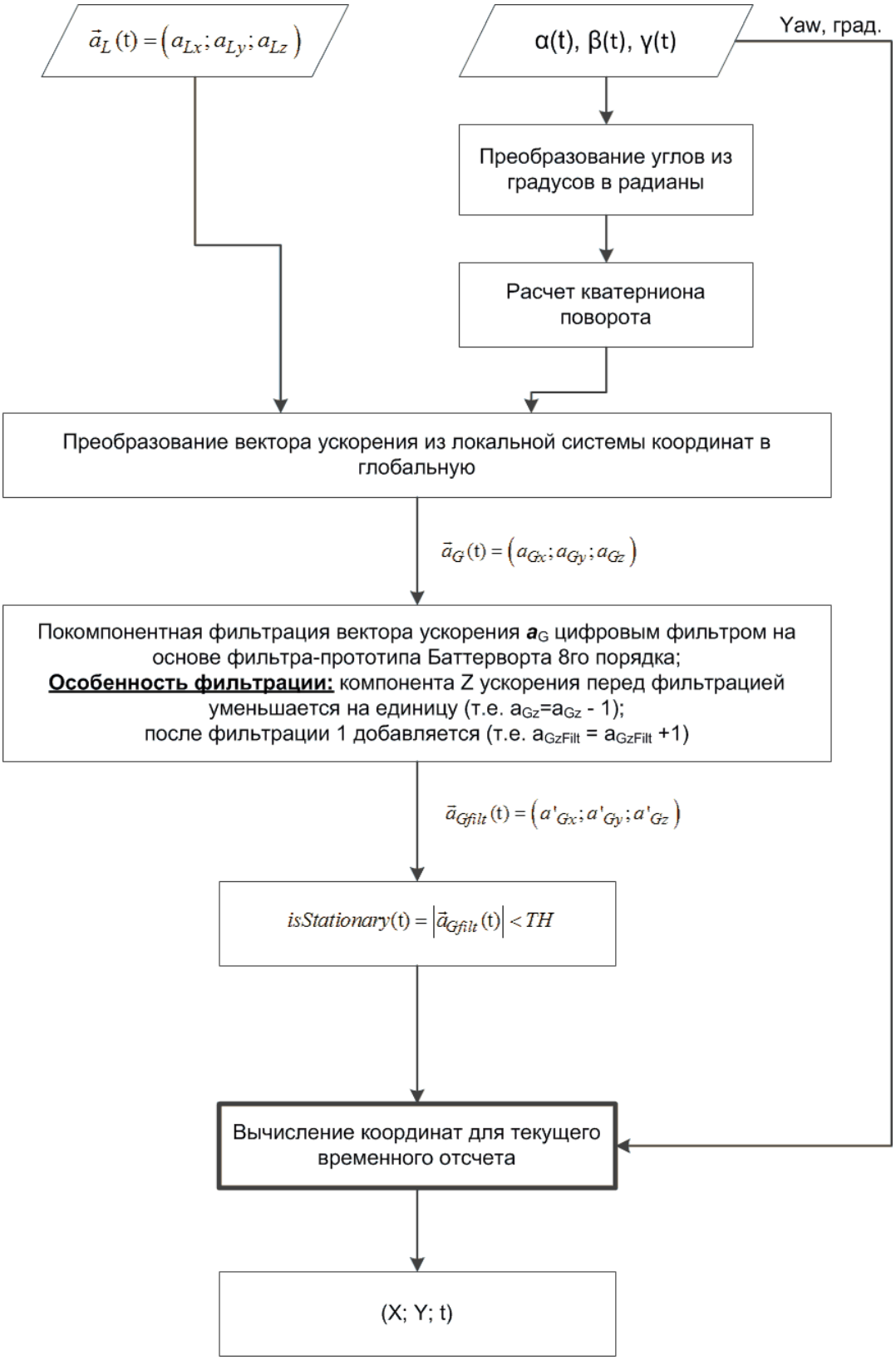
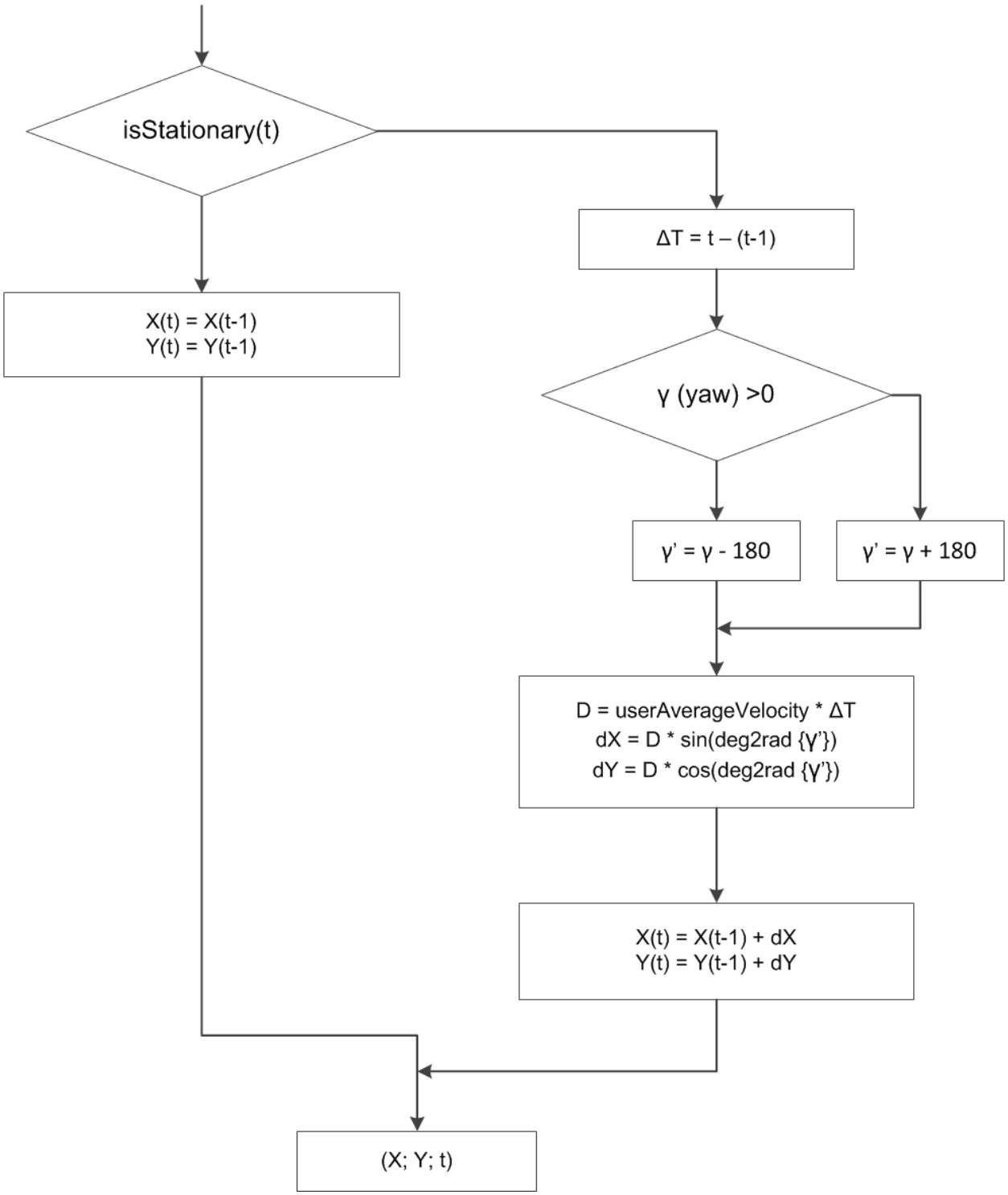
Алгоритм навигации по сенсорам (с фиксированным шагом)

ОБЩАЯ СХЕМА



Вторая часть алгоритма



Исходные данные

Входные данные для алгоритма:

Для каждого *timestamp* (по умолчанию считали, что интервал прихода показаний акселерометра и углов ориентации устройства равен около 20 мс) на входе алгоритма есть:

- показания акселерометра (три составляющих (a_x; a_y; a_z), измеренные в грав.единицах и ориентированные согласно правилам, описанным в документе "*Обертки для IMU для Android и iOS.pdf*");
- значения трех углов ориентации устройства (α, β, γ или *pitch*, *roll* и *yaw* в градусах в той ориентации, которая описана в документе "*Обертки для IMU для Android и iOS.pdf*");
- начальное положение пользователя (которое будет выдаваться процедурой Инициализации либо на этапе тестирования фиксироваться);

Выходные данные для алгоритма:

Для каждого значения *timestamp* на выходе алгоритма есть значения координат пользователя (X; Y) относительно карты.

1. Расчет кватерниона поворота из локальной системы координат в глобальную

Формирование на основе углов ориентации устройства кватерниона поворота.

Входные параметры:

α, β, γ (Pitch, roll, yaw) — углы ориентации (в рад.)

Выходные параметры:

$q=[q_1\ q_2\ q_3\ q_4]$ — Кватернион поворота локальной СК в глобальную

$$q = \begin{pmatrix} \cos\left(\frac{yaw}{2}\right) * \cos\left(\frac{pitch}{2}\right) * \cos\left(\frac{roll}{2}\right) - \sin\left(\frac{yaw}{2}\right) * \sin\left(\frac{pitch}{2}\right) * \sin\left(\frac{roll}{2}\right) \\ \cos\left(\frac{yaw}{2}\right) * \sin\left(\frac{\text{pitch}}{2}\right) * \cos\left(\frac{roll}{2}\right) - \sin\left(\frac{yaw}{2}\right) * \cos\left(\frac{\text{pitch}}{2}\right) * \sin\left(\frac{roll}{2}\right) \\ \cos\left(\frac{yaw}{2}\right) * \cos\left(\frac{\text{pitch}}{2}\right) * \sin\left(\frac{roll}{2}\right) + \sin\left(\frac{yaw}{2}\right) * \sin\left(\frac{\text{pitch}}{2}\right) * \cos\left(\frac{roll}{2}\right) \\ \cos\left(\frac{yaw}{2}\right) * \sin\left(\frac{\text{pitch}}{2}\right) * \sin\left(\frac{roll}{2}\right) + \sin\left(\frac{yaw}{2}\right) * \cos\left(\frac{\text{pitch}}{2}\right) * \cos\left(\frac{roll}{2}\right) \end{pmatrix}$$

2. Преобразование вектора ускорения из локальной системы координат (СК) в глобальную СК

Алгоритм, который преобразует зарегистрированный вектор ускорений из локальной СК устройства в глобальную СК (правосторонняя).

Входные параметры:

a_{Lx}, a_{Ly}, a_{Lz}	Текущее значение вектора ускорения в локальной СК устройства (в грав. единицах)
$q=[q_1\ q_2\ q_3\ q_4]$	Кватернион поворота локальной СК в глобальную

Выходные параметры:

a_{Gx}, a_{Gy}, a_{Gz}	Ускорения в глобальной системе координат (в грав. единицах)

Выражение для преобразования:

$$[A,\ a_{Gx},\ a_{Gy},\ a_{Gz}] = \text{quatProd}\ (\ \text{quatProd}\ (q,\ [0\ a_{Lx},\ a_{Ly},\ a_{Lz}]\ ),\ \hat{q});$$

где $\hat{q}=[q_1\ \ -q_2\ \ -q_3\ \ -q_4]$ – сопряженный кватернион;

A – скалярная составляющая результирующего кватерниона;

quatProd (a, b) – произведение двух кватернионов:

$$\begin{cases} result(1) = a(1)*b(1) - a(2)*b(2) - a(3)*b(3) - a(4)*b(4) \\ result(2) = a(1)*b(2) + a(2)*b(1) + a(3)*b(4) - a(4)*b(3) \\ result(3) = a(1)*b(3) - a(2)*b(4) + a(3)*b(1) + a(4)*b(2) \\ result(4) = a(1)*b(4) + a(2)*b(3) - a(3)*b(2) + a(4)*b(1) \end{cases}$$

3. Применение фильтра низких частот для покомпонентной обработки вектора ускорения в глобальной СК

Фильтрация данных акселерометра низкочастотным цифровым фильтром, спроектированным на основе низкочастотного аналогового фильтра-прототипа Баттерворта, для удаления шумов и более эффективного обнаружения паттернов шагов пользователя.

Входные параметры:

a_{Gx}, a_{Gy}, $a_{Gz}=a_{Gz}-1$	ускорения вдоль осей X, Y и Z в глобальной СК (в грав. ед.)
a, b	Коэффициенты цифрового фильтра (зависят от частоты дискретизации данных акселерометра, и частоты отсечки); задаются в виде констант; рассчитываются в Матлаб

Выходные параметры:

a'_{Gx}, a'_{Gy}, $a'_{Gz}=a'_{Gz}+1$	Отфильтрованные значения проекций ускорения в системе координат устройства (в грав. ед.)

Ускорение по оси Z на входе фильтра должно быть уменьшено на 1 с целью получения нулевого мат.ожидания, что позволяет без граничных эффектов провести фильтрацию. На выходе блока фильтрации необходимо провести обратную операцию – увеличить отфильтрованную Z компоненту ускорения на 1.

Каждая компонента ускорения фильтруется отдельно.

4. Детектор шага

Алгоритм, который на основании отфильтрованного показания акселерометра и заданного порогового значения ускорения определяет, соответствует ли зарегистрированное значение ускорения статическому (пользователь неподвижен) или динамическому (пользователь идет) случаям.

Входные параметры:

TH	пороговое значение для детектирования шага (если ускорение больше порогового, то это динамический случай, если меньше – статический)
a'_{Gx}, a'_{Gy}, a'_{Gz}	Отфильтрованные значения ускорения

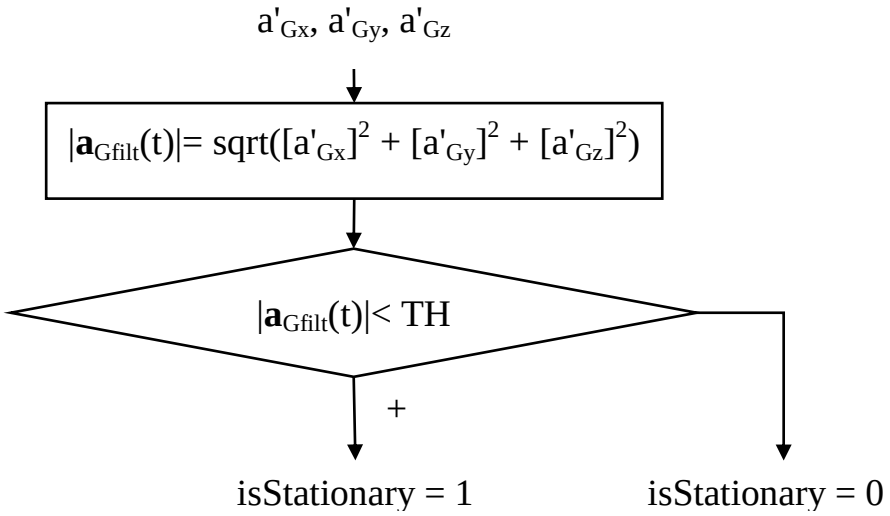
Выходные параметры:

isStationary	Признак движения: 1 – статический случай (движение нет), 0 – динамический случай (движение есть)

Значения по умолчанию

TH = 1.1 грав. ед. (определено экспериментально; данный параметр нужно вынести в файл с настройками);

Блок-схема



5. Вычисление координат для текущего временного отсчета (часть 2 блок-схемы)

Последовательность действий в целом понятна из блок-схемы (часть 2). Отмечу только параметр "userAverageVelocity". По определению это средняя скорость движения человека, в м/с. В Матлаб-версии установили ее значение равным 3,2 м/с. Скорость это очень большая, но так как момент времени, в течение которого алгоритм детектирует шаг, довольно короткий, то происходит взаимная компенсация.

6. Коррекция координат по карте и сетке

После расчета координат (X; Y) для момента времени *t*, не всегда их необходимо выдавать пользователю. Имею в виду, что положение в User Application передается на данный момент раз в секунду, а координаты вычисляются с частотой прихода показаний от сенсоров, т.е. примерно 50 раз в секунду.

Каждое положение точки (т.е. 50 раз в секунду) необходимо проверять на принадлежность карте при помощи метода "CheckXYOnTheMap":

ПСЕВДОКОД

```
if posX(tIndex) < 0
    posX(tIndex) = 0;
if posY(tIndex) < 0
    posY(tIndex) = 0;
end
if posX(tIndex) > mapLength
    posX(tIndex) = mapLength;
end
if posY(tIndex) > mapWidth
    posY(tIndex) = mapWidth;
end
```

где `posX(tIndex)` и `posY(tIndex)` – соответствующие координаты положения в метрах;

`mapLength`, `mapWidth` – соответствующие размеры карты в метрах.