Real-Time Gait Phase Detection on Wearable Devices for Unsupervised Gait

Jiaen Wu (wujiae@ethz.ch)
ETH Zurich https://orcid.org/0000-0001-5020-873X
Barna Becsek
Magnes AG
Alessandro Schaer
Magnes AG
Henrik Maurenbrecher
Magnes AG
George Chatzipirpiridis
Magnes AG
Olgac Ergeneman
Magnes AG
Salvador Pané
ETH Zurich https://orcid.org/0000-0003-0147-8287
Hamdi Torun
Northumbria University
Bradley Nelson
ETH Zurich

Article

Keywords: Real-Time gait phase detection, reduced support vector machine, embedded system algorithms, automatic gait analysis, haptic feedback, neurorehabilitation, free-living gait

Posted Date: March 31st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1461319/v2

Real-Time Gait Phase Detection on Wearable Devices for Unsupervised Gait

Jiaen Wu, Barna Becsek, Alessandro Schaer, Henrik Maurenbrecher, George Chatzipirpiridis, Olgac Ergeneman, Salvador Pané, Hamdi Torun, Bradley J. Nelson

***Abstract*— Detecting gait phases unobtrusively and reliably in real-time for long-term unsupervised walking is important for clinical gait rehabilitation and early diagnosis of neurological diseases. Due to hardware limitations in wearable devices (e.g., memory and computation power), reliable real-time gait phase detection remains a challenge for unsupervised mobility assessment. In this work, a hybrid algorithm combining a reduced support vector machine (RSVM) and a finite state machine (FSM) is developed to address this. K-means clustering is used to reduce the number of support vectors (SVs) by constructing a smaller dataset that contains the most informative data points. For each gait phase prediction, an FSM is designed to validate the prediction and correct misclassifications. After SV reduction, the model size is reduced by 88%, and the computation time is reduced by a factor of 36, with only a minor degradation in prediction performance of 4.12%, 2.34%, and 4.85% for sensitivity, specificity, and accuracy, respectively. The real-time classification performance of the algorithm is evaluated by twenty healthy subjects walking along a predefined route with unsupervised free-living gait. The proposed algorithm demonstrated promising real-time performance, with an accuracy of 91.51%, a sensitivity of 91.70%, and a specificity of 95.77% across all test subjects. The algorithm also demonstrated its robustness with respect to different values of walking speed, cadence, and stride length.**

***Index Terms*— Real-Time gait phase detection, reduced support vector machine, embedded system algorithms, automatic gait analysis, haptic feedback, neurorehabilitation, free-living gait.**

I. Introduction

GAIT characteristics (e.g., gait speed), considered as the sixth vital sign, can be used as a digital biomarker for personalized health monitoring and assessment [1], [2]. Probing and evaluating gait characteristics such as the gait cycle is essential for clinical applications, including but not limited to estimating the risk of falls [3], measuring the efficacy of interventions and rehabilitation [2], and early diagnosis of various neurological diseases [4]–[6].

A gait cycle is characterized by different gait phases. Robust intra-stride gait phase detection (i.e., detecting phases within one stride) is the basis for mobile gait analysis, as most gait characteristics, including inter-stride gait features (i.e., changes between different strides), are derived from the intra-stride temporal gait phases [7]. Real-time gait phase detection can be utilized for closed-loop real-time feedback, which is essential for clinical gait rehabilitation as it enables patients to recognize their walking abnormalities through the real-time feedback and make conscious corrections immediately, and also help patients with spinal cord injury or brain trauma to restore the walking function [8], [9]. It can also be utilized for many clinical rehabilitation applications, such as controlling the timing of stimulation sequences for functional electrical stimulation (FES) [9], [10] and epidural electrical stimulation (EES) [11], [12]. FES/EES uses small electrical pulses delivered by skin or implanted electrodes to artificially generate action potentials in subcutaneous efferent neurons during the swing phase of the paretic foot. Based on gait phase transition, patients can control the contraction of paretic muscles and induce movement in the affected limbs by changing the frequency or the amplitude of these pulses. Wagner et al. [11] found that real-time spatiotemporal gait pattern-based EES during walking can significantly improve the potency of leg movement when compared to continuous EES that is not based on gait patterns, and it allows paralyzed patients to walk and adapt their leg movements to stand after a few months. Liberson et al. utilized FES to unilaterally stimulate the ankle dorsiflexion muscles during the swing phase to compensate for the drop-foot problem [10].

Though various computational technologies have been developed for offline gait phase detection as reviewed in [13], [14], only a very small number of works on online gait phase detection have been published [7], [15]–[17]. Among those online algorithms, rule-based approaches have been mostly reported due to their intuitiveness, and low computational complexity [7]. However, those approaches usually involve rule and threshold setting, as well as signal peak detection. The rules and thresholds are determined empirically based on preliminary data, thus they need to be recalibrated continuously for different people. The presence of a signal peak can only be confirmed after both the rising and falling signal edges have appeared, which may cause a significant delay, on the order of hundreds of milliseconds [7].

This work was supported by the European Union's Horizon Research and Innovation program under the Marie Sklodowska-Curie grant agreement No.764977. (*Corresponding author: Jiaen Wu*)

J. Wu, S. Pané, B. J. Nelson are with the Multi-Scale Robotics Lab, ETH Zurich, Tannenstrasse 3, CH-8092 Zurich, Switzerland (e-mail: wujiae@ethz.ch; vidalp@ethz.ch; bnelson@ethz.ch).

J. Wu, B. Becsek, A. Schaer, H. Maurenbrecher, G. Chatzipirpiridis, O. Ergeneman are with Magnes AG, Selnaustrasse 5, 8001 Zurich, Switzerland. (e-mail: barna@magnes.ch; aschaer@magnes.ch; henrikm@magnes.ch; chgeorge@magnes.ch; oergeneman@magnes.ch).

H. Torun is with the Department of Mathematics, Physics and Electrical Engineering, Northumbria University, Newcastle upon Tyne, UK. (e-mail: hamdi.torun@northumbria.ac.uk).

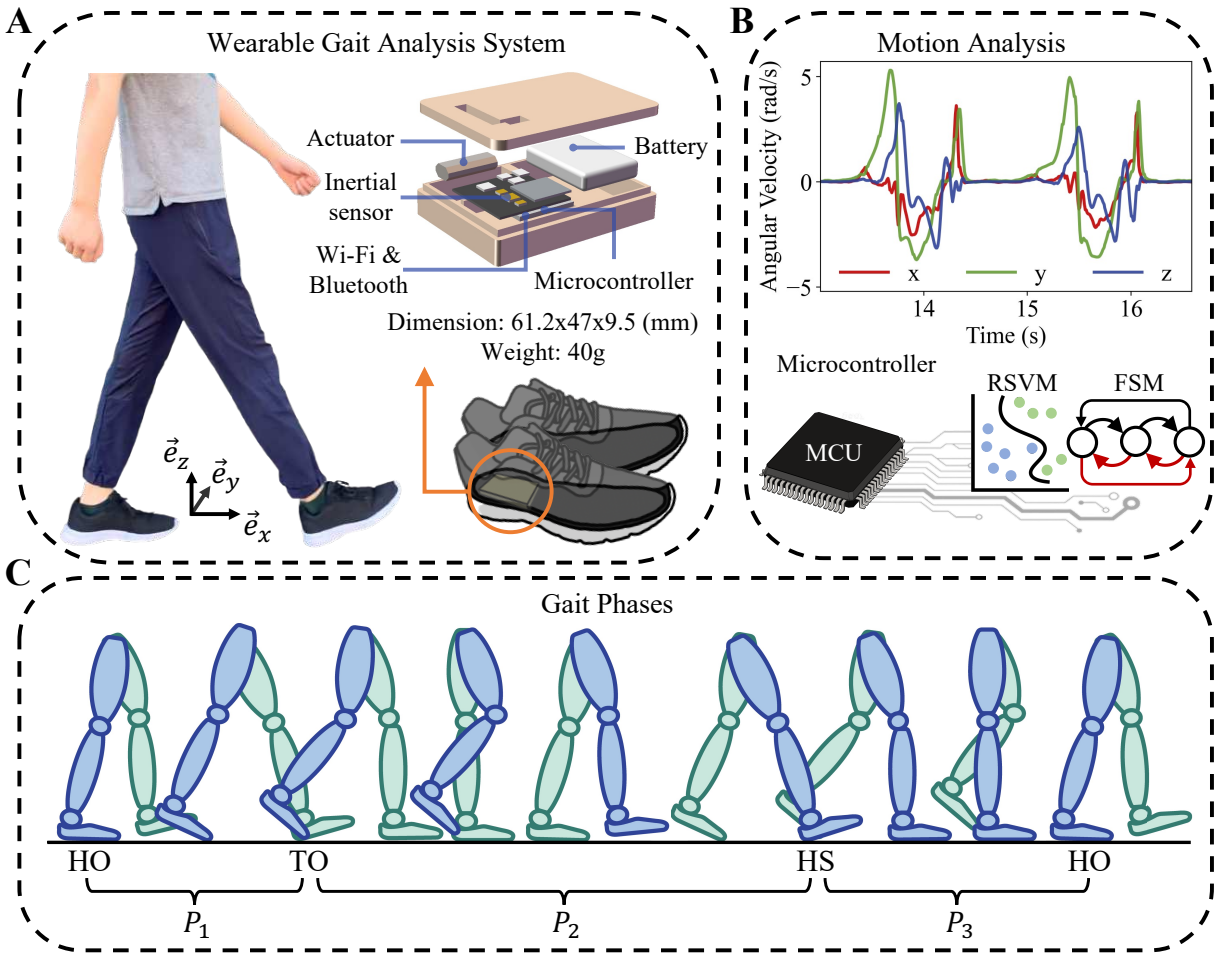

Fig. 1. A schematic illustration of the wearable gait analysis system. **A**. The wearable gait analysis system consists of an actuator, an inertial sensor, a Wi-Fi and Bluetooth module, a battery, and a microcontroller. The system is embedded in a pair of shoes at the location under the heel. **B**. The motion of the foot is captured by the sensing system during human walking. Only gyroscope signals are used for real-time gait phase detection. A hybrid algorithm of RSVM and FSM is developed to process raw motion signals on the local microcontrollers and output the gait phases in real-time. **C**. An illustration of gait phases considered in this paper. $\boldsymbol{P_1}$: HO–TO, $\boldsymbol{P_2}$: TO–HS, $\boldsymbol{P_3}$: HS–HO.

Machine learning techniques for gait analysis have grown in prominence in recent years due to their high accuracy and automated nature. Mannini et al. placed an IMU on the foot instep and detected gait phases in real-time by using Hidden Markov Model [16]. However, their algorithm was only validated with offline treadmill walking data by a leave-one-subject-out method. Chen et al. used a standard support vector machine (SVM) for detecting gait phases, i.e., flat foot (FF), HO, TO, and swing, based on seven force-sensitive resistors (FSR) and an accelerometer placed on shoes [17]. The algorithm was validated on treadmill walking data and achieved an overall accuracy of 94.08%. In this work, the sensor data were transferred to a remote computer via Bluetooth and the online algorithm was executed on the computer instead of a local microcontroller on shoes. This limits the use of this device to lab environments. A continuous full day free-living walking assessment is not possible with this system. Besides, the readout of FSR sensors are dependent on the user's body weight. Therefore, the system needs calibration for different users. Vu et al. proposed a deep learning algorithm based on IMU data to detect the gait cycle percentage [18], which is defined as the percentage of the current sample in the gait cycle. The algorithm's performance was only assessed on the offline testing data from seven healthy subjects, no real-time validation experiments were conducted. This deep learning algorithm also runs on a remote computer.

With an online phase detection algorithm running on a remote device, users can only walk in a confined environment. If users are not in the sensor range, the gait phases cannot be detected. This also limits the use of real-time closed-loop functionalities such as FES. Most importantly, the aforementioned automatic online algorithms are validated using walking data collected under controlled and supervised conditions, e.g., in a laboratory or a hospital [19]. The variability across those in-lab walking strides is quite low. However, the uncontrolled and unsupervised free-living gait, i.e., walking in the real world, is more complex and irregular, which often incorporates dynamic walking speed, varying walking surface, and inclinations [20]. A recent study from Khandelwal et al. has shown that the published gait event detection algorithms that perform well for walking in a controlled indoor environment exhibit significantly degraded performance when evaluated

with a less controlled outdoor walking, with a combined median F_1 score decreased from 0.98, 0.94 to 0.82, 0.53 for HS and TO, respectively [20]. Reliable real-time gait phase detection for unsupervised free-living walking still remains a big challenge, especially for memory and computation-limited embedded systems that require low power operation.

In this paper, a robust algorithm incorporating reduced support vector machine (RSVM) and finite-state machine (FSM) is proposed for real-time reliable phase detection on unsupervised free-living gait, with resource-limited microcontrollers (Fig. 1A-B). Compared to the standard support vector machine (SVM), we use a cascaded k-means clustering approach to reduce the model size and increase the computational speed that fits the embedded environment while still keeping the high classification accuracy. An FSM is designed to model the sequential property of gait phases and complement the RSVM with contextual information about a logical temporal sequence of gait phases. This algorithm is implemented on the microcontroller of a miniature wearable device that is embedded in the shoes for triggering real-time haptic feedback (Fig. 1A). The effectiveness of the proposed algorithm is validated with long-term fully uncontrolled and unsupervised gait performed by twenty healthy subjects, who were guided to wear the sensor-embedded shoes and walk around the predefined paths on the street at three different walking speeds.

The proposed algorithm exploits only the gyroscope signals, avoiding the need for additional sensors. No prior calibration is required for different people with different walking conditions, such as walking speeds and walking surfaces. This is the first work presenting the identification of gait phases in real-time in a wearable device under uncontrolled and unsupervised free-living walking conditions.

II. Data Collection

A. Subjects of the Study

Two groups of healthy subjects were recruited in this study. The first group consisted of four healthy subjects. They were instructed to walk on a treadmill. The datasets collected from these subjects were separated into training and testing datasets for training and optimizing the proposed algorithm offline. This study was conducted in accordance with Good Clinical Practice guidelines and the Declaration of Helsinki after receiving a declaration of clearance from the local ethics committee (BASEC Nr Req-2019-00715).

The second group consisted of (distinct) twenty healthy subjects with an average age, 29.75 ± 3.39 years, wearing European shoe sizes from 38 to 47. They were instructed to perform outdoor walking. The datasets collected from these subjects were used to validate the performance of the trained model in real-life conditions, running in real-time on the embedded electronics. The research protocol for this study was approved by grant (EK 2021-N-198) from the Ethics Commission of the ETH Zurich.

All subjects were able to walk normally with no known injuries or abnormalities that would affect their gait. Written informed consents were provided by all subjects before the experiment.

B. Data Collection System

All walking data were collected by the wearable gait analysis system *Nushu* developed by Magnes AG [21]. It consists of a pair of shoes and a mobile (iOS) application. The shoes are equipped with a low-power system on custom–developed electronics including on-board inertial sensors (STMicroelectronics, Geneva, Switzerland) and a dual-core microcontroller unit (ESP32, Espressif Systems Co., Ltd, Shanghai, China) with integrated WiFi and Bluetooth modules for wireless communication. The gyroscope data acquisition frequency is set to 100 Hz. The range of gyroscope is set to ± 2000 DPS.

C. Data Acquisition

The first group of healthy subjects was instructed to wear the sensor-embedded shoes and walk on a treadmill with a normal gait. Each subject was asked to walk three times on the treadmill at three different speeds, i.e., 0.53 m/s, 0.86 m/s, and 1.11 m/s. During each trial, the subject performed around 50 strides for each side of the left and right foot.

The second group of healthy subjects was instructed to wear the sensor-equipped shoes and walk outdoors freely along the selected path on the street, as shown in Fig. 2. The walking path includes turns, slopes, and long straight paths, in a total of approximately 500 m walkway for each bout. Each subject was asked to walk three times at three different speeds, i.e., slow speed, normal speed, and fast speed. Each subject was free to interpret what "slow", "normal" and "fast" meant for them. During these experiments, the real-time gait phase detection algorithm was running on the microcontroller of the shoe system along with the sensors during each subject's walking experiment. The gait phase prediction results were logged together with raw sensor data. The total number of free walking strides collected for real-time algorithm validation is around 50000.

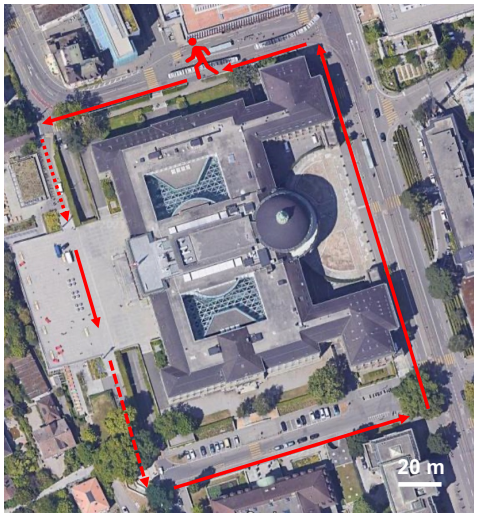

Fig. 2. Top view of the path utilized for the outdoor walking experiments. The red arrows show the closed-loop path defined for the experiments. The dotted arrow represents the uphill, the dashed arrow represents the downhill. The bouts started and stopped where the human stick figure is depicted.

Gait phases were labeled by a rule-based detection algorithm, which has been validated with the motion capture system and shows high accuracy [22]. Most importantly, it is not constrained by the lab environment and can label outdoor walking data at any time and anywhere for real-time algorithm validation.

III. Real-Time Gait Phase Detection Algorithm

A. Gait Phases

Human gait can be divided into a series of repetitive phases and events related to its cyclic nature. Each gait cycle can be divided into two main phases i.e., stance phase and swing phase [23]. These two main phases can further be divided into sub-phases by four gait events, which are HO, TO, HS, and FF. As shown in Fig. 1C, three gait phases $P_1,\ P_2,\ P_3$ that are delimited by HO, TO, and HS for each foot are considered in this paper for real-time phase detection. They are denoted as P_1: HO–TO, P_2: TO–HS, P_3: HS–HO.

B. Feature Selection

Contrary to accelerometers, gyroscopes do not carry a constant offset due to gravity. They are more immune to vibrations and environmental noises than accelerometers [16], [24]. Moreover, it has been shown that the angular velocity patterns are not affected much by inter–subjects or intra–subjects variability and different walking conditions [16], such as walking speed, or walking slope. Therefore, the data from a single gyroscope sensor are used for real-time gait phase detection.

By observing the signal distributions of angular velocity for different gait phases, angular velocity along the pronation axis, ω_z, angular-acceleration $\dot{\omega}_y$ and -velocity ω_y along the mediolateral axis, which is the most prominent movement, are selected as three input features. Fig. 1A depicts the reference frame for the coordinate axis. Due to the symmetric ankle rotation of the left and right feet, the angular velocity along the pronation axis for the right foot is flipped as $-\omega_z$. The angular acceleration, which is the discrete time derivative of the angular velocity, is computed as:

$$\dot{\omega}_y(t) = \frac{\omega_y(t+\Delta t) - \omega_y(t-\Delta t)}{2\Delta t}. \tag{1}$$

This first-order derivative is included as one of the input features to provide temporal dependence information to the algorithm. Higher-order time derivatives are discarded because they can introduce an amplification of high-frequency noise and degrade the signal-to-noise ratio [25], [26]. All signals are normalized with respect to their sample means and sample standard deviations that are obtained from the training data. The normalized input feature vector $\boldsymbol{x}$ can be denoted as: $\boldsymbol{x} = (\bar{\omega}_z, \bar{\dot{\omega}}_y, \bar{\omega}_y)$. The bar represents the normalization.

C. Reduced SVM for Real-Time Gait-Phase Detection

SVM has advantages in that it does not require a large number of training samples to construct a model and is not affected by the appearance of outliers [27]. Besides, with a clear geometrical interpretation, SVM training can always find the global optimum of the cost function [27]. Therefore, SVM is employed in this work as a nonlinear approach to classify walking data into different gait phases.

SVMs have been shown to be a promising classification model for gait phase detection, while the number of support vectors (SVs) increases with the size of the training dataset [28]. As explained in Appendix I-A and (15) an SVM classifier is constructed by a linear combination of kernel evaluation of SVs, which are the training samples lying within the soft margin of the decision boundaries. Therefore, the memory usage of an SVM model and the computation time of each SVM prediction is proportional to the number of SVs. To implement this algorithm in an embedded environment that has limited memory and computational capabilities in real-time, the number of SVs should be as small as possible. To reduce the SVM model size and the execution time while keeping the superior SVM performance, a cascaded k–means clustering method is proposed. The cascaded k-means clustering reduces the number of SVs by selecting a small portion of the most informative learning feature vectors that are close to the decision boundaries in the feature space. The rationale behind this strategy is the following: feature vectors close to the decision boundaries are more likely to become SVs of the final separation hyperplane, while feature vectors lying far from the decision boundaries have less effect on building the final separation hyperplane (explained in Appendix I-A), hence removing the SVs "far away" from the decision hyperplane has a low impact on the performance of the SVM. After removing the feature vectors with fewer contributions from the learning samples, the SVM classifier is constructed on the remaining samples. It has a much-reduced number of SVs and is referred to as RSVM for reduced SVM.

Note that in this paper, decision boundaries refers to the class boundaries in the original finite-dimensional feature space, the separation hyperplane refers to the class boundaries in the mapped high-dimensional space.

To identify this small subset of most informative feature vectors that can be used as the training samples instead of the entire training data set, an iterative data selection procedure is proposed as follows. For each iteration d, given the selected data $\mathcal{D}^d$ from the previous iteration, an unsupervised k–means clustering technique is applied to each gait phase from the selected samples to identify the predefined k_l^d number of cluster centroids. k_l^d is chosen to be $\lfloor \eta_1 \cdot N_l^d \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function, $\eta_1 \in (0,1)$, and N_l^d is the number of samples in gait phase l at iteration d. In each cluster, the samples that are far from the cluster boundary are also far from the decision boundaries of SVM, and thus are more likely to become insignificant SVs, i.e., SVs with small weight α_i as shown in (15). Therefore, those samples are thus removed for further iterations. To this end, for each cluster, the Euclidean distances from the cluster centroid to the samples are calculated and sorted from closest to furthest. To prevent excessive removal of samples that may become SVs around the cluster boundaries, a safety region close to the cluster boundaries is defined by the cluster radius r_{cluster}, i.e., the distance between the cluster centroid and the furthest data point of the cluster, and the

number of samples within the cluster. If the cluster radius is smaller than the predefined radius threshold ρ, i.e., the cluster centroid is closer to the cluster boundary, we remove $\eta_2^d \cdot 100\%$ percent of samples with a smaller distance to the centroid; If the cluster radius is large, i.e., the cluster centroid is further from the cluster boundary, we remove $\eta_3^d \cdot 100\%$ percent of samples with a higher distance to the centroid, where $\eta_2^d, \eta_3^d \in (0, 1)$. The samples within this safety region will be kept. The samples within the cluster but outside the safety region will be removed. The remaining samples within the safety regions from all clusters are used for SVM training. After training, the number of the obtained SVs is compared to the expected value of the SV number δ, which is required by the embedded memory and CPU, but with small performance degradation. Sample reduction threshold, η_2^d, η_3^d, will be decreased after each iteration d by a factor of $1 - \varepsilon$, where $0 < \varepsilon < 1$. We iterate until the number of SVs, N_{SV}^d, satisfies the requirement of the embedded environment. All the thresholds and rules are determined empirically based on the training data.

D. Finite State Machine

The SVM and RSVM show good performance for gait phase classification, while it does not consider the temporal sequence of the gait phases. Therefore, some illogical misclassifications occur between the successive gait phases, especially close to the phase transition point. To address this issue, a finite state machine (FSM) is designed to post-process the decision from the RSVM by considering the temporal gait phase sequence and making full use of the past information.

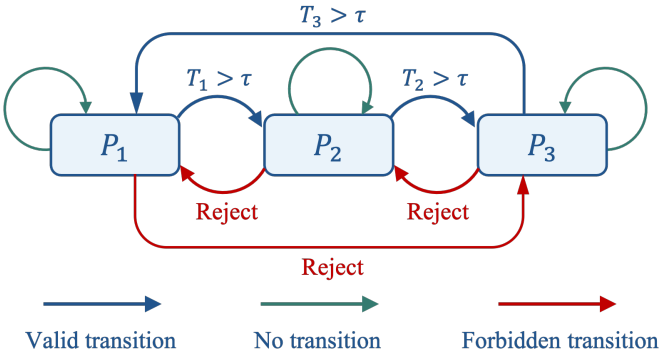

Fig. 3. A schematic illustration of algorithm FSM. $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, $\boldsymbol{P_3}$ are three gait phases, and they are considered as three states of FSM. $\boldsymbol{T_1}$, $\boldsymbol{T_2}$, $\boldsymbol{T_3}$ are the state duration for three gait phases. $\boldsymbol{\tau}$ is a predefined time constant. The blue arrows represent valid transitions, the green arrows represent no transition, red arrows represent forbidden transitions.

As illustrated in Fig. 3, each gait phase is considered as a state $\{P_1, P_2, P_3\}$ in the FSM. As gait phases occur in a repetitive temporal sequence during walking, there are only three allowed state transitions i.e., $P_1 \rightarrow P_2$, $P_2 \rightarrow P_3$, and $P_3 \rightarrow P_1$. Among those three state transitions, only the transition, with the previous state duration longer than a predefined time threshold τ, is considered as a valid transition. Other state transitions, e.g., $P_1 \rightarrow P_3$, $P_2 \rightarrow P_1$, and $P_3 \rightarrow P_2$, which cannot happen in (normal) walking, are considered as a forbidden transition and are rejected by the FSM. The pseudocode of this detailed procedure of the FSM can be found in the preprint [29]. P_E is the expected transition state from the previous state.

E. System Implementation

The kernel function used in this paper is the radial basis function (RBF) [30], which is:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2\right) = \exp(\hat{x}_{ij}), \quad (2)$$

where $\hat{x}_{ij} = -\gamma \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2 \leq 0, \ \forall \ \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}, \forall \ \gamma > 0$. In practice, the evaluation of the exponential function is an expensive operation, which is a matter of concern in particular on resource-limited platforms such as microcontrollers. To decrease the computational complexity, we approximate the RBF kernel function by another function $g(\hat{x}_{ij})$ defined as follows:

$$\exp(\hat{x}_{ij}) \approx g(\hat{x}_{ij}) := \begin{cases} 0 & \text{if } \hat{x}_{ij} \leq \chi_1 \\ \exp(\hat{x}_{ij}) & \text{if } \chi_1 < \hat{x}_{ij} < \chi_2 \\ 1 + \sum_{n=1}^{8} \frac{(\hat{x}_{ij})^n}{n!} & \text{if } \chi_2 \leq \hat{x}_{ij} \end{cases} \quad (3)$$

where χ_1 and χ_2 are the non-positive constants that determine the region of the approximation. When the exponent $\hat{x}_{ij}$ is small, the kernel is close to zero and thus is approximated to 0 in (3). When $\hat{x}_{ij}$ is close to zero, the Taylor series expansion of the exponential function is a good approximation with low computation cost as considered in (3). The degree of the Taylor series expansion, i.e., 8, is chosen by a trade-off between the approximation precision and the calculation speed. The constants χ_1 and χ_2 in (3) are determined such that the relative error between $g(\hat{x}_{ij})$ and $\exp(\hat{x}_{ij})$ is less than 10% for any $\hat{x}_{ij} \in (-\infty, 0]$. The values of the χ_1 and χ_2 used in this paper are -7.596, -2.205.

Both SVM and RSVM are constructed using Python library `sklearn` [1] on the first dataset acquired from treadmill walking. The hyperparameters of these models are optimized by the grid search approach with cross–validation [31]. After training with optimized hyperparameters, the RSVM model is transposed to C using `sklearn-porter` [2] in order to implement it on the MCU. The final, compiled model size of the RSVM is 180 KB.

IV. Results and Discussion

The gait phase detection model is trained and optimized using training data containing 75% samples of the first dataset that were collected from treadmill walking. To evaluate the performance of the classifiers for gait phase detection, three metrics are employed: accuracy, sensitivity, and specificity.

A. Offline Performance of Gait Phase Detection

The offline performance of RSVM is first assessed using the testing data containing 25% samples of the first dataset, which were collected on a treadmill at three different walking

[1] `https://github.com/scikit-learn/scikit-learn`
[2] `https://github.com/nok/sklearn-porter`

TABLE I
THE COMPARISON OF THE OFFLINE CLASSIFICATION PERFORMANCE METRICS FOR SVM AND RSVM BASED ON THE TESTING DATASET COLLECTED ON A TREADMILL.

Gait Phase	Sensitivity		Specificity		Accuracy	
	SVM	RSVM	SVM	RSVM	SVM	RSVM
P_1	91.67%	90.75%	99.03%	95.89%	96.11%	91.26%
P_2	97.20%	93.51%	98.74%	96.14%		
P_3	97.82%	90.05%	96.02%	94.74%		
Overall	95.56%	91.44%	97.93%	95.59%		

TABLE II
THE COMPARISON OF THE NUMBER OF SVS, THE MODEL SIZE AND THE COMPUTATION TIME FOR EACH PREDICTION ON MICROCONTROLLER BETWEEN SVM AND RSVM.

Classifier	N_{SV}	Model Size	Computation Time
SVM	30964	1.5 MB	557367 μs
RSVM	3793	180 KB	15348 μs

speeds: 0.53 m/s, 0.86 m/s, 1.11 m/s. The classification results are compared to the SVM as stated in Table I. The performance metrics of the classification for each gait phase are calculated by gathering all walking data from all subjects. It can be observed that as the number of SVs is reduced as a result of k–means clustering, sensitivity, specificity, and accuracy of RSVM are decreased by roughly 4%, 2%, and 5%, respectively when compared to SVM. However, the model size of RSVM and the computation time running on the microcontroller for each prediction of RSVM are significantly reduced, as shown in Table II.

The SVM has a total of 30964 SVs, and it takes up about 1.5 MB of memory on the embedded system. After reducing the model size of the SVM, the number of the SVs for RSVM is 3793. The RSVM only takes up 180 KB of memory on the embedded system. To evaluate the computational efficiency of the real-time algorithm implementation, a timing experiment for 100 gait phase predictions running on the microcontroller is conducted. The average execution time for each phase prediction is $557367\,\mu$s, and $15348\,\mu$s for SVM and RSVM, respectively. Hence, with a small degradation in the classification performance when comparing RSVM to SVM, the number of SVs and the model size are reduced by about 88%, and the computation speed for each prediction on the microcontroller is increased by 36 times. RSVM's model size and computational speed satisfy our requirements for real-time gait phase prediction on the microcontroller, i.e., the model size is less than 200 KB, and the prediction frequency is greater than 25 Hz.

In Fig. 4, an example of gait phase detection results for offline RSVM is illustrated, and recall that the true phases in Fig. 4(a)–(b) are labeled by the rule-based algorithm. Though RSVM shows good classification performance for the gait phase detection, as shown in Fig. 4(a), there are systematic misclassifications for $\mathrm{P}_2 \rightarrow \mathrm{P}_3$, and $\mathrm{P}_3 \rightarrow \mathrm{P}_1$ transitions. This phenomenon is discussed in Section III-D. To address this issue, the FSM is incorporated into the RSVM, which leads to the results illustrated in Fig. 4(b). Fig. 4 provides a clear visualization of the comparison results between the classification performance of RSVM incorporated with and without FSM. We can see that using the FSM can eliminate almost all the misclassifications of RSVM. The detailed performance metrics of RSVM–FSM based on testing data from the first dataset are listed in Table III. Compared to the RSVM in Table I, RSVM–FSM shows improved performance for all three metrics. Therefore, the RSVM–FSM model is employed for real-time gait phase detection on the microcontroller.

TABLE III
THE OFFLINE CLASSIFICATION PERFORMANCE METRICS OF RSVM–FSM BASED ON THE TESTING DATASET COLLECTED FROM TREADMILL.

Gait Phase	Sensitivity	Specificity	Accuracy
	RSVM–FSM	RSVM–FSM	RSVM–FSM
P_1	91.22%	97.14%	92.35%
P_2	93.05%	94.83%	
P_3	90.27%	96.23%	
Overall	92.12%	96.23%	

B. Inter-subject Real-Time Performance for Gait Phase Detection

Attributed to the high computational efficiency of the real-time RSVM-FSM algorithm, the entire process within one sampling interval on the microcontroller, including sensor data reading, gait phase prediction, and prediction results logging, can be completed within 20 ms. As a result, the frequency for the real-time gait phase prediction and prediction result logging is set to 33.33 Hz. While as stated in Section II-C, raw sensor data are still logged at 100 Hz.

To assess the inter-subject real-time performance of RSVM-FSM for gait phase detection, the recorded real-time prediction results are compared to the benchmark, i.e., labels identified by the rule-based algorithm applied to the raw sensor data. Fig. 5 presents the confusion matrix of the real-time RSVM-FSM classified results by gathering all walking data from twenty subjects. Table IV summarizes the three performance metrics calculated from gathering all walking data from all twenty subjects' walking trials and three different walking speeds. We can see that the real-time RSVM-FSM algorithm achieves promising gait phase detection results with the sensitivity, specificity, and accuracy values of 91.74%, 95.79%, 91.55%, respectively. Fig. 6 presents an intuitive comparison of the gait phases detected by real-time RSVM-FSM and benchmarks that are obtained by the offline rule-based algorithm. In Fig. 8, a

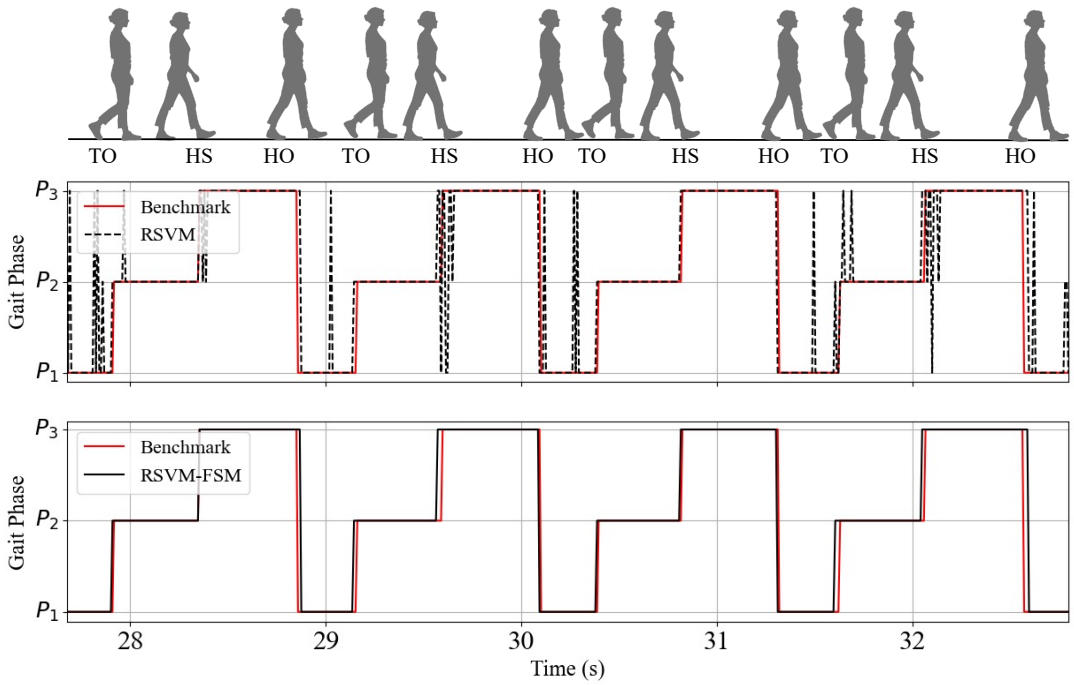

Fig. 4. Performance illustration of RSVM and RSVM–FSM for offline gait phase detection. (a) Gait phases detection results from RSVM. (b) Gait phases detection results from RSVM–FSM.

violin plot of walking speed for offline model training and real-time model testing is presented. Each dot represents a walking bout with an average speed at the dot.

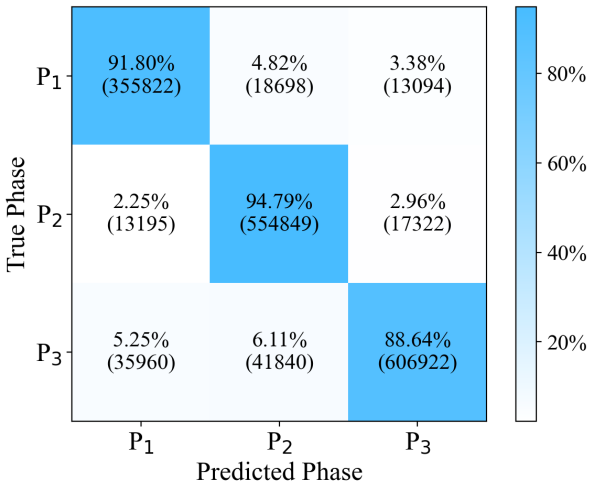

Fig. 5. Confusion matrix of real-time RSVM-FSM results for gait phase classification of twenty subjects.

To compare the performance of RSVM-FSM and SVM-FSM, the raw sensor data are fed into the offline SVM-FSM classifier. The classified gait phases of SVM-FSM are compared to the benchmark. The three performance metrics are calculated by averaging the walking bouts over all twenty subjects and three different walking speeds for both RSVM-FSM and SVM-FSM. The box plot of the performance metrics of different subjects is shown in Fig. 7. Comparing to offline performance of SVM-FSM with an accuracy of 91.55% $\pm$ 2.23%, a sensitivity of 91.16% $\pm$ 2.15%, a specificity of

TABLE IV
THE REAL-TIME CLASSIFICATION PERFORMANCE METRICS OF RSVM–FSM FOR LONG-TERM FREE WALKING.

Gait Phase	Sensitivity	Specificity	Accuracy
	RSVM–FSM	RSVM–FSM	RSVM–FSM
P_1	91.80%	96.13%	91.55%
P_2	94.79%	94.35%	
P_3	88.64%	96.88%	
Overall	91.74%	95.79%	

95.73% $\pm$ 0.98%, the performance of real-time RSVM-FSM presents an insignificant degradation, which are 91.39% $\pm$ 1.67%, 91.59% $\pm$ 1.59%, 95.67% $\pm$ 0.78% for accuracy, sensitivity, and specificity, respectively.

To evaluate the robustness of the real-time RSVM-FSM algorithm, the classification performance on different walking speeds, cadences, and stride lengths are shown in Fig. 9. In the left subplot of Fig. 9, the three performance metrics for three different walking speed levels are presented. The walking speed v_w for each walking bout of every subject is calculated from the raw sensor data. After ordering all walking bouts' walking speeds from slowest to fastest, the walking bouts are divided into three categories, i.e., low, medium, high walking speeds. The low walking speed category contains the walking bouts whose walking speed is slower than the first 3-quantile ($v_w \leq 1.21\,\mathrm{m/s}$) of the ordered list. The medium walking speed category contains the walking bouts whose walking speed is between the first and the second

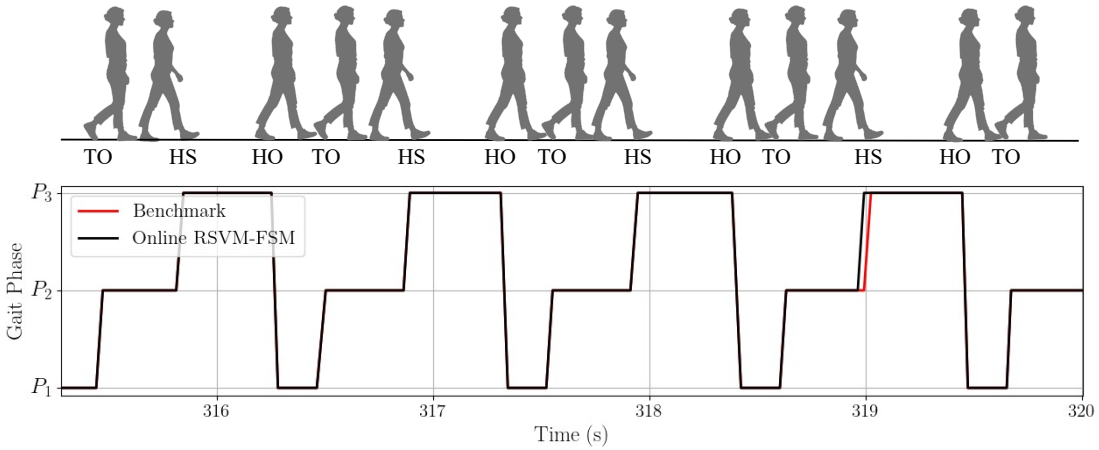

Fig. 6. An intuitive comparison of the gait phases detected by real-time RSVM-FSM and the benchmark.

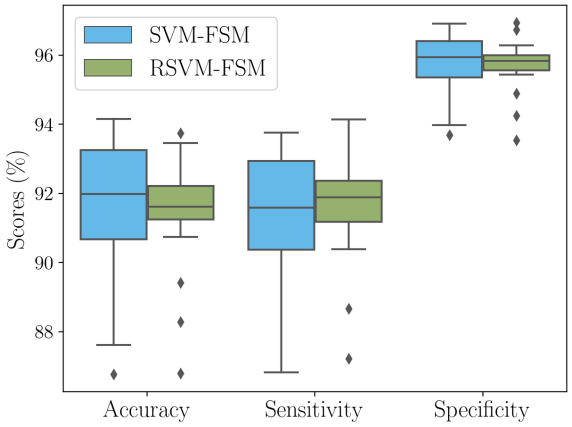

Fig. 7. Performance comparison of real-time RSVM-FSM and offline SVM-FSM gait phase detection for twenty subjects.

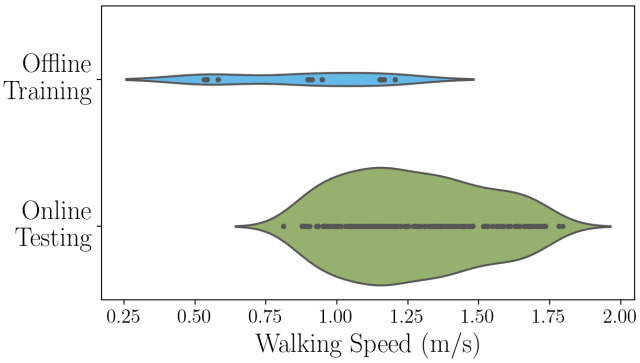

Fig. 8. The violin plot for walking speed distribution of offline training and real-time testing. Each dot represents one walking bout. The width represents the density of the data size. Wider areas of the violin plot constitute a higher density of data taking a given walking speed, whereas the thinner areas correspond to a lower density.

3-quantile ($1.21\,\mathrm{m/s} < v_w \leq 1.43\,\mathrm{m/s}$). The high walking speed category contains the remaining walking bouts ($v_w > 1.43\,\mathrm{m/s}$). All three quantiles of cadence and stride length are defined in the same way. The cadence quantile bounds are found to be $104.84\,\mathrm{SPM}$ (steps per minute) and $115.10\,\mathrm{SPM}$. The stirde-length qunatile bounds are found to be $1.37\,\mathrm{m}$ and $1.54\,\mathrm{m}$.

C. Discussion

When comparing the classification performance of SVM and RSVM as shown in Table I, SVM shows a slightly better performance in the gait phase detection than RSVM. This is because the RSVM hyperplane is constructed only using potential SV data points. Due to the lower density of boundary data points in smaller training datasets, it may not be enough to provide sufficient separation information of the training dataset, resulting in a slight performance drop in RSVM classification. In addition, this slight performance drop of RSVM is also due to the significant reduction of the number of SVs, which is decreased by 88% in total. While comparing to the SVM, the execution time of RSVM has reduced more than 36 times as the result of a considerable reduction in the number of SVs, as shown in Table II.

From Fig. 4(b) we can see that many illogical gait phase transitions are misclassified, especially near the transitions from one phase to the next. This is due to the fact that despite gait being a periodic motion, the RSVM processes the continuous gait signal in sections, neglecting the implicit information carried by the gyroscope as a whole, and thus ignoring the temporal sequence of gait phases. The developed FSM provides the RSVM classifier with a logical temporal sequence. After applying the FSM, the misclassified illogical gait phase transitions in Fig. 4(b) are corrected into a regular sequence of gait phase as shown in Fig. 4(c). These results demonstrate the advantages and the effectiveness of the developed FSM, which leads to a solid temporal sequence of sub-gait phases.

The real-time algorithm testing environment is long-term unsupervised walking on the street, as opposed to the supervised walking environment for model training data, in which subjects walked on a treadmill in a laboratory. In the daily living environment, the human gait is very dynamic, involving irregular gait speeds, fluctuating surface, and varying surface inclinations. Besides, as shown in Fig. 8, the training data for the algorithm contains discrete walking speeds lying around

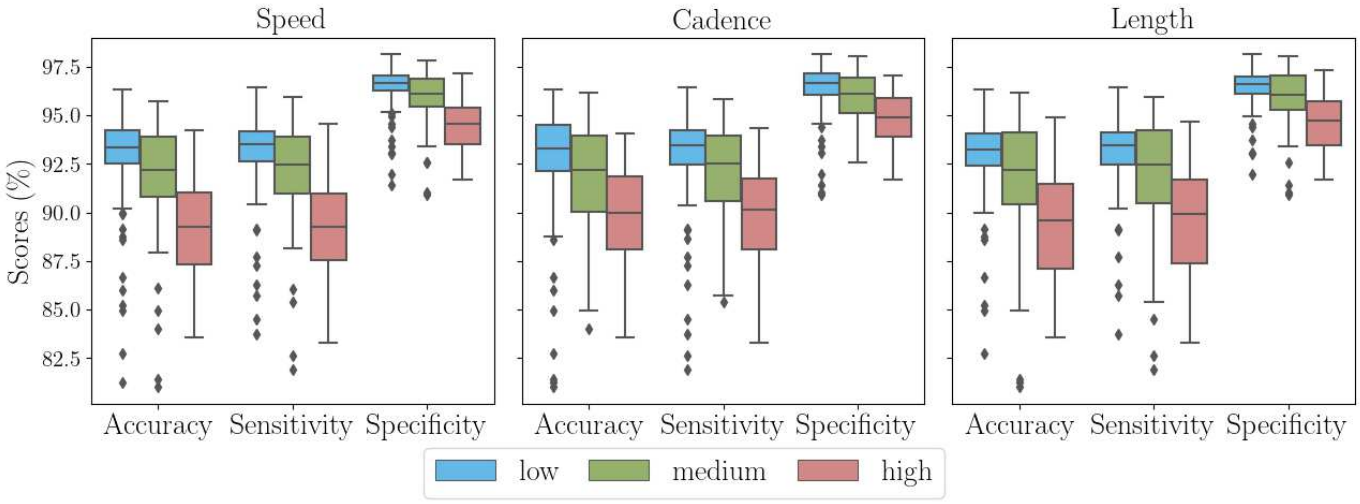

Fig. 9. The RSVM-FSM's performance of real-time gait phase detection in relation to various gait parameters, i.e., gait speed, cadence, stride length.

0.53 m/s, 0.86 m/s, and 1.11 m/s. However, the walking speeds for real-time testing range from 0.81 m/s to 1.80 m/s. There is only a small overlap between the walking speed of training and testing data. Compared to the prediction results of RSVM-FSM shown in Table III, which is intra-subject supervised walking, there is only a minor performance degradation of 0.42%, 0.46%, 0.84% in the inter-subject unsupervised walking real-time prediction results shown in Table IV, for sensitivity, specificity, and accuracy. These results exhibit that the proposed RSVM-FSM algorithm's promising performance is consistent with different walking subjects, changing walking speeds, and varying walking environments, including less controlled walking scenarios. It demonstrates the robustness and generalizability of the classifier for both intra-subject and inter-subject walking, and both controlled and uncontrolled walking conditions.

The prediction results in Fig. 9 demonstrate a promising generalization performance of the real-time RSVM-FSM algorithm for varying gait parameters, i.e., gait speed, cadence, and stride length. As shown in the left subplot of Fig. 9, the RSVM-FSM classifier performs well at all walking speeds. The minor performance degradation when the walking speeds increase may be due to the fact that the walking speeds in the training data are slow, ranging from $0.53\,\mathrm{m/s}$ to $1.11\,\mathrm{m/s}$, whereas the walking speeds in the medium and high category for real-time testing are higher than $1.21\,\mathrm{m/s}$. In terms of the generalization of the proposed algorithm regarding varied walking cadences, the classifier also shows consistent performance as demonstrated in the middle subplot of Fig. 9. It can be observed that the performance distributions among the three cadence groups are overlapping. No obvious performance decrease is observed when increasing the walking cadence. A similar performance level is obtained for classifying gait phases among different stride length groups, indicating that the proposed RSVM-FSM is robust to the various stride length, as well as the gait cadence. The slight performance drop of the high stride length group could be explained by the fact that the classifier is constructed by the data collected from the treadmill, where stride length is limited by the walking surface of the belt.

In addition, the algorithm only uses three signals to classify the gait phases, all of which come from a single gyroscope only. This reduces the amount of information used to characterize the gait cycle compared to the studies that use both gyroscope and accelerometers [22], and thus saves the energy and the cost of the wearable device. Moreover, no prior calibration is required for the sensor data, which is a significant improvement compared to the studies that need additional calibration procedures [32].

The real-time gait phase detection algorithm proposed in this paper has been implemented in our wearable gait analysis system *Nushu* and triggers a vibration signal to the foot based on gait phases through a vibratory motor. Based on this real-time algorithm, three vibration modes have been implemented and tested, i.e., vibrate 500 ms at HO and HS time points, vibrate during the swing phase, and vibrate during the stance phase. This could be further extended in other real-time rehabilitation applications, such as triggering precise and accurate stimulation sequences during walking.

The limitation of this study is that we assume the input activity data only contains walking rather than other activities, as the algorithm only classifies the walking data into predefined three gait phases. Hence, currently, the RSVM-FSM algorithm presented herein will output a gait phase even when the subject is not walking. The authors are currently working on a new algorithm for human activity recognition to filter out the walking data while rejecting other activity data in real-time. The new algorithm will detect the initiation and termination of walking among different human activities. By combining the two algorithms, it will be possible to monitor human activities and provide instant haptic feedback in long-term daily life.

Besides, this gait phase detection algorithm triggered vibration function has also been tested for neurological patients in clinics as a pilot study. The vibration gives haptic feedback to patients during their walking. The preliminary response from

patients has been positive. While further validation study for pathological gait is still needed.

V. Conclusions

In this paper, we propose a real-time RSVM-FSM-based algorithm for reliable gait phase detection on memory- and computation-limited microcontrollers. We use a cascaded k-means clustering approach to reduce the model size of an SVM classifier, i.e., the number of SVs and computation time, so that the model can run on the embedded environment in real-time. This RSVM-FSM-based algorithm was trained using treadmill walking data from four healthy subjects and validated in real-time with unsupervised walking that was performed in daily living environments by twenty healthy subjects. With a negligible degradation of performance, the model size is reduced by 88%, from 1.5 MB to 180 KB, and the computational speed on the microcontroller (ESP32) is increased by 36 times. The algorithm shows a promising real-time performance, with a total accuracy of 91.51%, a sensitivity of 91.70%, and a specificity of 95.77%. It does not require any prior calibration for the sensor data and presents a generalization performance for different walking speeds, cadences, and stride lengths. We have demonstrated that this algorithm can be used for real-time rehabilitation applications, such as haptic feedback systems.

Appendix I
Preliminary

A. The Model Size of Support Vector Machine

This section explains the relationship between the training samples and the number of the SVs, i.e., the model size. SVM is a binary classification algorithm that maps feature vectors into a high-dimensional feature space from which a linear separation hyperplane can be constructed [33]. Consider a two-class dataset $\mathcal{D}$ composed of N feature-class pairs $(\boldsymbol{x}_i, y_i)$ i.e.,

$$\mathcal{D} = \left\{(\boldsymbol{x}_i, y_i) | 1 \leq i \leq N, \boldsymbol{x}_i \in \mathbb{R}^M, y_i \in \{-1, 1\}\right\},$$

where $\boldsymbol{x}_i$ is the feature vector, y_i is its corresponding class label, M is the dimension of the feature vector. A soft-margin SVM classifier, $\mathrm{SVM} : \boldsymbol{x}_i \mapsto y_i$,

$$\mathrm{SVM}(\boldsymbol{x}_i; \boldsymbol{w}, \phi, b) = \mathrm{sign}\left(\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) + b\right) \tag{4}$$

can be constructed (trained) to separate those N training vectors into two classes, where $\boldsymbol{w}$ is the normal vector to the hyperplane, b is a bias term, $\frac{b}{\|\boldsymbol{w}\|}$ can be interpreted as the perpendicular distance from the hyperplane to the origin. This is done by solving the following quadratic optimization problem in feature space [33]:

$$\begin{aligned} &\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to } y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i \\ &\xi_i \geq 0,\ i \in \{1, \dots, N\} \end{aligned} \tag{5}$$

where $\boldsymbol{\xi} = [\xi_1, \dots, \xi_N]$ is the slack variable which allows some training samples violate the margin constraints with a distance ξ_i to their correct margin boundaries, C is the regularization parameter to control the trade–off between the training error and generalization, and $\phi(\cdot) : \mathcal{X} \to \Phi$ is a mapping function, which maps data points from the feature space $\mathcal{X}$ into a high dimensional space Φ.

To solve (5), one can transform the problem in its dual form using Lagrange multipliers for the constraints. Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_N]$, and $\boldsymbol{\beta} = [\beta_1, \beta_2, \dots, \beta_N]$ are introduced to form the Lagrange function:

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}, \boldsymbol{\xi}, \boldsymbol{\beta}) = &\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &+ \sum_{i=1}^{N}\alpha_i\left(1 - \xi_i - y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b)\right) \\ &- \sum_{i=1}^{N}\beta_i\xi_i \end{aligned} \tag{6}$$

By setting the partial derivatives of the Lagrange function (6) with respect to primal variables $\boldsymbol{w}$, b, $\boldsymbol{\xi}$ equal to zero, the primal constraint optimization problem (5) is solved in its dual formulation:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j) \\ &\text{subject to } 0 \leq \alpha_i \leq C,\ \forall i = 1, \dots, N, \\ &\sum_{i=1}^{N}\alpha_i y_i = 0. \end{aligned} \tag{7}$$

The Karush–Kuhn–Tucker (KKT) conditions [34] are as follows:

$$\alpha_i \geq 0,\ \beta_i \geq 0, \tag{8}$$

$$y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b\right) - 1 + \xi_i \geq 0, \tag{9}$$

$$\alpha_i\left(y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b\right) - 1 + \xi_i\right) = 0, \tag{10}$$

$$\xi_i \geq 0,\ \beta_i\xi_i = 0. \tag{11}$$

The optimization problem is thus transformed into a problem that searches for the optimal coefficient α_i for each input feature vector x_i from the training dataset. The bias term b can be determined from KKT conditions.

To simplify the computation of dot products $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j)$ in (7) between data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the high dimensional space, the kernel function $\kappa(\cdot,\cdot) : \mathbb{R}^M \times \mathbb{R}^M \mapsto \mathbb{R}$ is introduced:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}_j). \tag{12}$$

This kernel function also eliminates the need for the users to construct explicit mappings $\phi(\cdot)$ for input data $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively.

Determined from the partial derivative of the Lagrange function (6) with respect to $\boldsymbol{w}$ that

$$\boldsymbol{w} = \sum_{i=1}^{N}\alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}), \tag{13}$$

the primal decision function for an input feature $\boldsymbol{x}$ is thus given by:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) + b\right). \quad (14)$$

As evident from the KKT condition (10), any training feature-class pair $(\boldsymbol{x}_i, y_i)$ should satisfy either $\alpha_i = 0$ or $y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b\right) = 1 - \xi_i$. If $\alpha_i = 0$, this training sample does not affect the decision function $f(\boldsymbol{x})$, else if $\alpha_i > 0$, it must satisfy $y_i\left(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b\right) = 1 - \xi_i$, which means the sample lies within the soft margin boundary ($0 \leq \xi_i \leq 1$) or on the wrong side of separating hyperplane ($\xi_i > 1$). Only training samples that have positive Lagrange multipliers α_i and thereby contribute to the final decision function of a classifier:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{N_{SV}} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) + b\right). \quad (15)$$

We call these samples support vectors (SVs) and hence in (15) N_{SV} is the number of the SVs.

For multiclass classifications, the multiclass problem is broken down into multiple binary classification scenarios, using the same principle as the binary classification. In this paper, the one-vs-rest strategy is employed for multiclass classification.

B. K–Means Clustering

As seen from (14) and (15), the number of SVs increases with the size of the data set. A large number of SVs will lead to a slow computation speed in prediction phase as the SVM speed relies on the number of the SVs. Therefore, for a real-time SVM on a source-limited embedded system, the reduction of the number of the SVs is necessary. K-means clustering is used to reduce the number of SVs in this paper attributed to its minor effect in accuracy. This section details the background of k-means clustering.

K–means clustering is an unsupervised learning algorithm that partitions a data set into k non–overlapping clusters such that the intra-cluster similarity is high while the inter-cluster similarity is low [35]. The number of clusters k is chosen by the model designer. Given a data set of feature vectors $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, k–means clusters are constructed to group those feature vectors into pre–defined k number of clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ by minimizing the within-cluster variances as follows:

$$\arg\min_{\mathcal{C}} \mathcal{J} = \arg\min_{\mathcal{C}} \sum_{i=1}^{k} \sum_{\boldsymbol{x}\in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2. \quad (16)$$

$\boldsymbol{\mu}_i$ is the mean vector of cluster C_i, which is defined as:

$$\boldsymbol{\mu_i} = \frac{1}{|C_i|} \sum_{\boldsymbol{x}\in C_i} \boldsymbol{x}, \quad (17)$$

where $|C_i|$ is the cardinality of the set C_i.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Research and Innovation program under the Marie Sklodowska-Curie grant agreement No. 764977.

CONFLICT OF INTEREST

B.J.N. and S.P. are shareholders in Magnes AG.

REFERENCES

[1] S. Fritz and M. Lusardi, "White paper:"walking speed: the sixth vital sign"," *Journal of geriatric physical therapy*, vol. 32, no. 2, pp. 2–5, 2009.

[2] S. Lord, B. Galna, and L. Rochester, "Moving forward on gait measurement: toward a more refined approach," *Movement Disorders*, vol. 28, no. 11, pp. 1534–1543, 2013.

[3] P. Gray and K. Hildebrand, "Fall risk factors in parkinson's disease," *Journal of Neuroscience Nursing*, vol. 32, no. 4, p. 222, 2000.

[4] H. Jacobi, T.-K. Hauser, P. Giunti, C. Globas, P. Bauer, T. Schmitz-Hübsch, L. Baliko, A. Filla, C. Mariotti, M. Rakowicz *et al.*, "Spinocerebellar ataxia types 1, 2, 3 and 6: the clinical spectrum of ataxia and morphometric brainstem and cerebellar findings," *The Cerebellum*, vol. 11, no. 1, pp. 155–166, 2012.

[5] L. Nürnberger, C. Klein, S. Baudrexel, J. Roggendorf, M. Hildner, S. Chen, J.-S. Kang, R. Hilker, and J. Hagenah, "Ultrasound-based motion analysis demonstrates bilateral arm hypokinesia during gait in heterozygous pink1 mutation carriers," *Movement Disorders*, vol. 30, no. 3, pp. 386–392, 2015.

[6] M. Pistacchi, M. Gioulis, F. Sanson, E. De Giovannini, G. Filippi, F. Rossetto, and S. Z. Marsala, "Gait analysis and clinical correlations in early parkinson's disease," *Functional neurology*, vol. 32, no. 1, p. 28, 2017.

[7] H. Prasanth, M. Caban, U. Keller, G. Courtine, A. Ijspeert, H. Vallery, and J. Von Zitzewitz, "Wearable sensor-based real-time gait detection: a systematic review," *Sensors*, vol. 21, no. 8, p. 2727, 2021.

[8] M. R. Popovic, T. Keller, I. Papas, V. Dietz, and M. Morari, "Surface-stimulation technology for grasping and walking neuroprostheses," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 1, pp. 82–93, 2001.

[9] R. Kobetic and E. Marsolais, "Synthesis of paraplegic gait with multichannel functional neuromuscular stimulation," *IEEE Transactions on Rehabilitation Engineering*, vol. 2, no. 2, pp. 66–79, 1994.

[10] W. Liberson, "Functional electrotherapy: stimulation of the peroneal nerve synchronized with the swing phase of the gait of hemiplegic patients," *Arch Phys Med*, vol. 42, pp. 101–105, 1961.

[11] F. B. Wagner, J.-B. Mignardot, C. G. Le Goff-Mignardot, R. Demesmaeker, S. Komi, M. Capogrosso, A. Rowald, I. Seáñez, M. Caban, E. Pirondini *et al.*, "Targeted neurotechnology restores walking in humans with spinal cord injury," *Nature*, vol. 563, no. 7729, pp. 65–71, 2018.

[12] S. Harkema, Y. Gerasimenko, J. Hodes, J. Burdick, C. Angeli, Y. Chen, C. Ferreira, A. Willhite, E. Rejc, R. G. Grossman *et al.*, "Effect of epidural stimulation of the lumbosacral spinal cord on voluntary movement, standing, and assisted stepping after motor complete paraplegia: a case study," *The Lancet*, vol. 377, no. 9781, pp. 1938–1947, 2011.

[13] R. Caldas, M. Mundt, W. Potthast, F. B. de Lima Neto, and B. Markert, "A systematic review of gait analysis methods based on inertial sensors and adaptive algorithms," *Gait & posture*, vol. 57, pp. 204–210, 2017.

[14] H. Zhao, Z. Wang, S. Qiu, J. Wang, F. Xu, Z. Wang, and Y. Shen, "Adaptive gait detection based on foot-mounted inertial sensors and multi-sensor fusion," *Information Fusion*, vol. 52, pp. 157–166, 2019.

[15] D. Gouwanda and A. A. Gopalai, "A robust real-time gait event detection using wireless gyroscope and its application on normal and altered gaits," *Medical engineering & physics*, vol. 37, no. 2, pp. 219–225, 2015.

[16] A. Mannini, V. Genovese, and A. M. Sabatini, "Online decoding of hidden markov models for gait event detection using foot-mounted gyroscopes," *IEEE journal of biomedical and health informatics*, vol. 18, no. 4, pp. 1122–1130, 2013.

[17] W. Chen, Y. Xu, J. Wang, and J. Zhang, "Kinematic analysis of human gait based on wearable sensor system for gait rehabilitation," *Journal of Medical and Biological Engineering*, vol. 36, no. 6, pp. 843–856, 2016.

[18] H. T. T. Vu, F. Gomez, P. Cherelle, D. Lefeber, A. Nowé, and B. Vanderborght, "Ed-fnn: A new deep learning algorithm to detect percentage of the gait cycle for powered prostheses," *Sensors*, vol. 18, no. 7, p. 2389, 2018.

[19] E. Warmerdam, J. M. Hausdorff, A. Atrsaei, Y. Zhou, A. Mirelman, K. Aminian, A. J. Espay, C. Hansen, L. J. Evers, A. Keller *et al.*, "Long-term unsupervised mobility assessment in movement disorders," *The Lancet Neurology*, vol. 19, no. 5, pp. 462–470, 2020.

[20] S. Khandelwal and N. Wickström, "Evaluation of the performance of accelerometer-based gait event detection algorithms in different real-world scenarios using the marea gait database," *Gait & posture*, vol. 51, pp. 84–90, 2017.
[21] J. Wu, K. Kuruvithadam, A. Schaer, R. Stoneham, G. Chatzipirpiridis, C. A. Easthope, G. Barry, J. Martin, S. Pané, B. J. Nelson *et al.*, "An intelligent in-shoe system for gait monitoring and analysis with optimized sampling and real-time visualization capabilities," *Sensors*, vol. 21, no. 8, p. 2869, 2021.
[22] J. Wu, H. Maurenbrecher, A. Schaer, B. Becsek, C. A. Easthope, G. Chatzipirpiridis, O. Ergeneman, S. Pane, and B. J. Nelson, "Human Gait-labeling Uncertainty and a Hybrid Model for Gait Segmentation," 11 2021. [Online]. Available: https://www.techrxiv.org/articles/preprint/Human_Gait-labeling_Uncertainty_and_a_Hybrid_Model_for_Gait_Segmentation/16924102
[23] J. A. DeLisa, *Gait analysis in the science of rehabilitation*. Diane Publishing, 1998, vol. 2.
[24] J. Rueterbories, E. G. Spaich, B. Larsen, and O. K. Andersen, "Methods for gait event detection and analysis in ambulatory systems," *Medical engineering & physics*, vol. 32, no. 6, pp. 545–552, 2010.
[25] T. O'Haver and T. Begley, "Signal-to-noise ratio in higher order derivative spectrometry," *Analytical Chemistry*, vol. 53, no. 12, pp. 1876–1878, 1981.
[26] E. K. Antonsson and R. W. Mann, "The frequency content of gait," *Journal of biomechanics*, vol. 18, no. 1, pp. 39–47, 1985.
[27] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.
[28] I. Steinwart, "Sparseness of support vector machines," *Journal of Machine Learning Research*, vol. 4, no. Nov, pp. 1071–1105, 2003.
[29] J. Wu, B. Becsek, A. Schaer, H. Maurenbrecher, G. Chatzipirpiridis, O. Ergeneman, S. Pané, H. Torun, and B. Nelson, "Real-time gait phase detection on wearable devices for unsupervised gait," 2022.
[30] M. J. Orr *et al.*, "Introduction to radial basis function networks," 1996.
[31] C.-W. Hsu, C.-C. Chang, C.-J. Lin *et al.*, "A practical guide to support vector classification," 2003.
[32] M. Goršič, R. Kamnik, L. Ambrožič, N. Vitiello, D. Lefeber, G. Pasquini, and M. Munih, "Online phase detection using wearable sensors for walking with a robotic prosthesis," *Sensors*, vol. 14, no. 2, pp. 2776–2794, 2014.
[33] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[34] N. Cristianini, J. Shawe-Taylor *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.
[35] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.