# Real-Time, Data-Driven System to Learn Parameters for Multisite Pacemaker Beat Detection

Yamin Arefeen*, Philip Taffet†, Daniel Zdeblick*, Jorge Quintero*, Greg Harper*,
Behnaam Aazhang*, Joseph Cavallaro*, and Mehdi Razavi‡
*Department of Electrical and Computer Engineering, †Department of Computer Science
Rice University, Houston, TX
‡Department of Cardiology
Texas Heart Institute, Houston, TX
yarefeen@mit.edu, zdeblick@uw.edu, ptaffet@rice.edu, aaz@rice.edu, cavallar@rice.edu, razavi@bcm.edu

***Abstract*—While beat detection in a Cardiac Electrical Signal (CES) is a well-studied problem, we propose a novel algorithm for implementation in pacemakers that learns the heights and widths of atrial and ventricular peaks from processing a few seconds of cardiac data sampled at 1 kHz. This purely data-driven solution to learn the parameters of atrial and ventricular peaks will allow pacemakers to set their own detection parameters and adaptively tune the parameters over time for that specific patient. We have validated our algorithm on one minute of cardiac electrical signal data from 51 separate channels, coming from 17 electrodes connected directly to cardiac tissue on each of three separate animals. Additionally, we have implemented the algorithm on a Field Programmable Gate Array and tested it on an ex-vivo rat heart to illustrate that our algorithm can run in real-time on commodity hardware.**

## I. Introduction

Cardiac diseases are the most common cause of death in the United States. More than 600,000 people die each year due to some form of heart disease [1]. Many heart failures result from the heart's inability to generate or to conduct these electrical signals that stimulate muscle contraction.

To combat these forms of heart failure, doctors implant artificial electronic pacemakers that stimulate the heart to contract artificially through electrical pulses. Current pacemakers use the DDDR algorithm to determine whether the heart requires stimulation [2]. In the current approach, a doctor must manually specify parameters for each individual patient in order to tune the detection algorithm to the patient's physiology. Essentially, the pacemaker then applies the tuned beat detection algorithm to the signal it senses to determine whether a heartbeat has occurred. If the pacemaker does not detect a heartbeat within a certain period of time, it will deliver an electrical stimulation to the heart.

Beat detection itself is a well-studied problem, as several algorithms have been proposed to identify the beats within a cardiac signal [3]–[5]. However, current algorithms perform beat detection based on sensing from two to three electrical leads on the heart without high resolution information distinguishing which part of the signal corresponds to the atrial contraction of the heartbeat and which part corresponds to the ventricular contraction of the heartbeat [2]. Our paper serves as the first iteration of a multiple year long research project that aims to combat this issue by designing a pacemaker that utilizes multiple wireless sensors on the heart to both sense and stimulate with unprecedented resolution [6]. Fig. 1 showcases the ultimate vision of the multiyear research project. Additionally, accurate distinction between the two contractions allows for a more effective form of pacing known as Cardiac Resynchronization Therapy (CRT) [7]. However, 30% of patients with pacemakers do not respond to CRT implemented by current pacemakers [7], leading to some of the 600,000 deaths.

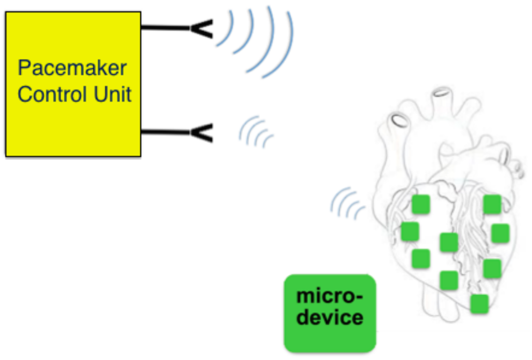

Fig. 1. Wireless micro-devices sensing the heart and communicating with the control unit that will house our algorithm. Our algorithm will learn beat detection parameters for each wireless micro device.

To efficiently learn parameters for the multiple wireless sensors and make beat detection more suitable for CRT, we propose a learning algorithm that learns the height of an atrial peak ($h_A$), the height of a ventricular peak ($h_V$), the width of an atrial peak ($w_A$), and the width of a ventricular peak ($w_V$) within a signal by processing a few seconds of data sampled at 1 kHz. Fig. 2 shows how these parameters correspond to portions of the cardiac electrical signal. We use these parameters in atrial and ventricular beat detection to illustrate that our learned parameters can be used to distinguish between atrial and ventricular beats in a 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.

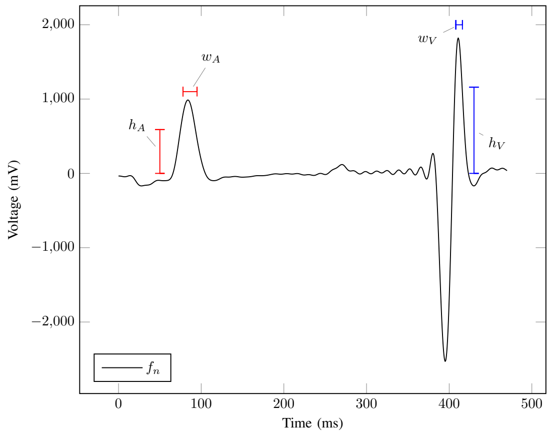

Fig. 2. A segment of CES containing a single atrial and ventricular beat, labeled with the parameters $h_A$, $h_V$, $w_A$, and $w_V$. Notice that the atrial beat is wider but smaller than the ventricular beat.

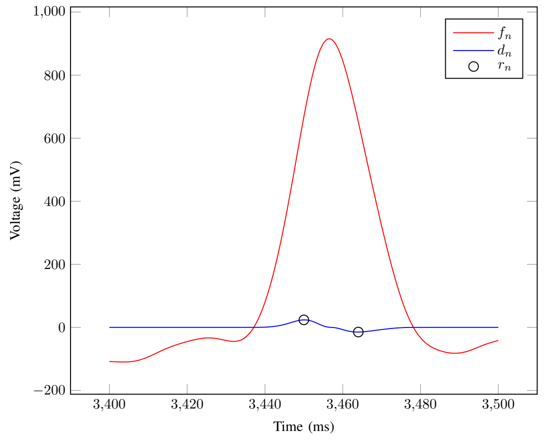

Fig. 3. Illustration of the relationship between $f_n$, $d_n$ and $r_n$.

Sections II and III explain our approach for learning $h_A$, $h_V$, $w_A$, and $w_V$. 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 the atrial and ventricular widths: $w_A$ and $w_V$.

### A. *Finding Peak Shaped Patterns in the Signal*

Let $f_1, f_2, \dots, f_N$ be CES sampled at 1 kHz. We choose N = 10000, as we found 10 seconds of CES data provided enough information for parameter learning. We begin by computing the weighted time difference of $f_n$, denoted $d_n$.

$$d_n = (f_n - f_{n-1}) \left| f_n - f_{n-1} \right| \left| f_n \right| .$$

Note that $d_n$ is essentially $(f_n - f_{n-1})^2 f_n$, but with the sign adjusted to match the sign of $f_n - f_{n-1}$. The following paragraph will explain our rational for constructing $d_n$. Next, the algorithm searches for when the local maxima and minima of $d_n$ occur, within a window of 200 ms, which we define to be the potential peak detection flag $r_n$. We choose to find the maxima and minima within a 200 ms window since there will be more than 100 ms of separation between a ventricular and atrial beat [8]. Thus, we compute:

$$r_n = \begin{cases} 1 & \text{if } f_n > 0 \text{ and } d_n = \max\limits_{-100 \le i \le 100} (d_{n+i}) \\ -1 & \text{if } f_n > 0 \text{ and } d_n = \min\limits_{-100 \le i \le 100} (d_{n+i}) \\ 0 & \text{otherwise} \end{cases}$$

Note, that a value of 1 in $r_n$ corresponds to the steepest rising edge of a positive peak in $f_n$, while a value of $-1$ in $r_n$ roughly corresponds to the steepest falling edge of a positive peak in $f_n$. The interval between a 1 and a $-1$ in $r_n$ then corresponds to a peak-like structure in $f_n$, where a rising edge is followed by a falling edge. Using a weighted difference, $d_n$, instead of a simple time difference pushes these edges closer to the tip of a peak, which allows for more appropriate feature extraction later on in the algorithm. Fig. 3 illustrates the relationship between $f_n, d_n$, and $r_n$ for a single peak.

### B. *Peak Feature Extraction and Clustering*

We restrict our attention to peaks that consist of a rising edge followed in less than 75 ms by a falling edge since neither atrial nor ventricular peaks can be greater than 75 ms in length [8]. More precisely, let $m_i$ and $n_i$ be integers such that $m_i < n_i, r_{m_i} = 1, r_{n_i} = -1, n_i - m_i < 75$ms, and $r_x = 0$ for $m_i < x < n_i$. The width and height of the peak will then be used as features. Thus, we define $P_i$, a peak characterized by its width and height, as follows:

$$P_i = \left\{ n_i - m_i, \left( \max_{m_i \le j \le n_i} f_j \right) - \frac{f_{n_i} + f_{m_i}}{2} \right\}.$$

Let $\mathbf{P} = [P_1, P_2, \dots P_s]$ be peaks characterized by their widths and heights from $f_n$. Now consider Fig. 4 where all of the $P_i$'s are plotted as points, with the $x$-axis as peak width, related to $w_v$ and $w_a$, and the $y$-axis as height, unrelated to $h_v$ and $h_a$ (We explain the difference in the following paragraph). With the intuition that ventricular peaks are generally taller and thinner than atrial peaks [8], $P_i$'s that correspond to ventricles cluster together. Likewise, $P_i$'s that correspond to atria also cluster. Thus, we can determine the average ventricle and atrial peak by finding the centers of these clusters. To do this, we cluster the peaks in $\mathbf{P}$ using a standard two cluster k-means algorithm [9]. We identify the ventricular cluster by choosing the corresponding mean with a greater height to width ratio, since ventricular peaks are generally taller and thinner. Let $C_V$ and $C_A$ be the centers of the ventricular and atrial clusters respectively. Since $C_V$ and $C_A$ essentially describe the average ventricular and atrial beat, the x-coordinates of the centers will give the average widths for ventricles and atria. The algorithm then stores the ventricular peak width ($w_V$) and atrial peak width ($w_A$) as the rounded width term, or x-coordinate, of the centers $C_V$ and $C_A$ respectively.

Note that the height that we use as a peak feature in this portion of the algorithm does not correspond to the actual

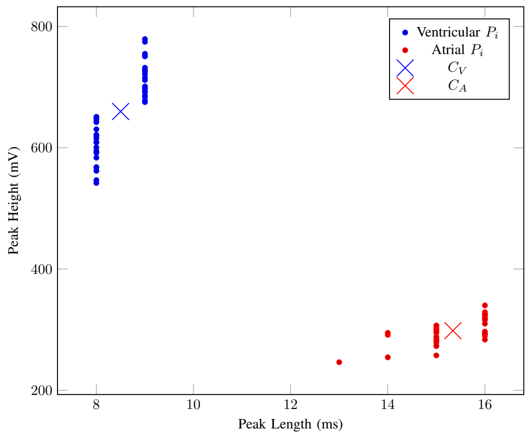

Fig. 4. K-Means clustering $\mathbf{P}$ into ventricles and atria.

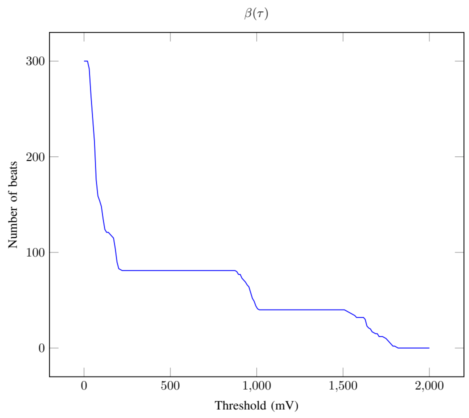

Fig. 5. Graph of the function $\beta(\tau)$ on example CES data. The long flat segments correspond to potentially good threshold choices.

values of $h_A$ and $h_V$ in the original signal $f_n$. This 'height' is the maximum height of the peak above the portions of the peak during which its value is most rapidly increasing or decreasing. In contrast, $h_A$ and $h_V$ are the median height above the baseline value of 0. There is no simple relationship between the two.

## III. Ventricular and Atrial Peak Height Learning

After learning $w_V$ and $w_A$, we proceed on the same segment of CES to learn the ventricular and atrial peak heights: $h_V$ and $h_A$.

### A. *Overarching Intuition for Learning $h_V$ and $h_A$*

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 ventricular 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. Thresholds lower than that range will detect more beats, and thresholds higher than that range will detect less beats. In the previously described range, however, the number of detected beats is constant.

We define the function $\beta(\tau)$ to be the number of beats detected in $f_n$ using $\tau$ as a threshold. Since $\beta(\tau)$ is a monotonically non-increasing for reasonable values of $\tau$, the only way that increasing the required threshold can increase the number of beats is if the threshold were so low that two adjacent beats were counted as one, but this only occurs at thresholds that are ignored by the algorithm. An example of $\beta(\tau)$ is illustrated in Fig. 5.

The height threshold learning algorithm is effectively a search algorithm for the longest and flattest interval of the function $\beta(\tau)$. However, computing $\beta(\tau)$ for even a single value of $\tau$ is somewhat computationally expensive, so we try to minimize the number of thresholds for which we compute it. Our algorithm uses bounded recursive refinement, and it essentially samples the function $\beta(\tau)$. We call this the "Flat-Finding" algorithm for ventricular and atrial heights.

### B. *Flat Finding Algorithm to determine $h_V$ and $h_A$*

As mentioned in the previous section, computing $\beta(\tau)$ for a given value of $\tau$ is computationally expensive because it requires computation proportional to the length of the data. Instead of computing $\beta(\tau)$ for each possible value of $\tau$, we run several iterations in which we pick thresholds $\tau_i$ for $i = 1, 2, \ldots, M$ evenly spaced between a minimum and maximum threshold $(\tau_{min}, \tau_{max})$. The algorithm then computes $\beta(\tau_i)$ for each $i$, and adjusts the minimum and maximum thresholds for the next iteration based on the results. We found that $M = 20$ sampled enough points from $\beta(\tau)$. For the first iteration, we set $\tau_{min} = 0$ and $\tau_{max} = \max f_n$. To choose the minimum and maximum thresholds in the next round, we first eliminate any thresholds $\tau_i$ such that $\beta(\tau_i)$ is biologically infeasible. For a human, we require that 10bpm $\leq \beta(\tau_i) \leq$ 200bpm. Note that $\beta(\tau_i)$ 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 a pair of indices that satisfies the following conditions:

$$\beta(\tau_i) = \beta(\tau_j), \quad \beta(\tau_{i-1}) \neq \beta(\tau_i), \quad \beta(\tau_j) \neq \beta(\tau_{j+1}).$$

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

If one or more flat intervals exist, choose $\tau_{max} = \tau_{j+1}$ and $\tau_{min} = \tau_{i-1}$ using the longest interval (i.e. the one that maximizes $j - i$). Because the flat interval $[i, j]$ is maximal and $\beta(\tau)$ is monotonic, we know that the true end of the flat interval is somewhere between $\tau_{i-1}$ and $\tau_i$. Choosing $\tau_{min} = \tau_{i-1}$ instead of $\tau_i$, gives the next recursive refinement iteration a chance to identify more closely the end of the flat interval. Similar logic applies to our choice of $\tau_{max}$.

If no flat intervals exist, choose the interval with the lowest derivative. Set $i = \arg\min_i |\beta(\tau_i) - \beta(\tau_{i-1})|$, and $\tau_{max} = \tau_i$, $\tau_{min} = \tau_{i-1}$.

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

$$h_V = \frac{\tau_{min} + \tau_{max}}{2}.$$

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, $h_A$.

### C. *Precise Specification of $\beta(\tau)$*

Our 'flat-finding' algorithm to determine $h_A$ and $h_V$ relies entirely on $\beta(\tau)$, the number of beats that occur in $f_n$ using $\tau$ to identify the beats. In order to calculate the number of beats that occur based on threshold, we use an algorithm that draws heavily from the beat detection algorithm described in Pan and Tompkins [3]. In the remaining portion of this section, we describe the beat detection algorithm that we use to compute $\beta(\tau)$. Since we independently learn $h_V$ and $h_A$, as a notational convenience, define $w$, the peak width, to be $w_V$ when computing $h_V$ and $w_A$ when computing $h_A$. Notice how this portion of the algorithm uses parameters computed previously in the algorithm, $w_V$ and $w_A$. Next, define the sequence:

$$q_n = \begin{cases} 1 & \text{if } f_n > \tau \\ 0 & \text{otherwise} \end{cases}$$

$q_n$ 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 $w$ (either $w_A$or $w_V$) in time. Thus, to compute $\beta(\tau)$, i.e. the number of beats that occur in $f_n$, we first compute

$$b_n = \begin{cases} 1 & \text{if } \left(\sum_{i=0}^{w} q_{n-i}\right) > w/2 \\ 0 & \text{otherwise} \end{cases}$$

Essentially, $b_n$ indicates which samples of $f_n$ are part of a beat, according to $\tau$. Finally, we define $\beta(\tau)$ to be the number of rising edges in the sequence $b_n$, i.e. the number of $n$ such that $b_n = 1, b_n = 0$.

After learning parameters, we also use essentially this same algorithm to detect beats in real-time. To detect ventricular beats, set $w$ and $\tau$ to $w_V$ and $h_V$ respectively. Similarly, to detect atrial beats, set $w$ and $\tau$ to $w_A$ and $h_A$ respectively. In addition, in accordance with standard pacemaking practice, we ignore any atrial beats within a fixed amount of time (usually around 250 ms) after a ventricular beat.

## IV. Hardware Implementation

In the future, we envision that our algorithm will be implemented in real pacemakers. In order to demonstrate that our proposed solution can run in real-time on 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 an analog preprocessing circuit board that cleans and amplifies the signal from the heart before it is digitized and processed by the algorithm running on the FPGA.

### A. *FPGA Algorithm Implementation*

In order to meet power and size constraints, current pacemakers contain an application-specific integrated circuit (ASIC) for running a beat detection algorithm. Although the FPGA chip that we used is large, and far too power-hungry for an implantable pacemaker, it serves as a first step towards an ASIC that could run our algorithm. In fact, we implemented our beat detection algorithm in Verilog on FPGA fabric to showcase how it could be shrunk down to an ASIC for future pacemakers. An illustration of our ASIC layout can be found in Fig. 6.

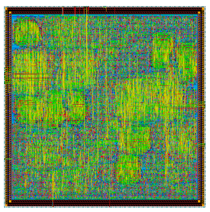

Fig. 6. ASIC Hardware Design of the algorithm implementation on the FPGA.

### B. *Pre-processor Circuit Board Design*

Since pacemakers take analog signals as input, we also 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 amplifying and filtering electrical activity directly from a heart. The filtered signal can then be properly digitized and used as input to the algorithm implementation on the FPGA. A design drawing of our board, which draws heavily from ideas in [10], can be seen in Fig. 7.

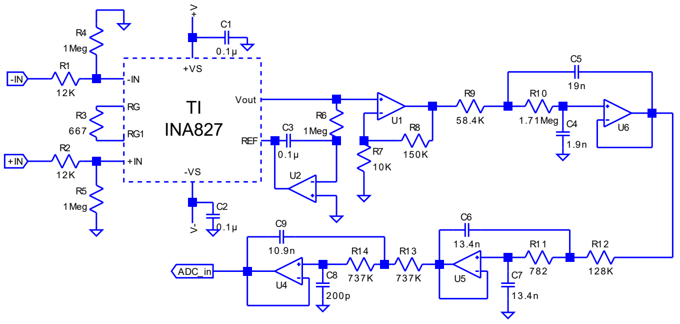

Fig. 7. Schematic for preprocessor circuit that senses and filters analog data from the heart.

The preprocessor consists of an instrumentation amplifier, followed by a DC offset removal op-amp (U2), a non-inverting amplifier (U1), and a 3 stage low pass filter (U4-6) before going into the ADC. The TI INA827 chip is an instrumentation amplifier that takes a differential signal as input, combines it to a single output, and amplifies the signal by 5x. The op amp that feeds back to the REF pin 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 16x. Finally, we utilize a lowpass filter with cutoff frequency at 150 Hz, not related to the 1 kHz sampling rate.

## V. Testing and Results

### A. Ventricular and Atrial Parameter Learning Validation

In Sections II and III of this paper, we presented an algorithm that learns $w_A$, $w_V$, $h_A$, and $h_V$ by simply processing 10 seconds of CES data sampled at 1 kHz. To validate our solution, we implemented our algorithms in Matlab and performed atrial and ventricular beat detection using learned parameters on data provided by colleagues 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 were placed on the left ventricle of the heart. Each electrode sensed electrical activity directly from the heart for one minute and digitized the signal at a 1 kHz sampling rate. This was done for 3 animals, giving us a total of 51 channels, where each channel is one minute of cardiac electrical signal.

We then applied our algorithm to the minute long signal from each of the 51 channels separately in order to learn $w_A$, $w_V$, $h_A$, and $h_V$ specifically for each signal. 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 learned channel-specific parameters. Finally, we manually identified all of the atrial and ventricular beats to establish a ground truth. Comparing the output of the algorithm to this ground truth, we computed the atrial and ventricular beat detection false positive (a beat is detected when none exist) and false negative (no beat is detected when one exists) error rates across all 51 channels. Table I illustrates our results.

TABLE I
Detection Error Rates using Learned Parameters

| Type | False Positive Error Rate | False Negative Error Rate |
|---|---|---|
| Ventricles | 0.16% | 4.77% |
| Atria | 0.89% | 4.59% |
| Combined | 0.50% | 4.70 % |

Since we performed atrial and ventricular beat detection based on the parameters that our algorithm learned, the algorithm clearly learned $w_A$, $w_V$, $h_A$, and $h_V$ correctly for the vast majority of channels. In fact, most of the atrial false negative errors are from our algorithm learning incorrect atrial parameters for one channel. When we manually looked at the values of $w_A$, $w_V$, $h_A$, and $h_V$, we found that the algorithm learned $w_V$ and $h_V$ to correctly detect ventricular beats for all 51 channels and learned $w_A$ and $h_A$ to correctly detect atrial beats for 50 of the channels.

### B. System Validation through a Langendorff Heart

In order to validate the hardware implementation of our algorithm described in Section IV, we connected our system to a Langendorff heart as described by [11]. Essentially, in a Langendorff heart experiment, a live heart is extracted from an animal, which in our case was a rat, and is kept beating by perfusing it with a solution containing oxygen and nutrients [11]. At an electrophysiological level, the externally beating heart behaves almost identically to a normal internal beating heart. We connected our preprocessor to wires sutured to the Langendroff heart, and then used our FPGA to run our algorithm on the cardiac signal. Our system correctly sensed signals using the preprocessor, learned pacing parameters ($w_V$ and $h_V$), and detected ventricular beats using the learned values of $w_V$ and $h_V$ 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 within the sensed signal itself,so our algorithm failed to learn valid parameters for it.

## VI. Conclusions and Future Direction

We have demonstrated a working algorithm implemented on hardware that correctly learns $h_A$, $h_V$, $w_A$, and $w_V$. Additionally, these parameters can be reliably used to distinguish between atrial and ventricular beats in a cardiac electrical signal in real time.

Future work on the algorithm will include incorporating knowledge of the physical location of the sensor (e.g. which chamber) into our parameter learning algorithm. We would also like to validate further our algorithm by testing it on data that responds to pacing decisions (e.g. testing during a live animal cardiac study). Additionally, we plan to utilize 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," in *Implantable Cardiac Pacemakers and Defibrillators: All You Wanted to Know*, A. W. Chow and A. E. Buxton, Eds. Blackwell, 2006, ch. 1, pp. 1–28.

[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] Y. H. Hu, S. Palreddy, and W. J. Tompkins, "A patient-adaptable ECG beat classifier using a mixture of experts approach," *IEEE transactions on biomedical engineering*, vol. 44, no. 9, pp. 891–900, 1997.

[6] H. Rahmani and A. Babakhani, "A fully integrated electromagnetic energy harvesting circuit with an on-chip antenna for biomedical implants in 180 nm SOI CMOS," in *SENSORS, 2016 IEEE*. IEEE, 2016, pp. 1–3.

[7] 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.

[8] W. Boron and E. Boulpaep, *Medical Physiology*. Elsevier, 2016.

[9] 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.

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

[11] 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.