and Prince Edward MTR stations) and stations in open area (e.g. Kwun Tong station and all KCR stations). For stations in subway, only one station cell was required to be recorded; for stations in open area, a series of cell changes occur and they were recorded as transition pairs. The result was similar to what we expected.

8.4.5 MTR Traveller Implementation

As mentioned before, cell information is stored as station cell set (i.e. station file) and transition pair set (transition file) for processing. They are inserted into a table in database engine as efficient data storage. Moreover, the railway maps of both MTR and KCR are also necessary as the input data file of MTR Traveller for the user interface and they are stored as real-time data (e.g. 2-D data array). The user interface is for presentation of current station and transition. The main MTR Traveller engine is responsible to process all of these modules: querying cell information from database engine, drawing the suitable station bitmaps on the screen (due to user's current station) and handling user input.

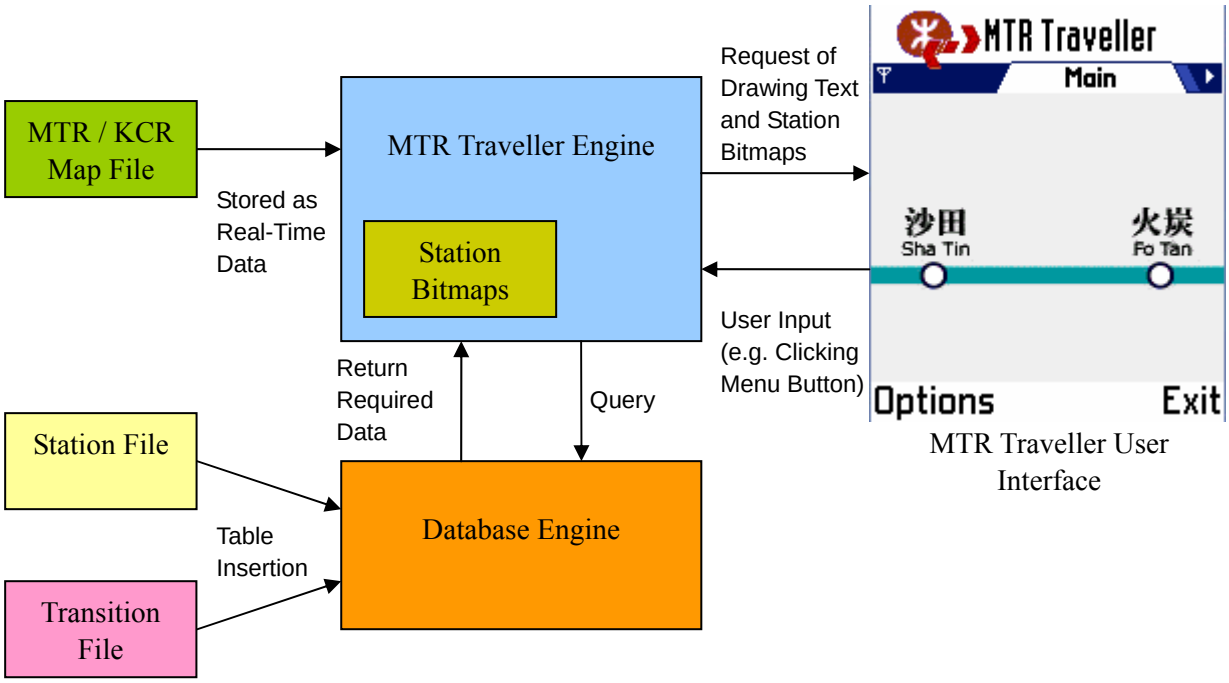


Fig 8.9: MTR Traveller structure and interaction between each module

8.4.5.1 Input Data File

Besides, the processing of central engine and the use of station file and transition file have been discussed already – the hybrid method. The map file is simply a two-dimensional table of bitmap IDs. The format and sample of those input data files

are located in the part of Appendix.

8.4.5.2 Database Engine and its Limitation

Symbian OS has a simple database manager which allows [21]:

1. Multiple tables in a single database,
2. Sharing of a single database between multiple client programs, using a server architecture, transactions, locks as well as notifications, and,
3. Convenient access to data through an SQL subset (but not all SQL statements) or APIs.

However, it does not support slightly advanced features, like table joining, due to the limited resources in a mobile phone. Besides, some basic SQL statements, like COUNT and UNION, are not available also. In our case, we make use of Symbian DBMS for convenient data storage and retrieval only. Therefore, those limited features are enough for our application.

Nevertheless, one of the great problems is indexing. We experienced a situation that two completely different entries cannot be inserted into the table when indexing is enabled. It was finally found out that it was a bug in Symbian DBMS that cannot be solved by general developers like us.

8.4.5.3 Bitmap Drawing

It is well-known that ordinary mobile phone has extremely limited amount of memory (e.g. Nokia 7650 has only 4 megabytes of internal memory). Therefore, the storage is usually one of the major concerns of building mobile applications. Although Symbian OS is able to compress the bitmaps in software packaging stage, extensively use of bitmaps may have out of memory problem. Therefore, the idea of bitmap splitting comes as follows:

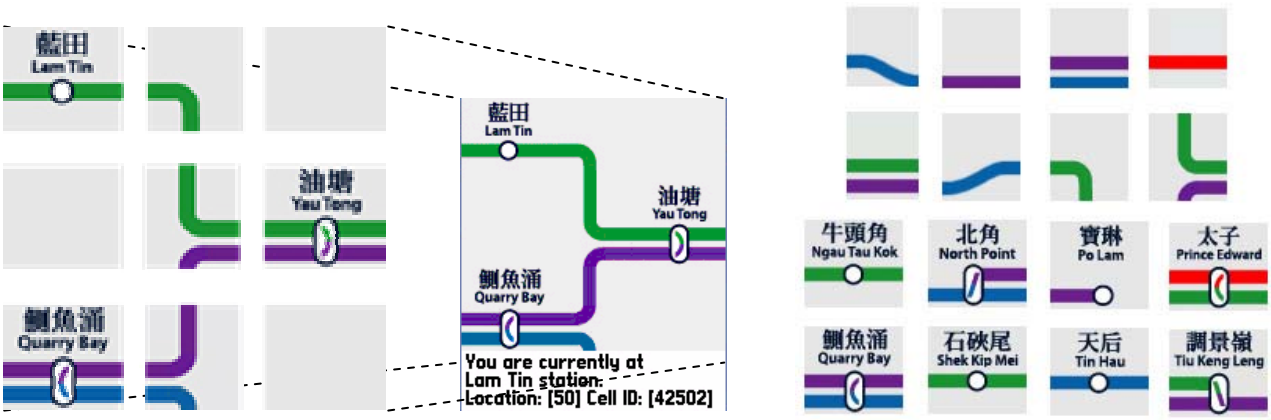


Fig 8.10: How portions are combined (left), station and connection bitmaps (right)

Instead of putting the whole bitmaps of MTR and KCR station maps (e.g. the one in Fig 8.7), the maps are cut down into small pieces of station bitmaps and connection bitmaps in order to avoid unnecessary duplication (e.g. connection between two stations is often a line) and reduce the redundant area (i.e. the area other than stations and links).

The 176 x 208 screen of Nokia 7650 is divided into nine portions as shown above. Different bitmaps are joined together to complete the map according to the input map file.

8.4.4 Potential Problem

In the discussion of various GSM positioning method, it was shown that GSM cell ID positioning is the simplest one and without the support of telcos. However, as all the cells are deployed, maintained and controlled by telcos themselves, the passive method adopted in this project is suffered most by the cell deployment of telco.

Recall that our initial observation is that exactly one cell change in between two stations in subway. In most extreme case, if telco has changed their configuration into a situation that two stations sharing one cell site, there would not be cell change in between these stations. As a result, MTR Traveller would be failed to detect current location.

In addition, we cannot control the time when cell change event occurs. Therefore, for example, a further application is built to show information of the current station and the cell change event happens when the train is very close to the destination station, user cannot browse all the content given.

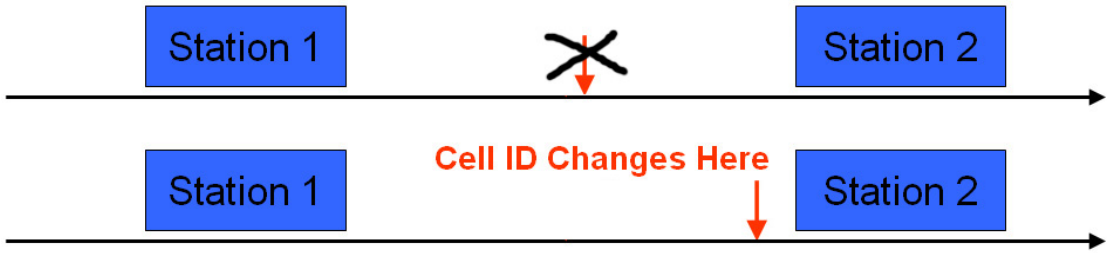


Fig 8.11: Potential problems – cell site sharing between two station (left) and uncontrollable time of cell change (right)

Also, as mentioned before, network operators may change their cell configurations (e.g. adding or removing cell sites) from time to time for efficient cost control. Therefore, an up-to-date cell information is required and this would affect

both parties: application developer / service provider and users. For developers or service providers, they are required to collect cell information regularly and include them in the software package. The idea of automatic cell information collection kit is raised so that overhead of cell data collection is reduced. For users owing the application, their data should also be updated and, hence, convenience update using SMS or GPRS becomes a demand. This may be the future work that we can try.

8.4.5 Benefits of Station Positioning

The discussion so far was around the positioning principle of MTR Traveller. However, what is the practical use of station positioning? In fact, MTR Traveller is just prototype and more should be improved to support different services around station positioning. The below gives several possible examples.

Content Providing: At each station arrival, some information about current station can be provided that suit the user's need, including, but not limited to, shops, tourist spots, weather, traffic condition, air pollution index, etc.

Train Changing: There are several railway lines in MTR (e.g. Island Line, Kwun Tong Line and Airport Express). Tourists may not know exactly where they should take off and change to another line. A personalized application can be built for this purpose.

Notification: People may want to do some other things when they are on the trains, such as sleeping, reading or playing mobile games. If the mobile phone can alert users when the destination is nearly arrived, this can be very useful forgetful users.

8.5 Statistics for Showing the Reliability of Cell Change Method

This part, in fact, is not completely related to MTR Traveller application. The aim of this section is to show the accuracy of proposed cell change method. In chapter 6, it was mentioned that GSM cell change method would provide more reliable location estimation at the cell boundary than traditional GSM cell ID detection. In this section, we would like to conduct an experiment on the reliability (and accuracy) of the GSM cell change using MTR Traveller .

Cell change event occurs when the original registered cell cannot provide enough signal strength so that the mobile phone replaces that cell by a stronger one. For two stations in subway, when the mobile leaves a station, exactly one cell change would also occur due to the same reason. As a result, by recording the time of cell change

occurrence for a number of times (e.g. the red arrows in the following figure), the accuracy of the proposed cell change method can be estimated.

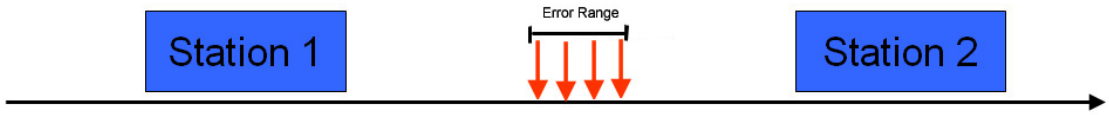


Fig 8.11: The idea of the experiment – finding out the error range as indicated

In the experiment, the time difference at which the cell change occurs and at the moment that the train stops at the destination station (denoted as t_d) would be recorded. The variation between maximum t_d and minimum t_d determines the error range, which is error of cell change method.

The experiment was done in November 2003 for 7 times. Nokia 7650 with MTR Traveller application was used for cell change event detector while another phone, Nokia 6510, was used as stop watch to record the time. The stations involved are some selected stations in the subway from Kowloon Tong to Po Lam. Besides t_d, the average time of journey between two stations was also recorded because the length of this time interval would indicate the size or coverage of the cells involved.

8.5.1 Assumption

The below shows the assumption of this experiment:

1. MTR speed was constant at the error range so that the data in time domain could be mapped to distance domain by "distance = time x velocity" equation. The MTR speed is assumed to be 33km/h, which is the average speed of MTR trains.
2. The cell ID records in the database and input data files were not modified in the period of whole experiment.
3. The measurements were done at the same position inside the train so that all the records would not have undesirable distance mismatch.

8.5.2 Known Source of Measurement Errors

In this part, the time recorded may have possible errors that cannot be avoided and they would be stated here.

1. Because MTR uses automated controlling system for driving, the overall speed in the WHOLE journey varies for each trial although the assumption 1, constant speed at the error range, is still valid. For example, in rush hours, as there are more trains running on the railway, the speed of the trains would be oscillating greatly and they even stop on the way between two stations. This would increase the value of t_d recorded.

2. The stop watch used was not accurate enough. It would only jump by 0.3s each time in the display.
3. Reaction time of starting and stopping the stop watch would contribute to measurement errors.
4. The cell information refreshing interval was set to 1.25s which may cause small uncertainty.

8.5.3 Experimental Results

Let t_j be the time of journey, v_{mtr} be the assumed MTR speed (i.e. 33km/h or 9.167m/s)

The percentage of error range to total time of journey can be calculated by

$$[\max(t_d) - \min(t_d)] / t_j$$

where $(\max(t_d) - \min(t_d))$ is the error range in time domain while t_j is proportional to the cell size.

The error range in distance domain is:

$$[\max(t_d) - \min(t_d)] \times v_{mtr}$$

From	To	Error range in Time (%)	Error Range in Distance (m)
Po Lam	Hang Hau	8.51	65.06
Hang Hau	Tseung Kwan O	15.05	146.65
Tseung Kwan O	Tiu Keng Leng	19.32	127.51
Tiu Keng Leng	Yau Tong	6.50	81.51
Yau Tong	Lam Tin	3.38	32.04
Choi Hung	Diamond Hill	14.41	105.41
Diamond Hill	Wong Tai Sin	8.73	59.62
Lok Fu	Kowloon Tong	31.69	282.36
Kowloon Tong	Lok Fu	15.84	108.17
Wong Tai Sin	Diamond Hill	24.00	132.00
Diamond Hill	Choi Hung	5.61	43.09
Lam Tin	Yau Tong	24.32	178.79
Yau Tong	Tiu Keng Leng	13.93	192.43
Tiu Keng Leng	Tseung Kwan O	10.21	72.44
Tseung Kwan O	Hang Hau	25.47	136.58
Hang Hau	Po Lam	17.54	141.17

(Notice that time recorded in question would not be counted.)

One may observe that the error range is quite different. The error can be quite

large for some of the stations. However, the result is encouraging because the maximum error in distance is less than 300m and it is comparable to GSM E-OTD, one of the positioning methods with accuracy around 150m (see Appendix 1).

8.6 Conclusion

In this chapter, we have investigated the use of GSM cell change method on one-dimensional environment for improved accuracy and reliability under the limitation of fixed path. The application of MTR Traveller was built for illustrating the power of this proposed positioning method. The problem encountered (e.g. the challenge of stations in open area), the cell data collected for stations and the implementation of MTR Traveller were also introduced. However, due to the passive character of pure GSM cell location estimation, any cell deployment or configuration changes by network operators should be regularly considered. The chapter ends with the experiment showing the acceptable reliability of GSM cell change method.

Chapter 9: Building LBS Middleware

The above chapters provide concrete studies on the topics of applying LBS through GSM cell identification. A series of experiments reveals the feasibility of adapting this approach. However, the proposed method is useful in different fields, but not just limited to MTR / KCR route. Therefore, this section would like to discuss a solution to provide different developers with a general LBS development environment – middleware.

9.1 Motivation

The aim of proposing this positioning mechanism is to provide developers a convenient way to develop their LBS applications and share them with a maximum number of users (i.e. all GSM handset users). However, the GSM cell ID mechanism may give a certain burden to developers:

Extra Effort for Developers: Developers have to fully understand the underlying issues, including cell information limitation (e.g. varying cell sizes and cell overlap) as well as cell change event enhancement, before starting coding. Also, they may need to use extra effort to study low-level system calls in accessing cell information as phone manufacturers often hide these functions.

Service as the Main Focus: As the name of LBS implies, it concerns the service rather than a dedicated application. Service designer or content provider would work mainly on service enrichment. It is obvious that cell data collection, analysis and processing are not the issues they would be interested in. To suit the need of service offering, there is no point for them to build all the things from stretch.

Need of Standardization: As mentioned in Chapter 3, as location-based services become more and more popular, it is necessary to define a generic interface between LBS developers and positioning service providers. Such interface is independent of the mechanism behind and, hence, this simplifies programming and enhances portability. For the case here, as GSM cell ID location estimation is not simply for MTR Traveller, the sample application, there is a need to make a generic interface to serve various possible purposes.

Therefore, we need to provide developers with a general and one-stop development experience when developing their LBS application using GSM cell ID positioning method. Middleware allows developers enjoy the benefits of the

introduced method easily.

9.2 LBS Development Streams

The part gives an overview of LBS development with our approach. This also gives the idea of how a middleware can ease the development process.

9.2.1 Major Types of Location-Based Services

As mentioned in earlier chapters, there are different LBS, including content-offering, billing and navigation. Here, two major types of LBS are classified in terms of the location information required:

1) The application generally performs a particular action upon entering or leaving a specific location (e.g. locating where you are, charging you more if you enter a particular region).
2) It searches the nearest point relative to current location (e.g. finding the nearest restaurant).

These two major types of location-based services should both be considered.

9.2.2 LBS Development Process

The following diagram illustrates the LBS development process:

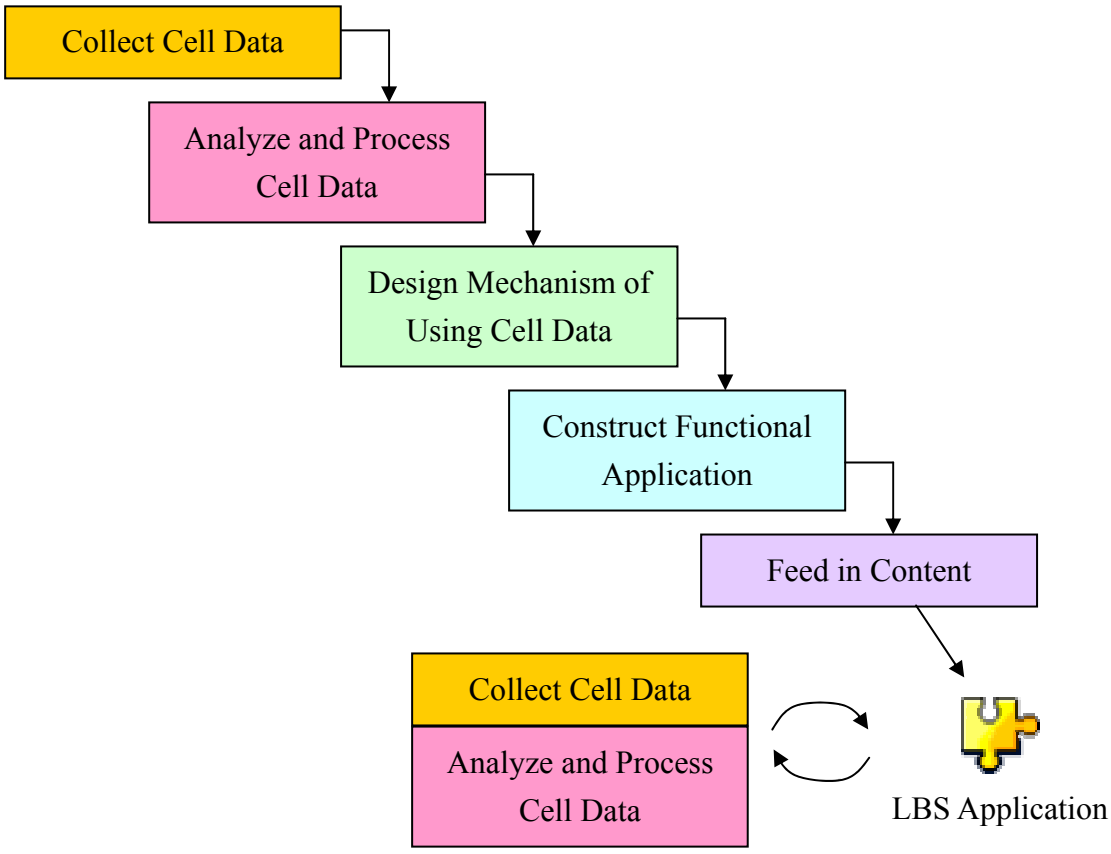


Fig 9.1: LBS Development Process

Cell Data Collection: Cell information is the basic element for the positioning method to work. Developers have to take cell data information by traveling the region of interest, such as MTR route for MTR Traveller, and gather the data.

Cell Data Analysis and Processing: Raw cell data collected should be analyzed before importing into the application. For example, user may remove cell change duplicates and adjust the sequence of cell pair ordering. Moreover, usually data from different network operators are taken in different time. Developer is responsible to combine all these data together into single data file so that application itself does not need to identify which network operators the user rely on (independent of network operators subscribed).

Mechanism Design: A mechanism of using cell data, such as cell change detection, has to be implemented programmatically. This is necessary prior to building an application.

Functional Application Construction and Content Import: Developers can start designing their applications. The word "functional" means that the application can accept content from a dedicated source (e.g. local data file or network). This separates the tasks of software development and content/service offering.

It should be noticed that cell data collection, followed by analysis, should be done regularly as network operators may reconfigure their base station settings from time-to-time.

Therefore, besides giving developers a way to perform handset positioning, complication at each step should be reduced in order to chop down the development effort.

9.2.3: Developers Involved

Concerning a general LBS development, there are different types of developers involved, while each of them has different concerns and requires different levels feasibilities.

Low-Level Developer: These kinds of developers would like to work on GSM cell ID and even optimize the underlying mechanism and algorithm to suit their application needs. They may need a simple interface to retrieve and "play around" with cell information.

General Application Developer: Rather than purely handling cell information, they may need a set of tools to facilitate cell data collection, analysis and distance measurement on a geometrical map.

LBS Content Builder: They would focus on how the service is offered through content enrichment. They only concerns on location changes in order to give relevant response to application users. A simple example is showing a message if users enter or leave a particular region. As a result, what they concern mostly are the information, in form of text, graphics, media, as well as the subsequent actions allowed.

To conclude, there should be a means to not only simplify LBS development process, but also tailor-design the desire of different types of developers.

9.3 Middleware as Solution

Nowadays, software companies usually ship their products, such as library, with a set of toolkits, forming an easy-to-use development environment. Typical examples are Microsoft Visual Studio, Java Sun ONE Studio, Symbian SDK and Emulator. We found that similar approach can be taken in our case – a middleware, consisting of a well-defined APIs and a set of tools for developers to build, and also maintain, a LBS application in a time-saving manner. This can be achieved by:

Encapsulation: Low level function calls can be invisible from developers by wrapping up them into library. A well-defined application programming interface, API, would be provided to developers to work with. Besides, the whole GSM cell ID handling can also be hidden through a set of toolkits.

Automation: Considering the whole LBS development process using cell ID, developers have to collect cell data, process them, build a LBS application for a particular purpose and all of them should be done manually. For example, in tradition, cell data are collected by reading data from phones or linking with extra devices, like PDAs, as mentioned in the section of cell data collection (Chapter 5). There should be some toolkits to automate some of the work, or at least reduce manual processing.

Layering: Interfaces and tools provided should be arranged in layer approach such that developers can view the underlying implementation as a black box when interacting with a particular interface or tool. Developers of different concerns can start their work at corresponding ‘layer’.

9.4 Some Definitions

Before investigating how the middleware provides necessary functions for developers, some definitions are introduced before hand.

9.4.1 Reference Point and Point of Interest

Location-based service usually focuses on a set of regions or points, such as building, shopping mall and tourist spot. A **point of interest**, or POI (PsOI for plural form) in short, represents a particular point on the map of which the LBS application would be aware. For instance, an interactive campus visitor application should have all buildings and canteens as points of interest.

However, as experimented before, GSM cell ID positioning method performs best on 1D path. This implies that the application should consider a path rather than the whole 2D map. Therefore, what the developers concern actually is the **reference point** on a path. Reference points are those points taken in cell data collection process on a predefined path.

To illustrate clearly the idea, an interactive tourist guide along a bus route is taken as an example. The application regards tourist spots interested as PsOI. However, such application may be designed for a particular bus route (or multiple bus routes), so bus stops involved can be considered as reference points as shown below:

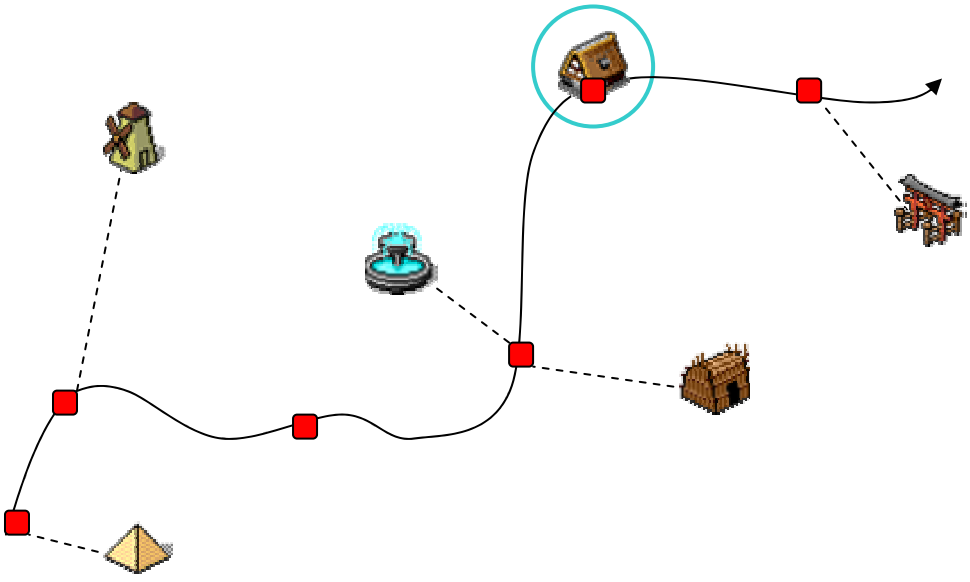


Fig 9.2: Reference Points and PsOI of an Interactive Tourist Guide

Each POI (i.e. tourist spot) has a corresponding reference point (i.e. bus stop) associated along the bus route. Also, a POI may also lie on the route so it is also a

reference point (indicated by the circle).

9.4.2 Location Definition and Distance Mapping

As mentioned in the previous section, user may need to know the nearest target region, such as fast-food restaurant nearby. However, purely with cell data, the application is not able to identify which cell is near or far away from current cell without giving also distance or geometric information. Therefore, it may require an extra step, called location definition, to associate the cell into to physical location, like this:

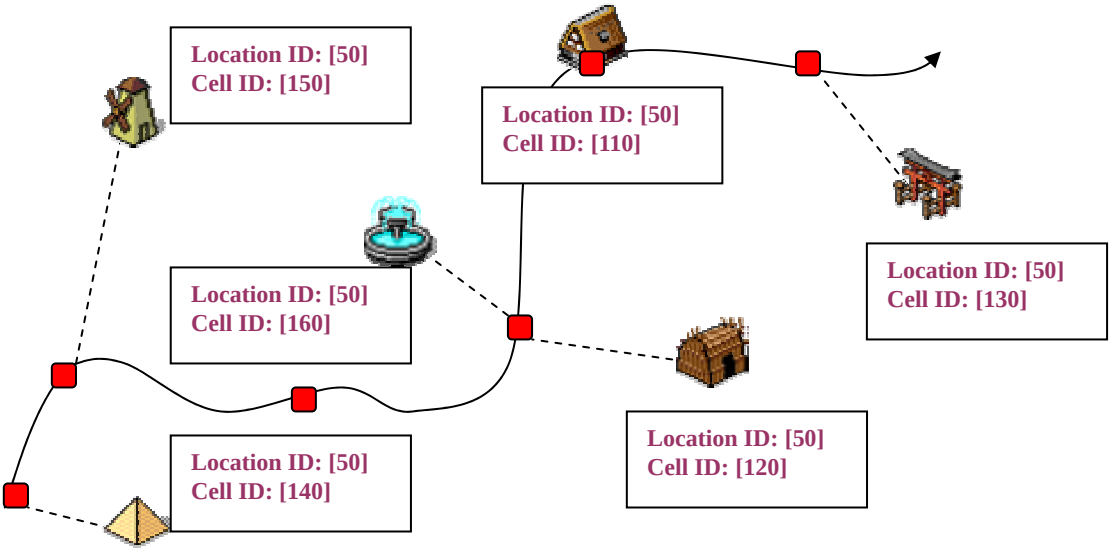


Fig 9.3: Mapping Cell with Geometrical Points on a Reference Map

As a result, the distance between cells can be calculated – distance mapping. It should be noted that what developers do is to approximate the cell because the actual location of a cell centre is unknown. However, this does not contribute a great problem as distances are used to find nearest point of interest but not precisely and accurately identify a location.

9.5 Overview of Middleware Architecture

To tackle each inconvenience given to developers, a middleware accompanied proposed positioning method is designed. The middleware-based development cycle includes two major components: an application programming interface (API) set and a software development kit (SDK).

API set would give programmers a library, with all necessary functions embedded, and a well-defined interface such that they can have work with cell data location-estimation more easily.

SDK contains all necessary tools in building LBS software. It helps to collect, process and analyze cell information. There should also be a tool to facilitate developers in setting the relative distances among those reference points, so that there is a distance definition file to be referred when searching for the nearest reference point. Last but not least, an application generator which creates location-based service application automatically with parameters set by developers. The following figure shows the layered view of the architecture of the development kit.

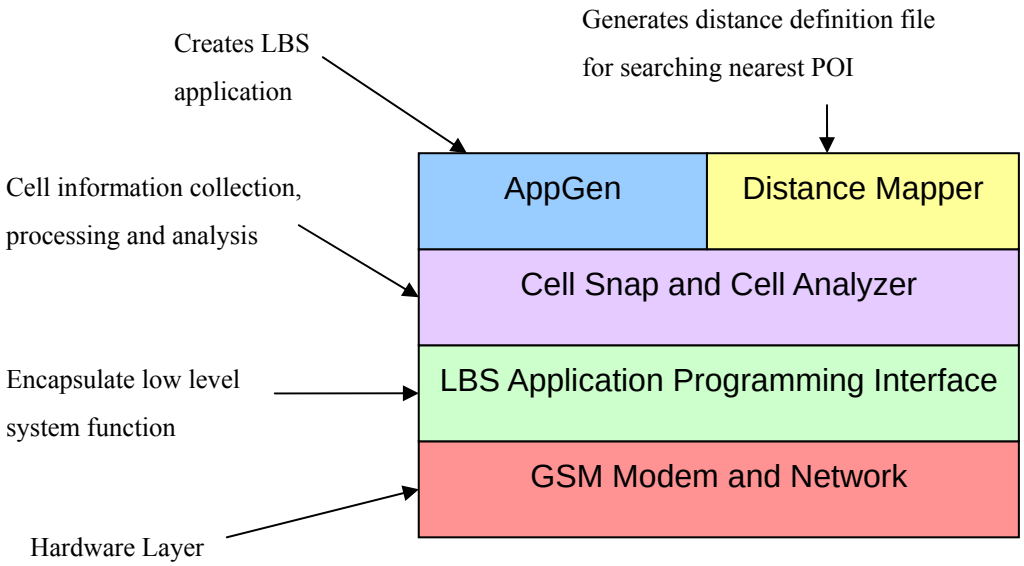


Fig 9.4: Overview of the development kit architecture

As seen from the above figure, the development kit contains three different layers built on top of the hardware layer, which is the GSM modem and network (i.e. sources of cell information).

Just above the hardware layer comes the **API** layer. This layer consists of APIs which encapsulate low level system calls, so that it allows developer to retrieve and manipulate location information without going through complicated steps in invoking those system calls.

Cell Snap and Cell Analyzer are used to collect and process cell information. **Cell Snap** enables automatic cell data collection through written program on mobile phones with camera for taking reference point. **Cell Analyzer** gives developer a convenient way to modify the collected raw cell data. With Cell Snap and Cell Analyzer, developers are now able to collect location information in a handy way.

At the top level there are Distance Mapper and AppGen. **Distance Mapper** helps to map collected location information to their physical location, and it generates a location definition file that will be read by the application generated by AppGen for providing the service of searching for nearest reference point.

AppGen is a powerful application that allows automatic generation of location-based mobile application. Developers can set several parameters such as what type of location-based service is going to be included, what action to be taken upon reaching some points of interest, which location change should be observed etc. This application allows easy generation of location-based application without having to write much code.

9.6 Flow of Development with Middleware

The flow can be completely matched with LBS development process (Fig 9.1). Cell Snap is responsible for cell data collection; Cell Analyzer and Distance Mapper aim to give automated data analysis and processing. The APIs ease developers from building mechanism to handle cell data programmatically; AppGen can even help in building a LBS application and content editing. The following diagram shows the flow of building a LBS application with the middleware.

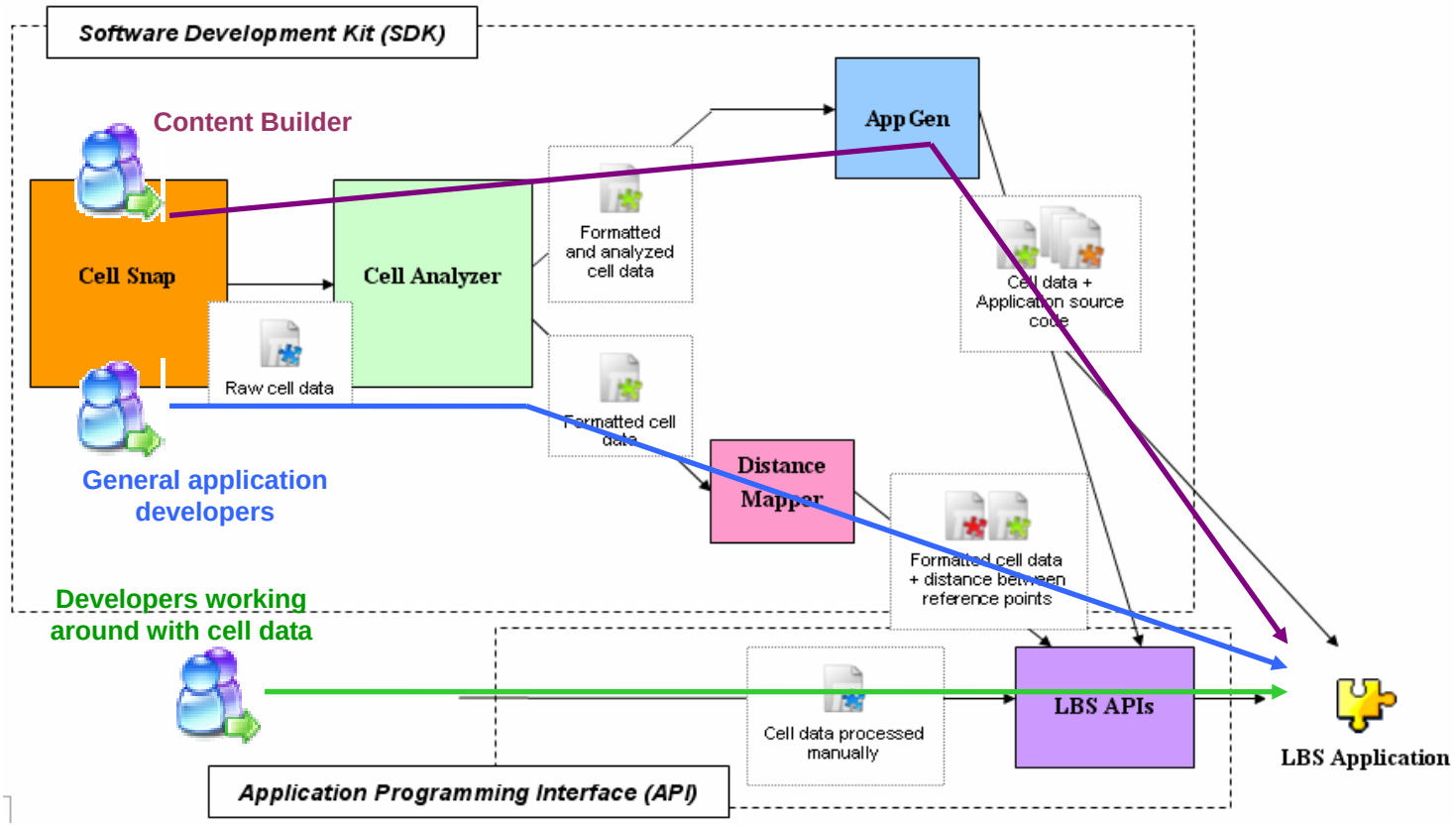


Fig 9.5: Flow of Building a LBS Application through the Middleware

Different developers may interact with our middleware in various entry points. Developers who would like to work around with cell data can find it convenience to adapt into the API given. For general developers, automated data collection and processing are the starting points for application construction. Content builder also has to begin from data collection, but they can modify simply the content and cell change action through AppGen whether building the application from stretch. Each component would be further explained in later chapters.

9.7 Conclusion

To conclude, there is a desire of having middleware to: 1) reduce the time and effort in building LBS application, 2) make the development more service/content-oriented, 3) serve various LBS requirements (but not just MTR Traveller), 4) automate certain manual processing, and 5) satisfy the needs for different types of developers. The architecture of the middleware has also been briefly introduced.

Point of interest, reference point and distance mapping are defined here as these would appear in the next few chapters, which would discuss individual components one by one.

Chapter 10: LBS Application Programming Interface (API)

There are existing system calls that allow developers to retrieve current cell information via GSM Modem. However, calling these system calls can be quite troublesome. We need to connect to the GSM modem, creating an active object for requesting those information and many more. However, all these operations seem to be a must-do when implementing location-based application. In order to simplify the operation of generating system calls in retrieving and manipulating location information, we designed a set of APIs which facilitates the development of location-based application.

10.1 API Design Basics

To write a program for particular operating system, we rely on given system calls and library functions (such as `printf()` for text display or `sin()`/`cos()` for sine/cosine values lookup in C). However, in order to build software with larger scale and more functionalities, a nice API, which contains individual components, subsystem and internal/external interfaces as shown below, should be designed well such that software can be viewed as a group of building blocks.

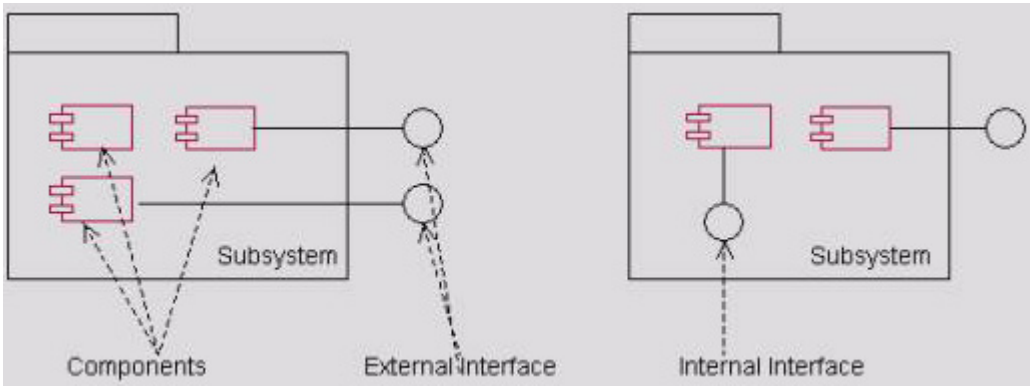


Fig 10.1: Sample Software Architecture [23]

A well-designed API should be thought carefully rather than designed and implemented simultaneously such that the quality of building blocks can be maintained currently and in the future. Also, each of the components should be highly cohesive (i.e. responsible to single function) while low coupling with others in terms of function and data sharing. It would result in high code reusability, portability, scalability as well as extensibility.

10.2 Introduction to LBS API

The API suggested is designed for cell data retrieval, manipulation, callback on

cell changes, etc, so as to save the developers from cell data handling and nearest cell searching, which are the two major classifications of LBS, as discussed in previous chapter. It should tightly follow the above rules so that the API design would be maintainable in the future.

It should be noted that the API design here is independent of the mobile platform used. That is, it is applicable in any platforms with primitive data structures and is not necessarily applied for Symbian OS. The API set implemented under Symbian OS could be regarded as a sample.

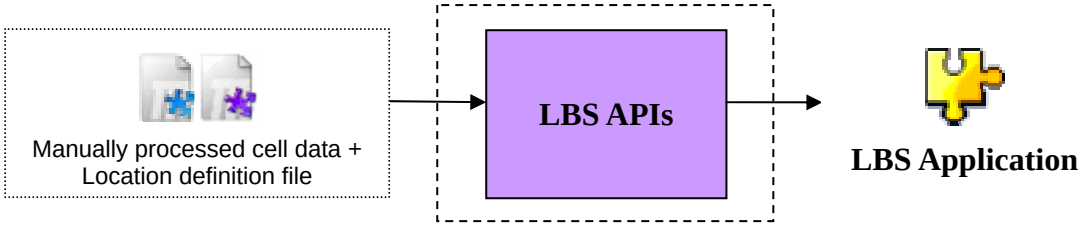


Fig 10.2: Input and Output of LBS API Module

Considering the above figure, developer has to import a list of cell data as well as a location definition file to the API. The cell data list is easy to understand – it is a set of cell data (i.e. location ID and cell ID pairs) collected. Location definition file stores all distances between any two cells so that API is able to search for nearest cell.

Developers may begin building LBS application from here by working around with cell data, cell changes and search for nearest reference points. They could also try to optimize the handling of cell data specific to their purposes.

10.3 Underlying Principles of LBS API

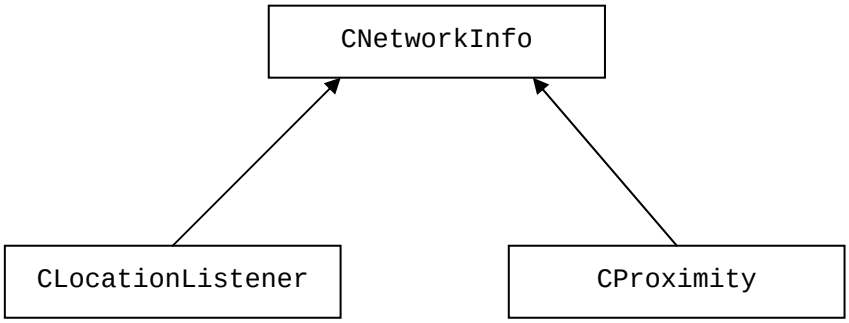


Figure 10.3: Class View of the API Structure

As shown in figure 10.3, our API mainly consists of three main parts. One is

`CNetworkInfo`, which helps to connect to the GSM modem, send information retrieval request and retrieve location information. `CLocationListener` keeps track of location change events and perform a specific action when specific cell change event take place. The last component, `CProximity`, searches for the nearest reference point relative to the current location. These components will be explained in detail in the next section.

These three components should form a complete set for location-based service. In fact, after an in-depth investigation, we find that most location-based service can be divided into two types: one is searching for nearest reference point, the other is keeping track of location change, then perform an action upon that cell change event occur.

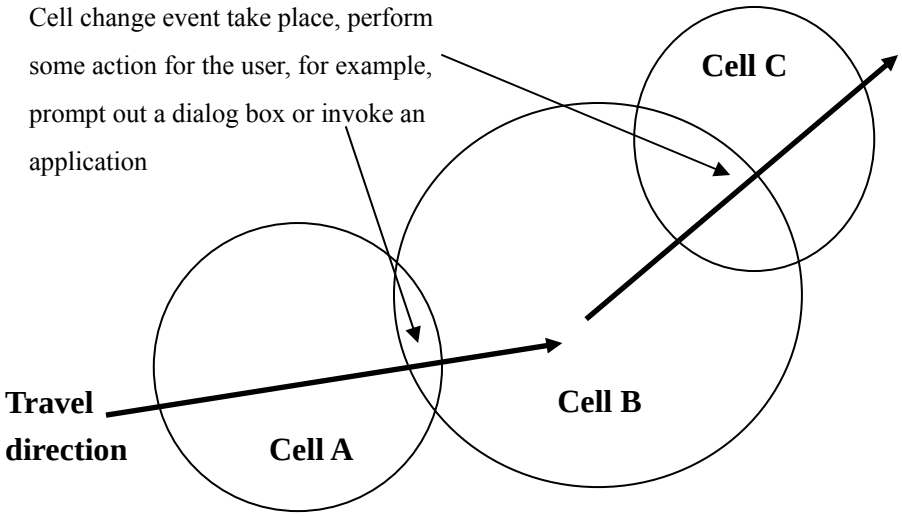


Figure 10.4: Performing an Action upon Specific Cell Change Event Occurs

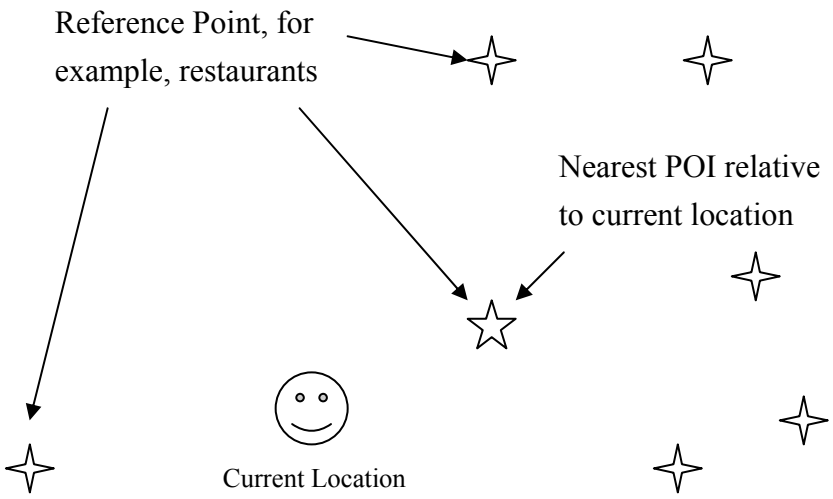


Fig 10.5: Searching for Nearest POI Relative to Current Location

10 3.1 `CNetworkInfo`

As mentioned before, `CNetworkInfo` is used to retrieve location information without the need for developer to perform low level system call. There are a number of functions in this class which can be used by developer while they are developing location-based application. Detailed specification for this API class can be found in the appendix section

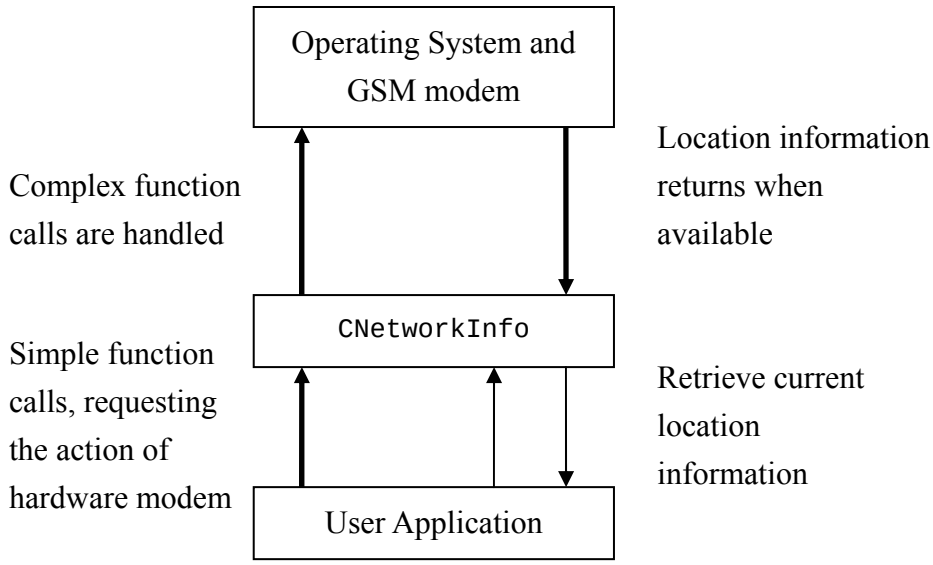


Fig 10.6: The role of `CNetworkInfo`

From the figure above, it clearly shows the role of `CNetworkInfo`. `CNetworkInfo`, once received user application request, sends out a request to the operating system for location information. The retrieval of location information is done in an asynchronous manner. Once a new set of information is available, it will be returned to `CNetworkInfo`. `CNetworkInfo` will store the most update location information and allow user application to retrieve by using simple function calls.

10.3.2 `CLocationListener`

One of the two most popular location-based services is to keep track of cell change events. `CLoctionListener` performs this task for developers.

`CLocationListener` makes use of `CNetworkInfo` in retrieving location information as it needs to keep track of location change event. Since `CLocationListener` keeps track of location change event, so it knows whenever cell change occurs. In order to increase the flexibility for developers, a list of cell ID which the program is interested in should be supplied to it. By referring to the list, `CLocationListener` will be able to perform specific tasks when entering cells that

are specified in the list.

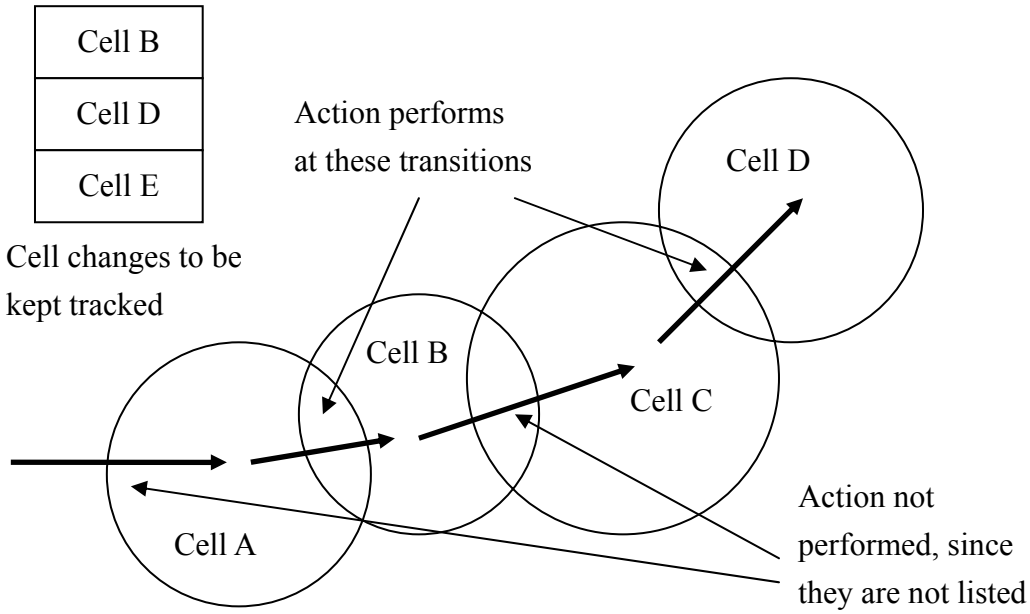
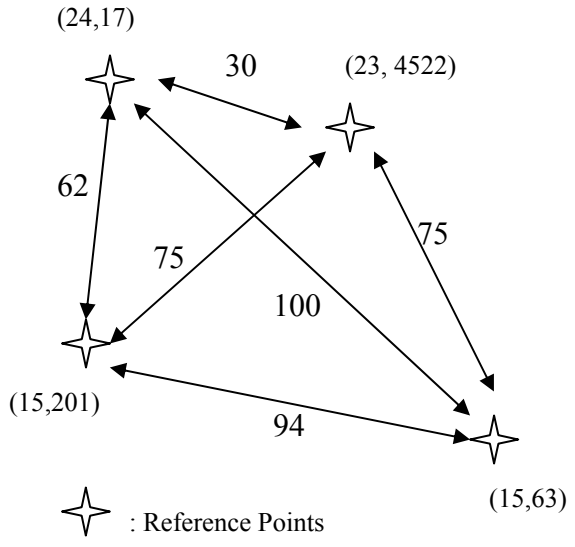


Fig 10.7: How `CLocationListener` Works

10.3.3 `CProximity`

This API class performs the task of searching for the nearest reference point relative to the current location. Users have to supply a location definition file which defines the geographical proximity of all reference points.




```
; Sample location definition file
4
24#17
23#4522
15#201
15#63
; start of distance definition
99999#30#62#100
30#99999#75#75
62#75#99999#94
100#75#94#99999
; end of distance definition
```

Fig 10.8: Physical Distribution of 4 reference Points and its Location Definition File

The above figure shows the physical distribution of four reference points and its

location definition file. In the figure, stars represent the reference points, such as restaurant, bus stop, toilet etc. Numbers in bracket represent the location ID and cell ID pair which the cell owns is located in; numbers on arrows represent physical distance between two cells.

The generation of location definition file can be done manually or with the help of Distance Mapper, which is a part of our development tool kit. We will explain the use of tool kits in next chapter.

10.4 Conclusion

This chapter gives the API design for the LBS provided by GSM cell ID location estimation. The API gathers all cell data handling methods in order to lower developer effort in accessing and manipulating cell information. It is designed for application developers who purely work on cell IDs in a low-level manner.

However, our API requires developer to manually import cell data and distance information between cells. This introduces the need of tools for simplified cell data collection and processing – Cell Snap and Cell Analyzer, which are part of the software development kit (SDK).

Chapter 11: Automated Cell Data Collection and Processing – Cell Snap and Cell Analyzer

As introduced before, cell data collection and processing are the necessary steps in LBS development process. However, these steps often require manpower and, hence, become time-consuming, especially when these must be done regularly to suit the change of cell configuration by network operators.

Also, developers have to deal with cell data from multiple network operators rather than a single one because the LBS application should support all operators with a single data file. At the same time, they have to do some analysis and filtering of collected raw data before use. As a result, the steps of combining cell data from different operators and processing all cell data at once are bothersome, especially when data size is huge.

For example, in Hong Kong, there are 6 operators, namely Orange, CSL, SmarTone, New World Mobility, Peoples and Sunday. Suppose that the LBS application is designed for a bus route with 15 bus stops and there are 8 cell changes in between each bus stop on average. Thus,

$$\text{The number of cell ID involved} = 6 \times (15 - 1) \times 8 = 672$$

Notice that this number is for single bus route. If the application concerns also multiple bus routes, or, for instance, links to other transport routes, the data size become large and it is impossible to process them by human in a quick period. Therefore, tools for automation are so desired.

In this chapter, the operations of Cell Snap and Cell Analyzer, part of the software development kit, are explained in details.

11.1 Cell Snap

In the past, cell data are collected by hand or devices like PDAs as reviewed in Chapter 5. Cell Snap is designed for collecting cell data automatically. This can be achieved now because Symbian OS offers developers with programming capability in accessing mobile phone hardware, including GSM modem, Bluetooth module and camera.

11.1.1 Operation of Cell Snap

Cell Snap is a Symbian program that allows:

1) storing cell change event automatically in a list of triples:
   `(Location ID, Cell ID, Time from Program Start)`
2) capturing photo at reference points such that developers can identify actual locations in between cell data sequence.

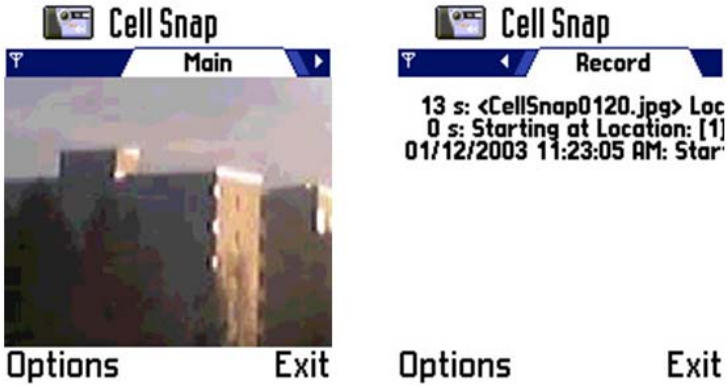


Fig 11.1: Cell Snap Screenshot – Photo Capture (Left) and Cell Data Record (Right)

The following diagram shows Cell Snap as a building block in the middleware. It captures cell data within the journey into a list of cell data and photos (Appendix includes some examples of real data taken by CellSnap).

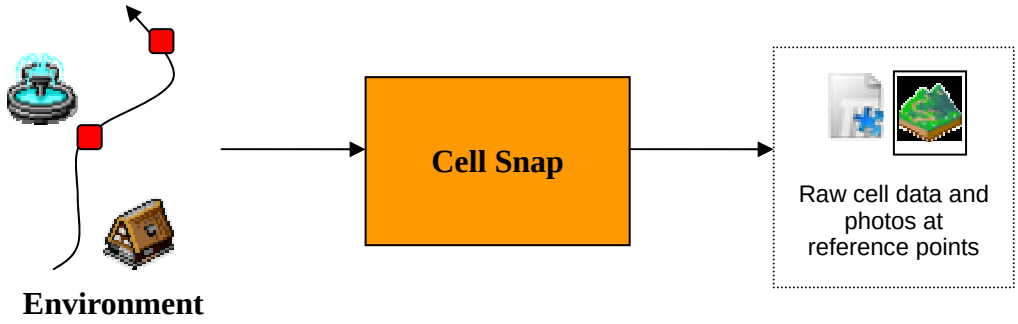


Fig 11.2: Cell Snap as a Building Block

11.1.2 Deficiencies of Cell Snap

Cell Snap is initially designed for small-scale cell data collection. This introduces certain problems:

1. Cell Snap is designed to be read by human, so the Cell Snap output data shows a list of raw cell change event log for human to read, like:

```
129 s: < CellSnap0002.jpg > Location: [ 130 ], Cell ID: [ 33731 ]
321 s: Location: [ 130 -> 130 ], Cell ID: [ 33731 -> 5812 ]
```

However, when there are more data, it is time-consuming to read this by human. Instead, it should be passed to machine to interpret. Besides, the API would require a list of cell data in specified format, so a conversion from this "human" text to concise text is required.

2. Photos and cell data files are stored separately. Developers should look at the content of data file and find the corresponding photo to map on their own. Therefore, it introduces complication to developers.

Also, Cell Snap only captures raw data. It is the responsibility of developers to edit the data by themselves. Therefore, this comes to the birth of Cell Analyzer.

11.2 Cell Analyzer

Cell Analyzer is a utility to edit cell data, combine data from all network operators and optimize the classification between reference points.

11.2.1 Features of Cell Analyzer

The operation of Cell Analyzer can be shown as follows:

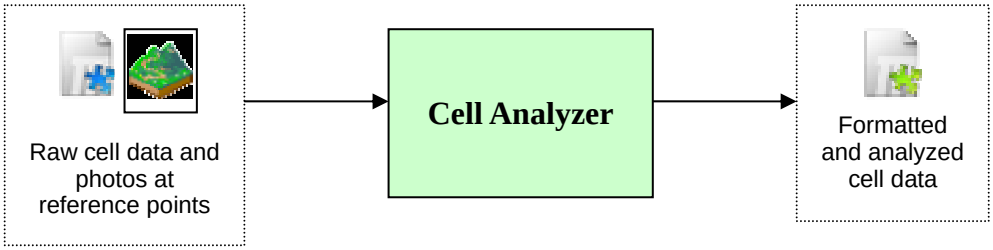


Fig 11.3: Cell Analyzer as a Building Block

Cell Analyzer is written in Java and acts as a user-friendly tool for developers to process cell data. It provides following functions:

1) **Data Format Transform**: Without any modification of data, it simply transforms raw cell data from Cell Snap into formatted data for further use (e.g. as an import file to construct location definition file through Distance Mapper).

2) **Data Presentation**: Cell Snap data is not simply a list of cell data, but it can also be regarded as a tree, with reference points as parent nodes and cell data in between two reference points as child nodes. Besides, in the situation that multiple network operators' data involved in the same route, a 4-level tree structure can be obtained.

Developer may find easier to understand the data through the user interface.

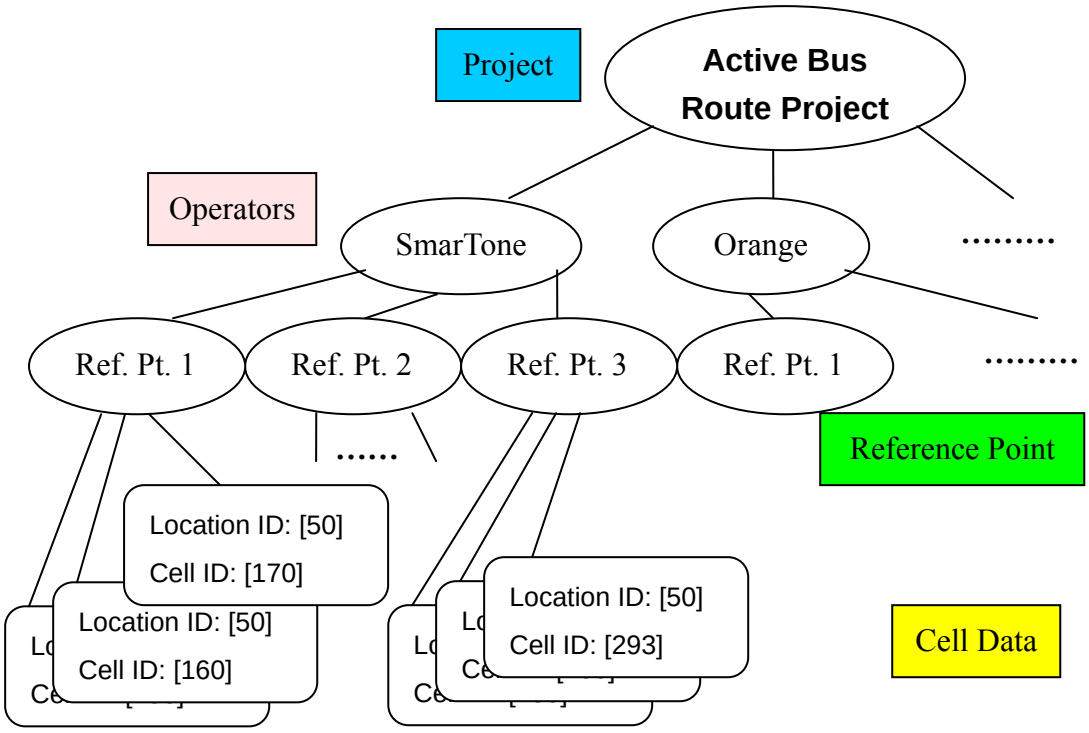


Fig 11.4: Cell Data in Tree Representation

The above is an example of a bus route project. It should be noticed that the leaf nodes are the cell data involved between reference points i and i+1. For example, the cells "Location ID [50] and Cell ID [160]" and "Location ID [50] and Cell ID [170]" are involved in cell changes in between reference point 1 and reference point 2.

3) **Cell Duplicates Removal**: Consider the following diagram:

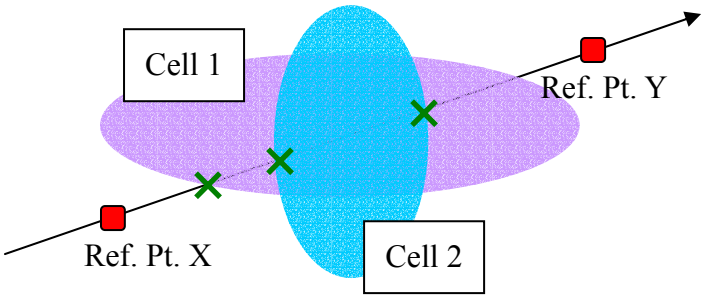


Fig 11.5: Cell Duplicates

There are two cells, cell 1 and 2, in between two reference points (indicated by the dots). Through the given path, the cell changes would be "Cell 1→ Cell 2→Cell 1" and result in Cell Snap data "`X:Cell 1, Cell 2, Cell 1`" (i.e. reference point X

contains Cell 1, Cell 2 and then Cell 1). However, the proposed location estimation method only requires the existence of Cell 1 and Cell 2 in between transition these two reference points. Therefore, the second "Cell 1" can be removed. This can be done automatically by Cell Analyzer.

4) **Intelligent Reference Point Classification**: The origin of this problem can be observed from building MTR cell data in open area. As mentioned before, in MTR Traveller implementation, cell data are classified as 1) station cell set and 2) transition cell set in order to identify the events of "You are in the station" (station cells) and "You are in the way of station X to station Y" (transition cells). Similar optimization is taken in this general LBS middleware.

Consider the following situation in an open area.

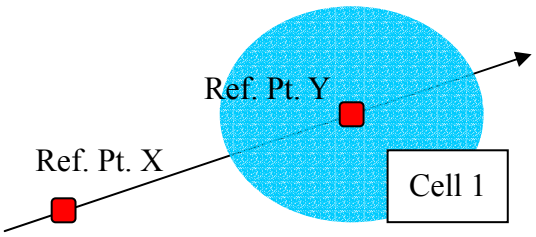


Fig 11.6: Situation in which Reference Point Classification is Needed

Both reference points X and Y contain a node of Cell 1. However, the expectation is, when mobile phone detects cell 1, it should represent the fact of "You are at reference point Y.". Therefore, Cell 1 should be detached from reference point X in order to reflect this condition. Cell Analyzer would be able to classify which reference point a cell should attach to for open area.

5) **Manual Editing**: Developers may manually adjust cell data, such as data removal, if necessary, rather than editing the import file from text editor.

6) **Reference Point Pair-up for Multiple Operators**: Developers can define data from one of the operators as master data. As a result, all reference points from master data are considered as the main reference point set for the entire project. For reference points from other operators, they should be mapped to corresponding reference points in master. Developers may choose data from operator with the smallest number of reference points as master data such that multiple reference points can be mapped to one master reference point. If developers only take some fixed points (like bus stops) as reference points, they may also perform "Auto Pair-up" to quickly map reference

points among all network operators' cell data. The interface is user friendly enough for developers to quickly pair-up data, as shown below:

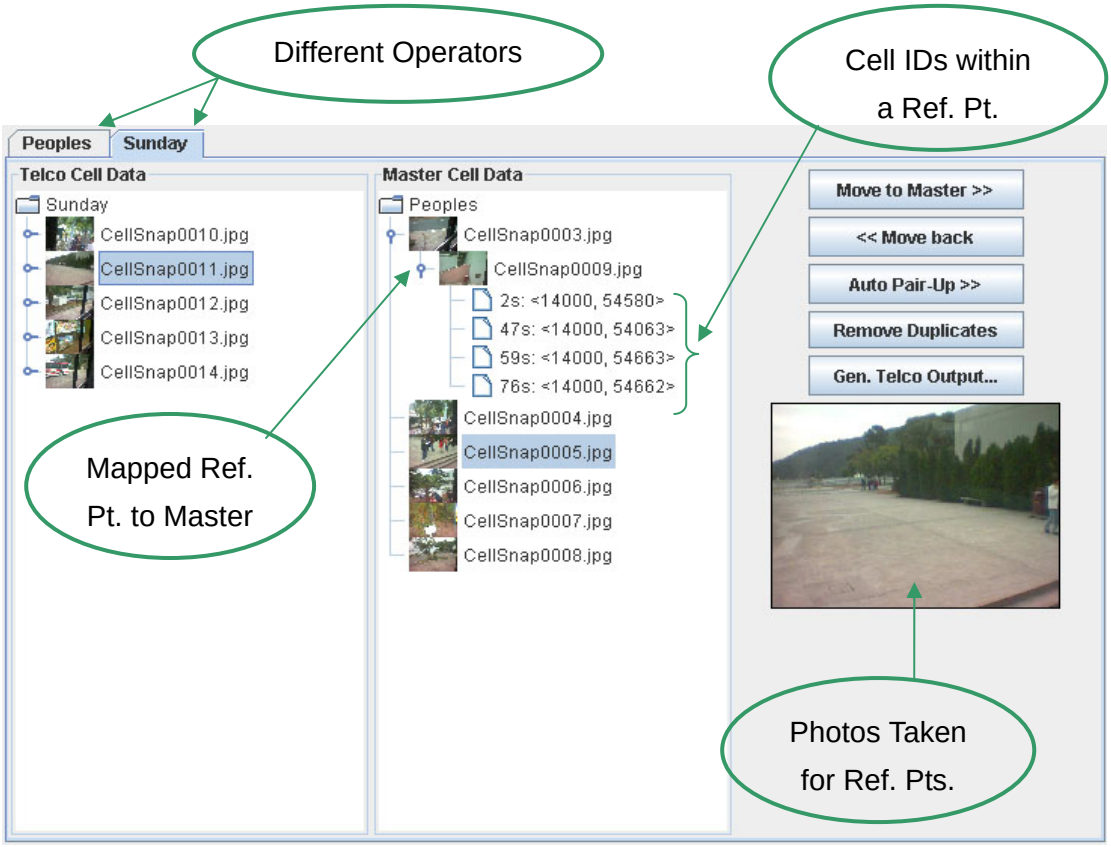


Fig 11.7: Cell Pair-up from Multiple Network Operators' Data

11.2.2 Cell Analyzer Output

As mentioned before, Cell Analyzer would produce a formatted output for Distance Mapper, for assigning approximate locations for each cell, or AppGen, for LBS application generation.

Developer may choose to output a single file for individual operator or combined data for all operators' data loaded. Similar to the case of Cell Snap, some of the real data would be attached to Appendix for reference.

11.3 Conclusion

Cell Snap and Cell Analyzer are the tools in the SDK package. Cell Snap is responsible for data collection and capture reference points by camera while Cell Analyzer provides various automated processing methods and manual cell data editing environment. These toolkits could reduce the development time much for general developers.

Chapter 12: Distance Mapper

As mentioned earlier, two major models for LBS application are 1) location estimation or region entry detection (i.e. detecting the event of entering a region) and 2) finding nearest target from current position. Cell Snap and Cell Analyzer have already eased the developers from the former purpose. However, to know the "nearest" object, merely cell data is not enough.

Of course, developers can handle this problem through their own algorithms and data structures. However, as the middleware aims to provide one-stop solution to assist in building LBS application of general purposes, the API set has already included such feature (`CProximity` class).

The use of Distance Mapper should be introduced right now. CProxmity class uses a location definition file, which consists of distances between any two cells, as input. Those "distances" are not the exact distances, but it would be useful to compare the magnitude in order to figure out which cell (or object represented by the cell) is the closest to current location. Developers may find not only troublesome in manual editing of location definition file, but also frustrated in starting. Especially when number of cells is huge, say N, they have to input N × N entries into the data file. Distance Mapper provides developers with a graphics user interface to handle such task.

12.1 Operation of Distance Mapper

Distance Mapper is one component of the development tool kit. It is responsible for generating location definition file which will be used by API for providing the function of searching the nearest cell. Here is a diagram to illustrate the idea.

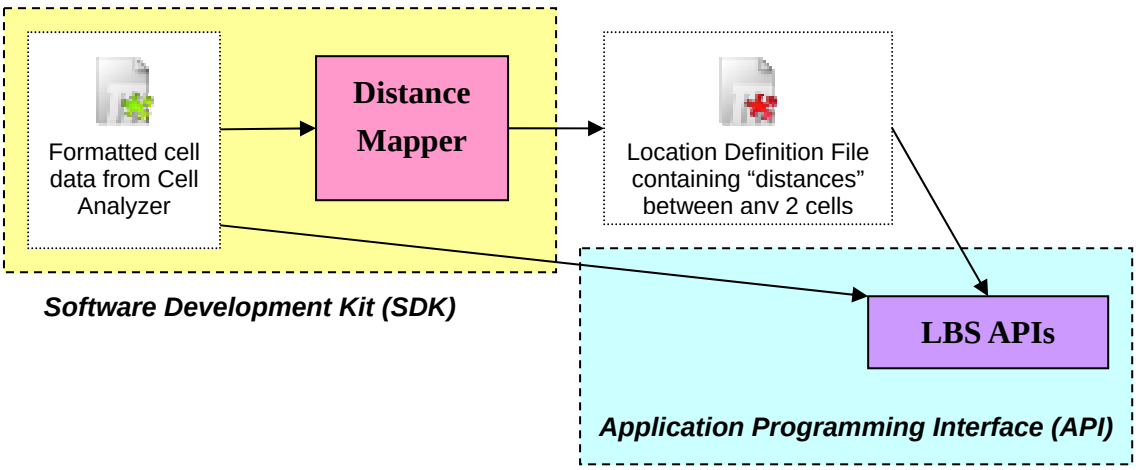


Fig 12.1 Distance Mapper in the LBS Development Flow

The input file for Distance Mapper is simply the output file of Cell Analyzer. Distance Mapper reads in the formatted cell data file and let users assign location information onto a physical map.

Fig 12.2: Distance Mapper in Action

Distance Mapper is written by Visual C++ with Microsoft Foundation Classes (MFC). The above figure shows the layout of Distance Mapper. On the left is the working space, a physical map can be loaded into the working space so as to map a cell onto their physical location. There is a tool bar on the right for users to load location information and physical map there. The top part of the tool bar is a list showing the cell data retrieved from cell data file from Cell Analyzer. Users can map those data onto the map on the left-hand side.

After mapping location information onto the physical map, the program generates the location definition file which records the logical distance data among all cells involved.

The operation of generating the location definition file is as follows. After mapping all cells onto the physical map, the program calculates the relative logical

distance in the pixel space. We simply use the distance equation in calculating the relative logical distance.

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

where x_i and y_i are the x-coordinate and y-coordinate of a location information in the pixel space respectively. Since the API only concerns which cell is closest to the current location, there is no point in calculating the exact physical distance. Relative logical distance can fulfill the need of developers.

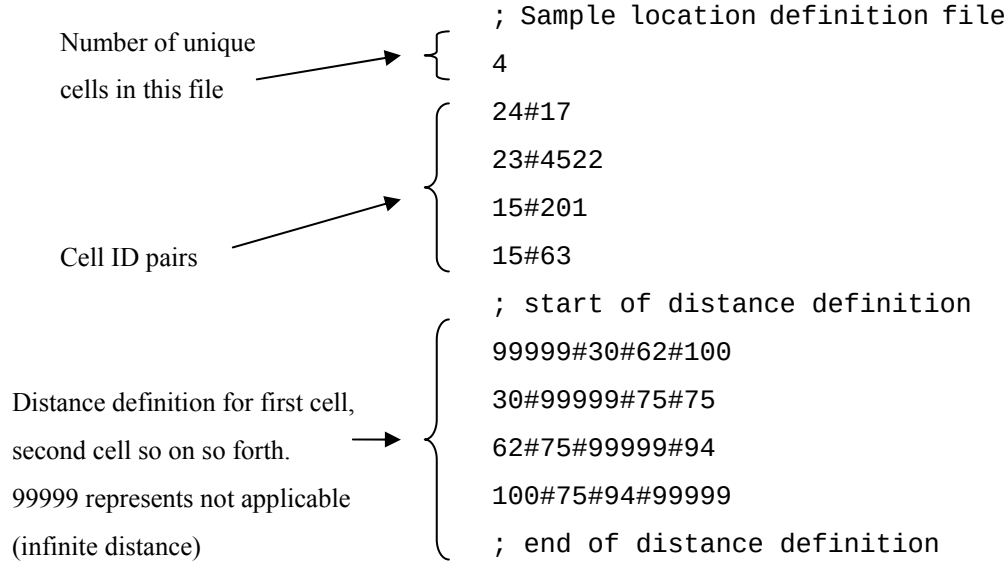


Fig 12.3: Sample Location Definition File Generated by Distance Mapper and Its Explanation

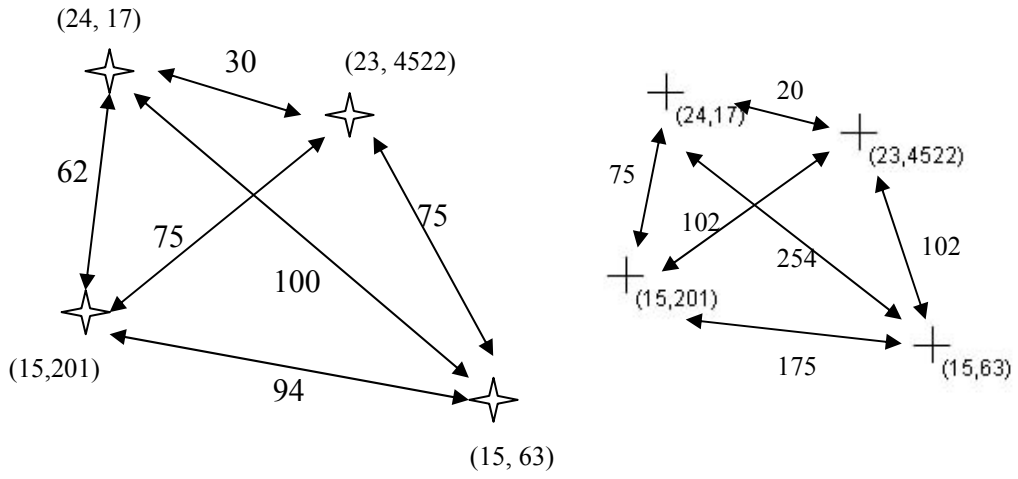


Fig 12.4: Cell Distribution in Actual Environment (Left) and Pixel Space (Right)

The above figure illustrates the difference between exact physical distribution and logical distribution (stars are the cells involved). It is found that the distributions are not necessarily the same - both geometry and distance can vary. Provided that the degree of separations among all cells is preserved, API can still distinguish the nearest cell.

The output file will be imported to LBS API for creating an LBS application.

12.2 Problem Associated with 2D Distance Mapping and Improvement for 1D Path

Distance Mapper can generate "distances" according to the background bitmap used. However, by having only cell data, general developers have to approximate the locations of the given cell ID pairs without concrete proof of the correctness because they do not know where the actual base stations are setup.

The GSM Cell ID positioning method focuses on 1D path for higher accuracy. Similarly, distance mapping can be designed better specifically for 1D path.

As mentioned before, Cell Snap stores entries in triples: `(Location ID, Cell ID, Time from Program Start)`. Assuming that the speed of traveling when collecting cell data is constant (or slightly varies), time from program start could be taken as factor to determine the distance. This parameter can be used as cross-checking measure in assigning location to cells.

12.3 Extension of Distance Mapper

Usually, through Cell Snap, reference points are also recorded (by photos) and their locations are well-known on a map. In LBS application, a map (a 2D image) is usually displayed also, so it would be useful distance mapper also exports the x,y-coordinates in pixel space of the reference map used in Distance Mapper.

It is not a must to have such output for the API. However, it is an optional feature that may suit some application programmers' demands.

12.4 Conclusion

As a part of LBS SDK implemented, Distance Mapper assists the developers in creating location definition file for the API to determine the nearest cell (objects). Together with Cell Snap and Cell Analyzer, it makes the life of general developer easier.

Chapter 13: LBS Application Generation: AppGen

The previous tools mentioned enables general developers to have a nice LBS application development environment. However, usually LBS application focuses on service rather than a piece of software. Therefore, application development may not be the major concern for content builders. Also, some developers may need a template as the starting point of building LBS application.

AppGen is specifically designed for content builders who concentrate on content provided and actions performed when application users enters or leaves a region. As the name of AppGen implies, it generates source code of a multi-functional LBS application using the given GSM Cell ID positioning method, while AppGen users can edit the text, images, etc, prior to source code creation.

13.1 Operation of AppGen

AppGen refers to data from Cell Analyzer to define the reference points as follows:

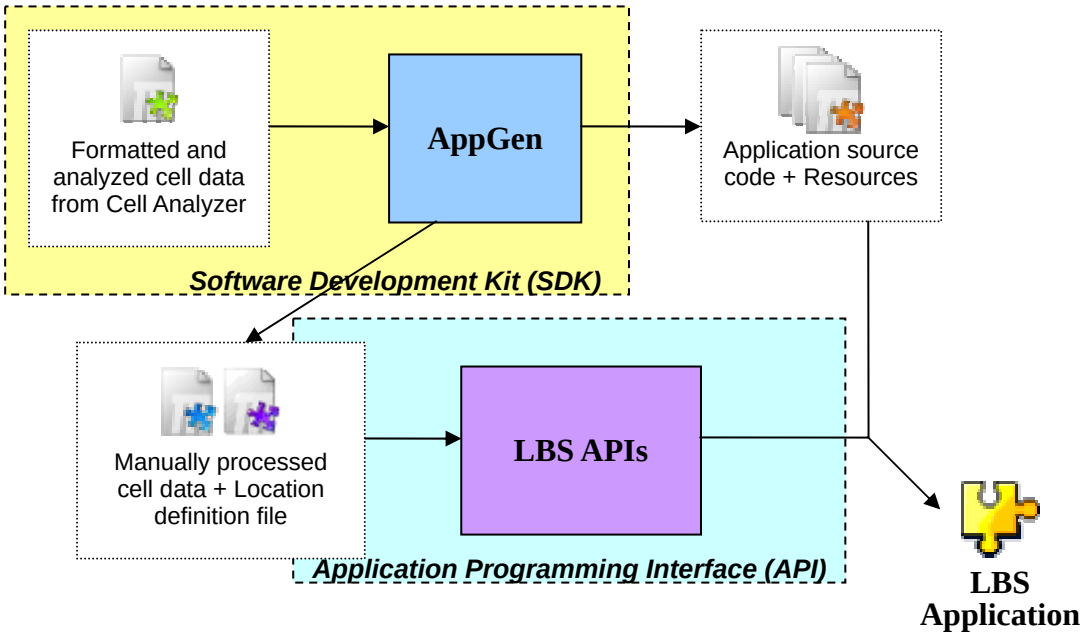


Fig 13.1 Role of AppGen in the LBS Application Development Flow

The role of AppGen is to generate source code for the application on the top of LBS APIs, so it requires cell data and location definition file for the code to work. At the same time, it also manages all resources involved in the application, including map, photos and icons.

As a result, content builders can modify the content while developers may consider the generated source code as the initial material to start build their own application.

13.2 Features of AppGen

Based on the assumption that content builders may not know deep in Symbian programming, AppGen aims to offer as much flexibility as possible to content builders (i.e. they can modify parameters and add new content in various formats) and ensure that changes provided should be closely relevant to content editing. Thus, AppGen provides three types of options for content builders, namely general options, reference point settings and point of interest (POI) settings.

13.2.1 Primary Functions of the Application Generated

One should understand the functions that the application provided in order to decide the entire content needed. The application generated would cover as much as possible that a general LBS application can do, including:

1) to keep track of interested cell changes and report to user (e.g. entering a new station for MTR Traveller) and;
2) to let user to specify a destination and acknowledge the user when he/she reaches a reference point that associates with that destination (Those destinations are usually defined as point of interest (POI) here).

13.2.1 General Options

The following screen shot shows part of the general options provided.

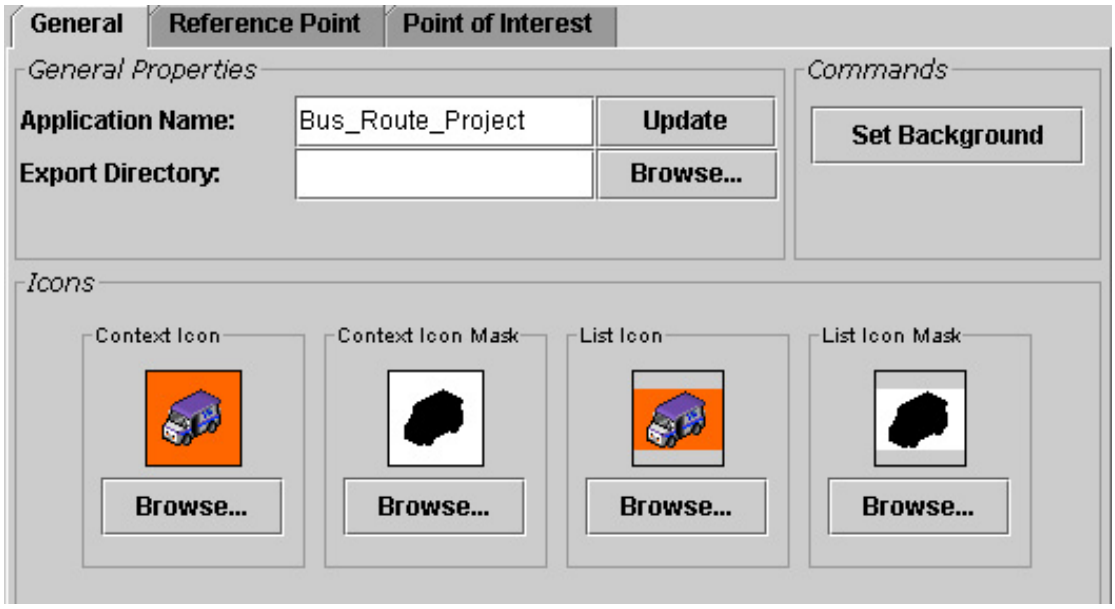


Fig 13.2: General Options Provided in AppGen

AppGen users can set their own application name for both display purposes and source code generation. If the application name provided is Bus_Route_Project, then the class names becomes, for instance, `CBus_Route_ProjectAppUi`, `CBus_Route_ProjectContainer`, etc.

AppGen users may also set their own map and icons used in the application. Different image formats are allowed, including JPEG, PNG, GIF, etc.

At the same time, content builders can edit the message displayed in different events, including detection of new reference point, startup, cell ID display format, etc., as shown below:

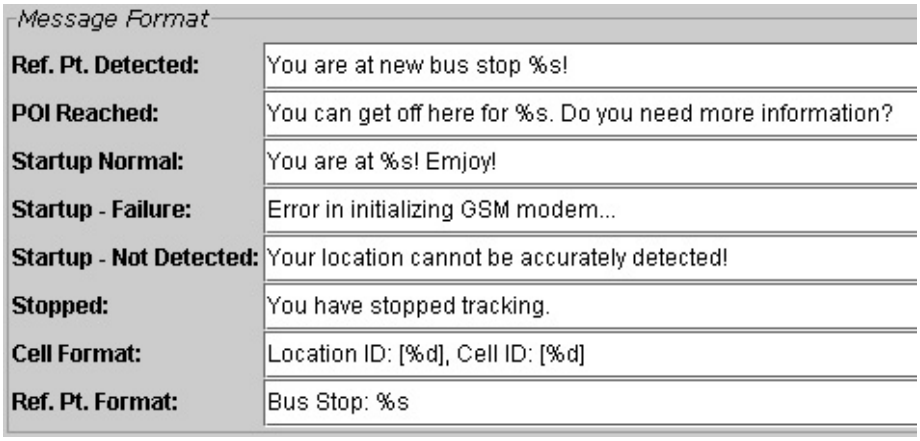


Fig 13.3: Message Editing

13.2.2 Reference Point Settings

Reference points are points taken in Cell Snap and further processed by Cell Analyzer. Usually, they are some fixed points in the target 1D path. The application generated would keep track of these reference points. When user enters a new reference point (or corresponding region), certain actions are taken, such as changing the display at the bottom of the screen or showing a non-modal message box.

Content builders can also associate a pixel point from the map to a reference point such that the screen would change the display (with that map point as centre) when users enter this reference point.

Fig 13.4 Selection of a Point on the Map for a Particular Reference Point

Most importantly, content builders may add certain point of interest (POI) for each reference point (In the starting chapter of middleware series, POI is first introduced). POI is a point or region, of which the application should be aware, such as Engineering Building in CUHK and University Library. They are not necessarily on the 1D path interested.

By default of AppGen, all reference points are also points of interest. AppGen users can add new points of interest within a reference point. Take MTR case as an example: developers have to map all important constructions around the corresponding MTR station as points of interest.

13.2.3 Point of Interest Settings

Most of the configurations are for POI. Content builders are allowed to change the name of POI (most likely the name of particular building or spot) and a description for it, including POI brief description as well as, upcoming news/events of this POI. At the same time, content builder can supply the application with photo of each POI so that application users can have the idea about the appearance to go to there.

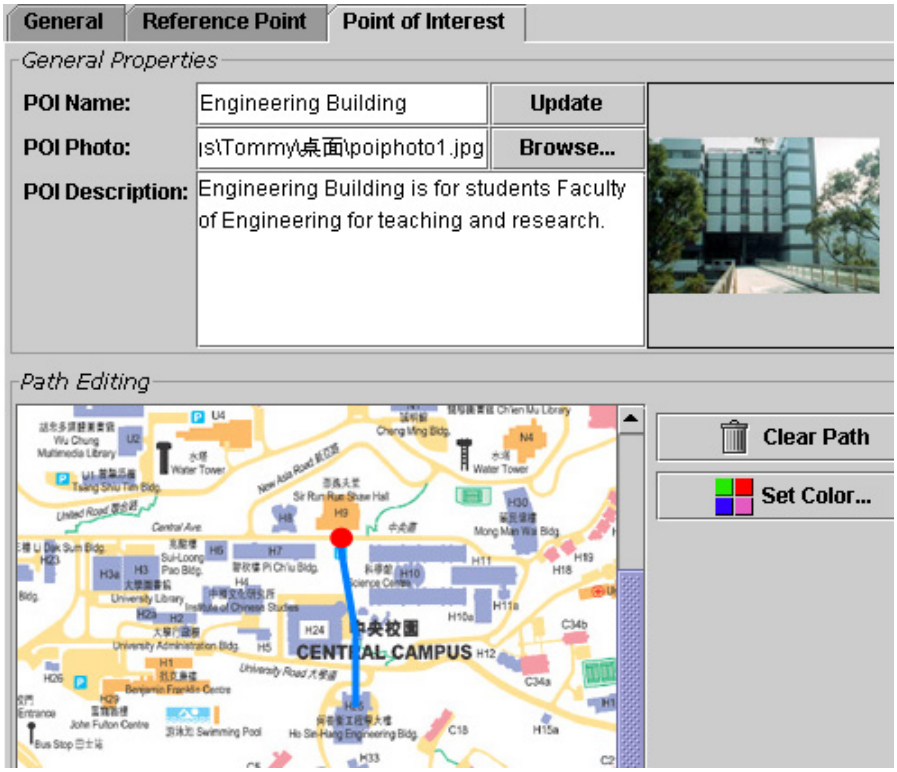


Fig 13.5: Interface for a Point of Interest

Moreover, developers are able to draw path from the corresponding reference

point (e.g. bus stop) to the target POI. For example, the above diagram shows how AppGen user drew a path from a reference point (Central Campus bus stop in CU) to a POI (Engineering Building in CU). This is useful to build a tour guide in the resulting LBS application – telling mobile users to go to a destination from certain well-known points.

13.3 Source Code Generation

After users have finished their configurations, they can start creating source code line by line by AppGen application. Also, all images are transformed into BMP format for Symbian application.

The following screenshot shows an example application generated without further modification on code. As one can observe, the application can consistently keep track of the change of location ID and cell ID and present user a path from a reference point (indicated by the big dot) to a target destination.

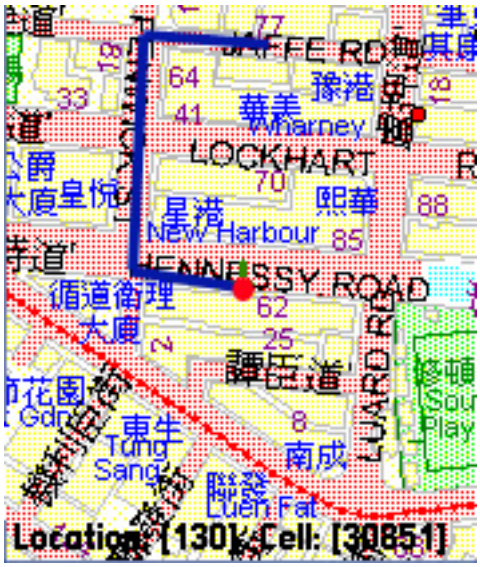


Fig 13.6 Sample Application Generated

The generated application does already have enough functions to build a general LBS application.

13.4 Conclusion

AppGen provides content builders with user interface to create their applications without writing any code. With control of reference points and points of interest, message formats, POI presentation in detailed text and photo as well as path to destination, generated application would be capable to give enough functions to build a general LBS application. Alternatively, application developers can start building an application from the source code generated.

Chapter 14: Experiment on LBS Middleware: MTR Traveller Remake and CU Campus Bus Route Guide

In this chapter, two sample applications built by the middleware are presented. MTR Traveller remakes show how the middleware can handle the same task in an efficient way. CU Campus Bus Route project demonstrates the how the package can apply in a real situation.

14.1 MTR Traveller Remake

Here we would like to recreate the MTR Traveller application (with similar behaviour) from stretch (i.e. from data collection to actual application running on the mobile phone) through the LBS API and SDK provided.

14.1.1 Difference between MTR Traveller and AppGen-Generated LBS Application

There are several differences between these two applications in terms of underlying mechanisms:

1. As mentioned in the chapter for MTR Traveller, it depends on the query to Symbian DBMS. However, both of our applications do not have parallel queries to the database. Moreover, Symbian DBMS does not support table joining and some of the SQL statement like UNION. As there is no significant performance and functional gain in using DBMS, the later LBS application simply gets rid of DBMS usage.
2. MTR Traveller has a dedicated algorithm to handle open area problem by classifying cells into transition cells and station cells. As a result, developers have to submit another data file to specify whether a cell is a transition / station cell or not. On the other hand, Cell Analyzer, the cell data processing tools, does have handled such problem by the reference point architecture (i.e. grouping cells under a reference point (MTR station in this case)). The following two figures illustrate the difference in data structures

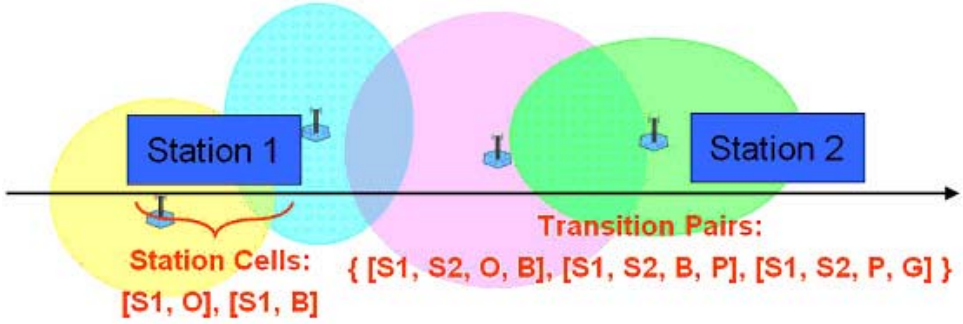


Fig 14.1: How MTRTraveller handles open area stations

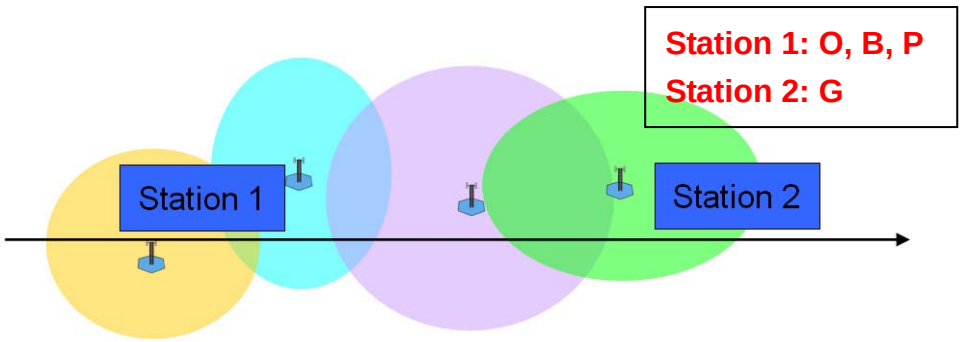


Fig 14.2: How AppGen-Generated application handles open area stations

3. MTR Traveller accesses to GSM modem directly through function calls while the later one is built on the top of LBS API.
4. In order to save storage, MTR Traveller chops the MTR / KCR map (in bitmap format) into tiles of stations and rails. However, as the AppGen-created application is designed for general purposes, there is no such storage optimization (i.e. the whole the map is stored rather than constructing the map from tiles).

14.1.2 Data Collection and Processing

With Cell Snap, MTR / KCR data can be simply recorded by taking photos at each station as reference point. Although collector still has to travel all involved stations, he/she does not need to manage how cell information changes.

Concerning data processing, MTR Traveller requires developers to:

1. type in collected data to a text file,
2. transform cell data into specific format for the program,
3. identify which cells are station cells and which cells are transition cells,
4. edit the data by themselves and,
5. combine data from different network operators manually.

Compared with processing with middleware, developers simply interact with Cell Analyzer user interface and get most of the things done automatically.

14.1.3 Application Generation

MTR Traveller was built by writing a Symbian program from stretch. Then, developers have to determine which tiles should be mapped for all stations by finding the pixel coordinates by imaging software.

On the other hand, the later one is created by AppGen. As the concept of point of interest does not exist in MTR Traveller, developers can generate the application by

just importing file from Cell Analyzer, entering the project name: `MTR_Traveller`, assigning x,y-coordinates of the map to each station, and, optionally, importing the icon specific to MTR Traveller. The whole process is less than 10 minutes.

The screenshots of two applications are shown in the following figures:

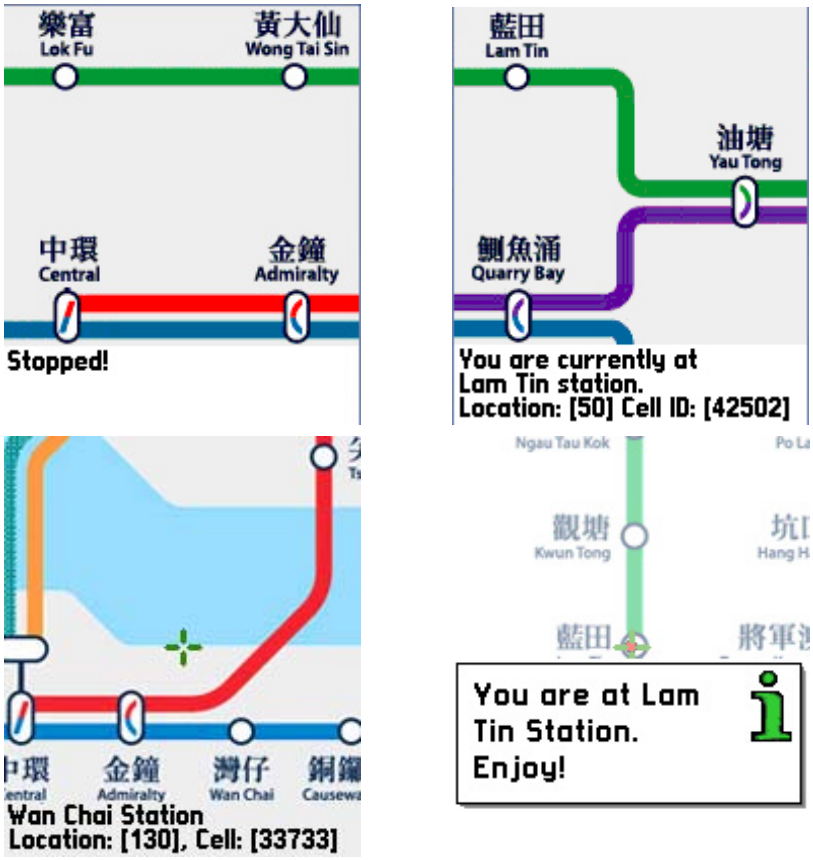


Fig 14.3 Screenshots for MTR Traveller (Top) and its remake version (Bottom): Comparison in general MTR map (Left) and arriving a new station (Right)

14.1.4 Comparsion between Two Applications

Actually, both of the two applications can have the similar interface and operations. In this section, they are compared with each other in terms of time required for development, package size and application extensibility.

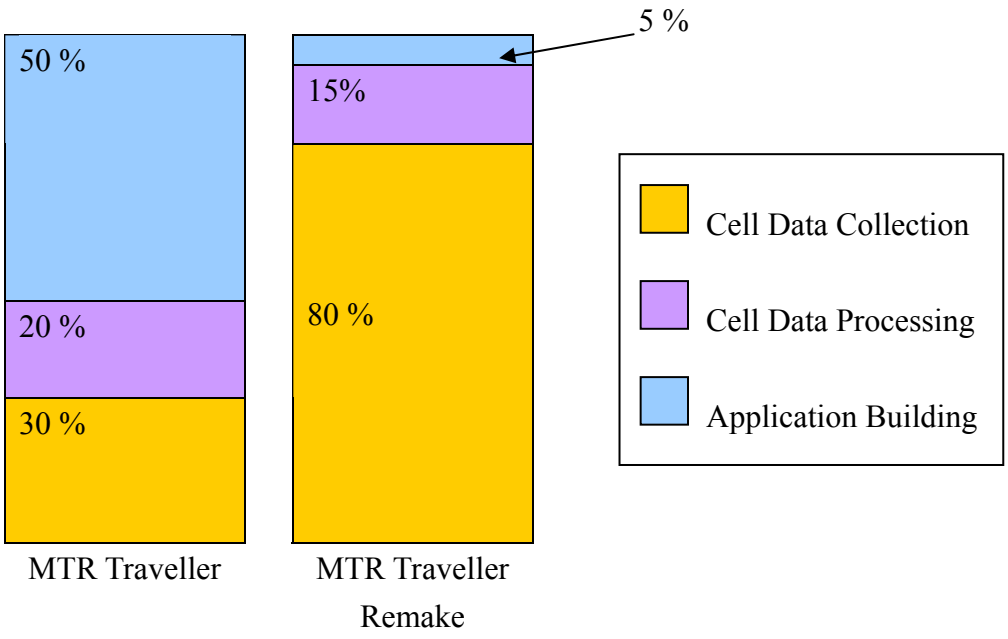
14.1.4.1 Time Required for Development

Both applications contain the data in three of the six MTR routes, namely Tseung Kwan O Line, Kwun Tong Line and Island Line. MTR Traveller required 3-day development with a week for testing and optimization (i.e. 1.5 week in total) for 2 telcos, SmarTone and Peoples.

On the other hand, the remake version was created in less than 8 hours, including

data recollection, processing and application building. This shows a significant advantage in time and effort needed for building similar application.

Another interesting point that should be discussed here is the time distribution in each process. The following bars show such factor:



To build MTR Traveller, developers should spend most of time in both data collection (and usually have to collect many times) and application development. However, data collection becomes most significant time contribution in remake version because there is not difference because both cases requires collector to bring the mobile phones to travel around all stations involved. Possible solutions are to put mobile phone devices in MTR trains or to take data from users afterwards when they are using the application.

14.1.4.2 Package Size

The resulting package sizes of two applications can be summarized in the following table:

	MTR Traveller	*MTR Traveller Remake*
Program Binary (i.e. .app file)	76 KB*	60 KB
Package (i.e. .sis file)	42 KB	122 KB

* MTR Traveller binary size is slight larger because it embeds with GSM Status (the program that displays currently registered cell information and signal strength) also. This information is not the concern of the discussion here.

The binary size of two applications can be regarded as similar. However, the package size (i.e. the total size of all files involved, including binary, icons, data files

and images) of the remake version is significantly larger than the original one due to the use of full-size bitmap for the MTR route map, compared with bitmap tiles in original version of MTR Traveller.

14.1.4.3 Application Extensibility

As we know, an application is not designed only for current use. In the future, there may have necessary modifications in order to fit the reality. Consider a real case that a new MTR station, Nam Cheong, came into service on 16th December 2003. In order to capable to handle this new station, the following changes are required for MTR Traveller:

1) collect cell data for this new station,
2) add this entry in the map data file and cell data file manually and,
3) make a station bitmap tile for Nam Cheong station, rearrange the map array value in the resource file as well as recompile the program.

Considering AppGen-generated MTR Traveller, what developers need to do are to:

1) collect cell data for this new station and regenerate a new data file and,
2) replace the new map.

Of course, both applications require developers to recollect the new data. However, MTR Traveller may require recompilation of the program because of the change in resource file, resulting in the situation that the application should be redistributed to users. On the other hand, the later application requires only to replace the bitmap and this can be done through network update without program redistribution. This case middleware provides a nice support when an application extends in terms of time and dependency.

14.1.5 Conclusion for MTR Traveller Remake

With the help of middleware, developers can reduce the time in producing a LBS application. Meanwhile, the application is more easily to be managed and extended.

14.2 New Application – CU Campus Bus Route

This application is created also through the help of middleware. This time, we had collected cell data from 4 network operators, namely SmarTone, Peoples, Sunday and Orange, and it was created at the end of March, after finishing the implementation of all middleware components. The bus route involved is the one from University KCR Station to New Asia College (Appendix).

Besides the major function of MTR Traveller, this application aims to be more informative so that it can be a complete and interactive CU campus guide. For example, it would show the details and upcoming events (e.g. seminars) within different locations in the campus, photos of each buildings and path to these destinations.

14.2.1 Data Collection and Processing

We recollected data in CU bus by Cell Snap, instead of applying cell data from experiment at the end of 2003. It is because we merely own campus-wide data for SmarTone and People only and, most importantly, we would like to show the whole process of building a specific application from the very beginning. We have taken all bus stops as reference points. Appendix contains more detailed data files.

The data processing here is completely handled by Cell Analyzer without manual editing (i.e. Cell Analyzer would remove duplicates, intelligently cut reference point, etc.). This process is similar to MTR Traveller, but it requires extra work to map data from different telcos into a single output data file.

14.2.2 Application Generation

AppGen is responsible for application generation for our CU Campus Bus Route Guide. Additionally, we had added a lot of content using AppGen fron-end. To consider major buildings of the campus, we inserted those buildings (as POI) to corresponding reference points (i.e. campus bus stops). Also, building names, descriptions, events, photos and the paths from a bus stop to destination location are also considered.

The resulting application is shown below. Users have to select the destination before starting tracking (of course they can select destination later on). Once the corresponding bus stop has been reached, related information would be shown up.

Fig 14.4: Destination Selection (University Gym has been selected)

Fig 14.5: Starting from KCR Station (Left), the mobile phone comes to CC Hostels bus stop eventually (Centre) and shows up information, photos (Right) as well as path from bus stop to University Gym (Centre)

The total development, including cell data collection through traveling with campus bus, processing, building information collection (including photos) and content editing, was done within 1 day which is appreciable.

14.2.3 Potential Problem

As mentioned, we have tested the application with 4 telcos. However, it was found that these four telcos behaves differently. Actually, it represented the same potential problem found in MTR Traveller (imagine when two stations share the same cell, although it does not actually occur from right now).

The most considerable case is Sunday, one of the telcos in Hong Kong, where it had only 3 to 4 cells in the campus, where there were 6 bus stops in the target route.

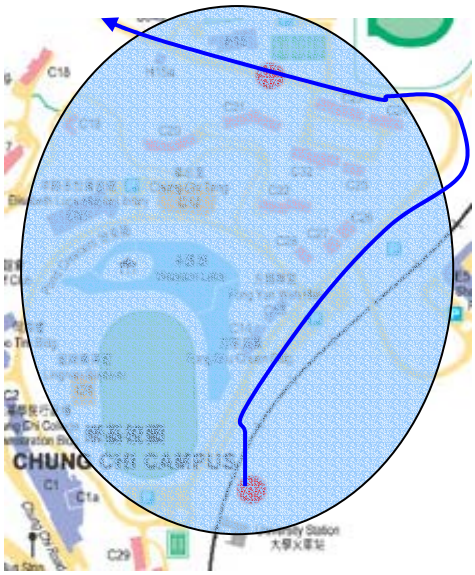


Fig 14.6: Estimated Cell Coverage of Sunday in CC College

As one can observe, KCR Station bus stop (bottom dot) and CC Hostels bus stop does not have considerable cell changes in between. Therefore, from cell data, there is no way to distinguish these two reference points. Therefore, here, again, illustrates the point that GSM cell ID location estimation relies on the configuration on base stations by network operators. However, this is the only case that exists for a particular telco while, in general, the application works fine to detecting locations and reference points.

14.3 Trade-off of Using Middleware-Assisted Application

This section concludes the trade-off of using middleware-assisted application and self-built application. Although middleware-assisted application can build an application in a quite development time such that developers can concentrate on further manual cell data processing and content enrichment, toolkits, especially AppGen, would introduce limitations – LBS application is confined into content providing. For example, when a new LBS game has to be built, there is no way to use AppGen for such purpose. LBS API and Cell Analyzer may somehow restrict developers to have new algorithm for cell data handling and processing. Therefore, there is a trade-off between convenience and flexibility as shown below. The higher the layer is, the more the convenience in LBS application development.

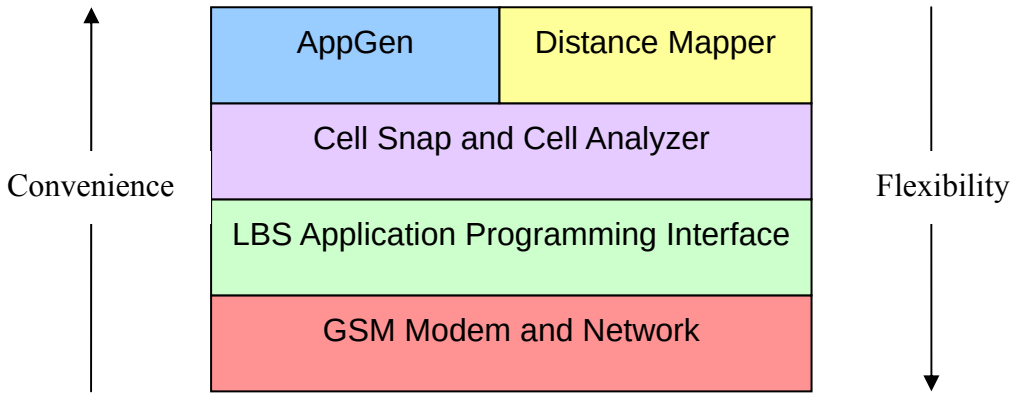


Fig 14.7: Trade-off between convenience and flexibility

14.4 Conclusion

From these two applications, MTR Traveller Remake and CU Campus Bus Route Guide, it is found that the middleware assists a lot in automated processing, manual data and content editing and software extensibility – these significantly reduce LBS application development time.

Chapter 15: Project Progress

The below table lists the project progress in this semester:

June 2003	- Plan the aim and approach of our final year project, which would be concentrated on mobile communication - Learn the basic of Symbian architecture and its features on mobile phones
July 2003	- Get familiar with C++ and Symbian programming
August 2003	- Understand wireless communication capability of Symbian, such as Bluetooth, socket, etc. - Develop Symbian programs, Nokia Square and Robot War, for showing the functionalities that can be achieved by Symbian OS
September 2003	- Finalize the topic – GSM cell ID positioning - Figure out the possibility for Symbian phone to collect cell information - Develop GSM Status for cell information display - Collect cell information in the CU campus to see whether it is possible to perform positioning in campus and reverse-engineering on actual locations of base station - Generate the idea of using cell change event for location estimation
October 2003	- Figure out the inadequacy of GSM cell ID positioning in two-dimensional area - Collect cell information of MTR stations and KCR stations - Decide to try out the proposed method on railway which has a fixed path
November 2003	- Design and implement MTR Traveller, including data storage, user interface, internal logic, etc. - Collect cell information of stations for another telco, Sunday, also - 1st Semester Project Report Writing
December 2003	- Cell Snap Implementation
January 2004	- LBS Middleware Design
February 2004	- Distance Mapper Implementation - Cell Analyzer Implementation

March 2004	- AppGen Implementation - MTR Traveller Remake by AppGen - CUHK Campus Bus Route Guide
April 2004	- Modifications on Cell Snap, Cell Analyzer and AppGen - Project Report Writing

Chapter 16: Conclusion

Symbian phone enhances the programming capability so that general developers can build applications on the mobile phone. The availability of interacting with phone's hardware generates a number of mobile opportunities.

Concerning location-base service, current technologies on positioning and LBS are quite mature and give high accuracy, such as GPS and GSM E-OTD, but most of them are at high cost, involve complicated design, and require the support from telcom companies. It seems that these are unaffordable to mainstream users who want to enjoy location-based services without much hardware upgrade.

Therefore, with a Symbian phone which accepts self-written applications and is GSM information accessible, it may be possible to introduce simple and telco-free location-based services to ordinary handset users with basic GSM cell information detection. However, the accuracy would be the major weakness, as figured out in the experiment targeted to the CU campus.

It has been, then, proposed the use of GSM cell change event, which is more accurate for positioning at the cell boundary. The turnout is that the hybrid method cannot perform well in presence of different drawbacks in two-dimensional region, like small cell size and high degree of overlapping.

On the other hand, applying the same method on path gives encouraging result from the initial observation of single cell change in closed MTR station. MTR Traveller, which is based on the proposed positioning method, has been built.

It is also observed that the desire of middleware to fasten LBS development period such that developers with different goals can enjoy the API and software tools provided. LBS API encapsulates all methods to manipulate cell data, perform callback upon cell changes and search for the nearest cell; Cell Snap facilitates cell data collectors with automated recording and reference point photo capturing; Cell Analyzer assists in cell data processing and convenient manual editing; Distance Mapper helps to map cells to physical location for the purpose of closest cell searching; AppGen offers content builders as well as developers to build a LBS application which stresses on content building or to further modify the source code for new services. Experiment on CU campus bus route has also been done to show the use of middleware.

Chapter 17: Acknowledgement

We would like to thank Prof. Michael Lyu, our project supervisor. He gave us valuable advices and necessary equipment for the project. Moreover, he offered us a number of opportunities, such as invitation to exhibition, company visit and public demostration, so that we can further understand the current situation of wireless communication as well as GSM location estimation.

Moreover, we would like to thank Edward Yau, Technical Manager of VIEW project. He gave us various innovative ideas on our project.

References

[1] K. Dixon, "Symbian OS Version 7.0s - Functional Description", Symbian Ltd., June 2003
http://www.symbian.com/technology/7.0s_functional_description2.1.pdf

[2] M. Tasker, J. Allin, J. Dixon, J. Forrest, M. Heath, T. Richardson, and M. Shackman, “Professional Symbian Programming – Mobile Solutions on the EPOC Platform”, Wrox Press Ltd., 2000

[3] Digia, "Programming for the Series 60 Platform and Symbian OS", Symbian Press, chapter 4, November 2002

[4] M. J. Jipping, “Symbian OS Communications Programming”, Symbian Press, chapter 14 and 15, June 2002

[5] M. Ali, “The System Applications of Novel Methods of Location and Tracking of Cellular Mobile Phones”, *IEE Colloquium on Novel Methods of Location and Tracking of Cellular Mobiles and Their System Applications (Ref. No. 1999/046)*, 17th May 1999

[6] H. Dieterich and S. Deshmukh, “High Level Requirements”, *Open Mobile Alliance Public Document*, 2nd October 2003

[7] “Mobile Location Protocol Specification Version 3.0”, *Location Inter-Operability Forum of Open Mobile Alliance - Public Document*, 6th June 2002

[8] I. Jami, M. Ali, and R. F. Ormondroyd, “Comparison of Methods of Locating and Tracking Cellular Mobiles”, *IEE Colloquium on Novel Methods of Location and Tracking of Cellular Mobiles and Their System Applications (Ref. No. 1999/046)*, 17th May 1999

[9] “Location Based Services”, *GSM Permanent Reference Document: SE.23*, January 2003
http://www.gsmworld.com/documents/lbs/se23310.pdf

[10] S. Y. Willassen, “A Method for Implementing Mobile Station Location in GSM”, section 6, March 1998
http://www.willassen.no/msl/

[11] “Welcome to Location Based Services”, *TOP Business AG Presentation*, 16th April 2002

[12] L. Lopes, E. Villier, and B.Ludden, “GSM Standards Activity on Location”, *IEE Colloquium on Novel Methods of Location and Tracking of Cellular Mobiles and Their System Applications (Ref. No. 1999/046)*, 17th May 1999

[13] J. Ashjaee and N. Ashjaee, “Basics of High-Precision Global Positioning Systems”, chapter 3, 24th October 1998

[14] “What is GPS?”, Garmin Ltd.
http://www.garmin.com/aboutGPS/

[15] “AT Command Set for GSM Mobile Equipment (ME) (GSM 07.07)”, *GSM Technical Specification from GSM Association*, July 1996

[16] K. Chadha, “The Global Positioning System: Challenges in Bringing GPS to Mainstream Consumers”, *Solid-State Circuits Conference*, 1998

[17] P. Stuckmann, “The GSM Evolution: Mobile Packet Data Services”, Wiley Europe, chapter 2, September 2002

[18] J. Eberspacher, H. Vogel and C. Bettstetter, “GSM Switching, Services and Protocols”, Wiley Europe, chapter 2, 1998

[19] M. Raento, "Symbian Programming - Getting the current Cell ID"
http://www.cs.helsinki.fi/u/mraento/symbian/cellid.html

[20] T. Farley, “Cellular Telephone Basics: AMPS and Beyond”, *Document in TelcomWriting.com*, section 2
http://www.privateline.com/Cellbasics/Cellbasics.html

[21] M. Smith, “Using the EPOC Database Manager”, Symbian Developer Library
http://www.symbian.com/developer/techlib/papers/v6/gt/sys/dbms/index.html

[22] “Developer Discussion Boards”, Nokia Ltd.
http://discussion.forum.nokia.com/forum

[23] C. K. Szeto, W. L. Law, C. C. Lui, B. Huguet, D. L. Pissart, S. Robertson, A. Alexander, and D. Stefan, “API Design in Software Architecture”, HKNet5
http://hknet.tm.tue.nl/section33/tech_design.html

[24] R. Lechich, “Orbis 2 Discussion of API’s”, 12th April 2000
http://www.library.yale.edu/orbis2/public/activity/API.html

Appendix 1 – GSM Positioning Method Accuracy [9]

Method	Rural Area	Suburban Area	Urban Area	Indoor
Cell ID	1km - 35km Typically 15km Extreme Case: ~100km	1km - 10km Typically 5km	Macro-cells: 500m – 5km Typically 2km Micro-cells: 50m – 500m Typically 200m	Pico-cell: 10m – 50m
E-OTD	50m – 150m	50m – 150m	50m – 150m	Good
A-GPS	10m	10 - 20m	10 – 100m	Variable (Still not proven)

Appendix 2 – Signal Attenuation of Different Construction Materials [10]

Construction element	Attenuation (dB)	Standard Deviation (dB)
Concrete block wall	7	1
Wood and brick siding	3	0,5
Aluminium siding	2	0,5
Metal walls	12	4
Attenuation past office furnishings (dB/m)	1	0,3

Appendix 3: Cell Information in MTR and KCR Stations for Sunday and Peoples

Fig A3.1: Cell IDs of selected MTR stations for Sunday

Fig A3.2: Cell IDs of selected MTR stations for Peoples

Fig A3.3: Cell IDs of KCR stations for SmarTone between Kowloon Tong station and University station

Appendix 4 – Sample Map File, Station File and Transition File in MTR Traveller

A4.1 Map File

The map file allows developers to construct the map of MTR or KCR railways for display. The format of map file is:

1st Line: Comment

2nd Line: Total number of station bitmaps

3rd Line: Width and Column of the Map

4th Line Up: Array of bitmap IDs defined in the program

```
#
40
3x23
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
029 113 030 113 031 113 032 113 033 113 034 113 035 113 036 113 037 113 038 113 039 113 040
000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000
```

A4.2 Station File

Station file stores the station name, station ID, location ID and cell ID (notice that cell information includes both location ID and cell ID). The below is a simplified example.

```
#Station Name to ID for SmarTone
Po Lam:1:50:13051
Hang Hau:2:50:13041
Tseung Kwan O:3:50:13031
Tiu Keng Leng:4:50:13021
Yau Tong:5:50:13012
Quarry Bay:6:50:4921
North Point:7:50:4938
Lam Tin:8:50:12812
Kwun Tong:9:50:12813
Ngau Tou Kok:10:50:4071
Ngau Tou Kok:10:50:43352
Kowloon Bay:11:50:41256
Choi Hung:12:50:4911
……..
```

A4.3 Transition File

The transition file stores all cell changes in between two stations. The sequence of entries is important because it may indicate the direction as stated in the report before (transition pairs in Section 8.4.2). The format is station name 1, station name 2, location ID, cell ID, bitmap x,y-coordinates on the map.

```
#Transition table for SmarTone
Po Lam:Hang Hau:50:13051:26:1
Po Lam:Hang Hau:50:13041:26:1
Hang Hau:Tseung Kwan O:50:13041:24:1
Hang Hau:Tseung Kwan O:50:13031:24:1
Tseung Kwan O:Tiu Keng Leng:50:13031:22:1
Tseung Kwan O:Tiu Keng Leng:50:13021:22:1
Tiu Keng Leng:Yau Tong:50:13021:20:1
Tiu Keng Leng:Yau Tong:50:13012:20:1
Yau Tong:Quarry Bay:50:13012:18:1
Yau Tong:Quarry Bay:50:4921:18:1
Yau Tong:Lam Tin:50:13012:18:1
Yau Tong:Lam Tin:50:4926:18:1
Yau Tong:Lam Tin:50:12812:18:1
Yau Tong:Lam Tin:50:42502:18:1
Lam Tin:Kwun Tong:50:42502:16:0
Lam Tin:Kwun Tong:50:12812:16:0
Lam Tin:Kwun Tong:50:42503:16:0
Lam Tin:Kwun Tong:50:12813:16:0
Kwun Tong:Ngau Tou Kok:50:12813:14:0
Kwun Tong:Ngau Tou Kok:50:40492:14:0
Kwun Tong:Ngau Tou Kok:50:5461:14:0
Kwun Tong:Ngau Tou Kok:50:5462:14:0
Kwun Tong:Ngau Tou Kok:50:43305:14:0
Kwun Tong:Ngau Tou Kok:50:20113:14:0
Kwun Tong:Ngau Tou Kok:50:43352:14:0
Kwun Tong:Ngau Tou Kok:50:4071:14:0
……
```

Appendix 5: Detailed Statistics for Time Recorded in Section 8.5

From	To	Total Time of Journey	11/17/2003	11/18/2003	11/20/2003	11/21/2003	11/22/2003	11/24/2003	11/25/2003	Error range in Time (%)	Error Range in Distance (m)
Po Lam	Hang Hau	83.4	32.0	34.5	35.3	39.1	38.1	36.8	36.4	8.51	65.06
Hang Hau	Tseung Kwan O	106.3	45.0	56.4	71.5*	49.0	61.0	49.6	57.3	15.05	146.65
Tseung Kwan O	Tiu Keng Leng	74.0	21.0	29.2	30.5	16.8	31.1	18.4	X	19.32	127.51
Tiu Keng Leng	Yau Tong	136.8	42.1	62.7*	43.0	46.1	47.9	41.5	39.0	6.50	81.51
Yau Tong	Lam Tin	103.4	1.0	X	4.5	X	X	1.6	1.8	3.38	32.04
Choi Hung	Diamond Hill	79.8	34.3	31.4	38.6	X	X	27.1	31.5	14.41	105.41
Diamond Hill	Wong Tai Sin	74.5	16.8	22.2	23.3	X	X	21.4	21.8	8.73	59.62
Lok Fu	Kowloon Tong	97.2	25.0	60.0*	X	X	X	16.6	47.4	31.69	282.36
Kowloon Tong	Lok Fu	74.5	18.5	13.2	14.9	7.1	X	18.9	14.7	15.84	108.17
Wong Tai Sin	Diamond Hill	60.0	28.7	28.9	30.0	32.1	36.2	15.6	X	24.00	132.00
Diamond Hill	Choi Hung	83.8	36.8	36.7	32.4	33.9	33.3	32.1	35.4	5.61	43.09
Lam Tin	Yau Tong	80.2	59.5	40.0	44.1	44.0	45.3	41.7	43.1	24.32	178.79
Yau Tong	Tiu Keng Leng	150.7	78.0	81.3	70.6	91.6	88.3	91.3	84.3	13.93	192.43
Tiu Keng Leng	Tseung Kwan O	77.4	27.0	23.2	28.3	29.9	28.3	29.9	31.1	10.21	72.44
Tseung Kwan O	Hang Hau	58.5	46.3	31.8	37.7	39.2	46.7	37.5	X	25.47	136.58
Hang Hau	Po Lam	87.8	X	22.7	X	38.1	35.5	30.9	25.4	17.54	141.17

*: Ignored due to abnormal speed (e.g. stop in between two stations)

X: Not Recorded

Appendix 6: API Documentation

The below is the documentation of our API functions provided to LBS developers:

Class CNetworkInfo

Description:
This class provides basic functions to retrieve the current location data from the mobile phone

void connectL()
This function connects the telephone server through which we can retrieve location data. It must be called before retrieving any location data.

void disconnect()
This disconnects the telephone server.

TBool isConnected()
This function return ETrue if the telephone server is already connected. Otherwise, it returns EFalse.

TInt getCurrentLocationID()
This function retrieves the current location ID and return in form of an integer.

Tint getCurrentCellID()
This function retrieves the current cell ID and return in form of an integer.

Class CLocationListener

Description:
This class provides function that allows users to develop location-sensitive function. By providing a list of desired location IDs, users can specify a function to be executed when the device detects the entrance of any of the cells specified in the list. To use this service, users have to extend this class and implement the function coreFunction().

void start()
Start monitoring cell change event, if an entrance of a specific location specified by the location list is matched, coreFunction() will be invoked.

void stop()
Stop monitoring cell change event.

void connectL()
Connect to telephone server via CNetworkInfo class

void disconnect()
Disconnect from telephone server

void setCheckingInterval(const TInt aInterval)
Set the rate of cell-change event checking. Parameter is time in microsecond. Default checking interval is 1 second

void setLocationList(RArray<TLocation> aLocationList)
Set the location list that is to be checked with the current location IDs. If the location list is empty, coreFunction() will be invoked every time a cell change event is detected

TInt getCurrentLocationID()
Retrieve the current location ID.

TInt getCurrentCellID()
Retrieve the current location ID

void coreFunction()
Users have to implement this function. This function will be invoked if the current location matches any of the location in the location list. This function should take no parameter and return nothing.

void readInL(const TDesC& aFileName)
Read in the file specifying a set of reference point. Upon reaching these reference point, coreFunction will be called.

void prepareSelectionList(const TDesC& aFileName)
Read in a file that contains a set of point of interest (POI) together with their relative reference point. The list of POI will be displayed in a list for users to choose.

void setPOI(Tint aIndex) (internal usage)
Set the POI so that when reaching that POI, an action will be taken(prompt message)

Class CProximity

Description:

This class provides functions that allow users to find out the object(s) nearest to the current location.

TBool readIn(TDesC& aFileName)

Read in the distance table specifying the distance of different objects with the current location.

void findNearest(TLocation aCurrentLocation&, RArray<TLocation> aResult)

Given a current location, the nearest object(s) will be inserted into the aResult.

void findSameRange(TLocation aCurrentLocation, TInt aDistance, RArray<TLocation>& aResult) ***(Deprecated)***

Given a current location and a specific distance, it will find out all the object(s) with distance equals to the specified distance. All results will be inserted into aResult.

Appendix 7: Cell Snap Data Format

The following example is the actual cell data of SmarTone, one of the telco in Hong Kong, from University KCR station to New Asia College in CUHK.

```
03/03/2004 9:59:16 AM: Starting cell collection by CellSnap...
   0 s: Starting at Location: [ 130 ], Cell ID: [ 30852 ]
   4 s: < CellSnap0001.jpg > Location: [ 130 ], Cell ID: [ 30852 ]
  12 s: Location: [ 130 -> 130 ], Cell ID: [ 30852 -> 26391 ]
  49 s: Location: [ 130 -> 130 ], Cell ID: [ 26391 -> 33731 ]
  97 s: Location: [ 130 -> 130 ], Cell ID: [ 33731 -> 26391 ]
 105 s: Location: [ 130 -> 130 ], Cell ID: [ 26391 -> 26372 ]
 122 s: Location: [ 130 -> 130 ], Cell ID: [ 26372 -> 33731 ]
 129 s: < CellSnap0002.jpg > Location: [ 130 ], Cell ID: [ 33731 ]
 143 s: Location: [ 130 -> 130 ], Cell ID: [ 33731 -> 33733 ]
 165 s: Location: [ 130 -> 130 ], Cell ID: [ 33733 -> 33731 ]
 217 s: Location: [ 130 -> 130 ], Cell ID: [ 33731 -> 26393 ]
 232 s: Location: [ 130 -> 130 ], Cell ID: [ 26393 -> 33733 ]
 240 s: Location: [ 130 -> 130 ], Cell ID: [ 33733 -> 32122 ]
 258 s: Location: [ 130 -> 130 ], Cell ID: [ 32122 -> 33733 ]
 274 s: Location: [ 130 -> 130 ], Cell ID: [ 33733 -> 26393 ]
 287 s: Location: [ 130 -> 130 ], Cell ID: [ 26393 -> 32122 ]
 321 s: Location: [ 130 -> 130 ], Cell ID: [ 32122 -> 5812 ]
 326 s: < CellSnap0003.jpg > Location: [ 130 ], Cell ID: [ 5812 ]
 341 s: Location: [ 130 -> 130 ], Cell ID: [ 5812 -> 32122 ]
 347 s: Location: [ 130 -> 130 ], Cell ID: [ 32122 -> 30852 ]
 354 s: Location: [ 130 -> 130 ], Cell ID: [ 30852 -> 5812 ]
 394 s: Location: [ 130 -> 130 ], Cell ID: [ 5812 -> 32122 ]
 401 s: Location: [ 130 -> 130 ], Cell ID: [ 32122 -> 30852 ]
 409 s: Location: [ 130 -> 130 ], Cell ID: [ 30852 -> 30851 ]
 424 s: < CellSnap0004.jpg > Location: [ 130 ], Cell ID: [ 30851 ]
 494 s: < CellSnap0005.jpg > Location: [ 130 ], Cell ID: [ 30851 ]
 520 s: Location: [ 130 -> 130 ], Cell ID: [ 30851 -> 32123 ]
 546 s: Location: [ 130 -> 130 ], Cell ID: [ 32123 -> 5812 ]
 574 s: < CellSnap0006.jpg > Location: [ 130 ], Cell ID: [ 5812 ]
03/03/2004 10:08:54 AM: Stopping cell collection by CellSnap...
```

Appendix 8: Cell Analyzer Output Data for CUHK Campus Bus Route

The below is the output data from Cell Analyzer after processing.

Prefix would determine the meaning of a line:

```
# = Comment
; = Name of a reference point
Others = [Location ID]#[Cell ID]#[Time Elapsed from Starting the Program]
```

The following data, in one single data file, consist of 4 telcos, namely SmarTone (Location ID: 130), Orange (Location ID: 850), Peoples (Location ID: 140) and Sunday (Location ID: 14000).

```
# Cell ID Data Generated by Cell Analyzer
; KCR Station
130#30852#4
130#26391#12
130#33731#49
130#26372#105
140#10873#0
14000#54580#2
14000#54063#47
14000#54663#59
850#7782#4
850#37782#50
850#16011#135
; CC Hostels
130#33731#129
130#33733#143
130#26393#217
130#32122#240
140#2443#76
140#10871#155
140#3261#208
14000#54662#80
14000#54063#84
14000#54580#139
```

```
850#11742#191
850#16011#244
; Central Campus
130#5812#321
130#32122#341
130#30852#347
130#30851#409
140#10871#238
140#10872#335
14000#54361#210
14000#54362#285
850#7781#299
850#37781#319
; K.K. Leung Building
130#30851#424
140#10872#359
14000#54362#293
850#7781#390
; UC College
130#30851#494
130#32123#520
140#10871#449
140#3261#498
850#5981#439
850#7781#450
850#37781#473
; NA College
130#5812#574
140#3261#531
140#3263#534
14000#54361#419
14000#54652#421
850#7782#485
```

Appendix 9: AbbreviationsUsed in the Report

3G/3GSM	3rd Generation GSM
API	Application Programming Interface
CC	Chung Chi College, CUHK
CDMA	Code Division Multiple Access
DBMS	Database Management System
EDGE	Enhanced Data Rates for GSM Evolution
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
J2ME	Java 2 Platform, Micro Edition
KCR	Kowloon-Canton Railway
LAN	Local Area Network
LBS	Location-Based Service
MLP	Mobile Location Protocol
MMS	Multimedia Messaging Service
MTR	Mass Transit Railway
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
SDK	Software Development Kit
SMS	Short Messaging System
WLAN	Wirelss Local Area Network
XML	eXtensible Makeup Language

~ End of the Report ~