

### 3.4 Algorithm implementation

Here, we describe the algorithm implementation for segmenting and evaluating exercise performances in a stand-alone Android<sup>7</sup> smartphone application.

#### 3.4.1 Teach-mode implementation

For any exercise to be monitored, the smartphone body attachment and the motion feature representing the sinusoidal pattern need to be chosen. Subsequently, during the Teachmode, the application records inertial sensor data from accelerometers, magnetometers, and the orientation API, and stores the data for the later analysis. Since the Teach-mode is performed under therapist supervision, no real-time feedback will be provided. Once the trainee completes an exercise session with a pre-set number of repetitions, the application loads the stored data and extracts the exercise model parameters. The following processing steps were applied to derive the model parameters.

**Filtering.** The selected motion feature was filtered using a moving average to remove tremor-induced noise and sensor noise. The window size was set proportional to the amount of data acquired (i.e.  $WindowSize = DataSize \times ScalingFactor$ , with  $ScalingFactor$  determined empirically and set to 80). We observed that this approach provided consistent results across the different exercises. Since the number of repetitions is preconfigured, we assume that the total data amount recorded is proportional to the movement speed during the exercise execution: the faster a trainee performs the exercise, the lower muscular tremor would be, and thus, data averaging can be reduced. Bounds were applied to the averaging window size, to prevent ineffective averaging for very fast/slow repetitions ( $15 \leq WindowSize \leq 31$  sa.)

**Period estimation.** By estimating the position of positive and negative peaks in the filtered motion feature, exercise repetitions were counted. For the Arm abduction exercise,  $O_y$  will be maximal when the arm is raised to shoulder height. It reaches its minimum value when the arm returns to the neutral position (arm aligned to the trunk). We used an adaptive hill climbing algorithm to detect positive and negative peaks (denoted in Figure 7 by  $pp_i$  and  $np_i$ , respectively), given a peak threshold  $\theta$ . The hill-climbing algorithm is a popular first choice among optimisation algorithms. While there are many alternatives, such as simulated annealing or tabu search, hill-climbing can achieve sufficient or better results if runtime is constrained, such as in the real-time system targeted here. Figure 7 illustrates an example motion waveform with the exercise parameters. In particular a maximum (minimum) was selected, if both sides (uphill  $h_u$  and downhill  $h_d$ ) are greater than the peak threshold  $\theta$ . In this work, we estimate  $\theta$  during the Teach-mode. Starting from an initial setting  $\theta = \theta_{in}$ ,  $\theta$  was adjusted in steps of 1.2 until the pre-set repetition number was obtained. We set the number of exercise repetitions to ten. The fitted  $\theta = \theta_{opt}$  was subsequently applied during Trainmode operation.

<sup>7</sup> Android OS, Google. <http://www.android.com>