Aerial Localization with Smartphone

Zhongli Liu, Yinjie Chen, Benyuan Liu, Jie Wang, and Xinwen Fu

Computer Science Department, University of Massachusetts Lowell
Email: {zliu, ychen1, bliu, xinwenfu}@cs.uml.edu

Abstract. This paper presents how we applied a smartphone for aerial localization. We have developed a fully functional aerial localization system *HAWK* and reported preliminary results in a related paper. In this paper, we focus on the technical details of using a smartphone Nokia N900 as a wireless sniffer on a mini helicopter and comparing the performance of three localization approaches for wireless device localization. The flight is controlled by a software controller on a laptop. The flight route can be specified in two ways: manually setting waypoints on Google map and automatically generating waypoints based on Moore space filling curve. The smartphone based sniffer captures the wireless traffic during flight and transmits the traffic dump files through a 3G network to a locator once the surveillance flight is finished. We applied three different approaches, maximum signal strength approach, centroid approach and Quasi-Newton method, for the locator on the laptop to calculate the position of the target device and compared the localization accuracy of these three localization approaches. Surprisingly, the simplest approach, maximum signal strength approach (which uses the location where the maximum signal strength is sensed as the target's location) has similar localization accuracy compared with the other two.

1 Introduction

Wireless localization techniques have enjoyed great success and pervasive deployment. In a wireless localization scene, there are three participants: target, positioning infrastructure and third party. Any of these three participants can calculate the location of the target. Based on who calculates the location of the target, we can classify wireless localization technologies into three categories: self positioning, infrastructure positioning and third party localization. In self positioning, the target interacts with the positioning infrastructure such as the GPS constellation and calculates its own location. In infrastructure positioning, the infrastructure such as the cellular towers can sense the signal of an active phone and use trilateration to locate the target phone. This paper is interested in the third type of localization technique, third party localization.

In a third party localization, a third party, not the target or infrastructure, can sense the signal of the target and locate the target without the help of the positioning infrastructure and target. Third party localization has broad applications including public safety, cyber forensics, and network management. For example, if travelers equipped with smartphones are lost in a forest and we want

to locate them, cellular towers may not exist over there for the localization. In this scenario, we may send a mini helicopter, which is a third party, to locate the travelers via locating their smartphone through its wireless signal. For example, we can either turn on the smartphone's WiFi access point mode or develop an app for the localization purpose.

We have developed a fully functional aerial localization system *HAWK*, a mini-helicopter based aerial wireless kit, and reported preliminary results on localization in a related paper [13]. In this paper, we focus on the technical details of using a smartphone Nokia N900 as a wireless sniffer on a mini helicopter and comparing the performance of three localization approaches for wireless device localization. The contribution of this paper can be summarized as follows:

- After reporting preliminary results of HAWK in [13], we conducted intense development and analysis. The flight is controlled by a software controller on a laptop. The flight route can be specified in two ways: manually setting waypoints on Google map and automatically generating waypoints based on Moore space filling curve (Moore curve). The smartphone based sniffer captures the wireless traffic during flight and transmits the traffic dump files through a 3G network to a locator once the surveillance flight is finished.
- We applied three different approaches, maximum signal strength approach, centroid approach and mean square error approach, for the locator on the laptop to calculate the position of the target device and compared the localization accuracy of these three localization approaches. Surprisingly, the simplest approach, maximum signal strength approach (which uses the location where the maximum signal strength is sensed as the target's location) has similar localization accuracy compared with the other two.

The rest of this paper is organized as follows: In Section 2, we introduce the system structure of aerial localization with smartphone. Section 3 shows three different approaches that are applied to analyze the experiments results. We present experimental evaluation of this system in Section 4. Section 5 discusses related work. The conclusion of this paper is in Section 6.

2 System

In this section, we first introduce the structure and basic idea of aerial localization system, then investigate a few challenge issues of this localization system. At last, we present our solutions to these issues.

2.1 Overview of HAWK

Figure 1 exhibits the system architecture of the aerial localization with a smartphone. There are four main components in this system: helicopter, wireless sniffer, controller and locator.

(i) *Helicopter*. We use a mini helicopter Draganflyer X6 as a flight tool [10], which is used to carry a wireless sniffer to do the aerial surveillance and localization. X6 can log its GPS coordinates during flight and transfer these GPS data

Fig. 1. Architecture of System

to the locator. X6 can be controlled by both handheld controller and software controller.

(ii) *Wireless sniffer*. We convert a smartphone Nokia N900 [3] to a wireless sniffer. This sniffer is attached to the mini helicopter and captures wireless traffic during helicopter's flight. The information collected by the sniffer will be transmitted to a locator through the 3G network.

(iii) *software controller*. The software controller runs on a Lenovo W500 laptop to autonomously maneuver the helicopter movement while flying. The software controller is able to draw helicopter's flight route on real time, and show all the wireless devices sniffed by N900 after the flight.

(iv) *locator*. After receiving sniffer's dump files, a software locator on the laptop will analyze the data and determine the location of the target.

2.2 Basic Idea

The basic idea of this aerial localization system is using a mini helicopter attached with a smartphone Nokia N900, which works as a wireless sniffer, to identify the location of a target wireless device. The helicopter, with the sniffer, performs the aerial surveillance over a given area while the sniffer is collecting wireless traffic such as RSSI (received signal strength indication) time series and coordinate information. When the surveillance process is finished, all the collected information is transmitted to the locator. Then, these data are analyzed to calculate the target device's position.

The surveillance flight route is derived through two ways: waypoints that are generated from Moore curve and waypoints that are set from Google map as shown in figure 2. The first Moore curve approach has been discussed in [13]. The locator also can draw the flight route on real time, given that helicopter can log its GPS coordinates which will be transmitted to the locator simultaneously. Figure 3 shows the flight route when the helicopter is doing the surveillance according to a predefined level 2 Moore curve flight route.

2.3 Issues and Solutions

The aerial localization system has several issues that need to be addressed in this paper:

(i) What are the primary functions of the software controller? How to monitor the aerial surveillance and localization, and show the localization result?

(ii) Which kind of sniffer should be selected for our system to capture the wireless traffic? Since the helicopter has a payload limitation, the sniffer's weight should be considered.

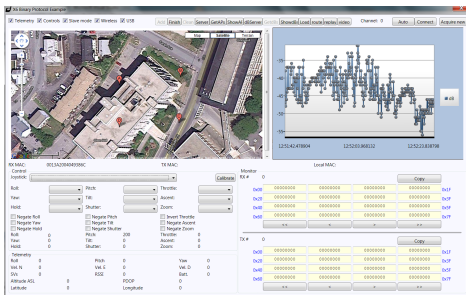

Fig. 2. Set Points on Google Map

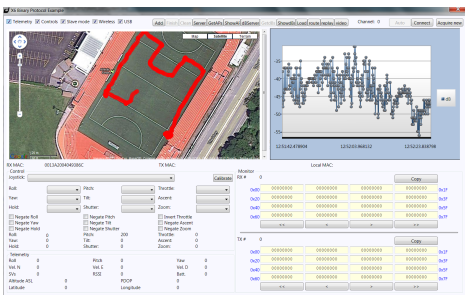

Fig. 3. Flight Route in Real Time

(iii) How does the locator obtain the logfiles from sniffer on real time? How to find the location of target device based on these logfiles?

2.4 Functions of Software Controller

The software controller is developed based on CSharp language, and uses a USB telemetry transceiver to communicate with helicopter. It first sets the surveillance route for helicopter by directly setting waypoints on Google map, or calculating a list of waypoints based on Moore Curve. Then, helicopter with sniffer is maneuvered to flight sequently according to these waypoints to do the surveillance. [13] has discussed in details with regarding to how to control the helicopter to achieve the waypoints function. Since the helicopter can log its GPS coordinates and transmit them to software controller, we can draw the flight route on Google map on real time. After the surveillance and the target's position is calculated, location of the target is also able to be displayed on Google map.

2.5 Using smartphone Nokia N900 as Wireless Sniffer

In this paper, we use smartphone to work as a wireless sniffer. We choose N900 simply because its weight is below the payload limit of the helicopter, even though there are many smartphones that qualify the system.

To convert N900 to a wireless sniffer, we need to update its kernel to the latest version and install several softwares: rootsh, Enhanced Linux kernel, networktools, and Kismet. Prior to the installations, we need to add the maemo devel repository into the source list [1] as listed below. Menu → More → App. Manager → Application Manager → Application catalogues → New. Then, a window showing as figure 4 pops up as below, and we fill the blanks and click Save.

Followings are the steps to install these softwares:

1. In the Application Manager, click Download, then type "rootsh" and "install". This helps us to gain a root shell.

2. The Enhanced Linux kernel refers to Enhanced Linux kernel for power users, which is needed by Kismet [2]. We install it through the Application Manager, too.

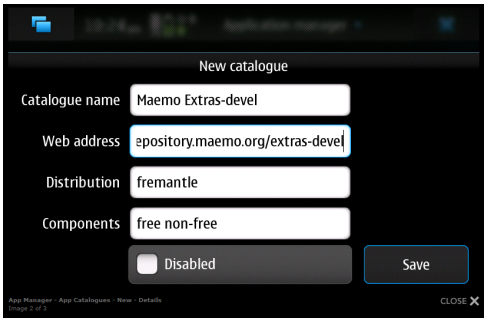

Fig. 4. Add Maemo Devel Repository

3. Install wireless-tools and Kismet: First Open X Terminal, and then input commands "sudo gainroot" and "apt-get install wireless-tools kismet".

Now N900 can run Kismet and work as a sniffer to collect the wireless traffic. In order to let Kismet capture the wireless packets, the wireless card should work under monitor mode. Thus, we use the following commands to set the wireless card before running kismet:

- ifconfig wlan0 down.
- iwconfig wlan0 mode monitor.
- ifconfig wlan0 up.

2.6 Downloading Logfiles and Locating the Target Device

To use the locator to derive the target device's location, we need to transmit the logfiles from sniffer to locator right after the aerial surveillance. Because the wireless card is under monitor mode when Kismet is working, we could not get these logfiles. To resolve the issue, we use 3G network to download the logfiles from sniffer to locator. However, the 3G network does not provide a public IP address to N900 and the locator can not connect to N900 directly. Therefore, we use $reverse\_SSH$, which relies on a common server that both the N900 and locator can reach. The N900 connects to the common server, waiting for the locator to connect, then, the locator connects to the common server, which in turn forwards that connection to N900.

[6] introduces how to set the $reverse\_SSH$ in detail. There are two basic parts, on N900 and on locator.

On N900: To connect to the common server, we do the following:

1. Creating a script named as: $reverse\_ssh.sh$, under direction $/usr/share/$.

2. Setting the content of $reverse\_ssh.sh$ as $ssh\ -Nf\ -R\ 2210 : localhost : 22\ root@server\_ip > /var/log/reverse\_ssh.log$;.

3. Modifying the privilege of $reverse\_ssh.sh$ with the command: $chmod\ 777\ reverse\_ssh.sh$.

4. Running this script as $./reverse\_ssh.sh$ to connect to server.

There is one more issue worth brief mention here. That is, password is required each time to connect to the server through N900. This can be resolved

by SSH keys and ssh-agent which allow us to type in a passphrase only once on your workstation [7].

On Locator: After the connection between N900 and server is established, we can use locator to connect to server then to N900. To establish this connection, using ssh and run the command:

$$ssh\ \text{-}p\ 2210\ root@localhos$$

Now locator can download logfiles directly from N900.

After the locator obtained the logfiles, it searches these files and calculates the location of the target device. Details of these approaches used for calculation will be discussed in section 3. Figure 5 shows the flow chart of transferring data and locating process.

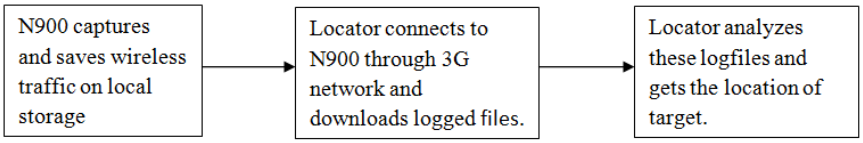

Fig. 5. Flow Chart of Transferring Data and Locating Process

3 Analysis

In this section, we will introduce three different approaches that are used to calculate the location of target device based on the logfiles collected by wireless sniffer. We also discuss relative merits among the three approaches.

3.1 Maximum Signal Strength Approach

Kismet has the capability to log the GPS coordinate where the wireless sniffer sensed the maximal RSSI which is saved as *.nettxt* file. Our first approach is to simply find the position where the sniffer receives the strongest RSSI of the target device from *.nettxt* file, and uses this position as the target device's location. Although this approach is simple and easy to be implemented, the localization is less accurate as this position with the strongest signal strength is where Kismet gets its first strongest RSSI.

3.2 Centroid Approach

To improve the accuracy, we use the centroid localization approach. That is, we search and select all GPS positions where all strongest RSSIs are sensed. Then, we use the average value of these GPS positions as the target location. Among the logfiles from kismet, there is a *.gpsxml* file which logs all relationship between the GPS coordinates and sensed RSSIs. Thus, we can derive the useful data from this logfile. Algorithm 1 shows the basic steps of how to calculate

Algorithm 1 Target Location Calculation

Require: Logfiles .gpsxml and .nettxt from sniffer
1: Set $P = 0$
2: Set list(x) and list(y)
3: Set *t-mac* as target's MAC address
4: Find the strongest RSSI from .nettxt file where the $MACaddress = t\text{-}mac$
5: Set P = strongest RSSI
6: Select all the GPS coordinates, including latitudes and longitudes, from .gpsxml file where $MACaddress = t\text{-}mac$ and $RSSI = P$
7: Add the latitudes to x
8: Add the longitudes to y
9: Set the target's coordinate as the average(x) and average(y)

the location of target device based on average localization approach. Even this approach needs a little bit more calculation than the first approach, it is more accurate.

3.3 Quasi-Newton Method

In practice, the distance between wireless devices and APs can only be derived from RSSI values as showed in formulas (1) and (2). Thus, our third approach is to calculate the target device directly from the distances and RSSIs.

$$p_i = P - 10 \times r \times \log d_i + R \tag{1}$$

$$d_i = \sqrt{(D_g)^2 + (alt_i - alt)^2} \tag{2}$$

Where p_i is the RSSI that sniffer gets at GPS location (lat_i, lon_i, alt_i). We can get these four parameters and their relationship from the logfiles. P is the RSSI one meter away from the target device. r is the path loss exponent that captures the rate of fall of RSSI in the vicinity of target device. (lat, lon, alt) is the location of target device, and we assume the height for target device is 0, $alt = 0$. R is a random variable that hopes to capture the variations in the RSSI due to multi-path effects, asymmetries in the physical environment and other imperfections in the model itself. D_g is the great circle distance between the target device's location and the GPS position where sniffer senses the RSSI. The formula used to calculate the great circle distance is in [9].

In formulas (1) and (2), there are four unknown parameters: P, r, lat and lon. Thus, we can select four different pairs of p_i and (lat_i, lon_i, alt_i) to form four nonlinear algebraic equations.

To solve these equations, we can use a matlab function "fsolve", based on quasi-Newton method [5], to solve sets of nonlinear algebraic equations. However, to use $fsolve$ to solve the nonlinear algebraic equations, we need to supply a routine to evaluate the function vector. Different routines will result in these functions with different solutions.

4 Evaluation

In this section, we first introduce how to setup the experiments, followed by discussion of the experiments results based on three different locating approaches mentioned in section 3.

4.1 Experiment Setup

We conducted real-world field experiments to evaluate the localization capability of our aerial localization system with smartphone. Before these experiments, we generated 3 sets of Moore curve over the campus track field as helicopters surveillance route: level 1, level 2 and level 3 Moore curves, and the warwalking route around the track field for surveillance by warwalking. The flight route is same as [13].

In the first experiment, we configured 12 smartphones as APs and uniformly distributed them on the track field. The Nokia N900 running Kismet was attached on the helicopter to log the locations of these 12 smartphones. Kismet was configured to hop among all the channels (default mode). The software controller on a Lenovo w500 laptop steered the flight along the three routes to locate these APs. The warwalking experiment emulated the scene where people cannot access dead ends, such as building roofs (the field in the experiment).

In the second experiment, all 12 APs were set to a same channel, and Kismet was configured to sniff on this single channel.

4.2 Comparison localization results among three approaches

After we got the logfiles from sniffer, we used the approach mentioned in section 3.1 to analyze both experiments. That is, we selected the GPS location where wireless sniffer received the strongest signal as the target device's position.

We applied the second approach discussed in section 3.2 to the second experiment. We used the Algorithm 1 in section 3.2 to find the location of target device.

To evaluate the approach in section 3.3, we selected the four pairs data, including signal strength and GPS location where sniffer got the signal strength. Then we constructed four different equations and used matlab function "fsolve" to locate the target device. The initial value for 1 meter signal strength was -5, the path loss exponent was 4, and the target device's position was the location where we got from the second approach in section 3.2. Figure 6 shows the localization results of these three approaches. This figure shows that the outcomes of both experiments are similar to the first approach. The performance of the second approach appears improved but not very obviously. The third approach is the least desirable as showed in the figure 6. When solving the equations with "fsolve" function, we notice that the routine value of this function can significantly affect the localization result.

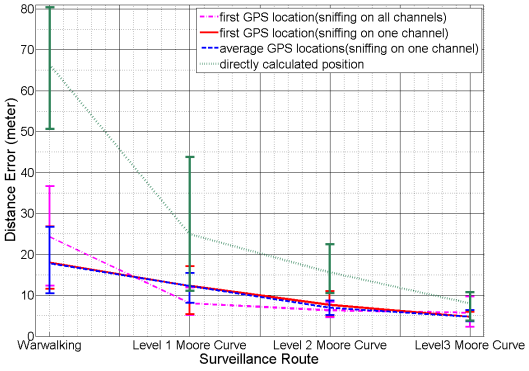

Fig. 6. Localization Error via Kismet

5 RelatedWork

There has been considerable amount of work on device positioning in WiFi and sensor networks. Due to space limitation, we only review most related and recent work. The most related work to this paper is W.A.S.P [8] and SkyNET [16]. W.A.S.P uses a mini airplane for wireless surveillance and attacks. However, the mini airplane has to fly at a relatively high speed in order to float in the air. We have proved in [13] that W.A.S.P is not appropriate for accurate wireless localization. SketNET is designed as an aerial botmaster to exploit weak wireless devices and form a botnet. It uses Ar Drone [4] to carry a single board computer (SBC) equipped with wireless adapters for this purpose. The SBC is attached to the top of the Ar Drone since Ar Drone has its sonar ranger finder at the bottom. This design is not appropriate for wireless localization since the body of Ar Drone will disturb the received signal.

In [14], [15], the authors proposed SensorFly, an aerial sensor network, where very small helicopter can self-locate itself using anchor nodes. The authors in [12] utilized biologically inspired rules of group behavior (flocking) to enable a group of UAVs to control its own motion. This project aimed at building an indoor flocking system using small co-axial rotor helicopters. Each of the swarm members is fitted with an onboard computer and a miniature wireless video camera, so that they can gather multiple views in a single pass and analyze them. Another project SensorFlock [11] utilizes a group of micro aerial vehicles (MAVs) for atmospheric sensing. This system requires human interaction in flight control and path planning, and it supports wireless communication networking between MAVs. In [17] the authors present 3DLoc which is a ground based system for locating an 802.11-compliant mobile device in a three dimensional space. However, the portability and flexibility of the system is very limited and it cannot search targets in high buildings.

6 Conclusion

This paper presented an infrastructure free and highly portable system for aerial localization of wireless device that is attached with a smartphone (Nokia N900). We developed a software controller to control the warflying tool, a mini helicopter. We used two approaches to generate the flight route for aerial surveillance. We converted the smartphone as the wireless sniffer to capture the wireless traffic and applied three different approaches to valuate the performance of our system and compared their performance. Our future work includes using the smartphone to control the helicopter's movement so that we can automatically perform the aerial surveillance and localization.

References

1. Install chinese input on the n900. `http://kenshinjeff.jp/2010/05/17/installing-chinese-input-on-the-n900/`, 2010.
2. Kismet. `http://www.kismetwireless.net/`, 2011.
3. Nokia n900. `http://en.wikipedia.org/wiki/Nokia_N900`, 2011.
4. Ar. drone. `http://ardrone.parrot.com/parrot-ar-drone/usa/`, 2012.
5. Quasi-newton method. `http://en.wikipedia.org/wiki/Quasi-Newton_method`, 2012.
6. Reverse_ssh. `http://wiki.maemo.org/Reverse_ssh`, 2012.
7. Setup ssh keys between machines. `http://fedoranews.org/dowen/sshkeys/`, 2012.
8. Wireless aerial surveillance platform. `https://rabbit-hole.org/`, 2012.
9. C. Carter. Great circle distances. May 2002.
10. I. Draganfly Innovations. Innovative uav aircraft and aerial video systems. `http://www.draganfly.com`, 2010.
11. A. B. Hasan, B. Pisano, S. Panichsakul, P. Gray, J. Huang, R. Han, D. Lawrence, and K. Mohseni. Sensorflock: A mobile system of networked micro-air vehicles. Technical report, Department of Computer Science University of Colorado at Boulder, 2006.
12. O. Holland, J. Woods, R. D. Nardi, and A. Clark. Beyond swarm intelligence: The ultraswarm. In *Proceedings of the IEEE Swarm Intelligence Symposium*, 2005.
13. Z. Liu, Y. Chen, B. Liu, C. Chao, and X. Fu. Hawk: An unmanned mini helicopter-based aerial wireless kit for localization. Inforcom '12, 2012.
14. A. Purohit, Z. Sun, M. Salas, and P. Zhang. SensorFly: Controlled-mobile sensing platform for indoor emergency response applications. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2011.
15. A. Purohit and P. Zhang. Sensorfly: A controlled-mobile aerial sensor network. In *The Seventh ACM Conference on Embedded Networked Sensor Systems*, November 2009.
16. T. Reed, J. Geis, and S. Dietrich. SkyNET: A 3g-enabled mobile attack drone and stealth botmaster. In *Proceedings of the 5th Usenix Workshop on Offensive Technologies (WOOT)*, 2011.
17. J. Wang, Y. Chen, X. Fu, J. Wang, W. Yu, and N. Zhang. 3DLoc: Three dimensional wireless localization toolkit. In *Proceedings of IEEE ICDCS*, 2010.