Context Aware Lifestyle: Initial Report Documentation

Release 1.0

Michael Bartling

October 19, 2014

Contents:

CHAPTER

ONE

INTRODUCTION

1.1 Background

Mobile platforms contain a large variety of sensors which can provide contextual information on user's activities. Although inferring user activities from sensor aggregates is not a new concept, previous research on the topic has generally been limited to one to two sensor experiments. Several studies show how accelerometers can be used to infer coarse-grained information such as user motion (walking, standing, sitting, running, etc) and others show how it is possible to achieve even finer granularities such as infering key-strokes and other touch patterns. Changing perspective of what constitues a "sensor", modern *virtual sensors* combine multiple hardware sensors to improve sensor readings or create entirely new sensor capabilities. For example, the Android Linear Acceleration Sensor builds upon the acccelerometer and is *triggered* whenever a significant change in motion occurs.

GPS and other Wireless interfaces have, for several years, provided robust contextual inference channels. Many applications are senstive to *location context*. In one situation, users can search for nearby shops. In another situation, marketing firms can gather information on clients shopping habits, which often help in building predictive market models.

It should be further noted that adding context to applications opens up a wide variety of privacy concerns. Context is, in a way, a side channel by nature.

1.2 Cal, a Context Aware Lifestyle

Cal, or Context Aware Lifestyle, is a multimodal sensor aggregation platform that is aware of its users context. At the surface, Cal is a project management system. Once a user starts or joins a registered project, Cal will automagically track, tag, and index user activity while inside the project context. In other words, the system will manage project bookmarks, Google Docs, web search history, and emails. Ultimately, a user should be able to ask Cal, "Ok Cal, show me everything from my meeting with Mohit last Tuesday."

Ideally, Cal should be relatively transparent to the user. The system should determine with reasonable confidence that a user has joined or left a project context with minimal user interaction. This way, the user is focused on their project and not Cals interface. Cals presence will be discussed in more detail later as well as potential machine learning pipelines.

To conserve power, Cal constrains most of its sensor services to run only while a project is active. In this sense, Cal is contextually dependent on activity. The system has several layers of wake-up states inherent in the protocol.

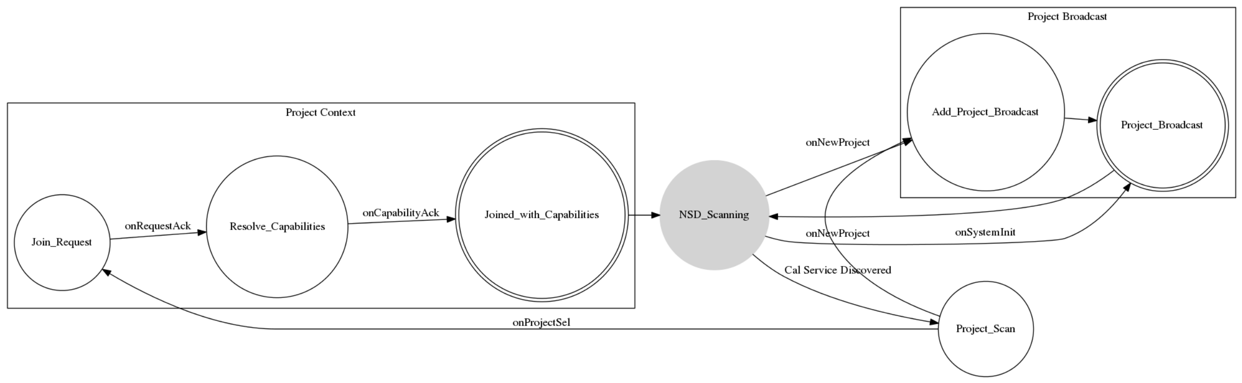
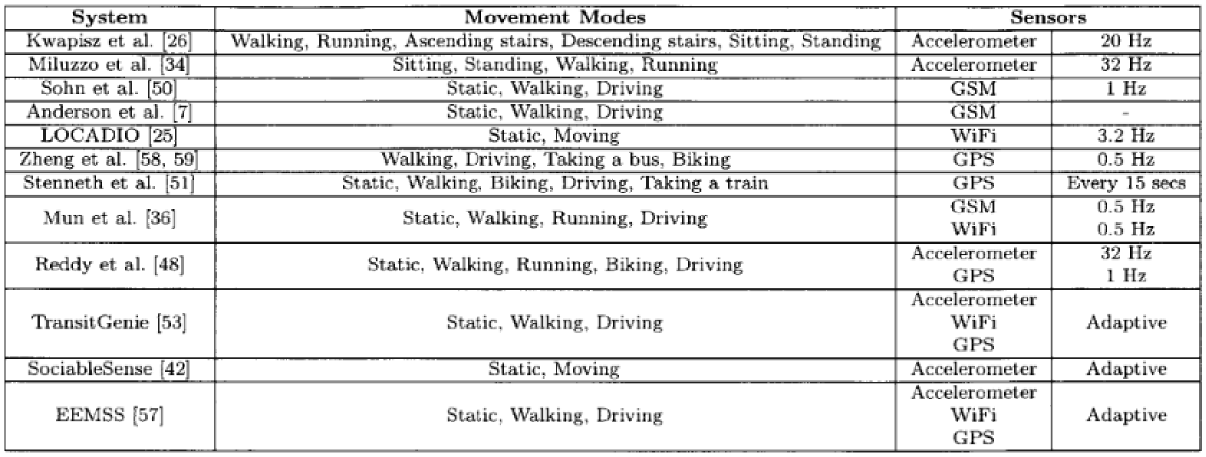


Figure 1.1: Simplified state machine of Cal Network service discovery. Some logic missing for clarity. Double circles mean fork as background process. Note, NSD scan has additional logic limiting its mobility to project scanning while currently in a project context. Most of the sensors are active in the Project Context only.

RELEVENT INFORMATION FROM PREVIOUS WORK



System	Movement Modes	Sensors	
Kwapisz et al. [26]	Walking, Running, Ascending stairs, Descending stairs, Sitting, Standing	Accelerometer	20 Hz
Miluzzo et al. [34]	Sitting, Standing, Walking, Running	Accelerometer	32 Hz
Sohn et al. [50]	Static, Walking, Driving	GSM	1 Hz
Anderson et al. [7]	Static, Walking, Driving	GSM	-
LOCADIO [25]	Static, Moving	WiFi	3.2 Hz
Zheng et al. [58, 59]	Walking, Driving, Taking a bus, Biking	GPS	0.5 Hz
Stenneth et al. [51]	Static, Walking, Biking, Driving, Taking a train	GPS	Every 15 secs
Mun et al. [36]	Static, Walking, Running, Driving	GSM WiFi	0.5 Hz 0.5 Hz
Reddy et al. [48]	Static, Walking, Running, Biking, Driving	Accelerometer GPS	32 Hz 1 Hz
TransitGenie [53]	Static, Walking, Driving	Accelerometer WiFi GPS	Adaptive
SociableSense [42]	Static, Moving	Accelerometer	Adaptive
EEMSS [57]	Static, Walking, Driving	Accelerometer WiFi GPS	Adaptive

Figure 2.1: [Yu] Features of some systems in activity detection using mobile devices. Note, most of these systems use additional external accelerometers.

In [Yau] [Yau1] and [Narseo], The authors explain how context aware applications play an important role in health administration, advertisements, improved user interfaces, and even advanced power management systems in phones. In particular, [Narseo] gives detailed overview on current methods of context awareness using Bluetooth, WiFi, GSM, and other wireless technologies. [Krum] gives an example where accelerometer information is combined with WiFi signal strenght indicators to estimate location and motion vectors.

The most detailed information, however, comes from [Xu] [Owusu] and [Aviv]. Taplogger ([Xu]) used the square of the L-2 Norm to distinguish tap events from walking and running. I tried recreating the results from *Acceleration readings in different contexts [Xu]* however, the *walking slowly* context really meant barely moving, and the *typing while sitting* context meant the phone was on the table while tapping the phone. Xu et al did give clever insight into what could constitute *contextual state*, which will help me when I design my HMMs.

Owusu et al, extend the work in [Xu] by experimenting with a larger set of feature selection criterea. I will take a look at how Cal performs given various feature selection criterea. They then used a Random Forest (RBF) support vector machine as their classifier. They suspect that Random Forests performed well because of the propensity for significant variability of feature values between instances of the same label. Although they had fairly poor results in the final password inference (probably due to the high granularity), they did provide a nice graph shown in *Inference accuracy by screen region granularity. The screen surface is partitioned into successively smaller blocks and evaluated for classification accuracy*. It should be noted, that we can determine a key press in a 8-partition screen with about 80% accuracy.

Unlike Owusu and Xu, Aviv takes a more formal signal processing approach to password inference using accelerometers. They differentiate statistical methods of feature extraction from short-sample Fourier polynomial methods. Aviv also explores multiple data normalization methods (Mean normalization, Linear Normalization, and Quadratic Normalization) and uses a simple windowing technique to sew sample bins together. They continue by construction a Hidden Markov Model, accounting for label trends, human and device effects, and movement noise.

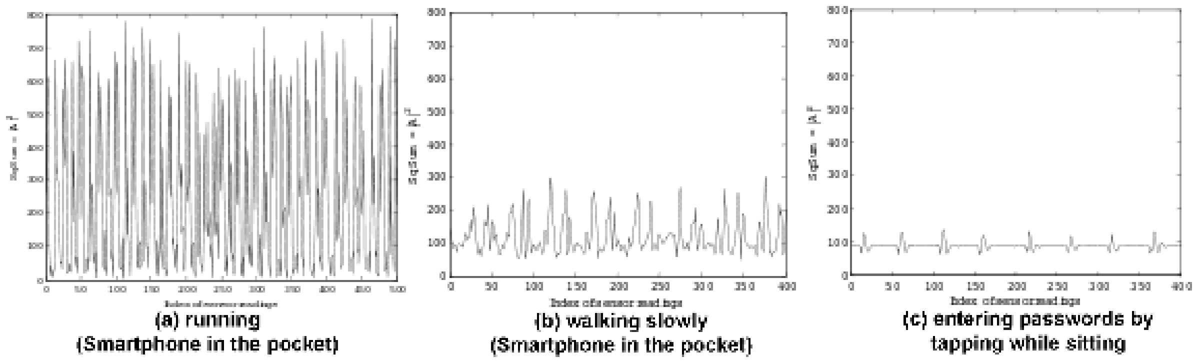


Figure 2.2: Acceleration readings in different contexts [Xu]

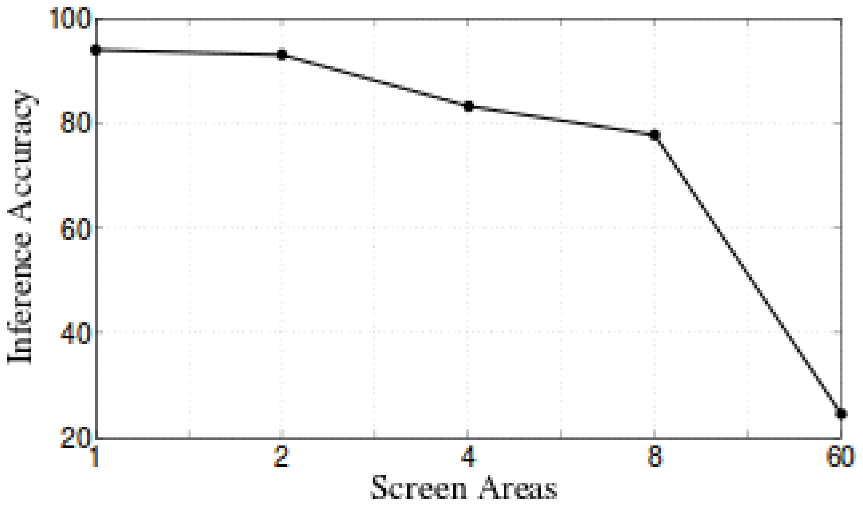


Figure 2.3: Inference accuracy by screen region granularity. The screen surface is partitioned into successively smaller blocks and evaluated for classification accuracy

CHAPTER

THREE

CAL SYSTEM

From here on, imagine Cal as two parts:

1. The User interface
2. A semi-transparent wrapper for existing Google Services

Obviously, the bulk of Cal is in 2.

3.1 Main Subsystems

3.1.1 NSD, Network Service Discovery

Network Service discovery as described by the Android API.

> "Adding Network Service Discovery (NSD) to your app allows your users to identify other devices on the local network that support the services your app requests. This is useful for a variety of peer-to-peer applications such as file sharing or multi-player gaming. Android's NSD APIs simplify the effort required for you to implement such features."

Since Cal, by default, should be interacting with Gmail, Google Docs, and a Web Browser we can assume that there is a network connection available. NSD also assumes an active network connection, whereas other service detection methods (e.g. WiFi Direct P2P) do not. Therefore, I will use NSD for simplicity.

As previously mentioned: To conserve power, Cal constrains most of its sensor services to run only while a project is active. In this sense, Cal is contextually dependent on activity. The system has several layers of wake-up states inherent in the protocol.

3.1.2 Nth-Sense

Sensor readings need to occur in the background, i.e. Activity/fragment independent. I will design a scheduled background service to manage sensor events. These sensor readings will be temporarily saved to an application local database or bin buffer. The system will pre-process this data and then notify the Cal system that a Data Package is ready for post processing/handling.

3.1.3 Context Detection System Machine Learning

The final machine learning pipeline is heavily dependent on what sensors I use, and how fine a granularity I wish to make the Context Detection System. I expect to use a combination of Support Vector Machines (SVMs) and Hidden Markov Models (HMMs) for the actual context prediction. Also, I may consider using genetic algorithms to help decide the optimum ML pipeline for a particular user. Again, this all depends on time constraints and what the data looks like. See Cal Time Line for more information.

3.1.4 Cal Remote Server and Database Interface

For privacy reasons, Cal will only keep users personal information (bookmarks, project contacts, etc) saved locally on the app or within Google's secure backup service.

However, I will still need to design a network interface for a remote database/server. This remote entity will handle most of the heavy ML computations and data management. The remote server will determine the optimal ML algorithm and parameters on a per-user basis, serialize these methods, and transmit them to the Cal Application. This way, at least latency critical Cal methods (e.g. walking away from a project) will be transparent to the user.

3.1.5 User Interface

For the most part, Cal should behave transparently and automatically to the user. The user still has the option to manually create a project or change projects. however, Cal should do its best to pick the correct project to join. Otherwise, if Cal is unsure of which project to join, or even if the user would like to join the project, Cal should inform the user and provide a filtered list of highly probable projects.

3.1.6 Voice Recognition (Optional)

It turns out that adding additional voice capabilities to either Google Now or a Custom in-app method is fairly simple. If time permits, I will try to add this feature. Voice capabilities enhance the user experience, which may make the app more popular. However, adding this feature is fluff and not completely necessary.

3.2 Cal Time line

CHAPTER

FOUR

INITAL EXPERIMENTS

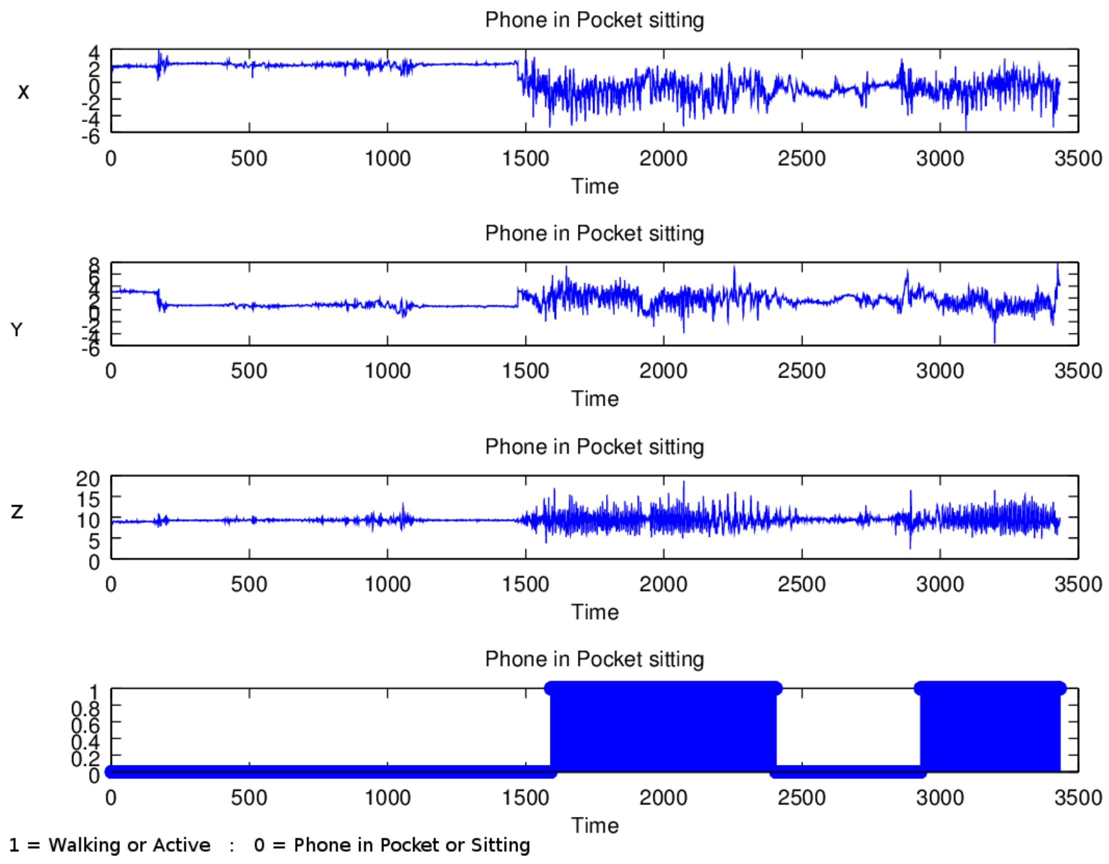


Figure 4.1: First experiment with phone accelerometer. I was sitting at my desk for a bout ten minutes, walked to class, waited for about 5 minutes, went and got a snack, then returned to class. Notice how the transitions between states are noisy.

4.1 Some Expected Graphs

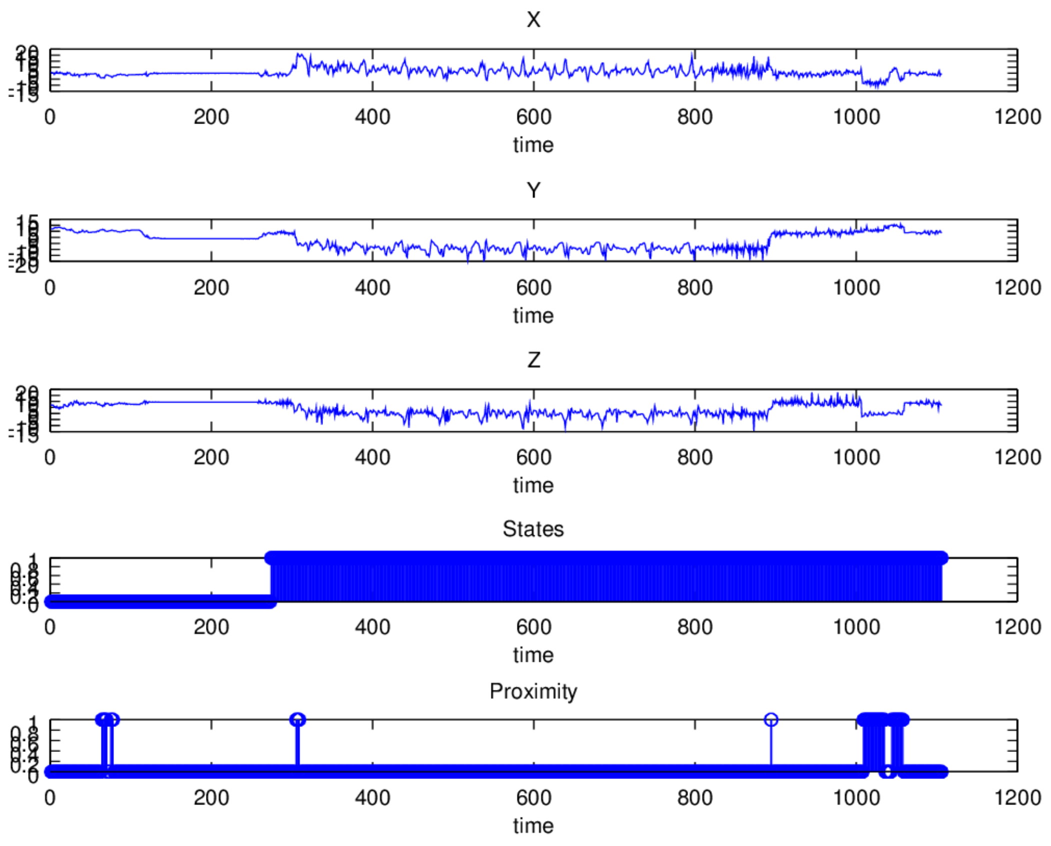


Figure 4.2: Here, I added a proximity sensor to the sensor manager. The first burst of spikes was me playing with the sensor. The second burst was me putting my phone in my pocket. The last group of spikes was me actually answering a phone call (my phone battery died shortly after that). A state value of 1 represents walking, 0 means phone was in my lap or I was playing with it at my desk.

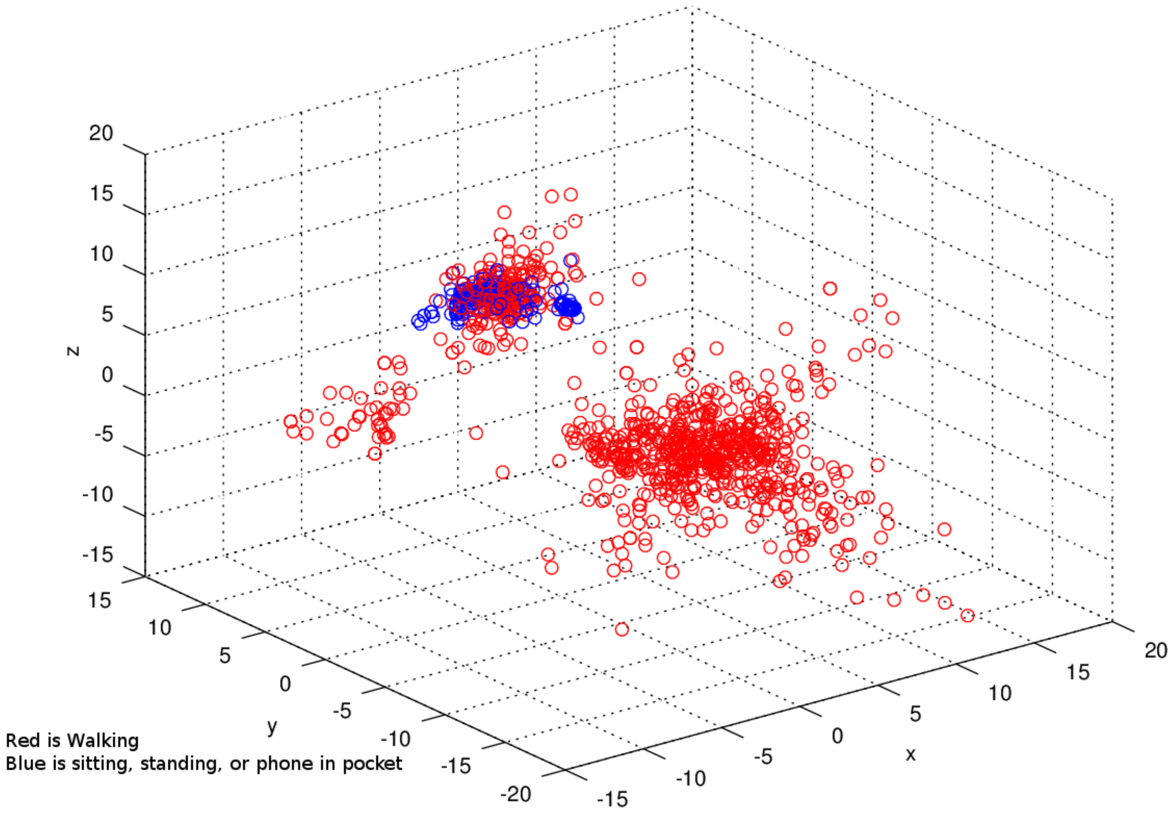
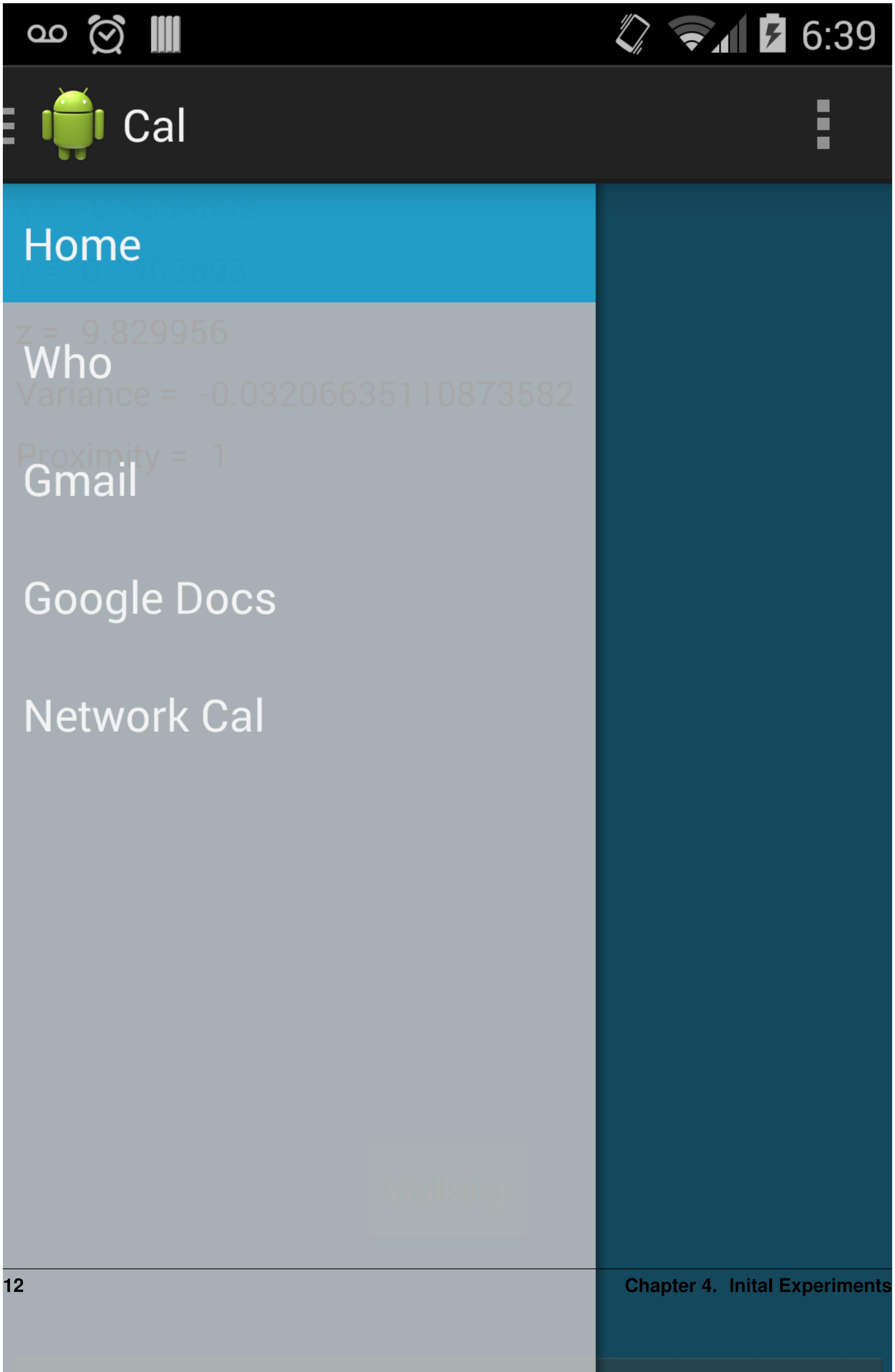
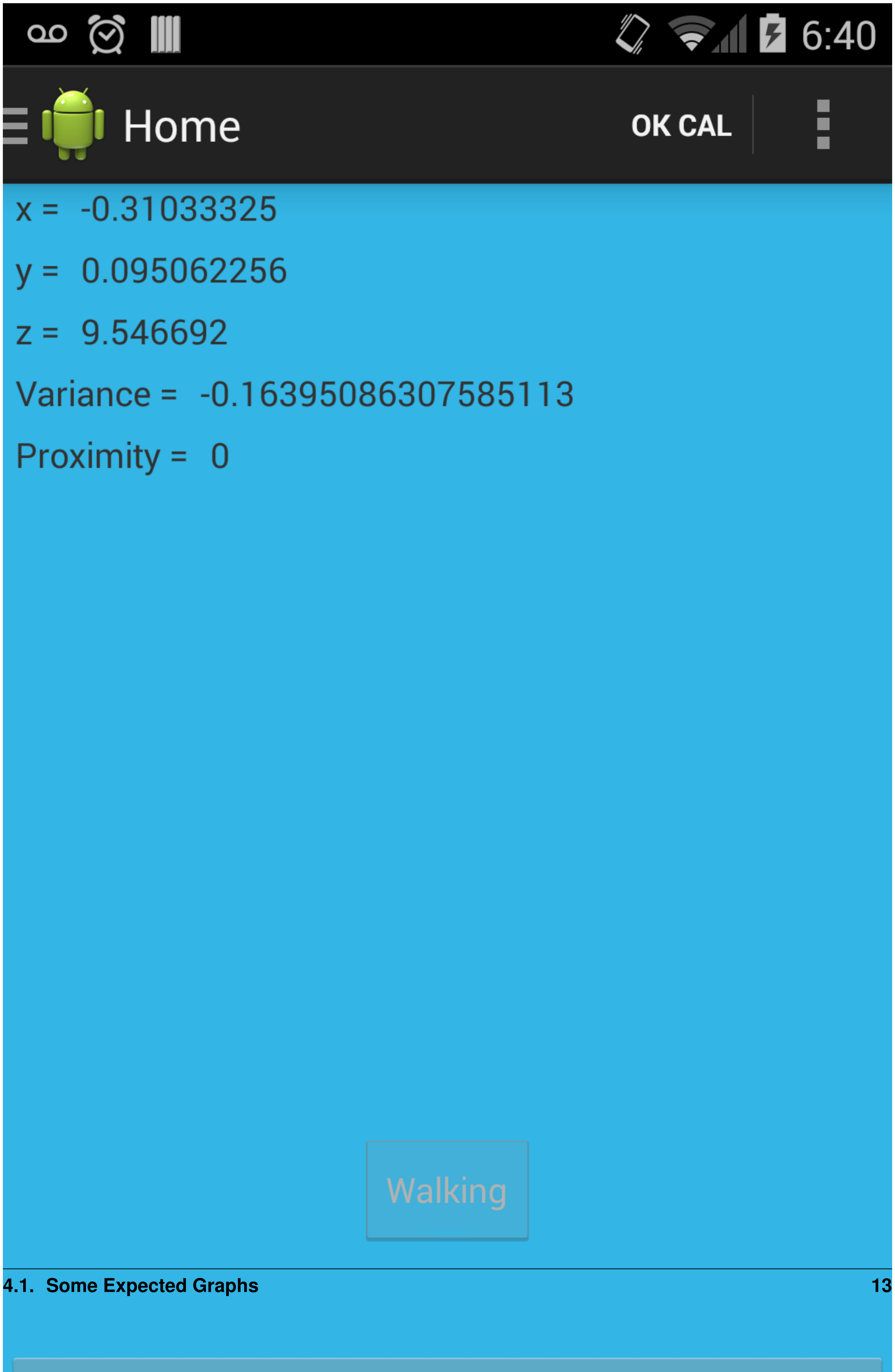


Figure 4.3: 3D Scatter plot of accelerometer data (same data samples as previous plot with proximity sensor). Notice the tight cluster of blue? This is where I was playing with my phone at my desk. One red cluster is where my phone is in my pocket, the other is where I was talking on the phone.



x = -0.3556803
y = 0.1402893
z = 9.829956
Variance = -0.032066351108735 82
Proximity = 1



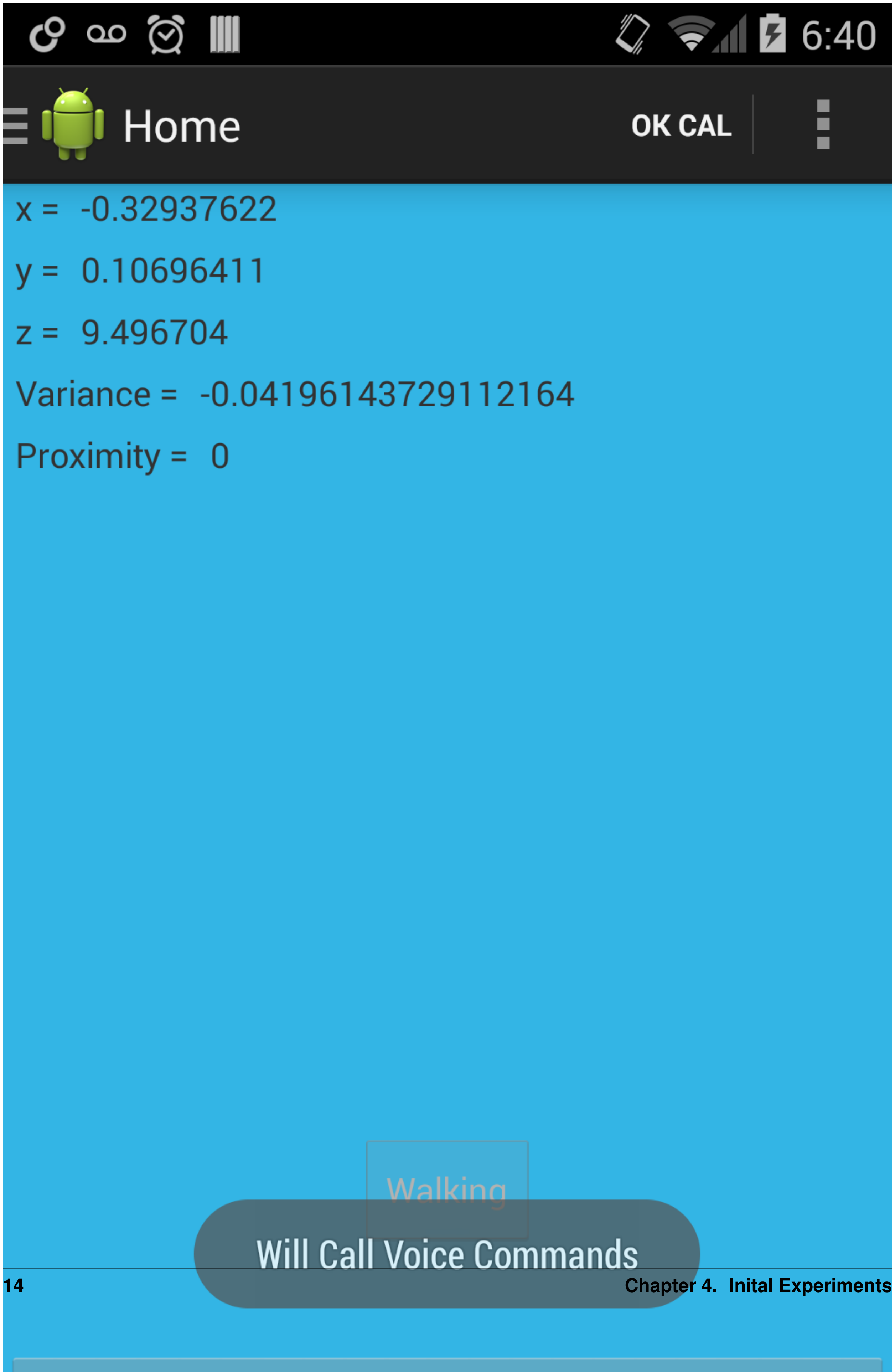
x = -0.31033325

y = 0.095062256

z = 9.546692

Variance = -0.1639508630758511З

Proximity = 0



x = -0.32937622

y = 0.10696411

z = 9.496704

Variance = -0.04196143729112164

Proximity = 0

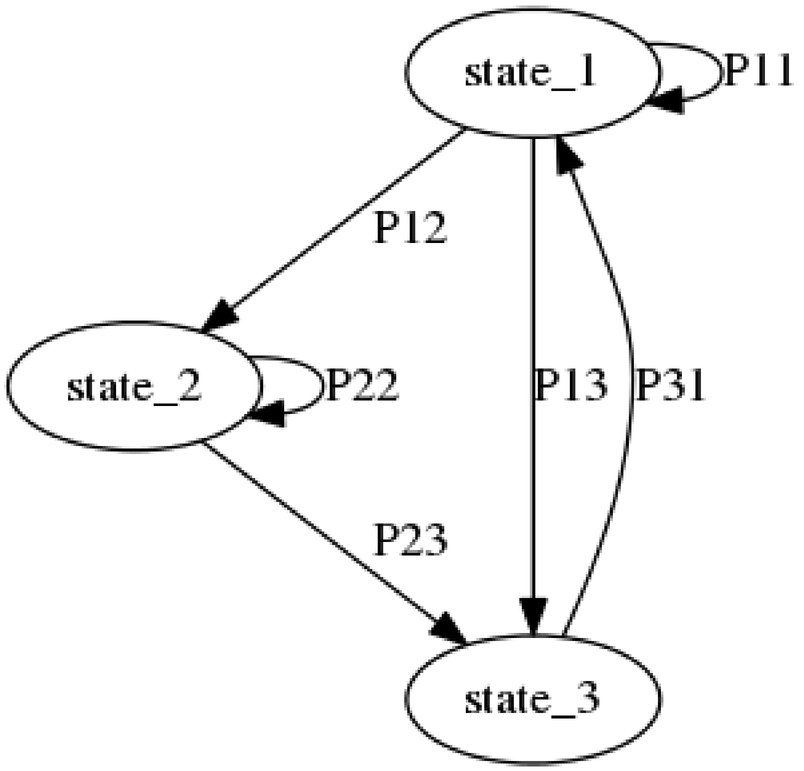


Figure 4.7: Placeholder Hidden Markov Model state machine. Will determine the actual model later.

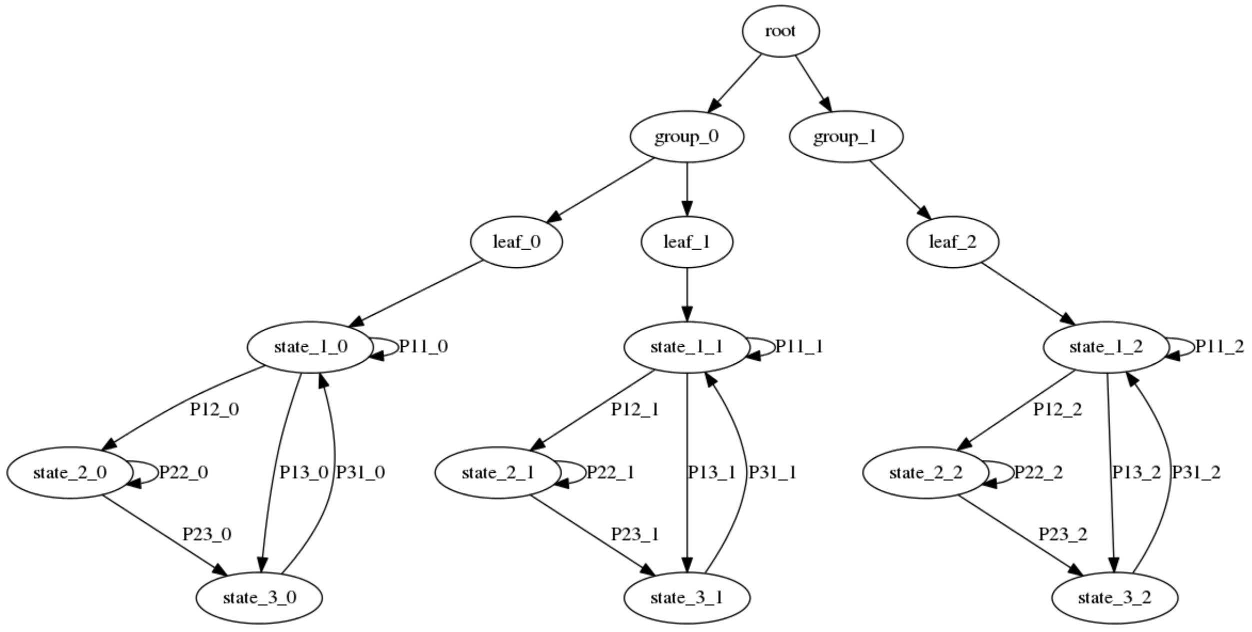


Figure 4.8: I expect that we might be able to classify different behavior patterns into a decision tree (may be in a calibration step). This way I can get improved performance on my different ML methods.

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

[Yu] Yu-Han Chen. Ada: Context-Sensitive Context-Sensing on Mobile Devices. http://dspace.mit.edu/bitstream/handle/1721.1/82377/862074810.pdf?sequence=1

[Yau] 19. Yau and F. Karim, "An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments," in *Real-Time Systems*, 2004

[Yau1] 19. Yau and F. Karim, "Reconfigurable Context Sensitive Middleware for Pervasime Computing", 2002

[Krum] 10. Krumm and E. Horvitz. Locadio: "Inferring motion and location from wi-fi signal strengths." IEEE Computer Society, 2004.

[Lee] Youngki Lee, S. S. Iyengar, Chulhong Min, Younghyun Ju, Seungwoo Kang, Taiwoo Park, Jinwon Lee, Yunseok Rhee, and Junehwa Song. 2012. MobiCon: a mobile context-monitoring platform. Commun. ACM 55, 3 (March 2012), 54-65. DOI=10.1145/2093548.2093567 http://doi.acm.org/10.1145/2093548.2093567

[Jagadish] Jagadish, H.V.; Beng Chin Ooi; Quang Hieu Vu; Rong Zhang; Aoying Zhou, "VBI-Tree: A Peer-to-Peer Framework for Supporting Multi-Dimensional Indexing Schemes," Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on , vol., no., pp.34,34, 03-07 April 2006 doi: 10.1109/ICDE.2006.169 keywords: {Binary trees;Databases;Indexing;Load management;Multidimensional systems;Peer to peer computing;Proposals;Query processing;Robustness;Tree data structures}, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1617402&isnumber=33902

[Xu] 26. Xu, K. Bai, S. Zhu; TapLogger: Inferring User Inputs On Smartphone Touchscreens Using On-board Motion Sensors. www.cse.psu.edu/~szhu/papers/taplogger.pdf

[Owusu] 5. Owusu, S. Das, A. Perrig, J. Zhang; ACCessory: Password Inference using Accelerometers on Smartphones. Carnegie Mellon University. http://www.hotmobile.org/2012/papers/HotMobile12-final42.pdf

[Aviv] 1. (a) Aviv, B. Sapp, M. Blaze, J. Smith: Practicality of Accelerometer Side Channels on Smartphones. http://www.cs.swarthmore.edu/~aviv/papers/aviv-acsac12-accel.pdf

[Narseo] 14. Vallina-Rodriquez, J. Crowcroft. The case for context-aware resources management in mobile operating systems. http://www1.icsi.berkeley.edu/~narseo/papers/context-awareness.pdf