Real-Time, Data-Driven Algorithm and System to Learn Parameters for Pacemaker Beat Detection (Extended Summary)

Yamin Arefeen*, Philip Taffet†, Daniel Zdeblick*, Jorge Quintero*, Greg Harper*,
Behnaam 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

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. To combat heart failure, doctors implant artificial electronic pacemakers that stimulate the heart to contract artificially through externally supplied electrical pulses. Current pacemakers use the DDDR algorithm to determine whether the heart requires stimulation from a cardiac electrical signal. Doctors must manually specify parameters in the algorithm for each individual patient in order to tune the detection algorithm to the patients physiology. Additionally, current pacemakers are often unable to distinguish between atrial and ventricular beats. Accurate distinction between the two portions allows for a more effective form of pacing known as Cardiac Resynchronization Therapy (CRT). However, 30% of patients with pacemakers do not respond to CRT implemented by current pacemakers.

To free doctors from having to manually tune parameters for each patient and to 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 simply processing 10 seconds of data sampled at 1 kHz (see Fig. 1). 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.

Our algorithm begins by learning w_A and w_V on 10 seconds of cardiac electrical data. In order to learn the two width parameters for peak detection, our system uses a nonparametric algorithm that finds and extracts features of all peak-shaped patterns in the training waveform that could correspond to beats. It then uses k-means, with k = 2, to segregate the patterns into ones that correspond to ventricular beats and those that correspond to atrial beats, and then extracts the lengths from the centers of each cluster.

After learning w_V and w_A, we proceed on the same segment of cardiac electrical signal to learn h_V and h_A. The basic

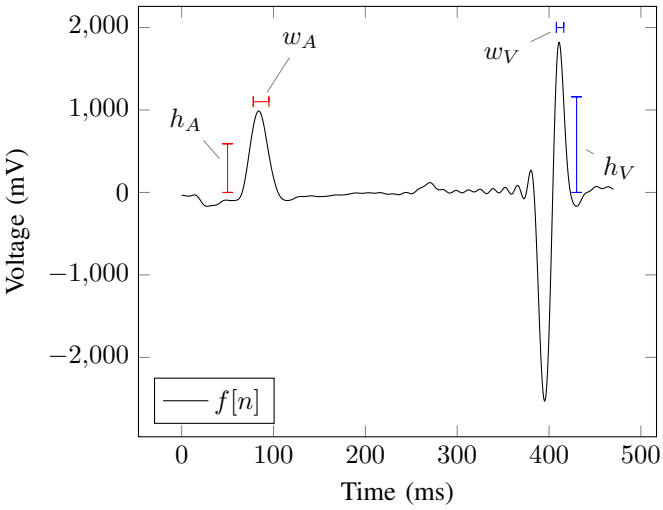

Fig. 1. A segment of CES containing a single atrial and ventricular beat, labeled with the parameters h_A, h_V, w_A, and w_V. Notice that the atrial beat is wider but smaller than the ventricular beat.

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 the cardiac electrical signal. 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.

The height threshold learning algorithm is effectively a search algorithm for the longest and flattest interval of the function $\beta(\tau)$, which we define as follows.

$\beta(\tau) =$
Number of beats detected in $f[n]$ using τ as a threshold

However, computing $\beta(\tau)$ for even a single value of τ is

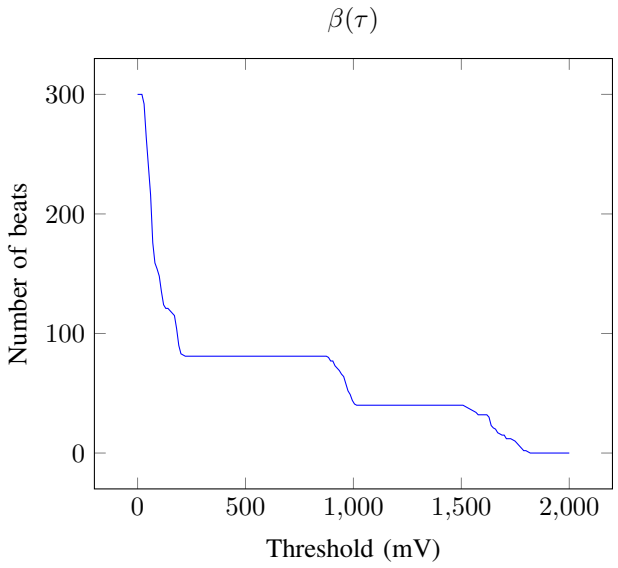

Fig. 2. Graph of the function $\beta(\tau)$ on example CES data. The long flat segments correspond to potentially good threshold choices.

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. An extensive description of both of our algorithms can be found in the complete paper.

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 filters and cleans the signal from the heart before it is digitized and processed by the algorithm running on the FPGA.

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 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, 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. Each channel resembles the signal a real pacemaker uses as input. Our detection error rates are sufficiently low across all 51 channels (see Table I). 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 order to validate the hardware implementation of our algorithm, we interfaced our system to a Langendorff heart. We connected our preprocessor to ordinary 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.

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.

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%
Total	0.50%	4.70 %