Estimation of Respiratory Rate from ECG signal in Python programming language

Eduard Žňava[1] [0000-0002-5769-1346], Fedor Lehocki[1,2] [0000-0003-0543-6017], Milan Tyšler[2] [0000-0001-7674-0927], Ana Madevska Bogdanova[3] [0000-0002-0906-3548], Magdalena Kostoska[3] [0000-0002-5594-6739], Oto Masár[4], Marko Spasenović[5] [0000-0002-2173-0972], Silvia Puteková[6] [0000-0002-3022-537X]

[1]Slovak University of Technology in Bratislava, Faculty of Informatics and Information Technologies, Slovakia
xznava@stuba.sk, fedor.lehocki@stuba.sk

[2]Institute of Measurement Science, Slovak Academy of Sciences, Slovakia
umertysl@savba.sk

[3] Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, North Macedonia
ana.madevska.bogdanova@finki.ukim.mk, magdalena.kostoska@finki.ukim.mk

[4]Faculty of Medicine, Comenius University Bratislava, Slovakia
oto.masar@fmed.uniba.sk

[5]Center for Microelectronic, Technologies Institute of Chemistry, Technology and Metallurgy, Belgrade, Serbia
spasenovic@nanosys.ihtm.bg.ac.rs

[6]Faculty of Health Care and Social Work, Trnava University, Slovakia
silvia.putekova@truni.sk

* Corresponding author: Fedor Lehocki, fedor.lehocki@stuba.sk

Abstract. In case of mass casualties, it is necessary to obtain different vital signs including respiratory rate effectively and accurately. The more physiological signals are measured individually - the more time it takes to obtain multiple vital signs. In addition, a lot of technical equipment is needed. Because of that, it is effective to derive multiple vital signs from measurement of one single physiological signal. It is possible to derive respiratory rate from ECG signal. In this paper, we are constructing an appropriate solution based on different methods for extraction of respiratory rate from ECG signal using Python programming language together with suitable Python libraries for data processing. We managed to implement three methods and validate the accuracy of the calculations by Pearson's and Spearman's coefficients of correlation, as well as by root mean square error between of the RR calculated from derived and measured respiration signal. For the best method, we completed the algorithm reaching the coefficients of correlation equal to 0.703 and 0.700. The root mean square error is equal to 1.84 breaths per minute.

Keywords: ECG, PQRST-complex, respiratory rate, signal processing

1 Introduction

Acquisition of different vital signs in case of mass casualties, for example war or natural disasters like earthquake or volcanic eruption, needs to be completed in order to accomplish triage procedure using START triage system [6]. In this procedure, individual casualties are sorted into different classes marked by four colors (green, yellow, red, black). Marking depends on how critical health status of a victim is. The great challenge is timely evaluation of critical changes in health conditions of triaged victims (e.g. from yellow to red). This process needs to be performed quickly and precisely.

However, it is challenging to obtain multiple vital signs in short time since measurement of each physiological signal appropriate for obtaining particular vital sign is time-consuming and it requires a lot of special technical equipment. Because of that, there is an idea to derive certain vital signs from measurement of another physiological signal. With this approach, one single biosensor could be used for obtaining multiple vital signs and detect critical changes of victim's health.

Respiratory rate (RR), i.e., number of breaths per minute, can be obtained using impedance pneumography measuring electrical activity in a chest while breathing. Another technology for measuring the RR is a capnography monitor which measures carbon dioxide levels of each breath [12]. However, this is the exact case of struggling with too much equipment and therefore, these technologies are not suitable for us since this struggling negatively impacts the whole usability of the system.

An electrocardiogram (ECG) is measurement of heart activity based on electric signals evoked by the heart as its parts contract [4]. It is possible to derive the RR from measured ECG signal. In ECG signal, we can see repetition of the PQRST-complex. It is a series of peaks and waves representing depolarization and repolarization of different parts of the heart [13].

There are several methods for extraction of the RR from ECG signal. They differ in what pattern in ECG signal is used to estimate the RR. Therefore, their advantages and drawbacks vary, too. In [1], the authors use Respiratory Sinus Arrhythmia to estimate the RR. This phenomenon causes heart rate variability (HRV). In [2], the authors took another approach when they used Peak Amplitude Variation (PAV) caused by chest movement of the patient while breathing. Finally, according to [3], cubic spline interpolation through all the R-peaks can be performed to derive respiratory signal while capturing the Baseline Wander (BW) phenomenon.

There is also a functionality in HeartPy library which is created to process the ECG signal [5]. This processing includes also calculating the RR. However, Jovanov et al. used this feature to estimate the RR and the results were insufficient, giving the Pearson's coefficient of correlation equal to -0.016 [4]. Eventually, there is a functionality in NeuroKit2 library used for derivation of respiration signal from ECG signal which is a key part in the process of calculating the RR from ECG [7]. However, the author of the documentation mentions that this functionality is not complete and needs further work.

Deriving particular vital sign often requires appropriate data preprocessing, especially different signal processing techniques. Of course, good quality of the data is important since only well-measured data can provide us with correct values.

In this paper, we are implementing different methods for extraction of the RR from measured ECG signal in Python programming language. We are also evaluating our implementation and discussing its weak points.

In section II, we are presenting our solution in Python, especially the libraries used, signal processing techniques performed, dataset used for testing the solution and steps of the whole algorithm of calculating the RR. Results are presented in section III and discussed in section IV. Conclusion is made in section V.

2 Methodology

We have implemented three methods mentioned above. In each of them, we are performing cubic spline interpolation through appropriate values to obtain respiration signal (see Table 1).

The method	**Values used for cubic spline interpolation**
HRV	Distances of every two subsequent R-peaks
PAV	Amplitudes of R-peaks (calculated as a difference between signal values of the R-peak and the S-peak)
BW	Signal values of the R-peaks

Table 1. The values which cubic spline interpolation is performed through depending on the particular method.

We used Python programming language and suitable Python libraries and modules for the implementation, especially NeuroKit2 [7] and SciPy [10] for processing the ECG signal, Pandas [11] and NumPy [8] for manipulation with the data and Matplotlib [9] for data visualization.

The dataset used for testing our solution is the well-known Fantasia Database [14, 15]. The dataset consists of 40 recordings of ECG and respiration signal measured simultaneously in supine resting position. The sampling rate is 250 Hz. From each of the recordings, we took first 60 minutes and split them into 1-minute-long sections, getting 2400 sections in total.

2.1 Signal processing

Before running individual methods, we applied a second order bandpass Butterworth filter [16] on it in order to reduce noise. In all the methods, we set the highcut of 60 Hz and except of the BW method, we also set lowcut of 1 Hz. Originally, we followed Sarkar et al. [1] when choosing the boundaries with highcut of 47 Hz. However, we did not want to have the R-peaks in the ECG signal cut so much, so we increased the highcut to 60 Hz. We did not apply the lowcut for the BW method, because the baseline wander which we want to capture is present in the lowest frequencies of the signal. After running the methods, we used 0.1 Hz and 1 Hz boundaries for filtering derived respiration signal. It means that in all the methods, we expect RR values from 6 to 60 breaths per minute. With these boundaries, we expect not only standard values of the RR, but also abnormal ones which is necessary in the cases of mass casualties since an injured person may breathe very intensively.

To calculate the RR from the derived respiration signal, we use a special functionality in the NeuroKit2 library in which cross-correlations between the changes in the respiration signal with a group of sinusoids of different frequencies are calculated to obtain the principal frequency of the signal [7].

To summarize the process, the common steps for all the methods are following:

Step 1: we read the ECG signal from an appropriate file.

Step 2: the signal is filtered using second order bandpass Butterworth filter in order to reduce noise.

Step 3: respiration signal is derived from the ECG signal by the method currently being used for calculating the RR.

Step 4: the respiration signal is filtered by second order bandpass Butterworth filter with lowcut of 0.1 Hz and highcut of 1 Hz (6 to 60 breaths per minute).

Step 5: the RR is calculated using the cross-correlation method implemented in NeuroKit2 library.

We calculate the RR values from the measured respiration signal in the same way as from the derived respiration signal. We filter the measured respiration signal with the same boundaries (0.1 and 1 Hz) and calculate the RR values using the cross-correlation method.

3 Preliminary results

Now that we have calculated the RR values from both derived and measured respiration signals, we are able to validate the accuracy of the implementation. To do so, we calculated Pearson's and Spearman's coefficients of correlation and root mean square error (RMSE) between the RR values calculated from the derived respiration signal and the ones calculated from the measured respiration signal. Results of these calculations are displayed in Table 2. The HRV method reaches the highest correlation coefficients, but on the other hand, the BW method provided us with the lowest RMSE equal to 1.684 breaths per minute. The PAV method lags a little behind having both coefficients of correlation lower than 0.5 and RMSE higher than 2 breaths per minute. The correlation diagram of the RR values calculated from the measured and the derived respiration signal for the BW method is displayed in Fig. 1.

When constructing the final algorithm, we will use the BW method because of the lowest RMSE which is the most important for us when considering accuracy of the calculation. Moreover, it is a little faster method than the HRV method.

Method	**Pearson's coefficient**	**Spearman's coefficient**	**RMSE**
HRV	0.615	0.601	1.949
PAV	0.445	0.464	2.018
BW	0.593	0.581	1.684

Table 2. Pearson's and Spearman's coefficient of correlation and root mean square error between the RR values calculated from the measured and the derived respiration signal by each of the implemented methods.

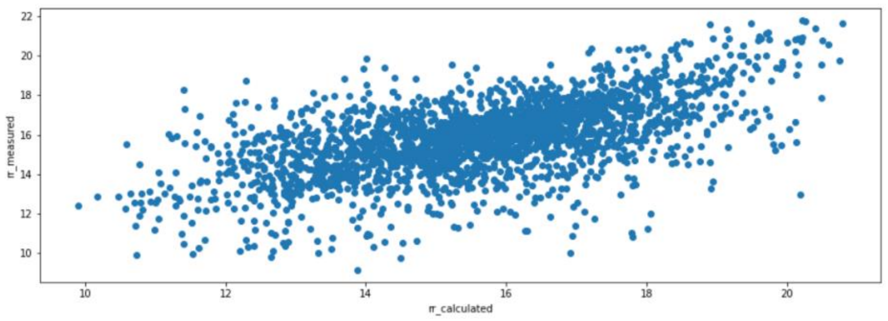

Fig. 1. The correlation diagram of the RR values calculated from the measured and the derived respiration signal for the BW method (Pearson's coefficient of correlation equal to 0.703 and Spearman's coefficient of correlation equal to 0.700).

4 Final algorithm

As we mentioned, we will use the BW method in the final algorithm. However, there are two weak points which we should care about before finalizing it.

Firstly, a 1-minute-long section is quite long when we consider that we want to calculate the RR in real time. To get more actual values, we created 30-seconds-long sections from the first 60 minutes of the recordings. Every next section starts 5 seconds after the start of the previous section. With this approach, we managed to obtain 28534 sections in total which is even better for more relevant validation of the algorithm.

Secondly, although the BW is simple and fast, the current algorithm takes quite long time because of the cross-correlation method used for calculation of the RR from the derived respiration signal. As we mentioned, this method compares the signal with a group of sinusoids. It works with a help of a window of specified length and hop size. The final RR value is calculated as an average of all the calculations using the window. By setting appropriate values of the window length and the hop size, we could speed up the calculation. We tested different combinations of values of these parameters, finally setting the window length of 10 seconds and the hop size of 7 samples.

In addition, we made the filter used for filtering the derived respiration signal adaptive at this point. The lowcut and the highcut is set dynamically according to the previous value of the RR. We did this because we don't have to expect sudden huge changes in the RR. For example, if the current RR is equal to 10 breaths per minute, the RR in 5 seconds will not be equal to 60 breaths per minute. The full mechanism is described below. We set the particular values experimentally.

Here is the final algorithm:

Step 1: Read the ECG signal.

Step 2: Take first 30 seconds of the signal.

Step 3: Filter the current section using second order lowpass Butterworth filter with the 60 Hz cutoff.

Step 4: Detect the R-peaks in the current section.

Step 5: Perform cubic spline interpolation through all the detected R-peaks. The respiration signal is derived.

Step 6: Clean the respiration signal with second order bandpass Butterworth filter. If the current section is the first one in the signal, lowcut is equal to 0.1 Hz (6 bpm) and highcut is equal to 1 Hz (60 bpm). Otherwise, the lowcut is the maximum from 0.1 and the previous calculated RR minus 0.16, and the highcut is equal to previous calculated RR plus 0.2.

Step 7: Calculate the RR using the cross-correlation method.

Step 8: Take a next 30-second-long section starting 5 seconds after the start of the current one.

Step 9: If the new section does not exceed the signal, repeat steps 3.-7. for it. Otherwise, finish.

The workflow diagram for the final algorithm is presented in Fig. 2. Validating the final algorithm by mentioned 30-second-long sections, we reach Pearson's coefficient of correlation equal to 0.703, Spearman's coefficient of correlation equal to 0.700 and RMSE of 1.84 breaths per minute. One calculation of the RR takes approximately 0.43 s. We can see that the RMSE is not as low as in the partial results, but the calculation is much faster and the increase of the RMSE is not significant.

5 Discussion

The results of the implementation are much more positive than the results obtained by HeartPy library (Pearson's coefficient of correlation equal to -0.016) [4].

Except of the quite solid accuracy, the algorithm is able to calculate the RR in real time since one calculation of the RR takes approximately 0.43 seconds and the algorithm checks the signal every 5 seconds. It means that it has enough time for the calculation in real time.

Furthermore, the algorithm is able to expect abnormal values of the RR thanks to the adaptive filter used for filtering the derived respiration signal. It is able to react on increase and decrease of the RR and set the lowcut and the highcut of the filter accordingly.

The recordings used for testing the solution were measured in a lying position. In such position, a patient is relaxed and there is no tension in his/her muscles. We assume that this fact helps the algorithm to be quite accurate. In case of mass casualties, the patient may be for example in shock or other undesirable state. It is questionable whether he/she is able to be relaxed enough to obtain appropriate data. The more his/her body moves, the more artifacts are present in the signal and the less accurate the calculation may be. It is important to bear this in mind when deploying the solution.

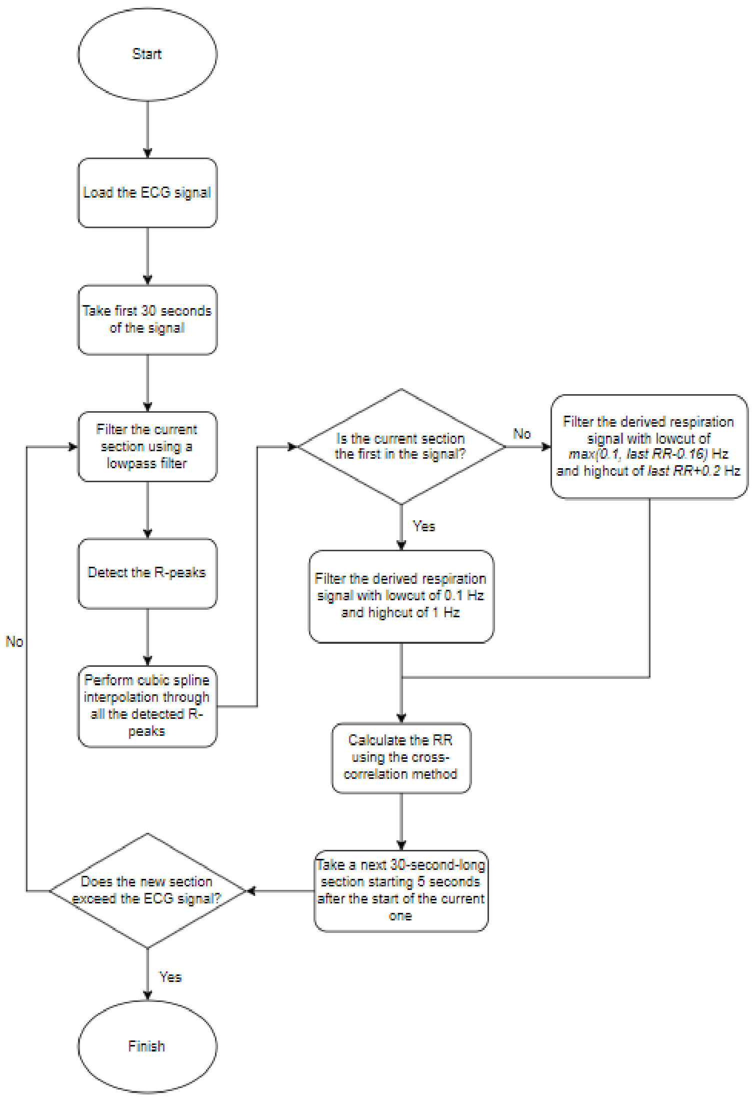
Start
Load the ECG signal
Take first 30 seconds of the signal
Filter the current section using a lowpass filter
Detect the R-peaks
Perform cubic spline interpolation through all the detected R-peaks
Is the current section the first in the signal?
No
Filter the derived respiration signal with lowcut of *max(0.1, last RR-0.16)* Hz and highcut of *last RR+0.2* Hz
Yes
Filter the derived respiration signal with lowcut of 0.1 Hz and highcut of 1 Hz
Calculate the RR using the cross-correlation method
Take a next 30-second-long section starting 5 seconds after the start of the current one
Does the new section exceed the ECG signal?
No
Yes
Finish

Fig. 2. The workflow diagram for the final algorithm of the RR calculation from the ECG signal.

6 Conclusion

We managed to implement three methods for extraction of the RR from ECG signal using Python programming language together with appropriate Python libraries. We validated the accuracy of the calculations by Pearson's and Spearman's coefficients of correlation, and root mean square error between the RR values calculated from the measured and the derived respiration signal. After that, we were able to determine the best method and construct the final algorithm.

The algorithm calculates the RR with root mean square error of 1.84 breaths per minute, reaching 70% accuracy. Its speed and wide range of expected RR values is suitable for cases of mass casualties. This solution is a good start when trying to use only one biosensor when checking the health status of the patient.

The accuracy of the solution may be improved by experimenting with other filters when filtering the ECG signal and the derived respiration signal (for example, the Chebyshev filter can be used). It is also a good idea to experiment with order of the filters.

Acknowledgement

This paper has been written thanks to the support of the Smart Patch for Life Support Systems (NATO project G5825 SP4LIFE).

References

1. Sarkar, S., Bhattacherjee, Pal, S.: Extraction of respiration signal from ECG for respiratory rate estimation. In: Michael Faraday IET International Summit 2015, pp. 336–340. (2015).
2. Helfenbein, E., Firoozabadi, R., Chien, S., Carlson, E., Babaeizadeh, S.: Development of three methods for extracting respiration from the surface ECG: a review. Journal of electrocardiology 47(6), 819–825 (2014).
3. Ruangsuwana, R., Velikic, G., Bocko., M.: Methods to extract respiration information from ECG signals. In: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 570–573. IEEE (2010).
4. Jovanov, S., Ristovski, B., Bogdanova, A., Kostoska, M.: Validation of Data correlation - Heart Rate and Respiratory Rate from ECG in Python. (2021), 13th ICT Innovations Conference 2021, Skopje, North Macedonia.
5. HeartPy documentation, https://python-heart-rate-analysis-toolkit.readthedocs.io/en/latest/, last accessed 2022/03/25.
6. Gursky, Elin A., and Hrečkovski, B.: Handbook for Pandemic and Mass-casualty Planning and Response. Vol. 100. IOS Press, 2012.
7. NeuroKit2 documentation, https://neurokit2.readthedocs.io/en/latest/, last accessed 2022/02/25.
8. NumPy documentation, https://numpy.org/doc/stable/, last accessed 2022/03/16.
9. Matplotlib 3.5.1 documentation, https://matplotlib.org/stable/, last accessed 2022/03/16.
10. SciPy documentation, https://docs.scipy.org/doc/scipy/, last accessed 2022/03/16.
11. Pandas documentation, https://pandas.pydata.org/docs/.

12. Wheatley, I.: Respiratory rate 3: how to take an accurate measurement. Nursing Times 114(7), 21-22 (2018).
13. Blahút, P.: EKG & Arytmológia, https://www.techmed.sk/ekg-a-arytmologia-kniha/, last accessed 2022/04/27.
14. Goldberger, A. L. et al.: PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. In: circulation 101.23 (2000), e215–e220.
15. Iyengar, N., Peng, C. K., Morin, R., Goldberger, A. L., Lipsitz, L. A.: Age-related alterations in the fractal scaling of cardiac interbeat interval dynamics. Am J Physiol. 1996 Oct;271(4 Pt 2):R1078-84. doi: 10.1152/ajpregu.1996.271.4.R1078. PMID: 8898003.
16. Hussin, Siti Farah, Gauri Birasamy, and Zunainah Hamid.: Design of butterworth band-pass filter. Politeknik & Kolej Komuniti Journal of Engineering and Technology 1.1 (2016).