Journal of
open research software

van Gent, P, et al. 2019 Analysing Noisy Driver Physiology Real-Time Using Off-the-Shelf Sensors: Heart Rate Analysis Software from the Taking the Fast Lane Project. *Journal of Open Research Software,* 7: 32. DOI: https://doi.org/10.5334/jors.241

SOFTWARE METAPAPER

Analysing Noisy Driver Physiology Real-Time Using Off-the-Shelf Sensors: Heart Rate Analysis Software from the Taking the Fast Lane Project

Paul van Gent[1], Haneen Farah[1], Nicole van Nes[2] and Bart van Arem[1]

[1] Department of Transport & Planning, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, NL

[2] Dutch Institute for Road Safety Research (SWOV), The Hague, NL

Corresponding author: Paul van Gent (P.vanGent@tudelft.nl)

This paper describes the functioning and development of HeartPy: a heart rate analysis toolkit designed for photoplethysmogram (PPG) data. Most openly available algorithms focus on electrocardiogram (ECG) data, which has very different signal properties and morphology, creating a problem with analysis. ECG-based algorithms generally don't function well on PPG data, especially noisy PPG data collected in experimental studies. To counter this, we developed HeartPy to be a noise-resistant algorithm that handles PPG data well. It has been implemented in Python and C. Arduino IDE sketches for popular boards (Arduino, Teensy) are available to enable data collection as well. This provides both pc-based and wearable implementations of the software, which allows rapid reuse by researchers looking for a validated heart rate analysis toolkit for use in human factors studies.

Keywords: heart rate analysis; photoplethysmogram; python; arduino; PPG
Funding statement: Part of the software has been developed within the "Taking the Fast Lane" project, funded by NWO TTW[1], project number 13771.

1. Introduction

In the field of transportation research one of the main goals is to get to a point where zero traffic fatalities occur [1]. The rise of smart in-car systems makes reaching this goal possible. For example, systems exist that automatically take over safety critical tasks of drivers when needed, such as autonomous emergency braking systems. When a driver fails to spot a hazard on the road in front of the vehicle, these systems intervene to avert a collision. However, these are reactions to outside events, whereas another improvement to traffic safety can be made by changing the way drivers and their cars interact. Human error, attentional failures, or driver states that are incongruent with the driving task (fatigue, overload) are a major cause of traffic accidents [2]. Sensing when a driver is underloaded, overloaded, distracted or tired can improve safety by enabling dynamic adjustments in the way in-car systems interact with the driver. For example, by timing when navigational or other in-vehicle information systems relay information to the driver, or by adapting the content of their messages to match the current driver state, safety can be improved [3].

Human factors research into driver states is an active field. To estimate driver states, physiological measures are often taken together with performance measures [4]. Heart rate data is collected in many studies, as it is sensitive to changes in workload [5–7] and general driver state [8]. However, capturing and analysing heart rate in the -often noisy- conditions of either a simulator or an on-road setting can be difficult or costly [4]. The recent advances in wearable technology and open hardware platforms, such as the Arduino[2] and Raspberry Pi,[3] create new possibilities for collecting and analysing physiological data at low cost, given that validated algorithms exist to analyse and process it. In this paper we describe the development of such an algorithm named HeartPy, which we validated as described in [9].

1.1. Overview of Project Context

Within the 'Taking the Fast Lane' project,[4] we are working towards lane-specific advice generation. One possible application is the reduction of congestion by using driver advices to distribute traffic across the available lanes more efficiently. This means advising drivers on where to drive. When interacting with a driver, the timing of messages to the driver is crucial not only for safety but also for the effectiveness of the advices [3]. Advising on those moments that workload is low and the driver can accommodate the advice, gives a higher chance of the driver following the advice. However, this means the driver state needs to be known. For this reason, a driver state monitoring system is being developed.

To facilitate the on-line capture and analysis of physiological data, a noise-resistant heart rate collection and analysis toolkit was developed. We created an easy to use, open source analysis toolkit that could handle the collection and analysis of data from available low-cost photoplethysmogram (PPG) sensors, as we could not find an openly available, robust analysis toolkit for this. In this paper we present its development. The toolkit has been used in two simulator studies that modelled driver workload [10, 11], as well as in a study looking at the cognitive effects of monitoring automated driving in different conditions [12, 13].

1.2. Similar software

Similar heart rate analysis software exists. These can be divided into commercial and open source variants.

Psychlab and Biopac both offer lab-based solutions including both hardware and software for psychophysiological and medical research. Both offer validated devices and algorithms that have been widely cited. These are, however, not openly available and come at substantial cost.

Kubios offers both a paid software version for HRV analysis, as well as a free version. The free version lacks peak detection functionality and instead requires pre-detected RR-intervals from which to calculate HRV measures.

Physionet [14] is a large open medical database of Electrocardiogram (ECG) recordings. They also implement WFDB, a software package to retrieve data from their online database and perform waveform analysis. Python bindings are available. WFDB is feature rich, however uses a custom data format and can be technical to implement. It also doesn't handle PPG data well.

HRVAS is a heart rate analysis package for Matlab. It offers many features and is openly available. It, however, still requires Matlab or an older version of its runtime to run, which is not always available. It suffers from the same setback of not handling PPG data well.

The presently discussed heart rate analysis toolkit aims to add to the current body of available software by providing a toolkit for both desktop written in Python, and for (embedded) open hardware platforms written in C. The toolkit focuses on Photoplethysmogram (PPG) recordings but handles ECG data as well.

2. Implementation and architecture

HeartPy has been developed to be sensor-independent, with the use of embedded systems with low computational resources in mind. We have tried to create a fast method of extracting heart beats, that is resistant to types of noise frequently occurring when recording ECG or PPG in field-based studies with low-cost sensors. A Python version is available for PC-based research, as well as limited implementations for several popular Arduino and ARM-based boards that assist in data collection, pre-processing and offer methods of real-time analysis.

2.1. Measuring the heart rate signal

Two often used ways of measuring the heart rate are the electrocardiogram (ECG) and the Photoplethysmogram (PPG). The ECG measures the electrical activations that lead to the contraction of the heart muscle, using electrodes attached to the body, usually at the chest. The PPG uses a small optical sensor in conjunction with a light source to measure the discoloration of the skin as blood perfuses through it after each heartbeat.

Most notably in the ECG is the QRS-complex (**Figure 1a, I–III**), which represents the electrical activation that leads to the ventricles contracting, expelling blood from the heart muscle. The R-peak is the point of largest amplitude in the signal. When extracting heart beats, these peaks are marked in the ECG. Advantages of the ECG are that it provides a good signal/noise ratio, and the R-peak that is of interest generally has a large amplitude compared to the surrounding data points (**Figure 1c**). The main disadvantage is that the measurement of the ECG is invasive in terms of human factors studies.[5] It requires the attachment of wired electrodes to the chest of the participant, which can interfere with experimental tasks such as driving. This can be undesirable because it can influence participant behaviour, or create potentially dangerous situations for example when driving.

The PPG measures the discoloration of the skin as blood perfuses through the capillaries and arteries after each heartbeat. The signal consists of the systolic peak (**Figure 1-b, I**), dicrotic notch (II), and the diastolic peak (III). When extracting heart beats, the systolic peaks (I) are used. PPG sensors offer a less invasive way of measuring heart rate data, which is one of their main advantages. Usually the sensors are placed at the fingertip, earlobe, or on the wrist using a bracelet. Contactless camera-based systems have recently been demonstrated [15–17]. These offer non-intrusive ways of acquiring the PPG signal. PPG signals have the disadvantages of showing more noise, large amplitude variations, and the morphology of the peaks displays broader variation (**Figure 2b, c**). This complicates analysis of the signal, especially when using software designed for ECG, which the available open source tools generally are. The toolkit described in this paper aims to provide an efficient means of analysing noisy PPG signals.

2.2. Heart Rate and Heart Rate Variability Measures

Analysis of the heart signal is split into heart rate (HR) and heart rate variability (HRV) measures. The heart rate is a simple measure of the heart period, expressed in the beats per minute and the inter-beat interval. Heart rate variability measures describe how the heart rate signal varies over time, and can be divided into time-domain measures and frequency-domain measures [18, 19].

When extracting heart beats from a signal, a marker is chosen that can reliably be detected at the same position on all heartbeat complexes in the signal. In the ECG the R-peak is often taken (**Figure 1a-II**), in the PPG signal the maximum of the Systolic wave is usually marked (**Figure 1b-I**). Common measures expressing the HR found in the literature are the beats per minute (BPM) and the mean inter-beat interval (IBI). HRV is expressed in the median absolute deviation of intervals between heart beats (MAD), the standard deviation of intervals between heart beats (SDNN), the root mean square of successive

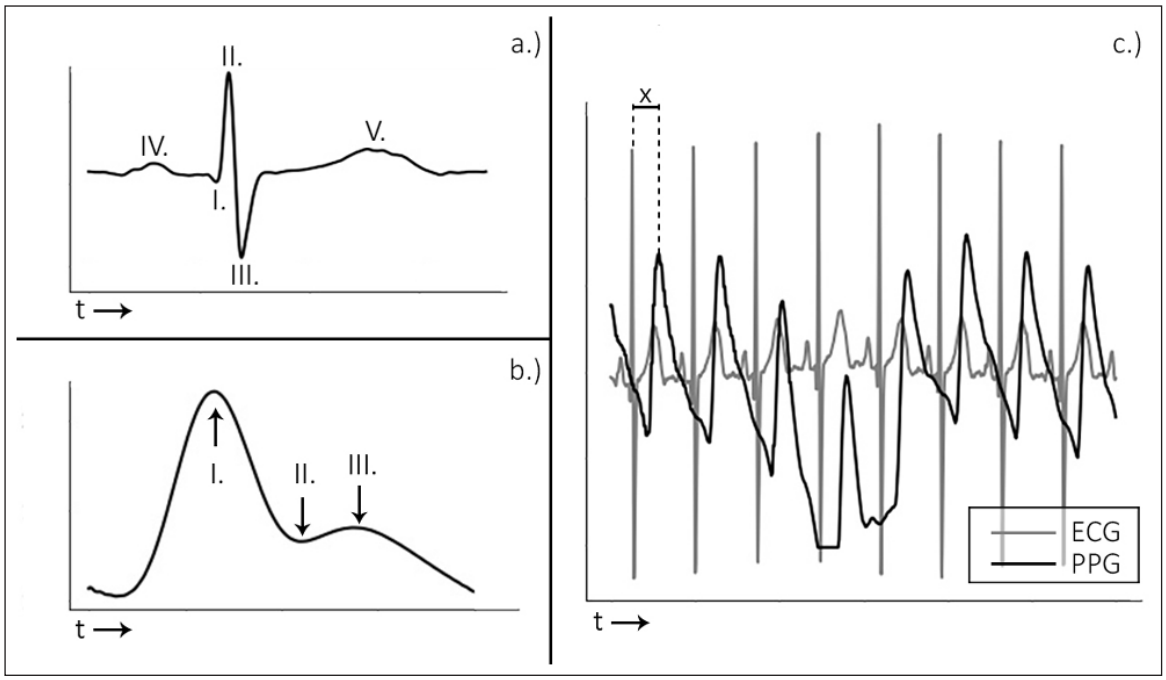

Figure 1: a. and **b.** display the ECG and PPG waveform morphology, respectively. The ECG is divided into distinct waves **(a, I–V)**, of which the R-wave **(a, II)** is used for heart beat extraction. With the PPG wave, the systolic peak **(b, I)** is used. The plot in c. shows the relationship between ECG and PPG signals.

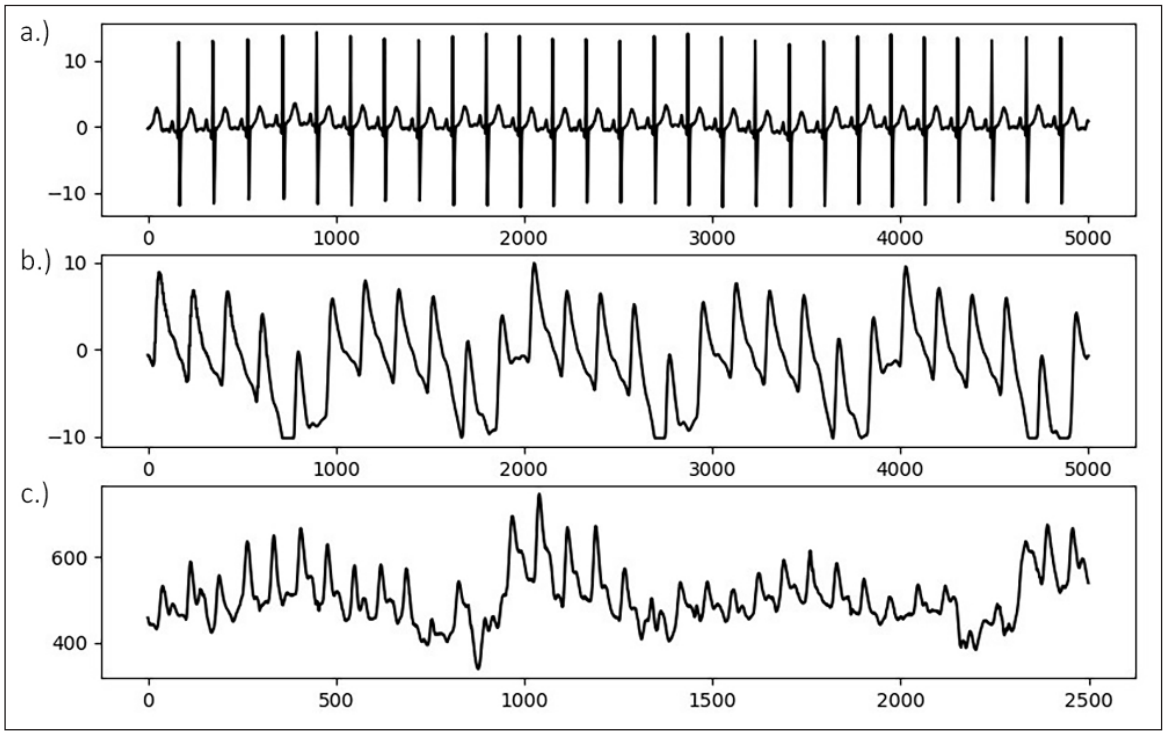

Figure 2: The ECG signal **(a.)** shows a strong QRS complex together with little amplitude variation. The PPG signal measured simultaneously while the patient is at rest in a hospital bed **(b.)** shows some amplitude variation but relatively stable morphology. When measuring PPG in a driving simulator using low-cost sensors **(c.)**, strong amplitude and waveform morphology variation is visible.

differences between neighbouring heart beat intervals (RMSSD), the standard deviation of successive differences between neighbouring heart beat intervals (SDSD), and the proportion of differences between successive heart beats greater than 50 ms and 20 ms (pNN50, pNN20, resp.). HRV can also be expressed in the frequency domain, where two frequency bands are usually included: low frequency (LF, 0,04–0,15 Hz), which is related to short-term blood pressure variation [20], and high frequency (HF, 0,16–0,5 Hz), which is a reflection of breathing rate [19].

2.3. Analysis Overview

This section describes the architecture of the algorithm and gives an overview of how the heart rate signal is processed and analysed.

2.3.1. Pre-processing

The pre-processing options available are peak enhancement, FIR filtering, and outlier detection. The peak enhancement function attempts to normalise the amplitude, then increases R-peak amplitude relative to the rest of the signal. A Butterworth filter implementation is available to remove high frequency noise. Outlier detection on the raw signal is implemented based on a modified Hampel Filter [21] with a window of half the sampling rate. By default, only the peak enhancement is performed. Details are discussed in the repository's documentation [22].

2.3.2. Peak detection

The peak detection phase attempts to accommodate amplitude variation and morphology changes of the PPG complexes by using an adaptive peak detection threshold (**Figure 3, III**), followed by outlier detection and rejection. To identify heartbeats, a moving average is calculated using a window of 0.75 seconds on both sides of each data point. The first and last 0.75 seconds of the signal are populated with the signal's mean, no moving average is generated for these sections. Regions of interest (ROI) are marked between two points of intersection where the signal amplitude is larger than the moving average (**Figure 3, I–II**), which is a standard way of detecting peaks. R-peaks are marked at the maximum of each ROI.

A special case arises when the signal clips, which can happen for example when a sensor has constraints on the range of the signal it can measure, or when digitising an analog signal. The algorithm has clipping detection for R-peaks and will attempt to reconstruct the waveform by spline interpolation whenever an R-peak displays clipping. To interpolate, 100 ms of data before clipping onset and 100 ms of data after clipping end is used. An example of the process is shown in **Figure 3-IV**.

During the peak detection phase, the amplitude of the calculated threshold is adjusted stepwise. To find the best fit, the standard deviation between successive differences (SDSD, see also 2.2) is minimised. The instantaneous heart rate (BPM) is computed and evaluated in tandem with the SDSD. This represents a fast method of approximating the optimal peak detection threshold by exploiting the relative regularity of the heart rate signal. As shown in **Figure 4**, missing one R-peak (III.) already leads to a substantial increase in SDSD compared to the optimal fit (II.). Marking incorrect R-peaks also leads to an increase in SDSD (I.). The lowest SDSD value that is not zero, in combination with a likely BPM value, is selected as the best fit. The BPM must lie within a predetermined range (default: 40 <= BPM <= 180, range settable by user).

2.3.3. Error detection

Due to the variable PPG waveform morphology, it is possible that after the initial peak fitting phase incorrectly marked R-peaks remain. Motion artefacts may be another cause of detection error. A correction is performed by thresholding the sequence of RR-intervals. R-peaks are considered low confidence if the interval created between two adjacent R-peaks deviates by more than 30% of the mean RR-interval of the analysed segment (**Figure 5**). The threshold is adaptive based on the current segment with a minimum value of 300 ms. We've found this to be a good approximation for incorrect detections. If any peaks are considered incorrect detections, the array of RR-values is recomputed to only contain intervals between two high confidence R-peaks.

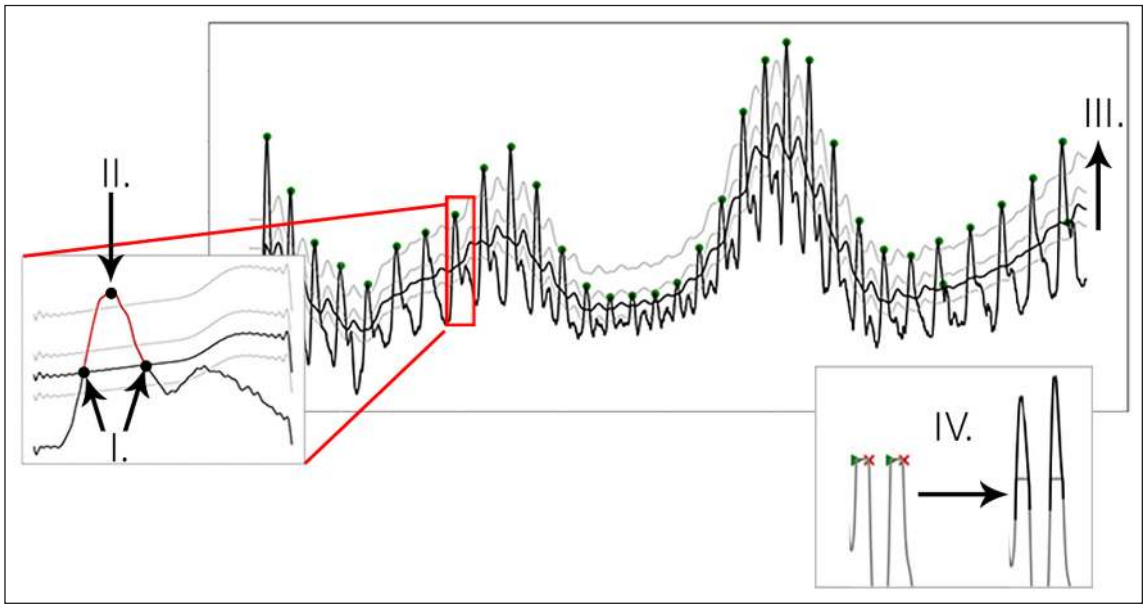

Figure 3: Figure showing the process of peak extraction. A moving average is used as an intersection threshold **(II)**. Candidate peaks are marked at the maximum between intersections **(III)**. The moving average is adjusted stepwise to compensate for varying PPG waveform morphology **(I)**. (**IV)**. shows the detection of the onset and end of clipping, and the result after interpolating the clipping segment.

An optional error detection pass is available. Using the method, the signal is segmented into n-peak sections and each segment evaluated. Segments are marked low quality if more than a predetermined percentage of peaks are marked low confidence (default n = 10, rejection percentage = 30%). We found that this pattern of short segments displaying multiple rejected peaks, are often indicative of periods of poor signal/noise ratio or signal loss, such as displayed in **Figure 6**. By eliminating these short periods from the analysis, the output measures remain reliable because only RR-intervals resulting from analysable segments are used in their calculation.

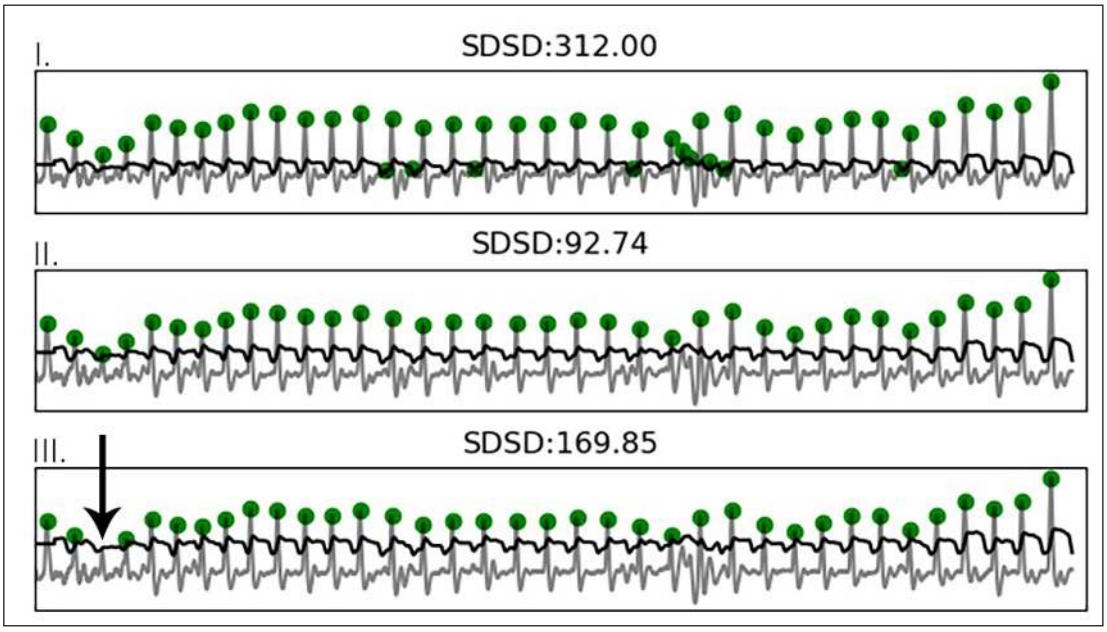

Figure 4: Image showing how the dynamic threshold is fitted using SDSD. The last image **(III.)** shows that even missing a single beat will lead to a large increase in SDSD compared to the optimal fitting. BPM is also taken into account when fitting.

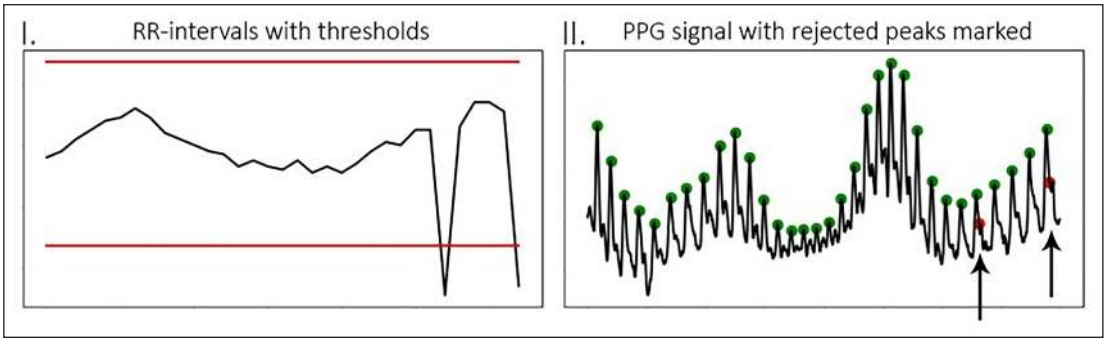

Figure 5: The plotted RR-intervals with thresholds **(I.)**, and the resulting rejected peaks **(II.)**.

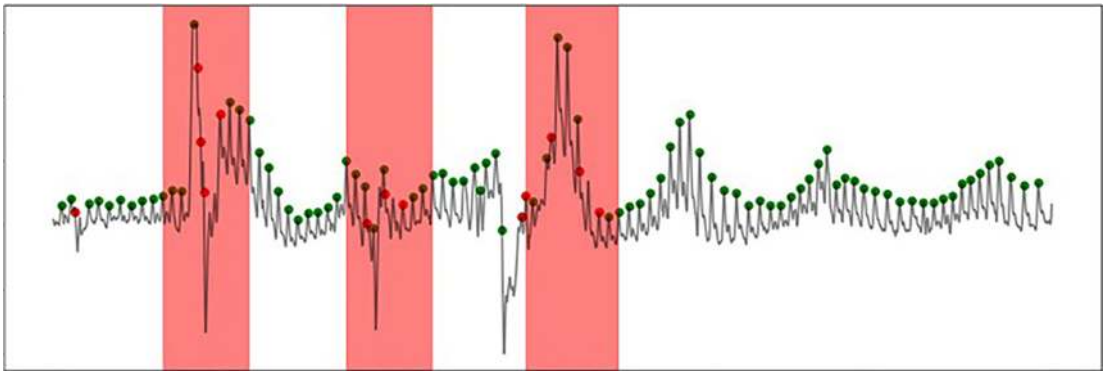

Figure 6: Plot from PPG dataset with low-confidence sections marked. These are ignored in the computation of output measures.

The heart rate analysis package was implemented in both Python and embedded C. The following two sections describe both implementations as well as their requirements, dependencies and availability.

2.4. Python implementation

Python is a flexible programming language that is well suited for scientific use [23]. During development the reliance on external dependencies was minimised. The package uses the following external packages:

- Numpy is used to handle the data, numerical computations, and the Fast Fourier Transform. For these purposes Numpy is much faster than the standard Python interpreter.
- Scipy is used for various filtering and interpolation tasks.
- Matplotlib is included to plot the results of the analysis if requested by the user.

The implementation of the functions had readability as the main aim. Pep-8 conventions were followed in code styling and function design. A quickstart and background information can be found in the documentation [24] and the code together with detailed Jupyter tutorial notebooks can be found on the repository [25].

2.5. Embedded C implementations

Several C implementations have been developed to facilitate data collection and analysis in lab-based and field-based studies that utilise wearable technology. Hardware interrupt timers are used to ensure a precise sampling rate is maintained. Most implementations contain a double switching buffer to collect the sensor data. As one of the buffers fills up, logging switches to the secondary buffer and the content of the first buffer is processed and stored. This ensures logging without interruption.

The repository contains Arduino IDE sketches for several popular boards. Wiring diagrams, and suggested PCB (printed circuit board) design files for various (wearable) applications are in development. The implementations available are briefly discussed below.

2.5.1. Data Logger

A data logging application is available. Users can set the desired sampling rate they wish to log. Adaptive input scaling is available (on by default), which attempts to normalise amplitude over time. This is especially useful when measuring at locations where the PPG signal is weaker (wrist, neck), or when measuring it on participants, with reduced perfusion, such as those with advanced age or a history of smoking.

2.5.2. Peak Finder

The peak finder implementation analyses the incoming signal real-time for peaks and returns both the peak position and RR-interval created between the current and the previous detected peak. Error detection based on the last 20 RR-intervals, as well as based on various settable parameters is available. See the documentation for more details [24].

2.5.3. Time Series Analysis

The time series analysis implementation is similar to the peak finder implementation, except that it calculates and outputs the time-series measurements of both heart rate and heart rate variability. It tracks detected peaks in time to computer RR-intervals and ignores intervals when there is a missing or rejected peak in between.

2.5.4. Full Implementation

The full implementation contains the HR and HRV online analysis. All the HR and HRV measures mentioned under 2.2 are derived from the signal and stored to an on-board SD card, together with the original signal. Since a full signal period is first collected, several pre-processing steps can be taken to improve signal quality prior to analysis. This makes the full implementation the most noise-resistant of the available versions. The memory and processing requirements, however, are also higher than of the other versions. This makes it less suited for long-term wearable solutions required in naturalistic studies, but very suited for environments where power is available (in-car, driving simulator, lab-based studies, bicycle with power bank) or situations of shorter measurement periods.

3. Quality control

3.1. General Quality Control

The code development was centred around ease-of-use and reusability of functions and methods. Coding best practices were followed [26]. Throughout the development process, cyclomatic complexity (cc) was frequently calculated for all functions using the python Radon package and the Lizard[6] package. Refactoring was applied for functions that had a cc of over 10,[7] to ensure maintainability and readability of the code. Git was used for version control [27] throughout the project.

Means for automatic source code validation and automated testing have been implemented. In the Python implementation, examples are available in the docstrings that double as doctests. Automated continuous integration (CI) testing is implemented through the Travis-CI platform. Code coverage, build status and supported Python versions are displayed as badges on the GitHub repository.

Several end-to-end examples are included in Jupyter notebooks on the repository, detailing how to handle various types of signals with HeartPy. Available examples deal with both good and poor quality PPG and ECG signals from various sources (sensors, electrodes, smartwatch, smart ring). The examples are designed to familiarise new users with the functionality of the package and to highlight possible use cases.

A tutorial series is available [28], detailing the basics behind the Python implementation of the algorithm. Users seeking deeper understanding in the mechanics behind the algorithm can follow these.

3.2. Validation

HeartPy was validated on a dataset collected by PPG sensor from a previous experiment [29]. Heart beats in the dataset were manually annotated to serve as a ground truth. The validation was performed on the set and compared to two popular available open source algorithms. Error

rates showed superior performance of our algorithm on the noisy PPG data in the test set. The full validation is described in [9].

4. Availability

4.1. Operating system

HeartPy has been tested to run on Python 2.7, 3.4, 3.5, 3.6 and 3.7. All updates are automatically built using Travis-CI and tested. Results are dynamically displayed on the repository as badges.

Several Arduino IDE sketch files have been provided as well for different boards. These have been tested on their respective boards, and developed in the Arduino IDE version 1.8.5. They are designed to enable researchers low-cost ways of collecting heart rate data as well.

4.2. Additional system requirements

The Python implementation's data handling happens in NumPy, which ensures efficient RAM usage and fast execution. The size of the input data will determine memory usage, although the requirements are low for most datasets. As an example, we loaded the first data file from the first participant in the PPG validation set included on the GitHub ('pp1_Eind_Som_C.csv'). The file represents 10:52 minutes of data sampled at 100 Hz. The loaded data takes 261 Kb in memory. Temporary containers created during analysis and (pre-)processing take up an additional 593 Kb. This indicates very low resource requirements for most analyses.

Requirements of the embedded hardware conform to RAM and CPU resources available on the SOC's for which the implementation has been designed. More information and absolute values are available in the documentation [22].

4.3. Dependencies

HeartPy is dependent on the Numpy, SciPy and Matplotlib packages. The lowest versions we've tested with HeartPy are NumPy==1.15, SciPy==1.1.0, Matplotlib==2.2.3. These versions allow functionality on Python 2.7.

The Arduino implementations depend on standard modules available in the Arduino IDE. The versions for Arduino and Teensy boards depends on the SDfat module for communication with the SD card for data storage. This module is installed in the Arduino IDE by default.

4.4. List of contributors

Jonathan de Bruin has provided valuable advice and suggestions during development and testing of HeartPy and will remain active in further development.

4.4.1. Software location of Python version

Archive (e.g. institutional repository, general repository) (required – please see instructions on journal website for depositing archive copy of software in a suitable repository)

Name: Zenodo.org
Persistent identifier: https://doi.org/10.5281/zenodo.3407802
Licence: GNU General Public License V3.0
Publisher: Paul van Gent
Version published: V1.2.4
Date published: 13-09-2018

Code repository: GitHub
Name: Python Heart Rate Analysis Toolkit
Identifier: https://github.com/paulvangentcom/heartrate_analysis_python
Licence: GNU General Public License V3.0
Date published: 13-09-2018

4.4.2. Software location of C version

Code repository: GitHub
Name: Arduino Heart Rate Analysis Toolkit
Identifier: https://github.com/paulvangentcom/heartrate_analysis_Arduino
Licence: GNU General Public License V3.0
Date published: 31-07-2018

5. Reuse potential

HeartPy can be used by researchers, makers, and engineers to create applications that make use of (real-time) heart rate data. The toolkit can be used in research settings both in the lab and 'in the wild'. HeartPy handles noise that is typically introduced into heart rate signals when recording outside the lab, and contains many pre-processing options to help clean up poor quality signals. The software has been used in for example lab-based simulator contexts [10, 11], real world driving contexts [12, 13], and as a backend for a pregnancy monitoring tool [29].

Detailed examples are available on the repository and in the documentation on handling different data types that serve to kick-start any new project based on HeartPy. These examples are available on the repository as Jupyter notebooks (https://github.com/paulvangentcom/heartrate_analysis_python/tree/master/examples). The examples cover how to analyse PPG signals from sensors, smartwatches and smart rings (and similar devices), as well as ECG signals ranging from good to very poor quality.

HeartPy is designed to easily be integrated into existing projects. All methods are documented with examples provided, and most can be used in isolation. Throughout the processing pipeline everything of interest is stored in a dict{} object, which can be accessed each step of the analysis. This facilitates integration with other projects by allowing a fine level of control over each step.

We are currently working on incorporating a GUI for use with HeartPy, which will expand the reuse potential further towards researchers without coding experience.

Notes

[1] See https://www.nwo.nl/en/.
[2] See http://www.arduino.cc.
[3] See http://www.raspberrypi.org.
[4] See http://tfl.tudelft.nl/.
[5] Note that the definition of 'invasive' in human factors studies refers to intrusion into the person's privacy, personal space or thoughts. It differs from the medical definition, where 'invasive' indicates that a foreign object intrudes into the body.
[6] See https://github.com/terryyin/lizard.
[7] See http://radon.readthedocs.io/en/latest/api.html#radon.complexity.cc_rank.

Competing Interests

The authors have no competing interests to declare.

References

1. **Belin, M-Å, Tillgren, P** and **Vedung, E** 2012 "Vision Zero – a road safety policy innovation." *Int. J. Inj. Contr. Saf. Promot.*, 19(2): 171–179. DOI: https://doi.org/10.1080/17457300.2011.635213
2. **Kaplan, S, Guvensan, M A, Yavuz, A G** and **Karalurt, Y** 2015 "Driver Behavior Analysis for Safe Driving: A Survey," *IEEE Trans. Intell. Transp. Syst.*, 16(6): 3017–3032. DOI: https://doi.org/10.1109/TITS.2015.2462084
3. **Van Gent, P, Farah, H, Van Nes, N** and **Van Arem, B** 2017 "A Conceptual Model for Persuasive In-Vehicle Technology to Influence Tactical Level Driver Behavior." *Transp. Res. Part F Traffic Psychol. Behav.*
4. **Brookhuis, K A** and **de Waard, D** 2010 "Monitoring drivers' mental workload in driving simulators using physiological measures." *Accid. Anal. Prev.*, 42(3): 898–903. DOI: https://doi.org/10.1016/j.aap.2009.06.001
5. **Mehler, B, Reimer, B, Coughlin, J F** and **Dusek, J A** 2010 "Impact of Incremental Increases in Cognitive Workload on Physiological Arousal and Performance in Young Adult Drivers." *Transp. Res. Rec. J. Transp. Res. Board*, 2138(1): 6–12. DOI: https://doi.org/10.3141/2138-02
6. **Stuiver, A, de Waard, D, Brookhuis, K A, Dijksterhuis, C, Lewis-Evans, B** and **Mulder, L J M** Aug. 2012 "Short-term cardiovascular responses to changing task demands." *Int. J. Psychophysiol.*, 85(2): 153–60. DOI: https://doi.org/10.1016/j.ijpsycho.2012.06.003
7. **Aasman, J, Mulder, G** and **Mulder, L J M** 1987 "Operator effort and the measurement of heart-rate variability." *Hum. Factors*, 29(2): 161–170. DOI: https://doi.org/10.1177/001872088702900204
8. **Danisman, T, Bilasco, I M, Djeraba, C** and **Ihaddadene, N** 2010 "Drowsy driver detection system using eye blink patterns." *2010 Int. Conf. Mach. Web Intell. ICMWI 2010 – Proc.*, pp. 230–233. DOI: https://doi.org/10.1109/ICMWI.2010.5648121
9. **van Gent, P, Farah, H, van Nes, N** and **van Arem, B** 2018 "Heart Rate Analysis for Human Factors: Development and Validation of an Open Source Toolkit for Noisy Naturalistic Heart Rate Data." In: *Proceedings of the 6th HUMANIST Conference*, pp. 173–178.
10. **Van Gent, P, Farah, H, Van Nes, N** and **Van Arem, B** 2017 "Towards Real-Time, Nonintrusive Estimation of Driver Workload: A Simulator Study." In: *Road Safety and Simulation 2017 Conference Proceedings*.
11. **van Gent, P, Melman, T, Farah, H, van Nes, N** and **van Arem, B** 2018 "Multi-Level Driver Workload Prediction Using Machine Learning and Off-The-Shelf Sensors." *Transp. Res. Rec. J. Transp. Res. Board*. DOI: https://doi.org/10.1177/0361198118790372
12. **Stapel, J, Mullakkal-Babu, F A** and **Happee, R** 2017 "Driver Behaviour and Workload in an On-road Automated Vehicle." In: *Proceedings of the RSS2017 Conference*.
13. **Stapel, J, Mullakkal-Babu, F A** and **Happee, R** 2018 "Attentive monitoring of automated driving requires more effort than manual driving." Manuscr. Submitt. Publ.
14. **Goldberger, A L,** et al. 2000 "PhysioBank, PhysioToolkit, and PhysioNet – Components of a New Research Resource for Complex Physiologic Signals." DOI: https://doi.org/10.1161/01.CIR.101.23.e215
15. **Sun, Y, Hu, S, Azorin-Peris, V, Kalawsky, R** and **Greenwald, S** 2012 "Noncontact imaging photoplethysmography to effectively access pulse rate variability." *J. Biomed. Opt.*, 18(6): 061205. DOI: https://doi.org/10.1117/1.JBO.18.6.061205
16. **Lewandowska, M, Ruminsky, J, Kocejko, T** and **Nowak, J** 2011 January "Measuring Pulse Rate with a Webcam – a Non-contact Method for Evaluating Cardiac Activity." In: *Proceedings of the Federated Conference on Computer Science and Information Systems*, pp. 405–410.
17. **Bousefsaf, F, Maaoui, C** and **Pruski, A** 2014 "Remote detection of mental workload changes using cardiac parameters assessed with a low-cost webcam." *Comput. Biol. Med.*, 53: 1–10. DOI: https://doi.org/10.1016/j.compbiomed.2014.07.014
18. **Mehler, B, Reimer, B** and **Wang, Y** 2011 "Comparison of heart rate and heart rate variability indices in distinguishing single task driving and driving under secondary cognitive workload." *Proc. Sixth Int. Driv. Symp. Hum. Factors Driv. Assessment, Training, Veh. Des.*, pp. 590–597. DOI: https://doi.org/10.17077/drivingassessment.1451
19. **Montano, N,** et al. 2009 "Heart rate variability explored in the frequency domain: A tool to investigate the link between heart and behavior." *Neurosci. Biobehav. Rev.*, 33(2): 71–80. DOI: https://doi.org/10.1016/j.neubiorev.2008.07.006
20. **Bernardi, L, Leuzzi, S, Radaelli, A, Passino, C, Johnston, J A** and **Sleight, P** 1994 "Low-frequency spontaneous fluctuations of R-R interval and blood pressure in conscious humans: A baroreceptor or central phenomenon?" *Clin. Sci. (Lond).*, 87(6): 649–654. DOI: https://doi.org/10.1042/cs0870649
21. **Davies, L** and **Gather, U** 1993 "The identification of multiple outliers," *J. Am. Stat. Assoc.*, 88(429): 782–792. DOI: https://doi.org/10.1080/01621459.1993.10476339
22. **van Gent, P** 2018 "Embedded Heart Rate Analysis Toolkit Documentation." [Online]. Available: https://embedded-heart-rate-analysis-toolkit.readthedocs.io.
23. **Oliphant, T E** 2007 "Python for scientific computing." *Comput. Sci. Eng.*, 9(3): 10–20. DOI: https://doi.org/10.1109/MCSE.2007.58
24. **van Gent, P** 2018 "Python Heart Rate Analysis Toolkit Documentation." [Online]. Available: https://python-heart-rate-analysis-toolkit.readthedocs.io.

25. **van Gent, P** 2017 "Python Heart Rate Analysis Toolkit." *GitHub Repository*. [Online]. Available: https://github.com/paulvangentcom/heartrate_analysis_python.
26. **Wilson, G,** et al. 2014 "Best Practices for Scientific Computing." *PLoS Biol.*, 12(1). DOI: https://doi.org/10.1371/journal.pbio.1001745
27. **Loeliger, J** and **McCullough, M** 2012 *Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development.*
28. **van Gent, P** 2016 "Analyzing a Discrete Heart Rate Signal Using Python." [Online]. Available: http://www.paulvangent.com/2016/03/15/analyzing-a-discrete-heart-rate-signal-using-python-part-1/.
29. **Gupta, Y, Kumar, S** and **Mago, V** 2019 "Pregnancy Health Monitoring System based on Biosignal Analysis." *2019 42nd Int. Conf. Telecommun. Signal Process.*, pp. 664–667. DOI: https://doi.org/10.1109/TSP.2019.8769074

How to cite this article: van Gent, P, Farah, H, van Nes, N and van Arem, B 2019 Analysing Noisy Driver Physiology Real-Time Using Off-the-Shelf Sensors: Heart Rate Analysis Software from the Taking the Fast Lane Project. *Journal of Open Research Software,* 7: 32. DOI: https://doi.org/10.5334/jors.241

Submitted: 01 August 2018 **Accepted:** 14 October 2019 **Published:** 29 October 2019

]u[*Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press.

OPEN ACCESS