CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Wearable Computing Approach for Indoor Positioning

A thesis submitted in partial fulfilment of the requirements
For the degree of Master of Science in
Software Engineering

By
Anu George Enchackal

May 2017

The thesis of Anu George Enchackal is approved:

Michael Barnes, PhD	Date
Steven Fitzgerald, D.Sc.	Date
Ani Nahapetian, PhD, Chair	Date

California State University, Northridge

Acknowledgement

I would like to express my heartfelt gratitude to my advisor Prof. Ani Nahapetian for the continuous support of my masters' study and research, for her patience, motivation, enthusiasm, and immense knowledge.

I would like to thank Professor Michael Barnes for all his support during my research phase and Professor Steven Fitzgerald for all his help and support in the implementation & results phase. Thank you, dear professors, for your kind and helpful insights.

I would also like to thank Kimon Rethis and Olivia Herstein from Department of Marketing and Communications -University Advancement of CSUN. Thank you Kimon, for helping me to get crucial data from Physical Plant Management and thank you Olivia for always motivating me in whatever way you could.

I would also like to thank Jenna, Mayan and Melissa, my roommates, for putting up with my aloofness for past year and giving me the space I needed to get my thesis done.

I thank my family back in a whole different continent for praying for my success and I express my heartfelt gratitude to my friends and everyone who were there to help me.

I thank Almighty god for keeping everything alright until the very end.

Table of Contents

List of Figures

List of Tables

Abstract

Wearable Computing Approach for Indoor Positioning

By

Anu George Enchackal

Master of Science in Software Engineering

Existing approaches for indoor navigation using wearable sensors weigh heavily on path to map matching techniques. This work is an attempt to identify the user's location within a pre-selected portion of a building from users' own activities. The study employs the concept of activity-mapping for locating the user. A comparative study of various machine learning classifiers for human activity recognition was conducted and found the Multiclass Forest Classifier as the most suitable classifier for this study. A supervised dataset for training the model was collected using a bespoke Android companion application running on Samsung Galaxy Core Prime mobile and Samsung Gear Live smartwatch. This data consisted of 20 samples each of 15 user activities, totaling 15346 sample points of accelerometer readings. The trained model has an average accuracy of 0.936209 and overall accuracy of 0.521569. The decrease in overall accuracy is due to the similarity of certain activities such as cabinet-open-close and fridge-open-close. Accuracies of each observed user activities are left-turn-walk: 34.4%, right-turn-walk: 28.7%, going-down-stairs: 59.9%, going-up-stairs: 44.2%, elevator-up: 50.2 %, elevator-down: 63.1%, elevator-button-press: 49.7%, cabinet-open-close: 64.8%, standing: 37.1%,

U-turn-walk: 35.8%, fridge-open-close: 52.0%, walking: 36.1%, switch-on-off: 45.4%, waving: 65.6% and open-close-door: 56.3%. A portion of a building for indoor positioning was implemented as a graph data structure for computation. The implementation was tested and evaluated by conducting 22 indoor positioning experiments consisting 7793 accelerometer readings. It is observed that the accuracy of human activity recognition model directly influences the accuracy of the indoor positioning.

Chapter 1 - Introduction

1.1 Introduction

This work presents an approach for solving the problem of indoor positioning using wearable sensors. Motivation for conducting this study was lack of an indoor navigation or positioning approach which has a minimum emphasis on map matching. Hence the main distinction between this study and earlier studies is that the approach presented here employs the concept of an 'activity map' to determine the user's location. The idea is to recognize and use, the user's activities, such as 'opening a door', within a building to locate the user in the building. Recalibration is done using minimalistic map matching techniques. Map matching is a graph data structure function rather than superimposing on actual maps. Another advantage of this study is that its implementation does not require location/GPS to be turned on for functioning.

This study was conducted using two Android applications – one for data collection and one for actual localization. Tools used include Azure Machine Learning Studio, Waikato Environment for Knowledge Analysis (WEKA) and Amazon Web Services. A bespoke controller web service was used for computation. Android applications were deployed on a Samsung Galaxy Core Prime mobile device and a Samsung Gear Live smart watch.

1.2 Research Goals

This research targets the following.

- Identify user activity within a time frame
- Identify current position of user within a building, based on user activities within the building

- Evaluate the accuracy of the above stated identifications

1.3 Research Approach

Accelerometer sensor data for various human activities of a single user is recorded into cloud database using an implementation developed in Android. Data is analysed and trained in WEKA and Azure Machine Learning Studio to find suitable classifier. Based on the results from data analysis a suitable machine learning classifier is chosen for the training experiment conducted in cloud. This training experiment is converted to a predictive experiment and deployed as a Representational State Transfer (REST) web service. An ‘activity map’ for various locations within the building is developed. In the final phase, a controller web service intakes an activity map for the building and attempts to predict the user’s location within the building using predictive web service. The results are displayed on a uniquely designed Android application.

Chapter 2 - Related Work

Growth and development of micro and small electronic appliances with high computational power for low cost, have immensely helped ubiquitous computing researchers in their studies [1], which further lead to the development of context aware systems - mobile systems that can sense their physical environment, and adapt their behavior accordingly. Human activity recognition techniques developed by various studies often track their users' hand and wrist movements to meet their purpose.

Wrist motion monitoring using wearable sensors - smartwatches, wrist bands, wrist mounted sensors - have many applications in improving human experience and quality of life such as rehabilitation applications [2], smoking action detection [3], detecting eating moments [4], estimating how much the user ate [5]. In most studies, data from motion sensors in the wrist, captured at a suitable frequency, is used to train a machine learning model for predicting human activity and events [3][4][5], although a recent study [6] took up the challenge of not employing machine learning algorithms. RISQ, a popular study relies on a wristband embedded with a single, low power 9-axis inertial measurement unit (IMU) to identify smoking gestures [3]. In a recent study [4], authors describe the machine learning implementation and evaluation of an approach for inferring eating moments based on 3-axis accelerometry collected with accelerometer from a popular off-the-shelf smartwatch. Another study [5] proposes that audio and motion data can be combined to accurately estimate food type and amount for each intake based on hypothesis that while acoustic sensors can distinguish between different food textures, motion sensors can help discriminate between soft foods based on head or wrist position.

Gesture recognition involves identifying arm gestures, hand gestures and finger gestures. Xu, H. Pathak and Mohapatra attempts to show in their study [7] that motion energy measured at the smartwatch is sufficient to uniquely identify user's hand and finger gestures using essential features of accelerometer and gyroscope data that reflect the movements of tendons (passing through the wrist) when performing a finger or a hand gesture. Glove based gesture recognition techniques are popular among scholars. A study developed Hand Data Glove [8] which is used to capture current position of the hand and the angles between the joints for recognizing gestures such as pointing and air-writing (path tracking). Another study [9] uses the IR camera which tracks the infra-red emitter attached to the user's hand gloves to produce a sequence of motion-points, which are then analyzed syntactically to recognize the intended hand gesture. Heumer, Guido, Heni Ben Amor, Matthias Weber, and Bernhard Jung addresses the problem of Identifying the classifier (or family of classifiers) which is best suited for the problem domain of "grasp recognition from raw data glove sensor data" [10]- grasp is the hand gesture of holding different shaped objects.

When trained models for human activity recognition go in to be practical predictive application, they face challenges due to 'activity spotting' [11]. Activity spotting is continuous activity recognition where a meaningful human activity is happening in between variety of other activities. Hence certain meaningful activities are hard to classify or recognize in real situations, which further emphasize the need for human activity recognition on a fine-grained level [11]. One of the studies [11] addressing gesture activity spotting problem presented an approach by using partitioning of human motions where simple similarity search is employed to pre-select signal

sections from body-worn sensors, which are likely to contain specific motion events, are then classified using hidden Markov models. Their case study activities include “opening the door” event. “Opening the door” event or activity is presented as one of the hard to classify events by another study that attempts to realize a unified model for multimodal continuous activity recognition also known as activity spotting [12]. Opening door activity was studied by some studies those attempted to identify users from objects used, where data from wearable sensors were used to identify objects or their ‘hallmarks’ [13] [15]. A study published in 2006 [14], presented a system which recognizes user activity from wearable sensors with a neural network and combined the information of the state-of-use artifact to an activity with a linkage condition. One of the examples would be ‘walking to the door’. As an attempt to resolve recognition issues between similar activities such as “standing”, “elevator-up”, “elevator-down” they combined those activities into one class “stand”. Their system to identify user of sentient artifact was presented in a previous work [15] which dealt with identify the user who opened a door.

Scholars have pursued indoor navigation using wearable sensors in the past. A study [16] explored indoor navigation problem using machine learning techniques to infer a person's location from naturally-occurring signals such as magnetic fields from steel beams in the walls, fixed arrangements of fluorescent lights, and temperature gradients across rooms in the environment along with user’s distinctive acceleration patterns while walking up or down staircases, or riding an escalator or elevator from a “utility belt”. Bao, Haitao, and Wong [17] infers that step counting-based dead-reckoning has been widely accepted as a cheap and effective solution for indoor pedestrian tracking using a hand-held device equipped with motion sensors. To compensate for the

accumulating error in a dead-reckoning tracking system, they proposed a map matching enhanced particle filter as a robust localization solution to filter out impossible locations. Perhaps closest to this work is that of [18] FootSLAM which approaches indoor navigation using a Bayesian estimation approach for simultaneous mapping and localization for pedestrians based on odometry - human step measurements - with foot mounted inertial sensors. It uses even noisy and drift-prone odometry measurements from indoor user movement to detect features like turns, doors, and walls, which can be used to build a form of a map of the explored area, especially when these features are revisited over time. Thus, results are achieved without prior building knowledge. Ascher, Christian, Kessler, Weis, and Trommer [19] propose a path to map matching algorithm for multi floor indoor environments to eliminate drift experienced by inertial sensor based pedestrian navigation system. Sensor basis for their approach is a Dual IMU System, which takes advantage of zero velocity updates from a foot mounted IMU and records the torso dynamics from a second, torso mounted unit. In multi floor scenarios, they propose to use not only flat floor plans but also transitions like staircases as well as ladders and elevators. As another study [20] points out, the need to recognize the motion mode of the user such as walking up or down stairs, taking the elevator, and standing or walking on an escalator, is useful in portable navigation to improve the positioning estimation, they chose to detect change in height to achieve that. Their height motion mode recognition module has been implemented in real-time on several brands of various consumer portable devices, including smartphones, tablets, smartwatches, and smart glasses.

This work approaches the problem of identifying a user's location within a pre-selected portion of a building using data from smartwatch accelerometer.

Chapter 3 - Data Collection and Analysis

As the objective of the study was to find an approach to indoor navigation using human activity recognition, a supervised dataset to train a machine learning predictive model was needed. Dataset was collected using an Android application.

3.1 Data Collection Method

Study was conducted on a single user who performed activities such as opening the door, waving, turning on light switches, walking upstairs, walking downstairs, pushing elevator buttons, walking: making right turns, walking: making left turns, walking: making U-turns. Data was collected at highest sampling rate offered by Android by registering sensor listener to constant SENSOR_DELAY_FASTEST. Code is given as below.

```
    public void onResume() {
        super.onResume();
        mSensorManager.registerListener( this,
mSensor,mSensorManager.SENSOR_DELAY_FASTEST);
    }
```

Hence wear sends data to mobile as soon as values changes, as given in the code below.

```
  private void detectShake(SensorEvent event) {
      long now = System.currentTimeMillis();
       if((now - mShakeTime) > SHAKE_WAIT_TIME_MS) {
          mShakeTime = now;
          params = "/color:"+mColor+"/sensor:acce/x:" + Float.toString(event.values[0]) + "/y:"
+ Float.toString(event.values[1])+ "/z:" + Float.toString(event.values[2]);
          Log.d("PARAMS acce", params);
          sendData.sendToast(params, getActivity());
      }
  }
```

An Android wear application was built to collect data, to detect wrist motion to recognize user activities and perform indoor navigation in a preselected building with the help of algorithm and prior knowledge about the building. Devices that are used to test application are Android mobile and Samsung Gear Live smart watch.

3.2 Data Collection Application Architecture

Android research application has two components. Host component which runs on Android mobile while wear component that runs on smart watch. Data was written on a cloud database and data analysis was done on laptop. Architecture of the application is depicted in Figure 3.2.1.

Wear module sends accelerometer readings to mobile module of the application over Bluetooth. Mobile module sends received accelerometer readings and user inputted observed user activity to a cloud database over HTTP using Android library named Volley. Data recorded by the application is used for supervised learning by training experiment in the cloud.

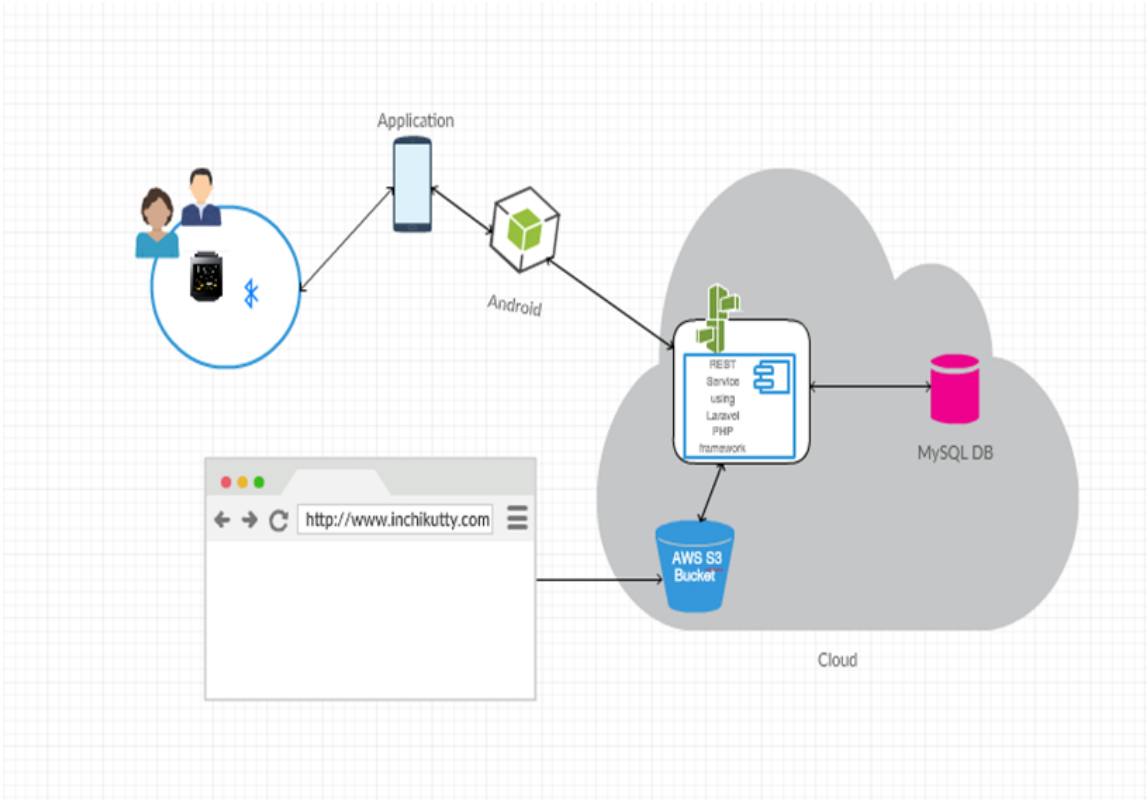

Figure 3.2.1 Architecture for Data Collection Application

Observed user activity is recorded by user from mobile module's main activity screen as depicted in Figure 3.2.2.

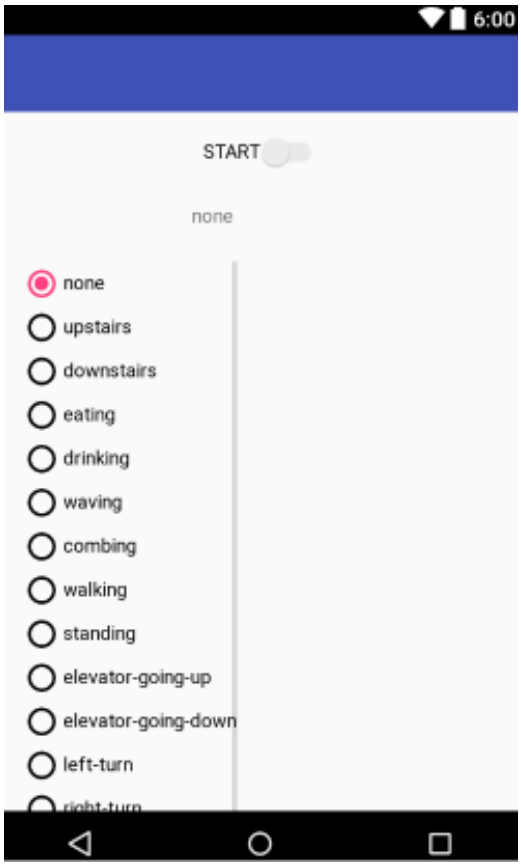

Figure 3.2.2 Screenshot of Data Collection Application Running on Mobile

3.3 Data

Data was stored on a MySQL database on Amazon Web Services account. Format of a single reading is as per Figure 3.3.1.

ID	User ID	Observed User Action	Sensor	x	y	z	Time	Date
Data set id	User's id in database	Observed human activity recorded by research application using the input from researcher	accelerometer	Acceleration along X axis	Acceleration along Y axis	Acceleration along Z axis	Time of occurrence of recorded data set	Activity date

Figure 3.3.1Format of Supervised Dataset in Database

Collected data comprised of 15346 sample points of accelerometer readings for 15 distinct user activities. Observed user activities are left-turn-walk, right-turn-walk, going-down-stairs, going-up-stairs, elevator-up, elevator-down, elevator-button-press, cabinet-open-close, standing, U-turn-walk, fridge-open-close, walking, switch-on-off, waving and opening-door. There were 20 samples of each activity. Sample distribution across observed user activities are depicted in Figure 3.3.1. Non-uniform distribution of data in the pie chart is because, certain events are shorter in time span than the others despite having equal number of samples. For example, turning the activity "switch-on-off" is a shorter event than the activity "walking" for 4 meters.

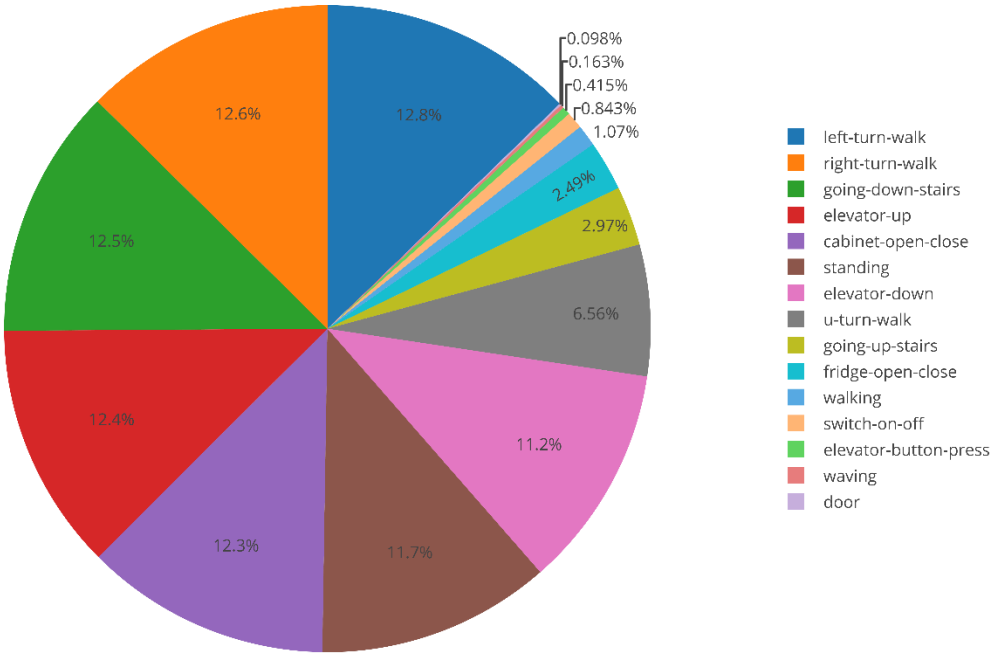

Figure 3.3.2 Activity Distribution Pie Chart of Collected Data

3.4 Machine Learning Classifier Analysis

Machine learning classifier analysis was performed in two different setups, WEKA and Azure Machine Learning Studio, to identify a more suitable machine learning algorithm for training experiment. Features of data used for machine learning experiments were acceleration along X, Y & Z axes.

3.4.1 Analysis of Various Machine Learning Classifiers in WEKA

Supervised dataset was analysed and studied in WEKA explorer to identify suitable machine learning algorithm for predictive web service. Classifiers ran at this stage were J48, Hoeffding tree and Decision stump. Other WEKA Classifiers were ruled out of study based on their inability to train on large data.

An overview of metrics for measuring performance for classification are given in Table 3.4.1 and J48 classifier proved to be more suitable based on their metrics (Refer Table 3.4.2) from experiments conducted in WEKA.

Table 3.4.1 Metrics for Measuring Performance for Classification

Accuracy	The proportion of true results to total cases.
Precision	The proportion of true results to positive results.
Recall	The fraction of all correct results over all results.
F-score	A measure that balances precision and recall.

AUC	A value that represents the area under the curve when false positives are plotted on the x-axis and true positives are plotted on the y-axis.
Average Log Loss	The difference between two probability distributions: the true one, and the one in the model.
Train Log Loss	The improvement provided by the model over a random prediction.

This study uses overall accuracy as a measure of performance to evaluate various classifiers.

Table 3.4.2 Classifier Training Outputs in WEKA

Classifier	Test Mode	Overall Accuracy in Percent
J48	10-fold cross-validation	50.215
J48	Percentage split at 50%	48.1298
Hoeffding Tree	10-fold cross-validation	31.5392
Hoeffding Tree	Percentage split at 50%	28.6198
Decision Stump	10-fold cross-validation	17.8744
Decision Stump	Percentage split at 50%	17.92

3.4.2 Analysis of Various Machine Learning Classifiers in Azure Machine Learning Studio

WEKA training experiments proved that a multiclass tree classifier such as J48 would be more suitable for running predictive experiment. Multiclass decision forest turned out to be more suitable in Azure Machine Learning setup based on their metrics

(Refer Figures 3.4.1 – 3.4.4). Classifiers studied in Azure Machine Learning setup were multiclass neural network, multiclass logistic regression, multiclass decision jungle and multiclass decision forest.

Table 3.4.3 Classifier Training Outputs in Azure Machine Learning Studio

Classifier	Test Mode	Overall Accuracy in Percent
Multiclass Neural Network	10-fold cross-validation	34.9146
Multiclass Neural Network	Percentage split at 50%	32.9988
Multiclass Logistic Regression	10-fold cross-validation	29.1933
Multiclass Logistic Regression	Percentage split at 50%	28.8805
Multiclass Decision Jungle	10-fold cross-validation	50.1694
Multiclass Decision Jungle	Percentage split at 50%	49.3809
Multiclass Decision Forest	10-fold cross-validation	52.1569
Multiclass Decision Forest	Percentage split at 50%	49.016

3.4.3 Rationale behind Choice of Classifier Based on Classifier Analysis

Multiclass Decision Jungle and Multiclass Decision Forest emerged as contenders for suitable classifier for the experiment based on the results from comparative study on various classifiers (Refer Table 3.4.2 and Table 3.4.3). Multiclass decision jungle, when run on cross-validation test mode, was unable to recognize “U-turn-walk” and misclassified this activity to “going-down-stairs” as depicted in confusion matrix given in Figure 3.4.1. Hence the classifier was ruled out for further experiments.

Multiclass Decision Forest Classifier when run on 10-fold cross validation test mode had an overall accuracy of 0.52. Confusion matrix for the model is given in Figure

3.4.2. When pitted against percentage split test mode results it was found that 10-fold cross validation had better performance in terms of individual and overall accuracy. Classifier had better accuracy (40.3%) for “walking” in percentage split test mode, as opposed to (36.1%) 10-fold cross validation mode.

Accuracies of each observed user activities in percentage split mode are left-turn-walk: 33%, right-turn-walk: 28.3%, going-down-stairs: 55.5%, going-up-stairs: 42.9%, elevator-up: 48.4%, elevator-down: 56.1%, elevator-button-press: 44%, cabinet-open-close: 64.8%, standing: 28.4%, U-turn-walk: 31.1%, fridge-open-close: 47.8%, walking: 40.3%, switch-on-off: 39.3%, waving: 63.3% and open-close-door: 54.4%. Individual accuracies of various activities in 10-fold cross validation test mode are left-turn-walk: 34.4%, right-turn-walk: 28.7%, going-down-stairs: 59.9%, going-up-stairs: 44.2%, elevator-up: 50.2 %, elevator-down: 63.1%, elevator-button-press: 49.7%, cabinet-open-close: 64.8%, standing: 37.1%, U-turn-walk: 35.8%, fridge-open-close: 52.0%, walking: 36.1%, switch-on-off: 45.4%, waving: 65.6% and open-close-door: 56.3%.

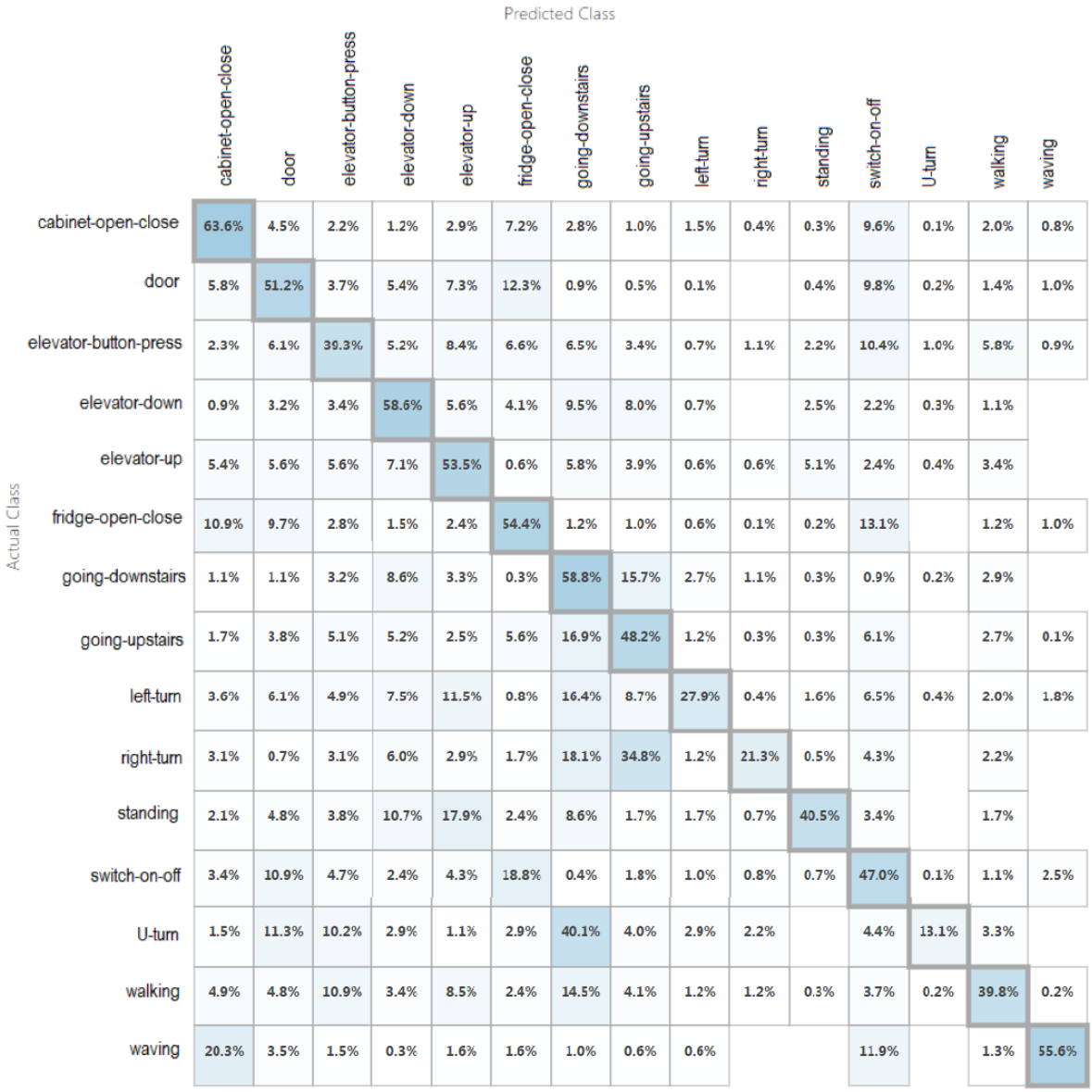

Figure 3.4.1 Confusion Matrix of Multiclass Decision Jungle Classifier's 10-Fold Cross Validation Trained Model

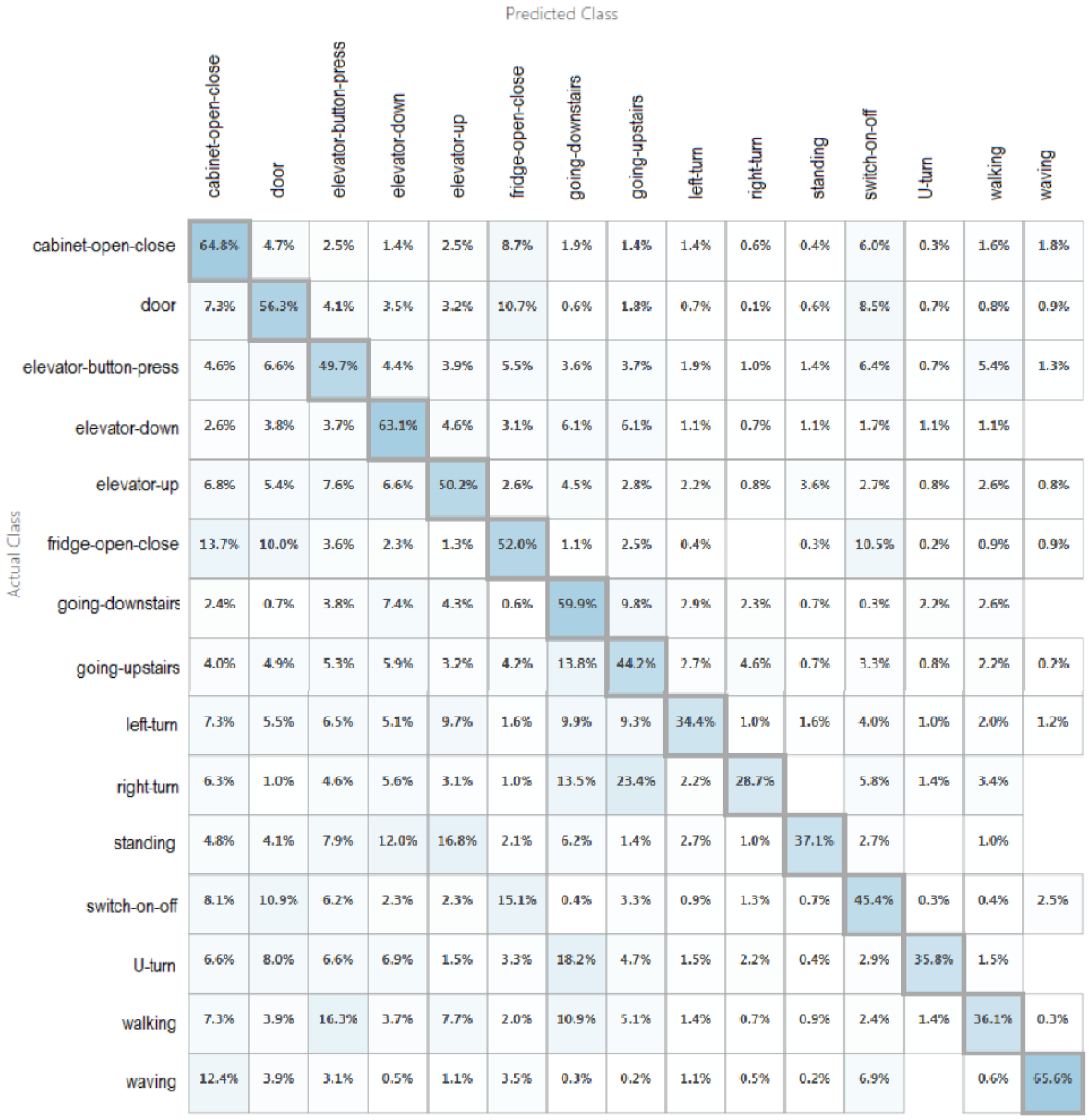

Figure 3.4.2 Confusion Matrix of Multiclass Decision Forest Classifier's 10-Fold Cross Validation Trained Model

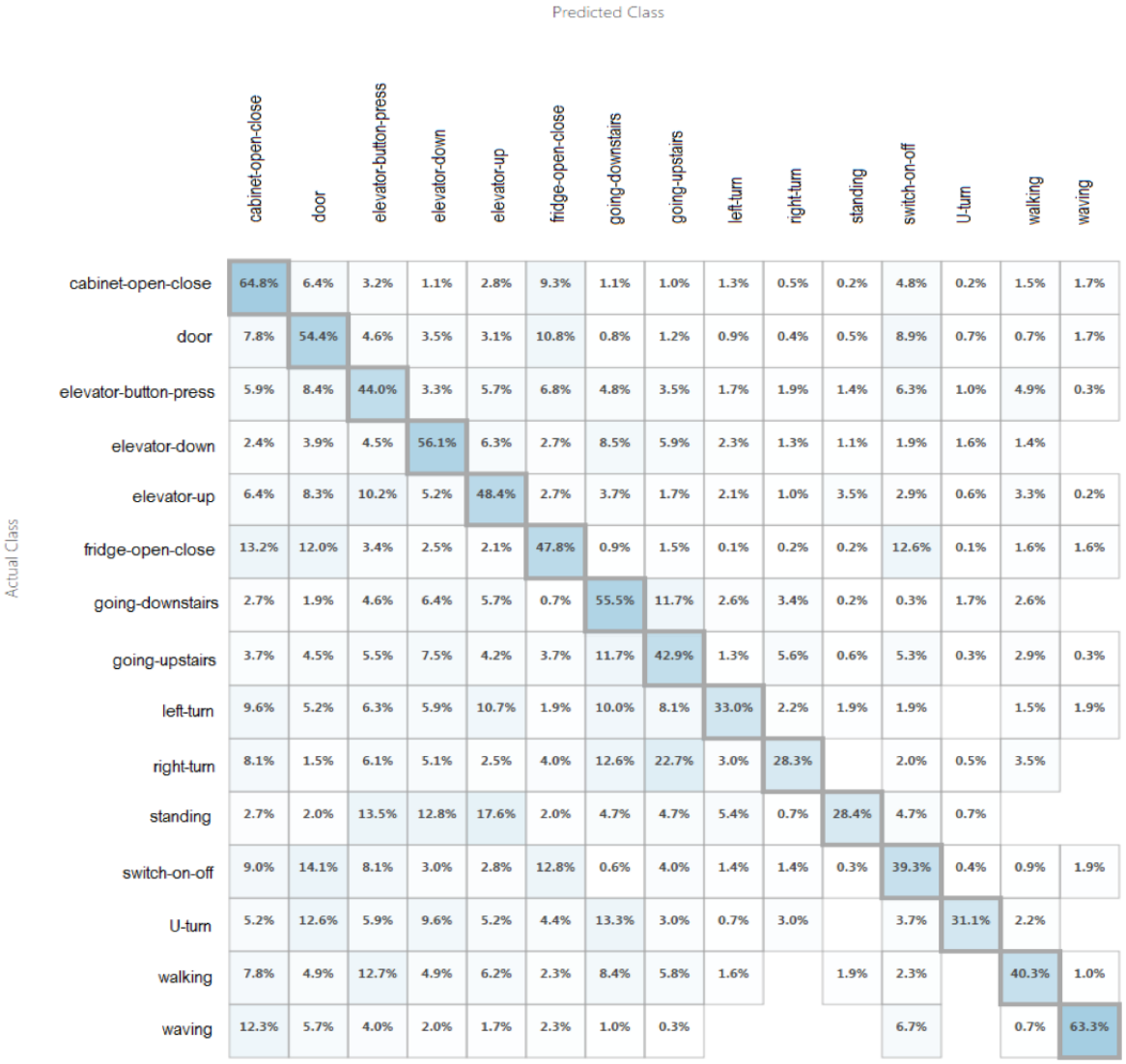

Figure 3.4.3 Confusion Matrix of Multiclass Decision Forest Classifier's Percentage Split Trained Model

3.5 Multiclass Decision Forest Classifier

Based on the analysis of training experiments conducted in WEKA and Azure Machine Learning Studio setup, Multiclass Decision Forest classifier was chosen for human activity recognition. Classifier had better performance in accuracy when compared to other classifiers.

“The decision forest algorithm is an ensemble learning method for classification. The algorithm works by building multiple decision trees and then voting on the most popular output class. Voting is a form of aggregation, in which each tree in a classification decision forest outputs a non-normalized frequency histogram of labels. The aggregation process sums these histograms and normalizes the result to get the “probabilities” for each label. The trees that have high prediction confidence will have a greater weight in the final decision of the ensemble.

Decision trees in general are non-parametric models, meaning they support data with varied distributions. In each tree, a sequence of simple tests is run for each class, increasing the levels of a tree structure until a leaf node (decision) is reached.

Decision trees have many advantages such as ability to represent non-linear decision boundaries, efficiency in computation and memory usage during training and prediction, ability to perform integrated feature selection and classification, resilience in the presence of noisy features.” [21].

The Decision Forest classifier in Azure Machine Learning Studio consists of an ensemble of decision trees. Generally, ensemble models provide better coverage and accuracy than single decision trees.

3.6 Building Data Collection

Study was restricted to the building Jacaranda Hall in California State University Northridge. Floor plans were provided by physical plant management of the university. To locate a user within the building a small portion of first floor was converted into a graph. Python’s Networkx was used to create and traverse graph.

Chapter 4 - Implementation

This work deals with a 3-part problem, recognizing user activity using smart watch sensors, create activity map and map activities to indoor locations.

To meet the objective of the study an Android application, several services and APIs were developed. Implementation has a service oriented architecture with various components interacting through REST APIs.

4.1 Implementation Components

Main components in implementation setup are companion Android application, database RESTful service developed using Laravel 4.2, predictive web service for human activity recognition deployed in Azure Machine Learning Studio, python-flask RESTful computation web service for indoor navigation & recalibration. Various APIs provided by each of these services are given in Table 4.1.1.

Android application records data for predictive experiment. They consist of wear and mobile modules and application uses REST architecture for sending and receiving data to and from cloud. Another task of Android application is to invoke computation service to locate user within the building. Figure 4.1.1 depicts main activity screen of the application.

Switch button is to control the process of sending data to cloud. When turned on data is sent to MySQL database over HTTP. Android Volley is used for REST calls within the application. Whenever user wants to locate themselves they can do so by clicking “locate me” button. This event triggers the REST call to computation component. Which returns user’s location and Android updates “location information” text to user’s location.

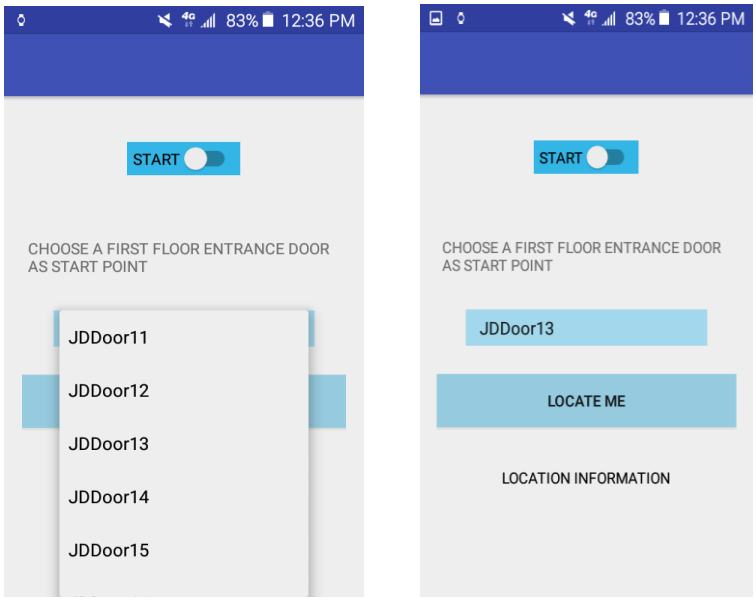

Figure 4.1.1 Android Application Main Activity Screen

Computation service is a python-flask application deployed on Amazon ec2 Ubuntu instance. It has 2 API's, one for computing location information and one for generating building graph. Computation service uses predictive web service to generate activity queue and database service to prune activity queue.

Table 4.1.1 APIs created for implementation

API	Method	Function	Service
http://ec2-54-202-229-14.us-west-2.compute.amazonaws.com/index.php/api/compute/getInput	GET	Reading data	Database
http://ec2-54-202-229-14.us-west-2.compute.amazonaws.com/index.php/api/compute/user_id:{user}/start_time:{start_time}/start_position:{start_position}/x:{val1}/y:{val2}/z:{val3}/time:{timenow}	POST	Writing data	Database

http://ec2-54-202-229-14.us-west-2.compute.amazonaws.com/index.php/api/compute/table:{tablename}	POST	Delete data table	Database
https://ussouthcentral.services.azureml.net/workspaces/9bb9f3b1a0c348279e65c28de877e024/services/8748eaf367004be49c33117534f80857/execute?api-version=2.0&details=ture	POST	Predict User Activity	Human Activity Recognition
http://ec2-54-200-177-132.us-west-2.compute.amazonaws.com/compute/building	POST	Compute location of user	Computation
http://ec2-54-200-177-132.us-west-2.compute.amazonaws.com/compute/activity/{activity-id}	POST	Computing activity-map	Computation

4.2 Training Experiment

Training experiment was done using multiclass decision forest which proved to be the most suitable when tested in Azure machine learning studio. Using column selector of "Select Columns in Dataset" component observed user activity, accelerometer x, y and z readings were selected for training. 10-fold cross validation was used as mode for the training experiment.

Figure 4.2.1 shows the workflow of training experiment on Azure cloud.

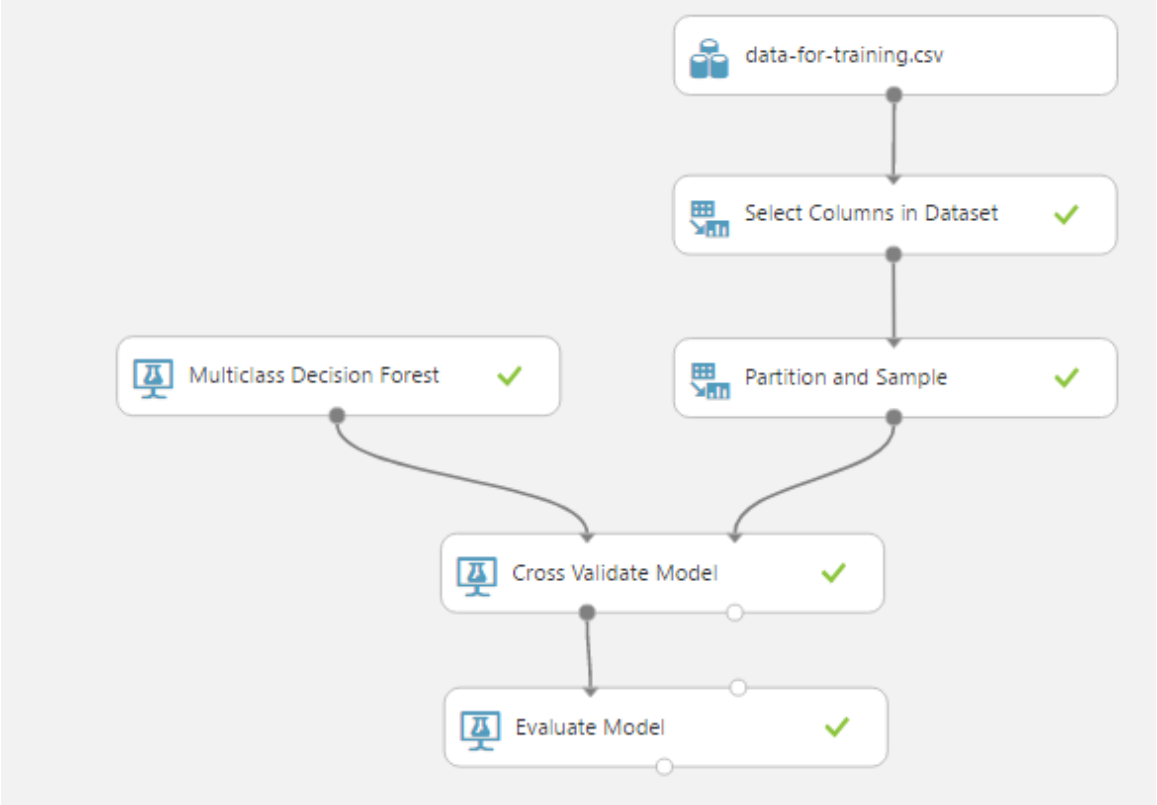

Figure 4.2.1 Training Experiment on Cloud

4.3 Predictive Experiment - Web Service

Predictive experiment intakes accelerometer readings and output user activity. Workflow of the experiment is shown in Figure 4.3.1. A web application (refer Figure 4.3.2), http://humanactivityindoormapping.azurewebsites.net, was deployed on azure portal to test experiment. API details of predictive web service is as described in Table 4.3.1.

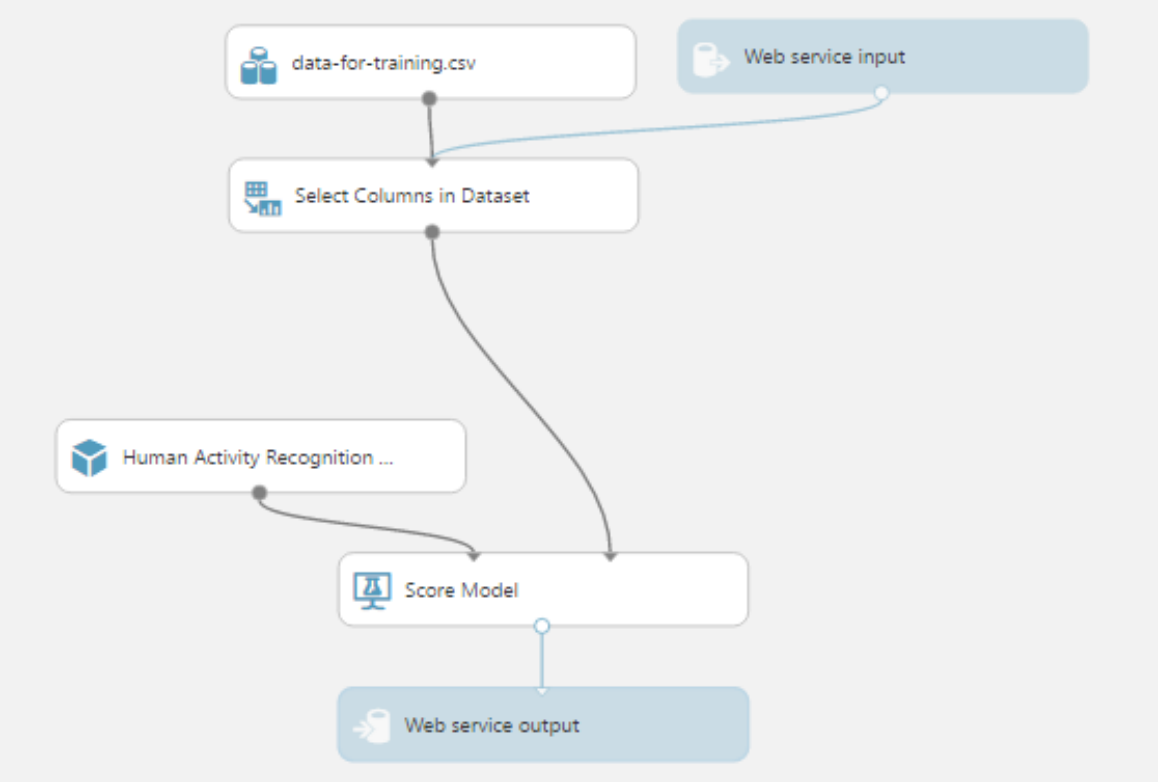

Figure 4.3.1 Predictive Experiment on Cloud

The API takes input parameters as given in Table 4.3.2 and returns output with scored probabilities and scored label as given in Table 4.3.3.

Table 4.3.1 Predictive Web Service Request Response API Details

Request URI	https://ussouthcentral.services.azureml.net/workspaces/9bb9f3b1a0c348279e65c28de877e024/services/8748eaf367004be49c33117534f80857/execute?api-version=2.0&details=ture
Method	POST
HTTP Version	HTTP/1.1

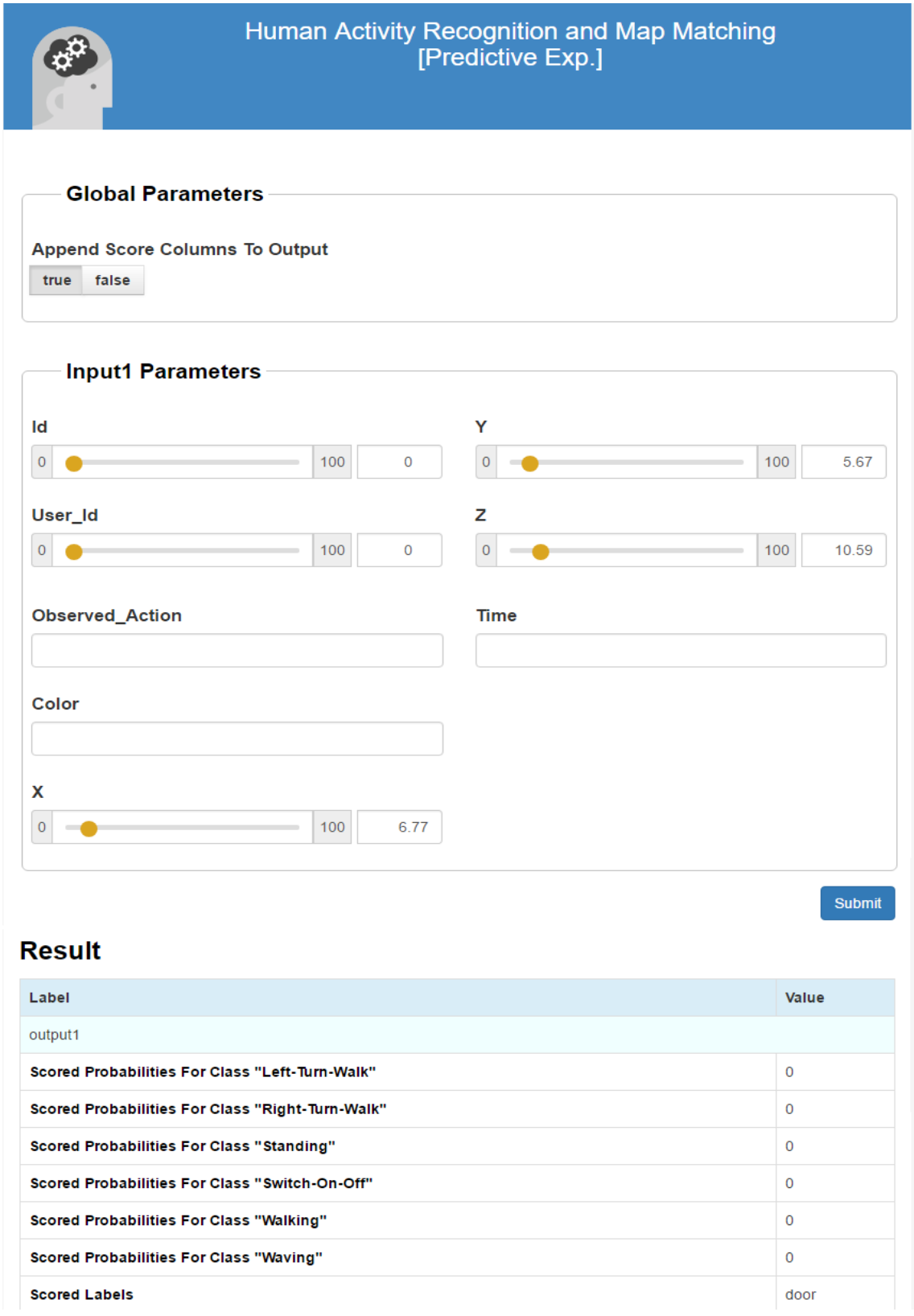

Figure 4.3.2 Web application deployed on Azure to test predictive experiment

Table 4.3.2 Predictive Web Service Input Parameters

Name	Type
id	Numeric
user_id	Numeric
sensor	String
observed_action	String
color	String
x	Numeric
y	Numeric
z	Numeric
date	Object
time	Object

Table 4.3.3 Predictive Web Service Output Parameters

Name	Type	Allowed values
observed_action	String	
x	Numeric	
y	Numeric	
z	Numeric	
Scored Probabilities for Class "cabinet-open-close"	Numeric	
Scored Probabilities for Class "door"	Numeric	
Scored Probabilities for Class "elevator-button-press"	Numeric	
Scored Probabilities for Class "elevator-down"	Numeric	
Scored Probabilities for Class "elevator-up"	Numeric	
Scored Probabilities for Class "fridge-open-close"	Numeric	
Scored Probabilities for Class "left-turn-walk"	Numeric	
Scored Probabilities for Class "right-turn-walk"	Numeric	
Scored Probabilities for Class "standing"	Numeric	

Scored Probabilities for Class "switch-on-off"	Numeric	
Scored Probabilities for Class "walking"	Numeric	
Scored Probabilities for Class "waving"	Numeric	
Scored Labels	Categorical	cabinet-open-close, door, elevator-button-press, elevator-down, elevator-up, fridge-open-close, left-turn-walk, right-turn-walk, standing, switch-on-off, walking, waving

This experiment was further consumed by a RESTful computation web service for providing inputs and confirming activity using recalibration function.

4.4 Computation Algorithms

An algorithm was developed and used for identifying user's location. This algorithm was implemented in the computational cloud service.

4.4.1 Algorithm for Recognizing Location of User within Building

INPUT: building plan as a graph where nodes are doors, start point within building

START

STEP 1: push identified activity to a list.

STEP 2: match identified activity to edge attribute

STEP 3: if matched, calculate displacement from start point to current point.

STEP 4: Predict the last location of user.

STOP

4.5 Implementation Key Concepts

4.5.1 Data for Positioning: Graph

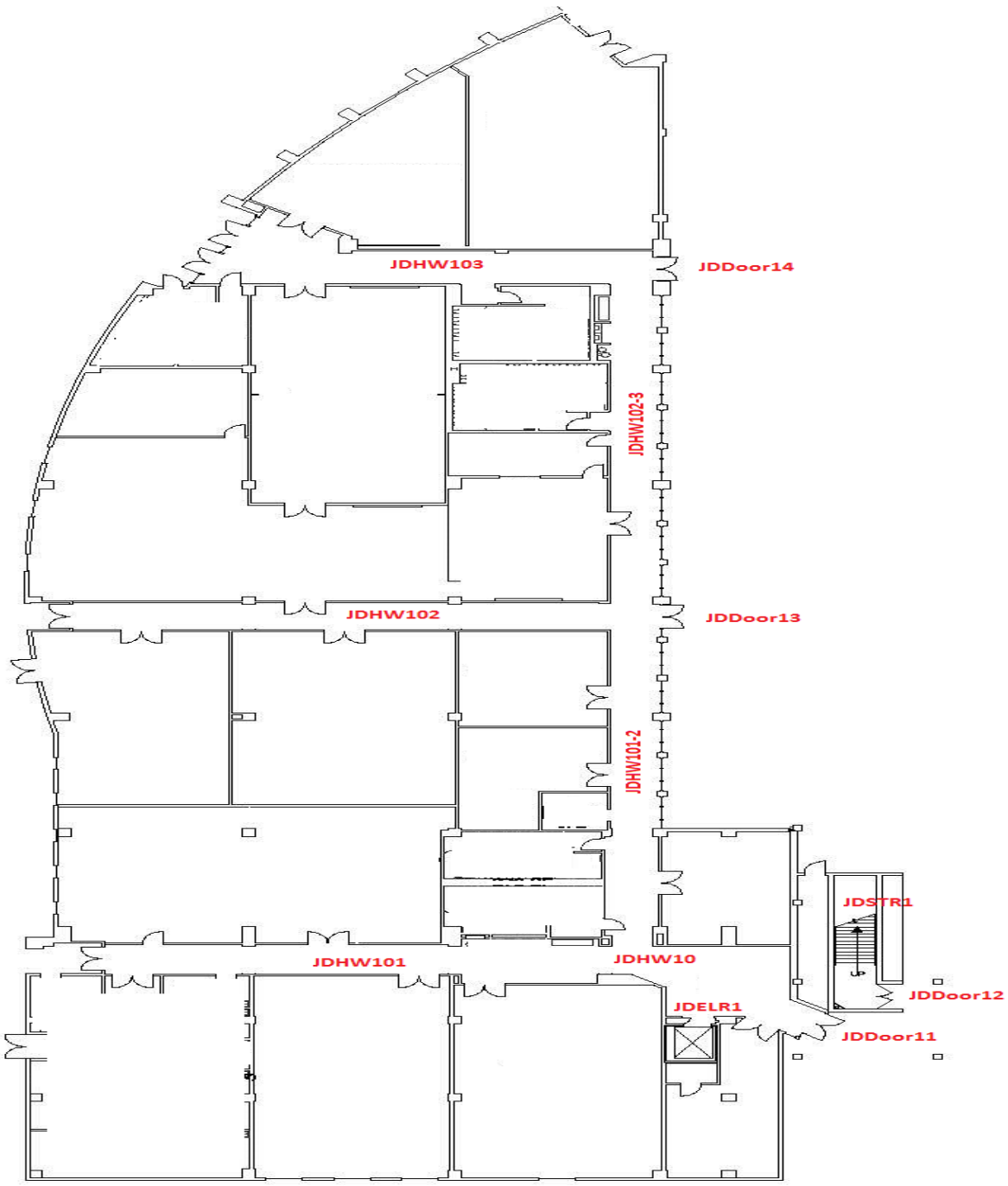

Figure 4.5.1 Graph Node Locations in Map for Indoor Positioning - Activity Mapping Experiments

Using Python's Networkx library a graph of selected portion of the building was generated. Graph nodes constructed for indoor positioning experiments is as depicted in Figure 4.5.1. Using Python's Networkx library a graph of selected portion of the building was generated.

4.5.2 Queuing

Accelerometer readings were queued before sending as input to predictive experiment.

4.5.3 Constraints

The constraints of this study are pre-known start point and prior knowledge about building.

Chapter 5 - Results

5.1 Introduction

Study objective to solve the problem of performing indoor positioning without map matching was successfully implemented. Implementation recognized user activity and matched activity map to building data graph edge attributes. Study revealed that application developed having algorithm to detect user activity to an overall accuracy of 0.521569. Classifier used for activity recognition is Multiclass Decision Forest Classifier.

5.2 Data

Android application records accelerometer readings in the cloud database. Data samples are recorded in the cloud as follows. Each field in all the following data table is explained in Table 5.2.1.

Table 5.2.1 Format of Input Data Table for Indoor Positioning – Activity Mapping Experiments

ID	**Start Time**	**Start Point**	**x**	**y**	**z**	**Time**	**Date**
Data set id in database	Starting Time	Front doors of Jacaranda Hall	Force experienced along X axis	Force experienced along Y axis	Force experienced along Z axis	Instant time of recorded data set	Activity date

Primary features for machine learning predictive web service are acceleration along X, Y and Z axes. Figure 5.2.1 shows the value distribution of x, y and z features. Value signatures over time imprinted by various activities are depicted in Figure 5.2.2, Figure 5.2.3 and Figure 5.2.4. Activity signature of a sample of “door” activity is recorded in Figure 5.2.5.

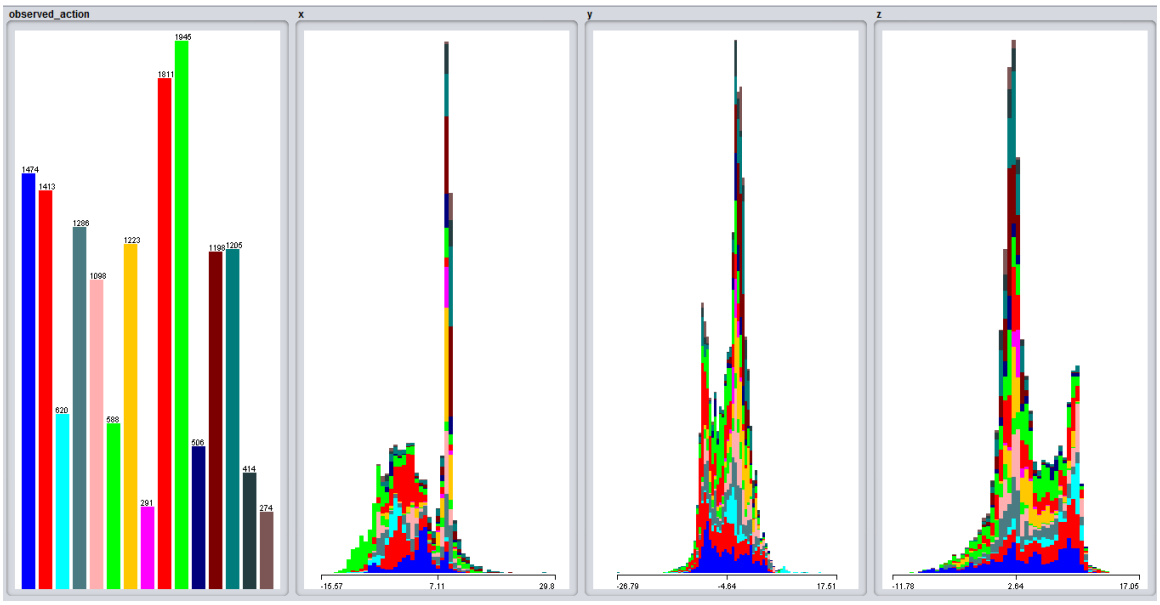

Figure 5.2.1 Accelerometer Readings Value Distribution in X, Y, & Z Axes for Various Human Activities

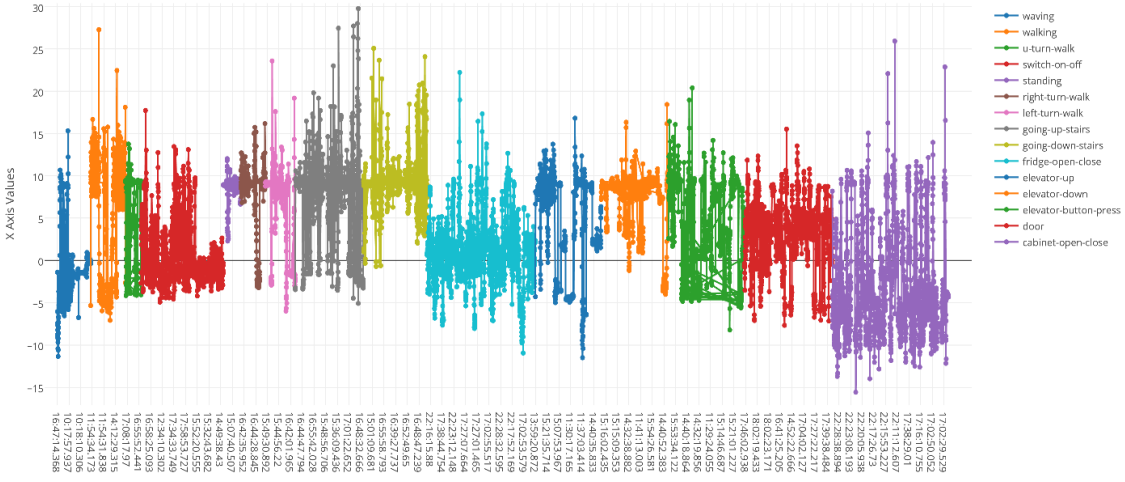

Figure 5.2.2 Force along X axis during various human activities

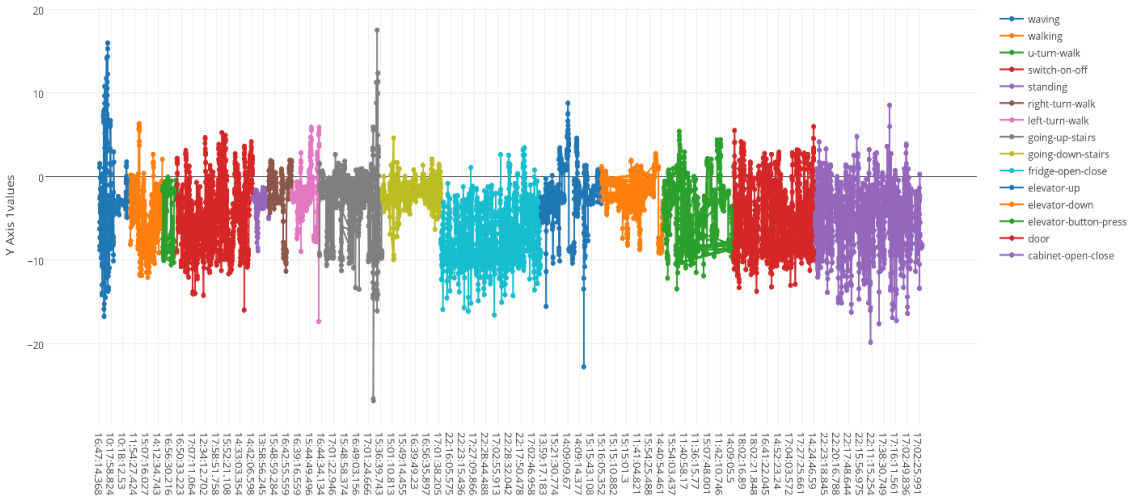

Figure 5.2.3 Force along Y axis during various human activities

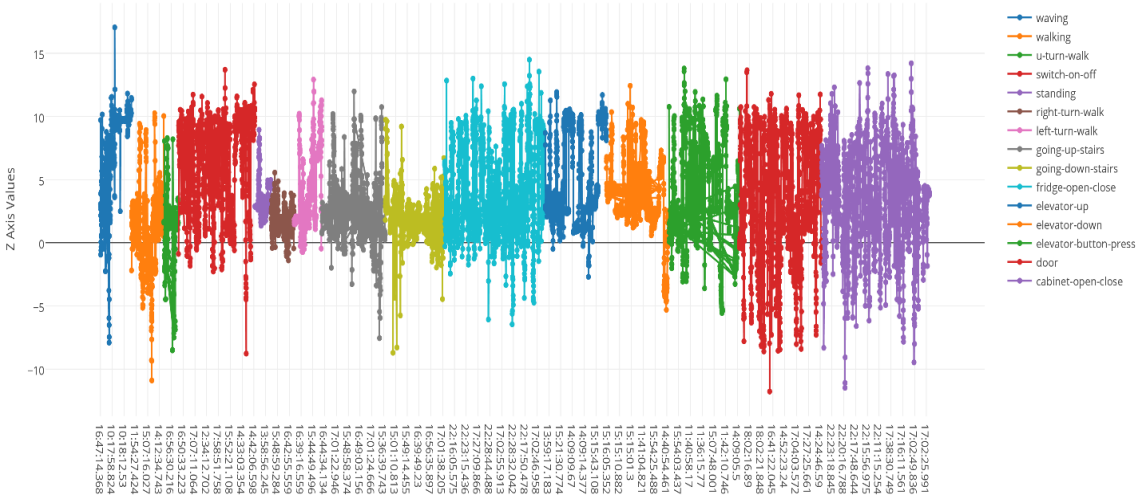

Figure 5.2.4 Force along Z axis during various human activities

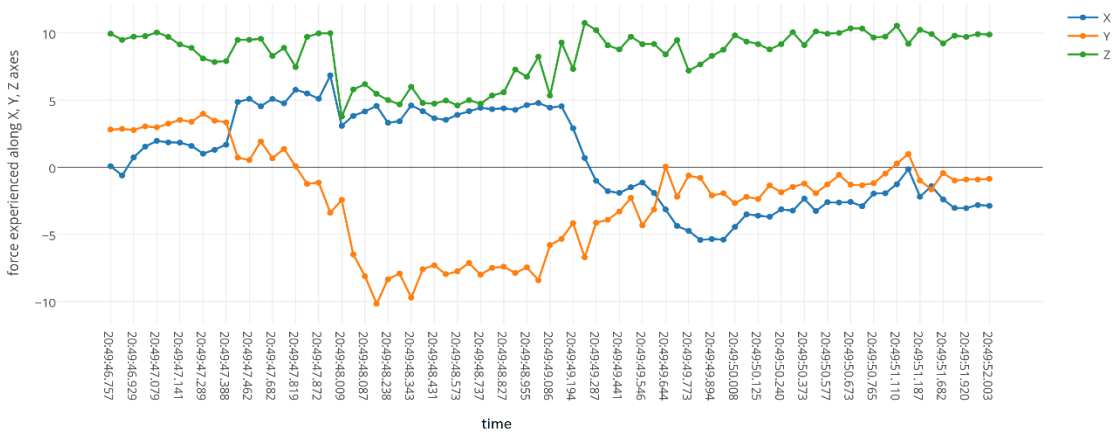

Figure 5.2.5 Force experienced along X, Y, &Z axes for a single opening-door activity

5.3 Computation Results

Main tasks of computation component are to create an activity map using predictive web service, convert indoor map into a traversable graph and perform activity-map matching to output location of the user.

Code for creating activity map is as given below. Parameter map is the formatted response from predictive web service. Each identified activity from predictive web service has a score associated with it. This score corresponds to predictive web service's estimation of probability of the result being true. All predicted activities having a score greater than 0.7 is added to activity map.

```
def recalibrate(map ):
    data = json.loads(map )
    j = 0
    move = dict()
    move = [“door” , “walking” , “elevator-button-press” , “elevator-up” , “elevator-down” , “going-up-stairs” , “going-down-stairs” , “left-turn-walk” , “right-turn-walk” , “u-turn-walk” , “standing”]
    acts = dict()
    for i in data:
        if float(data[i][“score”]) > 0.7 :
            if data[i][“activity”] in move:
                j = j+1
                acts[j] = data[i][“activity”]
 return json.dumps(acts )
```

Code for matching activity-map to building map-graph to output user's location is as given below.

```
def mapping ( G, current ):
    acts = json.loads( <output from activity map service> )
    acts = { int(k): v for k, v in acts.items() }
    activityMap = collections.OrderedDict( sorted ( acts.items() ) )
    activityMap = list( activityMap.values() )
    if current != "JDDoor12":
        activityMap = list(filter(lambda a: a not in [ "going-up-stairs" , "going-down-stairs"], activityMap ) )
 return locate( G, current, activityMap )

def locate ( G, current, activityMap ):
    act = dict()
    action = "none"
    for n in G.neighbors_iter(current):
        if G.has_edge_data( current, n) :
            act = G.get_edge_data( current, n )
            action = act[0]["activity"]
            if action != "none" and activityMap:
                for item in activityMap:
                    if action == item:
                        current = n
                        activityMap.remove( action )
                        if activityMap:
                            locate( G, n, activityMap )
                        return current
```

Method mapping in the computation component takes in location data graph-G and start position-current. It gets the activity map from activity map service from computation service. Mapping further filters out impossible locations by filtering out impossible activities based on start position and invokes method locate.

Method locate takes parameters graph - G, current position – current, and activity map –activityMap and iterates through the neighbouring nodes of current to find a match between activity map and edge attributes. Method is recursively called until either activity map is emptied or graph traversal is complete. Method locate is responsible for returning current position of the user.

5.3.1 Human Activity Recognition

Activities such as walking, standing, making U-turns, right turns, left turns, elevator button press, turning light switches on, elevator-up, elevator-down, going

upstairs, going downstairs, opening door activity, sitting were studied and attempted to be recognized at maximum possible accuracy from accelerometer data alone.

5.3.2 Activity Path Map

Instead of using pedometry based map matching technique, this work proposes activity matching technique. Hypothesis is that, to reach a position within the given building user must perform a minimum set of activities from start point. Hence it is important to identify bare minimum set of activities performed by user to reach each room in the building from a fixed start point. To match this activity path to location data there was a need of representing floor-plan in graph data structure. Graph generated from portion of floor plan used in the study is depicted in Figure 5.3.1. The graph contains 12 vertices (nodes) and 28 edges.

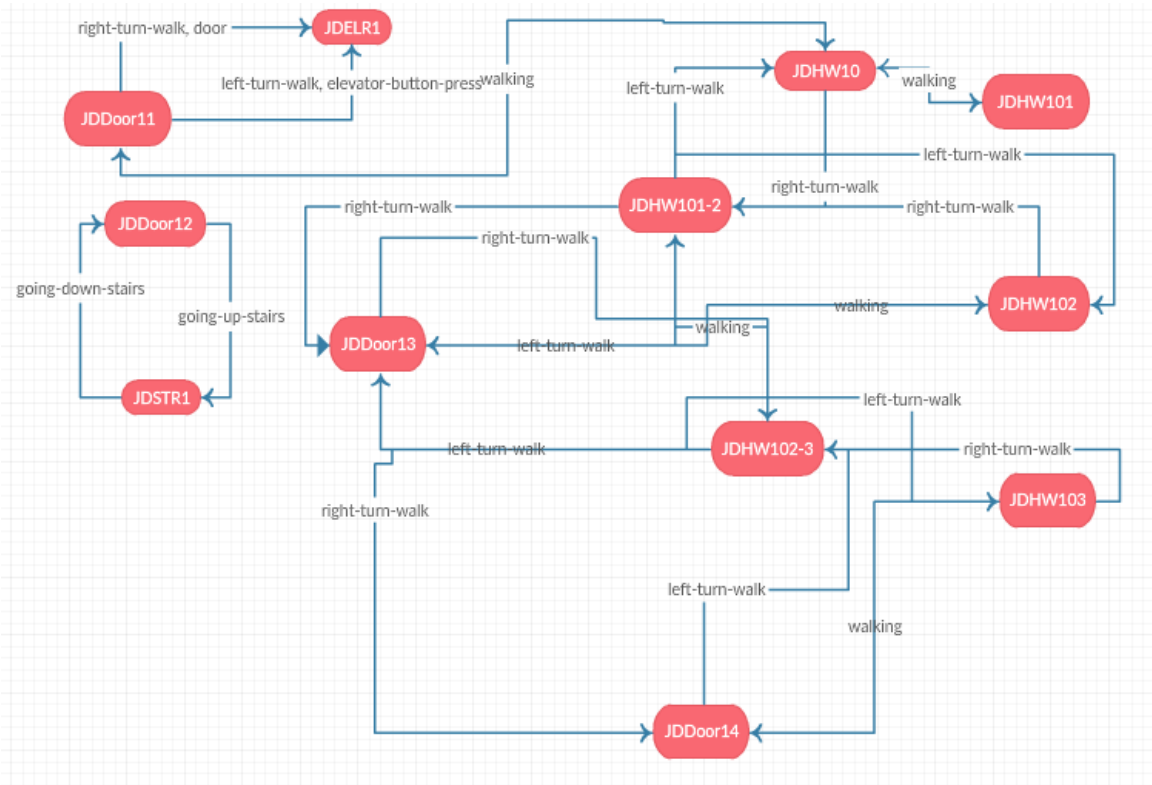

Figure 5.3.1 Graph Generated from Floor Plan for Computing Activity Path

5.3.3 Locate Me

Based on the results from activity matching against activity path map a prediction about user's location within the building is made.

Table 5.3.1 Observations and Results of 22 Indoor Positioning Experiments

Experiment Number	Starting Position	Actual End Position	Predicted End Position	Actual Activity Map	Predicted Activity Map
1	JDDoor11	JDELR1	JDELR1	door, walking, elevator-button-press	door, walking, elevator-button-press, elevator-button-press
2	JDDoor14	JDHW103	None	door, walking, left-turn-walk	U-turn-walk, elevator-down, elevator-button-press, going-up-stairs, going-down-stairs, elevator-up
3	JDDoor11	JDHW101	JDHW101	door, walking	door, walking
4	JDDoor11	JDHW10	JDHW10	door, walking, right-turn-walk, walking, standing	door, walking, right-turn-walk
5	JDDoor13	JDHW102	JDHW102	door, walking	door, walking, right-turn-walk
6	JDDoor12	JDSTR1	JDSTR1	door, right-turn-walk, going-up-stairs, left-turn-walk/U-turn-walk	door, going-up-stairs, walking
7	JDDoor11	JDHW102	JDHW102	door, right-turn-walk, walking, left-turn-walk,	door, right-turn-walk, left-turn-walk

				walking, standing	
8	JDDoor11	JDHW103	JDHW10	door, right-turn-walk, walking, walking(longer), left-turn, walking, standing	door, right-turn-walk
9	JDDoor12	JDFL2	JDFL2	door, right-turn, upstairs, left-turn-walk/U-turn-walk/walking, upstairs, walking, open-door, standing	door, going-up-stairs, U-turn-walking, walking
10	JDDoor12	JDFL3	JDSTR1	door, right-turn-walk, going-up-stairs, left-turn-walk/ U-turn-walk/ walking, going-up-stairs, left-turn-walk/U-turn-walk/walking, going-up-stairs, left-turn-walk/ U-turn-walk/ walking, going-up-stairs, walking, door, standing	door, going-up-stairs, walking
11	JDDoor12	JDFL4	JDSTR1	door, right-turn-walk, going-up-stairs, left-turn-walk/ U-turn-walk/ walking, going-up-stairs, left-turn-walk/ U-turn-walk/	door, going-up-stairs

				walking, going-up-stairs, left-turn-walk/ U-turn-walk/ walking, going-up-stairs, left-turn-walk/ U-turn-walk/ walking, going-up-stairs, left-turn-walk/ U-turn-walk/ walking, going-up-stairs, walking, door, standing	
12	JDDoor12	JDSTR1	JDSTR1	door, right-turn-walk, going-up-stairs	door, right-turn-walk, going-up-stairs
13	JDDoor14	JDHW102	JDHW102	door, left-turn-walk, walking, right-turn-walk, walking, standing	door, left-turn-walk, right-turn-walk, walking
14	JDDoor13	JDHW101	JDHW103	door, left-turn-walk, walking, right-turn-walk, walking, standing	door, right-turn-walk, left-turn-walk, walking
15	JDDoor13	JDELR1	JDELR1	door, left-turn-walk, walking, left-turn-walk, walking, elevator-button-press, standing	door, walking, elevator-button-press
16	JDDoor14	JDHW102	JDHW102	door, left-turn-walk, walking, right-turn-walk, walking,	door, left-turn-walk, walking, right-turn-

				standing	walk
17	JDDoor13	JDHW103	JDHW101	door, right-turn-walk, walking, left-turn-walk, walking, standing	door, walking, left-turn-walk, right-turn-walk, standing
18	JDDoor14	JDHW103	JDHW103	door, walking, standing	door, walking
19	JDDoor13	JDHW10	JDHW10	door, right-turn-walk, walking, standing	door, walking, right-turn-walk
20	JDDoor14	JDHW10	JDHW102	door, left-turn-walk, walking, standing	door, left-turn-walk, walking, right-turn-walk
21	JDDoor13	JDELR1	JDDoor13	door, left-turn-walk, elevator-button-press	door, left-turn-walk, elevator-button-press
22	JDDoor14	JDHW102	JDHW102	door, left-turn-walk, walking, right-turn-walk, walking	door, left-turn-walk, right-turn-walk

5.4 Evaluation of Results

Results from human activity recognition experiment and indoor positioning – activity mapping experiments were evaluated by measuring their overall accuracy and is summarized in Table 5.4.1.

Table 5.4.1 Results

Task	Overall Accuracy	Average Accuracy
Human Activity Recognition (Trained Model)	0.521569	0.936209

Human Activity Recognition - Activity map (22 Experiments)	NA	0.681818
Indoor Positioning (22 Experiments)	NA	0.681818

Confusion matrix of the trained model is given in Figure 5.4.1.

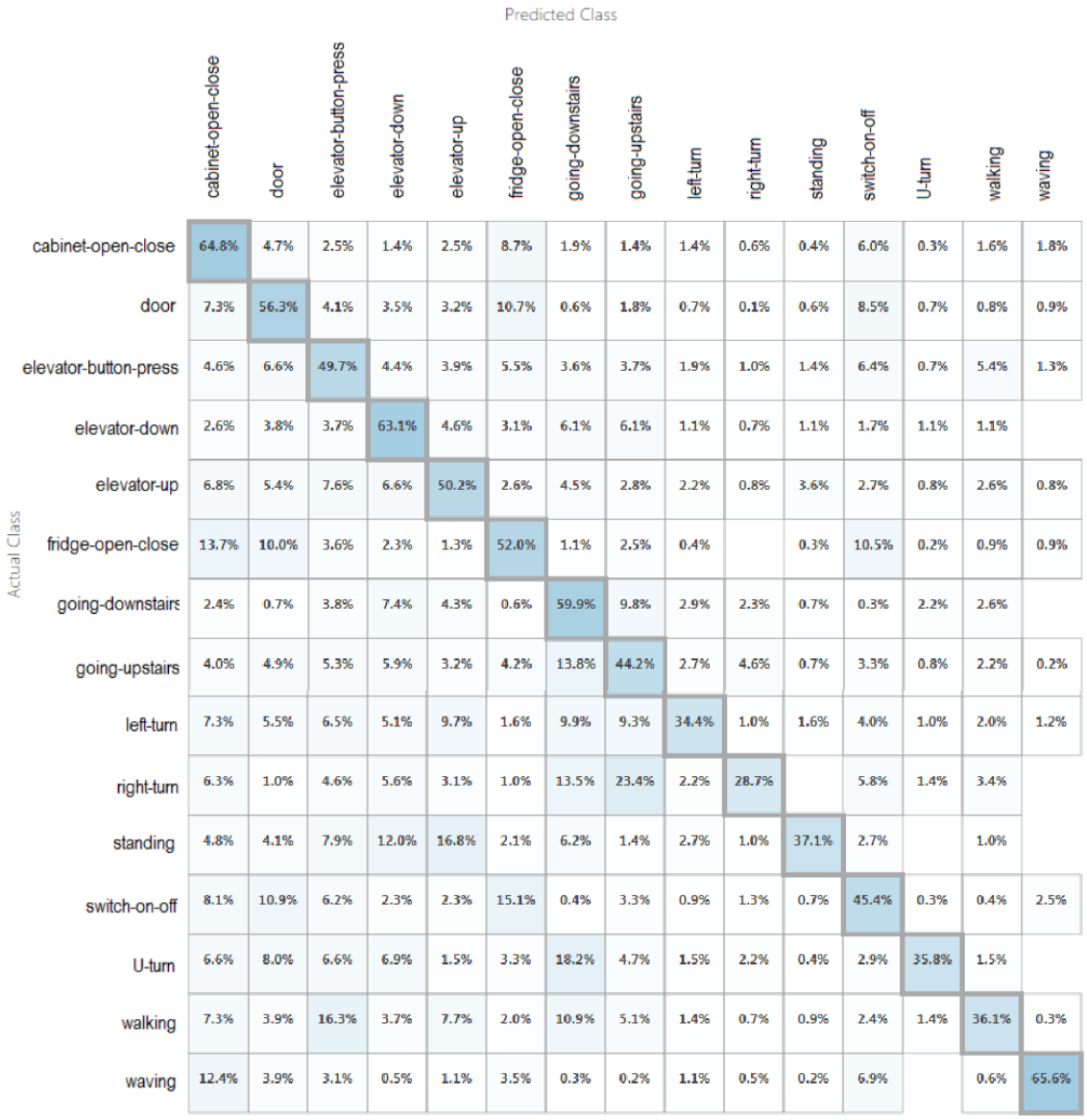

Figure 5.4.1 Confusion Matrix of Multiclass Decision Forest Classifier - Trained Model for Predictive Experiment

In confusion matrix depicted in Figure 5.4.1 certain similar activities were misclassified among each other. For instance, 8.7% of “cabinet-open-close” activity is classified as “fridge-open-close” activity. While 13% of “fridge-open-close” is classified as “cabinet-open-close”. To understand if classifier could classify similar activities several activities were grouped and trained using the classifier. Confusion matrixes from the experiments are depicted in Figures 5.4.2 – 5.4.3. One group consisted of activities “door”, “cabinet-open-close” and “fridge-open-close”. Other group consisted of activities “left-turn-walk”, “right-turn-walk”, “U-turn-walk” and “walking”.

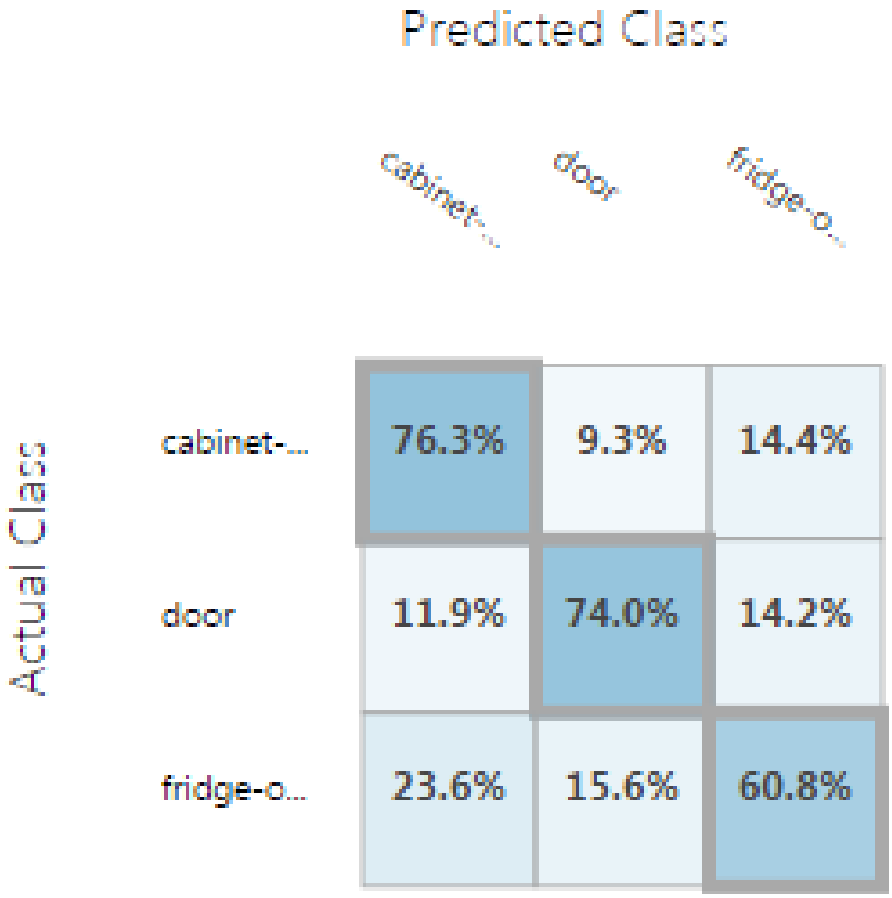

Figure 5.4.2 Confusion Matrix for Activities Cabinet-Open-Close, Door and Fridge-Open-Close

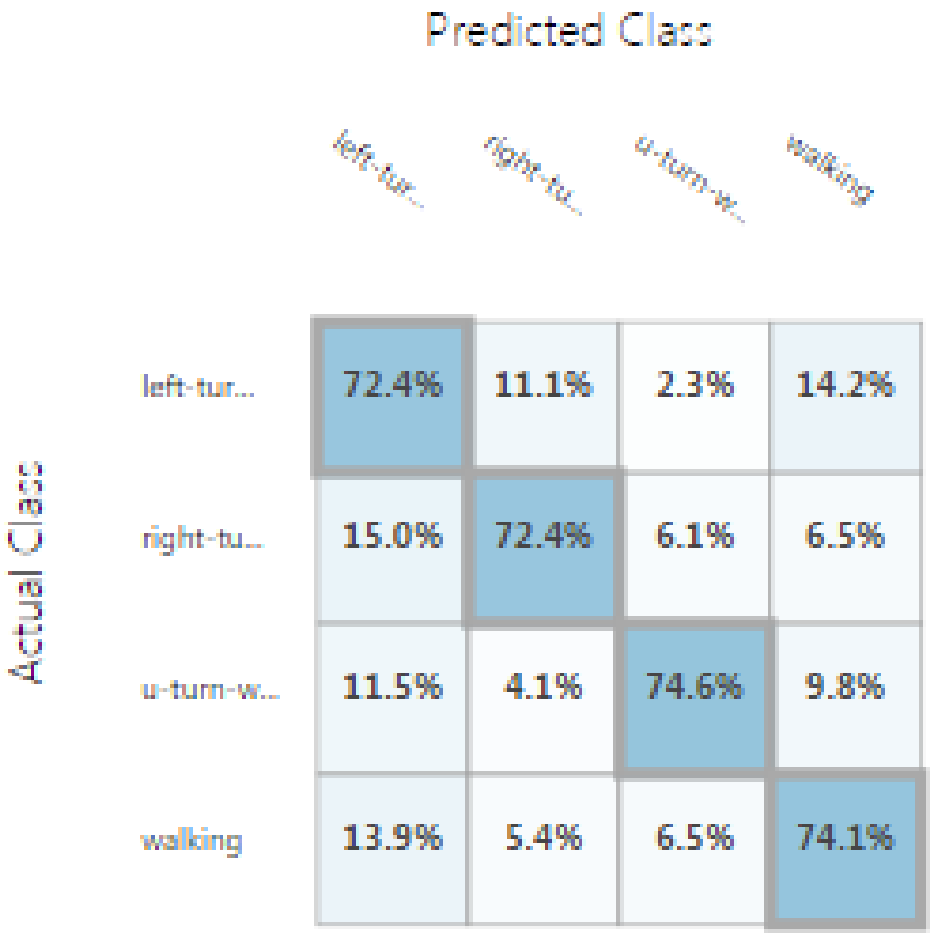

Figure 5.4.3 Confusion Matrix for Activities Walking, U-Turn-Walk, Left-Turn-Walk and Right-Turn-Walk

5.4.1 Limitations

Manually inputting floor plan elements is cumbersome and it limited scaling the graph to the whole building. Study was limited to a single user and a selected building.

5.4.2 Study Outcomes

Study was successful in implementing a solution without using path-to-map-matching techniques. Study objectives and study outcomes are recorded in Table 5.4.2.

Table 5.4.2 Study Objectives and Their Status of Implementation

Number	Objective	Outcome
1	Identify user activity within a time frame	Implemented in a predictive web service using Multiclass Decision Forest Classifier
2	Identify current position of user within a building, based on user activities within the building	Implemented. (Refer Table 5.3.1)
3	Evaluate the accuracy of the above stated identifications	Evaluated. (Refer Table 5.4.1)

Chapter 6 - Conclusion

Solutions from earlier studies on indoor positioning using wearable sensors depends upon the technique of map to user's path matching. Solution attempted by this study heavily depends upon human activity recognition. Criteria for success of this study is the implementation's measure of performance with respect to determining user's position within the pre-selected portion of the building. Keeping criteria for success in consideration the results produced by end user application are recorded in Table 5.3.1 and Table 5.4.1. Table 5.3.1 summarizes the indoor positioning experiments. Table 5.4.1 records the metrics of performance for human activity recognition and indoor positioning implementations.

6.1 Study Outcomes

As mentioned in chapter-1 this study aimed to achieve 3 main objectives - to identify user activity within a time frame, to identify current position of user within a building from user activities within the building and to evaluate the accuracy of the above stated identifications. The study could successfully implement an approach to perform indoor positioning using user activity recognition. Improvements made to human activity recognition accuracy could directly influence accuracy of implementation. Human activity recognition model implemented has an overall accuracy of 0.521569 and 15 out of the 22 indoor positioning experiments gave accurate results.

6.2 Problems Raised by This Study

There is no easy way of representing building or floor data other than images which itself is a problem and topic of interest for many researchers.

References

[1] Lara, O.D. and Labrador, M.A., 2013. A survey on human activity recognition using wearable sensors. IEEE Communications Surveys & Tutorials, 15(3), pp.1192-1209.

[2] Patel, Shyamal, Hyung Park, Paolo Bonato, Leighton Chan, and Mary Rodgers. 2012. A review of wearable sensors and systems with application in rehabilitation. Journal of neuroengineering and rehabilitation.

[3] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan, Evangelos Kalogerakis. 2014. Recognizing Smoking Gestures with Inertial Sensors on a Wristband. Published in MobiSys '14 Proceedings of the 12th annual international conference on Mobile systems, applications, and services

[4] Edison Thomaz, Irfan Essa, and Gregory D Abowd. 2015. A Practical Approach for Recognizing Eating Moments with Wrist-mounted Inertial Sensing. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15). ACM, New York, NY, USA, 1029 -1040.

[5] Mark Marchuk, Christopher Merck, Samantha Kleinberg. 2016. Automated Estimation of Food Type and Amount Consumed from Body-worn Audio and Motion Sensors. published in Ubicomp'16 Germany.

[6] Shen, S., Wang, H. and Choudhury, R.R., 2016. I am a Smartwatch and I can Track my User's Arm. In Proceedings of the 14th annual international conference on Mobile systems, applications, and services.

[7] Xu, Chao, Parth H. Pathak, and Prasant Mohapatra. "Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch." In Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, pp. 9-14. ACM, 2015.

[8] Kumar, Piyush, Jyoti Verma, and Shitala Prasad. "Hand data glove: a wearable real-time device for human-computer interaction." International Journal of Advanced Science and Technology 43 (2012).

[9] Rahman, A. S. M., M. Anwar Hossain, Jorge Parra, and Abdulmotaleb El Saddik. "Motion-path based gesture interaction with smart home services." In Proceedings of the 17th ACM international conference on Multimedia, pp. 761-764. ACM, 2009.

[10] Heumer, Guido, Heni Ben Amor, Matthias Weber, and Bernhard Jung. "Grasp recognition with uncalibrated data gloves-a comparison of classification methods." In 2007 IEEE Virtual Reality Conference, pp. 19-26. IEEE, 2007.

[11] Junker, Holger, Oliver Amft, Paul Lukowicz, and Gerhard Tröster. "Gesture spotting with body-worn inertial sensors to detect user activities." Pattern Recognition 41, no. 6 (2008): 2010-2024.

[12] Nguyen-Dinh, L.V., Calatroni, A. and Tröster, G., 2014, September. Towards a unified system for multimodal activity spotting: challenges and a proposal. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (pp. 807-816). ACM.

[13] Ranjan, Juhi, and Kamin Whitehouse. "Object hallmarks: Identifying object users using wearable wrist sensors." Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing. ACM, 2015.

[14] Pirttikangas, S., Fujinami, K. and Nakajima, T., 2006, October. Feature selection and activity recognition from wearable sensors. In International Symposium on Ubiquitious Computing Systems (pp. 516-527). Springer Berlin Heidelberg.

[15] Fujinami, K., Pirttikangas, S. and Nakajima, T., 2006. Who opened the door?: Towards the implicit user identification for sentient artefacts. Na.

[16] Golding, Andrew R., and Neal Lesh. "Indoor navigation using a diverse set of cheap, wearable sensors." In Wearable Computers, 1999. Digest of Papers. The Third International Symposium on, pp. 29-36. IEEE, 1999.

[17] Bao, Haitao, and Wai-Choong Wong. "A novel map-based dead-reckoning algorithm for indoor localization." Journal of Sensor and Actuator Networks 3, no. 1 (2014): 44-63.

[18] Robertson, Patrick, Michael Angermann, and Bernhard Krach. "Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors." In Proceedings of the 11th international conference on Ubiquitous computing, pp. 93-96. ACM, 2009.

[19] Ascher, Christian, Christoph Kessler, Rüdiger Weis, and Gert F. Trommer. "Multi-floor map matching in indoor environments for mobile platforms." In Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on, pp. 1-8. IEEE, 2012.

[20] Elhoushi, Mostafa, Jacques Georgy, Ahmed Wahdan, Michael Korenberg, and Aboelmagd Noureldin. "Using portable device sensors to recognize height changing modes of motion." In 2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, pp. 477-481. IEEE, 2014.

[21] Microsoft. Multiclass Decision Forest. Retrieved May 8, 2017 from https://msdn.microsoft.com/en-us/library/azure/dn906015.aspx, 2016.