Real-Time, Nonparametric Algorithm to Learn Parameters for Pacemaker Beat Detection

Yamin Arefeen*, Daniel Zdeblick*, Philip Taffet†, Jorge Quintero*, Greg Harper*,
Benhaam Aazhang*, and Joseph Cavallaro*
*Department of Electrical and Computer Engineering, †Department of Computer Science
Rice University, Houston, TX 77005
yarefeen@mit.edu, zdeblick@uw.edu, ptaffet@rice.edu, aaz@rice.edu, cavallar@rice.edu

***Abstract*—While beat detection in an Electrocardiogram (ECG) signal is a well-studied problem, we propose a novel algorithm, within the context of pacemakers, that learns the heights and widths of atrial and ventricular peaks from simply processing 10 seconds of ECG data sampled at 1 kHz. Utilizing a purely data-driven solution to learn the parameters of atrial and ventricular peaks will allow pacemakers to set their own detection parameters for a specific patient and adaptively tune their detection parameters, for that specific patient, over time. We have validated these results on 51 separate channels of ECG data. Additionally, we have implemented the algorithm on a Field Programmable Gate Array and tested it on a Langdendorff heart to illustrate that our algorithm can be implemented on hardware and run in real time.**

I. Introduction

Cardiac diseases are the number one cause of death in the United States. More than 600,000 people each year die due to some failure in the heart [1]. A healthy heart functions through a regular cardiac cycle, where the pace making neurons in the sinoatrial (SA) node of the heart generate an electrical signal, which travels from the atria down to the ventricles, to stimulate heart contraction. Many heart failures result from the heart's inability to generate or conduct these electrical signals and stimulate muscle contraction properly. To combat this, a combination of researchers and doctors developed a device called the pacemaker that can stimulate the heart to contract artificially through externally supplied electrical pulses. Current pacemakers utilize the state of the art algorithm called DDDR in order to determine whether the heart requires stimulation [2]. The devices utilize ECG beat detection algorithms to determine whether a heartbeat has occurred. If it does not see a heartbeat within a certain period of time, it will deliver an electrical stimulation to the heart. ECG beat detection itself is a well-studied problem, as several algorithms have been proposed to identify the beats within an ECG signal [3], [4]. However, current algorithms perform beat detection without distinguishing which part of the ECG signal corresponds to the atrial portion of the heartbeat and which part corresponds to the ventricular portion of the heartbeat. As a result, current pacemakers often times struggle to distinguish between atrial and ventricular beats: information that is crucial for Cardiac Resynchronization Therapy (CRT) [5]. In fact, up to 30% of patients with pacemakers do not respond to CRT implemented by current pacemakers [5]. To make ECG beat detection more suitable for CRT, we propose a nonparametric algorithm that learns the height of an atrial peak (*APH*), the height of a ventricular peak (*VPH*), the width of an atrial peak (*APW*), and the width of a ventricular peak (*VPW*) within an ECG signal by simply processing 10 seconds of ECG data sampled at 1 kHz. With these parameters, we then implement atrial and ventricular beat detection to illustrate that our learned parameters can be used to distinguish between atrial and ventricular beats within an ECG signal. Additionally, we implement the algorithm on a Field Programmable Gate Array to demonstrate that the algorithm can be run on a real-time system. Sections II and III thoroughly explain our approach for learning *APH*, *VPH*, *APW*, and *VPW*. Section IV details the hardware implementation of the algorithm. Finally, Section V documents our testing procedure and validates our algorithmic results and hardware system.

II. Atrial and Ventricular Peak Width Learning

Our algorithm begins by learning *APW* and *VPW*. We detail our approach in the following sections.

A. Finding Peak Shaped Patterns in the ECG Signal

Let $f[n]$ be a 10 second ECG signal sampled at 1 kHz. We begin by computing the weighted time derivative of $f[n]$, denoted $w[n]$.

$$w[n] = (f[n] - f[n-1]) * |f[n] - f[n-1]| * |f[n]|$$

Next, the algorithm searches for when the local maxima and minima of $w[n]$ occur, within a window of 200 ms. Thus, we compute:

$$r[n] = \begin{cases} 1 & \text{if } w[n] = \max\limits_{-100 \le i \le 100}(w[n+i]) \\ -1 & \text{if } w[n] = \min\limits_{-100 \le i \le 100}(w[n+i]) \\ 0 & \text{otherwise} \end{cases}$$

Note, that a value of 1 in $r[n]$ corresponds to the steepest rising edge of peaks in $f[n]$, while a value of -1 in $r[n]$ corresponds to the steepest falling edge of a peak in $f[n]$. Thus, $r[n]$ identifies all peak like structures in $f[n]$. Fig. 1 illustrates this concept.

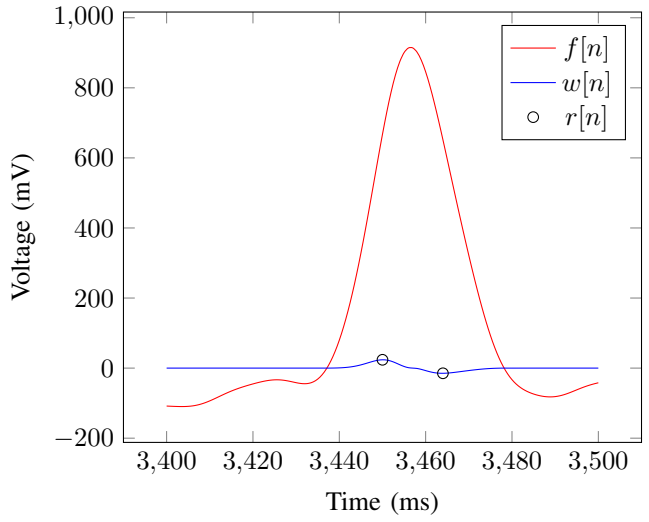

Fig. 1. Illustration of the relationship between $f[n]$, $w[n]$ and $r[n]$

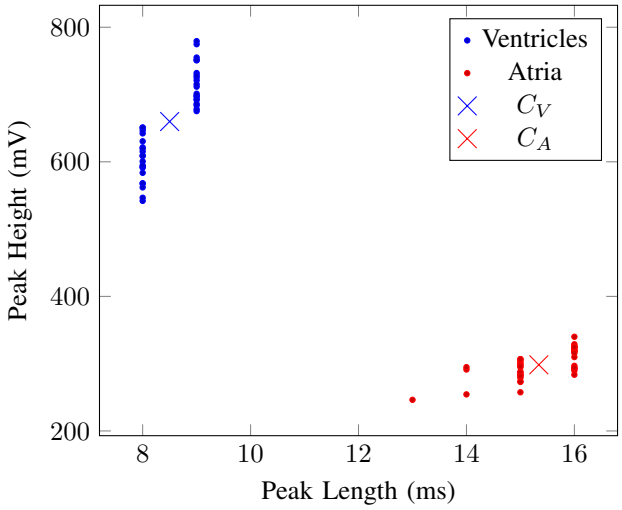

Fig. 2. K-Means clustering featurized peaks into ventricles and atria.

B. Featurizing Peaks and Clustering to Determine APW and VPW

Using $r[n]$, we identify a peak, P_i, to be a rising edge followed in less than 75ms by a falling edge. Let the rising edge and falling edges of P_i occur at time m and n respectively. The peak is then featurized by its width and height as follows:

$$P_i = \left\{ n - m, \left(\max_{m \le j \le n} f[j] \right) - \frac{f[n] + f[m]}{2} \right\}$$

Thus a two-dimensional data point characterizes each peak. We then construct $\mathbf{P} = [P_1, P_2, \ldots P_n]$ and then cluster the peaks in $\mathbf{P}$ using a standard two cluster k-means algorithm [6]. The ventricular cluster will have a center with a greater height to width ratio, since ventricles are generally taller and thinner. Let C_v and C_A be the centers of the ventricular and atrial clusters respectively. The algorithm then stores the ventricular peak width (*VPW*) and atrial peak width (*APW*) as the rounded width term of the centers as follows:

$$VPW = C_V\{1\} \text{ and } APW = V_A\{1\}$$

A visual illustration of how k-means clusters the featurized peaks $\mathbf{P}$ into atrial and ventricular peaks and determines the resulting centroids C_V and C_A can be found in Fig. 2.

Note, that the height that we use to featurize a peak in this portion of the algorithm does not correspond to the actual values of *APH* and *VPH* in the original ECG signal $f[n]$. The 'height' value used for peak featurization simply characterizes the height of the peak with respect to the portions of the peak during which its value is most rapidly increasing or decreasing. It does not give any useful information about the height of an atrial or ventricular peak with respect to the baseline value of 0.

III. Atrial and Ventricular Peak Height (*VPH* and *APH*) Learning

After learning *APW* and *VPW*, we proceed on the same segment of ECG signal to learn *VPH* and *APH*. In the following section, we give the intuition behind the algorithm and detail our approach.

A. Overarching Intuition for Learning VPH and APH

Again, consider $f[n]$. The basic strategy of the peak height threshold learning is to look for a large range of thresholds that detects the same number of QRS beats in $f[n]$. This idea follows from the observation that there is typically a wide range of thresholds that are high enough to exclude the lower (typically atrial) beats but low enough to detect each of the higher (typically ventricular) beats. As the algorithm tries thresholds lower than that range, it starts to detect lower beats, and as it tries thresholds higher than that range, it stops detecting some of the higher beats, so the number of detected beats begins to change. In the range, however, the number of detected beats is constant.

$$beats(threshold) = \text{Number of beats detected in } f[n] \text{ using } threshold \quad (1)$$

The algorithm looks for the longest and flattest interval of the function $beats(threshold)$. An example of $beats(threshold)$ is illustrated in Fig. 3. However, computing the number of beats we detect with a given threshold, takes a substantial amount of computation, thus we utilize a semi-recursive iterative approach in our algorithm. We name our algorithm "Flat-Finding Ventricular and Atrial Heights."

A key property of $beats(threshold)$ is its monotonicity.

B. Flat Finding Algorithm to determine VPH and APH

As mentioned in the previous section, computing $beats(threshold)$ for a given value of $threshold$ requires a

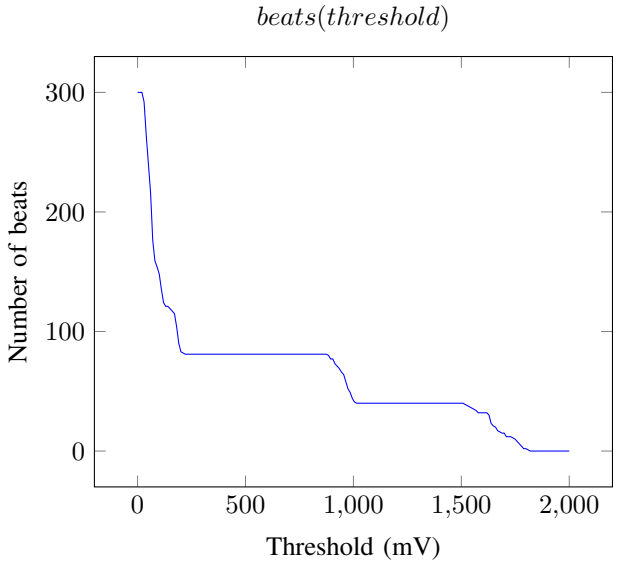

Fig. 3. Graph of the function $beats(threshold)$ on a sample $f[n]$

great deal of computation. Instead, we run several iterations in which we pick thresholds th_i for $i = 1, 2, \ldots, 20$ evenly spaced between a minimum and maximum threshold, compute $beats(th_i)$ for each i, and adjust the minimum and maximum thresholds for the next iteration based on the results. For the first iteration, we set $th_{max} = \max f[n]$ and $th_{min} = 0$. To choose the minimum and maximum thresholds in the next round, we first eliminate any thresholds th_i such that $beats(th_i)$ is biologically infeasible. For a human, we require that 10bpm $\leq beats(th_i) \leq$ 200bpm. Note that $beats(th_i)$ is not natively computed in the unit of beats per minute, so a simple unit conversion is required.

Then we define a maximal flat interval $[i, j]$ with $i < j$ to be one that satisfies the following conditions:

$$\begin{aligned} &beats(th_i) = beats(th_j), \\ &beats(th_{i-1}) \neq beats(th_i), beats(th_j) \neq beats(th_{j+1}) \end{aligned} \quad (2)$$

Because of the monotonicity of $beats(threshold)$, the first condition is necessary and sufficient for the flatness property. The second and third conidtions ensure that $[i, j]$ is maximal.

If one or more flat intervals exist, choose $th_{max} = th_{j+1}$ and $th_{min} = th_{i-1}$ using the longest interval (i.e. the one that maximizes $j-i$). Because of the maximality of the flat interval $[i, j]$ and the monotonicity of $th(threshold)$, we know that the true end of the flat interval is somewhere between th_{i-1} and th_i. Choosing $th_{min} = th_{i-1}$ instead of th_i, gives the next recursive refinement iteration a chance to identify more closely the end of the flat interval. Similar logic applies to our choise of th_{max}.

If no flat intervals exist, set $i = \arg\min_i |beats(th_i) - beats(th_{i-1})|$. Then choose $th_{max} = th_i$, $th_{min} = th_{i-1}$.

After 3 recursive refinement iterations, define the ventricular peak height (*VPH*) to be the midpoint of the flat interval, i.e.

$$VPH = \frac{th_{min} + th_{max}}{2}$$

Finally, after computing the ventricular thresholds, data near the detected ventricular peaks is replaced with zeros, and the entire algorithm in re-run to detect the atrial threshold, *APH*.

C. Precise Specification of $beats(threshold)$ using our Proposed Beat Detection Algorithm

Our 'flat-finding' algorithm to determine *APH* and VPH relies entirely on $beats(threshold)$, which we defined as the number of beats that occur in $f[n]$ using threshold to identify the beats. In order to calculate the number of beats that occur based on threshold, we utilize an algorithm that draws heavily from the ECG beat detection algorithm described in Alfonso and Tompkins [3]. In the remaining portion of this section, we describe the beat detection algorithm that we use to compute $beats(threshold)$. First define Peak Width, PW, to be the following:

$$PW = \begin{cases} VPW & \text{if computing } VPH \\ APW & \text{if computing } APH \end{cases}$$

Notice how our algorithm utilized a nonparametric approach to compute VPW and APW previously. Next, define:

$$potentialPeaks[n] = \begin{cases} 1 & \text{if } f[n] > threshold \\ APW & \text{otherwise} \end{cases}$$

$potentialPeaks[n]$ indicates where in $f[n]$ a beat could have occurred. We then use the intuition that for a beat to occur, it must have been about as long as PW in time. Thus, to compute $beats(threshold)$, i.e. the number of beats that occur in $f[n]$, we utilize the following procedure.

$$beats[n] = \begin{cases} 1 & \text{if } \left(\sum_{i=0}^{PW} potentialPeaks[n-i]\right) > PW/2 \\ 0 & \text{otherwise} \end{cases}$$

Essentially, $beats[n]$ indicates where in time beats occur in $f[n]$ according to $threshold$. In accordance with biological feasibility, we do not allow beats to occur within .27 seconds of each other. Finally, we precisely define

$$beats(threshold) = \sum_n beats[n]$$

To detect ventricular beats using this algorithm, simply utilize *VPW* and *VPH* for $threshold$ and PW respectively. Similarly, to detect atrial beats, utilize *APW* and *APH* for $threshold$ and PW.

IV. Hardware Implementation

In the future, we envision that our algorithm will be implemented on real pacemakers. In order to demonstrate that our proposed solution can be run in real-time through hardware that can be shrunk down to the size of an integrated circuit, we fully implemented our algorithm on a Field Programmable Gate Array (FPGA). In addition, to test on analog signals, we designed a preprocessing circuit board that filters and cleans analog ECG data before it is digitized and used for the algorithm implemented on the FPGA.

A. FPGA Algorithm Implementation

Though the FPGA chip is large, and far too power hungry for a finalized pacemaker, it suffices as a demonstrable proof of concept for our algorithm. In fact, we implemented our algorithm on FPGA fabric to showcase how it could be shrunk down to an application specific integrated circuit (ASIC) that could work with pacemakers in the future. An illustration of our ASIC lay out can be found in Fig. 4.

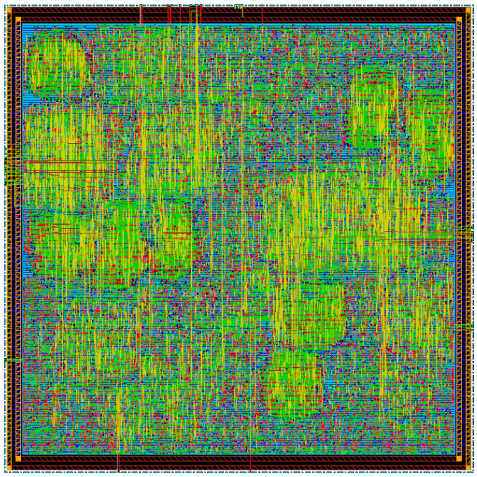

Fig. 4. ASIC Hardware Design of the algorithm implementation on the FPGA

B. Pre-processor Circuit Board Design

Since we aim to utilize our algorithm for pacemakers, we wanted to make sure that our implementation on the FPGA could interface with real analog signals. In order to do this, we designed a preprocessor board that is capable of sensing electrical activity directly from a heart and cleaning the signal so that it can properly be digitized and utilized by the algorithm implementation on the FPGA. A design drawing of our board, which utilizes many of the circuit design ideas used in [7] can be seen in Fig. 5.

The preprocessor consists of an instrumentation amplifier, followed by a DC offset removal op-amp, a non-inverting amplifier, and a 3 stage low pass filter before going into the logic controller for algorithm processing. The TI INA827 chip is an instrumentation amplifier that can take a differential signal as input, combines it to a single output, and amplifies the signal by 5. The op amp that feeds back to pin 6 is used to remove any DC offset or wandering baseline that exists in the signal. After this stage, the non-inverting amplifier amplifies the output of the instrumentation amplifier by 16. Finally, there is a 3 stage low pass filter that has a cutoff frequency at 150 Hz and a gain smaller than -40dB starting at 300 Hz. All the chips and op amps are powered with a +/- 3.3 rail voltage.

V. Testing and Results

We validate our ventricular and atrial parameter learning algorithm and our overarching hardware system with two separate experiments. The following sections will describe both tests separately.

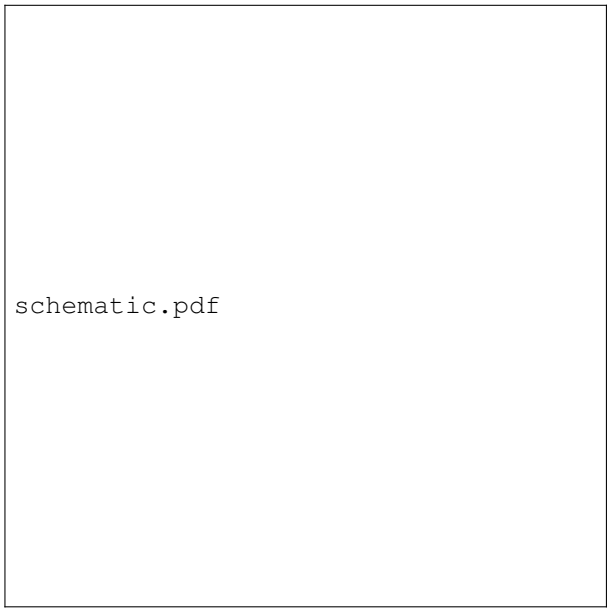

Fig. 5. Schematic for preprocessor circuit that senses and cleans analog data from the heart

A. Ventricular and Atrial Parameter Learning Validation

In Sections II and III of this paper, we presented an algorithm that learns *APW*, *VPW*, *APH*, and *VPH* by simply processing 10 seconds of ECG data sampled at 1 kHz. To validate our solution, we implemented our algorithms in Matlab and performed atrial and ventricular beat detection using algorithmically learned parameters on data provided by the Electrophysiology lab at the Texas Heart Institute. In order to record the data, an in vivo, live animal study procedure was performed. For a single animal, 17 bipolar electrodes populated the left ventricle of the heart. Each electrode sensed electrical activity directly from the heart for one minute, digitized the signal at a 1 kHz sampling rate, and stored the signal as a .csv file. This was done for 3 animals, giving us a total of 51 channels that directly resembles the kind of analog signals a real pacemaker would be sensing.

We then applied our algorithm on 10 seconds of data from each of the 51 channels separately in order to learn *APW*, *VPW*, *APH*, and *VPH* specifically for each channel. Next, using the peak detection algorithm described in Section III: C, we performed atrial and ventricular peak detection on each of the 51 channels using the aforementioned parameters that we computed for each specific channel. Finally, we computed the atrial and ventricular beat detection error rate through tracking false positives (a beat is detected when none exist) and false negatives (no beat is detected when one exists) across all 51 channels. Table I illustrates our results.

TABLE I
Detection Error Rates using Learned Parameters

False Positive Error Rate	False Negative Error Rate
0.5%	4.7 %

Our detection error rates are appropriately low across all 51 channels. Since we performed atrial and ventricular beat detection based on the parameters that our algorithm learned, the algorithm clearly learned *APW*, *VPW*, *APH*, and *VPH* correctly for the vast majority of channels. In fact, most of the false negative error rate results from our algorithm learning incorrect atrial parameters for one channel. When we manually looked at the values of *APW*, *VPW*, *APH*, and *VPH*, we found that the algorithm learned *VPW* and *VPH* correctly for all 51 channels and *APW* and *APH* correctly for 50 of the channels.

B. Hardware Implementation Validation through a Langendorff Heart

In order to validate the hardware implementation of our algorithm described in Section IV, we interfaced our system to a Langendorff Heart as described by [8]. Essentially, in a Langendorff Heart experiment, a live heart is extracted from a mammal, which in our case was a rat, and is kept beating. This externally beating heart will behave as if it was beating internally, so the Langendroff Heart will retain all of the appropriate electrical characteristics. We connected our preprocessor through simple wires sutured to the Langendroff heart, and then used our FPGA to digitize the analog signal sensed through the preprocessor. Our system correctly sensed signals using the preprocessor, learned pacing parameters (*VPW* and *VPH*), and detected ventricular beats using the learned values of *VPW* and *VPH* in real time. Due to the relatively small size of a rat's heart in comparison to a human's, the atrial part of the signal was not distinguishable on the sensed signal itself, so our algorithm did not learn any parameters for it.

VI. Conclusions and Future Direction

We have demonstrated a working algorithm implemented on hardware that correctly learns *APH*, *VPH*, *APW*, and *VPW*. Additionally, these parameters can be reliably used to distinguish between atrial and ventricular beats in an ECG signal in real time. Future work will involve utilizing the FPGA implementation of the algorithm to tape out an application specific integrated circuit that could be physically placed inside of a pacemaker.

References

[1] M. Heron and R. N. Anderson, "Changes in the leading cause of death: recent patterns in heart disease and cancer mortality," *NCHS data brief*, vol. 254, pp. 1–6, 2016.

[2] M. Kirk, "Basic principles of pacing," A. W. Chow and A. E. Buxton, Eds.

[3] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE transactions on biomedical engineering*, no. 3, pp. 230–236, 1985.

[4] V. X. Afonso, W. J. Tompkins, T. Q. Nguyen, and S. Luo, "Ecg beat detection using filter banks," *IEEE transactions on biomedical engineering*, vol. 46, no. 2, pp. 192–202, 1999.

[5] C. A. Rinaldi, H. Burri, B. Thibault, A. Curnis, A. Rao, D. Gras, J. Sperzel, J. P. Singh, M. Biffi, P. Bordachar *et al.*, "A review of multisite pacing to achieve cardiac resynchronization therapy," *EP Europace*, vol. 17, no. 1, p. 7, 2015.

[6] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[7] K. Soundarapandian and M. Berarducci, "Analog front-end design for ecg systems using delta-sigma adcs," *TI Rep. SBAA160A*, pp. 1–11, 2010.

[8] M. Skrzypiec-Spring, B. Grotthus, A. Szelag, and R. Schulz, "Isolated heart perfusion according to langendorff–still viable in the new millennium," *Journal of pharmacological and toxicological methods*, vol. 55, no. 2, pp. 113–126, 2007.