AUTOMATED ELECTROCARDIOGRAM INTERPRETATION FOR THE DETECTION OF POSTOPERATIVE JUNCTIONAL ECTOPIC TACHYCARDIA AT THE PEDIATRIC INTENSIVE CARE UNIT

TM30004 MSc Thesis
Gini Raaijmakers

Automated electrocardiogram interpretation for the detection of postoperative junctional ectopic tachycardia at the pediatric intensive care unit

TM30004 MSc Thesis
Gini Raaijmakers, 4551540
Sophia's Children Hospital, Erasmus Medical Center

March 20, 2024

Thesis in partial fulfillment for the joint degree of Master in Science in

Technical Medicine

Leiden University | Delft University of Technology | Erasmus University Rotterdam

Supervisors:

Dr. Ing. Nico Bruining
Dr. Marc Sylva
Drs. Eris van Twist

Thesis committee members:

Chair/technical supervisor: Dr. Ing. Nico Bruining
Medical supervisor: Dr. Marc Sylva
Daily supervisor: Drs. Eris van Twist
Independent member: Dr. Enno van der Velde
Independent member: Drs. Ulrike Krämer

An electronic version of this thesis is available at:
`http://repository.tudelft.nl`

Abstract

Background Postoperative junctional ectopic tachycardia (JET) is an arrhythmia associated with increased morbidity and mortality rates in children with congenital heart disease. Developing an automated detection algorithm could aid in early identification and timely treatment of JET.

Methods A retrospective study was conducted using monitor electrocardiogram (ECG) data of pediatric patients who experienced JET during their admission to the pediatric intensive care unit. A manual decision tree was developed that aimed to differentiate between JET and sinus rhythm based on distinctive characteristics. These features were derived using signal analysis on both two-dimensional vectorcardiograms and ECG data. For the latter, ECG metrics were detected in a fictive lead that was created in the direction with the highest amplitudes. Metrics were identified within adaptive intervals that were dependent on ECG morphology rather than relying on fixed time intervals.

Results A classification performance was achieved with a sensitivity of 96.3%, specificity of 71.4%, positive predictive value (PPV) of 86.7% and an accuracy of 87.8%. R peaks, Q peaks, S peaks, T peaks and P waves were detected with an accuracy of respectively 99.9%, 95.7%, 89.7%, 98.1% and 54.8%. The computational time of the classification of 41 minutes of data was 4 minutes and 48 seconds.

Conclusion A manual decision tree algorithm for JET detection was developed, using signal analysis for feature extraction based on JET characteristics. This method with a low computational time and a high sensitivity and PPV holds potential for clinical application as a bedside tool. Implementing this proposed algorithm would allow for treatment in an earlier phase, thereby potentially reducing JET associated morbidity and mortality rates.

Contents

List of abbreviations

AV atrioventricular
ECG electrocardiogram
HR heart rate
JET junctional ectopic tachycardia
PICU pediatric intensive care unit
PPV positive predictive value
SA sinoatrial
SD standard deviation
SR sinus rhythm
VCG vectorcardiogram

1 Introduction

Junctional ectopic tachycardia (JET) is an arrhythmia that is either congenital or occurs following congenital heart disease surgery. [1, 2] During JET, rapid electrical impulses originate from the atrioventricular (AV) junction, leading to tachycardia with AV dissociation. [2–4] This tachycardia in combination with suboptimal synchronization of the atria and ventricles can cause the heart to generate cardiac output ineffectively, which may induce hemodynamic instability. [5–8] Moreover, JET is associated with an increased risk of morbidity and mortality. [1, 7] Patients with JET usually require a longer admission to the pediatric intensive care unit (PICU) and are often dependent on cardiovascular support and mechanical ventilation for an extended period of time. [4] Mortality rates of patients developing JET after cardiac surgery are as high as 14%, highlighting the importance of effective treatment. [1]

Whereas the congenital variant is rare (less than 1% of pediatric arrhythmias), postoperative JET is considered as the 'most frequent hemodynamically significant tachycardia in the postoperative setting'. [1, 7, 8] After cardiac surgery, 5-11% of the pediatric patients develop JET, mostly within 24 hours. [1, 2, 4, 9, 10] The exact mechanism of JET development remains unknown, but it is hypothesized that JET may be triggered by mechanical trauma, direct tissue damage or hemorrhages around the AV node and His bundle during surgery. [2, 11] Several risk factors have been identified, including young age, the duration of cardiopulmonary bypass, aortic cross-clamp time, preoperative electrolyte imbalances, the use of inotropes, development of fever, and the specific type of surgery performed (e.g. Tetralogy of Fallot or AV canal defect correction). [1, 3, 6, 7, 11–15]

Although JET is typically self-limiting, various methods are available to restore hemodynamic stability. [6, 7] Treatment is focused on the reduction of automaticity. [15] Cooling of the patient is a critical first step because scientific evidence supports that hypothermia suppresses automaticity and reduces tachycardia. [8] Other essential measures include providing sedation and minimising the use of exogenous catecholamines, as catecholamines are known to increase both heart rate (HR) and automaticity. [1, 3, 10, 13, 16] Administering magnesium sulphate is recommended to stabilise the membrane potential, which reduces automaticity. [16] When this therapy is not effective, antiarrhythmic drugs can be added to the treatment. Possible options are amiodarone and ivabradine. [2, 3, 17, 18] Despite these efforts, treating JET remains challenging, as not all patients respond to these therapies. [2, 4]

Delayed treatment increases the likelihood of an extended period of hemodynamic instability, leading to a higher risk of complications. [19] Thus, early recognition of JET is of great importance.

However, early identification of JET in clinical practice is challenging for multiple reasons. For medical professionals it is impractical to continuously observe the monitor. Moreover, even if constant observation of the monitor were possible, subtle changes in the electrocardiogram (ECG) are easy to miss on visual inspection. Yet with the implementation of an automated detection algorithm, it could be possible to identify JET at an earlier stage. Such an algorithm can tirelessly analyse the ECG signal and enables activation of an alert immediately upon detection of the arrhythmia. An algorithm may enhance the ability to observe subtle changes in the signal. An additional benefit of a JET detection algorithm would be the ability to assess the duration of JET episodes over an unlimited amount of time, which is not achievable manually. In this way, medication effects can be monitored by analysing the frequency and duration of JET episodes.

For these reasons, the aim of this study is to create an algorithm capable of detecting JET based on bedside ECG monitoring at the PICU, using signal analysis to derive relevant characteristics.

2 Background

A normal heart rhythm originates in the sinoatrial (SA) node, with electrical activation of the atrium representing the P wave on the ECG, as shown in Figure 2a. Subsequently, as the ventricles are activated, the QRS complex is created. Finally, the repolarization phase corresponds to the T wave.
In JET, accelerated automaticity originates from the AV bundle. This leads to direct stimulation of the ventricles, causing AV dissociation and resulting in a dissociated P wave, which is visible in Figure 2b.

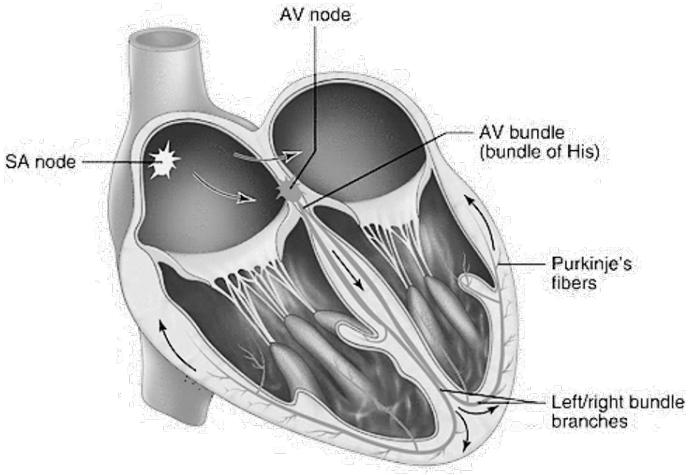
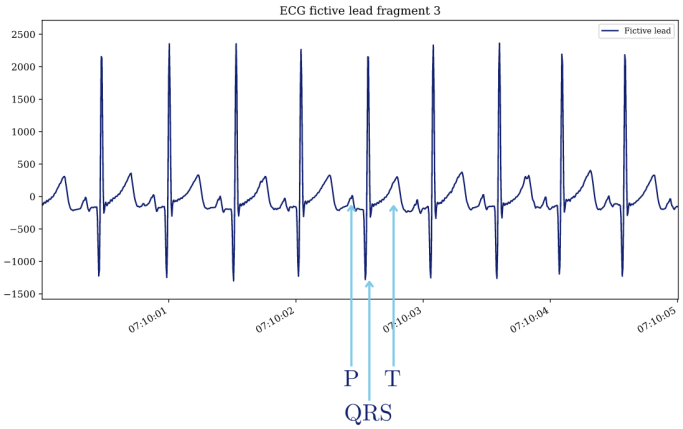
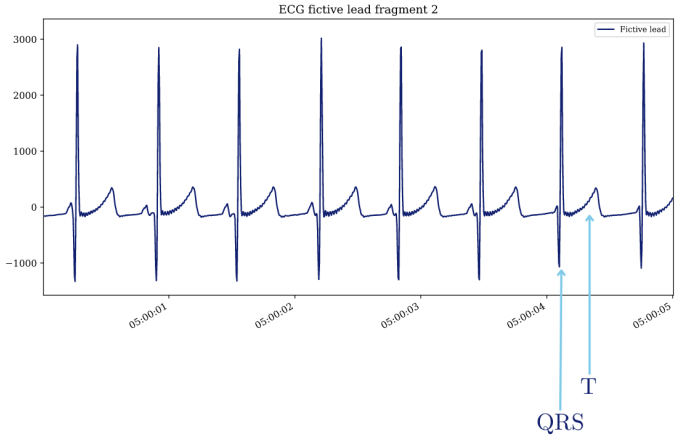


Figure 1: Heart conduction system



(a) SR



(b) JET

Figure 2: Examples of ECGs during SR and JET

While little research on JET detection has been conducted, Waugh et al.(2022) have attempted to develop an approach for this detection. [19, 20] However, their algorithm is less effective in case of a tachycardia, which is the prevailing circumstance within JET. Therefore, there is a need for a novel method which is not restricted to slower heart rates. This approach will be based on the typical characteristics of JET that are detectable on bedside monitor data, which may consist of a limited number of leads. However, besides features obtained from regular ECG leads, multiple leads can be combined in order to create a two-dimensional vectorcardiogram (VCG), which could be used to gain more insight into the direction of the electrical activity for each cardiac cycle. This combination of ECG and VCG characteristics will allow for the creation of a manual decision tree algorithm (not to be confused with the machine learning classifier) that can distinguish JET from sinus rhythm (SR).

3 Methods

A visual representation of the step-by-step process is provided in Figure 3. These steps will be addressed in detail in the following subsections.

Figure 3: Workflow overview

3.1 Data aquisition

This single-center retrospective study was conducted at the Erasmus MC Sophia Children's Hospital, using ECG data of children who experienced JET during their PICU admission between 2018 and 2023. From these patients, eight were randomly chosen, and multiple fragments were selected to ensure that at least one episode of SR and at least one episode of JET were available for each patient. A waiver for ethical approval was obtained for data collection using standard of care bedside monitoring (MEC-2021-0937). A minimum of three ECG leads were continuously recorded at 200 Hz and stored on a digital server, after which it was analysed in Python version 3.11. Appendix A and Appendix B provide an overview of both the code and the libraries that were used, respectively. From all of the time fragments, ranging between several hours up to one day, random segments of one minute were annotated by a pediatric intensivist. In this way, a gold standard was created, indicating whether the patient had SR or JET. Lead aVL, AVR, and aVF were calculated using the following equations:

$$aVL = \frac{1}{2}(I - III) \tag{1}$$

$$aVR = -\frac{1}{2}(I + II) \tag{2}$$

$$aVF = -\frac{1}{2}(II + III) \tag{3}$$

3.2 Data analysis

3.2.1 Classification based on features

To construct a manual decision tree algorithm, it is necessary to extract relevant features from the ECG data that can serve as an input. Using JET characteristics, it should be possible to differentiate between JET and SR. The specific features that were obtained are described in the following paragraphs.

VCG

The vectorcardiogram is a visualisation method which combines multiple ECG leads in order to gain more insight into the direction and magnitude of the electrical activity of the heart. [21] Usually, the VCG is reconstructed from 12-lead ECG data, but as monitor data often consists of three or five leads, in this case the X-axis was represented by lead I and the Y-axis was represented by lead AVF. This concept is illustrated in Figure 4.

As the initial 50 ms of the QRS complex have the most consistent direction and magnitude in a beat-to-beat comparison of a normal ECG, the standard deviation (SD) of the vector length within this interval was calculated and used as a distinctive feature. [22]

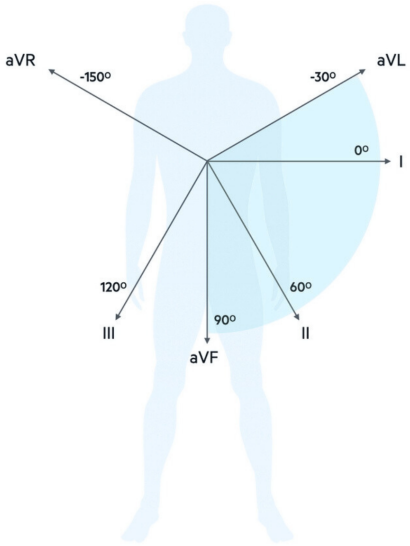


Figure 4: Cardiac axis visualisation

ECG

ECG features that were used for classification are:

- RR interval duration
- PR interval SD
- Fraction of detected P waves

Typical characteristics of JET are AV dissociation and tachycardia. [2–4] This is why RR interval duration and PR interval SD are considered distinctive features. In the case of AV dissociation, the P wave can be displaced, which reduces the amount of detectable P waves. [19] For this reason, another feature is the fraction of P waves that is detected within all of the searched intervals.

For each of the features mentioned in Section 3.2.1, multiple cutoff values and orders were tested. Given that sensitivity was considered as the most crucial performance metric, the decision tree displayed in Figure 5 was finally selected.

For the features to be obtained, accurate detection of the QRS complex and P peak is essential. Section 3.2.2 contains a description of the methods used for these detections.

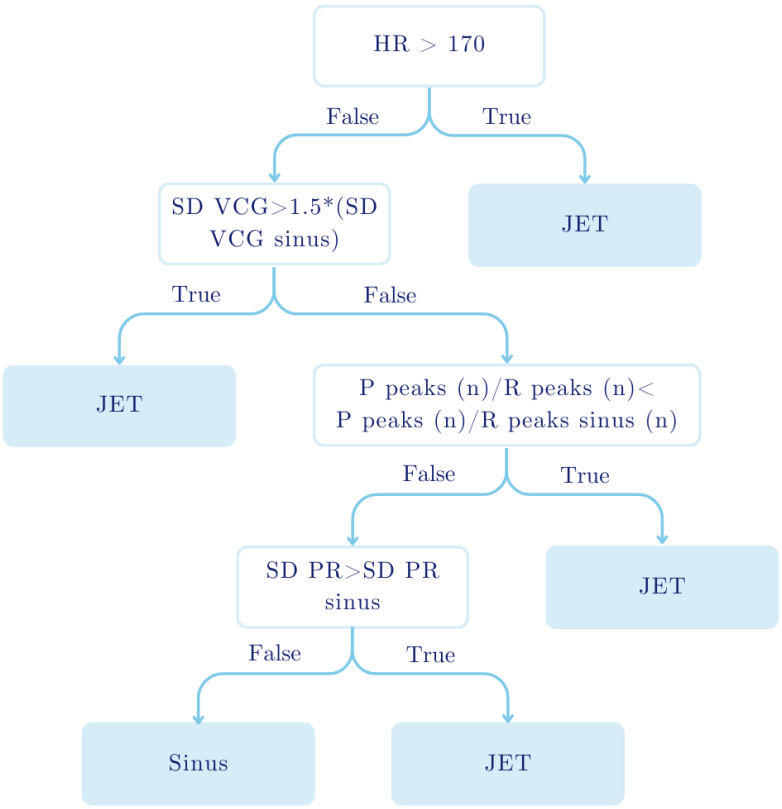


Figure 5: Overview of the selected decision tree. n = number.

3.2.2 Detection of ECG metrics

The detection of every ECG metric requires both a signal and a defined interval for locating that specific metric. Figure 6 presents an overview of every step in the detection process and the signals and metrics that were used as an input.

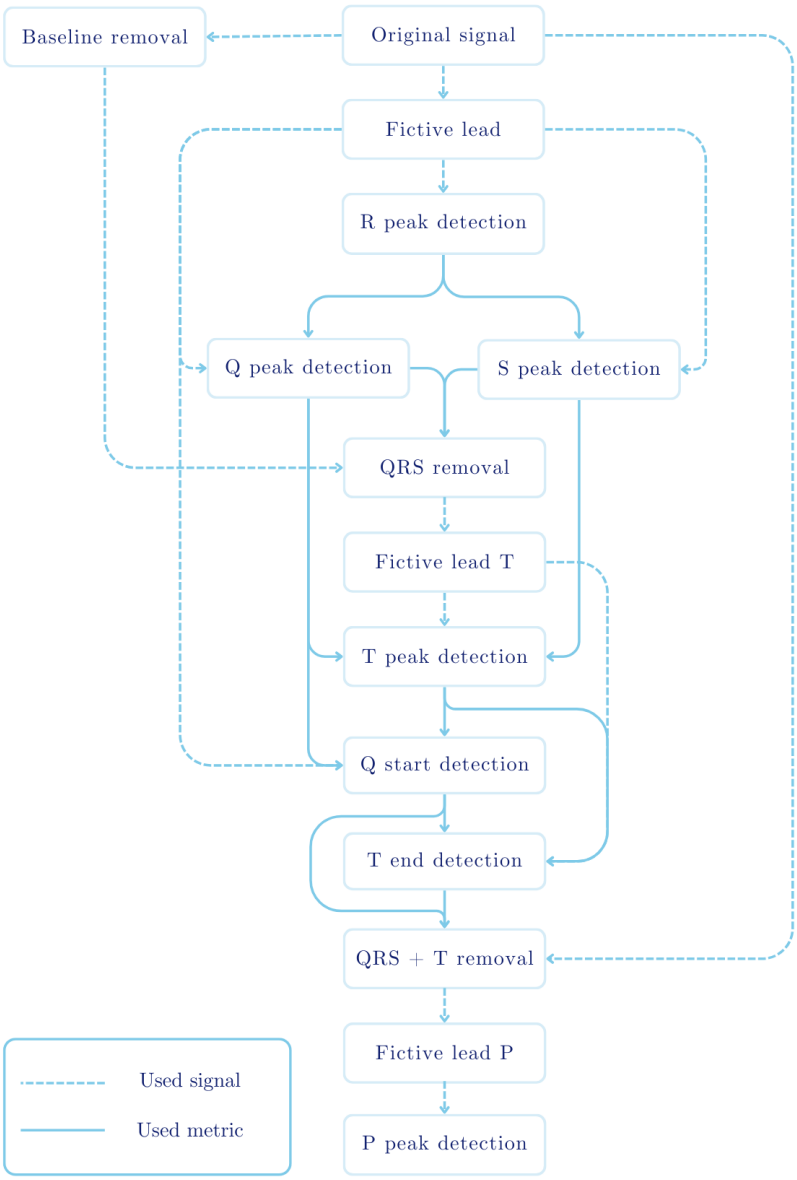


Figure 6: Flowchart detection of ECG metrics

R peak detection

To maximise the likelihood of accurate detection, peaks should be identified in the lead anticipated to have the highest peak amplitudes. However, as the available data is restricted to six leads, a fictive lead was generated within the angle corresponding to this highest amplitude. The cardiac axis was again recreated with lead I representing the X-axis and lead aVF the Y-axis. Using this coordinate system, where lead I represents an angle of 0 degrees and lead aVF an angle of 90 degrees, a vector was created for each time point. Subsequently, the length of each vector ($|\vec{b}|$) was calculated using the following equation[23]:

$$|\vec{b}| = \sqrt{I^2 + aVF^2}. \tag{4}$$

In the entire signal, the maximum vector length was searched for, as it is assumed that the maximum vector length is obtained during an R peak. At each time point the fictive lead, represented by the component of the signal along the maximum vector ($comp_{\vec{a}}\vec{b}$), was computed using the following equation:

$$comp_{\vec{a}}\vec{b} = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|}, \tag{5}$$

where $\vec{a}$ is the vector at the maximum angle, $\vec{b}$ is the vector at each time point, and $|\vec{a}|$ the length of the maximum vector. [23]

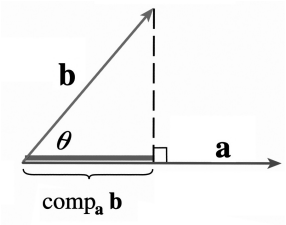


Figure 7: Scalar projection

The first ECG metric to be detected in this fictive lead was the R peak, because it is the most prominent characteristic and detection of other ECG metrics is based on the location of the R peaks. If the minimum amplitude of the signal was larger than the maximum amplitude of the signal, the signal was inverted in order to be able to search for positive peaks only. Automatic detection of possible R peaks involved the identification of local maxima, followed by determining the prominences of the detected peaks. This prominence depends on the absolute height of the peak and how much a peak stands out from the surrounding baseline of the signal. [24] To be able to distinguish between R peaks and other detected peaks, the prominences were plotted in a histogram. A curve was fitted over this histogram. Anticipating a bimodal distribution characterised by one peak in the histogram representing R peaks and another representing non-R peaks, the two largest peaks of the curve were identified. Subsequently, the minimum between these two peaks was found and set as the limit of the prominence. This prominence value was then used as a threshold to perform R peak detection.

Q and S peak detection

The Q and S peak were detected in the same fictive lead as used for the R peak detection. Physiologically the the Q and S peak will always occur before and after the R peak, respectively. The maximum duration of a normal QRS interval is 120 ms, which is equivalent to 24 time samples ($0,12 \cdot Fs = 0,12 \cdot 200 = 24$). [25] Assuming that the Q and S peaks are approximately symmetrically distributed around the R peak, the Q peak was searched for in an interval from 12 samples before the R peak until the R peak, and the S peak was searched for in an interval from the R peak up to 12 samples from the R peak. Peaks were searched for in the vertical direction opposite to the R peak. If there were multiple possible Q peaks detected, the last one occuring was selected. If there were multiple possible S peaks detected, the one with the lowest value was selected. If no possible peak was found for either, the point with the lowest value within the interval was selected as a peak. In this way, every R peak has a corresponding Q and S peak available at the approximately correct location.

T peak detection

After baseline correction of the original signal, the QRS complex was subtracted by setting the amplitudes within the QRS interval to the baseline value. Then, a new fictive lead was created for T wave detection using the same method employed for R peak detection. Subsequently, the highest amplitude within each S-Q interval was marked as the peak of the T wave.

Q start and T end detection

For the detection of the start of the Q wave and the end of the T wave, a similar method was used. This 'trapezium's area approach', as proposed by Vázquez-Seisdedo et al.(2011) involves selecting two points, denoted as m and r, and computing the area of each trapezium formed by connecting these two points with a mobile point i, as depicted in Figure 8. [26] The area A is calculated using the following formula:

$$A = 0.5(y_m - y_i)(2x_r - x_i - x_m), \tag{6}$$

where:

- m = the point with the highest absolute derivative within the descending slope of the T wave;
- r = a random point located on the isoelectric segment;
- i = a mobile point on the ECG signal positioned between points m and r.

The end of the T wave is then defined as the location with the maximum area A. For Q start detection, the same principle was applied, but with mirrored points for m and r, and searching from the Q peak in the opposite direction.

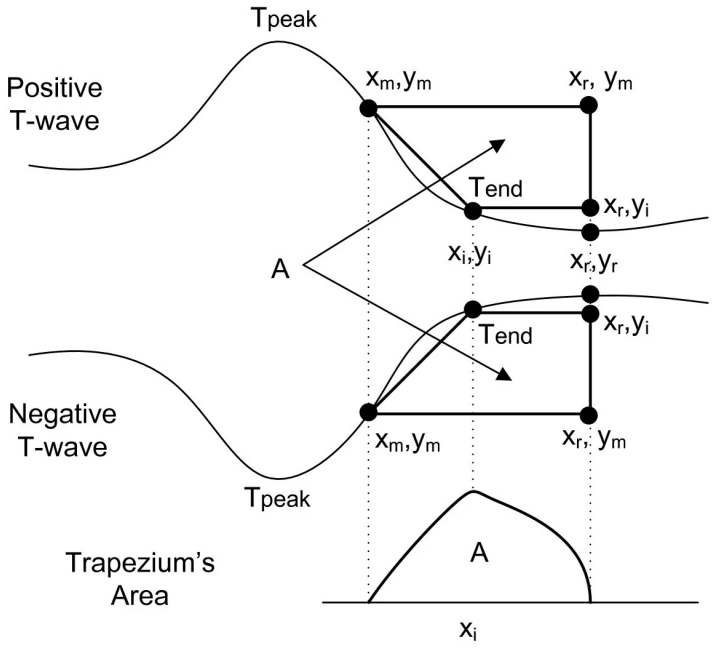


Figure 8: Trapezium method

P peak detection

Having localised the QRS complex and T wave, the next step involved their exclusion from the original ECG signal. This was accomplished by setting the interval from the onset of the Q peak to the end of the T peak to a new baseline. This baseline was defined as the Y-value at the termination of the T wave. Finally, in this signal, the peak of the P wave was identified using the same method employed for R peak detection (i.e. a new fictive lead was created and P peaks were located based on the prominence histogram).

3.3 Descriptive statistics

After classification, descriptive statistics were used to evaluate the performance of the algorithm. Performance was measured in terms of the following performance metrics:

- Accuracy
- Sensitivity
- Specificity
- Positive predictive value (PPV)

4 Results

4.1 Research population

The total number of JET patients admitted to the PICU between 2018 and 2023 is 20. Baseline characteristics of the eight randomly included patients are reported in Table 1.

Median age in days (Q1-Q3)	149 (83-176)
Gender (N)	F (1), M (7)
Surgery indications (N)	Fallot (4), VSD (3), TGA (1)

Table 1: Baseline characteristics. Q1 = lower quartile, Q3 = upper quartile, Fallot = Tetralogy of Fallot, VSD = ventricular septal defect, TGA = transposition of the great arteries.

4.2 Data description

From these eight patients, a total of 41 one-minute fragments were selected, resulting in 41 minutes of data containing a total of 5931 heartbeats. Among these fragments, 14 (34.2%) were annotated as SR, while 27 were labeled as JET (65.8%). In Figure 14 of Appendix C, an example is depicted of a fragment in all six leads that were used for annotation. In one fragment, an artifact was present in lead i, likely caused by poor electrode contact. [27] This is demonstrated in Figure 15 of Appendix C.

4.3 Data analysis

The total processing time for classifying the 41 minutes of data was 4 minutes and 48 seconds. Figure 9 illustrates a histogram plot of the prominences of detected potential R peaks, which varied across fragments but typically showed a cluster of R peak prominences on the right side and other detected peaks on the left. Consequently, the prominence limit varied accordingly. An example of the effect of the baseline correction that was applied, is depicted in Figure 16 of Appendix C.

In Figure 10, an example of the reconstructed VCG in SR during a single heartbeat is presented. Here, the largest loop in dark blue corresponds to the QRS complex, the smaller one in light blue to the T wave, and the smallest magenta loop matches the P wave. Figure 11 displays VCG comparisons for a patient in both SR and JET across a minute of data, where each loop again represents one heartbeat. This figure demonstrates a broader direction range for patients during JET. Figure 12 illustrates a patient who has a rhythm that is alternating between SR and JET. The rises in heart rate correspond to episodes of JET. The SD of the VCG is increasing at the same moments in time.

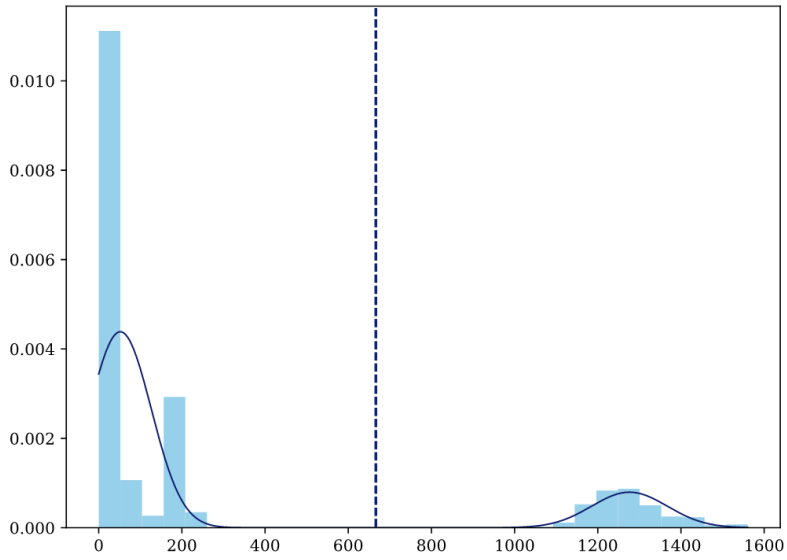


Figure 9: Prominence histogram of R peaks. The dotted line represents the prominence limit.

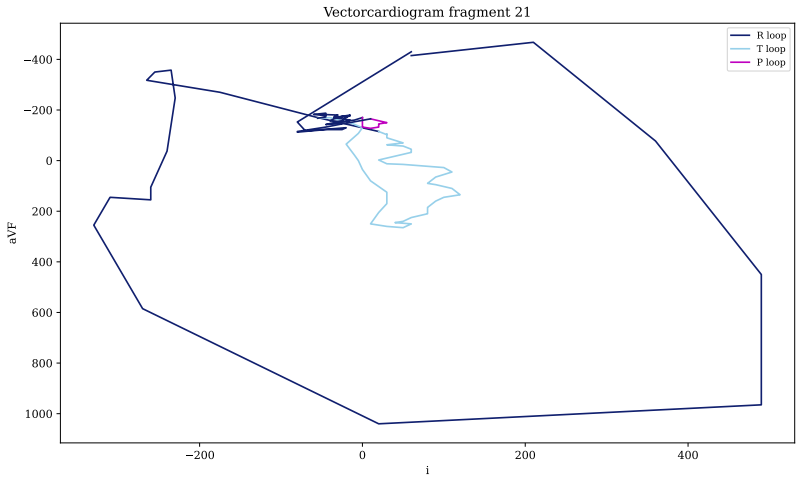
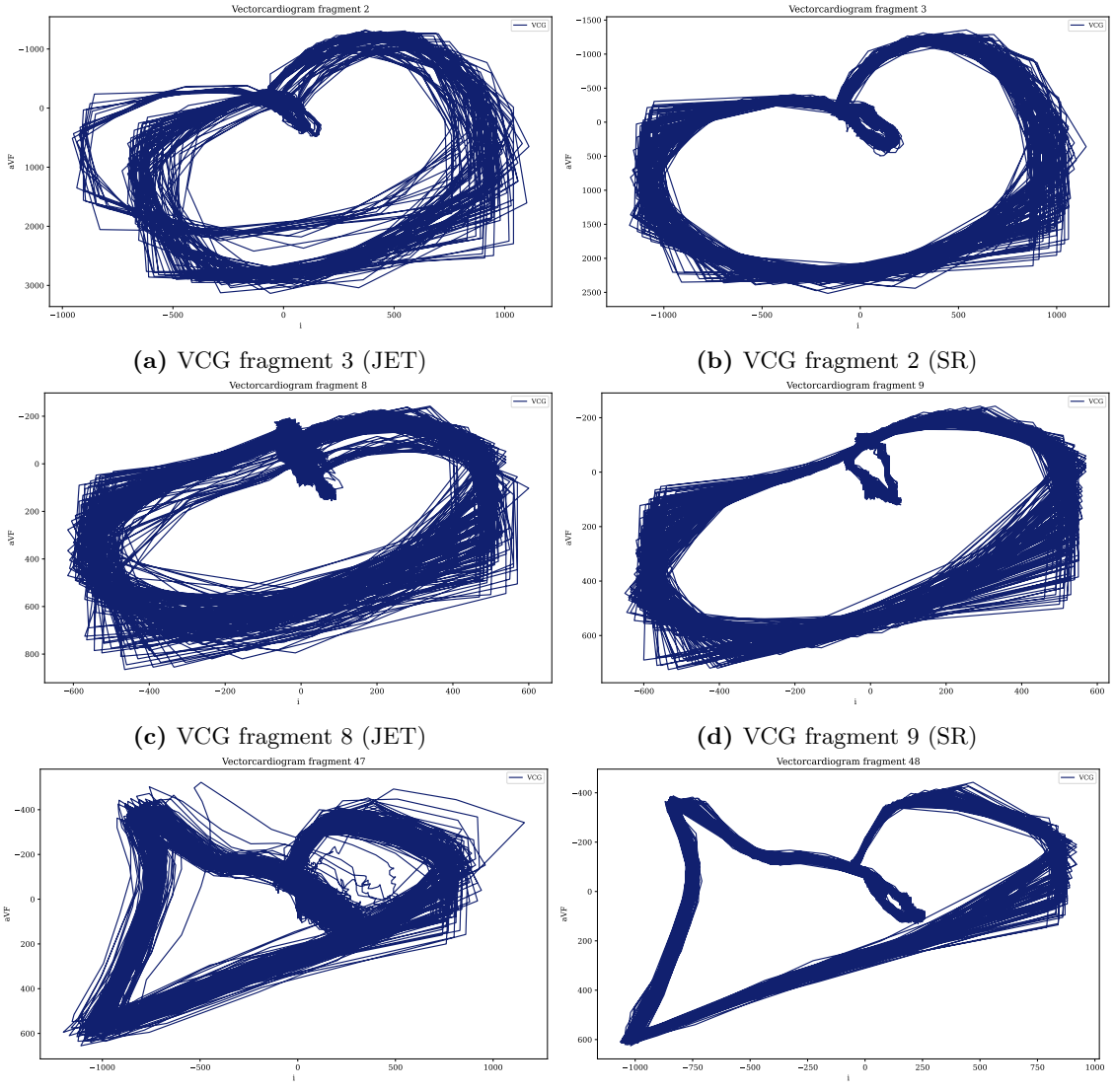


Figure 10: VCG during one heartbeat



(a) VCG fragment 3 (JET)

(b) VCG fragment 2 (SR)

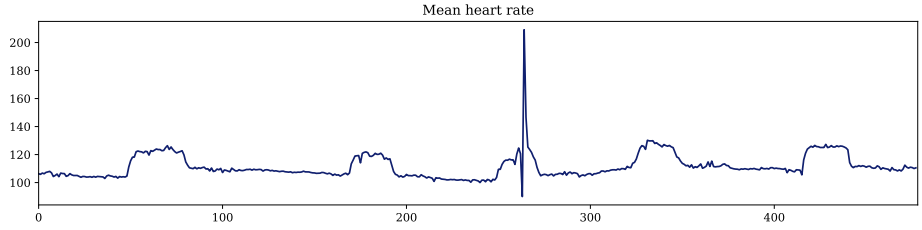
(c) VCG fragment 8 (JET)

(d) VCG fragment 9 (SR)

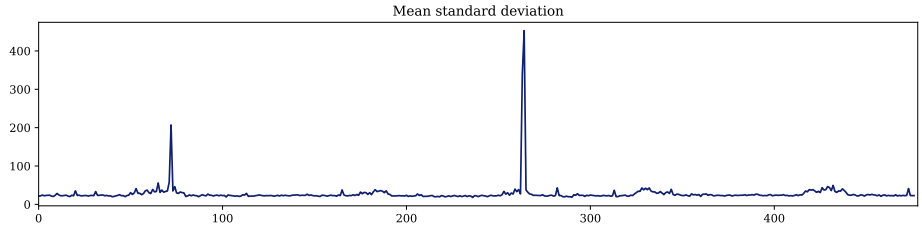
(e) VCG fragment 47 (JET)

(f) VCG fragment 48 (SR)

Figure 11: VCGs patient 2 (11a & 11b), 1 (11c & 11d), and 8 (11e & 11f)



(a) Heart rate



(b) Mean SD

Figure 12: Comparison HR and SD

4.4 Performance

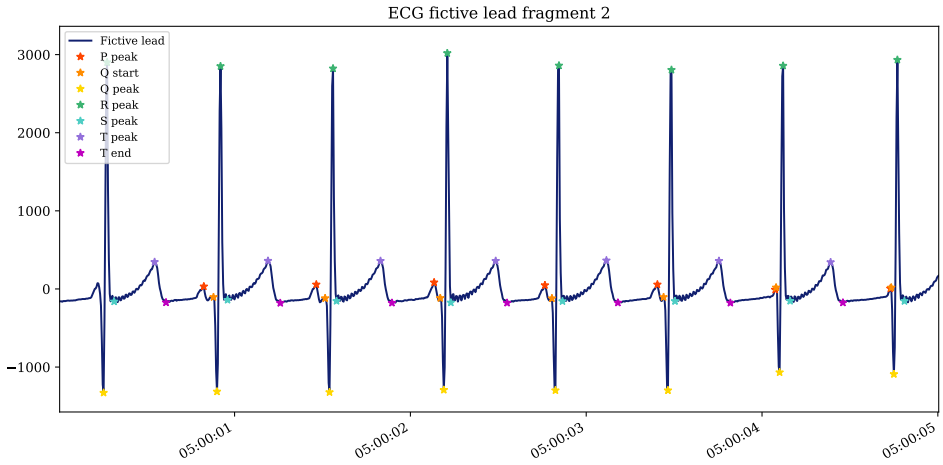
An overview of the classification performance is provided in the confusion matrix of Table 2 and in Table 3. Of the 27 JET fragments, 26 were correctly classified, resulting in a sensitivity of 96.3%. Out of 30 fragments classified as JET, 26 were actually JET fragments, corresponding to a PPV of 86.7%. An example of successful detection in both SR and JET fragments can be found in Figure 13. However, not all detections were accurate, as demonstrated in Table 4, which outlines the detection accuracy per ECG metric. Furthermore, Figure 17 in Appendix C presents examples of inaccurate detections for each ECG metric.

		Predicted rhythm		
		Sinus	JET	**Total**
Actual rhythm	Sinus	10	4	14
	JET	1	26	27
	Total	11	30	41

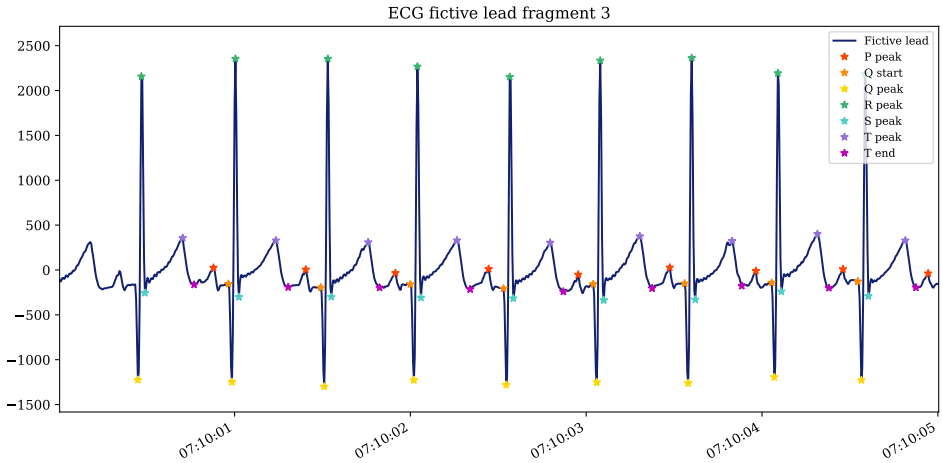
Table 2: Confusion matrix

Performance metric	**Value**
Sensitivity	96.3%
Specificity	71.4%
PPV	86.7%
Accuracy	87.8%

Table 3: Performance of classification



(a) JET



(b) SR

Figure 13: PQRST detection in fictive lead

ECG metric	R peak	Q peak	S peak	T peak	Q start	T end	P wave
% correct	99.9	95.7	89.7	98.1	90.2	76.0	54.8

Table 4: Performance of ECG metric detection

5 Discussion

5.1 Findings

We developed an algorithm capable of distinguishing JET from SR based on features derived from one-minute fragments of ECG monitor data. The algorithm achieves a classification performance with a sensitivity of 96.3%, a specificity of 71.4%, a PPV of 86.7%, and an accuracy of 87.8%. The high sensitivity indicates that JET detection is feasible with the proposed algorithm, which was the primary objective of this research. Furthermore, with a PPV of 86.7%, our algorithm shows promise for clinical application, as this indicates a relatively low false alarm rate. This PPV suggests that the detection tool would not contribute extensively to alarm fatigue, which is a well-known issue in ICUs. [28]

However, due to the class imbalance in the training data (more JET than SR) which is opposite from real-world data, application may still result in frequent alarms. Despite not being directly suitable for clinical use, the algorithm serves as a solid foundation for the development of an algorithm that can be integrated into the monitor alarm system. Additionally, the relatively low processing times further enhance its feasibility for clinical implementation.

Altogether, this research introduces three significant advancements:

1. To our knowledge, this is the first study to use a combination of VCG and ECG features to detect postoperative JET at the PICU. [20] Inclusion of the VCG enhances the performance of the algorithm by providing additional information on the direction and magnitude of electrical activity. Figure 12 shows that the SD of the VCG vector magnitude within the initial 50 ms of the QRS complex is a distinctive feature, as there is an increase in SD during JET episodes.

2. This is the first JET detection algorithm that is not restricted to specific heart frequencies, which is a crucial property given the the elevated HR during JET and the wide ranging HRs of patients admitted to the PICU in general. Whereas there is one article that developed a JET detection algorithm using just ECG features, their performance was lower for patients with higher frequencies, likely due to the fixed intervals used for analysis. [19, 20] Our algorithm overcomes this limitation by employing variable intervals based on ECG morphology, enabling robust detection of Q, R, S, and T peaks. Nevertheless, accurate detection of P peaks remains challenging, especially on noisy monitor data.

3. The use of a fictive lead for detection of ECG metrics is an innovative and promising approach that can be easily adopted by studies that involve detection of ECG metrics, offering a solution for improved analysis.

While R peaks and T peaks are detected with high accuracy, the detection of other ECG metrics is less reliable. Inaccuracies in identifying S and Q peaks are prevalent in patients with bundle branch blocks due to narrow search intervals. Errors in T wave end detection often occur when P waves are mistakenly identified as the end of the T wave, a challenging mistake to avoid utilising this approach. The most crucial issue is the lack of accuracy in P wave detection. Since this step is the final one of the detection process, inaccurate detection of other ECG metrics will often automatically lead to inaccurate P wave detection. Additionally, the low amplitude of the P wave further complicates its distinction.

Considering baseline characteristics, two observations stand out. Firstly, the age range is relatively narrow compared to the general PICU population (ranging from 0-18 years), which aligns with expectations given that congenital heart surgeries are typically performed at a young age. [29] Additionally, the male-to-female ratio was unevenly distributed. However, as no inferential statistics were applied, no definitive conclusions can be drawn from this finding. Given the limited number of included patients and the findings from Waugh et al. (2022), which reported a male-to-female ratio of 18:22 in a similar patient population, this inequality is likely due to coincidence. [19]

5.2 Limitations

While the use of adaptive intervals allows for the analysis of fragments covering a wide range of heart frequencies, it does have a drawback. Given the dependency among all ECG metrics, an inaccurate detection of one metric is likely to lead to inaccurate detection of others. Being

retrospective, this study also presents some limitations. Given that the classification is partially dependent on features in SR, it is crucial to acquire a sinus fragment with minimal noise and artifacts. In prospective research, extra attention can be paid to electrode placement, and all twelve leads can be utilised for a baseline sinus ECG. Although patients and fragments were randomly selected, a small probability remains of selection bias. Certain subgroups, such as age categories or gender, may be over- or underrepresented in the selection. Lastly, to simplify the annotation process, only small parts of one-minute fragments were presented to the pediatric intensivist. While the likelihood is low, there is a possibility of overlooking cases where JET and sinus are coinciding in one fragment, leading to a classification that lacks coherence.

5.3 Recommendations for future research

Improving current algorithm

For further development of the algorithm, several recommendations can be proposed. Due to time restrictions, no filtering or artifact detection was conducted in this study. It is advised to incorporate this step into the process as monitor data typically contains a significant amount of noise, complicating detection and classification. Another strategy to enhance classification is to expand the use of the P wave for feature extraction. Currently, the algorithm searches for the P wave within the T end-Q start interval. Since retrograde P waves are frequently observed in JET, detecting P waves within the S-T interval could enhance classification. [2] In addition to P wave location, inclusion of the P prominence median could increase the capability of the algorithm to differentiate between JET and SR. Waugh et al.(2022) demonstrated that this feature contributes to JET prediction, as P waves tend to become less distinguishable within the T wave or QRS complex. [19] However, before implementation, improvement of the existing P wave detection method is required, as its current performance is insufficient for accurate extraction of the P prominence. An additional promising feature to incorporate into the decision tree is the central venous pressure waveform. Tan et al.(2021) demonstrate that integrating this feature into a JET detection algorithm can improve performance. [30]

VCG analysis

A distinctive aspect of this JET detection approach is the usage of the VCG. The VCG has proven to be a clarifying means of visualising electrical activity and needs further exploration. CineECG could serve as a valuable tool for VCG analysis in JET patients. [31] Since strict real-time JET detection is not critical, utilising VCGs as input for a machine learning algorithm has the potential to achieve accurate JET detection with a slight delay. To our knowledge, there is no literature investigating the current extent of treatment delays without a JET warning system. Given that these delays are estimated to be on the scale of minutes of even hours, a delay of several minutes may be considered acceptable. The purpose of reducing delays is to initiate treatment sooner, aiming for an improvement in patient outcomes. However, it is necessary to gain more insight into the exact impact of the time component on treatment effectiveness to define a more specific acceptable time delay. Thus, for now, any delay reduction is favourable.

A last potential application of the VCG could be a bedside visualisation tool. VCG anomalies during JET may be more apparent than minor ECG deviations on the monitor. Therefore, integrating the VCG into a monitoring dashboard could assist in diagnosis.

Clinical application

For the algorithm to be clinically applicable, generalisability must be validated. A prospective study with a larger dataset is essential for validation. It should involve more patients, including those developing JET and those who do not, for a more representative sample. Fragments should be collected frequently to ensure a sufficient number of SR fragments, facilitating comparison of their features with features from JET fragments. Once the algorithm has been validated, given its low computational time and high performance on actual PICU monitor data, this detection approach is suitable for implementation as a bedside tool. This would facilitate early identification of JET, allowing for timely treatment, with the expectation of enhanced arrhythmia control. Ultimately, this could result in lower morbidity and mortality rates, shorter PICU admission times, and consequently lead to reduced healthcare costs.

6 Conclusion

An algorithm was developed using a unique method for the bedside detection of JET, achieving a high sensitivity and PPV. This approach uses JET characteristics derived from both ECG and VCG data, and has the potential to lead to the development of a clinically applicable JET detection tool, due to its low computational time and high performance. Implementing this early recognition method could potentially reduce morbidity and mortality associated with JET episodes during PICU admissions through timely intervention.

References

[1] Jaganmohan Tharakan and Kiron Sukulal. "Post cardiac surgery junctional ectopic tachycardia: A 'Hit and Run' tachyarrhythmia as yet unchecked". In: *Annals of Pediatric Cardiology* 7.1 (Jan. 2014), p. 25. ISSN: 09742069. DOI: 10.4103/0974-2069.126545. URL: /pmc/articles/PMC3959056/%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3959056/.

[2] C. (Christopher) Wren. *Concise guide to pediatric arrhythmias*. Chichester, West Sussex, UK: Wiley-Blackwell, 2012, p. 189. ISBN: 978-0-470-65855-0. URL: https://www.wiley.com/en-ie/Concise+Guide+to+Pediatric+Arrhythmias-p-9780470658550.

[3] Edward P. Walsh et al. "Evaluation of a staged treatment protocol for rapid automatic junctional tachycardia after operation for congenital heart disease". In: *Journal of the American College of Cardiology* 29.5 (Apr. 1997), pp. 1046–1053. ISSN: 07351097. DOI: 10.1016/S0735-1097(97)00040-5. URL: https://pubmed.ncbi.nlm.nih.gov/9120158/.

[4] Mohammad Alasti et al. "Junctional ectopic tachycardia (JET)". In: *Journal of Arrhythmia* 36.5 (Oct. 2020), pp. 837–844. ISSN: 18832148. DOI: 10.1002/JOA3.12410.

[5] A. Dodge-Khatami et al. "Impact of junctional ectopic tachycardia on postoperative morbidity following repair of congenital heart defects". In: *European journal of cardio-thoracic surgery : official journal of the European Association for Cardio-thoracic Surgery* 21.2 (2002), pp. 255–259. ISSN: 1010-7940. DOI: 10.1016/S1010-7940(01)01089-2. URL: https://pubmed.ncbi.nlm.nih.gov/11825732/.

[6] J. B. Andreasen, S. P. Johnsen, and H. B. Ravn. "Junctional ectopic tachycardia after surgery for congenital heart disease in children". In: *Intensive care medicine* 34.5 (May 2008), pp. 895–902. ISSN: 0342-4642. DOI: 10.1007/S00134-007-0987-2. URL: https://pubmed.ncbi.nlm.nih.gov/18196218/.

[7] Leena Mildh et al. "Junctional ectopic tachycardia after surgery for congenital heart disease: incidence, risk factors and outcome". In: *European journal of cardio-thoracic surgery : official journal of the European Association for Cardio-thoracic Surgery* 39.1 (Jan. 2011), pp. 75–80. ISSN: 1873-734X. DOI: 10.1016/J.EJCTS.2010.04.002. URL: https://pubmed.ncbi.nlm.nih.gov/20537549/.

[8] Navaneetha Sasikumar, Raman Kumar, and Seshadri Balaji. "Diagnosis and management of junctional ectopic tachycardia in children". In: *Annals of Pediatric Cardiology* 14.3 (July 2021), p. 372. ISSN: 09745149. DOI: 10.4103/APC.APC{_}35{_}21. URL: /pmc/articles/PMC8457265/%20/pmc/articles/PMC8457265/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8457265/.

[9] A. S. Batra et al. "A prospective analysis of the incidence and risk factors associated with junctional ectopic tachycardia following surgery for congenital heart disease". In: *Pediatric cardiology* 27.1 (Feb. 2006), pp. 51–55. ISSN: 0172-0643. DOI: 10.1007/S00246-005-0992-6. URL: https://pubmed.ncbi.nlm.nih.gov/16391972/.

[10] Jeffrey W. Delaney et al. "Early postoperative arrhythmias after pediatric cardiac surgery". In: *The Journal of thoracic and cardiovascular surgery* 131.6 (June 2006), pp. 1296–1300. ISSN: 1097-685X. DOI: 10.1016/J.JTCVS.2006.02.010. URL: https://pubmed.ncbi.nlm.nih.gov/16733160/.

[11] Ali Dodge-Khatami et al. "Surgical substrates of postoperative junctional ectopic tachycardia in congenital heart defects". In: *Journal of Thoracic and Cardiovascular Surgery* 123.4 (2002), pp. 624–630. ISSN: 00225223. DOI: 10.1067/mtc.2002.121046. URL: https://pubmed.ncbi.nlm.nih.gov/11986588/.

[12] Jean Pierre Pfammatter et al. "Successful management of junctional tachycardia by hypothermia after cardiac operations in infants". In: *The Annals of thoracic surgery* 60.3 (1995), pp. 556–560. ISSN: 0003-4975. DOI: 10.1016/0003-4975(95)00425-K. URL: https://pubmed.ncbi.nlm.nih.gov/7677480/.

[13] Timothy M. Hoffman et al. "Postoperative junctional ectopic tachycardia in children: incidence, risk factors, and treatment". In: *The Annals of thoracic surgery* 74.5 (Nov. 2002), pp. 1607–1611. ISSN: 0003-4975. DOI: 10.1016/S0003-4975(02)04014-6. URL: https://pubmed.ncbi.nlm.nih.gov/12440616/.

[14] B. Hugh Dorman et al. “Magnesium supplementation in the prevention of arrhythmias in pediatric patients undergoing surgery for congenital heart defects”. In: *American heart journal* 139.3 (2000), pp. 522–528. ISSN: 0002-8703. DOI: 10.1016/S0002-8703(00)90097-8. URL: https://pubmed.ncbi.nlm.nih.gov/10689268/.

[15] Ranjit I. Kylat and Ricardo A. Samson. “Junctional ectopic tachycardia in infants and children”. In: *Journal of Arrhythmia* 36.1 (Feb. 2020), p. 59. ISSN: 18832148. DOI: 10.1002/JOA3.12282. URL: /pmc/articles/PMC7011855/%20/pmc/articles/PMC7011855/?report=abstract%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7011855/.

[16] M. A. Ruder et al. “Clinical and electrophysiologic characterization of automatic junctional tachycardia in adults.” In: *Circulation* 73.5 (1986), pp. 930–937. ISSN: 00097322. DOI: 10.1161/01.CIR.73.5.930. URL: https://www.ahajournals.org/doi/abs/10.1161/01.cir.73.5.930.

[17] Lubica Kovacikova et al. “Amiodarone as a first-line therapy for postoperative junctional ectopic tachycardia”. In: *The Annals of thoracic surgery* 88.2 (Aug. 2009), pp. 616–622. ISSN: 1552-6259. DOI: 10.1016/J.ATHORACSUR.2009.04.088. URL: https://pubmed.ncbi.nlm.nih.gov/19632422/.

[18] J. Philip Saul et al. “Intravenous Amiodarone for Incessant Tachyarrhythmias in Children”. In: *Circulation* 112.22 (Nov. 2005), pp. 3470–3477. ISSN: 00097322. DOI: 10.1161/CIRCULATIONAHA.105.534149. URL: https://www.ahajournals.org/doi/abs/10.1161/CIRCULATIONAHA.105.534149.

[19] J L S Waugh et al. “A novel automated junctional ectopic tachycardia detection tool for children with congenital heart disease”. eng. In: *Heart Rhythm O2* 3.3 (2022), pp. 302–310. ISSN: 2666-5018. DOI: 10.1016/j.hroo.2022.02.014. URL: https://www.heartrhythmopen.com/article/S2666-5018(22)00060-5/pdf.

[20] Gini Raaijmakers. *Automated electrocardiogram interpretation for the detection of postoperative junctional ectopic tachycardia at the pediatric intensive care unit*. Tech. rep. Rotterdam: Erasmus MC, Sept. 2023.

[21] Jaroslav Vondrak and Marek Penhaker. “Review of Processing Pathological Vectorcardiographic Records for the Detection of Heart Disease”. In: *Frontiers in Physiology* 13 (Mar. 2022). ISSN: 22968016. DOI: 10.3389/FPHYS.2022.856590. URL: /pmc/articles/PMC9536877/%20/pmc/articles/PMC9536877/?report=abstract%20https://www-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/pmc/articles/PMC9536877/.

[22] Peter M. van Dam et al. “The relation of 12 lead ECG to the cardiac anatomy: The normal CineECG”. In: *Journal of Electrocardiology* 69 (Nov. 2021), pp. 67–74. ISSN: 0022-0736. DOI: 10.1016/J.JELECTROCARD.2021.07.014.

[23] James Stewart. *Calculus: Early Transcendentals*. 8th ed. Cengage learning, 2015, pp. 791–846.

[24] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[25] Xin Li et al. “Efficacy of dexmedetomidine in prevention of junctional ectopic tachycardia and acute kidney injury after pediatric cardiac surgery: A meta-analysis”. In: *Congenital heart disease* 13.5 (Sept. 2018), pp. 799–807. ISSN: 1747-0803. DOI: 10.1111/CHD.12674. URL: https://pubmed.ncbi.nlm.nih.gov/30260073/.

[26] Carlos R. Vázquez-Seisdedos et al. “New approach for T-wave end detection on electrocardiogram: Performance in noisy conditions”. In: *BioMedical Engineering Online* 10.1 (Sept. 2011), pp. 1–11. ISSN: 1475925X. DOI: 10.1186/1475-925X-10-77/FIGURES/6. URL: https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-10-77.

[27] Toshihiro Takeshita et al. “Relationship between Contact Pressure and Motion Artifacts in ECG Measurement with Electrostatic Flocked Electrodes Fabricated on Textile”. In: *Scientific Reports 2019 9:1* 9.1 (Apr. 2019), pp. 1–10. ISSN: 2045-2322. DOI: 10.1038/S41598-019-42027-X. URL: https://www-nature-com.tudelft.idm.oclc.org/articles/s41598-019-42027-x.

[28] Barbara J. Drew et al. “Insights into the problem of alarm fatigue with physiologic monitor devices: a comprehensive observational study of consecutive intensive care unit patients”. In: *PloS one* 9.10 (Oct. 2014). ISSN: 1932-6203. DOI: 10.1371/JOURNAL.PONE.0110274. URL: https://pubmed-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/25338067/.

[29] Dominique Vervoort et al. “Children at the Heart of Global Cardiac Surgery: An Advocacy Stakeholder Analysis”. In: *World journal for pediatric & congenital heart surgery* 12.1 (Jan. 2021), pp. 48–54. ISSN: 2150-136X. DOI: 10.1177/2150135120955189. URL: https://pubmed-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/33407026/.

[30] Xin Tan et al. “Detection of Junctional Ectopic Tachycardia by Central Venous Pressure”. In: *Artificial intelligence in medicine. Conference on Artificial Intelligence in Medicine (2005-)* 12721 (2021), p. 258. ISSN: 16113349. DOI: 10.1007/978-3-030-77211-6{_}29. URL: /pmc/articles/PMC8281976/%20/pmc/articles/PMC8281976/?report=abstract%20https://www-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/pmc/articles/PMC8281976/.

[31] Machteld J. Boonstra et al. “CineECG: A novel method to image the average activation sequence in the heart from the 12-lead ECG”. In: *Computers in biology and medicine* 141 (Feb. 2022). ISSN: 1879-0534. DOI: 10.1016/J.COMPBIOMED.2021.105128. URL: https://pubmed.ncbi.nlm.nih.gov/34973587/.

Appendices

- Appendix A: Python script
- Appendix B: Python libraries
- Appendix C: Supplementary figures

Appendix A: Python script

```
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 20 12:00:31 2024

@author: g.raaijmakers
"""

'''Load libraries'''
import os
import scipy.io
import math
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt, dates as md
from scipy import signal
from scipy.signal import find_peaks
from datetime import datetime
from sklearn.mixture import GaussianMixture
from pybaselines import Baseline
import warnings
import matplotlib as mpl

def find_sin(fragment, meta_fragment):
    '''This function finds the first sinus fragment number of the provided patient'''
    patient = meta_fragment.loc[fragment,'Patient']
    fragment_sin = meta_fragment.query(f'Patient=="{patient}" and Rhythm=="Sinus"').index[0]
    return fragment_sin

def flatten_data(lead, mat_file):
    '''This function unpacks the data from the provided lead.'''
    data = mat_file[lead]['data'].item().flatten()
    return data

def timeframe(date, time):
    '''Combines date and time for the fragment'''
    return datetime.strptime(f"{date} {time}", "%Y-%m-%d %H:%M:%S")

meta_fragment = pd.read_excel('Z:/ECG project TM/Gini/Annotaties.xlsx', names=['Fragment','Rhythm','Patient','File_number','Start_time','End_sec','Raw_folder','File_name','End_min'], index_col='Fragment') #Load excel file into dataframe, set fragment number as index
df_features = pd.DataFrame({'Rhythm':meta_fragment.Rhythm, 'Fragment_sin':[find_sin(i, meta_fragment) for i in meta_fragment.index], 'mean_HR':np.nan, 'VCG_SD':np.nan, 'VCG_SD_sin':np.nan,'frac_ppeaks':np.nan,'frac_ppeaks_sin':np.nan,'PR_SD':np.nan,'PR_SD_sin':np.nan}) #Create a dataframe that can be used for an overview of all of the extracted features

'''Loop for extracting features for each fragment'''
for fragment in df_features.index:
    raw_folder =  meta_fragment.Raw_folder[int(fragment)] #Find the folder corresponding to the selected fragment
    fragment_path = os.path.join(raw_folder, meta_fragment.File_name[int(fragment)]) #Find the file path corresponding to the selected fragment
    mat_file = scipy.io.loadmat(fragment_path) #Load the .mat file
    meta_file = mat_file['meta'] #Load metadata of the .mat file
    date_format = '%Y/%m/%d %H:%M:%S'
    start_date_file = datetime.strptime(meta_file.item()[2].item(), date_format) #Extract the start date of the file
    end_date_file = datetime.strptime(meta_file.item()[3].item(), date_format) #Extract the end date of the file
    time_range = pd.date_range(start_date_file, end_date_file, freq='5ms') #Timeframe of the file


    #Create necessary leads
    lead_i = flatten_data('i',mat_file)
    lead_ii = flatten_data('ii',mat_file)
    lead_iii = flatten_data('iii',mat_file)
    lead_aVR = -0.5*(lead_i+lead_ii)
    lead_aVL = 0.5*(lead_i-lead_iii)
    lead_aVF = 0.5*(lead_ii+lead_iii)

```

```
    #Create dataframe of complete file
    df_complete = pd.DataFrame({'Timestamps': time_range[:-1], 'i': lead_i, 'ii': lead_ii, 'iii': lead_iii,
                        'aVR': lead_aVR, 'aVL': lead_aVL, 'aVF': lead_aVF,
                        'Pwaves': False, 'Qpeaks': False, 'Rpeaks': False, 'Speaks': False, 'Tpeaks': False,
                        'RR_inv': np.nan, 'HR': np.nan, 'mean_HR': np.nan})


    start_time_fragment = meta_fragment.Start_time[int(fragment)]
    end_time_fragment = meta_fragment.End_min[int(fragment)]

    start_date_fragment = timeframe(start_date_file.date(), start_time_fragment) #Start date and time of the fragment
    end_date_fragment = timeframe(start_date_file.date(), end_time_fragment) #End date and time of the fragment

    df = df_complete[((df_complete.Timestamps > start_date_fragment) & (df_complete.Timestamps < end_date_fragment))].reset_index(drop=True)

    df_frag =  df[((df.Timestamps > start_date_fragment) & (df.Timestamps < end_date_fragment))].reset_index(drop=True)

    leads = ['i','ii','iii','aVR','aVL','aVF']

    x=df_frag.index
    baseline_fitter = Baseline(x_data=x)
    df_baseline = pd.DataFrame([baseline_fitter.snip(df_frag[y],max_half_window=40, decreasing=True,smooth_half_window=20)[0] for y in leads]).T
    df_baseline.columns = leads
    df_corrected = pd.DataFrame([df_frag[lead]-df_baseline[lead] for lead in leads]).T

    vector_mag = np.sqrt((df_frag.i).astype(float)**2 + (df_frag.aVF).astype(float)**2) #Vector magnitudes at each time point
    ind_max_mag = vector_mag.argmax() #Time point with the largest vector magnitude
    amp_max_mag = vector_mag[ind_max_mag] #Corresponding value of the largest vector magnitude
    i_max = df_frag.i[ind_max_mag] #Corresponding value along lead i
    aVF_max = df_frag.aVF[ind_max_mag] #Corresponding value along lead aVF

    #Use the hexaxial reference system to calculate the angle of the vector with the largest magnitude (aVF = negative y-axis, i = positive x-axis!)
    #Option 1
    if i_max>0 and aVF_max>0:
        alpha = math.atan(aVF_max/i_max)*180/math.pi
    #Option 2
    elif i_max>0 and aVF_max<0:
        alpha = 360 - math.atan(-aVF_max/i_max)*180/math.pi
    #Option 3
    elif i_max<0 and aVF_max<0:
        alpha = 180 + math.atan(-aVF_max/-i_max)*180/math.pi
    #Option 4
    elif i_max<0 and aVF_max>0:
        alpha = 90 + math.atan(-i_max/aVF_max)*180/math.pi

    dot_prod = np.dot(np.array([i_max, aVF_max]), np.array([df_frag.i, df_frag.aVF])) #Dot product of max vector and the vectors at each time point
    df_frag['fictive_lead'] = dot_prod/amp_max_mag #Calculate the components of the max vector along the new fictive lead

    warnings.filterwarnings("ignore", category=UserWarning) #Ignore unfixable warning

    fictive_lead_original = df_frag.fictive_lead

    if not np.max(fictive_lead_original)>abs(np.min(fictive_lead_original)): #Check whether R peaks are on positive or negative y-axis
        fictive_lead = -fictive_lead_original
    else:
        fictive_lead = fictive_lead_original

    peaks,_ = find_peaks(fictive_lead, height=0) #Detect positive peak locations
    prominences = signal.peak_prominences(fictive_lead, peaks)[0] #Calculate prominences of positive peaks

```

```
    gmm = GaussianMixture(n_components=2).fit(prominences.reshape(-1,1)) #Curve fitting model
    curve_x = np.linspace(prominences.min(),prominences.max(),len(prominences)).reshape(-1,1) #Create x-values from the start to end value of prominences
    curve_y = pd.Series(np.exp(gmm.score_samples(curve_x))) #Score_samples computes the log-likelihood of each sample --> e^ln(x) = x, with x = likelihood

    prominence_peaks,_ = find_peaks(curve_y, height=0) #Indices of the peaks in the prominence curve
    peaks_sorted = np.argsort(curve_y[prominence_peaks]) #Indices of the peaks, sorted based on corresponding y-values

    if not len(prominence_peaks)>1: #Check whether the prominence curve indeed has at least two peaks
        prominence_lim = curve_x[np.argmin(curve_y),0]
    else:
        two_peaks = peaks_sorted.iloc[-2:].index #Select indices of two largest peaks
        min_index = two_peaks.min() + np.argmin(curve_y[two_peaks.min():two_peaks.max()]) #Index of minimum between the two peaks
        prominence_lim = curve_x[min_index,0] #Corresponding prominence that is set as limit

    r_peaks,_ = find_peaks(fictive_lead, height=0, prominence=prominence_lim) #Find r peaks based on computed prominence limit
    df_frag.loc[r_peaks,'Rpeaks'] = True #Assign 'True' label on R peak locations

    if len(r_peaks)<90:
        mean_HR=90

    last_i = 0 #Initialise

    #Remove peaks that are within 40 samples (=300bpm) of each other
    for i,ind in enumerate(r_peaks):
         if i==0:
             last_i = ind
             continue
         else:
             if (ind-last_i)<40:
                 if fictive_lead[ind]>fictive_lead[last_i]: #If the second peak is larger
                     df_frag.loc[last_i,'Rpeaks'] = False #Remove the first peak
                     last_i = ind
                 else: #If the first peak is larger
                     df_frag.loc[ind,'Rpeaks'] = False #Remove the second peak
             else:
                 last_i = ind
                 pass
    r_peaks = df_frag[df_frag['Rpeaks']].index.tolist()

    # Q and S and detection based on R location
    min_q = -12
    max_q = -1

    min_s = 1
    max_s = 12

    for r in r_peaks:
        """Q-peak"""
        start_q = r + min_q
        if not start_q<0: #Check whether the start Q is within the time frame
            end_q = r + max_q
            q_peaks,_ = find_peaks(-fictive_lead[start_q:end_q]) #Find negative peaks
            if q_peaks.any(): #Check whether any peaks were found
                q_peak = q_peaks[-1] #Select last possible peak
            else:
                q_peak = np.argmin(fictive_lead[start_q:end_q]) #Select lowest point within the time range
            df_frag.loc[start_q+q_peak,'Qpeaks'] = True #Assign 'True' label on Q peak locations
        else:
            continue

        """S-wave"""
        start_s = r + min_s
```

```
        end_s = r + max_s
        if not end_s>len(fictive_lead): #Check whether the last S is within the time frame
            s_peaks,_ = find_peaks(-fictive_lead[start_s:end_s]) #Find negative peaks
            if s_peaks.any(): #Check whether any peaks were found
                s_peak = s_peaks[np.argmin(fictive_lead[start_s+s_peaks])] #Select lowest peak
            else:
                s_peak = np.argmin(fictive_lead[start_s:end_s])
            df_frag.loc[start_s+s_peak,'Speaks'] = True #Assign 'True' label on S peak locations
        else:
            continue

    q_peaks = df_frag.Qpeaks[df_frag.Qpeaks].index
    s_peaks = df_frag.Speaks[df_frag.Speaks].index

    # T wave detection
    signal_without_qrs = df_corrected[['i','aVF']].copy()
    for i,r in enumerate(s_peaks):
        if len(q_peaks)<len(s_peaks) and i == len(s_peaks)-1:
            break
        else:
            if s_peaks[i]>q_peaks[i]:
                if i==len(s_peaks)-1:
                        signal_without_qrs.loc[q_peaks[i]-15:len(signal_without_qrs)]=0
                else:
                    signal_without_qrs.loc[q_peaks[i]-15:s_peaks[i]+12]=0
                    signal_without_qrs.loc[0:q_peaks[0]]=0
            else:
                if len(q_peaks) == len(s_peaks) and i == len(s_peaks)-1:
                    break
                else:
                    if i==len(s_peaks)-1:
                        signal_without_qrs.loc[q_peaks[len(q_peaks)-1]:len(signal_without_qrs)]=0
                    else:
                        signal_without_qrs.loc[0:s_peaks[0]+12]=0
                        signal_without_qrs.loc[q_peaks[i]-15:s_peaks[i+1]+12]=0

    vector_t = np.sqrt((signal_without_qrs.i).astype(float)**2 + (signal_without_qrs.aVF).astype(float)**2)
    amp_max_t = vector_t.max()
    ind_max_t = vector_t.idxmax()
    x_vector_t = signal_without_qrs.i[ind_max_t]
    y_vector_t = signal_without_qrs.aVF[ind_max_t]

    if x_vector_t>0 and y_vector_t>0:
        alpha_t = math.atan(y_vector_t/x_vector_t)*180/math.pi
    elif x_vector_t>0 and y_vector_t<0:
        alpha_t = 360 - math.atan(-y_vector_t/x_vector_t)*180/math.pi
    elif x_vector_t<0 and y_vector_t>0:
        alpha_t = 90 + math.atan(-x_vector_t/y_vector_t)*180/math.pi
    elif x_vector_t<0 and y_vector_t<0:
        alpha_t = 180 + math.atan(-y_vector_t/-x_vector_t)*180/math.pi

    dot_prod_t = np.dot(np.array([x_vector_t, y_vector_t]), np.array([signal_without_qrs.i, signal_without_qrs.aVF])) #Dot product of max vector and the vectors at each time point
    df_frag['fictive_lead_t'] = dot_prod_t/amp_max_t #Calculate the components of the max vector along the new fictive lead

    fictive_lead_original_t = df_frag.fictive_lead_t

    if not np.max(fictive_lead_original_t)>abs(np.min(fictive_lead_original_t)): #Check whether R peaks are on positive or negative y-axis
        fictive_lead_t = -fictive_lead_original_t
    else:
        fictive_lead_t = fictive_lead_original_t

    for i,r in enumerate(s_peaks):
        if len(q_peaks)<len(s_peaks) and i == len(s_peaks)-1:
            break
        else:
```

```
            if q_peaks[i]>s_peaks[i]:
                t_wave_fict = np.argmax(fictive_lead_t[s_peaks[i]:q_peaks[i]])+
s_peaks[i]
            else:
                if len(q_peaks) == len(s_peaks) and i == len(s_peaks)-1:
                    break
                else:
                    t_wave_fict = np.argmax(fictive_lead_t[s_peaks[i]:q_peaks[i
+1]])+s_peaks[i]
        t_wave = t_wave_fict
        df_frag.loc[t_wave,'Tpeaks'] = True
    t_peaks = df_frag.Tpeaks[df_frag.Tpeaks].index

    from scipy.ndimage import gaussian_filter1d
    timestamps = df_frag["Timestamps"]

    qpeak_start = []
    for i,_ in enumerate(t_peaks):
        if not r_peaks[i+1]>t_peaks[i]:
            x=timestamps[t_peaks[i]:r_peaks[i+2]]
            y=df_frag.fictive_lead[t_peaks[i]:r_peaks[i+2]]
        else:
            x=timestamps[t_peaks[i]:r_peaks[i+1]]
            y=df_frag.fictive_lead[t_peaks[i]:r_peaks[i+1]]
        ysmoothed = pd.Series(gaussian_filter1d(y,sigma=2))
        derivative = ysmoothed.diff()
        double = derivative.diff()
        old_index = q_peaks[i+1]
        new_index = x.index.get_loc(old_index)
        x_zoekgebied = x.loc[old_index::-1]
        y_zoekgebied = double.loc[new_index::-1]
        cross_0 = np.where(np.diff(np.sign(y_zoekgebied)))[0][0]
        xm = old_index-cross_0 #To be able to subtract (not possible with timeframe
)
        xm_time = x[xm] #To be able to plot
        ym = y[xm]
        xr = x.index[0]
        xr_time = x[xr]
        yr = y[xr]
        oppervlakte = []
        for j in range(xm,xr,-1):
            xi = j
            yi = y[xi]
            oppervlakte.append(0.5*(yi-ym)*(xm+xi-(2*xr)))
        opp_x = list(range(xm,xr,-1))
        opp_peaks,_ = find_peaks(oppervlakte)
        if len(opp_peaks)==0:
            qpeak_start.append(t_peaks[i]+1)
        else:
            prominences_opp = signal.peak_prominences(oppervlakte, opp_peaks)[0]
            peaks_sorted_opp = np.argsort(prominences_opp)
            two_peaks_opp = np.min(peaks_sorted_opp[-2:])
            q_start_ind=opp_peaks[two_peaks_opp]
            qpeak_start.append(xm - q_start_ind)

    from scipy.interpolate import CubicSpline
    twave_end=[]
    for i,_ in enumerate(t_peaks):
        x=timestamps[s_peaks[i]:qpeak_start[i]]
        y=df_frag.fictive_lead_t[s_peaks[i]:qpeak_start[i]]
        ysmoothed = pd.Series(gaussian_filter1d(y,sigma=2))
        derivative = ysmoothed.diff()
        double = derivative.diff()
        old_index = t_peaks[i]
        new_index = x.index.get_loc(old_index)
        x_zoekgebied = x.loc[old_index::]
        y_zoekgebied = double.loc[new_index::]
        if not any(np.diff(np.sign(y_zoekgebied))):
            twave_end.append(t_peaks[i])
        else:
            cross_0 = np.where(np.diff(np.sign(y_zoekgebied)))[0][0]
            xm = old_index+cross_0 #To be able to subtract (not possible with
timeframe)
            xm_time = x[xm] #To be able to plot
            ym = y[xm]
            xr = x.index[-1]
```

```
            xr_time = x[xr]
            yr = y[xr]
            oppervlakte = []
            for j in range(xm,xr):
                xi = j
                yi = y[xi]
                oppervlakte.append(0.5*(ym-yi)*(2*xr-xi-xm))
            max_opp = np.argmax(oppervlakte)
            twave_end.append(xm+max_opp)

    signal_without_qrst = df_corrected[['i','aVF']].copy()
    for i,r in enumerate(twave_end):
        if len(qpeak_start)<len(twave_end) and i == len(twave_end)-1:
            break
        else:
            if twave_end[i]>qpeak_start[i]:
                if  i == len(twave_end)-1:
                    signal_without_qrst.loc[qpeak_start[i]:len(signal_without_qrst)
    ]=0
                else:
                    signal_without_qrst.loc[qpeak_start[i]:twave_end[i]]=0
                    signal_without_qrst.loc[0:qpeak_start[0]]=0
            else:
                if len(qpeak_start) == len(twave_end) and i == len(twave_end):
                    break
                else:
                    if i==len(twave_end)-1:
                        signal_without_qrst.loc[qpeak_start[len(qpeak_start)-1]:len
    (signal_without_qrst)]=0
                    else:
                        signal_without_qrst.loc[0:twave_end[0]]=0
                        signal_without_qrst.loc[qpeak_start[i]:twave_end[i+1]]=0

    vector_p = np.sqrt((signal_without_qrst.i).astype(float)**2 + (
    signal_without_qrst.aVF).astype(float)**2)
    amp_max_p = vector_p.max()
    ind_max_p = vector_p.idxmax()
    x_vector_p = signal_without_qrst.i[ind_max_p]
    y_vector_p = signal_without_qrst.aVF[ind_max_p]

    if x_vector_p>0 and y_vector_p>0:
        alpha_p = math.atan(y_vector_p/x_vector_p)*180/math.pi
    elif x_vector_p>0 and y_vector_p<0:
        alpha_p = 360 - math.atan(-y_vector_p/x_vector_p)*180/math.pi
    elif x_vector_p<0 and y_vector_p>0:
        alpha_p = 90 + math.atan(-x_vector_p/y_vector_p)*180/math.pi
    elif x_vector_p<0 and y_vector_p<0:
        alpha_p = 180 + math.atan(-y_vector_p/-x_vector_p)*180/math.pi

    dot_prod_p = np.dot(np.array([x_vector_p, y_vector_p]), np.array([
    signal_without_qrst.i, signal_without_qrst.aVF])) #Dot product of max vector
    and the vectors at each time point
    df_frag['fictive_lead_p'] = dot_prod_p/amp_max_p #Calculate the components of
    the max vector along the new fictive lead

    fictive_lead_original_p = df_frag.fictive_lead_p

    if not np.max(fictive_lead_original_p)>abs(np.min(fictive_lead_original_p)): #
    Check whether R peaks are on positive or negative y-axis
        fictive_lead_p = -fictive_lead_original_p
    else:
        fictive_lead_p = fictive_lead_original_p

    warnings.filterwarnings("ignore", category=UserWarning) #Ignore unfixable
    warning

    peaks,_ = find_peaks(fictive_lead_p, height=0) #Detect positive peak locations
    prominences = signal.peak_prominences(fictive_lead_p, peaks)[0] #Calculate
    prominences of positive peaks

    gmm = GaussianMixture(n_components=2).fit(prominences.reshape(-1,1)) #Curve
    fitting model
    curve_x = np.linspace(prominences.min(),prominences.max(),len(prominences)).
    reshape(-1,1) #Create x-values from the start to end value of prominences
    curve_y = pd.Series(np.exp(gmm.score_samples(curve_x))) #Score_samples computes
     the log-likelihood of each sample --> e^ln(x) = x, with x = likelihood
```

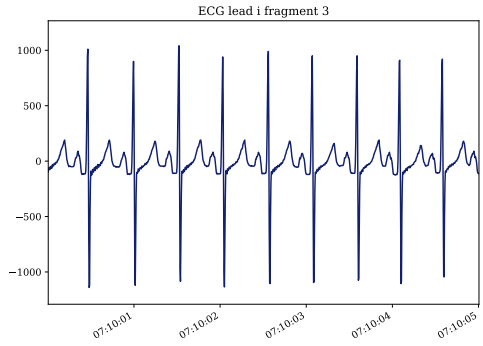
```
384
385     prominence_peaks,_ = find_peaks(curve_y, height=0) #Indices of the peaks in the
         prominence curve
386     peaks_sorted = np.argsort(curve_y[prominence_peaks]) #Indices of the peaks,
        sorted based on corresponding y-values
387
388     if not len(prominence_peaks)>1: #Check whether the prominence curve indeed has
        at least two peaks
389         prominence_lim = curve_x[np.argmin(curve_y),0]
390     else:
391         two_peaks = peaks_sorted.iloc[-2:].index #Select indices of two largest
        peaks
392         min_index = two_peaks.min() + np.argmin(curve_y[two_peaks.min():two_peaks.
        max()]) #Index of minimum between the two peaks
393         prominence_lim = curve_x[min_index,0] + 50  #Corresponding prominence that
        is set as limit
394
395     p_peaks,properties = find_peaks(fictive_lead_p, height=0, prominence=
        prominence_lim) #Find r peaks based on computed prominence limit
396
397     pr_interval=[]
398     for i,p in enumerate(r_peaks):
399         if i==len(r_peaks)-1:
400             break
401         else:
402             pr = p_peaks[(p_peaks>=p) & (p_peaks<=r_peaks[i+1])]
403             if len(pr)==1:
404                 pr_interval.append(r_peaks[i+1]-pr[0])
405             elif len(pr)>1:
406                 ind_max_p=np.argmax(fictive_lead_p[pr])
407                 del_ppeaks=np.delete(pr,ind_max_p)
408                 p_peaks=np.setdiff1d(p_peaks,del_ppeaks)
409                 pr_interval.append(r_peaks[i+1]-pr[ind_max_p])
410             else:
411                 continue
412
413     df_features.loc[fragment,'PR_SD'] = np.std(pr_interval)
414
415     df_frag.loc[p_peaks,'Pwaves'] = True #Assign 'True' label on R peak locations
416
417     if p_peaks.any():
418         df_features.loc[fragment,'frac_ppeaks'] = (len(p_peaks)/len(r_peaks))
419     else:
420         df_features.loc[fragment,'frac_ppeaks'] = 0
421         df_features.loc[fragment,'PR_SD'] = 0
422
423     # Calculate RR interval
424     df_r = df_frag[df_frag['Rpeaks']] #DF with only true R peak rows
425     df_r.loc[:,'RR_inv']=df_r.index.diff()
426     df_frag.loc[df_r.index,'RR_inv'] = df_r.loc[:,'RR_inv']
427
428     fs = 200 #Sample frequency in Hz
429
430     df_frag['HR'] = fs*60/(df_frag['RR_inv'])
431     df_frag['HR'].interpolate(method='linear', inplace=True)
432
433     df_features.loc[fragment,'mean_HR'] = np.mean(df_frag['HR'])
434
435     mean_vector = []
436     for peak in q_peaks:
437         mean_vector.append(np.mean(vector_mag[peak:peak+10]))
438     df_features.loc[fragment,'VCG_SD']=np.std(mean_vector)
439
440 for i,row in df_features.iterrows():
441     df_features.loc[i,'VCG_SD_sin'] = df_features.VCG_SD[row['Fragment_sin']]
442     df_features.loc[i,'frac_ppeaks_sin'] = df_features.frac_ppeaks[row['
        Fragment_sin']]
443     df_features.loc[i,'PR_SD_sin'] = df_features.PR_SD[row['Fragment_sin']]
444
445     if row['mean_HR']>170:
446         predicted_rhythm = "JET"
447     else:
448         if row['VCG_SD']>1.5*(row['VCG_SD_sin']):
449             predicted_rhythm = "JET"
450         else:
451             if (row['frac_ppeaks'] == 0) or not (row['frac_ppeaks']<row['
```

```
    frac_ppeaks_sin']):
                if row['PR_SD']>row['PR_SD_sin']:
                    predicted_rhythm = "JET"
                else:
                    predicted_rhythm = "Sinus"
            else:
                predicted_rhythm = "JET"

    if row['Rhythm'] == predicted_rhythm:
        predicted = 1
    else:
        predicted = 0

    filename = 'predictions.xlsx'

    if os.path.exists(filename):
        predictions = pd.read_excel(filename,index_col='Fragment')
    else:
        predictions = pd.DataFrame(columns=['Prediction','Real','Predicted'])
        predictions.index.name = 'Fragment'

    predictions.loc[int(fragment),'Prediction'] = predicted
    predictions.loc[int(fragment),'Real'] = row['Rhythm']
    predictions.loc[int(fragment),'Predicted'] = predicted_rhythm

    predictions = predictions.sort_index()
    predictions.to_excel(filename)

true_positive = len(predictions.query('Prediction==1 & Predicted=="JET"'))
tp_fn = len(predictions.query('Real=="JET"'))
if tp_fn==0:
    print('No JET patients')
else:
    print(f"Sensitivity: {round(true_positive/tp_fn*100,2)}%")

true_negative = len(predictions.query('Prediction==1 & Predicted=="Sinus"'))
fp_tn = len(predictions.query('Real=="Sinus"'))
if fp_tn==0:
    print('No sinus patients')
else:
    print(f"Specificity: {round(true_negative/fp_tn*100,2)}%") #Percentage van alle
     waarschuwingen, dus niet percentage van totaal aantal fragmenten

tp_fp = len(predictions.query('Predicted=="JET"'))
if tp_fp==0:
    print('No warnings')
else:
    print(f"PPV: {round(true_positive/tp_fp*100,2)} %") #Perecntage dat ook echt
    JET is bij een alarm

tp_tn = true_positive+true_negative
total = len(predictions)
if total == 0:
    print('No patients')
else:
    print(f"Accuracy: {round(tp_tn/total*100,2)}%")
```

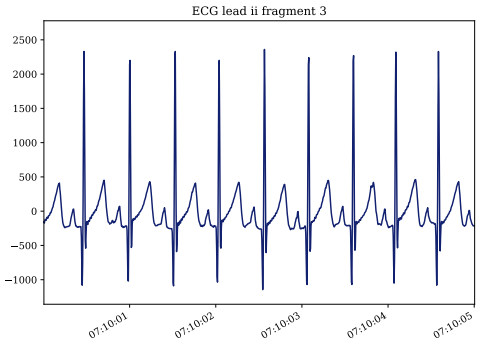
Appendix B: Python libraries

- os
- scipy
- math
- numpy
- pandas
- matplotlib
- datetime
- sklearn
- pybaselines
- warnings

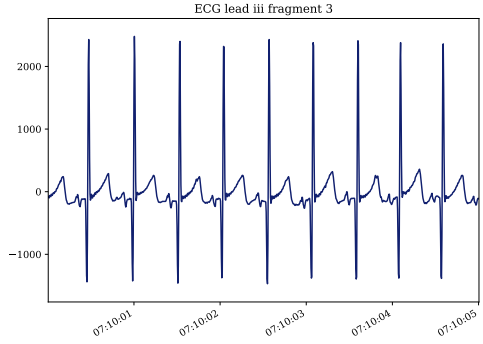
Appendix C: Supplementary figures



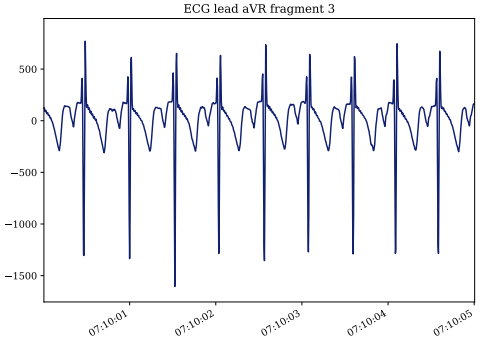
(a) Lead I



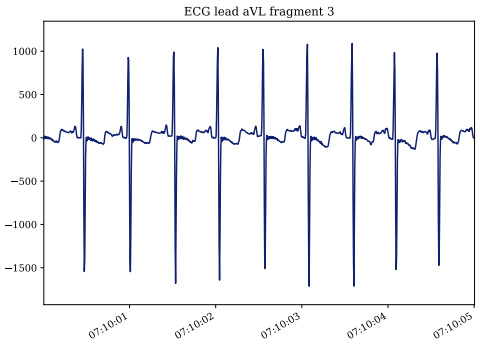
(b) Lead II



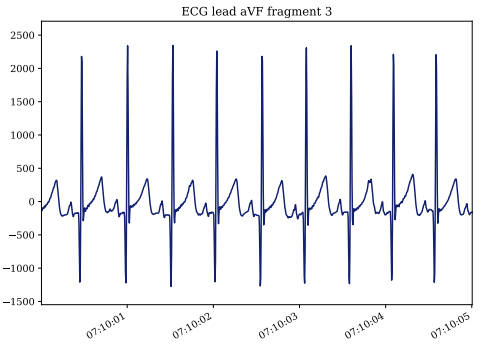
(c) Lead III



(d) Lead aVL

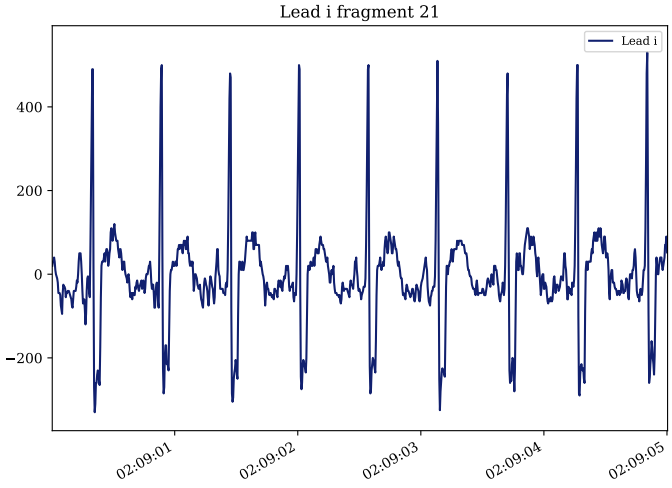


(e) Lead aVR

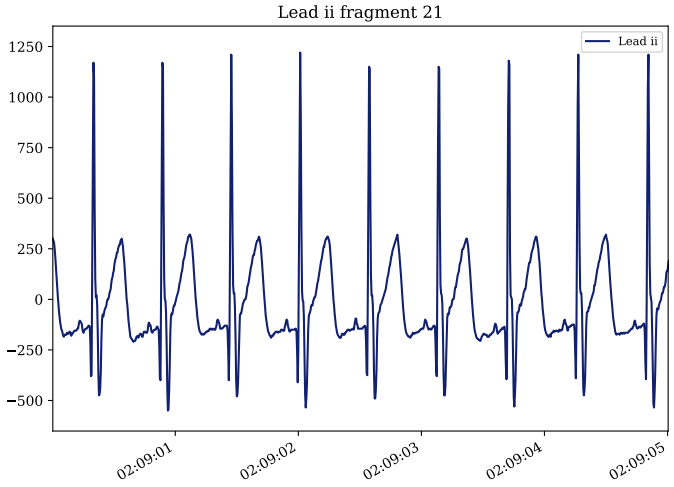


(f) Lead aVF

Figure 14: Example of leads I, II, III, aVL, aVR and aVF of a fragment to be annotated



(a) Lead i (with artifact)



(b) Lead ii (without artifact)

Figure 15: Artifact in lead i of fragment 21

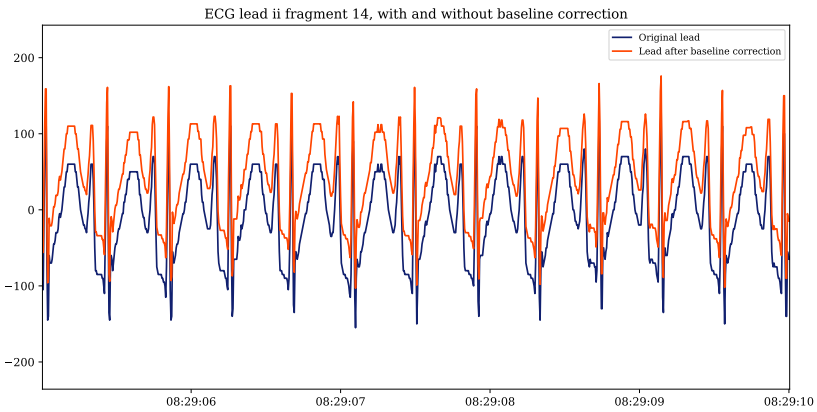
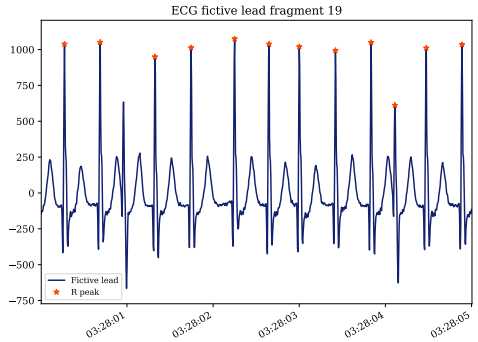
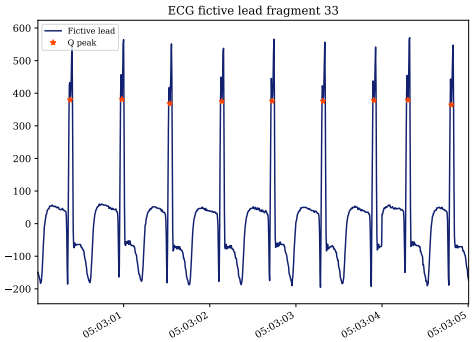


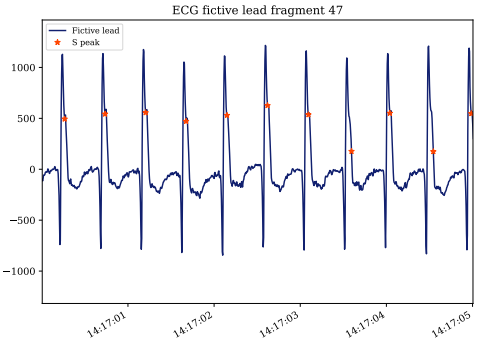
Figure 16: Baseline correction



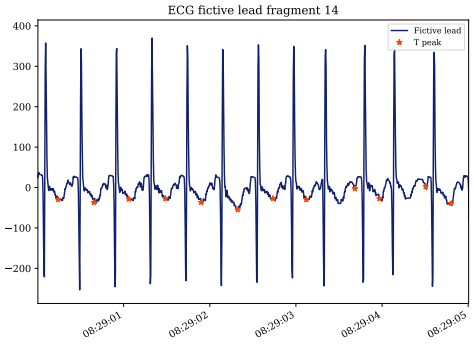
(a) Wrong R peak detection



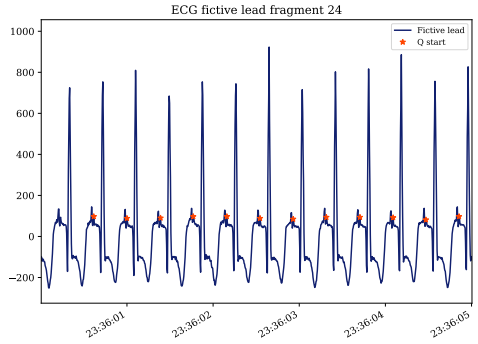
(b) Wrong Q peak detection



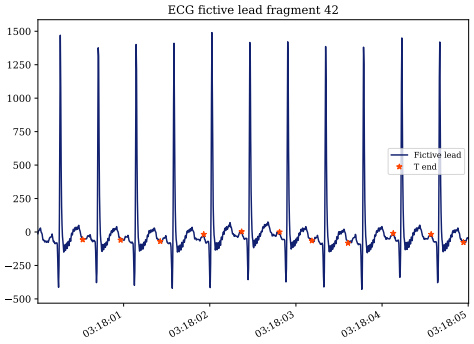
(c) Wrong s peak detection



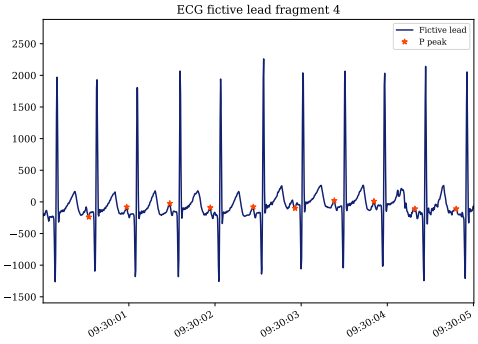
(d) Wrong t peak detection



(e) Wrong Q start detection



(f) Wrong T end detection



(g) Wrong P detection

Figure 17: Examples of wrong detections of ECG metrics