Intelligent Patient Monitoring: Detecting and Defining Significant Clinical Events

by
Adam K Hoyhtya
BS, Johns Hopkins University (1994)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author .....
Department of Electrical Engineering and Computer Science
May 22, 1996

Certified by .....
Roger G. Mark
Professor of Health Sciences and Technology and Electrical Engineering

Accepted by .....
Frederic R. Morgenthaler
Chair, Department Committee on Graduate Students

Intelligent Patient Monitoring: Detecting and Defining Significant Clinical Events

by
Adam K Hoyhtya

Submitted to the Department of Electrical Engineering and Computer Science on May 22, 1996 in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering

ABSTRACT

Computerized algorithms were developed which automatically analyze the arterial blood pressure measurements of intensive care patients. The primary task was to develop automated procedures to distinguish a blood pressure alarm signaling a true hypotensive or hypertensive episode from an alarm resulting from corrupted measurements. The real episodes were then visually categorized according to the direction of change for several physiologic measurements.

A clinical event was defined by the occurrence of a blood pressure alarm created by Hewlett Packard's "Merlin" bedside monitor, and each was initially classified by the author as either resulting from a real change in the blood pressure, or resulting from artifact. The methods for automatically classifying each alarm "event" used the systolic or mean blood pressure measurement, annotations from the monitor regarding the operation of the transducer (whether it was properly operating), and the continuous blood pressure waveform.

Using the results of the visual classification as the reference, results for the first 14 records (i.e., excluding 420 and 421) show that the linear trend algorithm using only the systolic or mean blood pressure measurements improved the average positive predictive accuracy from 61.9% (monitor unassisted) to 79.7% (sensitivity of 97.4%). Excluding artifactual data as indicated by the "INOP" annotations, the same linear trending algorithms using the first 14 records yielded an identical PPA of 79.7% and sensitivity of 97.4%. Detailed beat-by-beat analysis of the blood pressure waveform integrity allowed more sensitivity to real changes, resulting in a 97.9% sensitivity (99.4% gross sensitivity, with only 1 FN), but only a 65.8% positive predictive accuracy. The high number of false positives using the waveform analysis algorithm was a result of the algorithm declaring "real" an episode for which no stable template was located.

Finally, each non-artifact "clinical event" was categorized by considering the correlated changes in the systolic blood pressure, heart rate, and diastolic pulmonary arterial pressure measurements. Results were patient-specific, but the common finding was that a range of physiologic hypotheses could be suggested using the change in the physiologic measurements alone. Most alarms were of the same general type for a patient, and a change in the alarm type suggested interesting episodes.

Thesis Supervisor: Roger G. Mark

Title: Professor of Health Sciences and Technology and Professor of Electrical Engineering and Computer Science

Table of Contents

Table of Figures

1 Introduction

1.1 Motivation

The intensive care units of hospitals produce massive amounts of clinical data relating to each acutely ill patient. Although the purpose of gathering such a large quantity of data is to maximize the clinician's sensitivity to meaningful changes in the patient's state, physicians may experience "information overload", potentially hindering efficiency and accuracy. This research describes methods for automated verification and interpretation of data to reduce the amount of rote processing that the clinician must perform when an alarm is sounded. Specifically, the research describes automatic classification of the monitor's blood pressure alarms as valid or a result of artifact. Secondly, in addition to the blood pressure, other physiologic measurements within the vicinity of the alarm are analyzed in order to suggest a physiologic basis for the cause of the hypo- or hypertensive state. The purpose is to discount alarms that result from invalid measurements, and to aid the physicians and nurses in constructing a physiologic hypothesis of the state of the patient. An apropos quote from the late President John F. Kennedy gives perspective to the focus on the patient-computer interaction:

"Man is still the most extraordinary computer of all."

In short, the goal of this research is to perform analysis which aids the user, but does not usurp the authority of the physician and staff in assessing the physiologic state of a patient.

An extension to this project – which is not formally addressed in this thesis – is ultimately to construct an "intelligent monitoring" system, which considers all the information which the physician would typically utilize, for automatic hypothesis generation regarding the physiologic state of the patient. This information would include clinical data such as laboratory results, in addition to the output from the patient monitoring system. Devising an "intelligent monitoring"

system is the continuing project of Roger Mark and George Moody, in collaboration with the intensive care staff at Boston's Beth Israel Hospital.

The motivation behind developing an improved, "intelligent" monitoring system, is that the current problem of frequent false alarms in the ICU is not trivial. Indeed, the noise level in the ICU due to the monitors' alarms has been described as "disturbing" to both the patient and staff, according to Cropp et al[1]. Although the alarms producing these high noise levels are a means of safeguarding the patient's health in case of abrupt changes in patient physiology, because of the high false alarm rate, the alarms are frequently discounted by the staff, either through temporary silencing (e.g., Hewlett Packard's Merlin monitor has a 3-minute silencing option), permanent silencing (disabling the alarm), or by the staff person simply not reacting to subsequent alarm soundings. In addition to not being completely predictive of significant physiologic changes, the alarms are not easily discerned by the clinicians, the very persons they are intended to alert. Cropp et al studied the effectiveness of the alarms in communicating the type and severity of the problem associated with several alarms. They concluded that a more organized system of alarms would aid the ICU staff in identifying the problem. In short, the efficacy of an alarm system would be greatly enhanced through both organization and minimization of the number of false alarms.

1.2 Previous Work

1.2.1 ICU Monitors: The State-of-the-Art

Currently the treatment of intensive care patients requires a diligent focus by the ICU staff, who rely greatly on the automated bedside monitoring systems. Most ICU's have one staff person per patient in order to continuously monitor the patient's state and react to changes in the physiologic measurements. However, today's computerized monitoring consists mainly of obtaining measurements and detecting artifact, leaving it to the staff to assess the validity of alarms and to construct a physiologic hypothesis of the patient's state.

In 1991, Coiera (Hewlett Packard Laboratories) wrote a review of the current technology in ICU monitoring, and discussed the most popular methods for improving the monitoring methodology. He proposes that there are three basic layers in which the data is currently presented to the user. In layer 1 are the continuous waveforms, such as the ECG and ABP waveforms, and the numeric data, which are measurements from these waveforms, in addition to other non-waveform measurements, such as temperature and oxygen saturation. Layer 2 consists of measurement trends, which provides the history of the signals over a longer time than the waveforms can be displayed, along with superimposed range limits, which allows the user to identify graphically any changes in the measurement series with respect to preset limits. The third layer is the detection and display of range violations, as well as dynamic trend estimation, which differs from the trend display in layer 2 in that only a recent history is analyzed (for example calculating slope over past few minutes), so that abrupt changes can be found. The data is abstracted in an hierarchical fashion, with each layer dependent on the validity of previous layers.

Due to the dependence of the monitor's alarms on the validity of the signal, the first priority in designing a monitoring system for intensive care patients is collecting a valid set of physiologic measurements. Contaminated data is not only confusing to the staff monitoring the patient, but also detrimental to automated hypothesis generation algorithms. Therefore, built into the "Merlin" monitor are data conditioning units, which contain hardware for filtering noise and indicating when artifact occurs in the signals. This type of signal-specific processing is necessary when one considers the differences in how each signal is acquired, and therefore in how each is processed. When artifact is indicated as being present, the monitor will not signal an alarm even though the measurements may move outside the alarm limits. The HP reference[2] describes each of the modules involved in acquiring and conditioning the various signals (ECG, ABP, PAP).

1.2.2 Progressive Analysis

Considering the complexity of human cardiovascular physiology, replete with feedback interaction between the physiologic variables, the important question to answer is how a patient monitoring system can validate the data provided by the monitor, and subsequently track the patient's changing physiology. Since the task of completely specifying the patient's cardiovascular functioning from a few variables is intractable, one must accept some representation of, or hypothesis concerning, the CV system. The current hypothesis for standard patient monitoring algorithms is that any variation of a parameter within a specified range, regardless of its dynamics, is acceptable, and any excursion outside these static limits is pathological. Outlier detection algorithms flag this out-of-range condition, one in which a variable either exceeds a maximum or drops below a minimum threshold. An example of this method is an alarm sounding when the systolic radial arterial blood pressure (sABP) of a patient exceeds 110 mmHg. Obviously the sounding of such an alarm does not fully elucidate the physiology, but indicates a high probability of a significant change in the patient's blood pressure. Given this information, the ICU staff can either consider this state acceptable, or intervene pharmacologically to decrease the systolic blood pressure. However, there are two prominent shortcomings of single parameter alarms. First, many ICU patients, however infrequently, undergo important physiologic changes, yet homeostatic controls are able to maintain the parameters within broadly normal ranges, and thus no alarm is signaled. An example is the case of a slow gastrointestinal (GI) bleed where the cardiovascular control system maintains blood pressure (ABP) within the static boundaries until the patient enters a state of hypovolemic shock. The second shortcoming of single parameter alarms is that they frequently produce false alarms; that is, they indicate that a critical change has occurred when in fact the patient is performing acceptably. Clearly, a tracking system which only signals a pathology after it has been developing steadily for some time, or one which is marginally predictive of significant changes, is deficient in its utility.

Several other methods have been proposed for analyzing patient data. Zhao[3] has researched the utility of a detailed cardiovascular model, which estimates approximately 20 different parameters within the CV system given simulated data. The goal of the research is to devise methods for appropriately locating the parameter which caused the change in the simulated waveforms (which represent the patient). Other researchers[4] have analyzed the spectral characteristics of the beat-to-beat variation in heart rate, in order to estimate autonomic activity, as well as the transfer function characteristics between respiration, arterial blood pressure, and heart rate, also thought to be indicative of heart rate control. Since the autonomic nervous system elicits control over cardiovascular function. the ability to detect such neural changes gives important information regarding the patient's state.

1.3 Thesis Summary

This project explores methods which focus on the alarm sequence and the several minutes of measurements prior to each alarm. In particular, the ABP alarms are classified as valid or (resulting from) artifact, through analysis of the physiologic waveforms and the measurements. The gold standard for the effectiveness of these automated methods is the author's visual inspection and classification of each event. The number of alarm episodes, as well as how they were categorized, is summarized in section 3.1.1 Visual Artifact Detection. The proposed system further categorizes those episodes which are automatically classified as valid by analyzing the correlated changes between arterial blood pressure (ABP) – either systolic or mean, depending on which was used to trigger ABP alarms – diastolic pulmonary arterial pressure (dPAP), and heart rate (HR). The direction of change of these three measurements suggests a physiologic basis for the alarm condition.

2 The MIMIC Database

The data used in the algorithm development for this research is the MIMIC (or Multi-Parameter Intelligent Monitoring for Intensive Care) Database, which will eventually consist of approximately 100 24-48 hour recordings from the patient monitors of the three intensive care units (medical, surgical and cardiac) of Boston's Beth Israel Hospital. The database also includes relevant clinical data such as lab tests, medications, and clinical notes by physicians and nurses. The following diagram depicts how the data is collected and stored.

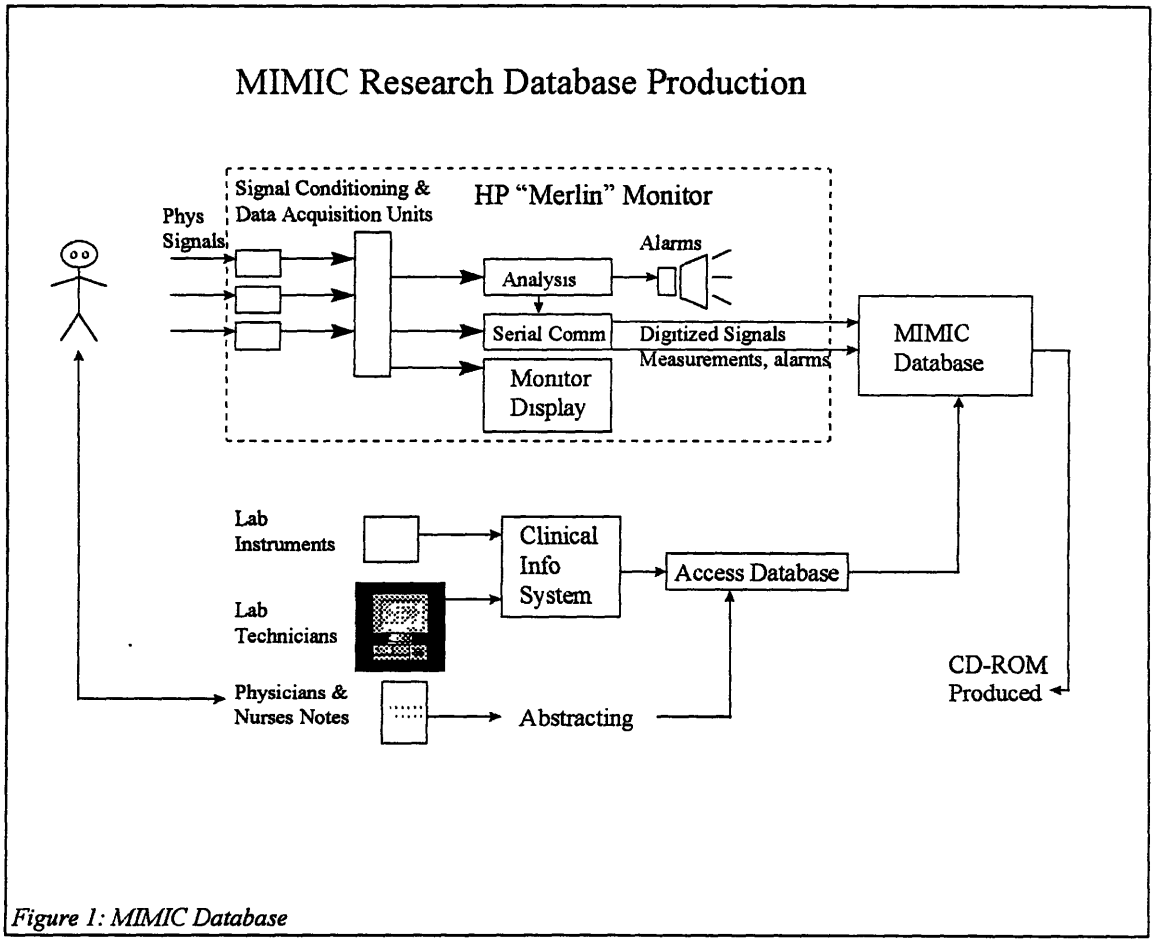

Figure 1: MIMIC Database

2.1 Continuous Signal Recordings

The signals were collected and digitized by the Hewlett-Packard Component Monitoring System (Merlin) bedside monitors. Two dual-ported serial communications cards in the monitors allowed the transmission of the data to a Gateway 2000 PC (model 4DX2-66V), for eventual

recording onto CD media. Signal (*.dat) files in the MIMIC database contain the samples of the waveforms acquired by the monitor (500 samples per second for each ECG signal, and 125 samples per second for all other signals). These waveforms are analyzed within the monitor for various features, most typically the maximum, mean, and minimum values of the blood pressure signals, and the locations and classifications (normal or VPB) of the QRS complexes of the ECG signals. The range of signals which may be recorded include several ECG leads, arterial blood pressure, pulmonary arterial pressure, central venous pressure, plethysmograph, pulse oxymeter (oxygen saturation), and respiration. Despite the fact that patients with different pathologies, and from three separate ICU's, do not necessarily have the same physiologic variables monitored, an attempt was made to record three ECG signals, all blood pressure signals, and the respiration signal of each patient. Other signals were recorded according to a hierarchical selection system, and as the bandwidth of the monitor's communications card permitted.

2.2 Physiologic Measurement Series

One feature of the Merlin monitors is the continuous calculation and display of several clinically significant measurements from the calibrated waveforms. Examples of these numeric measurements (made at intervals of 1.024 seconds) are systolic, mean, and diastolic blood pressure, heart rate, systolic, mean, and diastolic pulmonary arterial pressure, etc. These data are available to the care provider and are typically displayed on the monitor's screen throughout the patient's stay in the ICU. Although most of these measurements (with the notable exceptions of cardiac output and SpO_2, the O_2 saturation of the blood) can be reproduced by processing the digitized waveforms, it is useful, and perhaps essential for algorithm development, to have the pre-processed time series of numeric measurements from the Hewlett-Packard monitors, since these are the exact data that the clinicians consider in monitoring the status of the patient.

There are two pieces of key information which the Merlin monitors create along with the physiologic measurements. First, the monitors indicate the condition of inoperative or noisy transducers ("INOP" notations). The annotation "NOISY-CHK ECG LEAD" warns of noise in the ECG leads, while "SpO2 NON-PULSATILE" indicates that the oxygen saturation signal is non-pulsatile, and therefore the transducer is most likely disconnected. Secondly, the Merlin monitors indicate alarm or "out-of-range" values, such as the following: "ART 107 > 105", indicating that the arterial blood pressure exceeded the pre-set alarm limit of 105 mmHg, and "PAIR VPBs", indicating two sequential ventricular premature beats were detected in the ECG signals. Since noisy ECG leads suggest that the HR signal is less reliable, it is clear that this information is essential when analyzing the measurements series. The alarm and "INOP" annotations are used to analyze patient records for significant events; the methods are discussed in section 3.1 Artifact Detection.

2.3 Clinical Data

Each patient record in the MIMIC Database includes clinical data. This information is extracted from the subjects' medical records and from the hospital's clinical computing systems; it has been formatted to be read by Microsoft's Access® Database program, and includes the following subsections.

1. Patient History
2. System Review
3. Physical Exam
4. Problem List
5. SOAP Notes
6. Progress Notes
7 Flow Sheet Menu
8. Lab Data Menu

The structured database format enables rapid access to information essential to the development of intelligent patient monitoring. Most importantly, the format highlights key "event" information, which can be considered when attempting to characterize the nature of the changes

occurring in the data streams. Two subsections of the database which are most useful are Progress Notes, and the Medications from the Flow Sheet Menu, including the amounts and the times delivered.

2.4 Characterization of Cases in MIMIC Database (as of 3/5/96)

The following information is a summary of the type of records contained in the MIMIC Database. With this information, it is possible to study specific pathologies. However, in this study, no distinction is made between patients on the basis of their major pathology.

A. Place Recorded (N = 132)
- MICU 61 (46%)
- CCU 60 (45%)
- SICU 11 (8%)

B. Patient Characteristics (N =132)
- Male 65
- Female 67

- Age < 50 17
- Age > 70 66
- Age 51-69 49

C. Major Clinical Problem (N=132)
- Respiratory Failure 34
- CHF/Pulmonary Edema 30
- Sepsis 14
- MI/Cardiogenic Shock 13
- Post-OP Valve 9
- Bleed 8
- Angina, R/O MI 5
- Brain Injury / CVA 5
- Post-op CABG 5
- MI / Arrest 3
- Post Op, various 2
- Metabolic Coma 2
- Cord Compression 1
- Renal Failure 1

2.5 Selected MIMIC Records

Several records were chosen from the MIMIC database for this research, based on the type of data that the record contained. Each patient record (as of 3/5/96) which contained arterial blood pressure, ECG, and pulmonary arterial pressure waveforms were selected for this study, provided the signal quality was adequate. The records indicated in the table below had these three measurements. The two records in parentheses (405, 427) had unreadable blood pressure data and were excluded from the study. In order to efficiently access the measurement data, several of the MIMIC records were re-stored on a single CD-ROM, Volume MDBS0296 (February 1996); each

record was renamed to have a trailing n (i.e., record 055 becomes 055n). The subset of these measurements containing systolic (mean) ABP, HR, and diastolic PAP were written to a file using the "rdsamp" program, available on each MIMIC CD-ROM. The blood pressure signal was chosen as that which the Merlin's blood pressure alarm used in its algorithm for triggering alarms. (Since this signal was variable, the output of "rdsamp" had to be compared to the ABP/ART alarm values to ensure the correct ABP/ART signal was being used.) The HR signal is that obtained through the monitor's ECG analysis. The diastolic pulmonary arterial pressure (PAP) was chosen (versus mean or systolic) since it best approximates the filling pressure of the left ventricle. (The filling pressure is an important measurement to use in conjunction with the systolic arterial blood pressure, since these two measurements are the minimum and maximum pressures within the left ventricle.) The following table gives the signal number given as an argument in the program "rdsamp", which is used to obtain the data used in these studies (e.g., c:\> rdsamp -r 055n -f 0 -t 100:0:0 -s bpnum hrnum papnum).

MIMIC Record	Blood Pressure	Blood Press. Type	Heart Rate	diast. PAP
055	1	syst ABP	4	7
212	1	syst ABP	3	6
215	0	mean ART	3	9
216	1	syst ABP	3	9
230	1	syst ABP	3	9
231	1	syst ABP	3	6
240	1	syst ABP	3	9
241	1	syst ABP	3	9
242	1	syst ABP	3	9
245	1	syst ABP	3	6
248	1	syst ABP	3	6
415	1	syst ABP	3	9
417	1	syst ABP	3	9
418	0	mean ABP	6	12
420	1	syst ABP	5	8
421	1	syst ABP	5	9
(405) - unread.	n/a	n/a	n/a	n/a
(427) - unread.	n/a	n/a	n/a	n/a

Each of these records had at least one blood pressure alarm event, and most patients had "INOP" indications in regions of the blood pressure measurement series which were corrupted by noise. In

addition to these measurement data, all patients had either the arterial blood pressure (ART) or radial arterial blood pressure (ABP) waveform, found on the CD-ROM containing the entire patient record (e.g., MIMIC record 055 is found on Volume MDB110).

The blood pressure alarms were obtained from the MIMIC CD-ROM Volume MDBS0296 using the "rdann" program. The alarm files used in these studies were created using the script "getals", a C-shell UNIX script (see Appendix A). These alarm files include both the time of occurrence, the blood pressure alarm value, and the threshold exceeded to trigger the alarm. The blood pressure "INOP" files were created using the script "getins" (see Appendix A). These "INOP" files include the time of occurrence, and the arterial blood pressure "INOP" type, coded as follows: "1" – "ABP NO TRANSDUCER", signifying the transducer is disconnected; "2" – "ABP REDUCE SIZE", signifying the measurement is in error due to its exceeding an upper threshold; "3" – "ABP ZERO+CHECK CAL", signifying that the monitor is being calibrated during the time that this notation occurs; "4" – "ABP OVERRANGE", signifying that an impossible measurement (ABP < 0) has been obtained. No other "INOP" annotations were found in the 16 records used in this work.

3 Methods

Given the MIMIC database, patient records were located which had the following three measurements available: ABP (arterial), ECG, and PAP (pulmonary). In addition, a method was developed by which this patient data could be carefully inspected visually. UNIX© shell scripts were written which access the patient data on CD-ROM and plot out several hours of each measurement (systolic/diastolic blood pressure, heart rate, systolic/diastolic pulmonary pressure) (see Appendix A, "hptognu" UNIX C-shell code). These were viewed in order to give an understanding of the type of data available. In this research an "event" was defined by the occurrence of one or more continuous ABP alarms (the criterion was that an "event" could not have alarms separated by more than 1.024 seconds, the sampling rate of the measurements, and the update rate for the alarms). Since the ABP alarm is triggered off one of three measurements (either systolic ABP, mean ABP, or mean ART) for each patient, the blood pressure measurement which corresponded with the alarm was plotted over the entire 24-48 hour period, along with the alarms and the alarm limits. Since no indication exists in the alarm annotation file as to which measurement the blood pressure alarm algorithm is considering, the program "rdsamp" (available on each MIMIC CD-ROM) was run for each record containing only the numerics. In addition, a plot was made of each alarm region, so that any changes could be magnified, and methods could be devised for automatic classification of each alarm as "real" or "artifact". Using both the alarm plots and the information of whether artifact or noise was present in the blood pressure waveforms in the region of the alarm, each alarm was classified by the author as "real", meaning that the measurement is a faithful representation of the pressure in the artery, or "artifact", meaning that the measurement does not reasonably represent the arterial pressure. These alarm "episodes" were subsequently classified as true physiologic changes or as artifact, according to the automated methods described below.

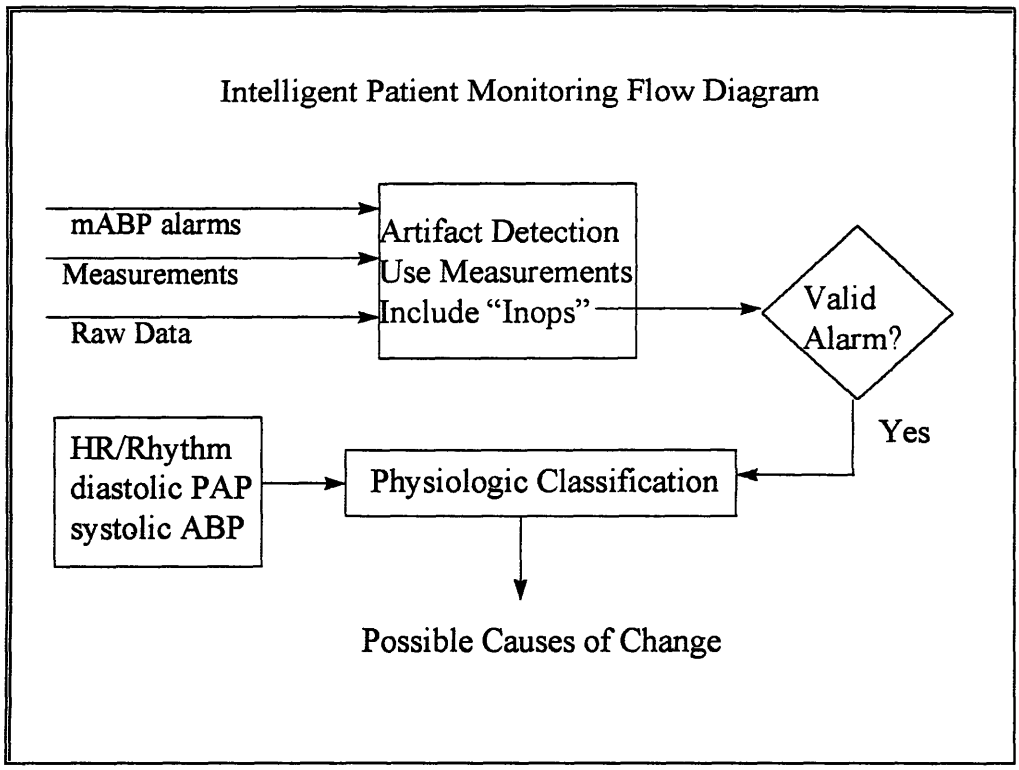

Figure 2: Monitoring Flow Diagram

3.1 Artifact Detection

As indicated in figure 2, there are three pieces of information to use when attempting to discern whether data is distorted by noise and artifact. The most readily available data are the measurement series themselves, since these are directly used to trigger the alarms. Secondly, the notations signifying inoperative transducers provide information regarding whether to discount an alarm. If a transducer is disconnected, the measurement is corrupted. Finally, an inspection of the waveform allows for verification of the waveshape integrity with respect to other regions. Processing these three classes of information helps to improve the alarm scheme, so that subsequent levels of analysis are assuredly processing non-corrupted data.

3.1.1 Visual Artifact Detection

The blood pressure measurements which were used in the Merlin monitor to trigger the blood pressure alarm (either systolic ABP, mean ABP, or mean ART – see section 2.5 Selected

MIMIC Records) were plotted, along with the alarms themselves, in order to study the alarm regions more closely. In addition, the raw waveforms were analyzed visually to locate any artifact which may have resulted in the alarm condition. This visual analysis was done in order to establish the standard to which the automated algorithms could be compared. Finally, the positive predictive accuracy of the Merlin monitor's blood pressure alarm algorithm was calculated for each patient and used for comparison against the positive predictive accuracy of newly developed methods which detect artifact.

The algorithm used by the Merlin monitor, which signals hypo- or hypertension, is a counting scheme. The count begins once the preset threshold is exceeded, and it continues to increment each time the ABP is updated (every 1.024 seconds), until a count of 10 is reached, whereby an alarm is sounded, or until the threshold is no longer exceeded and the count is reset. An example follows.

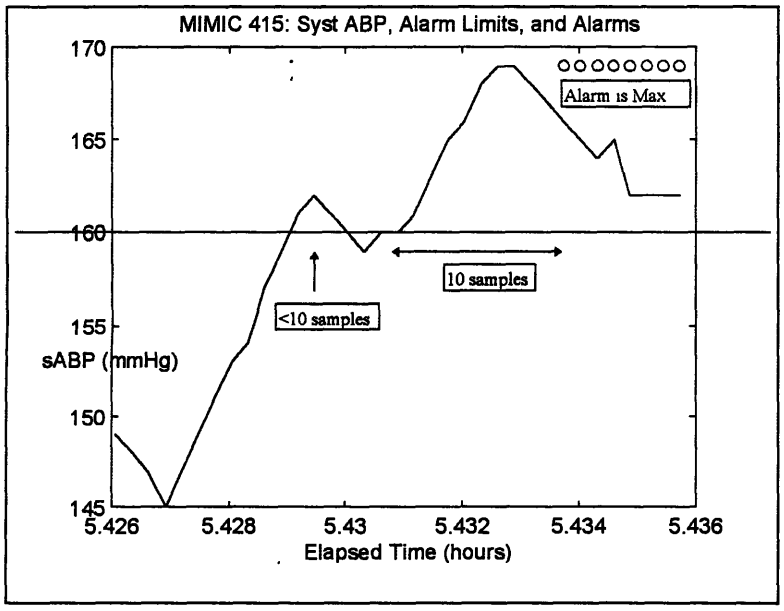

Figure 3: Alarm Example

Notice in figure 3 that the alarm begins 10 samples following the first sample that exceeded the 160 mmHg threshold. Prior to the excursion which sounded an alarm was a transient increase which was of a duration of 3 samples, so that an alarm did not sound. Of note is the fact that the alarm continues until the blood pressure ceases to exceed the threshold of 160 mmHg. In addition, the value of the alarm is the maximum blood pressure from the time the threshold was

exceeded. Since the pressure reached 169 mmHg at 5.433 hours, even though the alarm didn't sound until 5.434 hours, where the pressure was measured at 166 mmHg, the alarm assumes a value of 169 mmHg, the maximum pressure achieved since the blood pressure exceeded 160 mmHg at 5.431 hours. This is an example of a true alarm, since there is no artifact that is suggested either from these measurements, or from the raw waveform. Using plots such as these, as well as the ABP waveforms, the alarms were classified as resulting from artifact, or real physiologic change.

3.1.2 Automated Artifact Detection

3.1.2.1 Linear Trend Algorithm

The following algorithm (see program "class.m" in Appendix A), outlines the method for automatically detecting whether artifact is the cause of the blood pressure alarm.

1. Initially assume alarm is real; yields "aggressive" algorithm
2. If alarm value < 0 OR alarm value > 1.5*(Max_threshold) then alarm is "artifact"
3. Get 5 minute window and 1 minute window of data previous to alarm, both windows ending 10 seconds before alarm
4. Remove artifact data, based on "INOPs" and noise detection in ABP waveform
5. Find least squares best fit line to 5 minute window (trend), including error (y = mx + b ± 50percent_error)
6. Forecast to time of alarm using both trends
7. Trend test: If alarm value is less than forecast - α*error(for lower threshold), or alarm value is greater than forecast + α*error(for higher threshold), then condition suggests artifact
8. if 5 minute trend test suggests artifact then find mean, stdev of 1 minute window
9. if alarm value outside Bpval ± 2*α*stdev (Bpval is last pressure in 1 minute window), then alarm is artifact, on basis that it changes too quickly

Figure 4: Alarm Classifying Algorithm

One manner in which the measurement series themselves can be used to detect artifact is to define physiologically unrealistic measurements. Step two in Figure 4 defines physiologically realizable extrema, and compares the alarm values to those extrema. Those outside the range are artifact. The absolute maximum threshold is equal to 150% of the monitor's "hypertensive"

threshold setting, and the absolute minimum threshold is equal to 0 mmHg (negative pressures are considered artifact). These boundaries need to be very loose, since they provide a method for quickly detecting only the extreme outliers. Step three defines two trending windows, of 5 minute and 1 minute durations, both of which have the 10 seconds immediately prior to the alarm deleted in order not to bias the statistics by artifact which may have caused the alarm condition.

3.1.2.2 Utilizing "INOP" Information

The fourth step is a crucial one, since the notations signifying inoperative transducers provide much information regarding whether to discount an alarm. The algorithm which removes artifact deletes all data from 10 seconds prior to the occurrence of an "INOP" indicator, up to the time of the indicator itself. Since most indicators occur consecutively, there is considerable overlap of these regions, so that only longer (than 10 seconds), contiguous segments of data are deleted. These segments typically comprise only a small fraction (~3% for record 212) of the data from an entire record. Using the "INOP" indicators from the Merlin monitor provides a robust method of deleting obvious artifact, so that corrupted data is not used in the linear trending algorithm. However, the Merlin monitor, before sounding an alarm, already considers the information regarding the operation of the transducer, so that including the "INOP" information ought to change little in the efficacy of the linear trending algorithm. Thus, a method more sensitive to beat-by-beat changes in the blood pressure waveshape integrity may be necessary, and such a method is described in the following section.

3.1.2.3 Blood Pressure Waveshape Analysis

Finally, an analysis of the waveform allows for verification of the waveshape integrity with respect to an ideal template. This is done by first annotating all ECG QRS-complexes (using a program titled "sqrs") several minutes prior to an alarm. Since the shape of the blood pressure during the time between consecutive QRS-complexes is relatively conserved, aligning of blood

pressure waveforms can be most easily accomplished by extracting the blood pressure waveform beginning at the time of one QRS-complex and ending at the time of the subsequent QRS-complex.

It is desirable to compare the shape of an arbitrary blood pressure waveform segment to a template which in this problem defines the ideal waveform segment. Over a several hour epoch, a patient record will exhibit many fluctuations in both the pulse pressure (peak-to-peak magnitude) and the values of the systolic and diastolic arterial blood pressures (baseline values). However, despite the baseline and waveform magnitude changes, the overall waveshape ought to be conserved. A statistic which compares a waveform segment to a template, independent of the baseline and waveform magnitude, is the correlation coefficient. The correlation value has a range from -1 to 1, with 1 being perfect correlation, 0 being no correlation, and -1 being negative correlation. Only high degrees of correlation (close to 1) are acceptable, while correlation values near zero and less than zero are considered artifact.

Formally, the arterial blood pressure can be considered a random process X(t), and each blood pressure waveform is found by partitioning X(t) according to the ECG heartbeat annotations.

$$X(t) = [x_1(t)|x_2(t)|...|x_N(t)] \qquad (3.1)$$

The general Correlation Coefficient equation and Cross-covariance function for two random process X(t) and Y(t) follow[5].

$$r_{XY}(t_1,t_2) = \frac{C_{XY}(t_1,t_2)}{\sqrt{C_{XX}(t_1,t_1)\cdot C_{YY}(t_2,t_2)}} \qquad (3.2)$$

$$C_{XY}(t_1,t_2) = \sum_{j=1}^{N} X_j(t_1)\cdot Y_j(t_2) - \overline{X}(t_1)\cdot\overline{Y}(t_2) \qquad (3.3)$$

However, there is no need in this case to calculate the correlation for each pair of times (t_1,t_2), since the desire is simply to find the maximal correlation between two random variables,

regardless of the shift required to obtain the correlation. For two blood pressure waveforms, this is typically $t_1 = t_2 = 0$ (see artdet.c in Appendix A for actual algorithm used, including motivation for performing a certain number of shifts). Therefore, a template y(t), representing an ideal waveshape, is found and compared to each of the $x_i(t)$ using the correlation coefficient statistic ρ_i.

$$\rho_i = \frac{Cx_i y}{\sqrt{Cx_i x_i \cdot Cyy}} \quad (3.4) \qquad Cxy = \sum_{j=1}^{N} x(j) \cdot y(j) - \bar{x} \cdot \bar{y} \quad (3.5)$$

Where equation (3.5) is the cross-covariance function $C_{XY}(0,0)$ (i.e., evaluated at $t_1 = t_2 = 0$.), and equation (3.4) gives the "unshifted" correlation between x(t) and y(t). Provided the correlation coefficient ρ_i is greater than some threshold (0.9), the waveform is acceptable.

3.1.2.4 Correcting for Heart Rate Variation

In figure 5, a segment of an arterial blood pressure waveform is shown, with ECG beat annotations superimposed. Since a QRS-complex occurs at a variable time with respect to the previous QRS-complex, the $x_i(t)$ are not the same size, making it difficult to perform a direct correlation. This variation is seen with normal sinus arrhythmia, whereby a waveform from a heartbeat arrives before the blood pressure has sufficient time to decrease to the previous beat's diastolic pressure. Since there is a mechanical delay between the QRS-complex and the beginning portion of the blood pressure waveform, the waveforms that are correlated (which are those portions between consecutive QRS-complexes) are most similar over the middle portion of the waveform. That is, the shape of the end-diastolic phase of the blood pressure waveform is less conserved from beat-to-beat than the middle portion of the extracted waveform beginning at the upstroke of the blood pressure waveform. To correct for this, only the middle portion of each waveform segment is compared, thus deleting the few (3) blood pressure samples on either end which are less similar in value from beat-to-beat. This seems to be sufficient for regions where the heart rate does not change appreciably.

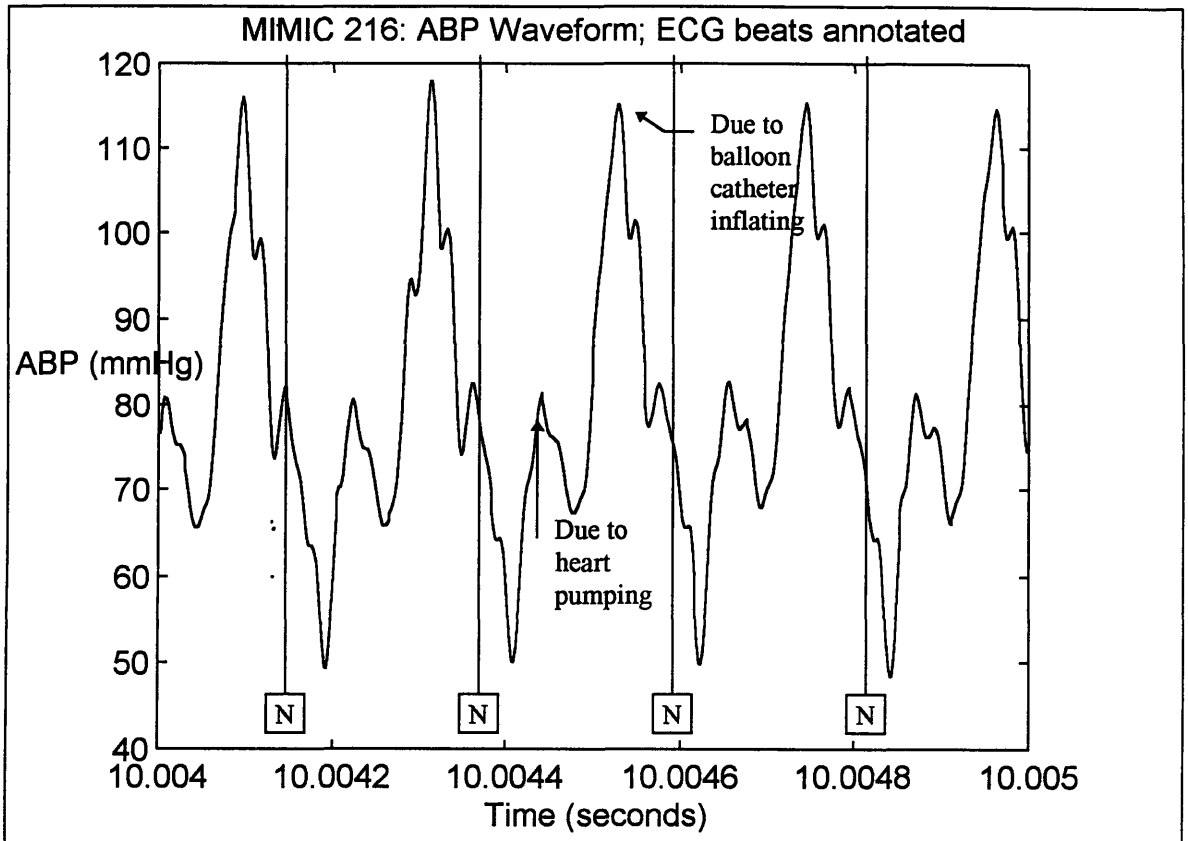

Figure 5: Annotated ABP Waveform

However, in regions where the heart rate does change more rapidly, a re-sampling scheme which yields two random processes of the same length should improve the correlation algorithm results. The assumption when two random processes (the template and the test waveform) are re-sampled is that if the R-R interval increases, the blood pressure waveform correspondingly increases. However, this may not always hold, so the template may need to be updated periodically to account for the change in shape.

The specific algorithm used to obtain the results in section 4.3 is shown in the outline below.

1. Locate the five-minute window of ABP waveform data prior to an ABP alarm
2. Run a QRS-detection algorithm on the data ("sqrs")
3. Partition the five-minute window of ABP waveform data according to the locations of the QRS complexes
4. Initially, consider the first single beat waveform (between QRS-complexes 1 and 2) as the "temporary" template
5. If template has high correlation (r > R_THRESH = 0.85) with 10 consecutive partitioned waveforms (the subsequent waveform after waveform 1 is between QRS-complexes 2 and 3), then "temporary" template is "accepted"
6. If low correlation is found before a count of 10 is reached (r <= 0.85) then the count starts over, and the following waveform is accepted as the "temporary" template
7. Given an "accepted" template, the correlation between the template and each waveform is calculated, and compared to an acceptable threshold (r > 70% of R_THRESH = 0.85)***
8. The final 10.24 seconds (the span of 10 measurements, sampled every 1.024 seconds) are analyzed, and the percentage of beats which are unacceptable (number "invalid" over number of beats in 10.24 seconds) is calculated and compared to α
9. If percent "invalid" > alpha then the alarm is classified as artifact
10. α is ranged from 0.025 to 0.5 (2.5% unacceptable to 50% unacceptable) to generate ROC curves

- ***In addition to the indication as to whether the data was well-correlated (0 meaning no correlation, and 1 indicating acceptable correlation), whether the ECG was likely to be noisy was considered. If the difference in the time between the current ECG beat annotations and the duration of the template exceeded 40%, then the ECG was considered noisy. ("ecgbad = 1").

Each episode for most patient records had a single "ideal" template established by searching for 10 consecutive "stable" waveforms. Most patients had rather typical ABP waveforms, so that a stable waveform was easily found. A special group of patients for which a different approach had to be developed, due to the effect of changing the balloon pump settings, were those which had an intra-aortic balloon pump (IABP). For each IABP patient, the algorithm had the option of two templates, one for the beat which was assisted by the balloon pump, and one which was not assisted. The "double-template" algorithm, the motivation for which can be seen by referencing the results, is discussed in the outline to follow. Note that in the "double-template" algorithm, each BP waveform was split into a first half and a second half, and correlated with the corresponding portion of the template. This allowed for the verification that a drop in correlation of the entire waveform was due to the absence or presence of the peak in pressure caused by the balloon pump. The verification was accomplished through correlating an arbitrary waveform with

each template, and ensuring that the correlation of the first half of the waveform, which is that due to the heart pumping, remained high for both templates, while the correlation of the last half of the waveform (where the peak, when it was present, was due to the balloon pumping) was high for only one template.

1. Assume one "temporary" template has been found (see outline of basic algorithm above)
2. If r < R_THRESH, then before restarting count of acceptable waveforms, first check whether a second template is needed
3. If number of templates = 1 and Corr.first_half > Corr.second_half and Corr.first_half > 0.7*R_THRESH, then create second template (this is case of 1:2 balloon pump to heart beat ratio)
4. Given two templates, the correlation of an arbitrary waveform is the maximum of the correlation between the two available templates.
5. The count of consecutive highly correlated waveforms increases from 10 to 20 (because twice as many templates need to be verified).
6. Given two "accepted" templates, the correlation of an arbitrary waveform is the maximum of the correlation between the two available templates
7. Assume that Corr2 (correlation with template 2) is maximum of Corr2 and Corr1. Condition 1: |Corr2.first_half - Corr1.first_half| < 0.2 and Corr2.second_half > Corr1.second_half
8. If Corr2 is accepted by the criteria described in step 7, then the arbitrary beat corresponds with template 2, the correlation is Corr2 and the "type" is 2.
9. Otherwise, the dual criterion for template 1 must hold for Corr1 to be the correlation and "type" = 1.
10. If criteria are not met for either template 1 or 2, then "type" is -1 (negative one).
11. For each waveform, the following information is output: time, Corr_all, Corr_1st_half, Corr_2nd_half, "invalid", "type", "ecgbad", beat_length

- "Invalid" assumes values of 0 or 1, 0 indicating acceptable; "Type" can assume "1" or "2", if 2 templates, or "0" or "1", if 1 template ("1" indicates change in rhythm from baseline); "ecgbad" assumes values "0" or "1", where "0" indicates the acceptable case, and "1" indicates that the correlations are most likely corrupted by the fact that the window is too large, and probably includes more than one blood pressure waveform.

It is clear from the above annotations that more information is available than simply whether artifact is present. Indeed, from the series of waveform "type" annotations, the exact setting of the intra-aortic balloon pump can be determined, which is useful in subsequent "automatic" diagnosis of the patient. However, for the purpose of classifying the type of alarm, only the information regarding the existence or non-existence of artifact was necessary.

3.1.3 Assessment of Detection Algorithm

In figure 6 is the general method used for assessing the efficacy of a detection algorithm. The goal of the cascaded systems is to use the strengths of each algorithm to remove as much artifact as possible without dismissing real changes as artifact.

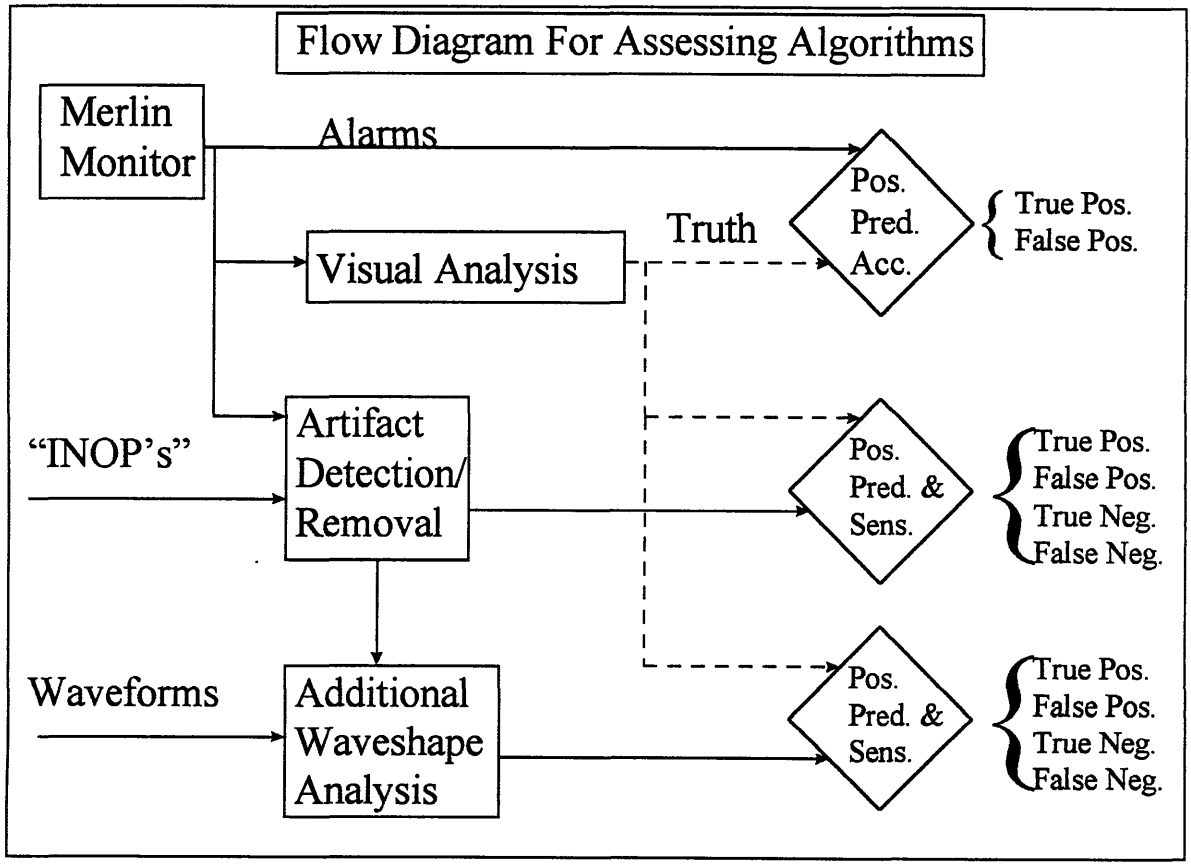

Figure 6: Algorithm Assessment

One of the most important steps in determining the efficacy of a classification algorithm is creating an ROC (Receiver operating characteristic) curve, which is a plot of the probability of a detect (P_D) versus the probability of failure (P_F) (which is the percentage of false positives over the total number of artifacts) for each value of the parameter α (one attempts to vary α such that the

range of P_D and P_F – that is, from 0 to 1 – is covered). The closer the algorithm comes to reaching, for some α, the point (0,1), which is a 100% probability of detection, and a 0% probability of failure, the more effective the algorithm is in accepting "real" changes and excluding "artifact" changes. Equations (3.6) and (3.7) were used to calculate the two probabilities, P_D and P_F, for creating the ROC curves. The value TP is equal to the total number of "True Positive" detections by the algorithm being assessed, and the value FN is equal to the total number of "False Negative" detections (here, this is classifying a "real" change as an "artifact"). The value FP is the total number of "False Positive" detections (classifying an "artifact" as a "real"), and TN is appropriately classifying an artifact. The conditional probabilities refer to two hypotheses: H_0 and H_1. H_1 is the hypothesis that the alarm is "real", while hypothesis H_0 is the case where the alarm is "artifact". P(D1 | H1) is the probabilistic description of the instance where a "real" alarm is classified as "real". P(D1 | H0) describes the instance where an "artifact" alarm is classified as "real" (probability of a false alarm).

$$P_D = P(D_1|H_1) = \frac{TP}{TP + FN} \tag{3.6}$$

$$P_F = P(D_1|H_0) = \frac{FP}{FP + TN} \tag{3.7}$$

For the particular α which optimizes P_D and P_F, the classification results are summarized by calculating positive predictive accuracy and sensitivity statistics. The method of optimization can be according to any reasonable "cost" function. The criterion used for accepting a value for a was to require a greater than 95% sensitivity, while not allowing the positive predictive accuracy to decrease below 60%. The positive predictive accuracy can be compared to the true alarm classifications, the method for which is described above in section 3.1.1 Visual Artifact Detection. The "confusion matrix" below is used to define both positive predictive accuracy and sensitivity.

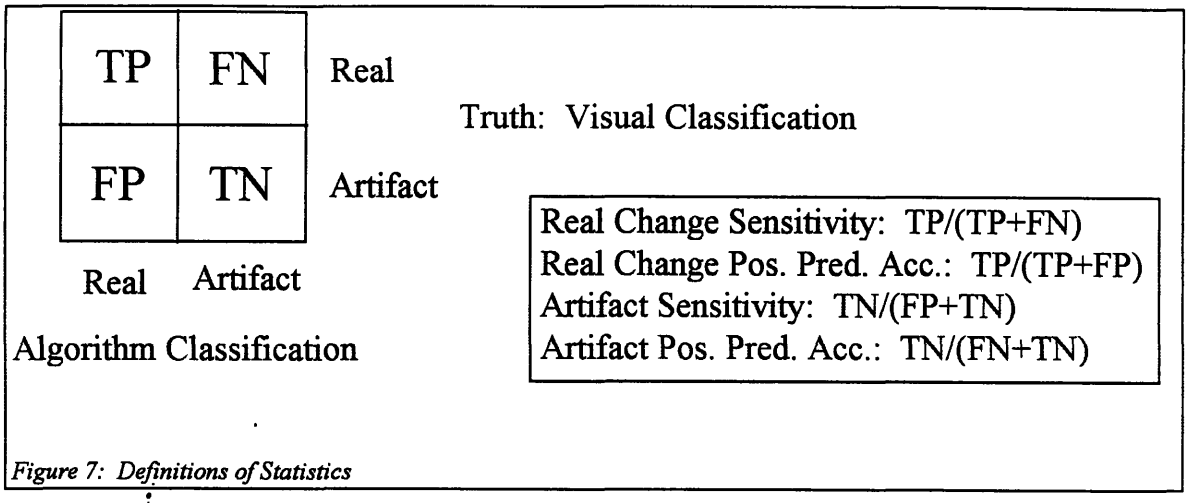

Figure 7: Definitions of Statistics

In the "confusion matrix", "T" is true, meaning the visual and automated classifications coincide. Conversely, "F" (false) indicates that the results of the visual analysis and the algorithmic analysis differ. "P" is the abbreviation for positive, meaning the automated algorithm classified as a real detect, while "N" is negative, meaning the automated algorithm classified the alarm episode as "artifact", or "no-detect". The ideal algorithm has a maximal (hopefully 100%) "real detect" positive predictive accuracy under the constraint of 100% "real change" sensitivity. In short, it is undesirable to lose the perfect sensitivity of the monitor – by definition perfect, since we have restricted ourselves to the alarms as the triggering events – at the expense of perfectly detecting artifact. In short, in order to obtain an optimal setting for each algorithm, a "cost function" must be defined and minimized, whereby certain errors are penalized more than other errors using a weighting scheme, and the sum of all weighted errors is then minimized.

3.2 Physiologic Classification

After deciding which alarms are valid, the parameters in each alarm region were analyzed, and each alarm "episode" was classified as one of several typical changes in the physiology. This type of automated analysis aids the physician in making the decision as to whether the change is due to a problem in the vasculature (i.e., a problem with blood volume), or a change in the pump (i.e., a problem with contractility or heart rate). All three measurements (systolic ABP, HR, and

diastolic PAP) in the alarm region were compared to the data from the beginning of the record to the time of the alarm in order to classify the change. In order to make the measurement and two-dimensional plots, a 20% trimmed mean of each variable was calculated over a window of 7 minutes. Successive trimmed mean windows overlapped by 90% each time, so that the moving window included 10% new data each time the calculation was performed. The effect of this procedure was a smooth waveform, thus rejecting the artifact which tends to confuse the visual analysis of large quantities of data. These data were plotted in two dimensions (e.g., ABP versus HR), and the direction of the change was noted, provided it was "significant" according to a visual analysis of the data.

Categorization of each alarm region was accomplished by using plots of the three measurements (systolic ABP, heart rate, and diastolic PAP) versus time and two-dimensional plots, which consist of two measurements plotted versus one another, and the same measurements plotted versus time in such a way as to line up the axis of each parameter.

Three types of changes were possible for the HR and PAP parameters: a positive change, a negative change, and no change. Because the hypotensive alarms, which indicate a drop in blood pressure, and the hypertensive ABP alarms, which indicate a rise in blood pressure, are those which are used to specify each episode, only two types of changes (positive or negative) were noted in the results. The following set of plots show one example of an alarm episode. The 12^{th} episode of record 230 is the occurrence of a hypotensive alarm at 2.22 hours into the patient record. An open circle (O) at that time indicates the initiation of the alarm. The large increase in heart rate at the same time is obvious from the HR measurement series. The set of 2 dimensional plots in figure 9 aids in correlating the alarm indicators in the two-dimensional plot of ABP versus HR with the time series plots of ABP and HR.

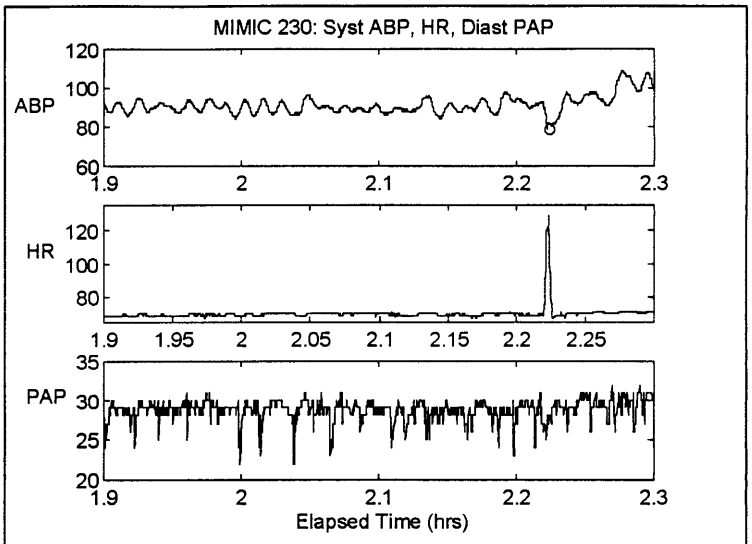

Figure 8: Measurements used in Physiologic Classification

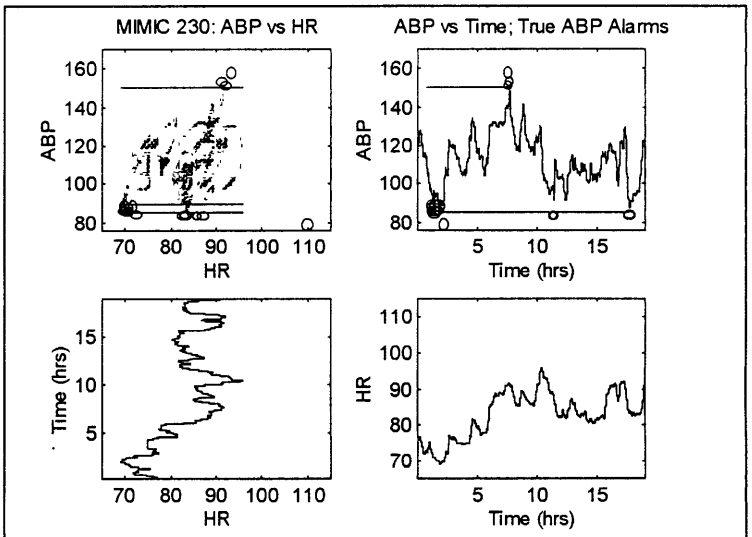

Figure 9: 2-D Plot with Projections

The above alarm episode shows a very distinct change in two of the three parameters (systolic arterial blood pressure, from which the alarm was triggered) and heart rate. The circle at point (80,110) – that is, 80 mmHg ABP and 110 bpm HR – in the plot of ABP vs HR is the alarm episode associated with the data in figure 8. By simply visually analyzing the measurement series, and using the method described above, this episode is of the class (-,+,0, which is ABP decreasing,

HR increasing and PAP remaining the same). When the arterial blood pressure drops and the heart rate increases, indicating that the two parameters are negatively correlated, the hypothesis can be one of two possibilities. The first possibility is that the heart rate began to increase rapidly (tachycardia), thus decreasing the amount of time the heart can fill (with no change in contractility), resulting in a decrease in ABP. Alternatively, the blood pressure could have begun to drop due to a lowered systemic vascular resistance or decreased blood volume, so that the heart rate attempted to compensate via the baroreceptor reflex. Using only the direction of change in the measurements, both hypotheses are consistent with the data, and only a more detailed analysis of the ECG would be able to determine that this episode was in fact due to a run of ventricular tachycardia. In addition, the temporal course of the changes in ABP and HR favors the hypothesis that the change was induced by an arrhythmia. However, the precise diagnosis is outside the scope of this research. In short, the method employed to physiologically classify each episode uses the correlated changes in direction for the three measurements.

The myriad time series and two-dimensional plots have value of themselves for further research in automated classification algorithm development, and these plots are included in a separate appendix.

4 Results

Two key results have come from this study of classifying arterial blood pressure alarms, and categorizing the physiologic change in each alarm region. The first important result was establishing the pivotal role of a waveform analysis algorithm in correctly classifying "real" and "artifact" alarm events. The second key finding, which was obtained using the ABP vs. PAP plots and the ABP vs. HR plots which indicated alarm episodes, was that an effective diagnosis can be obtained using only the correlated changes in three physiologic measurements.

4.1 Artifact Detection

In general, the artifact detection algorithms which utilized only the linear trend in the measurement series provided only minimal improvements in the positive predictive accuracies over those of the HP monitor. In addition, although the trending algorithms reduced the number of false positive detections, false negative detections were concurrently introduced, thus leading to the potentially life-threatening situation of neglecting a significant physiologic change requiring treatment. Therefore, an analysis of the waveshape integrity was performed, and very promising results were obtained. Combining the two methods – linear trending and waveshape analysis – allowed for further improvements in the classification results.

4.1.1 Visual Artifact Detection

Although both the ABP measurements and the ABP waveform were carefully inspected, the visual method used for detecting artifact was dependent mostly on the arterial blood pressure waveform. The results of this visual analysis were used as "truth", to which the results of automated algorithms could be compared. An example of a signal with artifact follows.

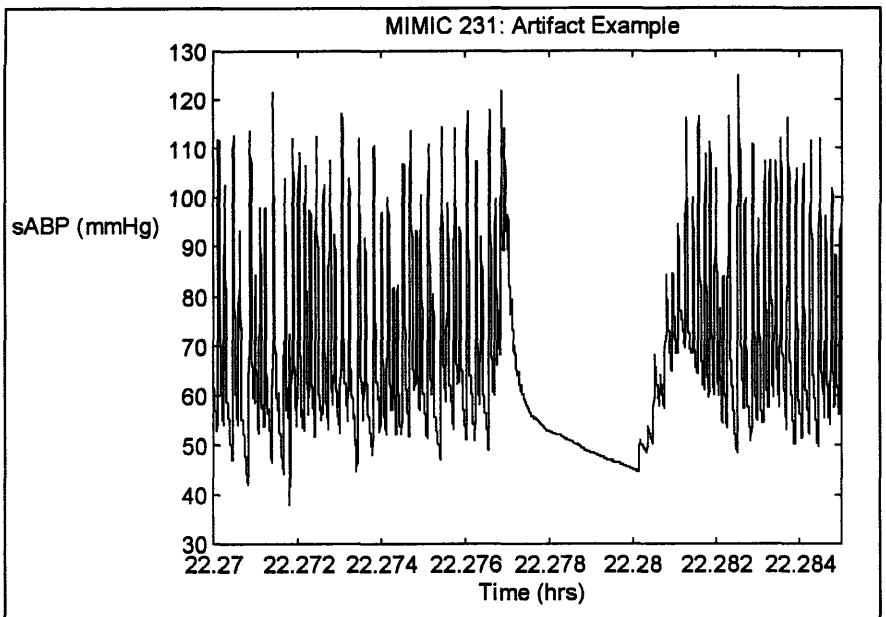

Figure 10: Visual Artifact Detection Plot

This artifact would cause a significant change in the systolic blood pressure, but the change is most obvious when viewing the raw waveform itself. At times, however, artifact was not so obvious, and more resolution was needed to discover the corrupted waveforms. Approximately 300 episodes were classified by similar visual inspection of both the measurement data and the waveforms..

The following table summarizes the number of alarms for each record used in this study, as well as the percentage of false alarms, using only the alarms resulting.

Patient	Artifact Detects	Real Detects	Pos. Pred. Accur.
055	5	4	44.4%
212	4	28	87.5%
215	2	27	93.1%
216	0	33	100.0%
230	1	19	95.0%
231	2	2	50.0%
240	0	31	100.0%
241	2	5	71.4%
242	2	0	0.0%
245	6	0	0.0%
248	1	4	80.0%
415	2	8	80.0%
417	8	2	20.0%
418	2	1	33.3%
420	21	42	66 7%
421	17	38	69 1%
All 16	**75**	**244**	**76.5%**
		Average	**61.9%**
First 14	**37**	**164**	**81.6%**
		Average	**61.1%**

Figure 11: "Merlin" ABP Alarm Summary

In the chart of figure 11, each continuous series of alarms is counted as one "detect". In particular, for a hypotensive alarm (low blood pressure), if the blood pressure continues to decrease after the alarm initially sounds, then a single alarm "detect" consists of many alarms; the alarm value is the minimum value attained since the blood pressure dropped below the minimum threshold. This "event" definition was chosen so that the positive predictive accuracy statistics were not biased by events of long duration. The summary statistics at the bottom of figure 11 show that the gross positive predictive accuracy (summing over all events) was 81.6%, while the average positive predictive accuracy (average over all patients) was 61.1%. Therefore, an ABP alarm from the Merlin monitor for a typical patient in this subset of the database is, on the average, 61.1% predictive of a real hypo- or hypertensive episode.

4.1.2 Automated Artifact Detection

The automated algorithm of section 3.1.2 was run on all 16 patients using the three basic protocols for detecting artifact. The first protocol implemented a linear foreacasting method, which used the trend of the data both 5 minutes and 1 minute previous to the alarm to determine whether the alarm should be classified as real or artifact. The algorithm is described in detail in section 3.1.2 Automated Artifact Detection.

The following ROC curve summarizes the operation of the linear trend algorithm for detecting artifact. The algorithm achieves a nearly 96% probability of detecting a real change (i.e., classifying a true "real" change as "real"), with a probability of failure (classifying a true "artifact" as "real") of only 54%. This seems acceptable, but the remaining 4% of the real cases classified as artifact is undesirable. In addition, the probability of detecting an artifact as real, that is, the probability of failure (abscissa in the ROC plot) is relatively high at 54%, as will be obvious when it is compared to subsequent algorithms.

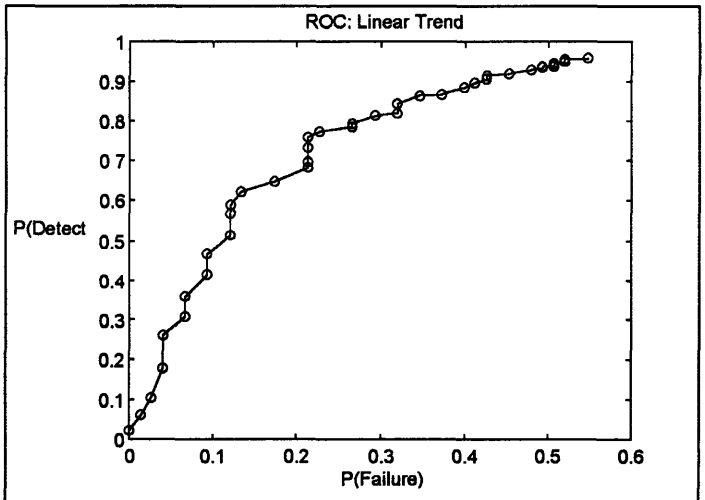

Figure 12: ROC: Linear Trend Algorithm

The overall shape of the ROC curve in figure 12 indicates that a linear trend algorithm, although relatively simple in design, is a reasonable first approximation for detecting artifact. The non-smooth nature of the plot suggests that certain real alarms are "clustered", in that once the threshold is increased such that one more blood pressure alarm is classified as "real", several blood pressure alarms "nearby" (in the sense of the blood pressure alarm value's distance from the linear trend forecast at that time) are also classified as "real".

From the data used to create the ROC curve, two separate tables were created to elucidate on which patient records the linear trend algorithm performed well, and on which patient records the algorithm did not perform well. The following data is for alpha = 0.17, which is an unusually

low threshold level for the linear trend algorithm, and therefore the algorithm should reject nearly everything. In this case, a low P_D is expected. Consequently, although the positive predictive accuracy is perfect (100%) the extremely low "real sensitivity" of 1.5% is expected, and it is clear that the threshold must be increased greatly in order to detect real changes. One point which may be made from this chart is that there were 6 TP detections even for such a low threshold value. These should remain to be detected as "real" as alpha increases.

Alpha=0.17	TP	FN	FP	TN	Real Sens	Real PPA	Art Sens	Art PPA
m055	0	4	0	5	0.0%	n/a	100.0%	55.6%
m212	4	24	0	4	14.3%	100.0%	100.0%	14.3%
m215	1	26	0	2	3.7%	100.0%	100.0%	7.1%
m216	0	33	0	0	0.0%	n/a	n/a	0.0%
m230	0	19	0	1	0.0%	n/a	100.0%	5.0%
m231	0	2	0	2	0.0%	n/a	100.0%	50.0%
m240	0	31	0	0	0.0%	n/a	n/a	0.0%
m241	0	5	0	2	0.0%	n/a	100.0%	28.6%
m242	0	0	0	2	n/a	n/a	100.0%	100.0%
m245	0	0	0	6	n/a	n/a	100.0%	100.0%
m248	0	4	0	1	0.0%	n/a	100.0%	20.0%
m415	0	8	0	2	0.0%	n/a	100.0%	20.0%
m417	0	2	0	8	0.0%	n/a	100.0%	80.0%
m418	0	1	0	2	0.0%	n/a	100.0%	66.7%
m420	0	42	0	21	0.0%	n/a	100.0%	33.3%
m421	1	37	0	17	2.6%	100.0%	100.0%	31.5%
Average					**1.5%**	**100.0%**	**100.0%**	**38.3%**
Gross	**6**	**238**	**0**	**75**	**2.5%**	**100.0%**	**100.0%**	**24.0%**

Figure 13: Results: Linear Trend Algorithm; Alpha=0.17

Alpha=6.7	TP	FN	FP	TN	Real Sens	Real PPA	Art Sens	Art PPA
m055	4	0	4	1	100.0%	50.0%	20.0%	100.0%
m212	27	1	2	2	96.4%	93.1%	50.0%	66.7%
m215	26	1	1	1	96.3%	96.3%	50.0%	50.0%
m216	25	8	0	0	75.8%	100.0%	n/a	0.0%
m230	19	0	0	1	100.0%	100.0%	100.0%	100.0%
m231	2	0	0	2	100.0%	100.0%	100.0%	100.0%
m240	31	0	0	0	100.0%	100.0%	n/a	n/a
m241	5	0	1	1	100.0%	83.3%	50.0%	100.0%
m242	0	0	0	2	n/a	n/a	100.0%	100.0%
m245	0	0	2	4	n/a	0.0%	66.7%	100.0%
m248	4	0	0	1	100.0%	100.0%	100.0%	100.0%
m415	8	0	2	0	100.0%	80.0%	0.0%	n/a
m417	2	0	4	4	100.0%	33.3%	50.0%	100.0%
m418	1	0	0	2	100.0%	100.0%	100.0%	100.0%
m420	42	0	16	5	100.0%	72.4%	23.8%	100.0%
m421	38	0	9	8	100.0%	80.9%	47.1%	100.0%
All 16	**Average**				**97.8%**	**79.3%**	**61.3%**	**86.9%**
Gross	**234**	**10**	**41**	**34**	**95.9%**	**85.1%**	**45.3%**	**77.2%**
First 14	**Average**				**97.4%**	**79.7%**	**65.6%**	**84.7%**
Gross	**154**	**10**	**16**	**21**	**93.9%**	**90.6%**	**56.8%**	**67.7%**

Figure 14: Results: Linear Trend Algorithm; Alpha=6.7

From figure 14, several comparisons can be made to the statistics from the "Merlin" monitor. At an alpha of 6.7 (i.e., the criterion for accepting an alarm is that it be within 6.7 error units, where an error unit is described in the methods above), the average value for the "real detect" positive predictive accuracy is 79.3% using all 16 records (for comparison to the waveform analysis algorithm discussed below, the average PPA is 79.7% and the gross PPA is 90.6% using the first 14 records, m055 through m417); the positive predictive accuracy calculated above for the "Merlin" monitor, without the aid of additional data analysis, was 61.1%, showing that the trending algorithm improved the predictive value of the ABP alarm by 17%. However, this was done at the expense of the real alarm sensitivity rate, which decreased from 100% (by definition) to 97.8%. Therefore, approximately 2.2% of the alarms which are actual hypo- or hypertensive episodes are ignored on the basis that they are thought to be caused by artifact.

The second protocol we implemented helped to assess the utility of considering the "INOP" annotations in appropriately removing artifact.. All data within a window 10 seconds preceding an "INOP" annotation, until the time of the annotation itself, was removed from consideration, and the identical linear forecasting method of the first protocol was used to determine whether the alarm was real or artifact. A 10-second window was used because this is the time over which the blood pressure must remain in an alarm state (above or below threshold) before an alarm is triggered. The ROC curve for the linear trend method with the INOP data removed is in figure 15 below.

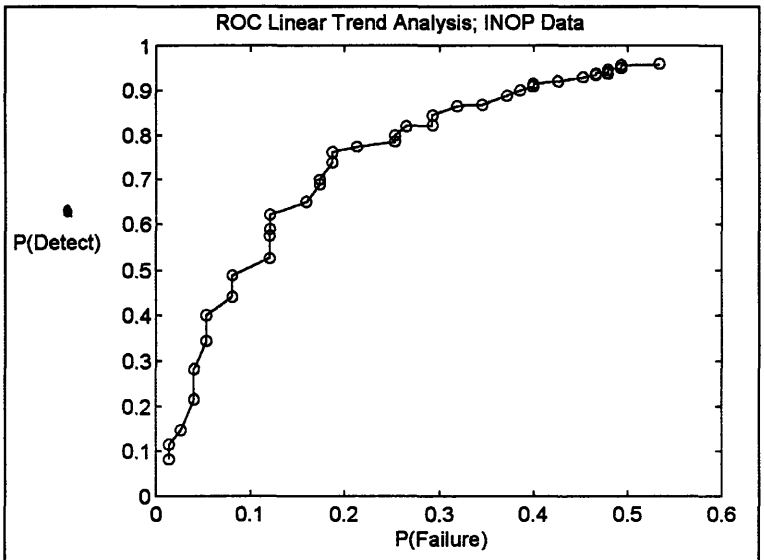

Figure 15: ROC: Linear Trend: INOPs Removed

The INOP conditions are used in the monitor such that, when a transducer is not operating correctly, no alarm for the measurement made by that transducer will be triggered. This type of conditional alarming in a monitor is expected. Therefore, the purpose of removing the data which was corrupted by INOP data was simply to improve the linear trending method, since the variance which would be acceptable in a region with much artifact would allow an "artifact" alarm to be classified as "real". Removing the data in an INOP region ought to remove the data with high variance, which theoretically should allow a more restricted acceptable region, thus decreasing the number of false positive detections.

Alpha=6.7	TP	FN	FP	TN	Real Sens	Real PPA	Art Sens	Art PPA
m055	4	0	4	1	100.0%	50.0%	20.0%	100.0%
m212	27	1	2	2	96.4%	93.1%	50.0%	66.7%
m215	26	1	1	1	96.3%	96.3%	50.0%	50.0%
m216	25	8	0	0	75.8%	100.0%	n/a	0.0%
m230	19	0	0	1	100.0%	100.0%	100.0%	100.0%
m231	2	0	0	2	100.0%	100.0%	100.0%	100.0%
m240	31	0	0	0	100.0%	100.0%	n/a	n/a
m241	5	0	1	1	100.0%	83.3%	50.0%	100.0%
m242	0	0	0	2	n/a	n/a	100.0%	100.0%
m245	0	0	3	3	n/a	0.0%	50.0%	100.0%
m248	4	0	0	1	100.0%	100.0%	100.0%	100.0%
m415	8	0	2	0	100.0%	80.0%	0.0%	n/a
m417	2	0	4	4	100.0%	33.3%	50.0%	100.0%
m418	1	0	0	2	100.0%	100.0%	100.0%	100.0%
m420	42	0	14	7	100.0%	75.0%	33.3%	100.0%
m421	38	0	9	8	100.0%	80.9%	47.1%	100.0%
All 16		**Average**			**97.8%**	**79.5%**	**60.7%**	**86.9%**
Gross	**234**	**10**	**40**	**35**	**95.9%**	**85.4%**	**46.7%**	**77.8%**
First 14		**Average**			**97.4%**	**79.7%**	**64.2%**	**84.7%**
Gross	**154**	**10**	**17**	**20**	**93.9%**	**90.1%**	**54.1%**	**66.7%**

Figure 16: Results Excluding "INOP" Data

The important values for comparing the results of linear trend algorithm which excludes "INOP" data, using only the first 14 records, to other algorithms using the same records are the average PPA of 79.7%, the gross PPA of 90.1%, in addition to the average sensitivity of 97.4% and the gross sensitivity of 90.1%.

In order to compare the improvement which excluding the INOP data provided, the ROC curve below includes the ROC curves from both protocols: the linear trending, and the linear trending with INOP data removed. The improvement of the algorithm by simply removing knowingly corrupted data is not drastic. The ideal ROC curve is close to the point (1,0), which is the case of detecting all real changes while rejecting all artifact changes. However, the linear trending algorithm which first removes the corrupted INOP data does perform slightly better.

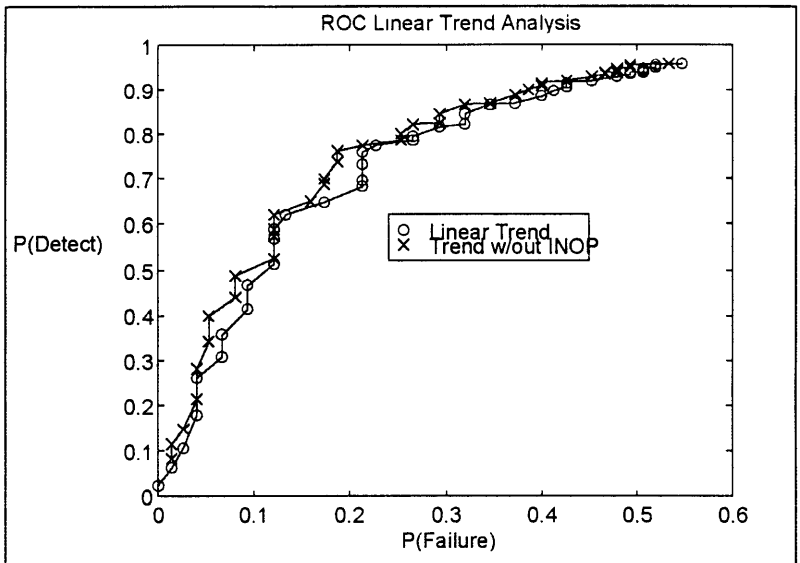

Figure 17: ROC: Trending With and Without INOP Data

In the legend of figure 17, "Linear Trend" denotes the standard linear trend method using all the data in the arterial blood pressure measurement series of all 16 records, while "Trend w/out INOP" denotes the protocol which implemented the same linear trend method on the same records, but with the data corrupted by an inoperative transducer removed. The removal of the corrupted data yields identical or improved results over the entire ROC curve.

Each of the false negative episodes were visually analyzed to motivate an improvement of the linear trending algorithm. Although from visual analysis of the arterial blood pressure waveforms it is obvious that artifact caused these changes, the measurements themselves exhibit no drastic change in value which suggests a limitation of the linear trending algorithm. This is due to the fact that the measurements are low-pass filtered, and do not reflect individual beat-to-beat changes in the waveform shape.

The fact that the first two protocols had less than optimal sensitivity results suggests the difficulty of detecting whether artifact is the cause of an ABP alarm using only the arterial blood pressure measurements and "INOP" notations. Unlike heart rate, which is frequently verified using the plethysmography "PULSE" rate, the blood pressure can not be verified using an alternative

signal, except in the extreme cases where the blood pressure is zero, but the heart rate is non-zero. However, this extreme case scenario was not present in these patients, where an incorrect classification would have been rectified if heart rate were also considered. These results motivate the need for an artifact detection scheme which includes analysis of the ABP waveform itself .

4.1.3 Beat-by-beat Waveshape Analysis

While assessing the utility of the linear trend method, it became obvious that there were several "real" changes which were consistently classified as artifact due to their rapid time course. An example is record 216 which had 11 intra-aortic balloon pump setting changes which resulted in the arterial blood pressure alarm being triggered. An automated system needs to robustly detect these real changes in blood pressure.

Through correlating an ideal arterial blood pressure waveform template with the blood pressure waveform in the five minute period prior to each alarm, correlation coefficients were obtained on a beat-by-beat basis, and the correlation values were stored. These data were used in an algorithm similar to that used for the "INOP" information, where measurements which corresponded to "artifact" beats were noted and excluded from the linear trending algorithm.

In figure 18 is a plot of the correlations for a portion of record 216 as determined by the waveform analysis program. The ABP measurements are in the upper plot, while the beat-by-beat correlations of each waveform with a template are in the lower plot. The extremely high degree of correlation ($r > 0.99$) for the entire waveform ("All") continues until time 0.745 hours, when a sudden drop in the blood pressure due to a brief interruption in the functioning of the balloon catheter (cardiac assist) causes the shape of the entire waveform to change drastically. However, the first half of the waveform, that due to the contraction of the heart, continues to have a relatively high degree of correlation (r changes from 0.999 while the balloon catheter was functioning to 0.87 when the balloon did not inflate). The second half of the waveform, corresponding to the portion

where the balloon creates an increased pressure, correlates very poorly with the template when the balloon ceased to inflate (r changed from 0.90 to -0.32).

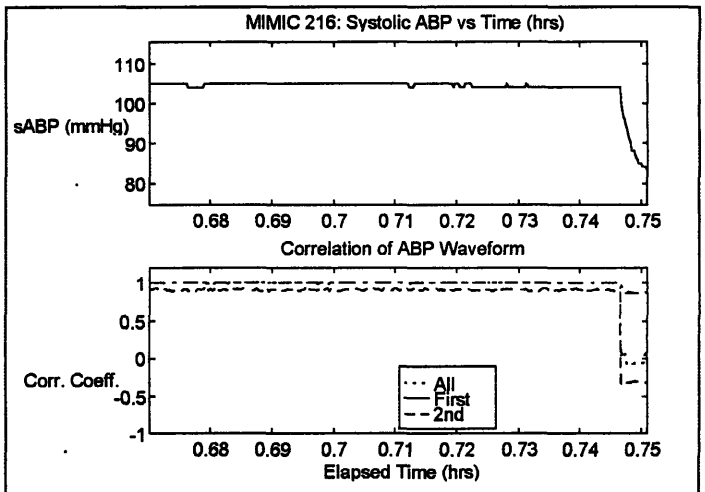

Figure 18: Beat-by-beat Artifact Detection

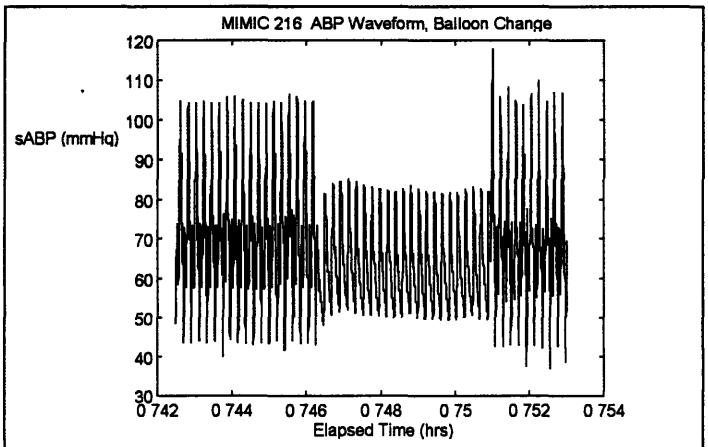

Figure 19: ABP Waveform - Transient Change

In figure 19 is a portion of the ABP waveform analyzed to create figure 18. In addition to the correlation coefficients of figure 18, the waveform analysis program automatically classifies the type of rhythm detected, if the patient has an intra-aortic balloon pump (as does patient 216), and whether each beat is artifact or real. The following plot shows the "type" (i.e., the change is assist ratio; type "0" is the original ratio, while type "1" is a novel ratio.) and whether the data is corrupted ("invalid") for the second episode (hypotensive ABP alarm) of MIMIC Record 216 (note that it includes part of the same data as the figures above). The first plot in figure 20 shows that

the waveform is never classified as "invalid" (i.e., "invalid" remains 0), while the "type" (i.e., the assist ratio class) changes from type "0" to type "1" transiently. These transient changes in waveform "type" are reflected also in the fact that the correlation of the first half of the waveform remains high (near 0.9) while the correlation of the entire waveform drops to near 0, synchronously with the changes in type.

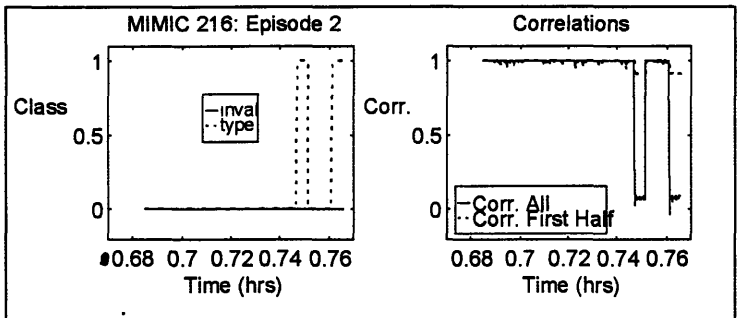

Figure 20: Waveform Analysis: MIMIC 216

These results show that an automated procedure which uses the correlation value over three ranges (entire waveform, first half, second half), functions exceptionally well in detecting changes in the balloon pump settings, and therefore allowing the arterial blood pressure alarm, which frequently occurs in patients with balloon catheters, to be correctly classified as a "real" alarm. Most of the problems which the linear trend algorithm experienced were with data of this type, where the trend is very fast – which is suggestive of artifact – but the waveform maintains its integrity throughout the alarm episode.

In addition to being able to detect drastic changes in the arterial blood pressure waveform integrity, the series of correlation coefficients are analyzed for periodic changes, which result when a patient with a catheter balloon pump undergoes changes in the assist ratio, whereby the balloon inflates after every other or every third heart beat (a 2:1 or 3:1 ratio, respectively). These ratio changes must be robustly detected in order to decrease the number of false alarms, since changes in the ratios impact greatly on the systolic and diastolic blood pressure measurements.

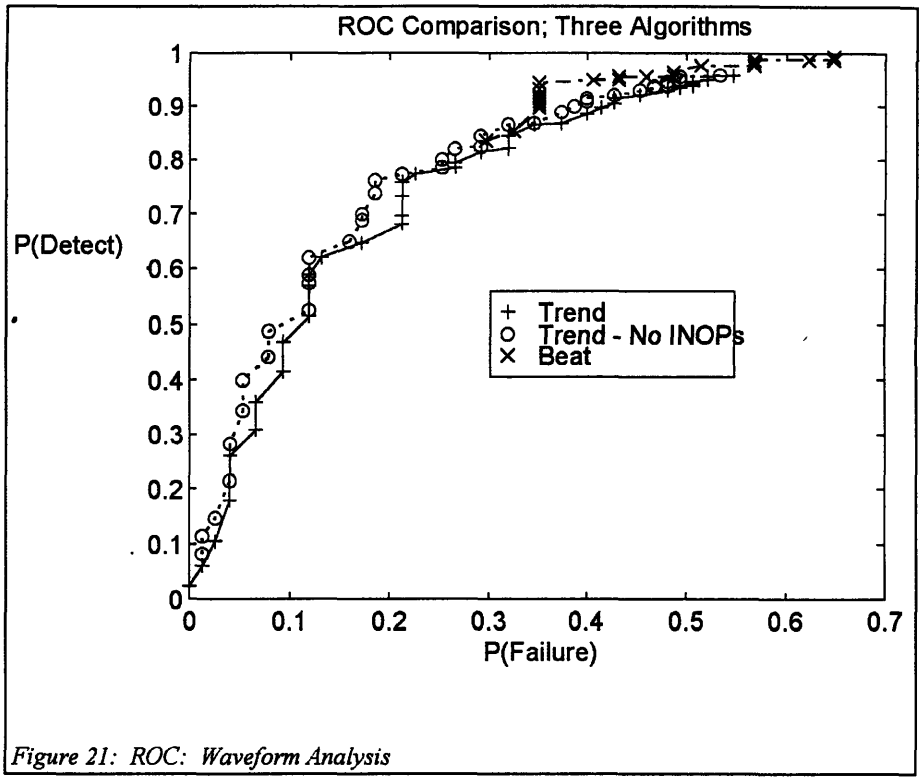

Figure 21: ROC: Waveform Analysis

Figure 21 shows the ROC for all three algorithms, including the results of the beat-to-beat waveform analysis. The parameter alpha ranged from 1.1% to 55.6%, and was the threshold for the percentage of "corrupted" waveforms above which an episode was classified as artifact. The data which was analyzed to obtain this was a window of duration 10.24 seconds, immediately prior to an ABP alarm. It is obvious from these curves that the waveform analysis curve is slightly better than both the linear trend algorithms, and most notably when the probability of failure is 35%.

The chart in figure 22 summarizes the result of the algorithm for alpha = 0.56, which is the upper-right most data point on the ROC curve. The purpose of this algorithm at the outset was to develop a method which maintained a 100% "real change" sensitivity, while maximizing the "real change" positive predictive accuracy (i.e., not introducing "false negative" detections, while minimizing the number of "false positive" instances). In the following chart, the results of the artifact detection algorithm are summarized. Only one FN occurred, and this happened in record

055. The average positive predictive accuracy, over patients, becomes 65.8%, which is only a mild increase over the positive predictive accuracy of the unassisted "Merlin" monitor. However, of the 37 false positive "Merlin"alarm episodes (in the 14 patients analyzed), 13 (30%) of these episodes were correctly rejected.

The reason for the relatively high number of false positive results for record 245 was obvious. In order to err on the side of caution, the waveform analysis algorithm classifies as "real" any episode for which no correlation data prior to the alarm exists. This was predominantly seen in record 245, where all 6 FP results were due to the fact that no correlation data existed for the 5-minute window prior to the alarm. The only time that no data exists for an episode is when no "stable" waveform is found with which to obtain the correlation data. Typically this does not occur because there is 5 minutes of sustained artifact, but rather because the transducer is not functioning correctly during the time the template is being located. Therefore, although the non-existence of data is highly predictive of artifact, the waveform analysis algorithm classified alarms cautiously, and classified the "artifact" alarms as "real". This behavior of the algorithm is conservative, and should be revised in the future.

alpha=0.56	TP	FN	FP	TN	Real Sens	Real PPA	Art Sens	Art PPA
m055	3	1	2	3	75.0%	60.0%	60.0%	75.0%
m212	28	0	1	3	100.0%	96.6%	75.0%	100.0%
m215	27	0	2	0	100.0%	93.1%	0.0%	n/a
m216	33	0	0	0	100.0%	100.0%	n/a	n/a
m230	19	0	0	1	100.0%	100.0%	100.0%	100.0%
m231	2	0	2	0	100.0%	50.0%	0.0%	n/a
m240	31	0	0	0	100.0%	100.0%	n/a	n/a
m241	5	0	0	2	100.0%	100.0%	100.0%	100.0%
m242	0	0	1	1	n/a	0.0%	50.0%	100.0%
m245	0	0	6	0	n/a	0.0%	0.0%	n/a
m248	4	0	1	0	100.0%	80.0%	0.0%	n/a
m415	8	0	2	0	100.0%	80.0%	0.0%	n/a
m417	2	0	5	3	100.0%	28.6%	37.5%	100.0%
m418	1	.0	2	0	100.0%	33.3%	0.0%	n/a
Average					97.9%	65.8%	35.2%	95.8%
Gross	163	1	24	13	99.4%	87.2%	35.1%	92.8%

Figure 22: Waveform Analysis: Alpha = 0.56

Using a threshold of 56% corrupted data with the waveform analysis algorithm, only 1 FN detection was obtained, which is a 99.4% (163/164) gross average sensitivity. In terms of maximizing the sensitivity of the algorithm, there is much benefit in implementing a method for analyzing the waveform of each beat. However, the gross positive predictive accuracy (87.2%) was lower than the gross PPA of the linear trend algorithm which removes INOP data (90.1%). This result suggests that the linear trend algorithm is more accurate in distinguishing artifact from real changes. However, looking in detail at the waveform analysis results, it was clear that the reason the number of false positives (24) was greater than the number of false positives (17) for the linear trend algorithm with INOP data removed, was that often the waveform program could not find a stable template waveform, and therefore the classification defaulted to "real". This is a problem with the waveform analysis program, which results because only data which is 5 minutes before the alarm is analyzed. If artifact corrupts the waveform for the entire 5 minutes, then 10 consecutive stable waveforms are not found, and no classification can be made. There were 12 episodes for which no stable template was obtained (episode 1 in record 212, episode 15 in 215,

episode 3 in 231, episode 1 of 242, all 6 episodes of 245, episodes 2 and 3 of 418). Changing the algorithm to classify each episode for which the waveform algorithm does not converge as "artifact", the sensitivity does not change (since all such episodes were indeed artifact), but the PPA would become 93.1%, which represents a 2% improvement over the linear trend algorithm which removes the INOP data. A potential method for accomplishing this positive predictive accuracy would be to use the waveform analysis result when the algorithm obtains a template, but refer to the linear trend algorithm when no result from the waveform analysis exists. Combining the results of two methods which are effective independent of one another should increase the PPA over any one method alone, and may increase the sensitivity to 100%.

4.2 Physiologic Classification Results

Each episode which was visually classified as "real", using the methods of section 3.1.1, was also visually categorized according to the direction of change from a "baseline" that each of the physiologic parameters underwent.

The following plots summarize record 230, and in particular focus on episode 12, which occurred at hour 2.22. The general trend of the alarms in record 230 consists mostly of positively correlated changes in ABP and HR (see figure 25) (i.e., for hypotensive alarms, this means that both ABP and HR decreased concurrently). Only alarm episode 12 suggests a significantly different behavior from the other alarms, since it is the only alarm instance where ABP decreases and HR increases. This episode does not stand out in the ABP vs PAP two-dimensional plot (figure 24), since the PAP measurement did not change significantly (ABP of 80 mmHg, which is the lowest ABP alarm value in figure 24, and PAP of 27 mmHg). According to the categorization scheme described in section 3.2, episode 12 is (-, +, 0) – see Appendix B for a categorization of each alarm episode.

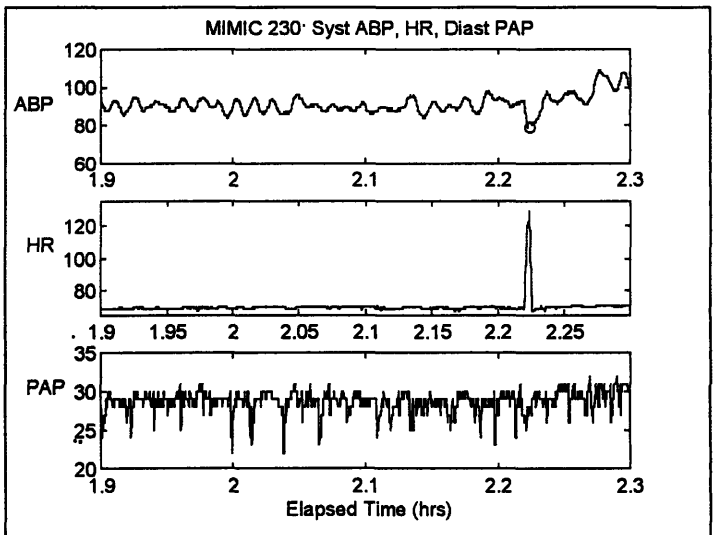

Figure 23: Measurements used in Physiologic Classification

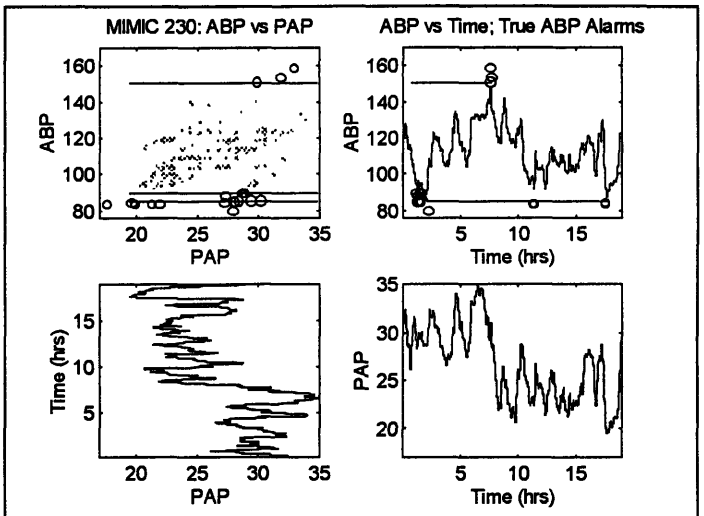

Figure 24: MIMIC 230: ABP vs PAP w/ Alarms

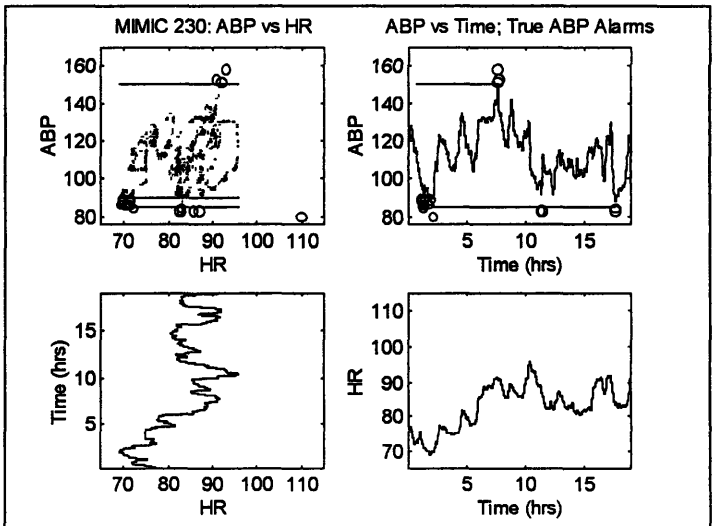

Figure 25: MIMIC 230: ABP vs HR w/ Alarms

The isolated circle at point (80,110) – that is, 80 mmHg ABP and 110 bpm HR – in the plot of ABP vs HR in figure 25 is the alarm episode associated with the data in figure 23. When the arterial blood pressure drops and the heart rate increases, indicating that the two parameters are negatively correlated, the hypothesis can be one of two possibilities. The first possibility is that the

heart rate began to increase rapidly (tachycardia), thus decreasing the amount of time the heart can fill (with no change in contractility), resulting in a decrease in ABP. Alternatively, the blood pressure could have begun to drop due to a lowered systemic vascular resistance or decreased blood volume, so that the heart rate attempted to compensate via the baroreceptor reflex. Using only the direction of change in the measurements, both hypotheses are consistent with the data. The time course of the HR and ABP data suggest that in this subject tachyarrhythmia might be the cause. Only a more detailed analysis of the ECG would be able to determine that this episode was in fact an episode of ventricular tachycardia.

Approximately 200 additional alarm episodes were categorized; however describing the potential pathology which led to the alarm condition is outside the scope of this research. The two-dimensional plots of the measurements (see Professor Roger Mark for Appendix C), together with the clinical information in the MIMIC database may be combined to formulate a more complete physiologic rationale for the alarms. In addition, these data can be analyzed to motivate automated algorithms for physiologic classification. Although results were patient-specific, the basic finding was that most alarms for a particular patient are of the same general type (see Appendix B), and that a range of physiologic hypotheses can be suggested by analyzing the trend of the alarms (e.g., all alarms may result from correlated changes in the measurements, with the exception of a few particularly interesting episodes).

5 Discussion

Augmenting the methods which analyze only the measurements to include an analysis of the raw waveforms proved to be essential in improving the positive predictive accuracy while maintaining a high sensitivity of the Merlin Monitor /Additional algorithm system. However, there were a few challenges which arose when implementing the program for artifact detection using the waveforms (artdet.c). The most notable challenge was using ECG analysis as the method for locating the fiducial marks for the arterial blood pressure waveform. The noise and artifact in the ECG waveforms gave spurious QRS-complex annotations, yielding an incorrect "window" of the blood pressure, and therefore giving a much different waveform to compare to the template. Although a notation of a poor ECG correlation was indicated by the waveform analysis program – therefore allowing the waveform to be excluded from consideration in calculating the percentage of "corrupted" waveforms – the analysis was dependent on the ECG signals having a high signal-to-noise ratio. A change in the algorithm to be able to handle the cases where the ECG SNR is low would be helpful. A second challenge in the waveform artifact detection algorithm was described in section 4.1.3. In short, the waveform program needs to be improved by selecting a default template when one cannot be found using the 5 minutes of data immediately preceding the alarm. However, despite the potential for error when correlating raw, unprocessed data, the algorithm robustly handled and appropriately classified most of the arterial blood pressure waveforms.

Combining the results of the two best classification algorithms – the waveform analysis and the linear trend algorithm which excludes the corrupted INOP data – it should be possible to obtain a perfect sensitivity (100%), and a very high PPA (>95%). The reason that a perfect PPA is difficult is that many visually classified "artifact" episodes were borderline cases. Record 055 had the only FN using the waveform analysis algorithm, in addition to two FP detections which seemed to be borderline cases, so that improving the functioning of the algorithm on patient 055, as

well as changing the waveform algorithm to declare "artifact" when no stable template was found, would increase the results to a sensitivity of 100% and a gross PPA of 94.2% (163/173), since 14 false positives would become true negatives. Since the correlation values themselves were not obtained before visually classifying each episode, it is possible that the visual classification was too sensitive to the artifact – that is, the artifact may have been minimal, and therefore not detectable using the threshold (~0.56) that was used to determine whether a single waveform was corrupted.

6 Conclusion

In conclusion, effective methods have been developed for automatically analyzing ABP alarm regions, detecting which result from artifact (a disruption of the waveform integrity) and which are real changes. Moreover, we have found that it is of critical importance to detect artifact appropriately at the level of the waveforms, and to consider methods for analyzing the blood pressure waveform that include splitting the waveform into two separate regions in order to detect special cases such as fluctuations in the cardiac assist pump. Future work should resolve the question of establishing a default template when the waveform analysis program cannot locate a template in the usual manner. Finally, an algorithm which effectively combines the output from the linear trend algorithm and the waveform analysis program should improve further the sensitivity and positive predictive accuracy compared to the results of each independent algorithm. In particular, the number of false negatives (real changes which are classified as artifact) should become zero, while the number of false positives (artifact changes classified as real) should be reduced by approximately 50% of the number of false positives in any one of the algorithms run independently.

Addressing the issue of physiologic classification, each non-artifact alarm region was categorized according to the type of change which occurred using three measurements, systolic arterial blood pressure (sometimes mean ABP), heart rate, and diastolic pulmonary arterial pressure. Plots were made which summarize each event which provide a means for viewing the alarm regions to motivate methods for automatic categorization. A particularly interesting episode was plotted in detail, and suggestions on how to automatically classify the physiologic change were given.

Acknowledgments

I want to acknowledge everybody who has helped me to finish this work. First, I want to thank God, who has helped me through the late, late nights with faith that an end is in sight. I want to thank Dr. Roger Mark, whose hard work and incredible support has helped me to do this research. Also, a very special thank you goes to George Moody, who has provided invaluable support, both by solving computing problems and providing a wealth of ideas. The rest of you I will only mention, although you all mean so much to me: Jim Ryan, Sajjan Sharma, Cedric Logan, Mark Rawizza, Mike Metzger, Jesse Tauriac, Roy and Chelly Larson, Jude Federspiel, William DeShazer, William Potter, Marlon Shows, Nicolas Pujet, Piper Keables, Maria Sisneros, Mom, Dad, Kim, Jeff, Eric, Michael, Corinne, Jennifer Dease. Thank you all so much. To the whole Harvard-MIT-Tufts ministry and Central Region, I love all of you very much. Thanks for the love and support.

For my girlfriend, Lea Dykstra, I want to preserve the lyrics to the song by John Micheal Montgomery, "I Can Love You Like That", the first song I played for her by the pier, overlooking the ocean at the beach in Santa Monica, May 25, 1996.

"They read you Cinderella
You hoped it would come true
That one day your Prince Charming
Would come rescue you

You like romantic movies
You never will forget
The way you felt when Romeo kissed Juliet

All this time that you've been waiting
You don't have to wait no more

CHORUS:
I can love you like that
I would make you my world
Move Heaven and Earth if you were my girl

I will give you my heart
Be all that you need
Show you you're everything that's precious to me
If you give me a chance
I can love you like that

I never make a promise I don't intend to keep
So when I say forever, forever's what I mean
I'm no Casanova but I swear this much is true
I'll be holdin' nothing back when it comes to you
You dream of love that's everlasting
Well baby open up your eyes
REPEAT CHORUS

BRIDGE:
You want tenderness – I got tenderness
And I see through to the heart of you
If you want a man who understands
You don't have to look very far

Bibliography

[1] Cropp AJ, Wood LA, Randy D, Bredle DL. Name That Tone: The Proliferation of Alarms in the Intensive Care Unit. Chest 1994; 105:1217-20

[2] Hewlett Packard Laboratories Merlin Manual.

[3] Zhao Ruilin MIT, Personal discussion. 1996.

[4] Yana K, Saul JP, Berger RD A Time Domain Approach for the Fluctuation Analysis of Heart Rate Related to Instantaneous Lung Volume. IEEE Transactions on Biomedical Engineering, Vol, 40, No 1, January 1993

[5] Shanmugan KS, Breipohl AM, Random Signals: Detection, Estimation and Data Analysis. John Wiley & Sons, 1988. Chapter 3.3.4, pp. 122-24.

Appendix A

The following program "artdet.c" , written in C, is that which was used used to create the results in section 4.1.3. The UNIX (csh) script following it puts the data into the appropriate format, and runs artdet.c.

```
/* Adam Hoyhtya  5/15/95  Blood Pressure Artifact Detector
  This program locates BP maxima and minima, utilizing physiologic
  boundary conditions in order to protect the calculations from noise
  and artifact.
*/

#include <math.h>
#include <stdio.h>
#include <ecg/db.h>
#include <ecg/ecgmap.h>
#include "constants.h"
#include "declars.h"
#include <string.h>

#define BALLOON 1 /* or 0 if no balloon */
#define R_THRESH 0.85 /* THIS IS CORRELATION THRESH FOR 10 CONSEC B4 ACCEPT */
#define BEATS 15  /* THIS IS NUMBER OF ABP WAVEFORMS AVERAGED */
#define SEGLEN 600
#define MAXLEN 20
#define MIN 0
#define MAX 1
#define BP 0
#define PL 1
#define RSP 2
#define SO2 3
#define L 3
#define NUM_PARAMS 3
#define RR_WINDOW 4
#define PARWIN 4
#define BP_PL_WINDOW 10
#define CORR_LIMITS 1000
#define min(A,B)  ((A) < (B) ? (A) : (B))
#define max(A,B)  ((A) > (B) ? (A) : (B))
#define sqr(A)    (A)*(A)
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

char *pname;
int len;

struct trio {
  double max_corr;
  int mc_shift;
  double fh_corr;
```

```
  int fh_shift;
  double lh_corr;
  int lh_shift;
};

double smear_corr (double * temp, double * signal, int size,
                   double *temp64, double *temp128, double * temp256,
                   double * stretch256, struct trio *data);

double find_trio (struct trio *data, double * temp, double *sig, int size,
                  int ltrim, int shift);

void generate_templates (double * template, int size, double temp64 [],
                         double temp128[], double temp256[]);



main(argc, argv)
int argc;
char **argv;
{
  /* variable declarations */
  char *getparam(), *prog_name();
  double calcHR();
  double calcpar();
  double mean();
  void addtime();
  void addpar();
  void initvect();
  double crosscor();
  double normdot();
  double find_good_template();
  void get_temps ();
  double find_trio ();
  char *p, *parameter = NULL, *record = NULL;
  char ifname[20], ann_name[15], *ifpath;
  char datarec[15], calfile[15];
  FILE *ifile;
  int i, n, segment, last_segment, type, evil,ecgbad;
  long rrtimes[RR_WINDOW],BP_PLtdelt[BP_PL_WINDOW],totsecs;
  double avBPv[PARWIN],mxBPv[PARWIN],mxBPtv[PARWIN];
  double mnBPv[PARWIN],mnBPtv[PARWIN],mxPLtv[PARWIN];
  double lastsec, cursec, average;
  double h, m, s, t, tstart = 0.0, tstop = 0.0, tdiv = 60.0;
  struct trio data;
  struct trio data_2;

  double temp64 [65];
  double temp128 [129];
  double temp256 [257];
  double stretch256 [257];
  double temp64_2 [65];
  double temp128_2 [129];
  double temp256_2 [257];
```

```
double stretch256_2 [257];
void help();

double corr[CORR_LIMITS], c1, c2;
double Rlinear;
double tcorr[CORR_LIMITS];
double template[SEGLEN];
double template2[SEGLEN];
double tempave[SEGLEN];
double sig[L][MAXTIME];
static struct siginfo si[MAXSIG];
static struct anninfo a[2];
static struct ann annot;
int nsig, numtemplates;
static int vin[MAXSIG];
long k, j, cal;
int BPn = -1, PLn = -1, RSPn = -1, SO2n = -1;
long prev_time, curr_time, delta_time, cont, bBP, bPL, bRSP;
long templen, templen2;
long time[L][2];
double gBP, gPL, gRSP, F, HR, aveBP;
double peakBP, peakBPt, pitBP, pitBPt, peakPLt;
FILE *fcal, *fp;
char outfil[MAXLEN], poutfil[MAXLEN];
int old_exists = 0;
int flag;
int no_template;
int happy;
double line[SEGLEN];
/* ********************************************************************** */
pname = prog_name(argv[0]);
for (i = 1; i < argc; i++) {
  if (*argv[i] == '-') switch (*(argv[i]+1)) {
      case 'f':  /* start time follows */
        if (++i >= argc) {
          fprintf(stderr, "%s: stop time must follow -t\n", pname);
          exit(1);
        }
        n = sscanf(argv[i], "%lf:%lf:%lf", &h, &m, &s);
    switch (n){
      case 1: tstart = h; break;
      case 2: tstart = 60.0*h + m; break;
      case 3: tstart = 3600.0*h + 60.0*m + s; break;
    }
    break;
  case 'h':      /* help requested */
    help();
    exit(0);
    break;
  case 'r':      /* record name follows */
    if (++i >= argc) {
      fprintf(stderr, "%s: record name must follow -r\n", pname);
        exit(1);
      }
```

```
            record = argv[i];
        break;
      case 'a':    /* annotator name follows */
        if (++i >= argc) {
           fprintf(stderr, "%s: annotator name must follow -a\n", pname);
           exit(1);
            }
        strcpy(ann_name, argv[i]);
        break;
      case 't':       /* stop time follows */
        if (++i >= argc) {
          fprintf(stderr, "%s: stop time must follow -t\n", pname);
              exit(1);
            }
            n = sscanf(argv[i], "%lf:%lf:%lf", &h, &m, &s);
        switch (n){
          case 1: tstop = h; break;
          case 2: tstop = 60.0*h + m; break;
          case 3: tstop = 3600.0*h + 60.0*m + s; break;
        }
        break;
      case 'V':      /* alternate verbose mode, print output sample times */
        tdiv = 1.0;
        if (argv[i][2] == 'm') tdiv = 60.0;
        else if (argv[i][2] == 'h') tdiv = 3600.0;
        break;
      default:
        (void)fprintf(stderr, "%s: unrecognized option %s\n",
                      pname, argv[i]);
        exit(1);
      }
          else {
        (void)fprintf(stderr, "%s: unrecognized argument %s\n",
                      pname, argv[i]);
        exit(1);
          }
  }
  if (record == NULL || ann_name == NULL) {
          help();
          exit(1);
  }
/* ************************************************************************ */
sprintf(outfil, "%s.%s", record, pname);
sprintf(poutfil, "%s.%s.ev",record,pname);
/* ************************************************************************ */
if((fp=fopen(outfil,"w")) == NULL) {
  printf("%s: can't open output file %s\n",pname,outfil);
  exit(1);
}

/* open signal and annotation files */
a[0].name = ann_name; a[0].stat = READ;
if ((nsig = dbinit(record, a, 1, si, MAXSIG)) < 1)  {
  printf("error in dbinit. Code: %d\n", nsig);
```

```
  exit(2);
 }
/*
/* ********************************************************************** */
 F = sampfreq(record);
 /*  printf("F = %f\n",F); */
 F = 124.95;
 /* find appropriate ABP signal */
 for(i=0;i<nsig;i++) {
  if(strcmp("ABP ",si[i].desc)==0  || strcmp("ABP",si[i].desc)==0 || strcmp("ART ",si[i].desc)==0  ||
strcmp("ART",si[i].desc)==0) BPn=i;
 }
 if(BPn<0) {
  fprintf(stderr,"Error in locating ABP.\n"); exit(1);
 }
/* set baseline and gain values */
 bBP = si[BPn].baseline;
 gBP = si[BPn].gain;
/* ********************************************************************** */
 /* Go to the first segment that contains data of interest. */
/* ********************************************************************** */
 k=0; /* k equals number of annotations obtained so far */
 /*  while(getann(0,&annot)==0 && (double)annot.time/(F*3600.)<tstart) {
 } */
 if(getann(0,&annot) ==0) prev_time = annot.time;
 lastsec = -1;
 flag = 0;
 no_template = 1;
 happy = 0;
 numtemplates = 0;
 while (no_template && getann(0, &annot) == 0 && (double)prev_time/(F*3600.0) < tstop) {
  if(isqrs(annot.anntyp)==1) {
   if ( (curr_time=annot.time)-prev_time<MAXTIME) {
        k++;
    isigsettime(prev_time);
    /*  min/max assumed to be within (prev_time,curr_time) window */
    delta_time = curr_time - prev_time;
    for(i=0;i < delta_time && getvec(vin) > 0;i++) {
        sig[BP][i] = (double)(vin[BPn]-bBP)/gBP;
    }

    if(flag==0) {
        for (i = 0 ; i < delta_time ; i++) {
          template[i] = sig[BP][i];
        }
        templen = delta_time;
        generate_templates (template, delta_time, temp64, temp128, temp256);
        printf("Template set.\n");
        flag = 1;
        numtemplates = 1;
    }

       if (numtemplates == 1) {
        corr [k] = smear_corr (template, &sig[BP][0], delta_time, temp64,
```

```
                                    temp128, temp256, stretch256, &data);
          printf (" checking 1 template -- corr %6.4f dt %d \n",corr[k],
                    delta_time);
         }

         else {
          c1 = smear_corr (template, &sig[BP][0],delta_time, temp64,
                            temp128, temp256, stretch256, &data);
          c2 = smear_corr (template2, &sig[BP][0],delta_time, temp64_2,
                            temp128_2, temp256_2, stretch256_2, &data_2);
          printf (" checking 2 templates -- corr %6.4f & %6.4f dt %d \n",
                    c1, c2, delta_time);
          corr[k] = max(c1,c2);
         }

         if(corr[k] > R_THRESH)  {
          happy++;

          if (happy >= BEATS*numtemplates) {
           no_template = 0;
          }
         }

         else {
          /* copy current template into template2, and copy current
             beat into template */
          if (numtemplates == 1 && BALLOON && data.fh_corr > data.lh_corr &&
              data.fh_corr > 0.7*R_THRESH) {
           numtemplates = 2;
           for (i = 0 ; i < delta_time ; i++)
            template2[i] = sig[BP][i];
           templen2 = delta_time;
           generate_templates (template2, templen2, temp64_2, temp128_2,
                                    temp256_2);
           printf("Template set.\n");
           happy = 0;
          }
          else {
           happy = 0;
           flag = 0;
          }
         }
         prev_time = curr_time;
    }
   }
}

if (no_template == 0) {
  for(i=0; i < SEGLEN ; i++)
    line[i] = (double) i+1.0;


while(getann(0,&annot) == 0 && (double)prev_time/(F*3600.0) < tstop) {
  if(isqrs(annot.anntyp)==1) {
```

```
if ( (curr_time=annot.time)-prev_time<MAXTIME) {
     k++;
 isigsettime(prev_time);

 /*  waveform assumed to be within (prev_time,curr_time) window */

 delta_time = curr_time - prev_time;
 for(i=0;i < delta_time && getvec(vin) > 0;i++) {
      sig[BP][i] = (double)(vin[BPn]-bBP)/gBP;
 }
     if (numtemplates == 1) {
      /* check template length versus current --> gives est. on ECG noiz */
      if ((float)abs(delta_time - templen)/(float)templen > 0.4)
       ecgbad = 1;
      else ecgbad = 0;

      corr[k] = smear_corr (template, &sig[BP][0], delta_time, temp64,
                            temp128, temp256, stretch256, &data);
      tcorr[k] = curr_time/(3600.*F);
      /* default type is 0 = NORMAL */
      type = 0;
      if ((1-corr[k]) > 0.3) {
       /* if low correlation, then is evil = 1 */
       evil = 1;
       if(data.fh_corr > 2*data.lh_corr && data.fh_corr > 0.7*R_THRESH) {
        evil = 0; /* case where first half is good!! */
        type = 1; /* must be a new type = 1 */
       }
      }
      else
       evil = 0;

      /* all bets are off if ecg is bad */
      if (ecgbad) { evil = 0; type = -1; }

      fprintf (fp, "%11.5f %6.3f %6.3f %6.3f %d %2d %d %ld\n", tcorr[k],
                corr[k], data.fh_corr, data.lh_corr, evil, type,
                ecgbad, delta_time);
     }
     else {
      c1 = smear_corr (template, &sig[BP][0], delta_time, temp64,
                            temp128, temp256, stretch256, &data);
      c2 = smear_corr (template2, &sig[BP][0], delta_time, temp64_2,
                            temp128_2, temp256_2, stretch256_2, &data_2);
      tcorr[k] = curr_time/(3600.*F);

      if (c1 > c2) {
       corr [k] = c1;
       /* check templ length versus current --> gives est. on ECG noiz */
       if ((float)abs(delta_time - templen)/(float)templen > 0.4)
        ecgbad = 1;
       else ecgbad = 0;

       if (data.lh_corr > data_2.lh_corr &&
```

```
                fabs(data_2.fh_corr - data.fh_corr) < 0.2) type = 1;
            else type = -1;

            if ((1-corr[k]) > 0.3) evil = 1;
            else evil = 0;

            /* all bets are off if ecg is bad */
            if (ecgbad) { evil = 0; type = -1; }

            fprintf (fp, "%11.5f %6.3f %6.3f %6.3f %d %2d %d %ld\n", tcorr[k],
                    corr[k], data.fh_corr, data.lh_corr, evil, type,
                    ecgbad, delta_time);
          }
          else {
            corr[k] = c2;
            /* check templ length versus current --> gives est. on ECG noiz */
            if ((float)abs(delta_time - templen2)/(float)templen2 > 0.4)
              ecgbad = 1;
            else ecgbad = 0;

            if (data_2.lh_corr > data.lh_corr &&
                  fabs(data.fh_corr - data_2.fh_corr) < 0.2) type = 2;
            else type = -1;

            if ((1-corr[k]) > 0.3) evil = 1;
            else evil = 0;

            /* all bets are off if ecg is bad */
            if (ecgbad) { evil = 0; type = -1; }

            fprintf (fp, "%11.5f %6.3f %6.3f %6.3f %d %2d %d %ld\n", tcorr[k],
                    corr[k], data_2.fh_corr, data_2.lh_corr, evil, type,
                    ecgbad, delta_time);
          }
        }
    }

    prev_time=curr_time;
   } /* end if(isqrs) */
  } /* end while */

  } /* if no_template */
  fclose(fp);
  dbquit();
}
/* ********************************************************************** */

/* ********************************************************************** */
double calcHR(RRtimes, N, f)
long RRtimes[];
int N;
double f;
{
  int i;
```

```
 double diff;

 diff = (double)(RRtimes[0]-RRtimes[N-1])/(f*60.0); /*minutes per N beats*/
 return (double)(N-1)/diff; /* convert from sample units to seconds */
}
/* ********************************************************************** */
/* ********************************************************************** */
double mean(vec,N)
double vec[];
int N;
{
 int i;
 double sum=0.0;

 for (i=0;i<N;i++) sum += vec[i];
 return (double) sum/N;
}
/* ********************************************************************** */
double calcpar(parvec, RRtimes, N, f, deltat)
double parvec[];
long RRtimes[];
int N;
double f;
double deltat;
{
 int i;
 double diff;
 i=1;
 /*  while(i<=N-1) { */


 diff = (double)(RRtimes[0]-RRtimes[N-1])/(f*60.0); /*minutes per N beats*/
 return (double)(N-1)/diff; /* convert from sample units to seconds */
}
/* ********************************************************************** */
void initvect(RRtimes, newtime, N)
long RRtimes[];
long newtime;
int N;
{
  int i;

  for (i=0;i<N;i++)
    RRtimes[i]=newtime;
}
/* ********************************************************************** */
double crosscor(w,v,N)
double w[];
double v[];
int N;
{
 int i;
 double prod_wv=0.0;
 double mag_w=0.0, mag_v=0.0, mean_w=0.0, mean_v=0.0;
```

```
 /*  mean_w = mean(w,N);
 mean_v = mean(v,N); */
 for(i=0;i<N;i++) {

  mean_w += w[i];
  mean_v += v[i];
 }

 if (N == 0) {
  printf ("ARRGHH! \n");
  getchar ();
 }

 mean_w /= N;
 mean_v /= N;


 for (i=0;i<N;i++) {

  prod_wv += (w[i] - mean_w)*(v[i]-mean_v);


  mag_w += pow(w[i]-mean_w,2);

  mag_v += pow(v[i]-mean_v,2);

 }

 if ((mag_w < 0) | (mag_v < 0)) {
  printf ("LowDown, Howdown! \n");
  getchar ();
 }

 mag_w = sqrt(mag_w);
 mag_v = sqrt(mag_v);

 if (mag_w*mag_v == 0) {
  return 0.0;
  printf ("Ender! \n");
  getchar ();
 }
 return prod_wv/(mag_w*mag_v);
}
/* ********************************************************************** */
double normdot(w,v,N)
double w[];
double v[];
int N;
{
 int i;
 double prod_wv=0.0;
 double mag_w=0.0, mag_v=0.0;
```

```
  for (i=0;i<N;i++) {
   prod_wv += w[i]*v[i];
   mag_w += w[i]*w[i];
   mag_v += v[i]*v[i];
  }
  mag_w = sqrt(mag_w);
  mag_w = sqrt(mag_v);
  return prod_wv/(mag_w*mag_v);
}
/* ********************************************************************** */
void addtime(RRtimes, newtime, N)
long RRtimes[];
long newtime;
int N;
{
  int i;

  if (N-2 >= 0 && RRtimes[N-2]==0) {
    for (i=0;i<N;i++)
      RRtimes[i]=newtime;
  }
  else {
    for (i=N-1;i>0;i--)
      RRtimes[i]=RRtimes[i-1];
    RRtimes[0] = newtime;
  }
}
/* ********************************************************************** */
void addpar(parvec, newpar, N)
double parvec[];
double newpar;
int N;
{
  int i;

  if (N-2 >= 0 && parvec[N-2]==0) {
    for (i=0;i<N;i++)
      parvec[i]=newpar;
  }
  else {
    for (i=N-1;i>0;i--)
      parvec[i]=parvec[i-1];
    parvec[0] = newpar;
  }
}
/* ********************************************************************** */
char *getparam(ifile, parameter)
FILE *ifile;
char *parameter;
{
   static char buf[80];

   while (fgets(buf, sizeof(buf), ifile))
```

```
        if (strncmp(buf, parameter, len) == 0)
            return (buf);
    return (NULL);
}

char *prog_name(s)
char *s;
{
    char *p = s + strlen(s);

#ifdef MSDOS
    while (p >= s && *p != '\\' && *p != ':') {
        if (*p == '.')
            *p = '\0';          /* strip off extension */
        if ('A' <= *p && *p <= 'Z')
            *p += 'a' - 'A'; /* convert to lower case */
        p--;
    }
#else
    while (p >= s && *p != '/')
        p--;
#endif
    return (p+1);
}

static char *help_strings[] = {
"usage: %s -r RECORD -a QRS_ANNOTATOR [OPTIONS ...]\n",
"where RECORD specifies the input, QRS_ANNOTATOR is output of aristotle\n",
"and OPTIONS may include:",
" -f TIME      start at specified TIME",
" -h           print this usage summary",
" -t TIME      stop at specified TIME",
" -Vs (or -V)  print elapsed time in seconds before each output sample value",
" -Vm          print elapsed time in minutes before each output sample value",
" -Vh          print elapsed time in hours before each output sample value",
NULL
};

void help()
{
    int i;

    (void)fprintf(stderr, help_strings[0], pname);
    for (i = 1; help_strings[i] != NULL; i++)
        (void)fprintf(stderr, "%s\n", help_strings[i]);
}
/*****************************************************************************/

double find_good_template (double * temp, double *sig, int size, int ltrim,
                           int shift)
/* requires: temp1 at least as longs as sig, size is # of elements in sig,
        2 * ltrim + 1 < size,  all ints positive
  modifies: none
  effects : trims temp and sig by rtrim and ltrim, and shifts them to
```

```
        attain highest correlation, and returns that correlation     */
{
 int i;
 double max_corr, tmp_corr;
 double * sigplace, * tempplace;
 int rtrim, debug, best_shift;

 debug = 1;
 if (size % 2 == 1)                    /* even number of points left */
  rtrim = ltrim;
   else rtrim = ltrim + 1;

 for (i = 0; i < ltrim; i++) {
  /* *(temp1 + i) = 0.0; */              /* trim signal as well? */
  *(sig + i) = 0.0;
 }
 for (i = size-1; i > size - 1 - ltrim; i--) {
  /* *(temp1 + i) = 0.0; */              /* trim signal as well? */
  *(sig + i) = 0.0;
 }
 tempplace = temp;
 sigplace = sig + shift;

 max_corr = 0.0;
 best_shift = -shift;
 for (i = -shift; i < shift; i++) {
  tmp_corr = crosscor (tempplace, sigplace, size - rtrim - ltrim);

  if (debug)
   printf (" %4.4f temp shift %d \n", tmp_corr, i);
  if (tmp_corr > max_corr) {
   max_corr = tmp_corr;
   best_shift = shift;
  }
  tempplace++;
  /*
  if (i < 0)
   sigplace--;
        else
         tempplace++;
  */
 }
 printf ("best shift %d \n", best_shift);
 return ( max_corr );
}


double find_trio (struct trio *data, double * temp, double *sig, int size,
                  int ltrim, int shift)
/* requires: temp1 at least as longs as sig, size is # of elements in sig,
        2 * ltrim + 1 < size,  all ints positive
  modifies: none
  effects : trims temp and sig by rtrim and ltrim, and shifts them to
        attain highest correlation, and returns that correlation     */
```

```
{
 int i;
 double max_corr, tmp_corr;
 double * sigplace, * tempplace;
 int rtrim, debug, best_shift, mid;
 double holder;

 debug = 0;
 if (size % 2 == 1)                    /* even number of points left */
  rtrim = ltrim;
   else rtrim = ltrim + 1;
 mid = (int) size / 2;

 for (i = 0; i < ltrim; i++) {

  *(sig + i) = 0.0;
 }
 for (i = size-1; i > size - 1 - ltrim; i--) {

  *(sig + i) = 0.0;
 }
 tempplace = temp;
 sigplace = sig + shift;

 max_corr = 0.0;
 best_shift = -shift;
 for (i = -shift; i <= shift; i++) {
  tmp_corr = crosscor (tempplace, sigplace, size - rtrim - ltrim);


  if (tmp_corr*tmp_corr > max_corr*max_corr) {
   max_corr = tmp_corr;
   best_shift = shift;
  }
  tempplace++;

 }
 data->max_corr = max_corr;
 holder = max_corr;
 data->mc_shift = best_shift;

 tempplace = temp;
 sigplace = sig + shift;

 max_corr = 0.0;
 best_shift = -shift;
 for (i = -shift; i <= shift; i++) {
  tmp_corr = crosscor (tempplace, sigplace, mid - ltrim);
  if (tmp_corr > max_corr) {
   max_corr = tmp_corr;
   best_shift = shift;
  }
  tempplace++;
```

```
  }
  data->fh_corr = max_corr;
  data->fh_shift = best_shift;

  tempplace = temp + mid;
  sigplace = sig + mid + shift;

  max_corr = 0.0;
  best_shift = -shift;
  for (i = -shift; i <= shift; i++) {
    tmp_corr = crosscor (tempplace, sigplace, mid - rtrim);


    if (tmp_corr*tmp_corr > max_corr*max_corr) {
      max_corr = tmp_corr;
      best_shift = shift;
    }
    tempplace++;

  }
  data->lh_corr = max_corr;
  data->lh_shift = best_shift;

  return (holder);
}

void four1(float data [], unsigned long nn, int isign)
{
  unsigned long n,mmax,m,j,istep,i;
  double wtemp,wr,wpr,wpi,wi,theta;
  float tempr, tempi;

  n=nn << 1;
  j=1;
  for (i=1; i<n; i+=2) {
    if (j > i) {
      SWAP (data[j], data[i]);
      SWAP (data[j+1], data[i+1]);
    }
    m=n >> 1;
    while (m >= 2 && j > m) {
      j -= m;
      m >>= 1;
    }
    j += m;
  }
  mmax = 2;
  while (n > mmax) {
    istep = mmax << 1;
    theta = isign * (6.28318530717959/mmax);
    wtemp = sin (0.5 * theta);
    wpr = -2.0*wtemp*wtemp;
    wpi = sin (theta);
    wr = 1.0;
```

```
    wi = 0.0;
    for (m=1;m<mmax;m+=2) {
      for (i=m;i<=n;i+=istep) {
            j=i+mmax;
            tempr=wr*data[j]-wi*data[j+1];
            tempi=wr*data[j+1]+wi*data[j];
            data[j]=data[i]-tempr;
            data[j+1]=data[i+1]-tempi;
            data[i] += tempr;
            data[i+1] += tempi;
      }
      wr = (wtemp=wr)*wpr-wi*wpi+wr;
      wi = wi*wpr+wtemp*wpi+wi;
    }
    mmax = istep;
  }
}



void realft (float data[], unsigned long n, int isign)
{
  void four1 (float data[], unsigned long nn, int isign);
  unsigned long i, i1, i2, i3, i4, np3;
  float c1 = 0.5, c2, h1r, h1i, h2r, h2i;
  double wr, wi, wpr, wpi, wtemp, theta;

  theta = 3.141592653589793/ (double) (n>>1);
  if (isign == 1) {
    c2 = -0.5;
    four1(data,n>>1,1);
  } else {
    c2 = 0.5;
    theta = -theta;
  }
  wtemp = sin(0.5*theta);
  wpr = -2.0*wtemp*wtemp;
  wpi=sin(theta);
  wr=1.0+wpr;
  wi=wpi;
  np3=n+3;
  for (i=2;i<=(n>>2);i++) {
    i4 = 1+(i3=np3-(i2=1+(i1=i+i-1)));
    h1r=c1*(data[i1]+data[i3]);
    h1i=c1*(data[i2]-data[i4]);
    h2r= -c2*(data[i2]+data[i4]);
    h2i= c2*(data[i1]-data[i3]);
    data[i1]=h1r+wr*h2r-wi*h2i;
    data[i2]=h1i+wr*h2i+wi*h2r;
    data[i3]=h1r-wr*h2r+wi*h2i;
    data[i4]=-h1i+wr*h2i+wi*h2r;
    wr=(wtemp=wr)*wpr-wi*wpi+wr;
    wi=wi*wpr+wtemp*wpi+wi;
  }
```

```
  if (isign == 1) {


    data[1] = (h1r = data[1]) + data[2];
    data[2] = h1r - data[2];
  } else {
    data [1] = c1 * ((h1r=data[1]) + data[2]);
    data [2] = c1 * (h1r-data[2]);
    four1(data,n>>1,-1);
  }
}

void generate_templates (double * template, int size, double temp64 [],
                         double temp128[], double temp256[])

{
  linear_interpolate (template, size, temp64, 64);
  linear_interpolate (template, size, temp128, 128);
  linear_interpolate (template, size, temp256, 256);
}

void get_temps (double * temp, int size, double temp64 [],
                double temp128 [], double temp256 [], float temp64f [],
                float *temp128f, float temp256f [])
{
  float *peek64;
  float *peek128;
  float *peek256;

  double *poke;
  float scale64, scale128, scale256;
  int i;

  peek64 = temp64f;
  peek128 = temp128f;
  peek256 = temp256f;
  poke = temp;

  for (i = 0; i < 64; i++) {
    if (i < size) {
      *peek64 =  (float) *poke;
      *peek128 = (float) *poke;
      *peek256 = (float) *poke;
      poke++;

    } else {
      *peek64 = 0.0;
      *peek128 = 0.0;
      *peek256 = 0.0;
    }
    peek64++;
    peek128++;
    peek256++;
  }
```

```
  for (i = 64; i < 128; i++) {
    if (i < size) {
      *peek128 = (float) *poke;
      *peek256 = (float) *poke;
      poke++;
    } else {
      *peek128 = 0.0;
      *peek256 = 0.0;
    }
    peek128++;
    peek256++;
  }

  for (i = 128; i < 256; i++) {
    if (i < size) {
      *peek256 = (float) *poke;
      poke++;
    } else {
      *peek256 = 0.0;
    }
    peek256++;
  }
  for (i = 1; i < 129; i++) printf ("%11.4f \n",temp128f[i]);
  realft (temp64f, 64, 1);
  realft (temp64f, 64, -1);
  realft (temp128f, 128, 1);
  for (i =1; i <= 128; i++) printf ("fft %11.3f \n",temp128f[i]);

  realft (temp128f, 128, -1);
  realft (temp256f, 256, 1);
  realft (temp256f, 256, -1);

  scale64 =  (double) 2/64;
  scale128 = (double) 2/128;
  scale256 = (double) 2/256;

  for (i = 1; i <= 256; i++) {
    if (i <= 64)
      temp64[i] = (double) temp64f[i] * scale64;
    if (i <= 128)
      temp128[i] = (double) temp128f[i] * scale128;
    temp256[i] = (double) temp256f[i] * scale256;

  }
}

int linear_interpolate (double * signal, int size, double *result, int target)
{
  double step, dist, fract;

  int mark, i, low;

  step = (double) size / target;
```

```
  dist = step;
  /* we know the first and last points */
  result [0] = signal [0];
  result [target-1] = signal [size-1];
  for (i = 1; i < target; i++) {

    low = (int) dist;

    fract = (double) (dist - low);

    result [i] = (1.0 - fract) * signal [low] + fract * signal [low + 1];

    dist += step;

  }

}

int smear_signal (double * signal, int size, float * resultf,
                  double * result) {
  int newsize;
  double scale;
  double scale_time;
  int i,k;

  if (size <= 64)
    newsize = 64;
  else
    if (size <= 128)
      newsize = 128;
    else
      if (size <= 256)
            newsize = 256;
            else return (0);

  scale = (double) 2/newsize;

 /*  for (i = 1; i <= newsize; i++) {
   if (i <= size)
      resultf[i] = (float) signal[i];
   else
     resultf[i] = (float) 0.0;
  } */

  for (i = 1; i <= newsize; i++) {
    resultf[i] = (float) 0.0;
  }

scale_time=(double)newsize/(double)size;
  for (i = 1; i <= size; i++) {
     k = (int) ceil(scale_time * (double) i);
     /*  resultf[k] = (float) signal[i]; */
resultf[k]=(float)k;
  }
```

```
 realft (resultf, newsize, 1);

 for (i = 1; i <= newsize; i++) {
  if (i > size)
   resultf[i] = (float) 0.0;
  }

  realft (resultf, newsize, -1);
  for (i = 1; i <=newsize; i++) {
   result[i] = (double) (resultf[i] * scale);

  }

  return newsize;
}

double smear_corr (double * temp, double * signal, int size,
                   double *temp64, double *temp128, double * temp256,
                   double * stretch256, struct trio *data)
{
  static float floater [256];
  int debug = 1;
  int newsize;

  if (size <= 64) {
   linear_interpolate (signal, size, stretch256, 64);
   return find_trio (data, temp64, stretch256, 64, 3, 3);
  }

  if (size <= 128) {
   linear_interpolate (signal, size, stretch256, 128);
   return find_trio (data, temp128, stretch256, 128, 3, 3);
  }

  if (size <= 256) {
   linear_interpolate (signal, size, stretch256, 256);
   return find_trio (data, temp256, stretch256, 256, 3, 3);
  } else {
   printf (" Signal too big, must less than 256 not %d \n", size);
   return 0.0;
  }
}
/* END OF artdet.c */
```

The following UNIX C-shell is used to run the program "artdet".

```
#!/bin/csh
# runart uses artdet to read in a file which has the times of alarms

foreach b (055 212 215 216 230 231 240 241 242 245 248 415 417 420 421)
```

```
 cat nh${b} nl${b} > ${b}.tmp
 sort -n ${b}.tmp >! $b
 echo $b
 /bin/rm ${b}.tmp
 @ i=0
 @ length = `cat $b | wc -l`
 echo $i $length
 while ($i < $length)
  @ i++
  sqrs -r $b -f `cat $b | head -${i} | tail -1 | awk '{print $1}' `:0:0 -t `cat $b | head -${i} | tail -1 | awk
'{print $2}' `:0:0
  # output is in $qrs.${b}
  artdet -r ${b} -a qrs -f `cat $b | head -${i} | tail -1 | awk '{print $1}' `:0:0 -t `cat $b | head -${i} | tail -1 |
awk '{print $2}' `:0:0
  cat ${b}.artdet >>! data/art${b}.${i}
  /bin/rm ${b}.artdet qrs.${b}
  # output is ${b}.artdet, containing times at which low correlations with
  # ideal template occur, along with the correlation value
 end
 echo 'sleeping ... change CD'
 sleep 100
end
```

The following Matlab programs were used to implement the linear trend algorithm and the artifact removal algorithms, from both the "INOP" data and the waveform analysis data. The M-files for the algorithms follow.

```
function [alclass] = classin(dat,alarm,inop)

%This function classifies the ABP data, removing the "INOP" data
%The form of the function is ==> [alclass] = classin(dat,alarm,inop)

dim = size(alarm);
n=dim(1);
p=dim(2);
T = 1.024; % in seconds
HOUR = 3600.0;
% Use delta as the time over which the slopes will be obtained
% sampling period is 1.024 seconds

alclass = ones(n,1); % set all alarms to real = 1 initially

for i=1:n
  SGN = sign(alarm(i,2)-alarm(i,3));
  te = alarm(i,1) - 10.0/3600.0; % remove 10 seconds
  ts5 = te - 5.0/60.0;
  ts1 = te - 1.0/60.0;
  fivewin = getmat(dat,ts5,te,3600);
  dim=size(fivewin);
  nt5 = dim(1);
  onewin = getmat(dat,ts1,te,3600);
  dim = size(onewin);
  nt1 = dim(1);
```

```
  plot(fivewin(nt5,1),fivewin(nt5,2),'gx');
  %%%%%%%%%%%%%%%%%%%
  % NOW PUT IN ALGORITHM WHICH TAKES OUT DATA CORRUPTED BY NOISE
  %%%%%%%%%%%%%%%%%%%
  tmpwin = fivewin;
  tmpinop = getmatx(inop,ts5,te,1);
  [fivewin,tstart,tstop]=rminopx(tmpwin,tmpinop,1);
  %%%%%%%%%%%%%%%%%%%
  tmpwin = onewin;
  tmpinop = getmatx(inop,ts1,te,1);
  [onewin,tstart,tstop]=rminopx(tmpwin,tmpinop,1);
  %%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%%
  % NOW PUT IN trimean PROGRAM TO REMOVE ADDITIONAL ARTIFACT
  % CHECK TO SEE THAT NUMBER OF POINTS IS SUFFICIENT TO PROJECT
  [p5,S5] = polyfit(fivewin(:,1),fivewin(:,2),1);
  [y5,delta5]=polyval(p5,alarm(i,1),S5);
  [p1,S1] = polyfit(onewin(:,1),onewin(:,2),1);
  [y1,delta1]=polyval(p1,alarm(i,1),S1);
  if alarm(i,2) < 0 | alarm(i,2)/alarm(i,3) > 1.5
    alclass(i) = 0.0;
  elseif SGN*(alarm(i,2) - y5) > delta5*2.0 % 5 min. suggest art?
    if SGN*(alarm(i,2) - y1) > delta1*2.0 % 1 min. suggest art?
      alclass(i) = 0.0; % that is, alarm is artifact, not real
    end
  end
end
```

The following Matlab M-file is used to categorize all changes which were visually classified as "real" changes.

```
function [physcat] = classcat(dat,alarm,inop)

%This function outputs physcat; each row shows how much data has changed
vs. previous
% 'WINTIME' hours (here WINTIME = 0.1 hours = 6 minutes
%The form of the function is ==> [physcat] = classcat(dat,alarm,inop)

WINTIME = 0.1; % set comparison window to six minutes
dim = size(alarm);
n=dim(1);
p=dim(2);
T = 1.024; % in seconds
HOUR = 3600.0;
% Use delta as the time over which the slopes will be obtained
% sampling period is 1.024 seconds

alclass = ones(n,1); % set all alarms to real = 1 initially
physcat = zeros(n,4); % time, abpdirect, hrdirect, papdirect

for i=1:n
  SGN = sign(alarm(i,2)-alarm(i,3));
  te = alarm(i,1) - 10.0/3600.0; % remove 10 seconds
  tsWINT = max(0,te - WINTIME);
  ts5 = te - 5.0/60.0;
  ts1 = te - 1.0/60.0;
  WINTwin = getmat(dat,tsWINT,te,3600);
```

```
  dim=size(WINTwin);
  ntWINT = dim(1);
  fivewin = getmat(dat,ts5,te,3600);
  dim=size(fivewin);
  nt5 = dim(1);
  onewin = getmat(dat,ts1,te,3600);
  dim = size(onewin);
  nt1 = dim(1);
  plot(fivewin(nt5,1),fivewin(nt5,2),'gx');
  %%%%%%%%%%%%%%%%%%%%
  % NOW PUT IN ALGORITHM WHICH TAKES OUT DATA CORRUPTED BY NOISE
  %%%%%%%%%%%%%%%%%%%%
  tmpwin = WINTwin;
  tmpinop = getmatx(inop,tsWINT,te,1);
  [WINTwin,tstart,tstop]=rminopx(tmpwin,tmpinop,1);
  %%%%%%%%%%%%%%%%%%%%
  tmpwin = fivewin;
  tmpinop = getmatx(inop,ts5,te,1);
  [fivewin,tstart,tstop]=rminopx(tmpwin,tmpinop,1);
  %%%%%%%%%%%%%%%%%%%%
  tmpwin = onewin;
  tmpinop = getmatx(inop,ts1,te,1);
  [onewin,tstart,tstop]=rminopx(tmpwin,tmpinop,1);
  %%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%%%
  % NOW PUT IN trimean PROGRAM TO REMOVE ADDITIONAL ARTIFACT
  % CHECK TO SEE THAT NUMBER OF POINTS IS SUFFICIENT TO PROJECT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  [p5,S5] = polyfit(fivewin(:,1),fivewin(:,2),1);
  [y5,delta5]=polyval(p5,alarm(i,1),S5);
  [p1,S1] = polyfit(onewin(:,1),onewin(:,2),1);
  [y1,delta1]=polyval(p1,alarm(i,1),S1);
  if alarm(i,2) < 0 | alarm(i,2)/alarm(i,3) > 1.5
    alclass(i) = 0.0;
  elseif SGN*(alarm(i,2) - y5) > delta5*2.0 % 5 min. predict suggest
art?
    if SGN*(alarm(i,2) - y1) > delta1*2.0 % 1 min. predict also suggest
art?
      alclass(i) = 0.0; % that is, alarm is artifact, not real
    end
  end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  if alarm(i,4) == 1.0
    [WINTave,WINTstd]=trimean(WINTwin,0.1);
    ALwin = getmat(dat,alarm(i,1)-5/3600,alarm(i,1)+5/3600,3600);
    ALave=mean(ALwin);
    for j = 2:4
      physcat(i,1) = alarm(i,1);
      if WINTstd(j) > 0
        physcat(i,j) = (ALave(j) - WINTave(j))/WINTstd(j);
      else
        physcat(i,j) = ALave(j) - WINTave(j);
      end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  end
end
```

Appendix B

The following chart summarizes the change in the physiologic parameters during the time of the alarm compared to the data in the record up to that time. The first column indicates the record, the second column the time of the alarm, and the third through fifth columns are the change in the measurements ABP, HR, and PAP. These data are used to categorize the type of change that has occurred for each alarm that was classified as a "real" alarm.

Record	Time (hrs)	dABP	dHR	dPAP
lo055	0.84	-	-	-
	0.892	-	-	-
	8.924	-	-	+
	8.978	-	-	+
hi212	9.347	-	0	0
lo212	9.45	-	0	+
	9.502	-	0	0
	9.592	-	0	+
	9.601	-	0	+
	19.46	-	0	+
	21.18	-	-	0
	21.21	-	-	0
	23.6	-	0	0
	31.59	-	-	-
	32.65	-	0	0
	33.22	-	-	-
	33.35	-	+	+
	33.4	-	0	0
	33.46	-	-	+
	33.59	-	-	0
	35.27	-	-	0
	36.04	-	-	0
	36.05	-	-	0
	36.22	-	-	-
	36.23	-	-	-
	36.28	-	-	-
	36.33			
	36.39			
	37.25			
	37.53			
	38.11			
	38.56			
hi215	6.425	+	+	+
	6.458	+	+	+

	6.487	+	+	+
	6.525	+	+	+
	6.532	+	+	+
	6.536	+	+	+
	6.546	+	+	+
	6.556	+	+	+
	10.73	+	+	+
	10.74	+	+	+
	10.76	+	+	+
	10.79	+	+	+
	10.8	+	+	+
	11.69	+	+	+
	11.74	+	+	+
	11.76	+	+	+
	11.79	+	+	+
	11.8	+	+	+
	11.82	+	+	+
	11.83	+	+	+
	11.87	+	+	+
	11.87	+	+	+
	11.93	+	+	+
	11.94	+	+	+
	12.41	+	+	+
	12.68	+	+	+
	12.69	+	+	+
lo216	0.7509	-	0	
	0.7654	-	0	
	2.755	-	0	
	4.758	-	0	
	5.783	-	-	
	5.835	-	-	
	5.846	-	-	
	9.347	-	0	
	11.17	-	-	
	11.17	-	-	
	12.77	-	0	
	13.61	-	-	
	14.78	-	0	
	15.39	-	-	
	15.4	-	-	
	16.38	-	0	
	16.78	-	0	
	18.02	-	0	
	18.24	-	0	
	18.78	-	-	
	18.78	-	-	
	18.84	-	-	
	18.89	-	-	
	20.31	-	-	

	20.36	-	-	
	21.68	-	-	
	21.74	-	-	
	21.79	-	-	
	22.69	-	-	
	22.74	-	-	
	22.77	-	-	
	22.83	-	-	
	22.89	-	-	
hi230	1.258	-	-	0
lo230	1.313	-	-	0
	1.336	-	-	0
	1.347	-	-	0
	1.373	-	-	0
	1.51	-	-	0
	1.526	-	-	0
	1.541	-	-	0
	1.774	-	-	0
	1.797	-	-	0
	2.224	-	+	0
	7.601	+	0	0
	7.655	+	0	0
	7.721	+	0	0
	11.41	-	-	-
	11.42	-	-	-
	17.61	-	-	-
	17.62	-	-	-
	17.63	-	-	-
lo231	6.5	-	-	-
	11.35	-	-	+
lo240	2.611	-	0	0
	3.071	-	-	0
	3.123	-	-	0
	3.178	-	-	0
	3.201	-	-	0
	3.272	-	-	0
	3.333	-	-	0
	3.411	-	-	0
	3.46	-	-	0
	3.47	-	-	0
	3.478	-	-	0
	3.505	-	-	0
	3.517	-	-	0
	3.538	-	-	0
	3.551	-	-	0
	3.6	-	-	0
	3.858	-	-	0
	3.911	-	0	0
	3.976	-	0	0

	4.037	-	0	0
	4.221	-	-	0
	4.261	-	-	0
	4.305	-	-	0
	4.479	-	-	+
	4.664	-	0	0
	4.679	-	0	0
	4.699	-	0	0
	4.733	-	0	0
	4.806	-	0	0
	4.85	-	0	0
	4.932	-	0	0
hi241	13.13	-	-	-
lo241	13.2	-	-	-
	13.23	-	-	-
	13.25	-	-	-
	13.42	-	-	-
hi242	no	real	episodes	
lo242				
lo245	no	real	episodes	
hi248	21.82	+	+	+
lo248	22.07	+	+	+
	22.08	+	+	+
	23.05			
hi415	1.55	+	-	+
lo415	1.596	+	-	+
	5.932	+	+	-
	11.27	-	0	0
	13.72	+	0	0
	17.97			
	17.97			
	19.97	+	-	0
hi417	10.67	-	0	-
lo417	10.67	-	0	-
lo418	1.927	-	+	0

Appendix C

Appendix C contains the two-dimensional measurement plots for each record as a whole (e.g., ABP vs PAP, and ABP vs HR), in addition to each ABP alarm episode on a single plot at much higher resolution. These were used for the physiologic classification of section 4.2. Dr. Roger Mark (MIT, Building E-25 Room 525) has this appendix.