Использование веб-интерфейсов для анализа перемещений мобильного устройства

И. А. Петров, Д. Е. Намиот

***Аннотация* — В данной работе рассматриваются алгоритмы использования веб-интерфейсов для анализа перемещений мобильных устройств, а также разработан и реализован алгоритм такого анализа перемещения с интерфейсом для пользователя. Задача состоит в создании некоторой универсальной библиотеки на языке JavaScript, которая может быть использована в мобильных веб-приложениях. Библиотека использовала веб-интерфейсы для доступа к данным о перемещении устройства для анализа перемещения мобильного устройства. Предложен и реализован алгоритм определения перемещений, основанный на сравнении подобия временных рядов. Проведен анализ свойств предложенного алгоритма. На данном этапе алгоритм может с высокой долей вероятности детектировать движение телефона для заранее заданных наборов движений. Для реализации алгоритма использовались последние стандарты языков веб-программирования и технологии разработки. Были проведены эксперименты на реальном мобильном устройстве и сделаны выводы по реализации построенного алгоритма. Использование веб-интерфейса позволит применять разработанный алгоритм как в браузерных приложениях на мобильных телефонах, так и в мобильных приложениях, которые используют веб-технологии в своём исходном коде. Разработанный алгоритм не зависит от платформы конкретного мобильного устройства, так как используемые веб-интерфейсы поддерживаются в большинстве существующих мобильных веб-платформ.**

***Ключевые слова* — WEB API, DTW-алгоритм, акселерометр.**

I. Введение

В современном мире большинство людей на нашей планете используют интернет для поиска информации, пользования услугами, игр и многого другого.

Для удобного использования приложений, просмотра веб-страниц и других использований возможностей интернета существуют специальные веб-дизайнерские и программные инструменты, которые помогают веб-разработчикам создавать продукты для пользования ими обычными людьми. Главным из таких инструментов для веб-программиста являются языки программирования и разметки.

Самым популярным языком программирования на сегодняшний день является JavaScript – интерпретируемый мультипарадигменный язык веб-программирования, поддерживающий объектно-ориентированный, императивный и функциональный стили программирования. JavaScript часто используется как встраиваемый язык для программного доступа к объектам приложения.

Для работы с мобильными устройствами разработчику на JavaScript необходим API (Application Programming Interface), который позволяет программисту «общаться» с мобильным устройством и использовать такое «общение» в своём программном продукте. Стандартизацией Web API занимается специальная рабочая группа W3C Web API организации, называемой Консорциум Всемирной паутины (W3C, World Wide Web Consortium) [8]. На странице [9] собран список различных JavaScript Web API.

Для анализа перемещений мобильного устройства непосредственно в мобильном устройстве находится специальный прибор, который называется акселерометр. Данный прибор позволяет получать значения ускорения телефона по трём координатным направлениям (Рисунок 1) относительно центра телефона, хотя более верно будет сказать - относительно места расположения акселерометра в мобильном устройстве.

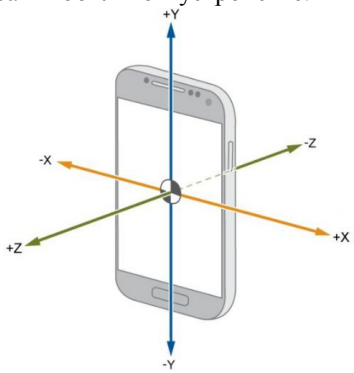

Рис. 1: Система координат акселерометра

При помощи Web API разработчик может брать показания этого прибора и использовать их внутри своего алгоритма анализа движения мобильного устройства либо для других целей.

При использовании акселерометра стоит учитывать, что в реальных условиях возможно наличие шумов в его показаниях. Чтобы избежать таких «артефактов» необходимо использовать фильтры.

Анализ движение мобильного устройства может

Статья получена 23 апреля 2020.
И. А. Петров – МГУ имени М. В. Ломоносова (e-mail: petrov.igor.a@gmail.com)
Д. Е. Намиот – МГУ имени М. В. Ломоносова (e-mail: dnamiot@gmail.com)

использоваться во многих областях веб-разработки, такие как игры, робототехника, приложения для отслеживания показателей здоровья человека и т. д. Можно рассматривать это, например, как некоторую специальную форму ввода данных, когда “данные” определяются махом руки с телефоном. Поэтому разработка такого алгоритма станет полезным инструментом для веб-специалистов в создании продуктов для мобильных приложений.

Использование веб-интерфейса позволит применять разработанный алгоритм как в браузерных приложениях на телефонах, так и в мобильных приложениях, которые используют веб-технологии в своём исходном коде. То есть, такой алгоритм будет являться многофункциональным. К тому же, данный алгоритм не будет зависеть от платформы конкретного мобильного устройства, так как используемые веб-интерфейсы поддерживаются в большинстве существующих платформ.

Для анализа результата анализа перемещений мобильного устройства проводились измерения при помощи Web API. Пример измерений на JavaScript ниже на Рисунке 2.

```
if (current_time <= start_time + MEASUREMENT_TIME) {
    // Low pass filter
    out_x = out_x + alpha * (event.acceleration.x - out_x);
    out_y = out_y + alpha * (event.acceleration.y - out_y);
    out_z = out_z + alpha * (event.acceleration.x - out_z);

    user_motion_str += out_x + " " + out_y + " " + out_z + " ";
}
```

Рис. 2: Измерения показаний акселерометра на JS

После анализа пользователь будет иметь строковое представление (наименование) движения из заранее заданного набора (Рисунок 3).

```
const motions = [ "up", "down", "left", "right" ];
```

Рис. 3: Набор движений для анализа в JS-файле

Вообще говоря, этот набор не обязательно делать заранее «вшитым» в программу. Можно задавать её непосредственно перед началом анализа. Такая модель была выбрана для конкретных экспериментов.

II. Модель системы

A. *Временные ряды показаний акселерометра*

Как нам уже известно, у программиста с помощью Web API есть возможность получать показания акселерометра по трём координатам в координатной плоскости, связанной с устройством, в котором, собственно, и расположен акселерометр. Получение этих значений происходит в течение времени, то есть, запоминая значения показаний прибора, мы получаем последовательность этих показаний, тем самым, образуя *временной ряд показаний акселерометра*.

По определению **временным рядом** называются последовательно измеренные через некоторые (не всегда равные) промежутки времени данные [1]. В нашем случае в качестве данных выступают показания акселерометра по трём координатам, то есть вектора $\bar{a} = (a_x, a_y, a_z)$, где

a_x – значение ускорения по координате x,
a_y – значение ускорения по координате y,
a_z – значение ускорения по координате z.

Отметим, что значения a_x, a_y и a_z являются скалярными величинами, равными длине вектора-компоненты ускорения по соответствующей координатной прямой.

Таким образом, получаем последовательность векторов $A = \{\bar{a_1}, \bar{a_2}, \dots, \bar{a_n}\}$, где число n – количество замеров показаний акселерометра. Эти замеры могут проводиться либо заданное число раз, либо в течение заданного промежутка времени, в зависимости от выбранного алгоритма. Также можно применять комбинации этих стратегий.

Два временных ряда показаний акселерометра $A = \{\bar{a_1}, \bar{a_2}, \dots, \bar{a_m}\}$ и $B = \{\bar{b_1}, \bar{b_2}, \dots, \bar{b_n}\}$ назовём *равными*, если $m = n$ (то есть количество векторов-элементов в них одинаково) и $\forall j = \overline{1 \dots n} : \bar{a_j} = \bar{b_j}$.

В дальнейшем понятие временного ряда будет основным при формулировках и выкладках.

B. *Общая модель*

Чтобы детектировать движения мобильных устройств, надо определиться, что именно мы хотим получить на выходе работы алгоритма. Наша цель определить движение устройства, то есть сказать пользователю, какое движение совершило его устройство. Пользователь в данном случае есть некоторое приложение на мобильном устройстве, которое будет использовать результаты определения в своих алгоритмах.

Для данной цели необходимо определить заранее заданный набор движений, которые может детектировать наш алгоритм.

Ранее было определено понятие временного ряда показаний акселерометра. Такой ряд характеризует определённое движение устройства в пространстве. То есть для определения перемещения устройства необходимо иметь некоторое множество его *эталонных движений*, которые известны до начала работы нашего основного алгоритма, в виде набора временных рядов показаний акселерометра.

После получения временного ряда от пользователя, нам будет необходимо сравнить данный ряд с имеющимися эталонными рядами и выбрать в соответствии с некоторой метрикой наиболее близкий из них, выдав его в качестве ответа.

Отметим, что длины входящего временного ряда пользователя и эталонных временных рядов, вообще говоря, зависят от некоторого заранее заданного времени замера t и внешних факторов в системе телефона (ведь нам заранее не известно какие физические и программные процессы повлияют на частоту замеров показаний), которые мы будем считать случайной величиной ξ, зависящей от времени начала замера последовательности.

Таким образом, основная задача состоит **в поиске**

наиболее близкого временного ряда из заранее заданного множества к входящему временному ряду.

C. Математическая модель

Пусть имеется непустое множество $\boldsymbol{E} = \{E_1, \ldots, E_s\}$ эталонных временных рядов показаний акселерометра:

$$E_1 = \{\overline{e_1^1}, \overline{e_2^1}, \ldots \overline{e_{h_1}^1}\},$$
$$E_2 = \{\overline{e_1^2}, \overline{e_2^2}, \ldots \overline{e_{h_2}^2}\},$$
$$\ldots$$
$$E_s = \{\overline{e_1^s}, \overline{e_2^s}, \ldots \overline{e_{h_s}^s}\},$$

где все вектора $\overline{e_j^i} = \left(e_x^{ij}, e_y^{ij}, e_z^{ij}\right) \in \mathbb{R}^3 \; \forall i = \overline{1 \ldots s}, \; j = \overline{1 \ldots h_i}$ – показания акселерометра в определённый момент времени.

Пусть также в соответствии с движением пользователя на вход подаётся упорядоченная последовательность показаний акселерометра (временной ряд) $A = \{\overline{a_1}, \overline{a_2}, \ldots, \overline{a_m}\}$, где $\overline{a_k} = (a_x^k, a_y^k, a_z^k) \in \mathbb{R}^3 \; \forall k = \overline{1 \ldots m}$.

Числа $m, h_1, \ldots, h_s \in \mathbb{N}$ являются некоторой функцией, зависящей от времени замера и от внешних факторов системы:

$m = f(t, \xi(t_A))$, t_A-начало замера последовательности A,

$h_1 = f(t, \xi(t_{E_1}))$,

...

$h_s = f(t, \xi(t_{E_s}))$

где $f(t, \xi(t_0)) \in \mathbb{N}$ – некоторая функция, зависящая от времени замера $t \in \mathbb{R}_+$ последовательности и случайной величины, характеризующей внешние факторы системы и зависящей от времени начала измерений $t_0 \in \mathbb{R}_+$ последовательности.

Такую функцию f будем называть *функцией длины временного ряда показаний акселерометра*.

Необходимо найти такое значение p, при котором расстояние между двумя временными рядами A и $E_p \in \boldsymbol{E}$ является минимальным, то есть $p = arg \min_{i: E_i \in \boldsymbol{E}} D(A, E_i)$, где $\forall X, Y$ – временных рядов показаний акселерометра:

1) $D(X, Y) \geq 0$, $D(X, Y) = 0 \Leftrightarrow X = Y$,

2) $D(X, Y) = D(Y, X)$

Функцию D назовём *расстоянием между временными рядами*.

III. Некоторые методы

Итак, нам необходимо из непустого множества временных последовательностей $\boldsymbol{E} = \{E_1, E_2, \ldots, E_s\}$ выбрать такое $p \in \{1, \ldots, s\}$, что функция расстояния между входящим временным рядом и рядом E_p является минимальным по множеству $\boldsymbol{E}$.

A. Последовательности одинаковой длины

Можно описать самый простой способ получения функции расстояния для двух временных последовательностей, если длины сравниваемых последовательностей являются равными, то есть $m = h_1 = h_2 = \ldots = h_s$. Для этого необходимо, чтобы функция длины временного ряда показаний акселерометра была фиксированной величиной, то есть $f(t, \xi(t_0)) = c = const, \forall (t, t_0) \in \mathbb{R}_+^2$. Константу c можно выбрать несколькими вариантами:

1) c – заранее заданное натуральное число. В этом случае нам не важно время, в течение которого происходил замер, нам лишь важно количество таких замеров;

2) $c = \min_{\mathbf{E}_i \in \mathbf{E}} |\mathbf{E}_i|$, так как множество эталонных движений известно заранее, то можно выбрать минимальную из всех длин временных рядов данного множества. Однако стоит учитывать, что в этом случае мы утрачиваем информацию о других последовательностей.

Таким образом, имея последовательности одинаковой длины, достаточно просто задать расстояние между этими последовательностями в виде суммы расстояний между соответствующими элементами данных временных рядов:

$$\forall X = \overline{x_1}, \overline{x_2}, \ldots, \overline{x_m}, Y = \overline{y_1}, \overline{y_2}, \ldots, \overline{y_m}:$$
$$D(X, Y) = \sum_{i=1}^{m} d(\overline{x_i}, \overline{y_i}),$$

где d – некоторая метрика для двух трёхмерных векторов. Везде далее мы будем использовать евклидову метрику и в качестве символа d использовать квадрат её значения:

$$\forall \bar{a} = (a_x, a_y, a_z) \in \mathbb{R}^3, \bar{b} = (b_x, b_y, b_z) \in \mathbb{R}^3:$$
$$d(\bar{a}, \bar{b}) = \sum_{r \in x,y,z} (a_r - b_r)^2.$$

При таком задании последовательностей легко выписать общую формулу для решения поставленной задачи:

$$p = arg \min_{i: E_i \in \boldsymbol{E}} \sum_{j=1}^{m} \sum_{r \in x,y,z} \left(a_r^j - e_r^{ij}\right)^2$$

Основные преимущества данного метода заключаются в следующем:

1) Данный метод лёгок для понимания
2) Легко запрограммировать
3) Не требует дополнительной памяти
4) Определение расстояние между двумя временными последовательностями имеет линейную сложность.

Однако у данного метода есть ряд критических недостатков, которые нельзя не учитывать:

1) При выборе константной функции длины временного ряда показаний акселерометра никак не учитывается влияние внешних факторов на систему, что является важной частью замеров
2) При выборе константной функции длины временного ряда показаний акселерометра время, в течение которого будут производиться замеры, может сильно отличаться (с учётом п. 1))
3) Такой метод имеет низкий процент точных определений движений

Итак, данный метод является простым в понимании и непосредственном его программировании. Однако использование такого метода приводит к неприемлемым

результатам работы. Далее мы не будем строго опираться на длину временных рядов показаний акселерометра.

B. DTW-алгоритм

DTW-алгоритм (Dynamic Time Warping Algorithm), или алгоритм динамической трансформации временной шкалы [2], [3], [4] – это алгоритм, позволяющий найти оптимальное соответствие между двумя временными последовательностями. Данный алгоритм нередко используется в распознавании голоса [5], [6] и текста [7], поскольку в этих областях также приходится иметь дело с временными рядами.

В алгоритме рассматриваются два временных ряда:

$$A = a_1, a_2, \dots, a_m,$$
$$B = b_1, b_2, \dots, b_n.$$

Отметим, что в общем случае $m \neq n$.

Для элементов данных рядов задано расстояние d между ними в пространстве, соответственно, данных элементов.

Двум исходным рядам ставится в соответствие матрица трансформаций $D = (D_{ij}) \in \mathbb{R}^{(m+1)\times(n+1)}$, где её элементы определяются рекурсивной формулой:

$$D_{ij} = \begin{cases} 0, & i = j = 0 \\ \infty, & i = 0 \text{ или } j = 0 \ (i \neq j) \\ d(a_i, b_j) + min(D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}), & \text{иначе.} \end{cases}$$

Таким образом, в элементе D_{mn} можно получить расстояние между последовательностями A и B. Будем обозначать функцию расстояния между двумя временными рядами A и B, найденного по DTW-алгоритму, через $DTW(A, B)$.

У данного алгоритма есть ряд преимуществ:

1) Длины последовательностей могут иметь (и так чаще всего и происходит) различные значения
2) Алгоритм учитывает смещение временной последовательности во времени, то есть в отличии от обычного евклидового расстояния смещение во времени ряда с «похожей» формой не приводит к возрастанию этого расстояния
3) Имеет не самую трудную программную реализацию
4) Данный алгоритм успешно применяется во многих областях, в которых используется анализ временных рядов
5) Данный алгоритм можно успешно применить к поставленной задаче

Однако, стоит обратить внимание и на некоторые недостатки данного алгоритма:

1) Алгоритм имеет сложность $O(mn)$ как по времени работы, так и по занимаем памяти
2) В некоторых случаях может выдавать неверный ответ, в силу того что он попытается объяснить изменения значений в последовательностях с помощью трансформации временной оси. То есть некоторой точке одной из последовательностей может быть поставлено в соответствие группа точек другой последовательности.

Итак, данный алгоритм даёт хорошие результаты нахождения расстояний между двумя временными последовательностями. Он учитывает форму временного ряда при сравнении с другим рядом, что делает этот метод высокоточным.

Этот метод и будет взят за основу в части нашего алгоритма, связанной со сравнением двух временных рядов. Как будет показано далее, недостатки предложенного алгоритма в конечном итоге не являются критичными.

Можно отметить, что задача анализа подобия временных рядов является весьма часто используемой в самых различных применениях [10,11]. Обзор методов сравнения подобия временных рядов есть, например, в работе [12]. К числу основных достоинств DTW относят тот факт, что алгоритм может работать с рядами разной длины.

IV. Алгоритм анализа перемещения

Шаги данного алгоритма выглядят следующим образом:

0) Сделать замер эталонных движений устройства. С этими эталонными движениями будет происходить сравнение детектируемого временного ряда от пользователя
1) Происходит замер показаний акселерометра при движении мобильного устройства пользователя
2) При помощи DTW-алгоритма вычисляем расстояния между временными рядами эталонных движений и рядом, полученном после движения устройства пользователя
3) Из вычисленных расстояний выбираем минимальное и в качестве ответа выдаём пользователю эталонное движение, соответствующее данному минимальному расстоянию.

A. Фильтр низких частот

Как отмечалось выше в реальных условиях в системе возможно наличие шумов, которые могут повлиять на итоговый результат. Тем более, что показания акселерометра на мобильном устройстве порой достигает 0.08g [10], что для нас непозволительная роскошь. В таких условиях необходимо использовать фильтры для сглаживания и собственно фильтрации данных акселерометра устройства.

При использовании фильтров данных важно учитывать одно существенное требование – использование такого фильтра должно быть достаточно производительным, чтобы его применение не создавало больших задержек при замерах показаний акселерометра в реальном времени.

Группа фильтров низких частот существует для фильтрации показаний, превышающих определённую частоту. Таким образом, фильтр будет пропускать низкие частоты, и фильтровать высокие. В качестве частоты в нашем случае выступают показания акселерометра. Фильтр низких частот хорошо подходит для нашей задачи, так позволяет нивелировать скачки в показаниях прибора.

На вход фильтру на каждом шаге поступают некоторые показания, которые называются *входными значениями*. На выходе фильтра мы получаем отфильтрованные показания, называемые *выходными значениями*.

Рассмотрим работу фильтра на одном из шагов n замера показаний: пусть I_n – входное значение прибора на шаге n, $\alpha \in (0,1]$ - коэффициент фильтрации, O_{n-1} - выходное значение фильтра на шаге $n-1$. Тогда очередное выходное значение O_n вычисляется по следующей формуле:

$$O_n = O_{n-1} + \alpha(I_n - O_{n-1})$$

Как можно видеть параметр α отвечает за то, какой процент входного значения учитывать на каждом шаге. При $\alpha = 1$, мы получим полное отсутствие фильтрации.

B. Замер эталонных движений

Теперь, применяя фильтр низких частот с параметром α, необходимо построить непустое множество $E = \{E_1, E_2, \dots, E_s\}$ эталонных временных рядов показаний акселерометра. Типы движений и количество эталонных движений определяются заранее. Для начала мы хотим научиться детектировать самые простые движения пользователя. Считаем, что мобильное устройство находится в лежачем положении экраном вверх. Будем пытаться определить следующие движения: вертикально вверх, вертикально вниз, горизонтально вправо, горизонтально влево. Отметим один из плюсов нашего алгоритма – он является довольно масштабируемым, в том смысле, что легко добавить в множество эталонных движений новые движения.

Итак, пусть мы делаем замер эталонного временного ряда показаний акселерометра E_i ($i = \overline{1 \dots s}$). Тогда на замере под номером j имеем входной вектор значений показаний акселерометра $\overline{{}_{in}e_j^i} = ({}_{in}e_x^{ij}, {}_{in}e_y^{ij}, {}_{in}e_z^{ij})$. Далее, полагая $\overline{e_0^i} = (0,0,0)$, $\forall j = \overline{1 \dots h_i}$ делаем:

$$\overline{e_j^i} = \overline{e_{j-1}^i} + \alpha\left(\overline{{}_{in}e_j^i} - \overline{e_{j-1}^i}\right) = (e_x^{ij-1}, e_y^{ij-1}, e_z^{ij-1}) + \alpha\left({}_{in}e_x^{ij} - e_x^{ij-1}, {}_{in}e_y^{ij} - e_y^{ij-1}, {}_{in}e_z^{ij} - e_z^{ij-1}\right)$$

Напомним, что длина каждого эталонного временного ряда показаний акселерометра E_i определяется некоторой (не известной нам) функцией длины временного ряда показания акселерометра $f(t, \xi(t_{E_i}))$, которая зависит от времени замера t и некоторых (не известных нам во время замера показаний) внешних факторов ξ системы. При указанных нюансах заметим, что длины эталонных рядов, вообще говоря, заранее неизвестны и могут (и, скорее всего, будут) отличаться между собой. Единственное, что мы можем контролировать при данных условиях, это – время замера движения.

Все эталонные показания акселерометра мы сохраняем в системе, и будем использовать их, в дальнейшем, для сравнения с перемещением устройства пользователя, чтобы определить полученное движение. Теперь перейдём непосредственно к измерению передвижения устройства пользователя.

C. Замер движения пользователя

Описание замера перемещения устройства пользователя в целом повторяет описание замера эталонных движений, только теперь мы не контролируем движение – оно нам заранее неизвестно.

Мы хотим сделать замер перемещения пользовательского устройства, то есть получить временной ряд показаний акселерометра $A = \{\overline{a_1}, \overline{a_2}, \dots, \overline{a_m}\}$. Как и в случае с эталонными замерами, через $\overline{{}_{in}a_k} = ({}_{in}a_x^k, {}_{in}a_y^k, {}_{in}a_z^k)$ обозначим входной вектор на шаге k и, полагая $\overline{a_0} = (0,0,0)$, $\forall k = \overline{1 \dots m}$:

$$\overline{a_k} = \overline{a_{k-1}} + \alpha\left(\overline{{}_{in}a_k} - \overline{a_{k-1}}\right) = (a_x^{k-1}, a_y^{k-1}, a_z^{k-1}) + \alpha\left({}_{in}a_x^k - a_x^{k-1}, {}_{in}a_y^k - a_y^{k-1}, {}_{in}a_z^k - a_z^{k-1}\right)$$

Таким образом, получаем временной ряд показаний акселерометра для перемещения мобильного устройства пользователя. Заметим ещё раз, что время замера данного временного ряда должно совпадать со временем замера эталонных временных рядов, так как иначе сравнение полученных показаний просто не имеет смысла. Теперь можно перейти к анализу этого ряда.

D. Анализ пользовательского ряда

После всех проделанных измерений у нас есть все компоненты, указанные в математической постановке задачи для детектирования движения мобильного устройства.

У нас есть система $E = \{E_1, E_2, \dots, E_s\}$ эталонных временных рядов показаний акселерометра и временной ряд показаний акселерометра $A = \{\overline{a_1}, \overline{a_2}, \dots, \overline{a_m}\}$ полученный в результате движения пользователя. Напомним, что наша цель найти $p = arg \min_{i:E_i \in E} D(A, E_i)$.

В качестве функции расстояния D выбирается DTW-расстояние между временными рядами. То есть мы решаем задачу $p = arg \min_{i:E_i \in E} DTW(A, E_i)$.

Тогда, чтобы определить расстояние между временной последовательностью A и некоторой эталонной временной последовательностью $E_i \in E$, необходимо построить матрицу трансформации $D_{A,E_i} \in \mathbb{R}^{(m+1)\times(h_i+1)} = (D_{kj}^i)$ для данных временных последовательностей, где каждый её элемент вычисляется по рекурсивной формуле:

$$D_{kj}^i = \begin{cases} 0, & k = j = 0 \\ \infty, & k = 0 \text{ или } j = 0 \ (k \neq j) \\ d\left(\overline{a_k}, \overline{e_j^i}\right) + min\left(D_{k,j-1}^i, D_{k-1,j}^i, D_{k-1,j-1}^i\right), & \text{иначе.} \end{cases}$$

Здесь в качестве расстояния d между векторами-элементами временных рядов используется квадрат эвклидовой метрики. Именно квадрат, поскольку производить лишнее действие вычисления квадратного корня не имеет смысла. В добавок к этому необходимо надо сказать, что при расчёте квадратного корня числа с плавающей точкой теряется точность вычислений, что для нас является нежелательным эффектом.

В самом начале расчёта значений матрицы, все элементы первой строки и первого столбца полагаются равными ∞, кроме самого первого, Это начальная инициализация элементов, которые не соответствуют ни одной паре векторов из A и E_i. Матрица трансформаций будет выглядеть следующим образом:

		$\overline{e_1^i}$	...	$\overline{e_{h_i}^i}$
	0	∞	...	∞
$\overline{a_1}$	∞	$D_{1,1}^i$	...	D_{1,h_1}^i
...	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\overline{a_m}$	∞	$D_{m,1}^i$	...	D_{m,h_i}^i

После полного рекурсивного обхода всей матрицы в последнем (расположенном в нижнем правом углу) её элементе D^{i}_{m,h_i} будет вычислено расстояние между входными временными рядами показаний акселерометра, то есть:

$$DTW(A, E_i) = D^{i}_{m,h_i}$$

Имея на руках способ вычисления расстояния между временными рядами показаний акселерометра, можно получить ответ, действуя следующим образом:

0) Пусть искомое значение $p = 0$, а переменную, обозначающую минимальное расстояние между пользовательским временным рядом и эталонным временным рядом, назовём w и положим равной ∞. Положим $i = 1$;
1) Мы находимся на шаге i. Тогда при помощи DTW-алгоритма вычисляем значение $D_i = DTW(A, E_i)$;
2) Если $w < D_i$, переходим к шагу 3), иначе – полагаем $w = D_i$, $p = i$;
3) Увеличиваем i на 1, и если $i < s$, то переходим на шаг 1);
4) В качестве ответа выдаём значение p.

Отметим, что все D_i являются конечными значениями, а значит и алгоритм выдаст нам ненулевое значение p. Теперь мы получили требуемое значение и можем выдать пользователю движение, соответствующее полученному значению p.

V. Эксперименты

В самом начале, чтобы продемонстрировать недостатки метода последовательностей одинаковой длины приведём процент успешных определений движений для него. Как уже неоднократно оговаривалось, измерения проводятся для четырёх типов движений: вертикально вверх – Up, вертикально вниз – Down, горизонтально вправо – Right, горизонтально влево – Left. Телефон находится в горизонтальном положении экраном вверх. В таблице 1 указан процент успешно определённых движений для каждого из четырёх приведённых типов. Для каждого типа движения было проделано 100 попыток анализа перемещения.

Таблица 1: Анализ для метода последовательностей одинаковой длины.

Тип движения	Точность
Up	**42 %**
Down	**40 %**
Right	**37 %**
Left	**38 %**

Как можно увидеть, точность вычислений является очень низкой. В среднем ниже 40 %. В таких условиях для детектирования определённого движения метод подбрасывания монетки даёт лучший результат, чем представленный алгоритм.

Теперь попробуем сделать измерения алгоритма с использованием DTW-расстояния, но без применения фильтра низких частот. В таблице 2, как и в таблице 1, для каждого типа движения указан процент правильно определённых движений.

Таблица 2: Анализ DTW-метода без применения фильтра

Тип движения	Точность
Up	**83 %**
Down	**87 %**
Right	**82 %**
Left	**81 %**

Точность результатов возросла более чем в два раза для каждого из направлений. Средняя точность находится в районе отметки в 80 %. То есть лишь в двух из 10 случаев алгоритм даёт помарку. Как мы видим, алгоритм динамической трансформации временной шкалы демонстрирует очень хорошие результаты.

Чтобы нивелировать неожиданные скачки в показаниях, которые могут повлиять на результат анализа и сгладить кривую временного ряда, применим фильтр низких частот. В таблице 3 представлены результаты измерений уже с использованием фильтра низких частот.

Таблица 3: Анализ перемещения с использованием DTW-расстояния и фильтра низких частот

Тип движения	Точность
Up	**89 %**
Down	**93 %**
Right	**89 %**
Left	**88 %**

Как видим, применение фильтра позволило повысить точность анализа. Средняя точность составляет чуть менее 90 %, то есть лишь в одном из 10 случаев мы получим ошибку вычисления.

VI. Заключение

В рамках данной статьи был разработан алгоритм анализа перемещений мобильного устройства. Для этого были поставлены цели и задачи в работе, предоставлены общая и математическая формулировки задач. Использование веб-интерфейсов позволило реализовать разработанный алгоритм на языке программирования JavaScript.

Для проверки качества алгоритма было произведено его тестирование на реальном мобильном устройстве. Полученные результаты демонстрируют высокое качество разработанного алгоритма. Такой алгоритм можно распространить на большее число движений, то есть он является масштабируемым. Анализ более сложных движений может потребовать более совершенных фильтров и методов анализа движений.

Библиография

[1] Временной ряд http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9_%D1%80%D1%8F%D0%B4 Retrieved: Apr, 2020

[2] Ghazi Al-Naymat, Sanjay Chawla, Javid Taheri. Sparse DTW: A novel approach to speed up Dynamic Time Warping [Электронный ресурс]. URL: https://arxiv.org/pdf/1201.2969v1.pdf

[3] Eamonn J. Keogh, Michael J. Pazzani. Derivative Dynamic Time Warping, Section 1 [Электронный ресурс]. URL: https://www.cs.rutgers.edu/~pazzani/Publications/sdm01.pdf Retrieved: Apr, 2020

[4] Pavel Senin. Dynamic Time Warping Algorithm Review [Электронный ресурс]. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.4905&rep=rep1&type=pdf Retrieved: Apr, 2020

[5] Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques // JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, MARCH 2010, ISSN 2151-9617 [Электронный ресурс]. URL: https://arxiv.org/ftp/arxiv/papers/1003/1003.4083.pdf Retrieved: Apr, 2020

[6] Titus Felix FURTUNĂ. Dynamic Programming Algorithms in Speech Recognition // Revista Informatica Economică nr, 2008 [Электронный ресурс]. URL: http://www.revistaie.ase.ro/content/46/S%20-%20Furtuna.pdf Retrieved: Apr, 2020

[7] Maher Khemakhem and Abdelfettah Belghith. Towards A Distributed Arabic OCR Based on the DTW Algorithm: Performance Analysis // The International Arab Journal of Information Technology, Vol. 6, No. 2, April 2009 [Электронный ресурс]. URL: https://ccis2k.org/iajit/PDF/vol.6,no.2/7TDAOBDAPA153.pdf

[8] W3C Web API https://www.w3.org/2006/webapi/ Retrieved: Apr, 2020

[9] JavaScript Web API https://developer.mozilla.org/en-US/docs/Web/API Retrieved: Apr, 2020

[10] Lin J., Khade R., Li Y. Rotation-invariant similarity in time series using bag-of-patterns representation //Journal of Intelligent Information Systems. – 2012. – Т. 39. – №. 2. – С. 287-315.

[11] Lhermitte S. et al. A comparison of time series similarity measures for classification and change detection of ecosystem dynamics //Remote sensing of environment. – 2011. – Т. 115. – №. 12. – С. 3129-3152.

[12] Namiot D., Pokusaev O. On mobility patterns in Smart City. – 2019.

On using web interfaces to analyze mobile device movements

Igor Petrov, Dmitry Namiot

Abstract — **In this paper, we consider the algorithms for using web interfaces for analyzing the movements of mobile devices, and we also developed and implemented an algorithm for such a movement analysis with a user interface. The task is to create some universal JavaScript library that can be used in mobile web applications. The library used web interfaces to access device movement data. to analyze the movement of a mobile device. An algorithm for determining displacements, based on a comparison of the similarity of time series, is proposed and implemented. The analysis of the properties of the proposed algorithm is carried out. At this stage, the algorithm can with high probability detect the movement of the telephone for predefined sets of movements. To implement the algorithm, the latest standards of web programming languages and development technologies were used. Experiments were conducted on a real mobile device and conclusions were drawn on the implementation of the constructed algorithm. Using the web interface will allow you to apply the developed algorithm both in browser applications on mobile phones and in mobile applications that use web technologies in their source code. The developed algorithm does not depend on the platform of a specific mobile device, since the web interfaces used are supported in most existing mobile web platforms.**

Keywords — **WEB API, DTW- algorithm, accelerometer.**

REFERENCES

[1] Vremennoj rjad http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B9_%D1%80%D1%8F%D0%B4 Retrieved: Apr, 2020

[2] Ghazi Al-Naymat, Sanjay Chawla, Javid Taheri. Sparse DTW: A novel approach to speed up Dynamic Time Warping [Jelektronnyj resurs]. URL: https://arxiv.org/pdf/1201.2969v1.pdf

[3] Eamonn J. Keogh, Michael J. Pazzani. Derivative Dynamic Time Warping, Section 1 [Jelektronnyj resurs]. URL: https://www.cs.rutgers.edu/~pazzani/Publications/sdm01.pdf Retrieved: Apr, 2020

[4] Pavel Senin. Dynamic Time Warping Algorithm Review [Jelektronnyj resurs]. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.4905&rep=rep1&type=pdf Retrieved: Apr, 2020

[5] Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques // JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, MARCH 2010, ISSN 2151-9617 [Jelektronnyj resurs]. URL: https://arxiv.org/ftp/arxiv/papers/1003/1003.4083.pdf Retrieved: Apr, 2020

[6] Titus Felix FURTUNĂ. Dynamic Programming Algorithms in Speech Recognition // Revista Informatica Economică nr, 2008 [Jelektronnyj resurs]. URL: http://www.revistaie.ase.ro/content/46/S%20-%20Furtuna.pdf Retrieved: Apr, 2020

[7] Maher Khemakhem and Abdelfettah Belghith. Towards A Distributed Arabic OCR Based on the DTW Algorithm: Performance Analysis // The International Arab Journal of Information Technology, Vol. 6, No. 2, April 2009 [Jelektronnyj resurs]. URL: https://ccis2k.org/iajit/PDF/vol.6,no.2/7TDAOBDAPA153.pdf

[8] W3C Web API https://www.w3.org/2006/webapi/ Retrieved: Apr, 2020

[9] JavaScript Web API https://developer.mozilla.org/en-US/docs/Web/API Retrieved: Apr, 2020

[10] Lin J., Khade R., Li Y. Rotation-invariant similarity in time series using bag-of-patterns representation //Journal of Intelligent Information Systems. – 2012. – T. 39. – #. 2. – S. 287-315.

[11] Lhermitte S. et al. A comparison of time series similarity measures for classification and change detection of ecosystem dynamics //Remote sensing of environment. – 2011. – T. 115. – #. 12. – S. 3129-3152.

[12] Namiot D., Pokusaev O. On mobility patterns in Smart City. – 2019.