ACTIVITY RECOGNITION USING K-NEAREST NEIGHBOR ALGORITHM ON SMARTPHONE WITH TRI-AXIAL ACCELEROMETER

Sahak Kaghyan, Hakob Sarukhanyan

Abstract*: Mobile devices are becoming increasingly sophisticated. These devices are inherently sensors for collection and communication of textual and voice signals. In a broader sense, the latest generation of smart cell phones incorporates many diverse and powerful sensors such as GPS (Global Positioning Systems) sensors, vision sensors (i.e., cameras), audio sensors (i.e., microphones), light sensors, temperature sensors, direction sensors (i.e., magnetic compasses), and acceleration sensors (i.e., accelerometers). The availability of these sensors in mass-marketed communication devices creates exciting new opportunities for data mining and data mining applications. So, it is not surprising that modern mobile devices, particularly cell phones of last generations that work on different mobile operating systems, got equipped with quite sensitive sensors. This paper is devoted to one approach that solves human activity classification problem with help of a mobile device carried by user. Current method is based on K-Nearest Neighbor algorithm (K-NN). Using the magnitude of the accelerometer data and K-NN algorithm we could identify general activities performed by user.*

Keywords: *human activity classification; K-NN algorithm; mobile devices; accelerometer; Android platform*

Introduction

The data to recognize human's activity is from the physical hardware sensors, and the combination of the accelerometer, the compass sensors and GPS are the most commonly used sensor devices. This project's objective is to explore how effective is in general the K-NN algorithm and, in future works, its modifications in user activity classification problem solving. For our current research we have taken the base algorithm without any serious modifications, although in our future works we shall use different combinations of this one and other methods such as, for example, decision trees. In order to know the accuracy of this algorithm we also created two applications. One of them is a smartphone application that works on the Android platform and is able to get and store data concerning user's physical activity using incoming signals from tri-axial accelerometer that comes with mobile device that he or she cares. It stores raw data on security disk card or just SD card of given device. After the data saved, it will be transferred on server for further work. Second application is a desktop application that does the rest – activity classification using K-nearest neighbor algorithm. It analyzes incoming data in order to classify transferred activity.

In sections below we shall briefly describe what Android operating system is and what an accelerometer is and it works on a mobile device. Then we shall give a description of K-NN algorithm itself. Finally, our approach to recognize activity from accelerometer data using this algorithm will be introduced and them it will be followed by results.

Android operating system

First step of activity classification problem, discussed in this article, is to create an application that will be able to retrieve acceleration values from smartphone. So, it will be logical to start from choosing a platform that will be used. There are several platforms for mobile phones. Most popular of them are Android, IOS and Windows

Mobile platforms. And here we used Android as a target platform for our experiments. Thus, here we shall focus on explaining how this operating system is organized and what advantages it has. Whether you're an experienced mobile engineer, a desktop or web developer, or a complete programming novice, Android represents an exciting new opportunity to write innovative applications for mobile devices. So, the main question, from which we shall start, will be the following one:

What is Android?

Google describes Android as: The first truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.

Generally, Android is a combination of three components:

- ✓ A free, open-source operating system for mobile devices.

An open-source development platform for creating mobile applications.

Devices, particularly mobile phones, that run the Android operating system and the applications created for it.

More specifically, Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK (Software Development Kit) provides the tools and APIs (Application Programming Interfaces) necessary to begin developing applications on the Android platform using the Java programming language [Meier, 2010].

So we decided to use the Android-based cell phones as the platform for our experiments because the Android operating system is free, open-source, relatively easy to program, and according to October of 2011 statistical data, it claims quite impressive position among other mobile operating system manufacturers. Our project currently tested on the following type of Android phone: HTC Desire HD. Data that was collected from accelerometer of this cell phone was stored on cell phone's SD card, although we expect to change this way of data storing because modern cell phones have all necessary interfaces and means to send data directly to server, for example via wireless networks, such as Wi-Fi, or via the Internet. However, data in this work was transferred to server via a USB (Universal Serial Bus) connection, but will make it at least less common or it just will be fully replaced by wireless data transfer mode in our future works. We also expect to modify software application so that later it will be able to do the activity recognition process right on user's cell phone.

At the same time Android platform is widely used not only because of it is free or open-source, but also because of the mechanisms, designed to protect the privacy and security of Android users, as well as the operating system. These methods include the Android security architecture, application certificates and application permissions. The purpose of the Android security architecture is to prevent applications from being able to automatically perform operations that could jeopardize the security of other applications, the operating system or the user. Certificates are used to identify the author of a specific application and to prevent users from installing fraudulent software on their devices. Android will not install an application that has not been signed with a certificate. Therefore, the origin of all published applications is traceable. Android security permissions are handled by the *AndroidManifest.xml* file present within all application files. When a user downloads an application onto their device, they are automatically notified of the permissions the application has access to. This informs the user of what type of information an application is able to collect from the device as well as the hardware the application can use. The *AndroidManifest.xml* file takes care of both software and hardware permissions. But while Android does require permissions for the use of hardware devices such as the camera and vibrator, it does not require permissions to be set in place for the use of any available sensors, including the accelerometer, orientation, and gyroscope sensors. Therefore, alongside with other tools, such as the internet and GPS, can also pose as security threat to the user. And it is possible for an application to collect user information from these sensors without the user's knowledge. Android's application-neutral APIs provide low-level access to the

increasingly diverse hardware commonly available on mobile devices. The ability to monitor and control these hardware features provides a great incentive for application development on this platform.

Accelerometers

The data to recognize human's activity is from the physical hardware sensors, and the combination of the accelerometer, the compass sensors and GPS are the most commonly used sensor devices. The accelerometer is another component that is becoming a standard item in new devices. Several types of accelerometers, the most currently used are based on electro-mechanical devices (micro electro-mechanical systems or MEMS) that include a series of needle-like structures that detect motion, generating the readings are then transmitted to the main circuit.

A tri-axial accelerometer is a sensor that returns a real valued estimate of acceleration along the x, y and z axes from which velocity and displacement can also be estimated. Accelerometers can be used as motion detectors. It measures proper acceleration, which has an experience relative to gravity and is the acceleration felt by people and objects. Accelerometers have been proposed by previous studies as a tool to monitor and assess physical activities of subjects in a free-living environment. The acceleration signals recorded through accelerometers have been used to classify daily living activities. There are extensive researches on using accelerometers to classify activities such as walking, running, falling, sitting, cycling, etc. [Lee, 2010], [Nishkam, et al.,2005], [Fomby, 2008], [Kwapisz et al., 2010]. This sensor can be used to detect movement and the rate of change of the speed of movement. One of benefits of Android platform is that the using of accelerometers in Android applications does not require the application to have permission to use it. So, it is possible to collect accelerometer data from user without his or her knowledge. This will make process of application executing easier to user, because he will not be prompted to give agreement to use built-in accelerometer each time the program runs. The most obvious application for the accelerometer is to change the screen orientation when we rotate the device (Figure 1). The big question is almost all devices use screens with vertical orientation, but activities such as surfing the web or watch videos require a screen with horizontal orientation. Normally you would need to activate an option to change the orientation, but with an accelerometer that can be done automatically. It's something simple, but ends up having a great effect with respect to usability. But there are also other situations when this sensor is also used.

Figure 1: Smartphone changes display of screen from vertical to horizontal when user rotates it.

The accelerometer can also be used to shortcut functions, such as changing the track on MP3 player, answer or reject a call, open or close applications and so on. A good shake-up band, two put the phone in silent mode and so on. Two other important areas are the games and the applications of GPS. In the case of games, accelerometer lets you deploy controls in the style of the Nintendo Wii [Nintendo Official Site], which opens a whole new range of possibilities. For GPS applications, the accelerometer enables the software to detect ripped brakes, cornering and so on. Variables, which can then be used to make software more intelligent, detecting when you missed a turn, or keeping a rough estimate of the location when it loses the satellite signal for a few seconds, for example.

These are improvements that alone does not say much, but together they end up making a big difference, more than enough to pay the small cost increase resulting from the additional component [Hall et al., 2008].

Human activity classification

Human physical activity recognition has been receiving increasing attention in recent years. Human behavior and its classification are significant for the disciplines such as medicine, behavioral sciences, physiotherapy, etc. An accelerometer is an inexpensive, effective and feasible body-worn sensor which has been frequently used in daily physical activity classification. Use of accelerometer in patient's mobile device will help to know whether what kind of moves he does and do that without disturbing him.

Many research groups have studied activity recognition as part of context awareness research [Parkka et al., 2006]. Context sensing and use of context information is an important part of the ubiquitous computing scenario. Context sensing aims at giving a computing device (e.g., cellular phone, wrist-top computer, or a device integrated into clothes) senses, with which it becomes aware of its surroundings. With the senses the device is capable of measuring its user and environment and it becomes context aware. The context describes the situation or status of the user or device. Different devices can use the context information in different ways, e.g., for adapting its user interface, for offering relevant services and information, for annotating digital diary (e.g., energy expenditure), etc. Location and time belong to the group of the most important contexts and the use of these contexts has been studied extensively. However, to recognize the physical activities of a person, a sensor-based approach is needed.

Activity recognition is formulated as a classification problem. In this study we consider following activities performed by user: standing; walking; running; sitting; climbing up stairs; climbing down stairs. The reason of selecting these activities was quite simple. They were selected because they are performed regularly by many people in their everyday life. These activities also involve motions that repeat in time and this in ideal way the data that comes from accelerometer will be periodic and will we think that it can make the recognition process easier. When we record the data from for each of these activities, we record acceleration in three axes. Process of our human activity classification project can be represented by diagram below (Figure 2).

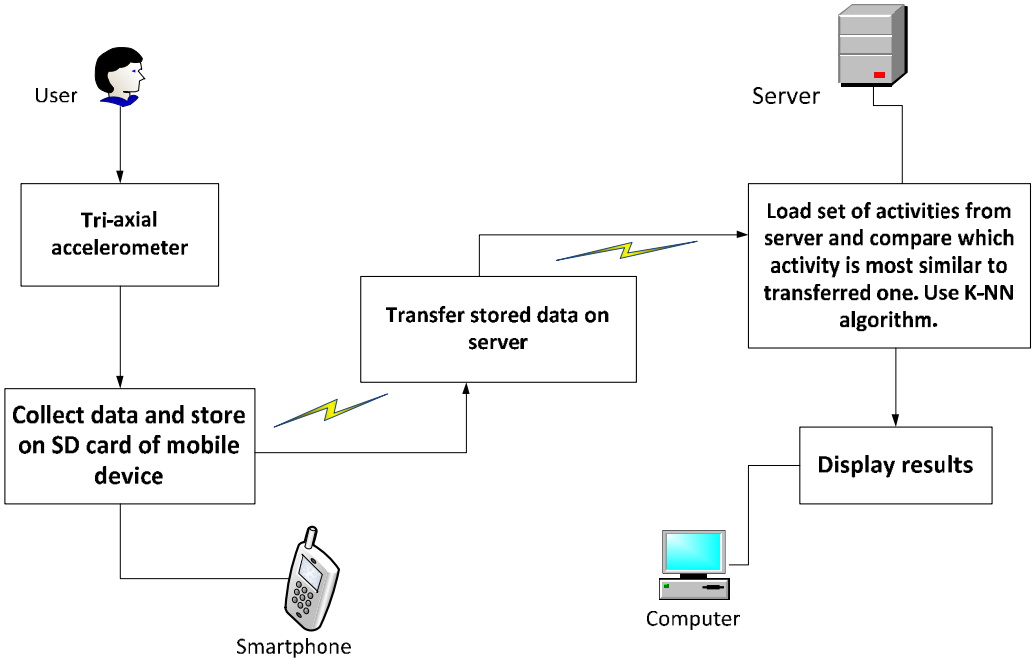

Figure 2: Process of collecting and storing data from smartphone, data transfer and analyzing (logic flow)

First step is to collect data from smartphone user carries in his pocket and store it in memory of mobile device. Second step is to transfer data on server and save it in proper way. Third step is the classification process itself

where unknown activity template will be compared in a loop with every activity template from predefined training set. After calculating distances between each type of activity and target activity, algorithm will display as result that activity which will have minimal distance from the unknown one. Thus, the accuracy of results mostly depends on the templates that were chosen for selected activities.

K-Nearest Neighbor

K-Nearest Neighbor is a supervised learning algorithm where the result of new instance query is classified based on majority of K-Nearest Neighbor category. It is one of the most popular algorithms for pattern recognition. The purpose of this algorithm is to classify a new object based on attributes and training samples. The classifiers do not use any model to fit and only based on memory. K-Nearest Neighbor algorithm used neighborhood classification as the prediction value of the new query instance. Many researchers have found that the K-NN algorithm accomplishes very good performance in their experiments on different data sets. The traditional K-NN text classification algorithm has three limitations: (a) calculation complexity due to the usage of all the training samples for classification, (b) the performance is solely dependent on the training set, and (c) there is no weight difference between samples. The best choice of *k* depends upon the data; generally, larger values of *k* reduce the effect of noise on the classification, but make boundaries between classes less distinct. A good *k* can be selected by various heuristic techniques, for example, cross-validation. The special case where the class is predicted to be the class of the closest training sample (i.e. when *k*=1) is called the nearest neighbor algorithm. In pattern recognition field, K-NN is one of the most important non-parameter algorithms and it is a supervised learning algorithm. The classification rules are generated by the training samples themselves without any additional data. The K-NN classification algorithm predicts the test sample's category according to the *k* training samples which are the nearest neighbors to the test sample, and judge it to that category which has the largest category probability.

The accuracy of the K-NN algorithm can be severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. Much research effort has been put into selecting or scaling features to improve classification. A particularly popular approach is the use of evolutionary algorithms to optimize feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes. In binary (two class) classification problems, it is helpful to choose k to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal k in this setting is via bootstrap method [Suguna et al., 2010].

In similar activity recognition works this algorithm came in combination with other helping methods. For example in [Das et al., 2010] there were also used decision tables and decision trees. They were able to increase the accuracy after the device was properly calibrated for given user.

In pattern recognition, the k-nearest neighbor algorithm (K-NN) is a method for classifying objects based on closest training examples in the feature space. K-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The K-Nearest Neighbor algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its *k* nearest neighbors (*k* is a positive integer, typically small). If *k*=1, then the object is simply assigned to the class of its nearest neighbor. Figure 3 illustrates situation when k is taken 5. Unknown point will be compared with 5 closest neighbors and depending on results will be marked as belonging to proper set. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most

frequent among the *k* training samples nearest to that query point. Usually Euclidean distance is used as the distance metric; however this is only applicable to continuous variables. In cases such as text classification, another metric such as the overlap metric or Hamming distance [Math32031, 2007], for example, can be used.

A drawback to the basic "majority voting" classification is that the classes with the more frequent examples tend to dominate the prediction of the new vector, as they tend to come up in the *k* nearest neighbors when the neighbors are computed due to their large number.

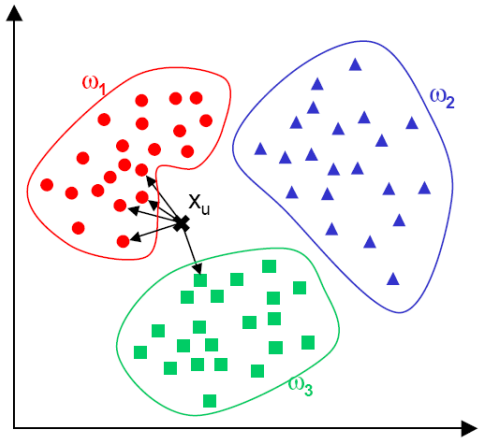

Figure 3: X_u is point unknown template. In this example k=5. Euclidean distance between this point and its 5 closest neighbors is calculated. 4 of them belong to ω_1 and 1 belongs to ω_3, so X_u assigned to ω_1 set.

One way to overcome this problem is to weight the classification taking into account the distance from the test point to each of its *k* nearest neighbors.

There are several ways to calculate the distance between two points in multidimensional space. Suppose we have two points x, y where each point is an n-dimensional vector, i.e. $x = \{x_1, x_2, \ldots, x_n\}$, $y = \{y_1, y_2, \ldots, y_n\}$.

Distance measuring functions can be taken the following ways. We can define distance function $d_E(x, y)$ between two points by measuring their distance according to Euclidean formula or $d_A(x, y)$ distance function that measures absolute distance between them using formulas below:

$$d_E(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2},\ d_A(x, y) = \sum_{i=1}^{n} |x_i - y_i|.$$

Our approach uses Euclidean formula to calculate distance between any given 3-dimensional points.

However, this method it has limitations such as: great calculation complexity, fully dependent on training set, and no weight difference between each class. This once again confirms that one of important means of improving the accuracy of result data is a good choice of training set. In the prediction problem the K-NN “tuning” parameter is the neighborhood size *k*. In the binary classification problem, the K-NN model requires the tuning of two parameters, the neighborhood size and, for each neighborhood size, the cutoff probability for choice. In each of these cases the tuning parameter(s) is (are) chosen to optimize the scoring performance in the K-NN model in the validation data set. Finally, scoring an optimal K-NN model on the test data set provides the opportunity to obtain unconditional performance measures of the optimal K-NN model.

Data Collection, Analyzing and Results

Data from the accelerometer has the following attributes: time, acceleration along x axis, acceleration along y axis and acceleration along z axis.

There are cases, when program must be preliminarily calibrated for given user in order to make the accuracy more effective like it was done in [Das et al., 2010].

We used K-NN algorithm to analyze raw data that we got from mobile device. We used sequential comparing of our activity pattern with training sets, each time calculating the distances between incoming acceleration data points and points of our training sets.

Standard classification algorithms cannot be directly applied to raw time-series accelerometer data. Instead, we first must transform the raw time series data into examples. To accomplish this we divided the data into 10-second segments and then generated features that were based on the 200 readings contained within each 10-second segment. We refer to the duration of each segment as the example duration (ED). We chose a 10-second ED because we thought that it provided sufficient time to capture several repetitions of the (repetitive) motions involved in some of activities. Although we did not perform experiments to determine the optimal example duration value, but we shall explore that in our future works and will find out how a longer period of ED will effect on accuracy.

After user runs the mobile application, he will be able to save incoming data from device into text file where each line contains date and time of accelerations along each axis and the acceleration values themselves.

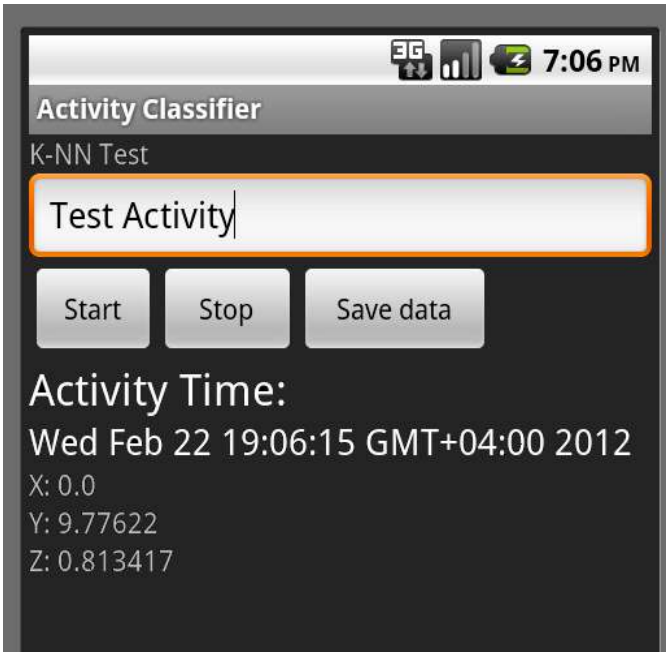

Figure 4: User interface to collect and store data from smartphone sensor

Here is part of code that is responsible for retrieving sensor values from smartphone:

```
1.  // check sensor type
2.  if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER){
3.      // assign directions
4.      float x=event.values[0];
5.      float y=event.values[1];
6.      float z=event.values[2];
7.      ...
8.      Calendar now = Calendar.getInstance();
9.      ...
10.     s1.add(String.valueOf(x));
11.     s1.add(String.valueOf(y));
12.     s1.add(String.valueOf(z));
13.     currentDT.add(now.getTime());
14. }
15. ...
```

As it was mentioned before, modern mobile devices, especially smartphones, can be equipped with various sensors. So, when we program function that will listen to changes that will occur when sensor data changes, first we must be sure that these changes come from accelerometer and not from any other sensor. Thus, we preliminary must enclose all commands of proper acceleration values retrieving in a block that checks whether incoming data is from accelerometer or other sensor.

The user interface and acceleration values of our Android based acceleration acquiring application is shown on following figures (Figures 5-7):

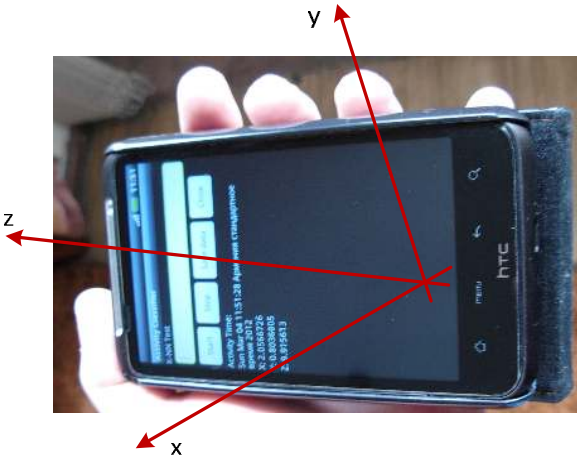

Figure 5: Acceleration along z-axis.
x: 2.056672, y: 0.8036005, z: 9.91561≈ 1g

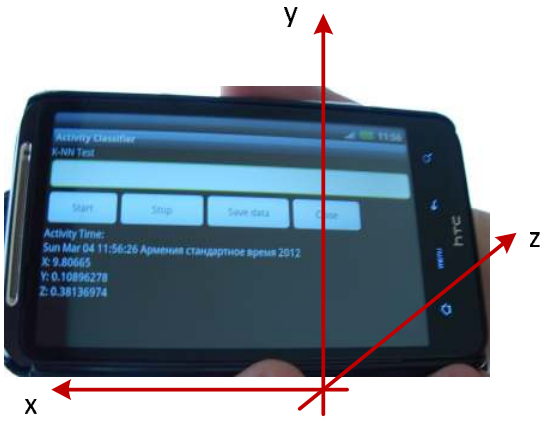

Figure 6: Acceleration along x-axis.
x: 9.80665 ≈ 1g, y: 0.10896278, z: 0.38136974

Figure 7: Smartphone is in vertical position.
Acceleration along y-axis.
x: 0.6403563, y: 9.806434 ≈ 1g, z: 1.6334297

Three different positions of this smartphone are shown on this pictures: first one illustrates the situation when smartphone is laying in user's hand (figure 5), or it can just lay on table; figure 6 displays horizontal position of device; finally, last picture shows acceleration values when device stands vertically.

As we can see from these pictures when smartphone is in position shown on figure 5, acceleration along z-axis approaches to 9.8 m/s^2, i.e. to 1g. The same behavior is noticed when mobile device is in position (horizontal or vertical) in which acceleration along proper axis also approximates to 1g.

Also, when device is in horizontal position the display of smartphone automatically turns on 90 degrees.

When inner timer of application stops, data retrieving step overs and the next one, i.e. collected data saving step starts. After data is saved in mobile phone's memory, it will be manually transferred via USB cable on server. Then, after it is on server, the algorithm implementation does the program that runs on server. Modern personal computers have several cores and in order to decrease the time that application spends on calculation process, we used multithreading. To compare incoming data with data representing each activity (each training set) we gave a single thread to it. Although for these experiments k was taken equal to 1 and training sets had less than

thousand points representing each activity, so it gave us only a little bit of efficiency, but if training set will increase and k will increase, this computing method will be very helpful.

Application finds out which template is closer to the current activity vector by measuring distance of each point of our target pattern with all points of each template.

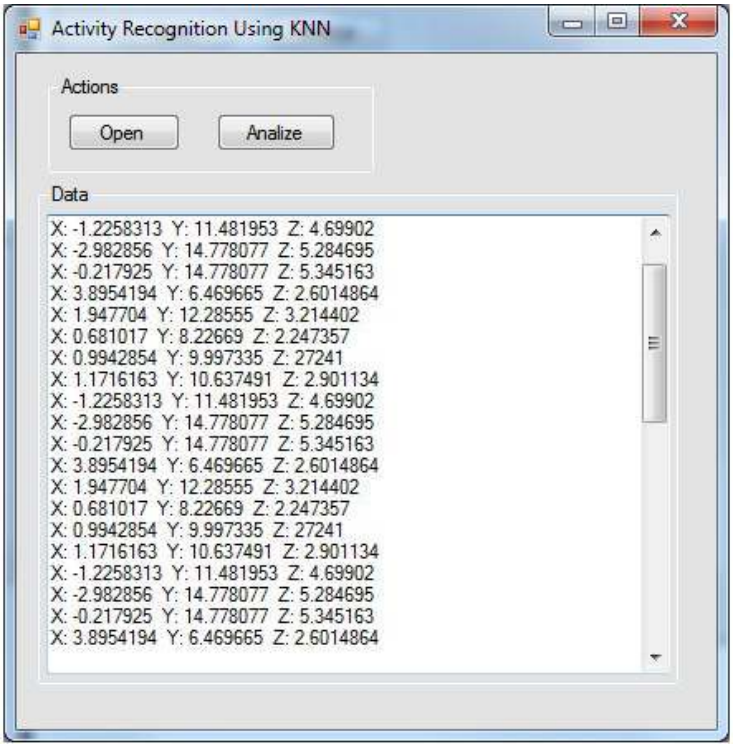

Figure 4: Windows desktop application. User selects data and program does the classification using K-Nearest Neighbor algorithm.

The algorithm that does the classification can be given as follows:

1. TS = { Set of templates describing each activity; Each template is represented as a 3-dimentional array} ;
2. TT = { Target template };
3. MD = { Minimal distance between element of training set and target template };
4. MD = Calculate_Distance(TS[0], TT);
5. **for each** ts $\in$ TS \ TS[0]
    a. LD = { Local value of distance between element of training set and target element };

LD = Calculate_Distance(ts, TT);

    b. **if** (MD $\geq$ LD)
        i. MD = LD;
        ii. Remember_Activity_Class();
    c. **else continue;**

Output = Get_Proper_Activity_Class_Name();

Here *Calculate_Distance(), Remember_Activity_Class()* and *Get_Proper_Activity_Class_Name()* are functions that help to do distance calculation, given activity saving operation and final results display respectively.

It is not hard to notice that every template is a multidimensional vector and, thus, in real program there will be one *for* loop nested inside another, so the complexity of this method is $O(n \cdot p^2)$, where p is the length of vector representing single element from training set and n is total number of activity vectors. Software application, implementing this algorithm and doing the classification process, was written on C# programming language [Mackey, 2010], [Freeman, 2010] and used multithreading concept in order to increase speed of calculation.

As it was mentioned above, k was taken equal to 1. When k increases, calculation process becomes more complicated. Despite of that is not so big problem for modern personal computers (because data can be processed on personal computers with multiple cores or even clusters), but processing time increases anyway. This can be a serious problem for data analysis on mobile phone where battery power is limited and power saving problem always plays one of major roles.

For *sitting* and *standing* activities method gives 100% accuracy. For all other activities accuracy can be increased by increasing training set of each template.

Conclusion and future work

The accelerometer proved that it is a useful tool in identifying activities based on user's phone's movements and that the activities can be recognized with fairly high accuracy using a single tri-axial accelerometer. Despite that, activities that are limited to the movement of just hands or mouth are comparatively harder to recognize using a single accelerometer worn near the pelvic region.

We expect in our future work to increase the accuracy of collected data analyzing with K-NN algorithm by studying its modifications. Also using the means of Android platform programming we can more effectively collect data from sensor by choosing other delay mode. At the same time we shall study other methods that help to classify user activity (for example fast Fourier transformations or Hidden Markov Models).

It is also possible to combine data that will be retrieved from accelerometer with the data that will come from GPS sensor. As a result, this combination can make activity recognition process more efficient.

Bibliography

[Lee, 2010] Jungoo Lee, Mobile phone based training application for kayaking, http://cs.anu.edu.au/student/projects/10S2/Reports/Jungoo%20Lee.pdf

[Meier, 2010] Meier, Rito. Professional AndroidTM 2 Application Development, Wiley Publishing, Inc., 2010

[Царьков] Сергей Царьков. Алгоритм ближайшего соседа. http://www.basegroup.ru/library/analysis/regression/knn/

[Das, Green, Perez, Perring, 2010] Sauvik Das, LaToya Green, Beatrice Perez, Michael Murphy, Adrian Perring, Detecting user activities using the accelerometer on Android smartphones, 2010

[Nishkam, Dandekar, Mysore, Littman, 2005] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, Michael L. Littman, Activity recognition from accelerometer data, 2005

[Fomby, 2008] T. Fomby, K-Nearest Neighbors Algorithm: Prediction and Classification, 2008

[Kwapisz, Weiss, Moore, 2010] J.R. Kwapisz, G.M. Weiss, Samuel A.Moore, Activity Recognition using Cell Phone Accelerometers, 2010

[Suguna, Thanushkodi, 2010] N. Sugunal, and Dr. K. Thanushkodi, An Improved k-Nearest Neighbor Classification Using Genetic Algorithm, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 2, 2010

[Tenkomo, 2006] K. Tenkomo. K-Nearest Neighbor Tutorial. 2006. http://people.revoledu.com/kardi/tutorial/KNN/index.html

[Mackey, 2010] Alex Mackey, Introducing .NET 4.0, 2010

[Freeman, 2010] Adam Freeman, Pro .NET 4 Parallel Programming in C#, 2010

[Ancillotti] Ancillotti. Smartphones Accelerometers. http://ancillotti.hubpages.com/hub/SmartphonesAccelerometers

[Hall, Park, 2008] P. Hall; B. U. Park; R. J. Samworth (2008). Choice of neighbor order in nearest-neighbor classification. Annals of Statistics 36: 2135–2152. doi: 10.1214/07-AOS537.

[Parkka, Ermes, Korpipaa] Juha Parkka, Miikka Ermes, Panu Korpipaa. Activity Classification Using Realistic Data from Wearable Sensors. IEEE Transactions on information technology in biomedicine, vol. 10, No. 1, 2006.

[Math32031, 2007] Coding Theory. Part 2 – Hamming Distance, 2007, http://www.maths.manchester.ac.uk/~pas/code/notes/part2.pdf

[Nintendo Official Site] http://www.nintendo.com/wii

Authors' Information

Sahak Kaghyan *– PHD student, Department of System Programming, Russian-Armenian (Slavonic) University, e-mail:* sahak.kaghyan@gmail.com

Major Fields of Scientific Research: Digital Signal and Image Processing, Data Mining, Object Oriented Systems, Soft Computing

Hakob Sarukhanyan *– Head of Digital Signal and Image Processing Laboratory, Institute for Informatics and Automation Problems of NAS of RA; e-mail: hakop@ipia.sci.am*

Major Fields of Scientific Research: Digital Signal and Image Processing, Fast Orthogonal Transforms, Parallel Computing