Temporal Characterization of Patient State with Applications to Prediction of Tachycardia in Anesthesia via Induction of Inhaled Desflurane

by

Gil Alterovitz
B.S., Electrical and Computer Engineering (1998)

Carnegie Mellon University

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

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

Signature of Author……
Department of Electrical Engineering and Computer Science
January 19, 2001

Certified by……
David H. Staelin
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by……
James H. Philip
Associate Professor, Harvard Medical School
Associate Professor of Anesthesiology, Harvard/MIT Division of Health Science and Technology
Thesis Supervisor

Accepted by……
Arthur C. Smith
Chairman, Committee on Graduate Students
Department of Electrical Engineering and Computer Science

To curiosity,
and innocence…

ACKNOWLEDGEMENTS

It goes without saying that my MIT advisor, Professor D. H. Staelin, really helped to shape me and how I approach problems. Clearly, I learned much about engineering and mathematical methods. But, I also learned about problem solving skills not just applicable to research, but to life in general. For example, I can recall my first few research meetings with Prof. Staelin. I remember making numerous mental notes of his method of approaching strategizing and approaching problems. Throughout the Fall, I found myself using a few strategies myself in dealing with multiple constraints. He was not just a pragmatic advisor and mentor, but also one that really cared about my course at MIT. I recall my first meeting with Prof. Staelin he was my academic advisor and I was a 'freshman' graduate student. Instead of just hearing what courses I had intended to take for the semester, he started by asking questions and taking notes, thought it all seemed irrelevant at first. Yet, I soon realized that he was taking the time to build a mental map of my past and future direction so that he could give me better advice. Thank you, Prof. Staelin.

Without my co-advisor J. H. Philip, MD, MEE, Director of Bioengineering and head of the Bioengineering Laboratory at Brigham and Women Hospital's Anesthesia Department, this project would never have been conceived. Though MIT regulations did not permit Dr. Philip to be my official supervisor, he was my co-advisor on the Harvard side of the equation. It was through discussions with Dr. Philip that the initial clinical issues were brought to light. He was also a mentor. Through our meetings, I learned about the clinician perspective, how it varied from that of the engineer, and how to work at this interface between medicine and engineering. Thank you, Dr. Philip.

I would like to thank the entire Remote Sensing Laboratory for creating a friendly and inspiring atmosphere. Thank you, Mr. J. Barrett, Bill Blackwell, Fred Chen, Hua Fung, Jay Hancock, R. Vincent Leslie, Dr. P. Rosenkranz, and Herb Viggh. Without them, it would have been hard to get adjusted to the lab and start on the actual project as quickly as I did. Thanks especially to Bill, Jay, and Vincent for their help and suggestions on my research. Thank you Seth for helping me with a number of computer configuration issues related to the group's computer network infrastructure.

It certainly has been quite an adventure to complete my thesis. From a crash that led me to lose and then rediscover my thesis in a cryptic temporary swap file… to power failures where I had to resort to my Palm Vx via portable keyboard to complete my thesis on time, my family and friends were there for me. Thanks especially to Dad, Mom, and my brother Ron (whose fascinating personal tales of adventures to Pittsburgh provided much amusement during this time) for their continuing encouragement and support.

This research has been generously funded by the Department of Defense during my tenure at MIT by a National Defense Science and Engineering Graduate Fellowship. Thanks to the Department of Defense and thank you to the American people for supporting such programs which can literally change the lives of young scientists.

TABLE OF CONTENTS

1. ABSTRACT 5

2. CLINICAL BACKGROUND 6

3. MATHEMATICS / ENGINEERING MODELS 7

3.1. Developing a Model 7
3.2. Principal Component Analysis 12
3.3. Iterated Order and Noise Estimation, and Blind Adjusted Principal Component Analysis 12

4. APPROACH 14

4.1. Strategy 14
4.2. Required Equipment 23

5. DESIGN, IMPLEMENTATION, AND TESTING ISSUES 24

6. RESULTS 33

7. CONCLUSIONS AND DISCUSSION 59

8. REFERENCES 64

9. LIST OF PERTINENT ACRONYMS AND ABBREVIATIONS (CLINICAL & TECHNICAL) 66

10. APPENDIX: SOURCE CODE LISTING 68

10.1. Selected Files Listings 68

**CHARACTERIZATION OF PATIENT STATE
WITH APPLICATIONS TO PREDICTION OF TACHYCARDIA IN ANESTHESIA VIA INDUCTION OF INHALED DESFLURANE**

by

GIL ALTEROVITZ

Submitted to the Department of Electrical Engineering and Computer Science
on January 19, 2001 in partial fulfillment of the
requirements for the Degree of Master of Science in
Electrical Engineering and Computer Science

1. ABSTRACT

It has always been assumed that using clinically measurable parameters is the most *efficient* way to characterize patient state. By adding additional sensors, monitors, and derived statistics (e.g. mean arterial blood pressure from diastolic and systolic), it was hoped that more information could be garnered about patient state.

This thesis challenges the assumption that providing the physician with a full set of clinically measurable parameters is the most efficient way to characterize patient state. The thesis presents a novel way to consider patient state by utilizing reduced dimensionality and by estimating noise. It then explores an application, namely prediction of tachycardia, which often occurs at the onset of induction of inhaled desflurane. One unexpected initial finding was that all 46 patients exhibited tachycardia or hypertension within the first hour of the operation.

Three models for predicting tachycardia episodes are proposed, including one model based on use of Blind Noise Adjusted Principal Component Analysis[1] (using Iterative Order and Noise Estimate (ION)[2] and Principal Component Analysis (PCA)[3]). Without ION, PCA-based methods alone yielded only 2 useful degrees of freedom, with the rest being relegated to noise. The ION PCA-based method allows one to capture with 5 principal components the information contained in 31 fundamental and derived patient variables, while at the same time reducing the effects of noise. Furthermore, the five discovered significant principal components representing patient state were characterized quantitatively and their physiologic correlates are hypothesized qualitatively. Examination of the 31 original patient parameters in the ION PCA model that predicts tachycardia revealed the relative importance of the original patient parameters to the tachycardia problem. The receiver operating characteristic (ROC) curve for the ION PCA-based predictor suggested a 70% detection rate with 3% false alarms when predicting tachycardia two minutes and twenty seconds into the future. While the patient state characterization method was used for tachycardia prediction, it is potentially useful in myriad medical domains involving multivariate analysis.

Thesis Supervisor: Daniel H. Staelin
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: James H. Philip
Title: Associate Professor, Harvard Medical School
Associate Professor of Anesthesiology, Harvard/MIT Division of Health Science and Technology

2. CLINICAL BACKGROUND

A number of anesthetic drugs have been found to affect the Central Nervous System and produce general anesthesia- comprised of sleep, hypnosis, analgesia, muscle relaxation, and loss of reflexes. A drug in this class, desflurane, is commonly used during surgery in the operating room.

One protocol for administrating anesthetics such as desflurane, involves induction via inhalation of the agent as a gas. In the case of desflurane, a specific protocol is described below[4]. First, intravenous induction (i.e. administrating an initial anesthetic intravenously) and tracheal intubation (i.e. placing the breathing circuit in the patient's throat) take place. Next, the desflurane vaporizer (an instrument that transforms desflurane to gaseous phase) is set to 18%. The rate of super-oxygenated air (fresh gas flow) is set to 1 L/min. As the patient breathes in the air with the anesthetic, the concentration of the drug in the inspired and expired air increases. Finally, the vaporizer is set to 9% once inspired and expired concentrations reach 8% and 6% respectively.

Desflurane is expensive to administer in an open circuit in which the patient breathes only fresh gas. However, closed circuit (in which the patient rebreathes his/her own air supplemented with oxygen and new anesthetic) desflurane administration is cost efficient, effective, and fast (due to it's low blood/gas solubility).[5] Yet, desflurane does present a few issues related to tachycardia[6] which occur during the induction of the inhaled desflurane. This initial period can be assumed to be less than 15 minutes in duration.

This thesis seeks to explore a method of characterizing patient state as well as its use in tachycardia prediction (defined as heart rate > 100 beats per minute (BPM)) in advance, so that the clinician can be aware of the pending tachycardia as well as possibly correct it.

3. MATHEMATICS / ENGINEERING MODELS

3.1. Developing a Model

As more sensors are added in an increasingly technology-dependent operating room (OR) setting, doctors such as anesthesiologists must sift through an ever greater number of patient parameters (dozens every few seconds) in addition to carrying out their duties at the OR. Yet, as this thesis proposes, it possible that many of these parameters are correlated and contain redundant information. Looking at all of them at once may not be the optimal manner of assessing patient state or predicting future patient state, or a subset thereof such as tachycardia presence, to determine the course of action. Especially in an operating room setting, where time is critical, the ability to consolidate information into an effective patient model and make quick decisions based on *the most significant relevant data* can directly impact the mortality rate in the OR. Previous studies have examined various variables necessary for monitoring as well as optimization of the monitoring tools employed[7], but this paper seeks to seek a model for examining noise estimation and data set dimension reduction for more efficient future patient state characterization.

Modeling patient state temporally can be approached in many ways. One method is to employ mathematical models to quantify the patient state at time t (i.e. $\underline{\mathbf{s}}(t)$) and use this information to predict patient state at time: $t + \Delta t$. In this case, recorded variables from sensors on the patient could be hypothesized to contain enough information about $\underline{\mathbf{s}}(t)$ to accurately predict a subset of $\underline{\mathbf{s}}(t+\Delta t)$, namely heart rate (HR). Patient state need not necessarily contain simply heart rate or tachycardia information. In fact, one can conceptually think of patient state as a vector $\underline{\mathbf{s}}(\mathbf{t})$ where t is the current time or time sample (in discrete time) and the vector has N (the number of parameters) elements. A parameter here is defined to include such patient recorded variables as heart rate or

level of inspired agent concentration (anesthetic level). It can also include seemingly unrelated variables like expired CO_2 (carbon dioxide) concentration and oxygen saturation in the blood. These other variables represent current patient state as well and may provide addition predictive power. For instance, the sympathetic nervous system can affect blood pressure, heart rate, and respiration rate. Therefore, these variables may yield additional information of the patient state via a principal component. For example, given the above variables, an unobservable derived variable such as sympathetic nervous system state may be uncovered.

Since patient state is available at discrete times (i.e. sensor recorded information are sampled at finite intervals), a representation of the state vector as $\underline{\mathbf{s}}(n)$ will be more suitable here. The n represents the time sample when the patient vector was recorded. This will become an issue later, in the "Approach" section, since different sensors sample at different intervals, effectively making it difficult to establish exact patient state $\underline{\mathbf{s}}(n)$ for a given $n=n_1$.

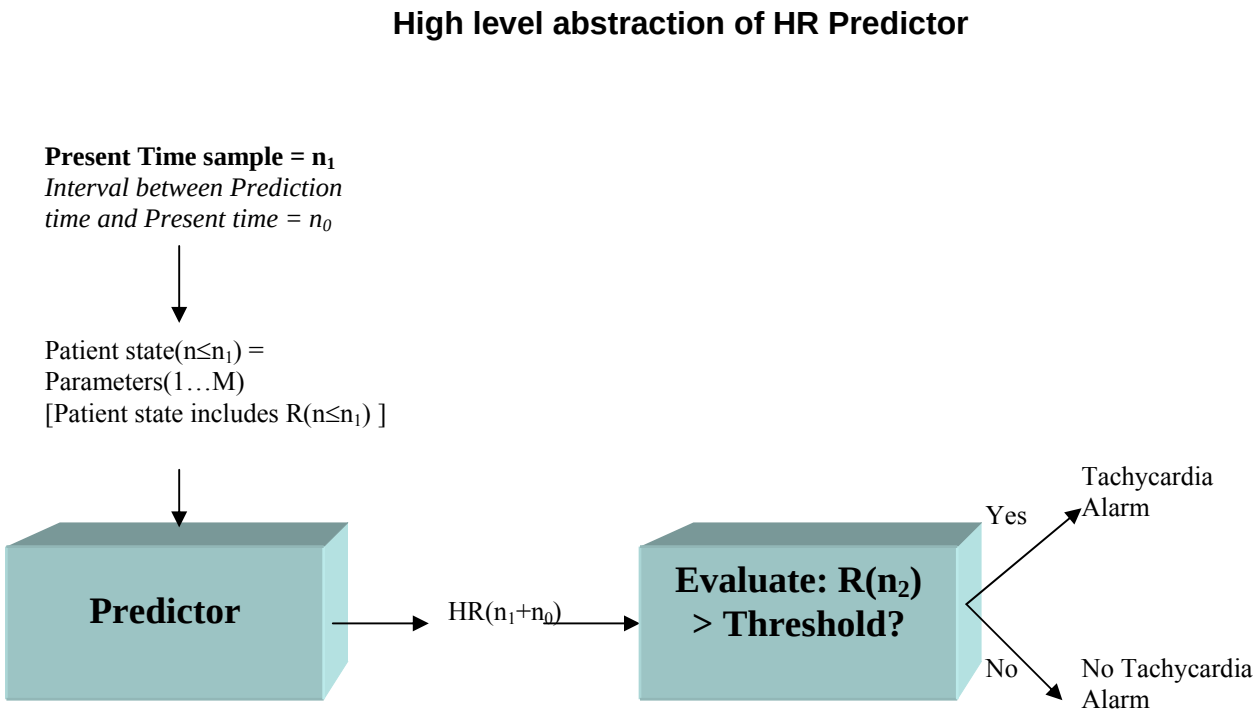
Since tachycardia is a variable derived from heart rate, creating a predictor for HR and then setting a threshold for tachycardia (e.g. 100 BPM) allows for a wider dynamic range (as opposed to the two-value Boolean variable tachycardia). It would allow for quantification of when tachycardia is being approached. For example, while a heart rate of 101 BPM is technically tachycardia, a doctor would take very different action at such a prediction compared to a heart rate of 200 BPM. Thus, in order to add finer granularity, two additional thresholds can be defined. These will be referred to in this thesis as sub-tachycardia (above normal but below 100 BPM, e.g. 80 BPM) and super-tachycardia (above 100 BPM, e.g. 120 BPM). Clearly, if heart rate is approaching 100 BPM, it must first pass through the sub-tachycardia threshold. Put together with other information, such as the derivative of heart rate (a high slope implies the tachycardia stage might be next), these

additional stages provide finer granularity for patient monitoring and an early warning system for actual tachycardia.

One of the goals of this thesis is to present a framework for prediction of tachycardia in advance so that the clinician can take steps (e.g. turn down the vaporizer) in order to 1) reduce the period of impending tachycardia and/or 2) prevent its occurrence. Let us assume that the time delay for the anesthetic to affect the patient when applied at time n_1 is approximately 2 1/2 minutes (see 'Design, Implementation, and Testing Issues' section, Figure 9 for discussion). It can then be hypothesized that if the clinician was aware that the patient would have tachycardia at $n_2 = n_1+n_0$ where $n_0 \geq 2.5$ minutes, then he/she could turn down the vaporizer setting or take other action in order to stem the tachycardia at time n_1. In fact, given that the proposed predictor will provide heart rate information rather than simply tachycardia predictions, it is conceivable that an artificial tachycardia threshold be developed to help the clinician prevent tachycardia. For example, a threshold for sub-tachycardia or super-tachycardia could be developed to give the clinician additional qualitative information as well as advance warning.

What is currently being proposed is, at present, a black box. It takes $\underline{\mathbf{s}}(n \leq n_1)$ as input, where n_1 is the current time sample, to estimate heart rate $R(n>n_1)$ as in Figure 1.

Figure 1



There are a number of models that can be used in constructing this 'black box.' For example, two such predictors include a "static heart rate" and "heart rate average-based" predictors. The former involves setting:

Equation 1

$$R(n_1+n_0) = R(n_1).$$

where n_1 is the present time sample and n_0 is the number of time samples ahead that we are predicting heart rate.

The latter involves setting:

Equation 2

$$R(n_1+n_0) = [\ \sum_{i=[n-N+1,n]} R(i)\ ] / N$$

where N = number of samples (including present sample) to take average over.

n_1 is the present time sample.

n_0 is the number of time samples ahead that we are predicting heart rate.

An analogy for the former method is predicting tomorrow's weather to be the same as today's weather. An analogy for the latter method is assuming tomorrow's weather will be about the same as it has been in the past week. Both of these methods use some information from the patient state in prediction, but not all of the inputs are taken into account. For example, a change in the gas vaporizer setting has little influence on these models. Yet, it is precisely because desflurane is thought to induce tachycardia that a prediction scheme has been proposed. On the other hand, these easily implementable models can serve as a starting point in tachycardia prediction and as a means of comparison with others.

Other models for prediction include linear regression and neural networks. These are discussed further in the 'Approach' section, once the use of principal components and ION is explored further within the context of the tachycardia problem.

3.2. Principal Component Analysis

Regardless of the model employed, the degrees of freedom/parameters in the various models increase as the number elements in the patient state vector increase. Thus, either way, if the number of parameters could somehow be reduced, it would help in producing faster predictions as well as give insight into which parameters yield more information about the dataset. One way to reduce the number of parameters without losing information is to determine the principal components, PC's, via Principal Component Analysis (PCA). Each principal component is a sum of weighted parameters. The weights are contained in the eigenvectors, which provide virtually lossless reconstruction of the original data. Essentially, Principal Component Analysis allows one to create a reduced number of uncorrelated channels from a larger initial set of correlated ones.

The PC's are found as follows. First, the covariance matrix (e.g. $\underline{\mathbf{E}}$) of the data matrix $\underline{\mathbf{L}}$ is calculated. Next, the eigenvalues and eigenvectors of $\underline{\mathbf{E}}$ are found. The eigenvectors are sorted (to form a matrix, $\underline{\mathbf{Q}}$) so that they are in descending order based on the eigenvalues. Next, the first n eigenvectors in $\underline{\mathbf{Q}}$ (with largest eigenvalues) are selected based on a scree plot[3] to form matrix $\underline{\mathbf{Z}}$. A scree plot involves plotting the log(eigenvalues) for each eigenvector and comparing the difference between them in order to select those above a noise baseline (low, constant slope prevalent at less significant components). Multiplying the selected eigenvectors by the state vectors over time gives the principal components' values across time.

3.3. Iterated Order and Noise Estimation, and Blind Adjusted Principal Component Analysis

When data is collected experimentally, there are often many sources for noise, whether it be 60-Hz line noise from electrical outlet or a loose or improper attachment of a sensor to the patient. In

many situations, such as this one, the noise is not known *a priori*. However, if it were somehow possible to estimate the noise's variance for each parameter (across time) in a patient state dataset, then one could use this to normalize the parameters across time before computing the covariance matrix and principal components (as is done when noise is known a priori in Noise Adjusted Principal Component Analysis (NAPC)). This would allow for a more accurate model of patient state and hypothetically lead to more accurate predictions of future state. A recently developed technique at the Professor D. H. Staelin's laboratory, namely Iterated Order and Noise Estimation (ION), can be used to produce such an estimate. In doing so, one can then proceed to normalize the parameter data channels based on the noise variance estimate, a method known as Blind Adjusted Principal Component Analysis (BAPC).

Briefly, the ION algorithm consists of the following. First, the input matrix of m x n (m vectors with n dimensions) is normalized (unity variance and zero mean for each variable). In fact, there is another way to visualize the dataset consistent with this. The **s(n)** vectors, each of length m, can be put together across time samples into matrix **L** with dimensions m x n. **L** can be defined as follows:

Equation 3

$$\underline{\underline{\mathbf{L}}} = \underline{\underline{\mathbf{F}}}\underline{\mathbf{P}} + \underline{\underline{\mathbf{G}}}^{1/2}\underline{\boldsymbol{\omega}}$$

In this way, **L** is the sum of linearly transformed stochastic signal **P** and an independent noise vector. In the above equation, **F** is the unknown mixing matrix[2]. The **ω** vector refers to a Gaussian noise vector with $\mu=0$ and covariance matrix being the identity matrix.

Next, let $\underline{\underline{\mathbf{G}}}_1$ be defined as the diagonal noise covariance matrix.

First, initialize $\underline{\underline{\mathbf{G}}}$ and $\underline{\underline{\mathbf{F}}}$ to the identity matrix. Next, the estimated noise variance are normalized to unity:

Equation 4

$$\underline{\underline{\mathbf{L}}}_k' = \underline{\underline{\mathbf{L}}}\ \underline{\underline{\mathbf{G}}}_k^{-1/2}$$

The signal order, p_{k+1}, is then estimated via scree plot of $\underline{\underline{\mathbf{L}}}_k'$. Next, $\underline{\underline{\mathbf{G}}}_{k+1}$ and $\underline{\underline{\mathbf{F}}}_{k+1}$ are estimated by using p_{k+1} and the Expectation-Maximization (EM) algorithm[8]. Lastly, the algorithm iterates on k (looping back to the normalization of the estimated noise variance) until convergence.

In the next section, the use of the mathematical toolbox delineated here will be explored for design of the heart rate predictor.

4. APPROACH

4.1. Strategy

To develop of strategy for modeling the patient state, the actual feedback system involving the patient and anesthesiologist was examined at the site where data was collected (Brigham and Women's Hospital (BWH), Harvard Medical School, Boston, MA). As shown in a simplified view in Figure 2, the closed loop system involves the anesthesiologist making adjustments to the drug vaporizer setting (which changes the amount of drug delivered to the patient) based on feedback from the patient via the patient-connected sensors and a monitor. These sensors record the patient state through 25 parameters (see Figure 3) every 20 seconds. The monitor then saves each sampled parameter in a patient history file. Such setups, including the patient sensors and monitoring

devices, are common in operating rooms in developed nations. However, the number of parameters data sampling, and recording capabilities vary substantially among institutions.

Figure 2

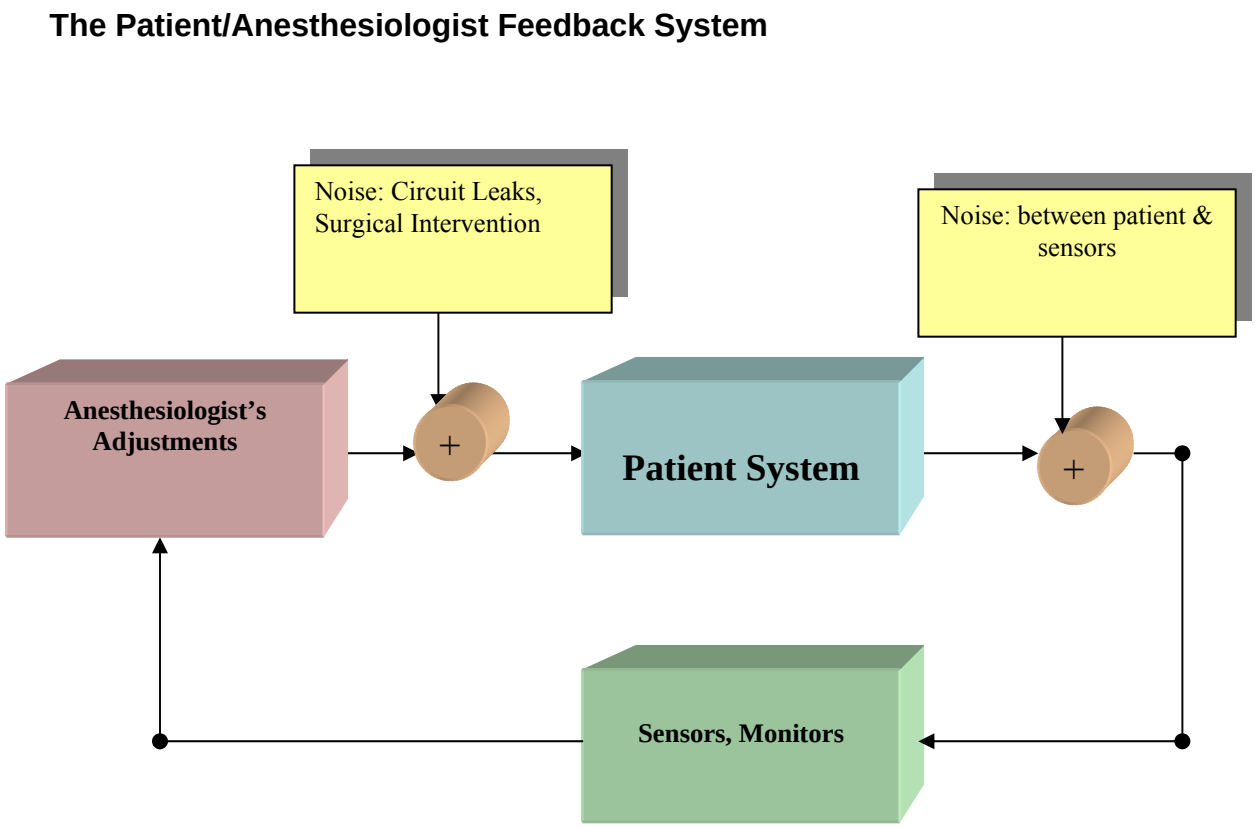


Figure 3

Patient Parameters Recorded via Sensors or Anesthesia Monitor

Number	*Parameter*	*Units*	*Description*
1	SYS	mmHg	Systolic Blood Pressure
2	DIA	mmHg	Diastolic Blood Pressure
3	MAP	mmHg	Mean Arterial Blood Pressure
4	HR	BPM (beats per minute)	Heart Rate
5	PR	BPM	Pulse Rate
6	SpO2	%	Oxygen Saturation
7	CO2 I	mmHg	Inspired CO_2
8	CO2 E	mmHg	Expired CO_2
9	RR	BPM	Respiration Rate
10	VE	L	Ventilation expired
11	VT	mL	Tidal Volume
12	Pmax	cmH_2O	Pressure Max.
13	Pmin	cmH_2O	Pressure Min.
14	PPlat	cmH_2O	Pressure Plateau
15	I:E	N/A	Inspired: Expired Ratio
16	O2 I	%	O_2 inspired
17	N2O	%	N_2O concentration
18	Agt I	%	Agent Inspired
19	Agt E	%	Agent Expired
20	Agt	N/A	Agent Identity
21	VT-sp	N/A	Tidal Vol. #2
22	Vent-st	N/A	Ventilation
23	iT-st	N/A	Miscellaneous
24	NIBPint	N/A	Miscellaneous
25	Marker	N/A	Clinician marked event

At Harvard Medical's Brigham and Women's Hospital, Dr. J. H. Philip created a collection of patient history files from operations including those where the aforementioned desflurane protocol was utilized. With a large number of samples, parameters, and patients, a method is needed to find the essential underlying information which can then be used in predicting heart rate.

A number of data parameters are correlated. For example, mean arterial blood pressure (MABP) is related to diastolic and systolic blood pressure:

Equation 5

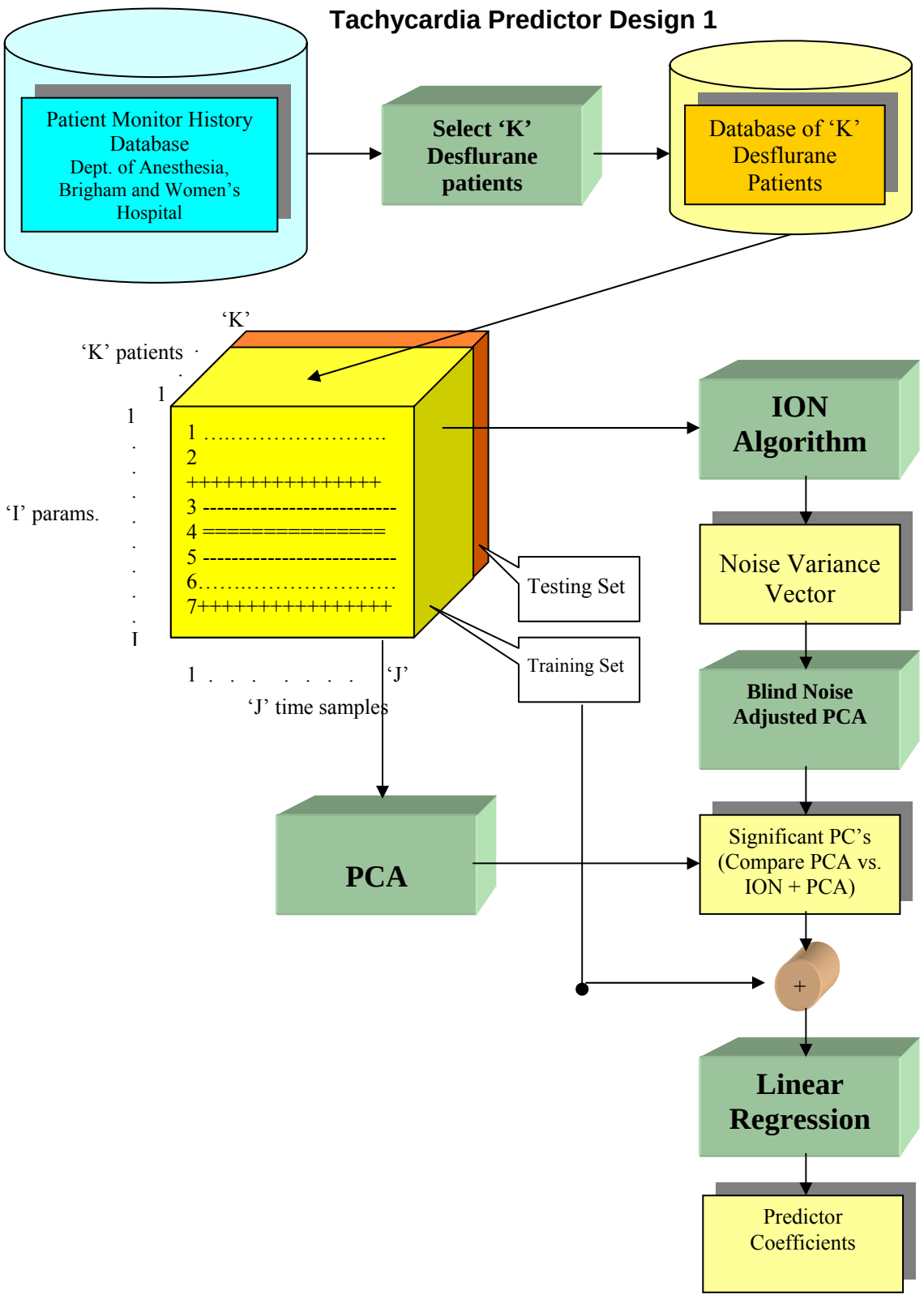
$$\text{Mean Arterial BP} = 2/3*\text{Diastolic BP} + 1/3*\text{Systolic BP}$$

It is expected that several of the respiratory and as well as other variables should be correlated. Thus, in this thesis, Principal Component Analysis (PCA) was used to reduce the dimensionality of the dataset for further analysis (as discussed in the 'Mathematical / Engineering Models' section). This method has been used effectively in similar multivariate estimation problems in manufacturing and remote sensing applications at Professor Staelin's Laboratory[9].

There are myriad sources of noise in the patient/anesthesiologist feedback system as depicted in Figure 2. For example, breathing circuit leaks can result in inaccurate sampling and recording of gases (including the drug agent). Surgical intervention may result in a change in heart rate and blood pressure (e.g. due to the incision as well as gradual blood loss during the operation). Random noise may also be introduced by various other mechanisms (e.g. if the patient is moved slightly). Lastly, each of the various sensors have certain limitations and must be connected properly to the patient to get an acceptable Signal to Noise Ratio (SNR) for recording.

Thus, it is hypothesized that the BWH-based data set may be a good candidate for testing the applicability of an ION algorithm recently developed in Professor Staelin's group. By employing this algorithm, it may be possible to estimate and remove noise so that additional information can be recovered from the dataset above baseline noise. This is done by normalizing the dataset using the noise variance vector from ION (see 'Mathematical / Engineering Models' section). As outlined in Figure 4 below, the result will then be compared to the normal Principal Analysis method (via scree plot) in order determine whether or not there is an improvement with the ION algorithm over the traditional technique. Thus, this thesis will serve as the first test of the ION algorithm's validity and applicability subsequent to the original ION work.

Figure 4


Tachycardia Predictor Design 1
Patient Monitor History Database
Dept. of Anesthesia, Brigham and Women's Hospital
Select 'K' Desflurane patients
Database of 'K' Desflurane Patients
'K'
'K' patients
'I' params.
1 ……
2 ++++++++++++++++
3 ----------
4 ==========
5 ----------
6……
7++++++++++++++++
Testing Set
Training Set
'J'
'J' time samples
ION Algorithm
Noise Variance Vector
Blind Noise Adjusted PCA
PCA
Significant PC's (Compare PCA vs. ION + PCA)
+
Linear Regression
Predictor Coefficients

In doing the PCA analysis, a number of questions arise. Should only the fundamental 25 parameters be used- or can certain derived featured help capture information in the dataset better for prediction of future heart rates? Examples include time shifted versions of certain parameters, integrals that preserve time history of drug administration, derivatives that capture rate of increase of drug agent or other parameter, and average drug delivery concentration over time. Also, some of the 25 parameters could be removed since they were not connected to a sensor and/or contained no recorded information. After several modifications, the fundamental and derived parameters that were analyzed in this thesis involved 32 parameters are depicted in Figure 5, with derived parameters starting at parameter number 23 continuing until parameter 32. The 'derivative' and 'integral' parameters are defined as follows. The derivative involves finding the first difference of the current and previous sample. Then, using the sampling rate (3 data points per minute), the first difference is multiplied by a coefficient to yield the change of parameter per minute (instead of per sample). The same is done with the integral except that instead that the first difference is replaced by a summation of the parameter value from time 0 (when the induction process began) up to the present time sample.

Figure 5

Fundamental and Derived Patient Parameters

Number	*Parameter*	*Units*	*Description*
1	SYS	mmHg	Systolic Blood Pressure
2	DIA	mmHg	Diastolic Blood Pressure
3	MAP	mmHg	Mean Arterial Blood Pressure
4	PR	BPM	Heart/Pulse Rate
5	SpO2	%	Oxygen Saturation
6	CO2 I	mmHg	Inspired CO_2
7	CO2 E	mmHg	Expired CO_2
8	RR	BPM	Respiration Rate
9	VE	L	Ventilation expired
10	VT	mL	Tidal Volume
11	Pmax	cmH_2O	Pressure Max.
12	Pmin	cmH_2O	Pressure Min.
13	PPlat	cmH_2O	Pressure Plateau
14	I:E	N/A	Inspired: Expired Ratio
15	O2 I	%	O_2 inspired
16	N2O	%	N_2O concentration
17	Agt I	%	Agent Inspired
18	Agt E	%	Agent Expired
19	VT-sp	N/A	Tidal Vol. #2
20	Vent-st	N/A	Ventilation
21	iT-st	N/A	Miscellaneous
22	NIBPint	N/A	Miscellaneous
23	PR_shift_7	BPM	HR 7 samples (2 min. 20 sec.) in the future
24	DERIV_AGT	% / sample	Derivative of Agent Expired
25	INT_AGT	% * sample	Integral of Agent Expired
26	AVG_AGT	%	Average of Agent Expired over time samples so far
27	HI_HR	N/A	Tachycardia present
28	HI_BP	N/A	Hypertension present
29	HI_HR_BP	N/A	Tachycardia or Hypertension present
30	DERIV_SYS	N/A	Slope of Systolic BP
31	DERIV_DIA	N/A	Slope of Diastolic BP
32	DERIV_HR	N/A	Slope of HR

Once the dataset has been transformed from 25 parameters and additional derived variables to a reduced number of significant principal components (PC's), this information can then be used to design a predictor of future heart rate. It should be noted that while the predictor was abstracted away as a black box in Figure 1, linear regression (unlike neural networks) can be used to develop an insight into the prediction process itself (i.e. allow one to peer inside the black box). By examining the predictor's weights for various state vector elements, the relevance of various parameters (from heart rate to blood pressure) to tachycardia prediction can be determined. This is more difficult in neural networks, where the hidden layer weights add additional complexity to the relationship between input and output.

As shown in Figure 4, linear regression can be used to find constant coefficients for each of the principal components across time plus a constant based on the training set data. Thus,

Equation 6

$$\underline{\mathbf{Y}} = \underline{\mathbf{A}}\underline{\underline{\mathbf{X}}} + \underline{\mathbf{B}}$$

Where: $\underline{\mathbf{Y}} = \underline{\mathbf{R}}(n+n_0)$ of size 1 x n

$\underline{\mathbf{A}}$ = Coefficients determined by linear regression of size 1 x p

$\underline{\underline{\mathbf{X}}}$ = Input matrix of p x n. Represents the 'p' significant PC's over 'n' time samples.

$\underline{\mathbf{B}}$ = Constant term determined by linear regression of size 1 x n

And: n = present time sample

n_0 = number of time samples between present and prediction

m = number of patient recorded parameters and derived features

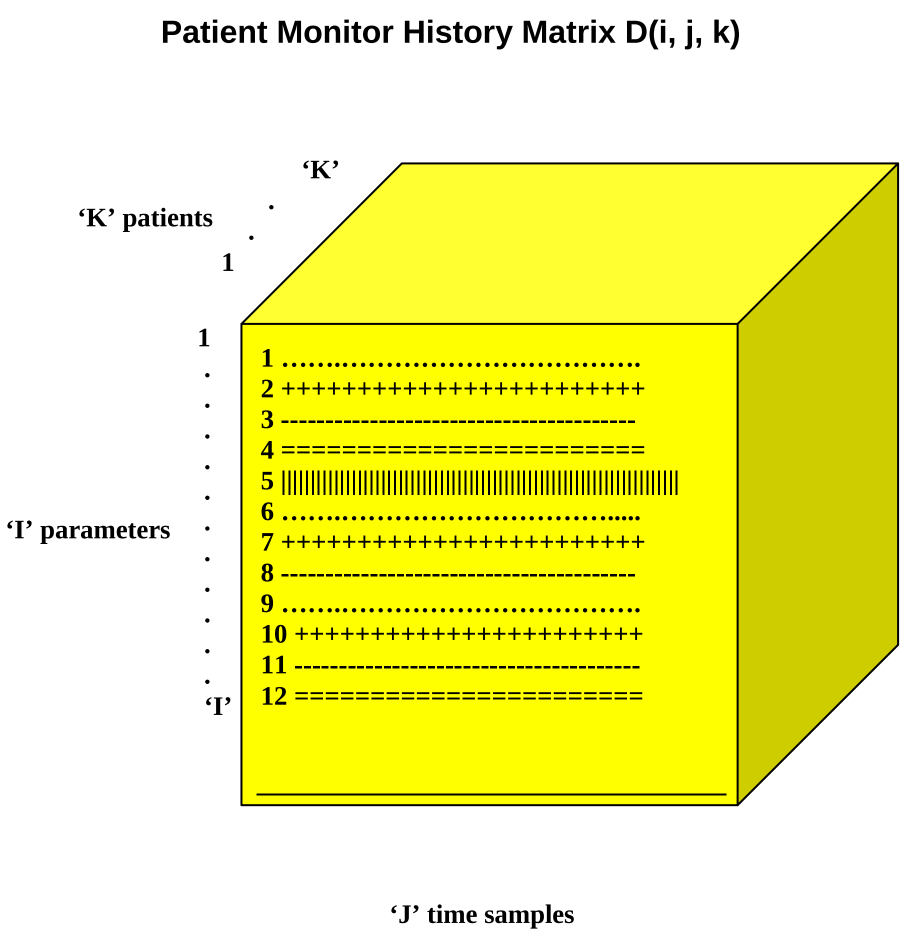
p = number of principal components used in reconstruction

In this thesis, a static heart rate(see 'Mathematics / Engineering Models' section), heart rate average-based (see 'Mathematics / Engineering Models' section), and linear regression-based predictors will be constructed and evaluated. Additional design issues are discussed more in the 'Design' section.

4.2. Required Equipment

The type of equipment necessary for exploration of this problem depends on the dimension of the patient monitor history database (illustrated in Figure 6).

Figure 6



For a rough calculation, one can assume 8 bytes of RAM per stored data value. As in Equation 7, let 'i' be the number of parameters, 'j' the number of time samples, and 'k' the number of patients. Thus, the number of megabytes (not including overhead for Windows and Matlab) required to hold one copy of the database in RAM is:

Equation 7

$$\text{Number of Megabytes of Memory} = \frac{8 \text{ bytes} * (j \text{ samples}) * (i \text{ parameters}) * (k \text{ patients})}{(1024*1024 \text{ bytes/Megabyte})}$$

In order to allow for a large database along with extra RAM for manipulating and transforming it, a machine of at least 64 Megabytes of RAM (required for Matlab for Windows) was used in this project. Ultimately, a Dell Dimension 4100 series 800 MHz Pentium III-based machine with 128 Megabytes RAM was used for this purpose. In addition, network capability was needed for connectivity to the Internet as well as access to the patient database located remotely at the Brigham and Women's Hospital.

Microsoft Excel was used for reading, analyzing, and plotting individual patient files. Matlab was then employed to actually create the algorithms to implement the design.

5. DESIGN, IMPLEMENTATION, AND TESTING ISSUES

A number of topics needed to be dealt with in the analysis and design of the predictor hypothesized in the 'Approach' section. First, a subset of patients (from the hundreds of patient monitor history

files) needed to be selected to create the database of patients for analysis. A number of files were corrupt, had bad data, missing data, or data from anesthesia administered by various other protocols (i.e. not the one outlined in the 'Clinical Background' [4]).

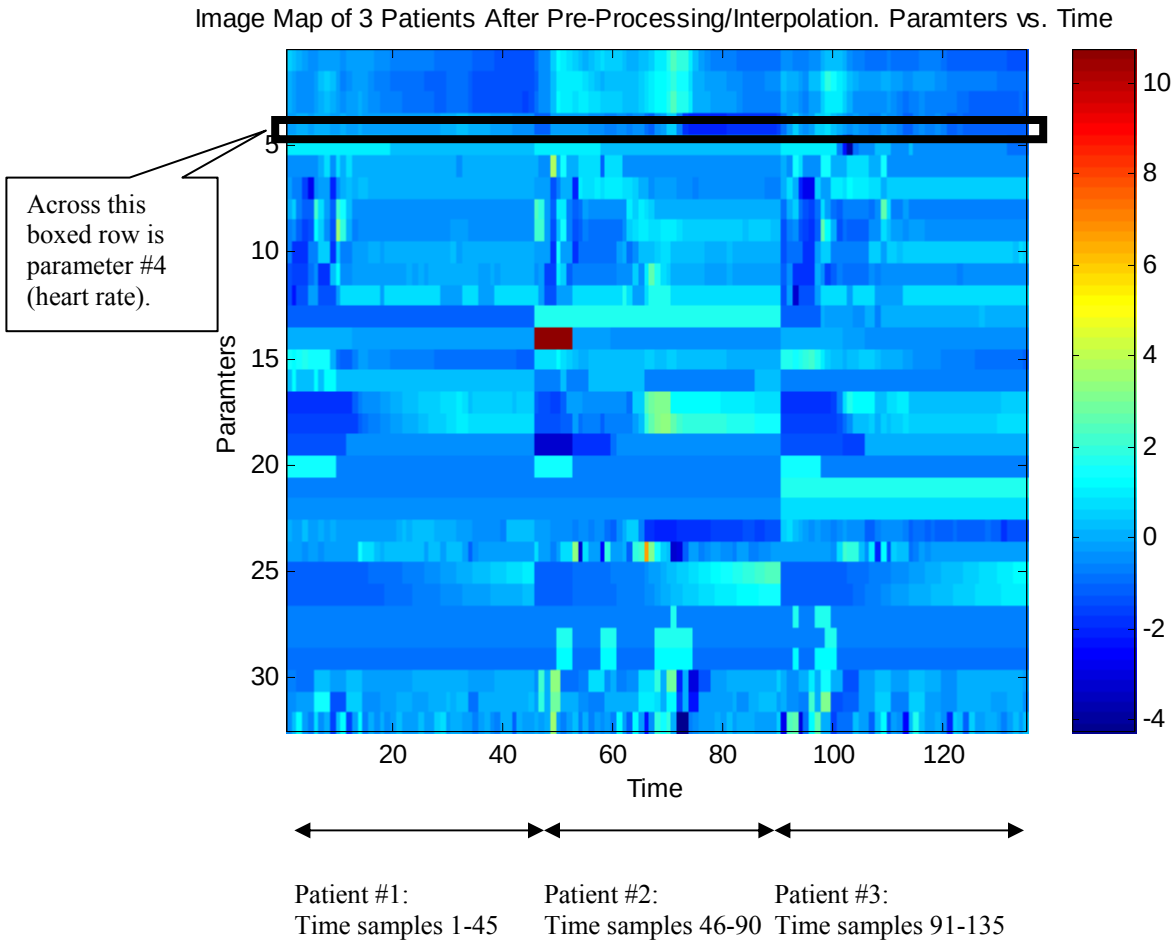
Also, the files are in a huge hierarchy of thousands of files and over 700 directories. A total of 23 patients were initially randomly selected from a pool of patients to test the model developed. Ultimately, this patient database was doubled to 46 patients randomly taken from the period of January 1998-July 2000.

With such an exorbitant amount of data, a number of other issues arose. For instance, an effective data structure had to be designed to allow for efficient access to data while conserving memory. When determining how to analyze data, it can also help to visualize it to facilitate discovery of patterns and intuition for further explorations. Yet, with 25 parameters for just one patient and around 180 samples per hour, visualizing just one patient (with an operation duration of around 4 hours) would involve over 18,000 points. Looking across patients adds a 3rd dimension to create 3-d matrix **D**(i, j, k) in Figure 6. It seemed that another visualization method besides plotting each parameter versus time is a prerequisite to making sense of the data.

The method selected for visualization consists of an image mapping of **D**, a process which essentially involves converting **D** into a series of patients concatenated together to form the all-inclusive **L** matrix representation of the dataset used as an input to ION (as discussed in 'Mathematics / Engineering Models'). Basically, it consists of taking slices of **D** shown in Figure 6 holding k constant. Thus, the data for one patient appears first followed by the next patient. As seen in Figure 7 based on Figure 6 from the previous section, the 'Parameters' y-axis is equivalent to 'i' in **D** and 'j' is equivalent to the 'Time Samples' x-axis for 0 to J. The next patient's data is

presented sequentially at J+1, right after the first one ends. Due to the scaling of the image map, each data point is represented as a vertical bar colored according to the corresponding **D** data point value (or a normalized version of it). For example, Figure 7 clearly shows three patients with 45 time samples each (notice how the pattern repeats after time samples 45 and 90). Here, data values are normalized so that variance is unity for each parameter. The fact that this figure suggests a pattern in patient behavior during desflurane induction hinted information could be garnered for predicting future state of an unknown patient in real-time by using a predictor first trained on a historical patient database.

Figure 7

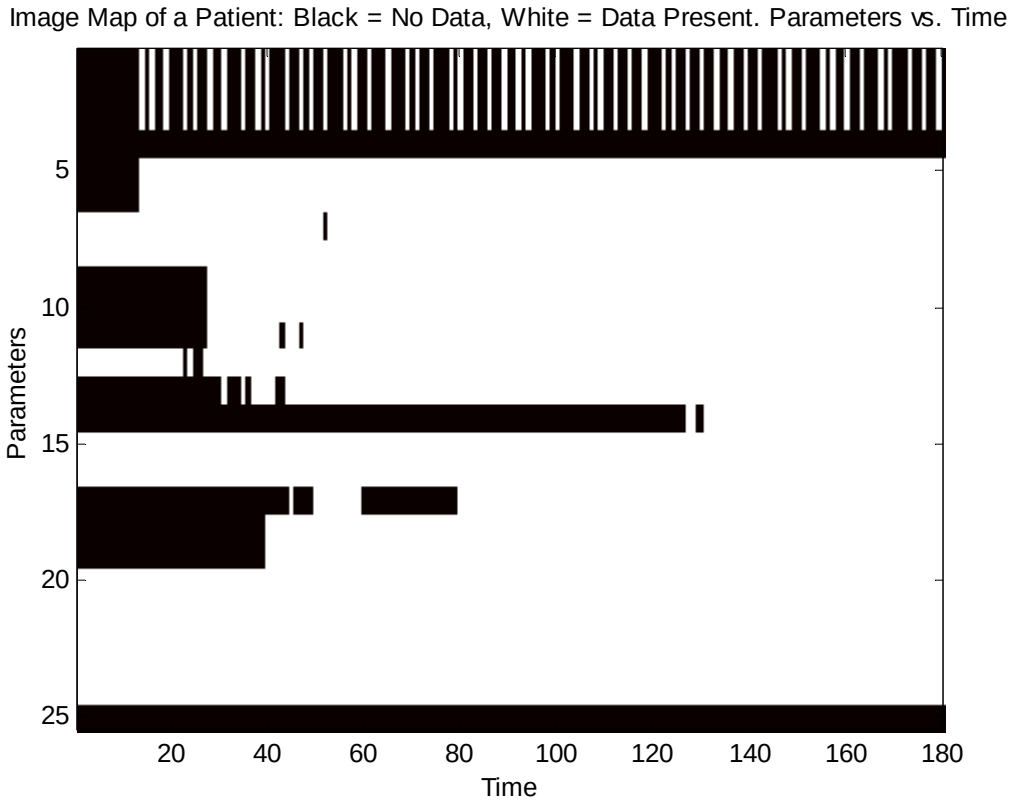


Special routines need to be written to parse the data and concatenate patient monitor files (which contain only one hour of data per file). Patient files contained patient information based on the patient's operation date and the order of the patient on that day's schedule for a given operating room (e.g. second patient to be operated in the given OR, etc.). Patient names were not included in the files, although schedules with the patient name and doctor transcribed patient history could be ordered based on this information. Additional identifiers to uniquely specify a patient (due to the fact that patients' file names/numbers are not unique) need to be developed. Ultimately, patients were assigned serial numbers based on the patient's operation date and initiation time of the operation.

Since the patient monitor history files contain the patient data for all of the operation (not just the induction protocol), it was necessary to determine which time samples should be used for analysis. For example, the blood pressure, pulse rate, and oxygenation sensors are usually attached by the clinician separately after the monitoring device is recording values for other variables. Also, certain parameters, such as blood pressure, are effectively asynchronous events which arrive from the blood pressure machine with 1/3 the sampling rate compared to the other parameters. Figure 8 shows the aforementioned issues visually using the same image map notion delineated earlier for the first hour of the operation. In this case, a data point is colored black if no data or invalid data was recorded at a given time and parameter. White represents valid data. One can see that the black extends to various different time samples for different parameters (and, in fact, stretches for all time for parameters 4 and 25). This suggests that the sensors for each parameter were connected at different times. Parameter 4 allows for an additional optional method of measuring parameter 5 (pulse rate) which was not taken advantage of in this case. Parameter 25 is initiated by clinician input to mark various events during the operation. Also, the top three rows (parameters 1 to 3 are blood pressure

measurements) can be seen to have valid data (white) every third sample as discussed previously. This all suggested that some pre-processing would be needed before a meaningful patient data subset could be used to train a predictor.

Figure 8

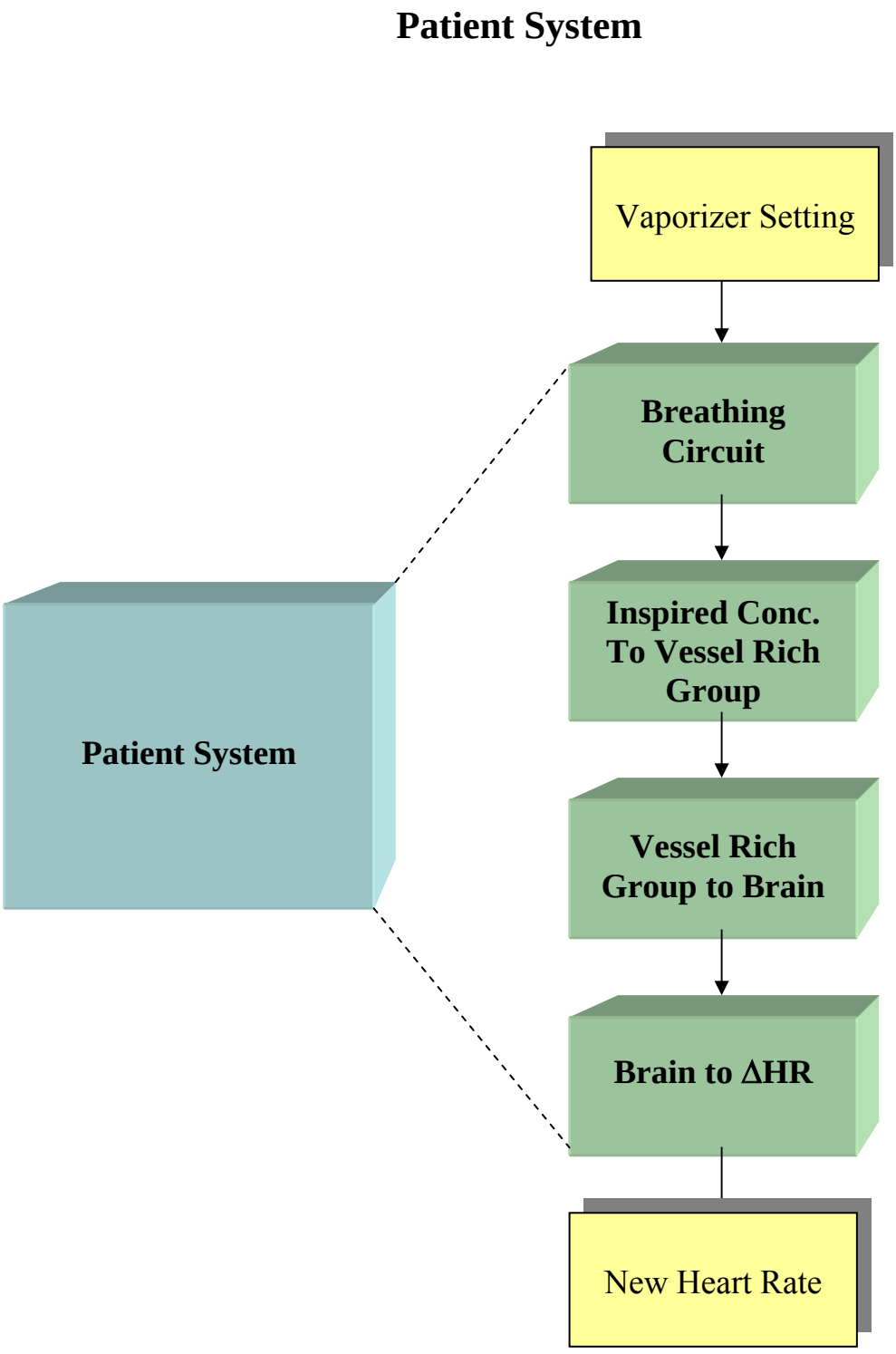


In order to facilitate automated pre-processing, a program was written to look for the first time sample when all of the sensors were connected and functioning. Once this starting point has been found, the time of initiation of induction by inhaled desflurane is located and time samples are taken for the next 15 minutes (45 samples at 3 samples/minutes). This starting point is determined based on when the inspired agent concentration increases above the 0%. In addition, to correct for missing data and less frequently sampled data, linear interpolation is used. If there are too many

consecutive points missing for a given parameter (other than parameters 4 and 25 which were not used), then the patient file was flagged. The result of such pre-processing is shown visually in Figure 7 for three patients as discussed previously.

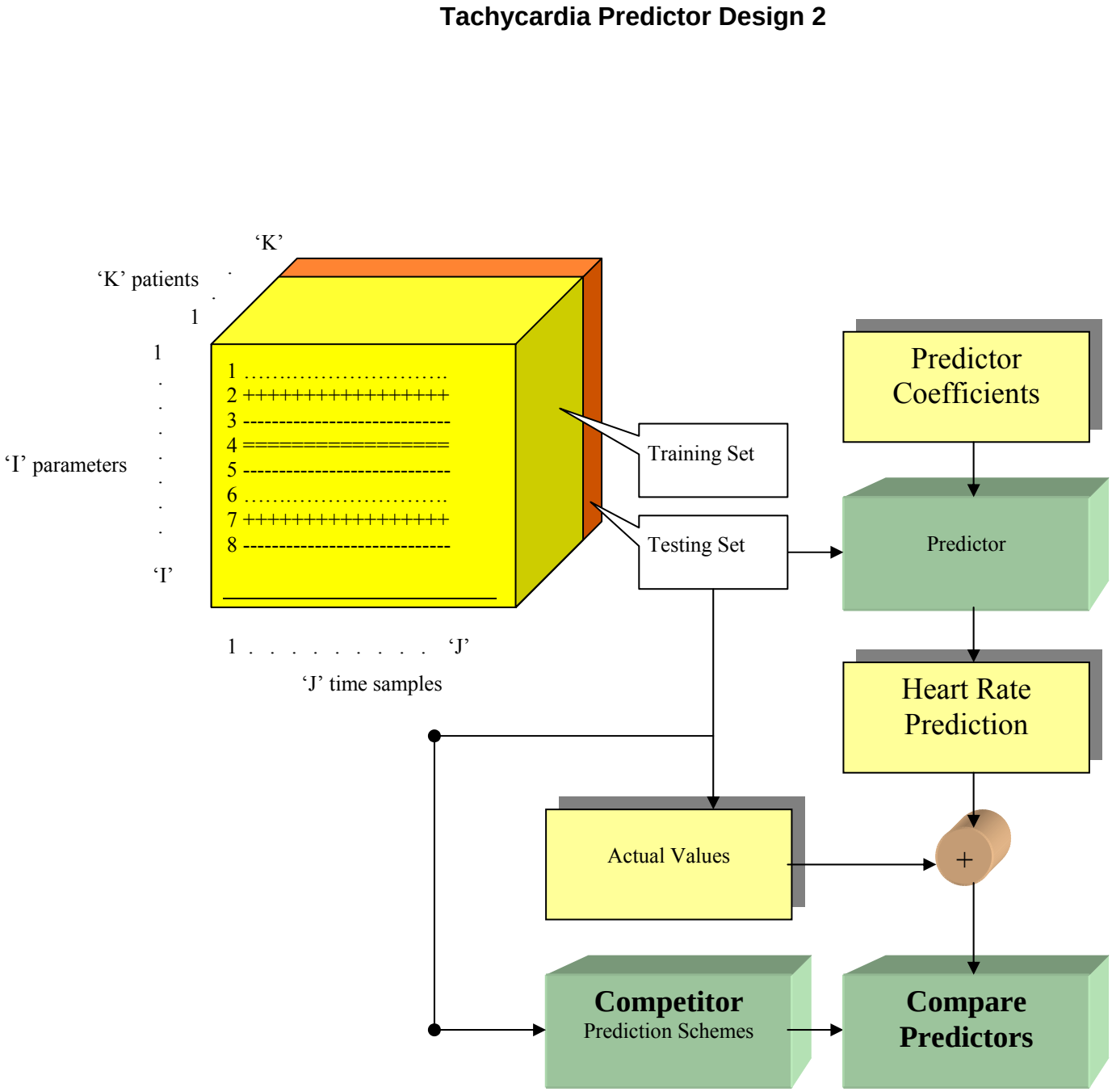
In designing the predictor for tachycardia, some areas need to be examined closely. For instance, how much ahead should such a prediction be made. This depends on a number of factors including response times for: breathing circuit, inspired drug concentration to vessel rich group (VRG), VRG to brain, and brain to effector sites (where a change in heart rate is observed). This is outlined in Figure 9. It is thought that this time may be approximately 2 ½ minutes from inspired drug concentration (an observable variable) to a delta heart rate[10].

Figure 9


Patient System
Vaporizer Setting
Breathing Circuit
Inspired Conc. To Vessel Rich Group
Vessel Rich Group to Brain
Brain to ΔHR
New Heart Rate
Patient System

Another issue is how a predictor like this can be evaluated. Certain standards such as root mean square (RMS) error and residue plots can provide an objective measure, but what should the predictor be compared against? One proposed idea is to compare the predictor's results with 'dumb' predictors as discussed in the 'Mathematical / Engineering Models' section. Comparing the RMS errors and residue plots of such a predictor, one can determine if any knowledge about tachycardia can be derived from the dataset in predicting the condition in other patients. An additional comparison can be done by looking at the ROC curves (probability of detection (Pd) vs. probability of false alarm (Pfa)) for each predictor as the threshold for detection is systemically varied. If a clinical assessment of the costs/risks associated with each was ascertained, then the usefulness of a given predictor could be surmised. This approach is shown schematically in Figure 10 below.

Figure 10


Tachycardia Predictor Design 2
'K'
'K' patients
1
1
'I' parameters
'I'
1 .....
2 ++++++++++++++++++
3 ----------
4 ==========
5 ----------
6 .....
7 ++++++++++++++++++
8 ----------
1 . . . . . 'J'
'J' time samples
Training Set
Testing Set
Predictor Coefficients
Predictor
Heart Rate Prediction
Actual Values
+
Competitor Prediction Schemes
Compare Predictors

6. RESULTS

Ultimately, the dataset examined included 46 patients, each with 45 data samples of induction by desflurane, and 32 parameters as outlined in Figure 5. The derived parameters included actual target heart rate 7 samples (2 min. 20 sec.) in the future, derivatives (agent expired, HR, systolic BP, Diastolic BP), integral of agent expired, average of agent expired over time samples so far in given patient, and presence of certain conditions for current sample (e.g. tachycardia, hypertension).

Using this dataset, the approach outlined in the 'Strategy' and 'Design' sections was implemented. After Blind Adjusted Principal Component Analysis, a scree plot suggested 5 PC's were significant. Shown in Figure 11 are the scree plots for PC's derived using Blind Adjusted Principal Component Analysis (i.e. with ION) and PCA without ION. Figure 12 shows a zoomed in view focusing on the PC's corresponding to the top 5 eigenvalues. While only approximately 3 PC's are significant for PCA without ION, the ION method allows for additional components to be exposed above the noise. The ION-based PCA values are also several orders of magnitude above those yielded without ION as shown in Figure 12.

Figure 11

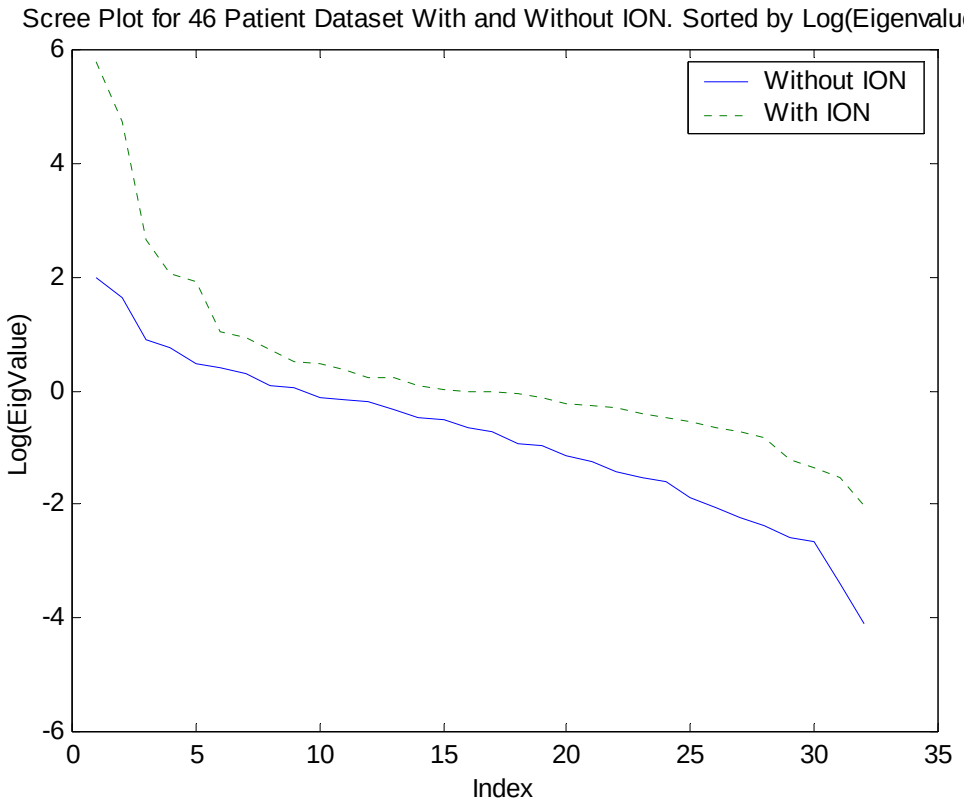
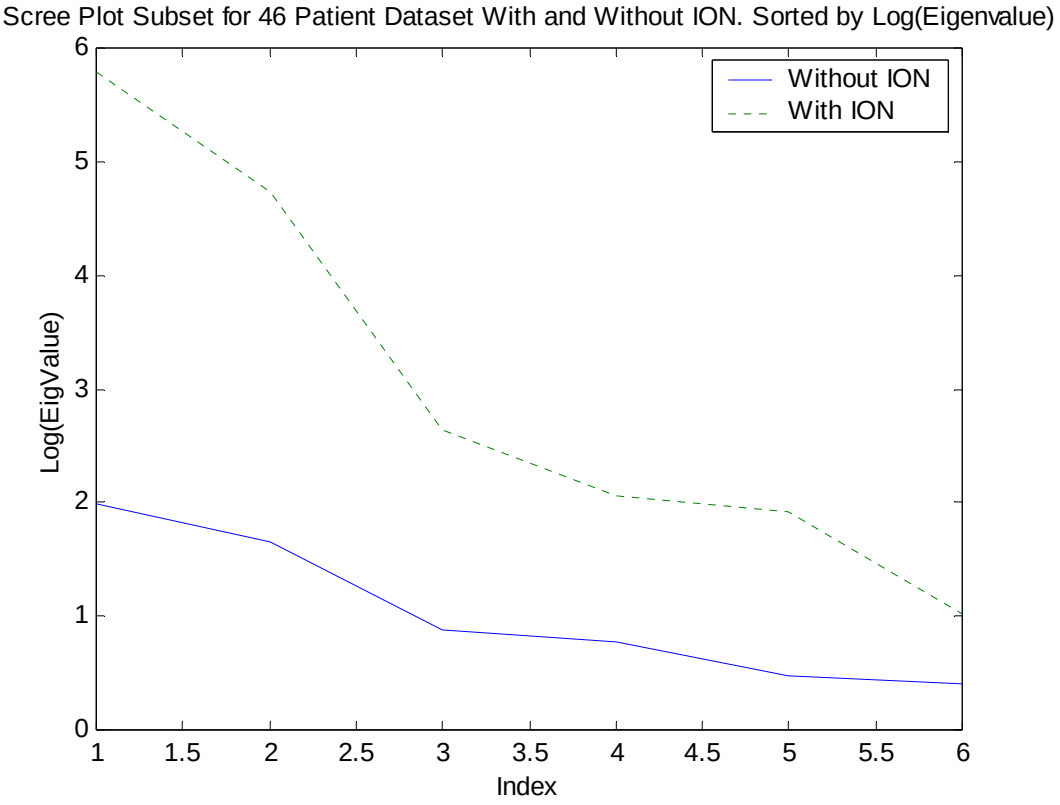

Scree Plot for 46 Patient Dataset With and Without ION. Sorted by Log(Eigenvalue)
Without ION
With ION
Log(EigValue)
Index
6
4
2
0
-2
-4
-6
0
5
10
15
20
25
30
35

Figure 12



In order to examine the effect of the underlying parameters on each principal component, the top five weighted parameter coefficients of each PC were calculated. The weights were then renormalized to effectively yield unity variance for each parameter in order to facilitate comparison. Looking at the PC's revealed some interesting information regarding the dataset. As shown in Figure 13, the first and second PC's weigh various blood pressure measurements heavily along with the agent concentration. It is expected that these be grouped together since they are related as shown in Equation 5 previously. It is interesting the expired agent concentration is also grouped here, exhibiting a correspondence between blood pressure and agent as suggested in the literature[11].

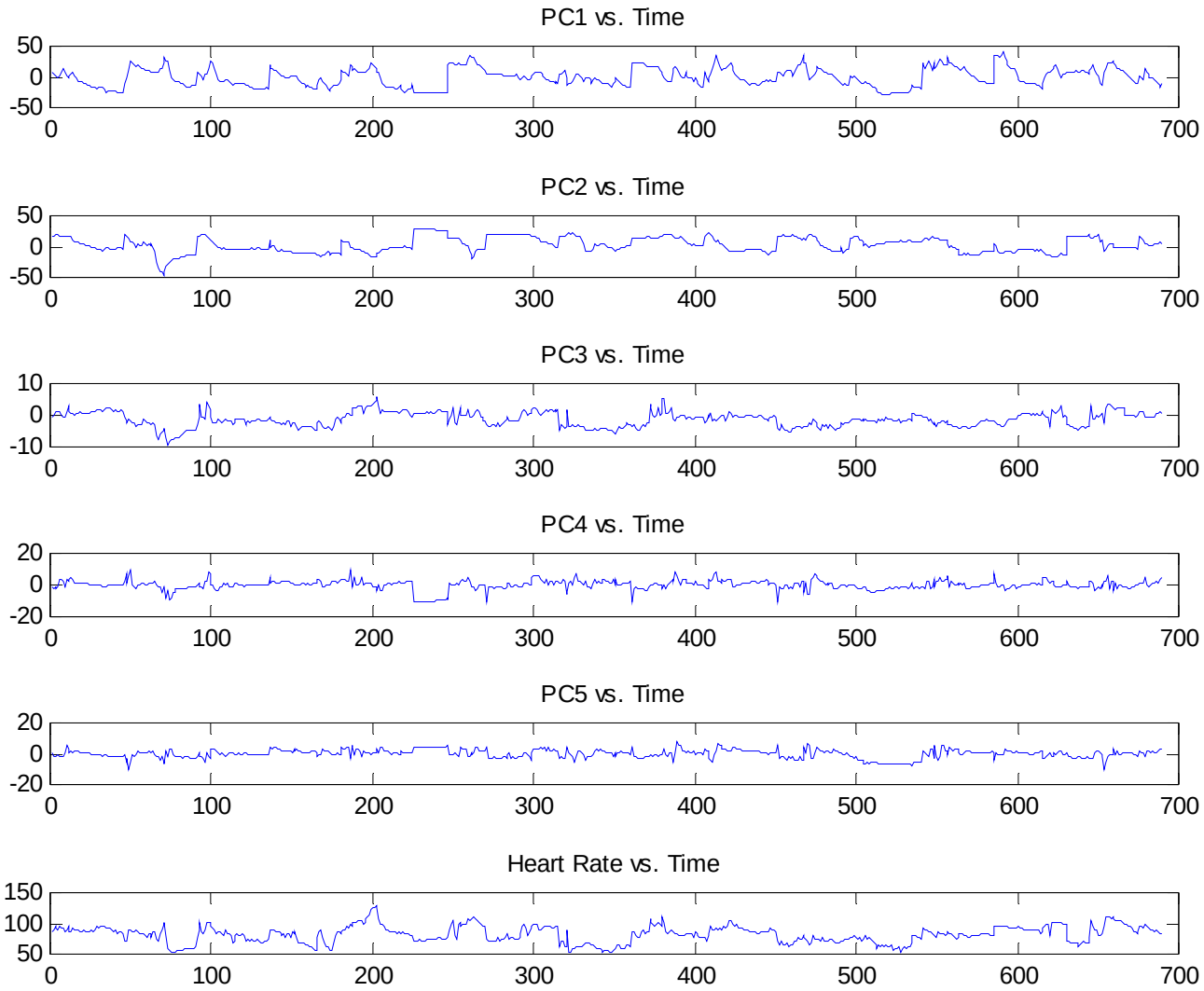
The third PC is basically the heart rate and blood pressure dimension along with agent interaction. PC #4 categorizes the interaction of the agent with the derivative of blood pressure and ventilation variables (VT, the tidal volume and Pmax, the maximum pressure in lungs during breathing). The fifth component contains similar variables, with different signs and less evenly distributed weights. Looking one PC beyond the 5 significant ones, the sixth principal component relates to the agent past and present concentration in combination with BP (specifically systolic). It includes the following agent components: the integral of expired agent (intuitively proportional to the amount of drug given so far), the agent inspired concentration (approximately proportional to current drug concentration being administered), and the current average of expired agent (recently administered dosage).

Figure 13

PC #1	BP / Agent Interaction I		PC #2	BP / Agent Interaction II	
Number	Name	Value	Number	Name	Value
3	MAP	15.20736	18	Agt E	-9.509
18	Agt E	-2.73056	3	MAP	-4.231
2	DIA	0.966681	17	Agt I	-0.717
1	SYS	0.539533	2	DIA	-0.241
4	PR	0.250657	15	O2 I	0.216
PC #3	**Agent Interaction: HR and BP**		**PC #4**	**Agent Interaction: BP and Ventilation I**	
Number	Name	Value	Number	Name	Value
4	PR	2.478	18	Agt E	-1.803
3	MAP	-1.965	31	DIA slope	1.081
18	Agt E	-0.750	30	SYS slope	1.065
27	Tachy	0.594	10	VT	0.699
29	Tachy or Hypertension	0.382	11	Pmax	0.609
PC #5	**Agent Interaction: BP and Ventilation II**		**PC #6**	**Agent Monitors / BP**	
Number	Name	Value	Number	Name	Value
18	Agt E	-1.430	17	Agt I	1.148
31	DIA slope	-0.891	25	Integral of Agent Expired	-0.701
30	SYS slope	-0.843	1	SYS	-0.643
10	VT	0.801	26	Avg of Agent Expired over time so far	-0.582
11	Pmax	0.723	29	Tachy or Hypertension	-0.385

The 5 significant PC's were plotted against time in Figure 14 below and compared to the heart rate variable to examine if tachycardia events (HR > 100 BPM in the graph) could be traced to a preceding fluctuation in one or more of the PC's by inspection. Even visually, it can be seen that tachycardia periods and peaks correspond to PC #3 at several time samples including 200 and 370 (among others). As is later confirmed, it would make sense that such a principal component would be weighted heavily in a ION PCA-based prediction scheme.

Figure 14



The relationships between PC's was also explored. Figure 15, Figure 16, and Figure 17 are scatter plots of all 2 dimensional graph permutations of PC3, PC2, and PC1. In PC1 vs. PC2, one can see two clusters. One shows a linear relation between PC1 and PC2 with a slope of approximately -4 and the other is more scattered. Two scattered clusters are also apparent in PC1 vs. PC3. PC2 vs. PC3 has several closely clustered groups.

Figure 15

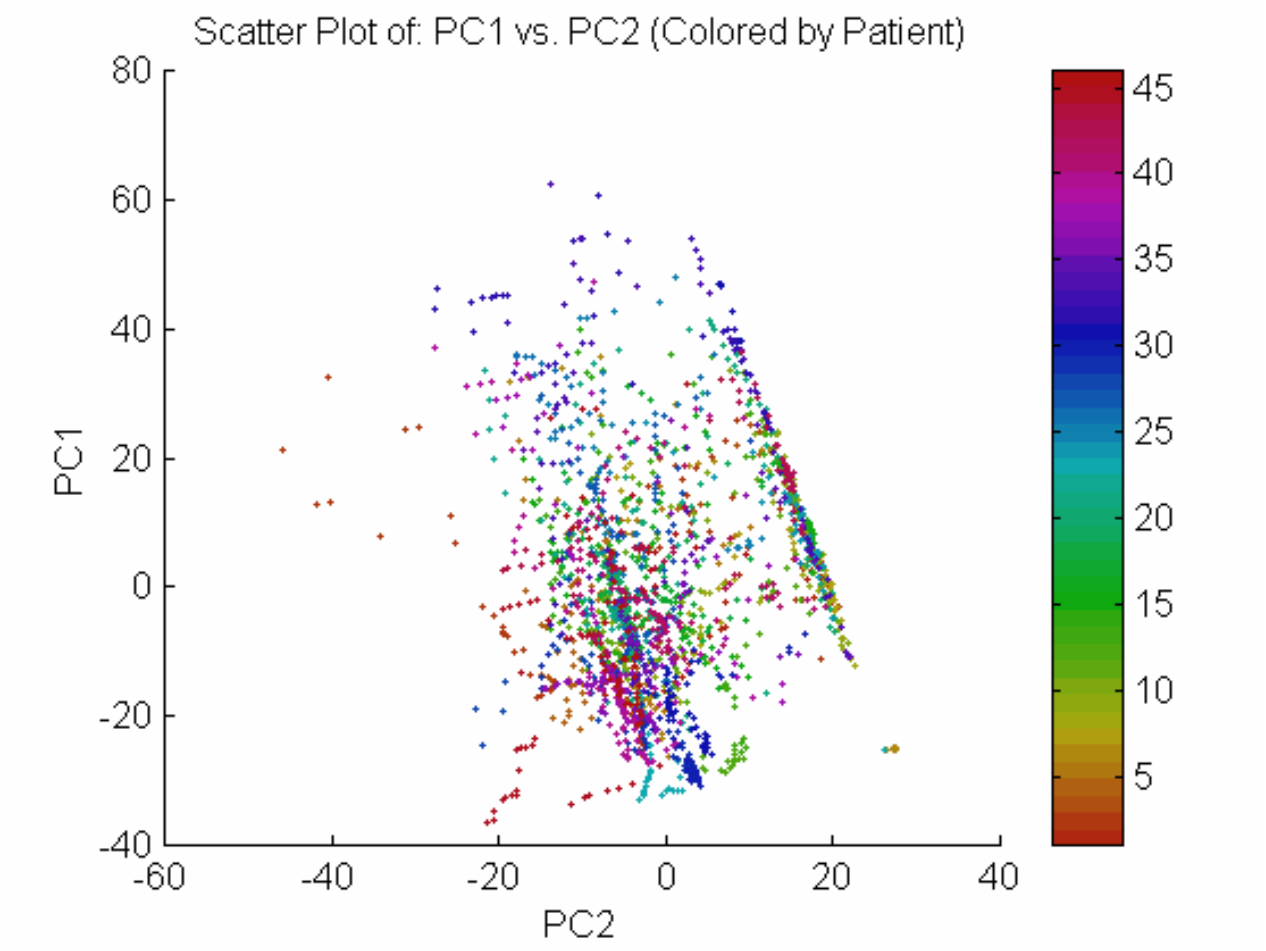


Figure 16

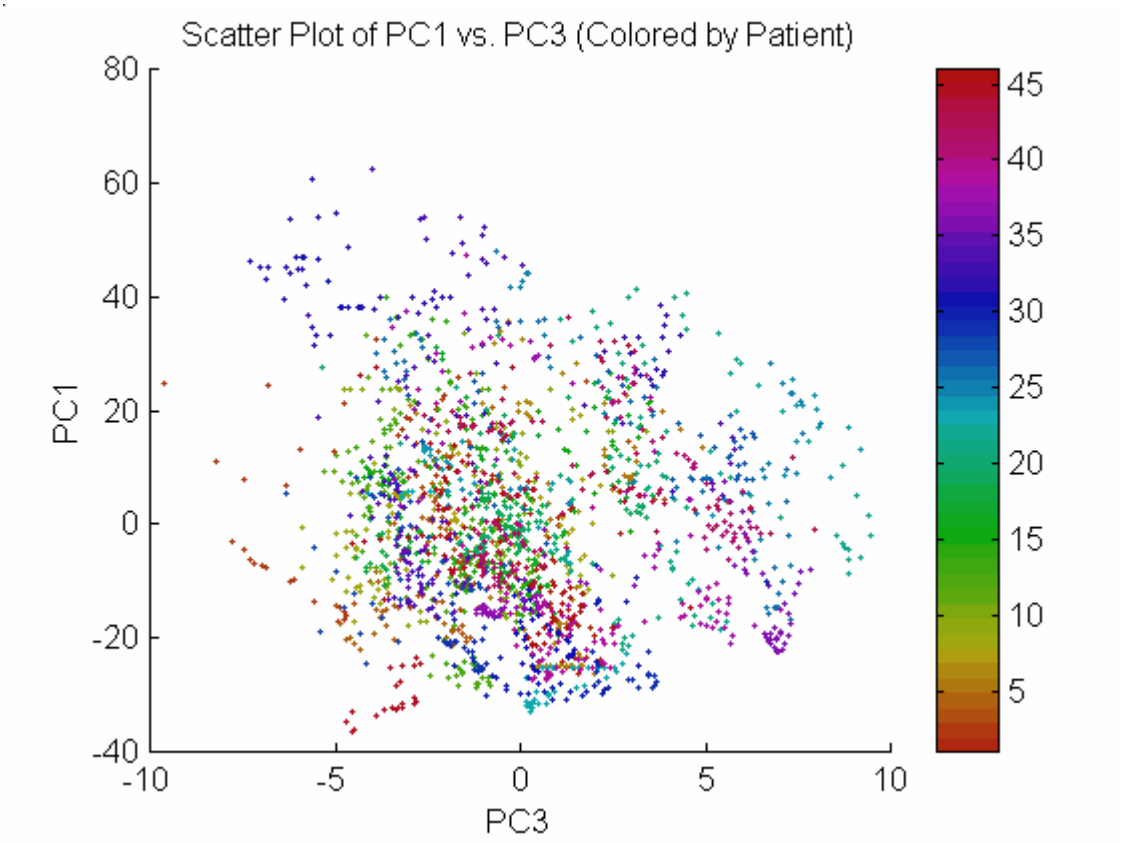
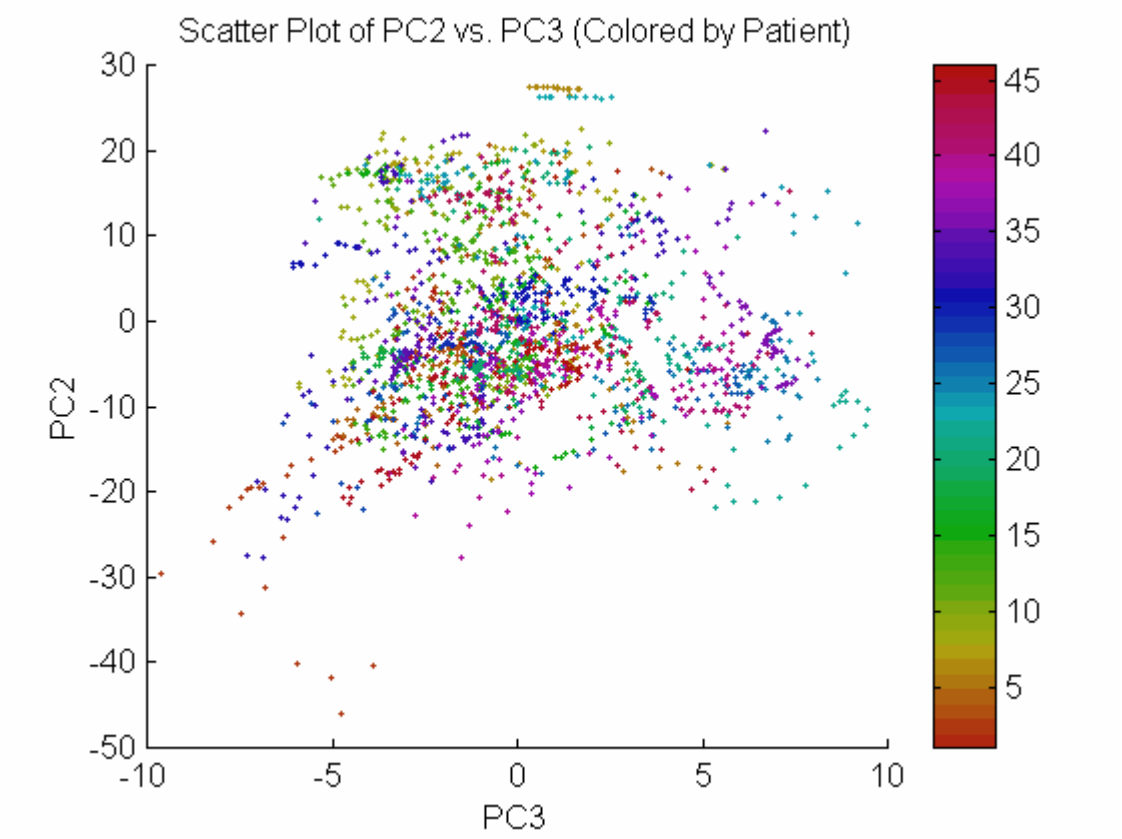

Scatter Plot of PC1 vs. PC3 (Colored by Patient)
PC1
PC3
80
60
40
20
0
-20
-40
-10
-5
0
5
10
45
40
35
30
25
20
15
10
5

Figure 17



A better division of the clusters can be seen in Figure 18, a 3-D plot of the three most significant PC's. From most perspectives, the 3-D plot looks like a single amorphous cluster. However, when rotated to the perspective as in Figure 18 (and Figure 19), it is clear that a plane bisects two hemispheres that represent two distinctive clusters. In order to explore this further, the Figure 19 was colored by time from the start of the induction of inhaled gas protocol (as opposed to coloring by patient as was done in Figure 15 - Figure 18). This time colored figure suggests that the two clusters are not according to different relative times from the start of the induction of desflurane. Thus, a closer look was taken at the coloring by patients. Three random patients were colored

differently (aqua, blue, and green) from the mass of others (colored red). As evident in Figure 20, patients can extend across the boundary of clusters. Thus, the clusters don't separate patients either.

When thinking about the notable pattern evident in the figure (e.g. plane separating two clusters), it can be helpful to think of an analogy of the shape of the scatter plot to that of a 'p' electron orbital from quantum theory. In 'p' orbital, there are two clusters separated by a node. At the node, there is zero probability of an electron existing due to physical constraints. Thus, the electron can be either on one side or the other, but is with probability approaching zero at the node. It is possible that the patient state scatter plot can be similar in nature, but with the constraint being physiological rather than physical. Instead having a node, the two clusters are separated by a plane. Patient state can pass from one state (cluster #1) to another (cluster #2) while not going through the bisecting plane, or transition area.

To examine this further, the PC's involved in the gap were examined. It appears that the grouping involves an interaction between HR, BP, and agent concentration. This interaction suggests that as HR is increased, the combination of BP and agent in PC1 and PC2 are constrained so that permit patient state values are restricted to a certain distance from the bisecting plane.

Interestingly, as shown in the figure, the patients form sub-cluster within each cluster and have few, if any, points between these sub-clusters. The blue and aqua patients' pairs of sub-clusters are clearly evident. The green patient also has a couple of points on the upper cluster (just under the blue patient) besides the sub-cluster on the lower cluster. The position of these sub-clusters within the two larger clusters might yield valuable information as well. Additional, quantitative exploration of these constraints and sub-clusters is left open for further work in this area.

Figure 18

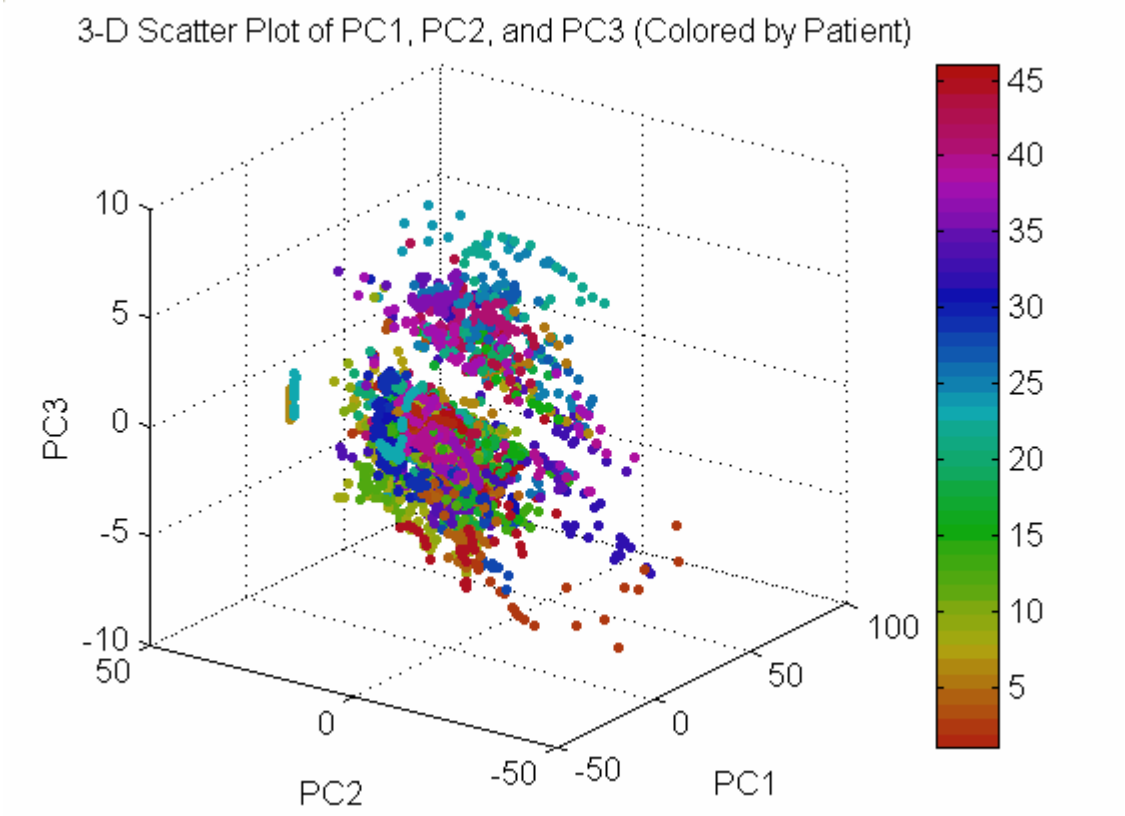

3-D Scatter Plot of PC1, PC2, and PC3 (Colored by Patient)
PC3
10
5
0
-5
-10
PC2
50
0
-50
PC1
-50
0
50
100
45
40
35
30
25
20
15
10
5

Figure 19

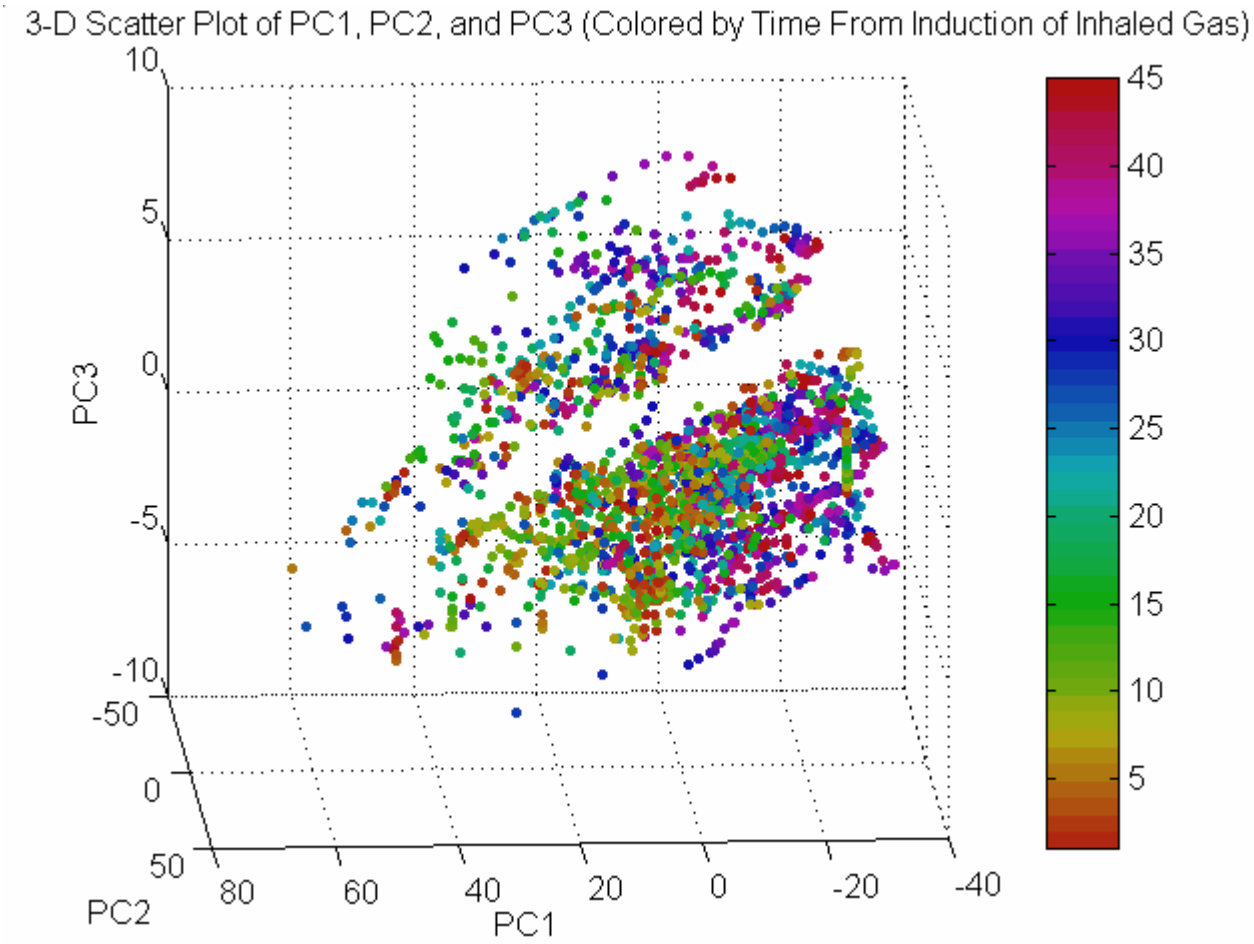
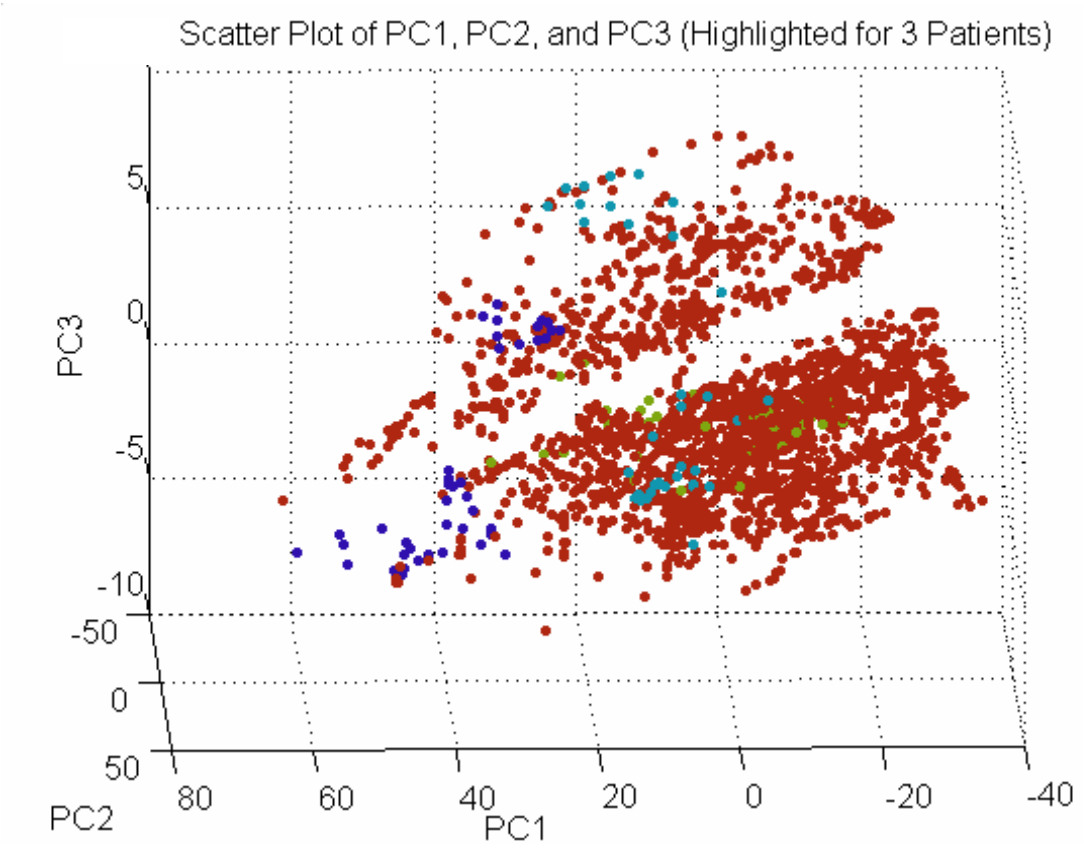

3-D Scatter Plot of PC1, PC2, and PC3 (Colored by Time From Induction of Inhaled Gas)
PC3
10
5
0
-5
-10
PC2
-50
0
50
PC1
80
60
40
20
0
-20
-40
45
40
35
30
25
20
15
10
5

Figure 20



From these figures, it is evident that one can trace a trajectory which tracks patient state temporally across these PC's (see the vertical appearing line of points extending from PC1=-1 to PC1=-3 along PC2=50, PC3=0). In a five dimensional space, such a trajectory would encompass essentially the complete specification of patient state. A zoomed in view in shown in Figure 21 and Figure 22, again color coded by patient and time respectively. Figure 22 confirms that direction of trajectory of the aqua colored patient seen in Figure 21 starts from lower right and moves upward varying along PC1 (vertical axis) and PC3 (diagonal axis nearly orthogonal to plane of paper). A number

of other patients exhibited similar trajectory patterns, a discovery whose ramifications will be discussed in further detail in the 'Conclusions' section.

Figure 21

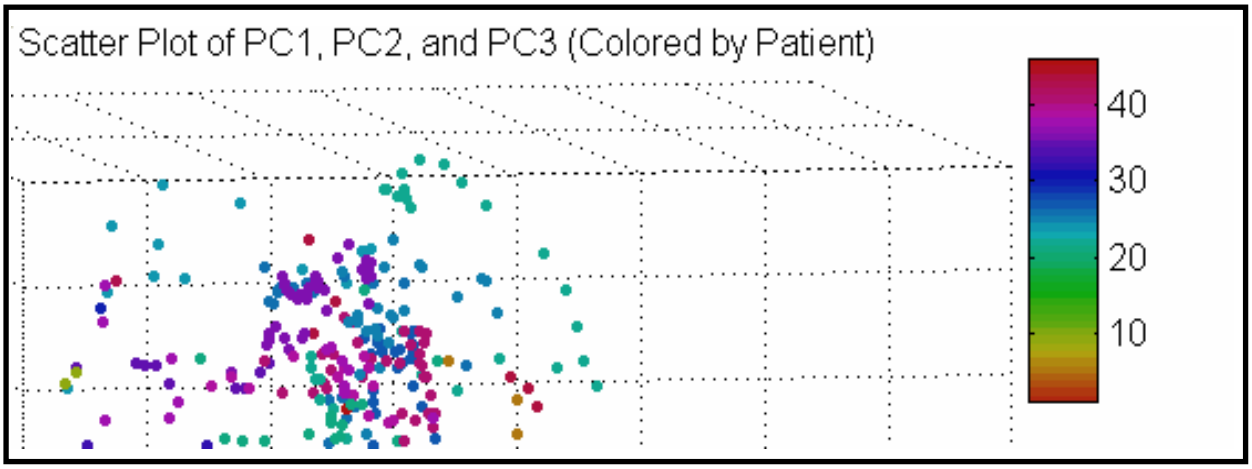
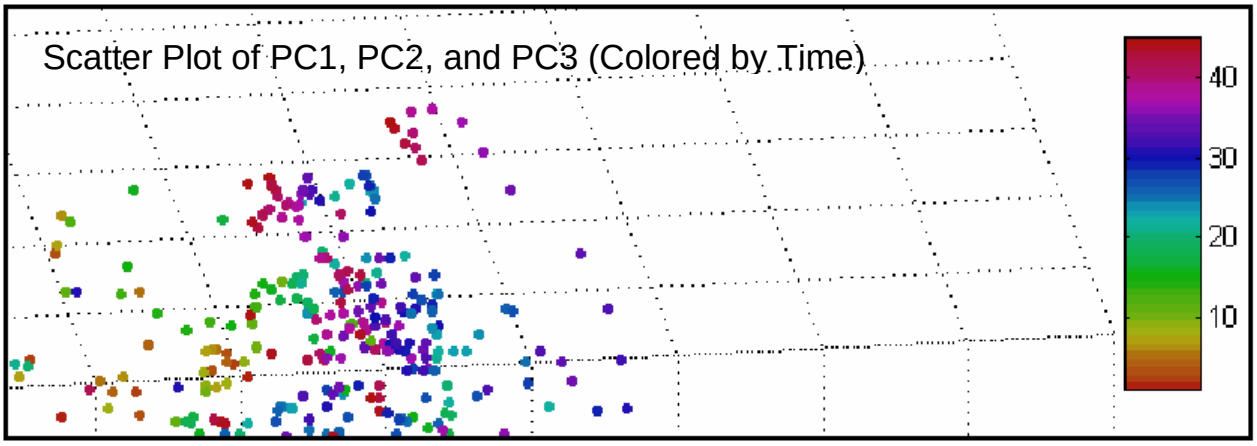
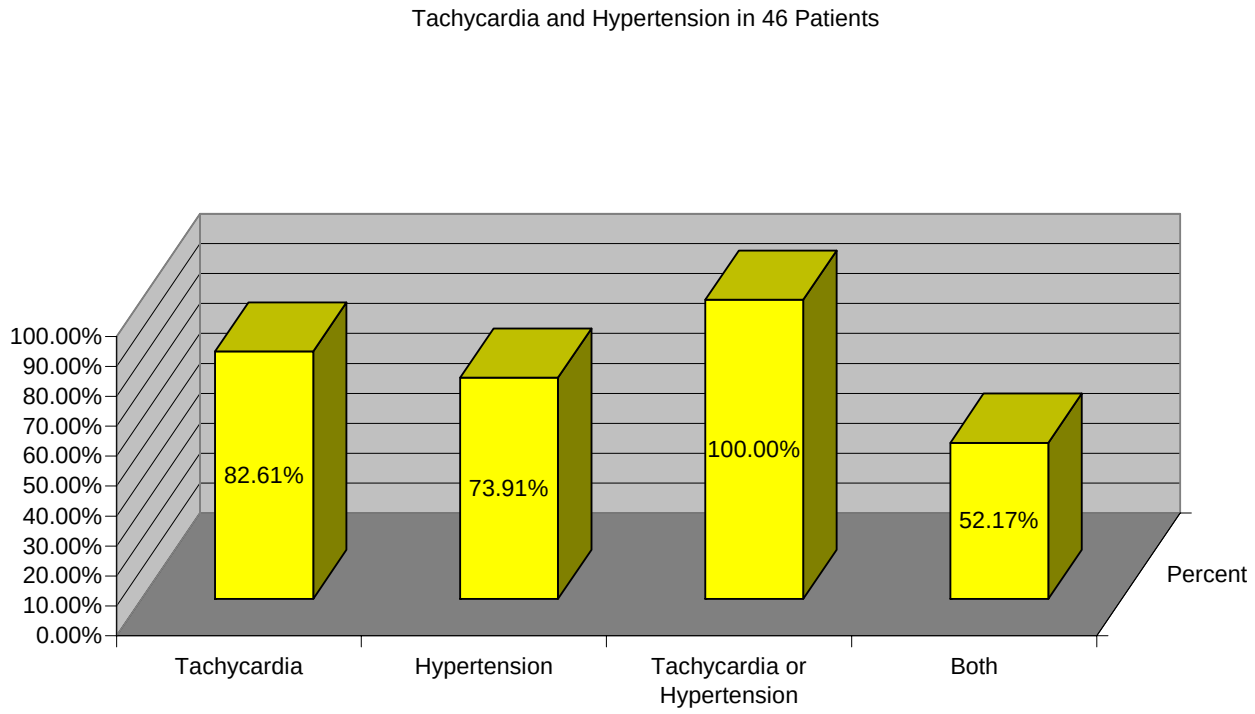


Figure 22



Next, the number of cases of tachycardia and hypertension were explored. While it was expected that some patients would exhibit one or the other of these conditions based on work by Muzi[12], it was surprising to find that ***all*** of the patients exhibited tachycardia, hypertension, or both at some point during the first hour of the operation. In fact, more than 3/4 of patients had at least one episode of tachycardia. See Figure 23 below for details. Thus, there were plenty of example cases for training and testing a predictor model.

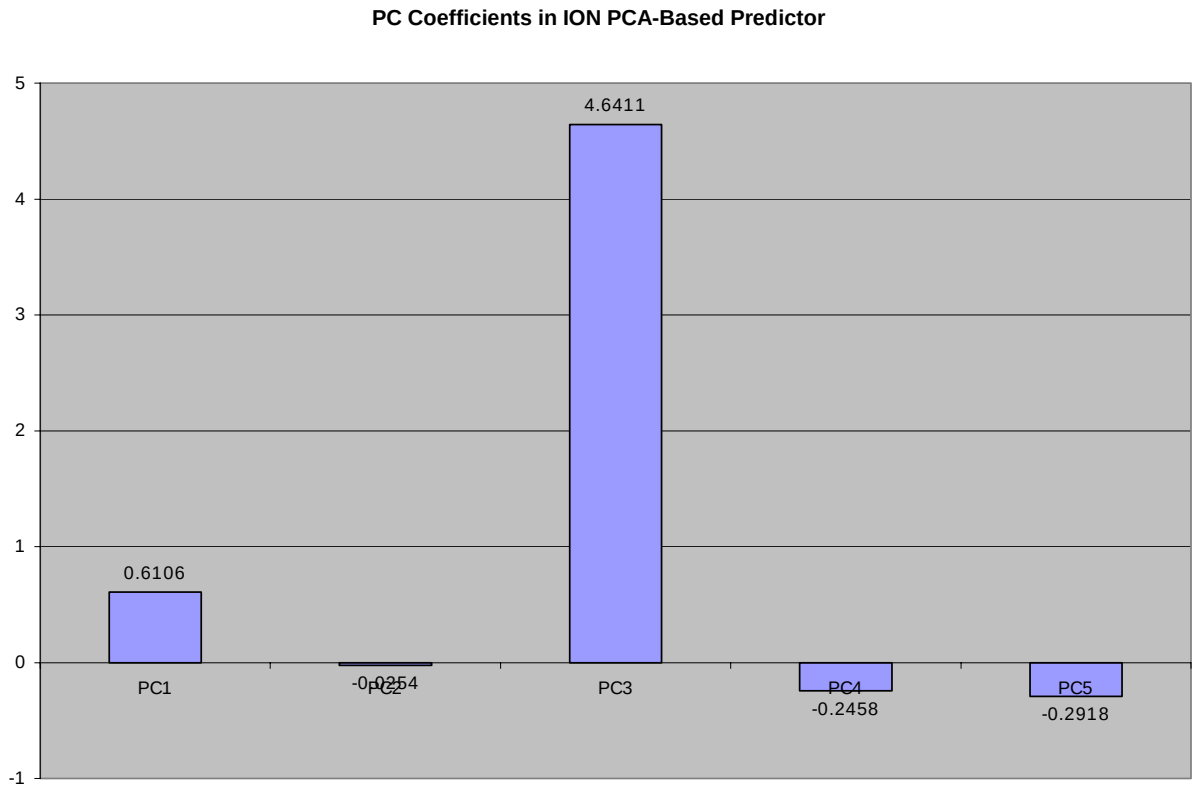
Figure 23



In the 'Mathematics / Engineering Models' section, three types of predictors were discussed: ION PCA-based predictor, static heart rate predictor, and heart rate average-based predictor. These three predictors each trained on the first 2/3 of the 46 patient history database and then tested on the remaining 1/3 (which they had not seen before). For the static heart rate predictor, the heart rate in the patient database is shifted 7 samples forward in time (2 min, 20 sec) to yield the predicted value. Thus, the present heart rate was predicted based on the HR value seen 7 samples ago. The heart rate average-based predictor takes the average of the current and last two samples (0, 20, and 40 s) of heart rate to make a prediction (i.e. N=3 in Equation 2). For the ION PCA-based predictor, the **A** and **B** constants are determined by linear regression. In addition to using the 5 significant PC's, shifted versions of these were also used in constructing the **X** described in the Equation 6. That is, not only was the present sample of each PC presented as input to the predictor, but so were some historical value of each PC (i.e. a value of a principal component U time units ago). Five values of each PC from 4 time samples back in time to the present sample were used here (U=0 to 4). Thus, the total number of variables (p in Equation 6) examined are: number of significant PC's * number of time shifts = 5 *5 = 25. Thus, there will be 25 constant coefficients (i.e. **A** will be 25 elements in length). Also, a constant B is determined by linear regression as well (89.30 is this case). The **A** coefficients give more weight to those principal components that are more useful in reducing the square error between predicted and actual future heart rate. Thus, insight can be gained on the importance of the various PC components. Figure 24 below shows the relative importance of each PC (inclusive of all 5 time shift variants) in ION PCA-based model. As expected, the heart rate and tachycardia-relevant information component, i.e. PC #3, was the most heavily weighted PC. The

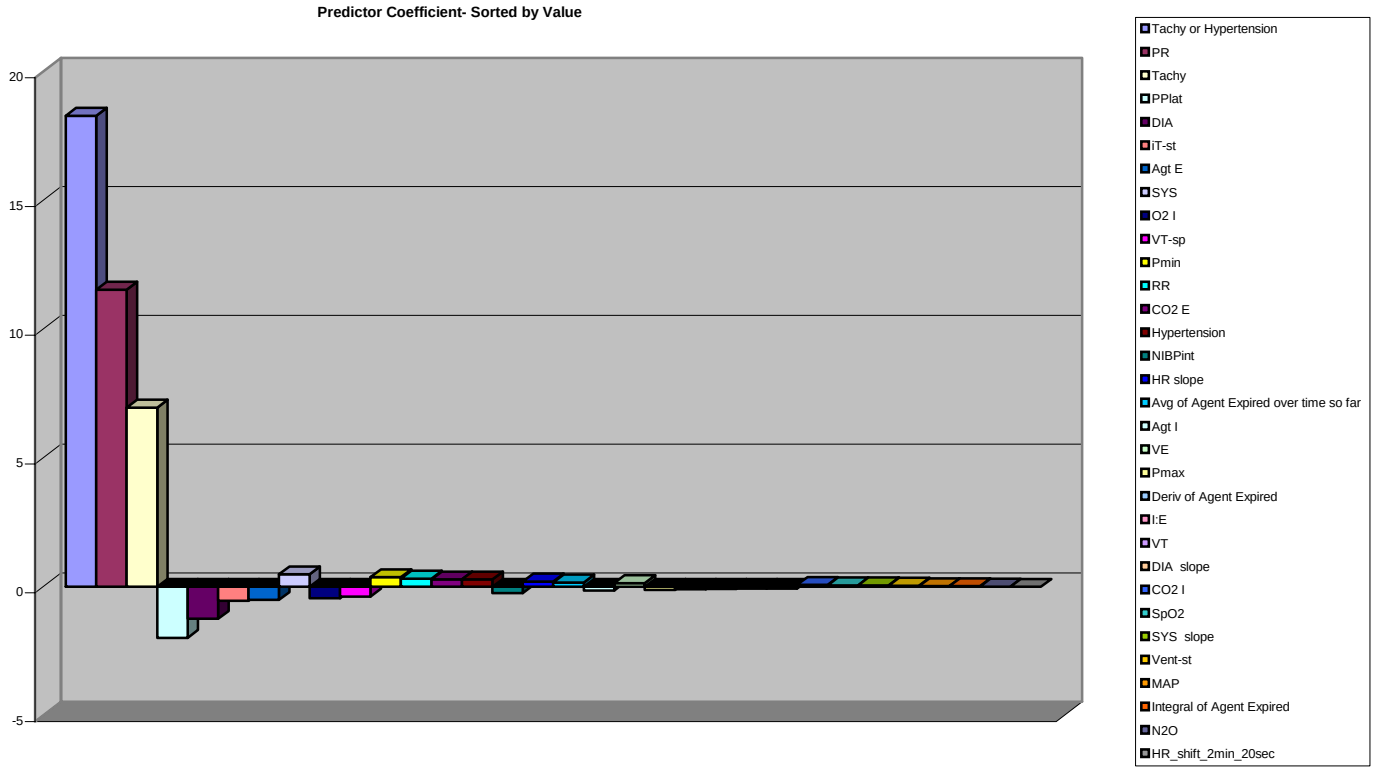
second most heavily weighted PC was PC #1, which includes mostly blood pressure and agent concentration information.

Figure 24



By looking at the parameters that comprise each PC, it is possible to gauge the relative weights of each of the 31 utilized fundamental and derived parameters in the principal component ION PCA-based model. Thirty-one parameters, rather than 32 were used since the future heart rate parameter information, PR_shift_7, was explicitly weighted at zero so that the predictor would not be able to see the actual future heart rate. The relative importance of the patient state parameters in the model are shown in Figure 25. It shows the value of the weight of each parameter sorted by the value's magnitude. Clearly, pulse rate (i.e. heart rate) is high on the list as is the presence of present tachycardia and hypertension. This suggests a possible link between hypertension and tachycardia.

Figure 25



Below (Figure 26) is a graph outlining the ION PCA-based predictor best fit based on the training data. The X's mark the prediction made by the model for the heart rate in the future. The circles denote the actual future heart rate values at the aforementioned points. In this case heart rate in BPM against time samples for all patients. Thus patient #1 goes from time samples 1-45 and patient #2 data is from samples 46-90, etc. Figure 27 shows the resulting residue plot.

Figure 26

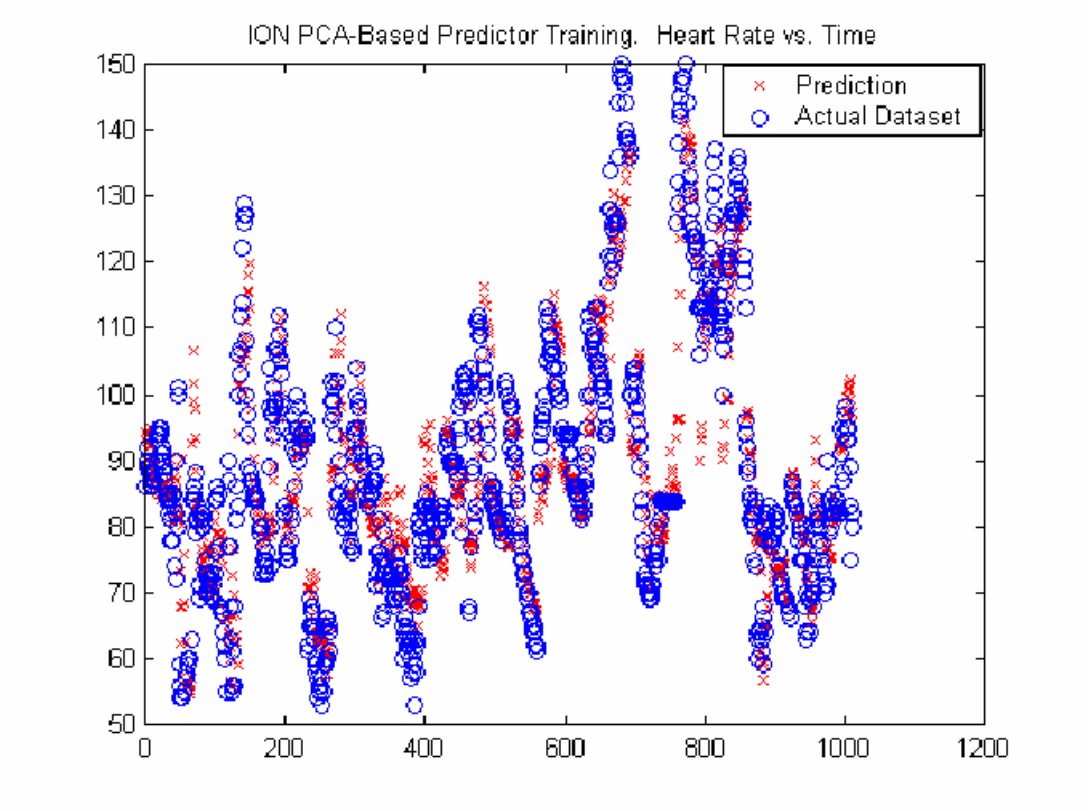
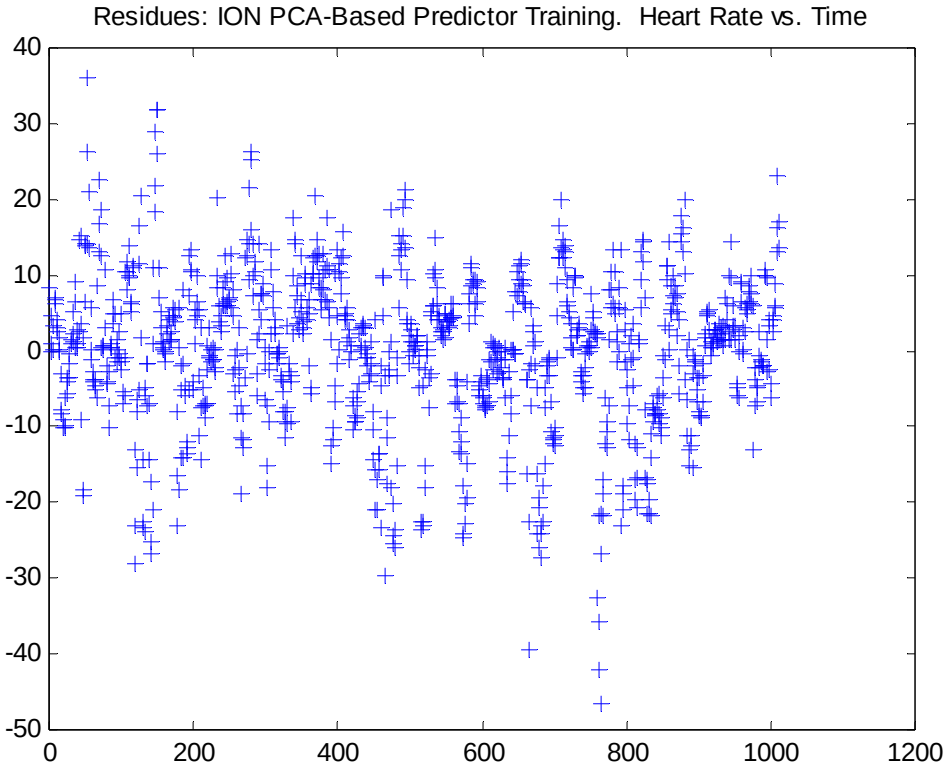


Figure 27



The models were then tested on the last third of the data (never seen previously) and asked to predict heart rate. The results are shown below, Figure 29 and Figure 30 show the plots for the ION PCA-based, static heart rate, and heart rate average-based predictors respectively. The residue plot for the ION PCA-based predictor is shown in Figure 31. The largest residues occur when the heart rate is changing quickly (i.e. magnitude of derivative of heart rate is large).

Figure 28

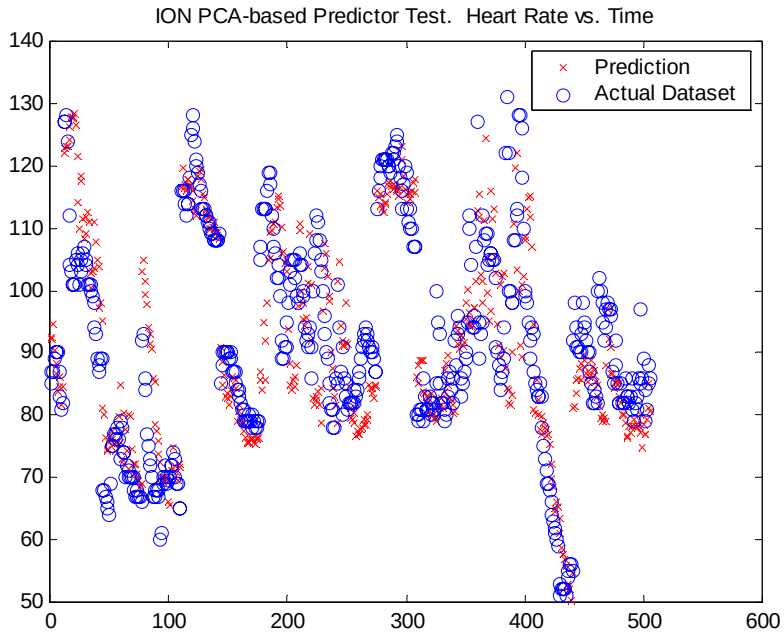


Figure 29

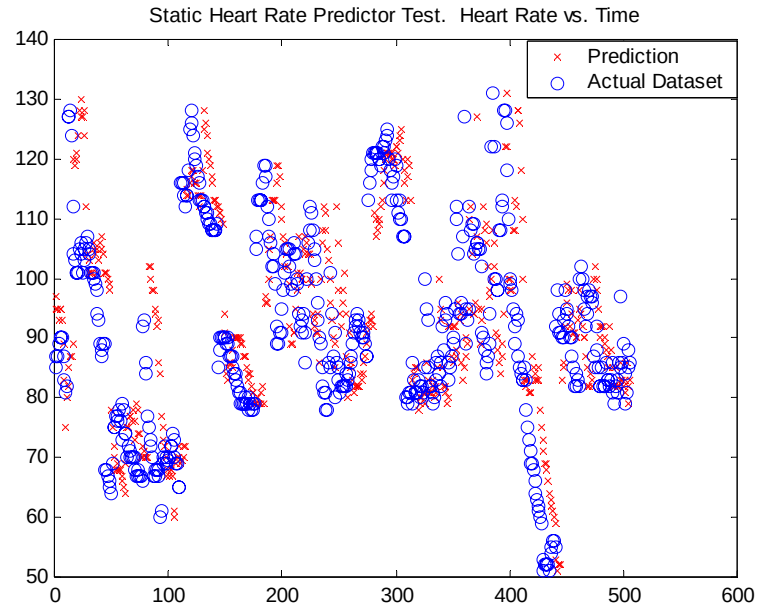


Figure 30

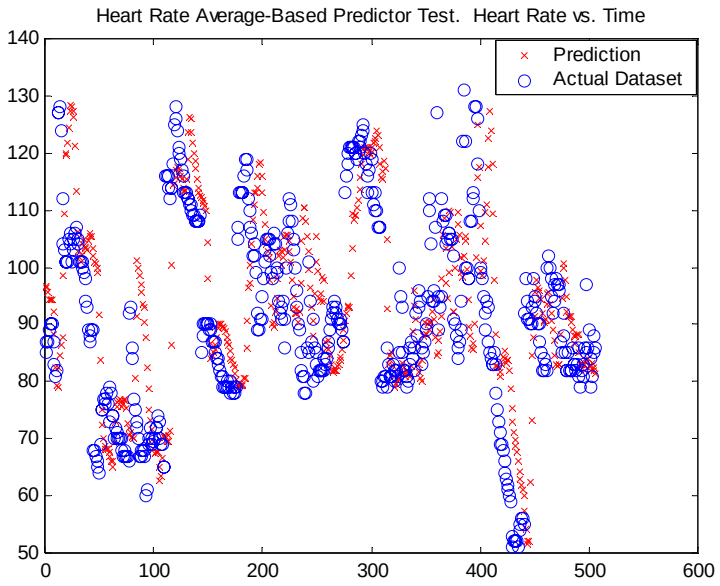
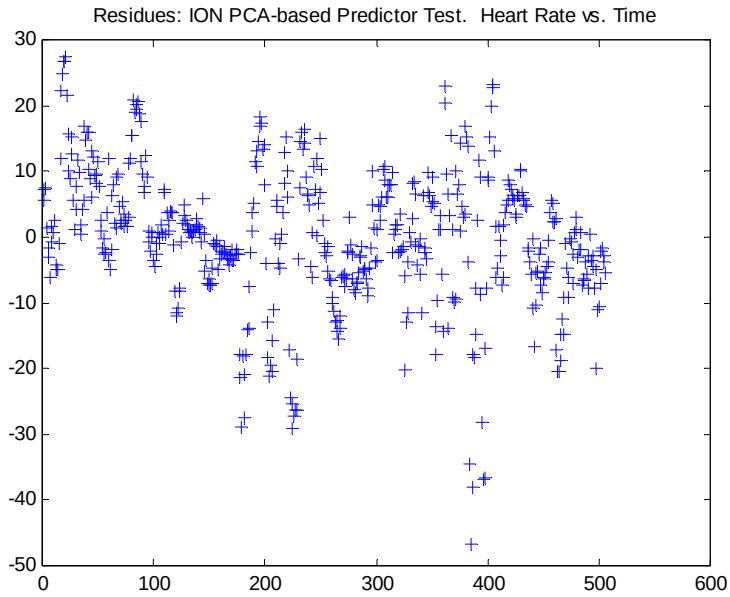


Figure 31



The accuracy of prediction was explored next. The RMS error for each of the three predictors outlined in this thesis is shown in Figure 32. The ION PCA-based predictor had the lowest error. However, it is important to note that the other predictors were quite successful in making predictions as well (see 'Conclusions' for further discussion).

Figure 32

Predictor Type	Absolute Values RMS Error	Residue Values RMS Error
ION PCA-Based Predictor	**10.44 BPM**	**10.44 BPM**
Static Heart Rate Predictor	**15.76 BPM**	**15.67 BPM**
Heart Rate Average-Based Predictor	**16.06 BPM**	**15.96 BPM**

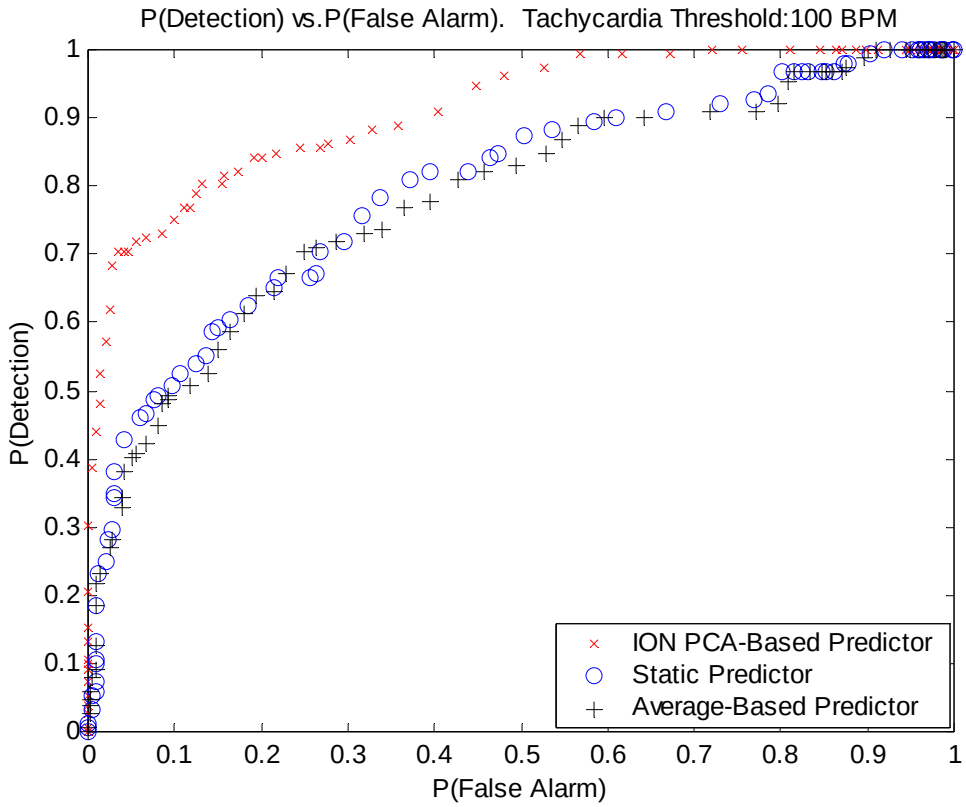
It is difficult to evaluate the clinical usefulness of tachycardia predictions from residues and RMS errors. What is needed is a method to determine how often tachycardia itself is predicted correctly. In addition, incorrect predictions of tachycardia must also be taken into account. Otherwise, a the trivial predictor below would always be the best performer:

Equation 8

Tachycardia ($n=n_1+n_0$) = True for n time samples.

To do this evaluation, Receiver Operating Characteristic (ROC) curves (probability of detection (Pd) vs. probability of false alarm (Pfa)) were derived for each of the three predictor types. As seen in Figure 33, the ION PCA-based predictor performed best while the static heart rate model was slightly better than heart rate average-based predictors for most Pd, Pfa possibilities.

Figure 33



The sensitivity of the predictors to a change in the tachycardia threshold was examined. Figure 34 and Figure 35 show new tachycardia thresholds set at 65 and 120 BPM arbitrarily. Since the upper left corner represents the most desired location (low false alarms, high detection), the 65 BPM is slightly better than the 100 BPM threshold while raising it to 120 BPM results in a larger in Pfa to achieve similar detection

Figure 34

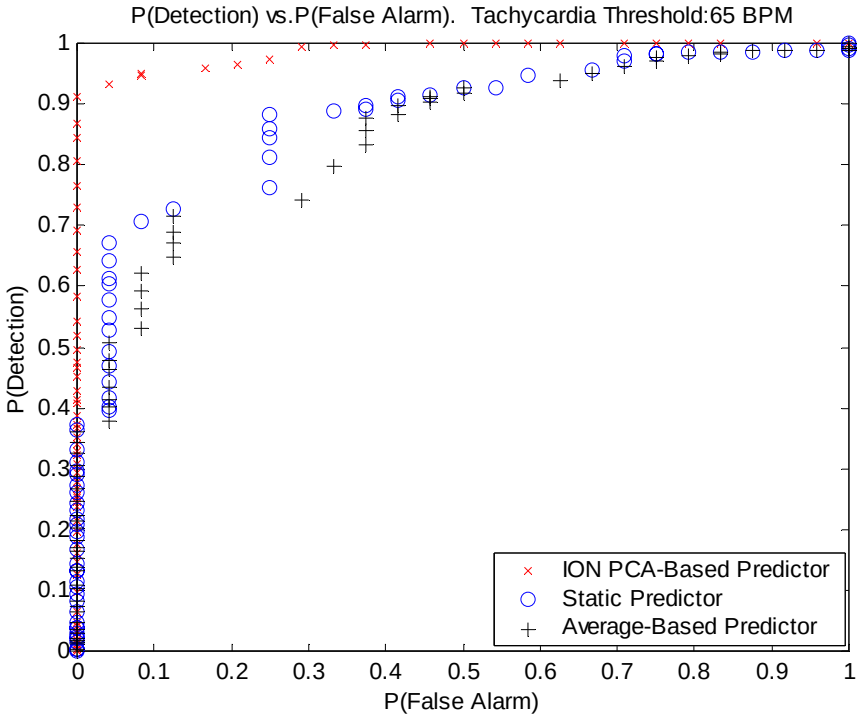
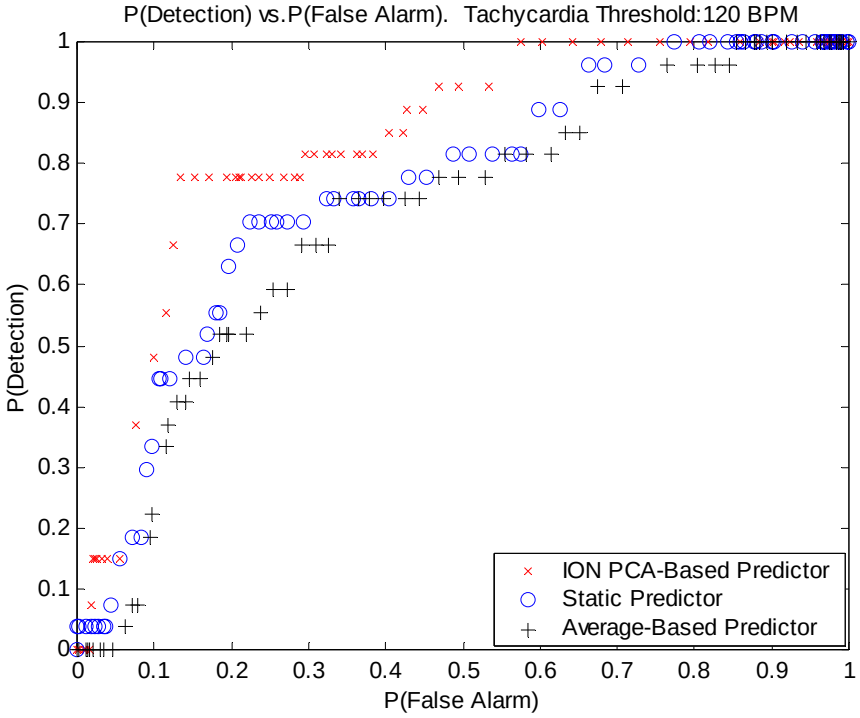
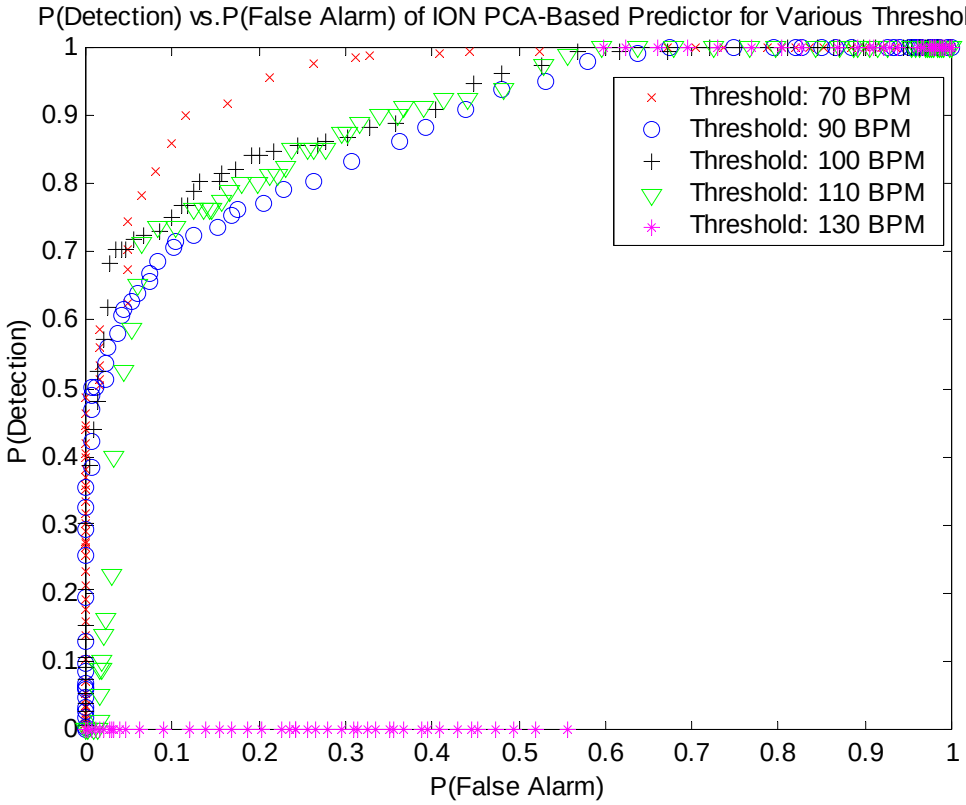


Figure 35



Lastly, to provide finer granularity, the threshold was varied from 70 to 130 BPM in steps (See Figure 36). Based on this figure, one could theorize a sub-tachycardia threshold of 90 BPM, for example, along with a super-tachycardia threshold of 110 BPM and expect to get similar ROC curves. The detection and false alarm probabilities depend on the training and testing data set ranges. Also, at lower thresholds, there are few points below the threshold and at higher thresholds, there are few points above it. So, predicting 65 BPM is easy since almost all predictions are above this point. Simply alerting tachycardia every time will yield a high detection and low false alarm rate. Thus, the training of the predictor depends on the input training dataset range that it sees and as well as its patterns. In addition, the limited-size testing dataset's range of values can affect the ROC curves. Yet, even given these constraints, it is still evident that the predictor remains within specific detection and false alarm ranges in the figure below (except for the 130 BPM pathological case, which had only one value above its threshold in the testing portion of the dataset).

Figure 36



7. CONCLUSIONS AND DISCUSSION

This thesis sought to find an effective way to quantify temporal patient state by encapsulating relevant variables processed by blind noise adjustment into the data set's significant principal components. In doing so, the validity and applicability of ION PCA-based methods was put to the test. The application of tachycardia prediction was selected due to the multivariate nature of the patient data, its ease of collection via anesthesia monitors and sensors in the OR, and the availability of the pre-existing patient data.

It was discovered that an ION PCA-based method could be used to find new information previously considered to be at the noise threshold. It was also found to delineate patient state in a simpler, more efficient manner. Since the ION processed Principal Components actually encode more information than the 32 fundamental and derived patient variables after estimated noise is removed, having to only monitor a reduced set of five more accurate variables could make patient monitoring more efficient, both for medical analysis and patient care. It also helps to provide information of the relevance and importance of the numerous statistics presented by patient monitors. While the basic meaning, physiologic significance, and clinical correlates of these principal components have been defined here (namely: PC #1 & #2: blood pressure/agent, PC#3: agent interaction: HR and BP, PC #4 & #5: agent interaction: BP and ventilation), further exploration into the relations and intuition behind these could be beneficial for clinicians in characterizing patients under anesthetic care.

With regard to the clinical application, it was discovered that desflurane led to ubiquitous tachycardia or hypertension (in all 46 patients) and both conditions in more than half of them. Thus, an interesting scientific question was: what action of the drug causes these short tachycardia and hypertension episodes during induction via the inhaled anesthetic. Was it simply related to the present or last few agent concentration values? Was it the slope of the agent concentration (how fast is was increased/decreased)? Could it be proportional to the total anesthetic given so far (represented via integral of agent expired)? As Figure 25 illustrated, the most significant of these variables in HR prediction were the present agent concentration and the average agent expired over time so far. Yet, in examining the data, it was found that most of the aforementioned fundamental and derived agent variables were in the noise (as in Figure 13) even after use of ION. Thus, a

definitive statement about the trait of agent linked to tachycardia can not be made conclusively at this time.

As discussed previously, several models for heart rate prediction were tested and compared via RMS error and ROC curves. Based on clinician discussions, one might consider Pfa = 1-Pd as an optimal operation point. The in case of the ION PCA-based predictor, the best of the three, this point would involve correct tachycardia detection approximately 82% of the time and a false alarm 18% of the time (see Figure 33).

Eventually, such a system could be put into clinical practice by integrating the 'Smart Alarm' ideas espoused by Philip, et. al.[13] by providing future derivatives in heart rate that could allow the alarms to provide early warning.

Several methods could be used to explore raising Pd and lowering Pfa. The monitors used had a sampling rate of 3 per minute. Though this was based on the best available commercial monitors with a recording feature, this rate could be improved slightly by manual transcription of parameter variables or by constructing a proprietary analog to digital converter to record the signals. In addition, exploration of scaling the patient database up might lead to an improved the model based on a larger training set. If a larger dataset is used, it would be expected that ION execution time as well as training time would increase.

Other models that take nonlinearity into account, such as neural networks, could also be used with the derived principal components as its input. The training of the weights of the nodes and biases would then correspond to the predictor model as **A** and **B** were for the linear regression model (See

Equation 6). It could open up avenues for future work into a number of clinical issues as well as serve as an additional tool for answering scientific questions.

Since heart rate plays such a major role in tachycardia prediction, it was expected that even the non-ION PCA predictors should do well in tachycardia prediction application since they capture the principal parameter of the most weighted PC. Perhaps testing this method on another patient monitoring application, which involves more observable variables relevant to the condition, would provide an additional means of separating between the different model strategies as well as a test of its validity and applicability in other domains.

With growing trend toward Internet appliances, it is conceivable that patients could be monitored remotely in the future. With such an influx of data from hundreds of out-patients, a given doctor would have a much larger time load per patient than today. Yet, if an intermediate middle-man involving a monitoring system that sorted out the significant patient state characteristics and paged the doctor when certain emergency or care necessitating combinations could potential occur, then dynamic out-patient monitoring could become a reality. While at first glance it might seem that such monitoring devices will not be available to consumers until a distant future, the first generation of such devices are already available. These include Casio's JP200W-1V Watch that can monitor heart rate or Boston start-up Fitsense's wearable FS-1, which can monitor everything from time, speed, distance, calories burn, and heart rate through several devices attached to the body. The FS-1 also includes a wireless network link to transmit the recorded information remotely (e.g. to person's home or even a doctor's office). Both of these are currently priced under $200 (i.e. within consumer market range) and may pave the way for future developments which might be more useful for clinical monitoring.

In the 'Introduction,' methods ranging from linear regression to neural network were proposed as possibilities for predicting heart rate, and hence, tachycardia. Another idea for further work would be to chain one predictor to another. Also, those mentioned in the 'Clinical Background' were completely deterministic in nature. Tachycardia might be completely predictable if every variable were known, but then again, so would a coin toss if all air flow patterns and models of the coin were perfectly modeled. Therefore, probability might be useful here. It could also be used to provide the clinician with a gauge of the perceived certainty/confidence level of the prediction.

One option that incorporates these ideas is the combination of a neural network front end for nonlinear prediction with a Markov Chain. The neural network could be used to calculate the probabilities of entering a different state in the Markov chain. The Markov chain need not be limited to two nodes (Tachycardia and No Tachycardia). In fact, by adding additional nodes, a history of patient state (like the time-shifted ones used in the linear regression model) can be used to capture tachycardia information between 'Tachycardia' and 'No Tachycardia.'

Another area for future exploration involves designing a framework for dynamic, time varying principal components and noise estimation as well as PC's and noise estimators with specific patient or patient cluster dependence. Thus, the resultant ION PCA-based principal components could be altered for different phases of an operation (just as an example) to account for the differences between IV induction, induction by inhaled gas, wake-up, post-operative care, etc. In fact, one could use the aforementioned methods to cluster patient populations based on principal components consisting of various relatively static patient variable such as sex, weight, height, medical history, etc. Hopefully, the future work will be able to answer some of these questions as well and open up further avenues exploration.

8. REFERENCES

[1] Lee, Junehee. Blind Noise Estimation and Compensation for Improved Characterization of Multivariate Processes, (PhD Thesis), Massachusetts Institute of Technology, 2000.

[2] Lee, Junehee and D H Staelin. Iterative Signal-Order and Noise Estimation for Multivariate Data, *IEE* Letters, (Accepted for publication, 2001).

[3] Joliffe, I. *Principal Component Analysis*. New York, Springer-Verlag, 1986.

[4] Hargasser S, Hipp R, Breinbauer B, Mielke L, Entholzner E, Rust M. A lower solubility recommends the use of desflurane more than isoflurane, halothane, and enflurane under low-flow conditions. *J Clin Anesth* (1995 Feb) 7(1):49-53.

[5] Philip B K, Lombard L L, Road E R, Drager L R, Calalang I, Philip J H. Comparison of Vital Capacity Induction with Sevoflurane to Intravenous Induction with Propofol for Adult Ambulatory Anesthesia. *Anes Analg* 1999; 89:623-627.

[6] Philip, James H. How to make Desflurane work for you - Use its low solubility, (Technical Paper), Brigham and Women's Hospital, 2000.

[7] Philip, J. H. and Raemer, D. B. Selecting the Optimal Anesthesia Monitoring Array. *Medical Instrumentation*, Vol 19, No. 3, pp. 122-126, 1985.

[8] Weinstein, E., Oppenheim, A., Feder, M., and Buck, R. "Iterative and Sequential Algorithms for Multi-sensor Signal Enhancement," *IEEE Transactions On Signal Processing*, Vol. 42, no. 4., pp. 846-859, 1994.

[9] Rawizza, Mark A. Time-Series Analysis of Multivariate Manufacturing Data Sets, S.M. Thesis, Massachusetts Institute of Technology, 1996.

[10] Miller R D, Lichtor J L. *Atlas of Anesthesia. Preoperative Preparation and Intraoperative Monitoring.* Philadelphia, Churchill Livingstone, Vol III, 1997, Ch.

[11] Muzi M Ebert TJ A comparison of baroreflex sensitivity during isoflurane and desflurane anesthesia in humans. *Anesthesiology*, 82:919-25, 1995.

[12] Ebert TJ, Muzi M. Sympathetic hyperactivity during desflurane anesthesia in healthy volunteers. A comparison with isoflurane. *Anesthesiology*, 79, 444-53.

[13] Philip J. H, Scott, D., Topulos G.P., Calalang I, Raemer D.B. Development of a Monitoring-for-Change Algorithm for a New Anesthesia Machine. ASA, Las Vegas, NV, Oct. 1990.

9. LIST OF PERTINENT ACRONYMS AND ABBREVIATIONS (CLINICAL & TECHNICAL)

Figure 37

Clinical Term	Acronym/Definition
OR	Operating Room
Tachycardia	Heart Rate > 100 beats per minute
VRG	Vessel Rich Group, comprised of tissue well perfused with blood, as opposed to muscle (intermediate) and fat (low).
Hypertension	Systolic Blood Pressure > 140, Diastolic Pressure > 90
Solubility, Blood	How easily a substance dissolves in blood
Desflurane	Used as an anesthetic in its vapor form
IV	Intravenously
CNS	Central Nervous System- Brain and Spinal cord, as opposed to PNS: Peripheral Nervous System
BPM	Beats per minute
Breathing Circuit	An apparatus that connects the patient to the agent as well as the mechanical ventilator.
MABP Mean Arterial Blood Pressure	Defined as: 1/3*Systolic BP + 2/3*Diastolic BP
Induction Intravenously	Administration of a drug intravenously to induce an unconscious and/or anesthetized patient state.
Induction of Inhaled Gas	Administration of a drug intravenously to induce an unconscious and/or anesthetized patient state.
Intubation	Placing the breathing circuit in the airway (throat).

Figure 38

Mathematical/Engineering	Acronym/Definition
ION	Iterative Order and Noise Estimate
SNR	Signal to Noise Ratio
RAM	Random Access Memory
NAPC	Noise Adjusted Principal Component Analysis
BAPC	Blind Adjusted Principal Component Analysis
PC	Principal Component
PCA	Principal Component Analysis
EM	Expectation-Maximization

10. APPENDIX: SOURCE CODE LISTING

10.1. Selected Files Listings

FILE 1: PCA_DATA.M. 69
This is the main program file. This program loads the preprocessed patient records. It calculates the derived parameters. It then sequentially assembles the patients across time. Next, it does PCA and ION. Lastly, the coefficients of the linear predictor are calculated.

FILE 2: PCA_FUNCT.M 101
This function performs the mechanics of PCA.

FILE 3: PRE_PROCESSING.M. 102
This file does the preprocessing of the raw patient data files. It locates the start of valid data, does linear interpolation, and determines the starting point for the induction of inhaled desflurane.

File 1: Pca_data.m

```
function pca_data(num_pat_id)
%% need to re-load ION values if number of parameters, etc. changes...



%  all_data.sel_npdata(:, :) = 25xnum_time_col*num_real_pat = 1 patient's
'imagemap'
% all_data.pdata = arr_all_pat.pdata (truncated)


PARAM_BP1 = 1
PARAM_BP2 = 2
PARAM_BPM = 3
PARAM_HR = 5 % - by pat file col.
PARAM_AGTE = 19  % - by pat file col.
PARAM_HR_SHIFT_m240 = 31;   % 'PR_shift_m8 (BPM)' at 34 - by pat file col.
PARAM_HR_SHIFT_m240_mod = 23;
PARAM_DERIV_HR = 55

PARAM_HR_MOD = 4;   % 'PR' at 5

%10/25- changed to have only 180's in data set (plus adjusted all array
values->num_order found = 2 by ION -> much better!


%rerun using this program- added cols 6->7
% ion->norm by noise & graph
% add pca func
% do linear model
% fix up func. make it exp. based parameter meta-func (like Jay's prog.)
% ana pca eig value strengths, compare to hr.
% ana pca pc1 vs pc2 grpahs, etc.  write down patterns.



% compare two vectors to see if any elements are dif.:
% find((all_data.pdata(:,:)-all_data.npdata(:,:)) == 0)


      addpath 'c:\users\gil\matlab\new'


addpath 'c:\Users\gil\matlab\jhlee\IonDistribution'
addpath 'c:\Users\gil\matlab\jhlee\IonDistribution'

ID_DIR=7;


if (computer == 'PCWIN')
```

```
%if (exist('c:\users\gil') == ID_DIR)
%     computer_subtype='WIN98_JANSKY'
%     str_drive ='f'
%else
%     computer_subtype='WIN95_JANSKY'
%     str_drive ='e'
%end

% assume on 'c'

str_drive = 'c'

% \CD Data 07%2F06%2F00 OR 19
      %%%%%%%%%%%%%%%%
      % PCWIN
      % 1 = PCWIN, 2 = UNIX
      OS.TYPE = 1
      OS.SEP = '\'
      root_path = strcat(str_drive,':\users\gil')
      rel_path = 'bwh_data\CD Data 07%2F00'
      rel_path = 'bwh_data\CD Data 1999-1'
      rel_path = 'examples'
      rel_path = 'test'

      % rel_path = 'bwh_data'
      log_path = strcat(str_drive,':\users\gil\matlab\new\log')

      %%%%%%%%%%%%%%%%
else
      %%%%%%%%%%%%%%%%
      % PCWIN
      % 1 = PCWIN, 2 = UNIX
      OS.TYPE = 2
      OS.SEP = '/'
      root_path = '/usr/users/gil/matlab/'
      rel_path = 'bwh_data'
      % rel_path = 'bwh_data'
      log_path = '/usr/users/gil/matlab/log'
      %%%%%%%%%%%%%%%%

end




global PARAM_LINE_START
global PARAM_LINE_STOP
more off

PARAM_MAX_LINE = 1;
PARAM_OS_TYPE = 2;
PARAM_SERIAL = 3;
```

```
PARAM_GOOD_PAT = 4;
PARAM_LINE_START = 5;
PARAM_LINE_STOP = 6;
PARAM_AGTI_START = 7;
SAMP_PER_MIN = 3;


MAX_NUM_PARAM = 25;
% fil_load =
strcat('c:\users\gil\matlab\new\log\patient_post_',num2str(num_pat_id))
 fil_load =
strcat('c:\users\gil\matlab\new\log\patient_post_',num2str(num_pat_id))

load (fil_load,'arr_all_pat')




NUM_PAT_COL = 180;
%{
idx_idx_pat = find(arr_all_pat.misc(:, PARAM_MAX_LINE) < NUM_PAT_COL*1.1);


arr_tmp1 = arr_all_pat.pdata(idx_idx_pat,:,:);
arr_tmp2 = arr_all_pat.misc(idx_idx_pat,:);
arr_tmp3 = arr_all_pat.pname(idx_idx_pat);


arr_all_pat.pdata = arr_tmp1;
arr_all_pat.misc = arr_tmp2;
arr_all_pat.pname = arr_tmp3;

%}


% num_shift = SAMP_PER_MIN * 2; 3 per min * 3 min. to shift

% Make next 7*3 = 21 parameters be shifts (param 26 -> 21+25=46).
% before modification- so use orginal values for paramters' columns!!!


real_num_pat =  size(arr_all_pat.pdata,1);
idx_pat=[1:real_num_pat];


num_time_col = size(arr_all_pat.pdata,3);       % not all are valid- since
table makes it for largest patient's time points.  also, need to take
start/stop into account for which time points are valid for a given patient.

for (num_shift=SAMP_PER_MIN * 2:1:SAMP_PER_MIN * 4)


      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, 1:num_time_col -num_shift)
= arr_all_pat.pdata(idx_pat, PARAM_BP1, num_shift+1:num_time_col );
```

```
%     arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, num_shift+1:num_time_col) =
arr_all_pat.pdata(idx_pat, PARAM_BP1, num_time_col -num_shift ) *
ones(1,num_time_col-num_shift);



      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+2, 1:num_time_col -num_shift)
= arr_all_pat.pdata(idx_pat, PARAM_BP2, num_shift+1:num_time_col );
%     arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+2, num_shift+1:num_time_col) =
arr_all_pat.pdata(idx_pat, PARAM_BP2, num_time_col -num_shift ) *
ones(1,num_time_col-num_shift);


      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+3, 1:num_time_col -num_shift)
= arr_all_pat.pdata(idx_pat, PARAM_HR, num_shift+1:num_time_col );
%     arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+3, num_shift+1:num_time_col) =
arr_all_pat.pdata(idx_pat, PARAM_HR, num_time_col -num_shift ) *
ones(1,num_time_col-num_shift);



MAX_NUM_PARAM = MAX_NUM_PARAM+3;

end




% Make next parameter 47 be agt expired slope = [diff(agte) / (1/3 = sampling
time)] = diff*3;
for(idx_pat=1:real_num_pat)


arr_diff(1:size(diff(arr_all_pat.pdata(idx_pat, PARAM_AGTE, :)),3) ) =
diff(arr_all_pat.pdata(idx_pat, PARAM_AGTE, :)) * SAMP_PER_MIN;

% make all transitions from one to another be 0 (i.e. diff for
[NUMP_SAMP*n+1] -> 0)
      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, :) = [0,arr_diff];

arr_diff=''
end

MAX_NUM_PARAM = MAX_NUM_PARAM+1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Make next parameter 48 be agt expired integral = [sum(agte (1:cur_time)) /
(3 = sampling time)] = units: [Agt E %*min.];
for(idx_pat=1:real_num_pat)

for(idx_data=1:length(arr_all_pat.pdata(idx_pat, PARAM_AGTE, :))  )


arr_sum = arr_all_pat.pdata(idx_pat, PARAM_AGTE, 1:idx_data);

arr_sum(  find(isnan(arr_sum))  ) = 0;
```

```
      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, idx_data) = sum(arr_sum)/3;

end

end
MAX_NUM_PARAM = MAX_NUM_PARAM+1;


% Make next parameter 49 be agt expired avg = [sum(agte (1:cur_time)) /
(number samples)] = units: [Agt E %];
for(idx_pat=1:real_num_pat)

for(idx_data=1:length(arr_all_pat.pdata(idx_pat, PARAM_AGTE, :))  )


arr_sum = arr_all_pat.pdata(idx_pat, PARAM_AGTE, 1:idx_data);

arr_sum(  find(isnan(arr_sum))  ) = 0;


      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, idx_data) =
sum(arr_sum)/length(arr_sum);

end

end
MAX_NUM_PARAM = MAX_NUM_PARAM+1;




% Make next parameter 50 be high hr 'alert signal'


%keyboard

for(idx_pat=1:real_num_pat)
      for(idx_data=1:num_time_col)
            arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, idx_data) =
arr_all_pat.pdata(idx_pat, PARAM_HR, idx_data) > 100;
      end
end

MAX_NUM_PARAM = MAX_NUM_PARAM+1;

% Make next parameter 51 be high bp 'alert signal'


%keyboard

for(idx_pat=1:real_num_pat)
      for(idx_data=1:num_time_col)
            arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, idx_data) =
arr_all_pat.pdata(idx_pat, PARAM_BP1, idx_data) > 140 |
arr_all_pat.pdata(idx_pat, PARAM_BP2, idx_data) > 90;
```

```
      end
end

MAX_NUM_PARAM = MAX_NUM_PARAM+1;

% Make next parameter 52 be high hr/bp 'alert signal'


%keyboard

for(idx_pat=1:real_num_pat)
      for(idx_data=1:num_time_col)
            arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, idx_data) =
arr_all_pat.pdata(idx_pat, PARAM_HR, idx_data) > 100 |
arr_all_pat.pdata(idx_pat, PARAM_BP1, idx_data) > 140 |
arr_all_pat.pdata(idx_pat, PARAM_BP2, idx_data) > 90;
      end
end

MAX_NUM_PARAM = MAX_NUM_PARAM+1;




% Make next parameter 53 be diff sys  slope = [diff(sys) / (1/3 = sampling
time)] = diff*3;
for(idx_pat=1:real_num_pat)


arr_diff(1:size(diff(arr_all_pat.pdata(idx_pat, PARAM_BP1, :)),3) ) =
diff(arr_all_pat.pdata(idx_pat, PARAM_BP1, :)) * SAMP_PER_MIN;

% make all transitions from one to another be 0 (i.e. diff for
[NUMP_SAMP*n+1] -> 0)
      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, :) = [0,arr_diff];

arr_diff=''
end

MAX_NUM_PARAM = MAX_NUM_PARAM+1;

% Make next parameter 54 be diff dia  slope = [diff(sys) / (1/3 = sampling
time)] = diff*3;
for(idx_pat=1:real_num_pat)


arr_diff(1:size(diff(arr_all_pat.pdata(idx_pat, PARAM_BP2, :)),3) ) =
diff(arr_all_pat.pdata(idx_pat, PARAM_BP2, :)) * SAMP_PER_MIN;

% make all transitions from one to another be 0 (i.e. diff for
[NUMP_SAMP*n+1] -> 0)
      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, :) = [0,arr_diff];

arr_diff=''
end

MAX_NUM_PARAM = MAX_NUM_PARAM+1;
```

```
% Make next parameter 55 be diff hr  slope = [diff(sys) / (1/3 = sampling
time)] = diff*3;
for(idx_pat=1:real_num_pat)


arr_diff(1:size(diff(arr_all_pat.pdata(idx_pat, PARAM_HR, :)),3) ) =
diff(arr_all_pat.pdata(idx_pat, PARAM_HR, :)) * SAMP_PER_MIN;

% make all transitions from one to another be 0 (i.e. diff for
[NUMP_SAMP*n+1] -> 0)
      arr_all_pat.pdata(idx_pat, MAX_NUM_PARAM+1, :) = [0,arr_diff];

arr_diff=''
end

MAX_NUM_PARAM = MAX_NUM_PARAM+1;









%keyboard
num_pat = size(arr_all_pat.pdata, 1);

% arr_all_pat.pdata( 1:size(arr_all_pat.pdata, 1), 1:size(arr_all_pat.pdata,
2), size(arr_all_pat.pdata, 3) )=0;






idx_length = 0
for idx_pat=1:num_pat
% keyboard
%     line_start = arr_all_pat.misc(idx_pat, PARAM_LINE_START);
      line_start = arr_all_pat.misc(idx_pat, PARAM_AGTI_START);
NUM_SAMP = SAMP_PER_MIN * 15; % = 3/min * 15 min.
      line_stop = min( arr_all_pat.misc(idx_pat, PARAM_LINE_STOP),
arr_all_pat.misc(idx_pat, PARAM_AGTI_START) + NUM_SAMP - 1);


      idx_length = idx_length + (line_stop - line_start + 1)
      idx_length_pat(idx_pat) = (line_stop - line_start + 1);
```

```
end

%all_data.pdata(1:MAX_NUM_PARAM, 1:idx_length) = NaN;
idx_length









%%%%%%%%%%%%%%%%%%%%%%%%%%%

NUM_COL = 256

max_len = -1
cur_spot = 1
for idx_pat=1:num_pat



      len_start_stop = line_stop - line_start + 1;

for (i=1:len_start_stop)
      hsv_col_map_tim(cur_spot + i-1,:) = (([(i*NUM_COL/len_start_stop),
(NUM_COL*.9), (NUM_COL*2/3)]/NUM_COL)' * ones(1, 1 ))';
end




if (max_len < (line_stop-line_start + 1) )
      max_len = (line_stop-line_start + 1);

      hsv_col_map_tim_bar(1:max_len, :) = hsv_col_map_tim(cur_spot:(cur_spot+
max_len-1),:);

end

      cur_spot = cur_spot + line_stop-line_start+1;

end







for idx_param=1:MAX_NUM_PARAM
idx_spot = 1;
arr_spot(idx_spot) = 1;

      idx_pat=1;
```

```
%     line_start = arr_all_pat.misc(idx_pat, PARAM_LINE_START);
      line_start = arr_all_pat.misc(idx_pat, PARAM_AGTI_START);

      line_stop = min( arr_all_pat.misc(idx_pat, PARAM_LINE_STOP),
arr_all_pat.misc(idx_pat, PARAM_AGTI_START) + NUM_SAMP -1);




      all_data.pdata(idx_param, 1:line_stop-line_start+1) =
arr_all_pat.pdata(1,idx_param,line_start:line_stop);

      arr_spot(idx_spot+1) = arr_spot(idx_spot) + line_stop-line_start+1;
idx_spot = idx_spot+1;

%%keyboard



      if (num_pat>1)
            for idx_pat=2:num_pat
%                 line_start = arr_all_pat.misc(idx_pat, PARAM_LINE_START);
                  line_start = arr_all_pat.misc(idx_pat, PARAM_AGTI_START);

                  line_stop = min( arr_all_pat.misc(idx_pat,
PARAM_LINE_STOP), arr_all_pat.misc(idx_pat, PARAM_AGTI_START) + NUM_SAMP -1);




                  aa= all_data.pdata(idx_param,:);
                  bb(1:line_stop-line_start+1) =
arr_all_pat.pdata(idx_pat,idx_param,line_start:line_stop);



length([aa,bb ]);
%%keyboard
size(bb);
 (line_stop-line_start)+1 + arr_spot(idx_spot);




                  all_data.pdata(idx_param,
arr_spot(idx_spot):(arr_spot(idx_spot) + length(bb)-1)  ) = bb ;
%keyboard
bb=0;
      arr_spot(idx_spot+1) = arr_spot(idx_spot) + line_stop-line_start+1;
idx_spot = idx_spot+1;
            end

      end
end
```

```
display('stop here!')
%%%%keyboard









for idx_param=1:MAX_NUM_PARAM



num_min = min(min(all_data.pdata(idx_param, :) ))
if isnan(num_min)
      num_min  = -1;

end


% get one parameter's whole vector across files:
b(1:length(all_data.pdata(idx_param, :)) )=all_data.pdata(idx_param, :);

b(find(isnan(  b  ))) = num_min;
      all_data.pdata(idx_param, :) = b;


num_std = std(all_data.pdata(idx_param, :) );
all_data.npdata(idx_param, :) = norm_funct(all_data.pdata(idx_param, :),
num_std )


end

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^

% [npdata, pmean, pstd] = prestd(all_data.pdata(idx_param,:) );

%%keyboard
%all_data.pmean(idx_param) = pmean;

%all_data.pstd(idx_param) = pstd;



% isnan(all_data.npdata(:,:))
% change nan's!!!

all_data.npdata (find(isnan(all_data.npdata(:,:)))) = 0
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 file_save = strcat(log_path, OS.SEP, 'patient_pca_',num2str(num_pat))



% load(file_save, 'arr_all_pat') ;
save (file_save, 'all_data', 'arr_all_pat') ;
%%
%keyboard
%     save file_save arr_all_pat.pdata(:,:,:) -ASCII ;

cd (log_path)







%&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

PARAM_HR_HI = 50 % by pat col no.
PARAM_BP_HI = 51 % by pat col no.
PARAM_HR_BP_HI = 52 % by pat col no.

idx_pat=1
%%for (idx_pat=1:real_num_pat)

%if (any(arr_all_pat.pdata(idx_pat,48,:)>0))
%display('Yes')
%else
%display('Nope')
%end



        fid = fopen(strcat('exp', num2str(idx_pat),'.txt'),'w');
for(idx_param=1:MAX_NUM_PARAM)
 fprintf(fid,'Param #%d',idx_param);
      for(idx_data=1:num_time_col)
              fprintf(fid,'%6.2f  ', arr_all_pat.pdata(idx_pat, idx_param,
idx_data),  arr_all_pat.pdata(1, idx_param, idx_data));
      end
 fprintf(fid,'\n\n');
end
        fclose(fid);
%%end

%%%%keyboard



      str_title = strcat('Number of Patients: ', num2str(num_pat), '  ')
      str_x = 'Samples';
      str_y = strcat(' Param 1-6, Patient #', num2str(idx_pat) );

      figure
```

```
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ',
str_y, ' vs. ', str_x ))


      for(idx_param=[1,2,3,4,5,6])
            subplot(7, 1, idx_param), plot(  1:idx_length_pat(1)   ,
all_data.pdata(idx_param,1:idx_length_pat(1) ) )
      end
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ',
str_y, ' vs. ', str_x ))


%     figure
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ',
str_y, ' vs. ', str_x ))


      for(idx_param=[1,2,3,4,5,6])
            subplot(7, 1, idx_param), plot(  1:idx_length_pat(2)  ,
all_data.pdata(idx_param,idx_length_pat(1)+1:idx_length_pat(1) +
idx_length_pat(2) ) )
      end
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ',
str_y, ' vs. ', str_x ))


%     figure
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ',
str_y, ' vs. ', str_x ))


      for(idx_param=[1,2,3,4,5,6])
            subplot(7, 1, idx_param), plot(  1:idx_length_pat(3)  ,
all_data.pdata(idx_param,idx_length_pat(1) +
idx_length_pat(2)+1:idx_length_pat(1) + idx_length_pat(2) + idx_length_pat(3)
) )
      end
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ',
str_y, ' vs. ', str_x ))









%     figure
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ',
str_y, ' vs. ', str_x ))


      for(idx_param=[1,2,3,4,5,6])
```

```
            subplot(7, 1, idx_param), plot(  1:idx_length_pat(4)  , 
all_data.pdata(idx_param,idx_length_pat(1) + idx_length_pat(2) + 
idx_length_pat(3)+1:idx_length_pat(1) + idx_length_pat(2) + idx_length_pat(3) 
+ idx_length_pat(4) ) )
      end
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', 
str_y, ' vs. ', str_x ))




%     figure
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', 
str_y, ' vs. ', str_x ))


      for(idx_param=[1,2,3,4,5,6])
            subplot(7, 1, idx_param), plot(  1:idx_length_pat(5)  , 
all_data.pdata(idx_param,idx_length_pat(1) + idx_length_pat(2) + 
idx_length_pat(3)+idx_length_pat(4)+1:idx_length_pat(1) + idx_length_pat(2) + 
idx_length_pat(3) + idx_length_pat(4) + idx_length_pat(5)) )
      end
      Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', 
str_y, ' vs. ', str_x ))


%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%b = [all_data.npdata(1:3,:) ; all_data.npdata(5,:) ; 
all_data.npdata(8:19,:) ; all_data.npdata(21:24,:) ];
keyboard


%b = [all_data.npdata(1:3,:); all_data.npdata(5:19,:) ; 
all_data.npdata(21:24,:); all_data.npdata(26:MAX_NUM_PARAM,:)];
b = [all_data.npdata(1:3,:); all_data.npdata(5:19,:) ; 
all_data.npdata(21:24,:); all_data.npdata(PARAM_HR_SHIFT_m240,:); 
all_data.npdata(47:MAX_NUM_PARAM,:)];
bp=[all_data.pdata(1:3,:); all_data.pdata(5:19,:) ; all_data.pdata(21:24,:); 
all_data.pdata(PARAM_HR_SHIFT_m240,:); all_data.pdata(47:MAX_NUM_PARAM,:)];




% removed shifted versions above...

all_data.sel_npdata(:,:) = b(:,:);
all_data.sel_pdata(:, :) = bp(:,:);
num_sel_param = size(all_data.sel_npdata(:,:), 1);



      figure
%     imagesc(all_data.npdata(:,:))
```

```
      imagesc(all_data.sel_npdata(:,:))
Title(strcat('Image Map of Dataset: ',num2str(idx_pat),' Patients.  Paramters
vs. Time'))
xlabel('Time')
ylabel('Paramters')
      colorbar



      imagesc(all_data.sel_npdata(:,1:135))
Title(strcat('Image Map of 3 Patients After Pre-Processing/Interpolation.
Paramters vs. Time'))
xlabel('Time')
ylabel('Paramters')
      colorbar

%%keyboard



arr_del=PARAM_HR_SHIFT_m240_mod;
[num_evec_1, x, num_scr_plot] = pca_funct(all_data.sel_npdata, arr_del);


x_pca = x;




display('22')

%%%%keyboard
for (i=1:7)
[num_evec_1_srt, num_evec_1_idx] = sort(abs(num_evec_1(:,i)));
num_evec_1_srt_des = flipud(num_evec_1_srt);
num_evec_1_idx_des = flipud(num_evec_1_idx);

str_msg = sprintf('Eig_vec: #%d, max idx: %d %d %d %d %d, max. values: %f %f
%f %f %f\n', i, num_evec_1_idx_des(1), num_evec_1_idx_des(2),
num_evec_1_idx_des(3), num_evec_1_idx_des(4), num_evec_1_idx_des(5),
num_evec_1_srt_des(1), num_evec_1_srt_des(2), num_evec_1_srt_des(3),
num_evec_1_srt_des(4), num_evec_1_srt_des(5))



      arr_param_name=make_arr_param_name;




      % NOTE: strcat = trick to get cells to be converted to strings!!


      for (i=1:5)
```

```
            display(arr_param_name_mod(num_evec_1_idx_des(i),
arr_param_name))

      end


end


%keyboard
%for idx_param=1:3
%     num_std = std(x(:, idx_param) );
%     nx(:, idx_param) = norm_funct(x(:, idx_param), num_std );
%end



%[num_evec_pca, pca_x, num_scr_plot_pca] = pca_funct(nx', arr_del); % note
the '




% cov should make all approx 1 if pca done before already-> checks ok!

%str_title = strcat(datestr(now,2), ', ', datestr(now,14),':      # Patients:
',num2str(num_pat),', Log(EigValue) vs. Index ');
%eig_graph(num_scr_plot_pca, '', length(num_scr_plot_pca), str_title)







%str_title = strcat('Number of Patients: ', num2str(num_pat), '  ')
%str_x = 'Samples';
%str_y = ' PC1, PCA of PC1: PC PC1, PC PC2, PC PC3 and HR';


%figure
%Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', str_y, '
vs. ', str_x ))

%num_pc=1
%subplot(7, 1, 1), plot(  1:length(x(:, num_pc) )  , x(:, num_pc) )

%for(num_pc=1:3)
%subplot(7, 1, num_pc+1), plot(  1:length(pca_x(:, num_pc) )  , pca_x(:,
num_pc) )
%end

%% Param. #5 = PR = [Beats per min.]

%subplot(7, 1, 5+2), plot(  1:length(all_data.npdata(5,:) ),
all_data.npdata(5,:) )
```

```
%Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', str_y, '
vs. ', str_x ))

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^


keyboard
%[num_s, num_noise, num_order] = ion(all_data.sel_npdata(:,:)', 10);    %%

cd (log_path)
%save('ION_data', 'num_s', 'num_noise', 'num_order')              %%
num_s='';num_noise=''; num_order=''; load('ION_data', 'num_s', 'num_noise',
'num_order')                  %
display('ion done...')



%%%%keyboard



for idx_param=1:num_sel_param
all_data.ionpdata(idx_param, :) = all_data.sel_npdata(idx_param, :)./
sqrt(num_s(idx_param)) ; % num_s = ion norm
end



arr_del = PARAM_HR_SHIFT_m240_mod;
[num_evec_ion, x_ion, num_scr_plot_ion] = pca_funct(all_data.ionpdata,
arr_del);
x='';
x=x_ion;

display('22')


% sorting within a eigenvector for parameter weight value...

display('params');
for (i=1:7)
      [num_evec_1_srt, num_evec_1_idx] = sort(abs(num_evec_ion(:,i)));
num_evec_ion(num_evec_1_idx_des(1),i)

      num_evec_1_srt_des = flipud(num_evec_1_srt);
      num_evec_1_idx_des = flipud(num_evec_1_idx);

      arr_param_name=make_arr_param_name;




      % NOTE: strcat = trick to get cells to be converted to strings!!
```

```
	str_msg = sprintf('Eig_vec: #%d, max idx: %d %d %d %d %d, max. values: 
%f %f %f %f %f\n', i, num_evec_1_idx_des(1), num_evec_1_idx_des(2), 
num_evec_1_idx_des(3), num_evec_1_idx_des(4), num_evec_1_idx_des(5), 
num_evec_1_srt_des(1), num_evec_1_srt_des(2), num_evec_1_srt_des(3), 
num_evec_1_srt_des(4), num_evec_1_srt_des(5))
	for (i=1:5)
		display(arr_param_name_mod(num_evec_1_idx_des(i), 
arr_param_name))

	end

end

% REMOVE the predicted HR value so no answer given to predictor!!!
num_evec_ion(arr_del,:)=0;

display('final lin. estimator');
keyboard

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^

%_SHIFT_m240


%x_ion=x_pca;

%,24:32


%x_ion=all_data.sel_pdata([1:22,24:32],:)';
%x_ion='';
%x_ion=all_data.sel_pdata([1:5],:)';



num_samp_pred = 7;
num_samp_shift = 5;
%num_ion=31;

num_ion=5;
num_time_col_cat_new = length(x_ion(:,1)) - num_samp_pred*num_pat - 
num_samp_shift*num_pat;   % NOTE: num_samp_pred (since deleted upon shifting)




%x_ion(1:num_train, 2)' x_ion(1:num_train, 3)' x_ion(1:num_train, 4)' 
x_ion(1:num_train, 5)' x_ion(1:num_train, 6)'



size( x_ion(:, 1)');

num_train = fix( (num_time_col_cat_new)*(2/3) );      % 3/4

% note the '
```

```
y = all_data.pdata(PARAM_HR, :)';

%y = all_data.pdata(PARAM_BPM, :)';

Y_comp = y;
%% shift y by shift_pred = 8 (=2 min, 40 sec)
shift_pred = num_samp_pred;

if (~isempty(find(isnan(y))))
      error('NaN found in data!')
end

shift_pred = num_samp_pred;
for (cur_spot = arr_spot(1:length(arr_spot)-1) )

      for (cur_loc = 1:num_samp_pred + num_samp_shift)

      y(cur_spot + cur_loc - 1)=NaN;

      end

end
      y=y (find(~isnan(y)));       % must delete only after all are found-
else alignment markers get off...


shift_pred = num_samp_pred;

for (cur_spot = arr_spot(2:length(arr_spot)-1) )      %

      for (cur_loc = 1:num_samp_shift)  %num_samp_shift     %% 2
num_samp_pred  num_samp_shift

      Y_comp(cur_spot + cur_loc - 1)=NaN;

      end

end




% delete trailing shift_pred from Y_comp
for (cur_spot = arr_spot(2:length(arr_spot)) )        %%

      for (cur_loc = 1:num_samp_pred)                 %%

      Y_comp(cur_spot - cur_loc )=NaN;          %%

      end                                     %%

end                                           %%
%     Y_comp_old = Y_comp;
      Y_comp = Y_comp (find(~isnan(Y_comp)));       % must delete only after
all are found- else arr_spot alignment markers get off...
```

```
for(idx_ion=1:num_ion)
      if (~isempty(find(isnan(x_ion(:, idx_ion)))))
            error('NaN found in data!')
      end
%x_pred(:,idx_ion) = x_ion(:,idx_ion);
end

for(idx_shift=1:num_samp_shift)
      x_hr(1:size(all_data.pdata, 2), idx_shift) =
all_data.pdata(PARAM_DERIV_HR, : )';      %%%%^^^^  % _SHIFT_m240
      x_pred(1:size(x_ion, 1),1:num_ion, idx_shift) = x_ion(1:size(x_ion,
1),1:num_ion);
end


if (~isempty(find(isnan(x_hr))))
      error('NaN found in data!')
end

if (~isempty(find(isnan(x_pred))))
      error('NaN found in data!')
end


%

      for (cur_spot = arr_spot(1:length(arr_spot)-1) )

            for (shift_pred=0:1:num_samp_shift-1)
                  for (cur_loc = 1:num_samp_shift - shift_pred)
                        % remove leading edge
                        for (idx_ion=1:num_ion)
                              x_pred(cur_spot + cur_loc - 1, idx_ion,
shift_pred+1) = NaN;
                        end

                              x_hr(cur_spot + cur_loc - 1, shift_pred+1) =
NaN;

                  end


            end
      end




      for (cur_spot = arr_spot(2:length(arr_spot)) )

            for (shift_pred=0:1:num_samp_shift-1)
                  for (cur_loc = 1:num_samp_pred + shift_pred)
```

```
                        % remove leading edge
                        for (idx_ion=1:num_ion)
                              x_pred(cur_spot - cur_loc, idx_ion,
shift_pred+1) = NaN;

                        end
                              x_hr(cur_spot - cur_loc, shift_pred+1) = NaN;

                  end


            end
      end

x_pred_new='';
x_pred_new(size(x_pred, 1) ,num_ion,num_samp_shift)=0;







for (i=1:num_ion)
      for (idx_shift=1:num_samp_shift)
            x_pred_new(1:length(find(~isnan(x_pred(:, i, idx_shift))) ),  i,
idx_shift )=x_pred (find(~isnan(x_pred(:, i, idx_shift))),  i, idx_shift ); %
must delete only after all are found- else alignment markers get off...
end
end

x_hr_new(size(x_pred, 1) ,num_samp_shift)=0;
      for (idx_shift=1:num_samp_shift)
            x_hr_new(1:length(find(~isnan(x_hr(:, idx_shift))) ), idx_shift )
= x_hr (find(~isnan(x_hr(:, idx_shift))), idx_shift); % must delete only
after all are found- else alignment markers get off...
      end




X_train = [ones(num_train,1 )  ]  ;

for (i=1:num_ion)
      for (idx_shift=1:num_samp_shift)
            X_train=[X_train, x_pred_new( 1:num_train , i, idx_shift) ];
      end
end
%%
%     for (idx_shift=1:num_samp_shift)
%           X_train=[X_train, x_hr_new( 1:num_train , idx_shift) ];
%     end
%%
```

```
arr_test_set = (num_train+1:(num_time_col_cat_new));




X_test = [ones((num_time_col_cat_new-num_train),1 )   ];


for i=1:num_ion
      for (idx_shift=1:num_samp_shift)
            X_test=[X_test, x_pred_new( arr_test_set , i, idx_shift) ];
      end
end

%%
%     for (idx_shift=1:num_samp_shift)
%           X_test=[X_test, x_hr_new( arr_test_set , idx_shift) ];
%     end
%%

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^
%
%

%h=X_train;
%X_train=X_test;
%X_test = h;


%X_train = [ones((num_train),1 ) , rand(num_train,num_samp_shift*num_ion)];
%X_test = [ones((num_time_col_cat_new-num_train),1 ),
rand(num_time_col_cat_new - num_train,num_samp_shift*num_ion)];

%%%%keyboard
a = X_train\(y(1:num_train) );

% NOTE: 1:num_train = UNDEFINED for Y_test!!
Y_test=''
Y_test(num_train+1:num_time_col_cat_new) = X_test*a
Y_test=Y_test';

Y_train = X_train*a;


display('stop here')
%%%%keyboard
res_test = Y_test(num_train+1:num_time_col_cat_new) -
y(num_train+1:num_time_col_cat_new); % = predicted - actual

res_train = Y_train - y(1:num_train); % = predicted - actual
```

```
% note the '
% Y_comp = all_data.pdata(PARAM_HR, :)';
% Y_comp defined earlier!

res1 = Y_comp(num_train+1:num_time_col_cat_new) -
y(num_train+1:num_time_col_cat_new); % = predicted - actual


Y_avg=''
for(i=num_train+1:num_time_col_cat_new)
      Y_avg(i) = mean(Y_comp(i-2:i));
end

Y_avg = Y_avg';

res2 = Y_avg(num_train+1:num_time_col_cat_new) -
y(num_train+1:num_time_col_cat_new); % = predicted - actual



MaxErr_YTestTrain = max_error(Y_train(1:num_train),y(1:num_train) )
SumSq_YTestTrain = rms_error(Y_train(1:num_train),y(1:num_train) )


MaxErr_YLinTest = max_error(Y_test(num_train+1:num_time_col_cat_new),
y(num_train+1:num_time_col_cat_new)  )
SumSq_YLinTest = rms_error(Y_test(num_train+1:num_time_col_cat_new),
y(num_train+1:num_time_col_cat_new) )
mean(res_test)
RMS_YLinTestRes = rms_error(res_test , mean(res_test) )

% ------------------------------------------------------------
MaxErr_YCompTrain = max_error(Y_comp(1:num_train),y(1:num_train) )
SumSq_YCompTrain = rms_error(Y_comp(1:num_train),y(1:num_train) )


MaxErr_YCompTest = max_error(Y_comp(num_train+1:num_time_col_cat_new),
y(num_train+1:num_time_col_cat_new))
SumSq_YCompTest = rms_error(Y_comp(num_train+1:num_time_col_cat_new),
y(num_train+1:num_time_col_cat_new) )
mean(res1)
RMS_YCompTestRes = rms_error(res1 , mean(res1) )

% ------------------------------------------------------------
MaxErr_YAvgTrain = max_error(Y_avg(1:num_train),y(1:num_train) )
SumSq_YAvgTrain = rms_error(Y_avg(1:num_train),y(1:num_train) )


MaxErr_YAvgTest = max_error(Y_avg(num_train+1:num_time_col_cat_new),
y(num_train+1:num_time_col_cat_new))
SumSq_YAvgTest = rms_error(Y_avg(num_train+1:num_time_col_cat_new),
y(num_train+1:num_time_col_cat_new) )
mean(res2)
RMS_YAvgTestRes = rms_error(res2 , mean(res2) )
```

```
%10.44
%15.67
%15.96


%_____________________________________

figure, plot(1:num_train, res_train, '+')
Title('Residues: ION PCA-Based Predictor Training.  Heart Rate vs. Time')

figure, plot(1:num_train, Y_train(1:num_train), 'rx', 1:num_train,
y(1:num_train), 'bo')
Title('ION PCA-Based Predictor Training.  Heart Rate vs. Time')
legend('Prediction','Actual Dataset');

figure, plot(1:length( x_ion((num_train+1:num_time_col_cat_new), 1)),
res_test, '+')
Title('Residues: ION PCA-based Predictor Test.  Heart Rate vs. Time')

figure, plot(1:length( x_ion((num_train+1:num_time_col_cat_new), 1)),
Y_test(num_train+1:num_time_col_cat_new), 'rx', 1:length(
x_ion((num_train+1:num_time_col_cat_new), 1)),
y(num_train+1:num_time_col_cat_new), 'bo')
Title('ION PCA-based Predictor Test.  Heart Rate vs. Time')
legend('Prediction','Actual Dataset');



figure, plot(1:length( x_ion((num_train+1:num_time_col_cat_new), 1)), res1,
'+')
Title('Residues: Static Heart Rate Predictor Test.  Heart Rate vs. Time')
figure, plot(1:length( x_ion((num_train+1:num_time_col_cat_new), 1)),
Y_comp(num_train+1:num_time_col_cat_new), 'rx', 1:length(
x_ion((num_train+1:num_time_col_cat_new), 1)),
y(num_train+1:num_time_col_cat_new), 'bo')
Title('Static Heart Rate Predictor Test.  Heart Rate vs. Time')
legend('Prediction','Actual Dataset');



figure, plot(1:length( x_ion((num_train+1:num_time_col_cat_new), 1)), res2,
'+')
Title('Residues: Heart Rate Average-Based Predictor Test.  Heart Rate vs.
Time')

figure, plot(1:length( x_ion((num_train+1:num_time_col_cat_new), 1)),
Y_avg(num_train+1:num_time_col_cat_new), 'rx', 1:length(
x_ion((num_train+1:num_time_col_cat_new), 1)),
y(num_train+1:num_time_col_cat_new), 'bo')
Title('Heart Rate Average-Based Predictor Test.  Heart Rate vs. Time')
legend('Prediction','Actual Dataset');

%_____________________________________
```

```
%str_title = strcat(datestr(now,2), ', ', datestr(now,14),':      # Patients:
',num2str(num_pat),', Log(EigValue) vs. Index ');
str_title = strcat('Scree Plot Subset for 46 Patient Dataset With and Without
ION. Sorted by Log(Eigenvalue)');

eig_graph(num_scr_plot, num_scr_plot_ion, 6, str_title)
legend('Without ION', 'With ION');
str_title = strcat('Scree Plot for 46 Patient Dataset With and Without ION.
Sorted by Log(Eigenvalue)');
%eig_graph(num_scr_plot, num_scr_plot_ion, 16, str_title)
eig_graph(num_scr_plot, num_scr_plot_ion, length(num_scr_plot), str_title)
legend('Without ION', 'With ION');

display('Delta(1-2)')
(log(num_scr_plot_ion(1)) - log(num_scr_plot_ion(2))) - (log(num_scr_plot(1))
- log(num_scr_plot(2)))
display('Delta(2-3)')
(log(num_scr_plot_ion(2)) - log(num_scr_plot_ion(3))) - (log(num_scr_plot(2))
- log(num_scr_plot(3)))
display('Delta(3-4)')
 (log(num_scr_plot_ion(3)) - log(num_scr_plot_ion(4))) -
(log(num_scr_plot(3)) - log(num_scr_plot(4)))
display('Delta(4-5)')
 (log(num_scr_plot_ion(4)) - log(num_scr_plot_ion(5))) -
(log(num_scr_plot(4)) - log(num_scr_plot(5)))


keyboard
display('really stop here!')




for (idx_param=1:size(num_evec_ion, 1))
      for (idx_shift=1:num_samp_shift)

            num_data(idx_param,idx_shift) = 0;
            for (idx_ion=1:num_ion)

      num_data(idx_param,idx_shift)=num_data(idx_param,idx_shift)+a(idx_ion+(
idx_shift-1)*  num_ion  +1)*num_evec_ion(idx_param, idx_shift);
            end

            idx_param
            if (idx_param==4)
                  display('here')
                  num_data(idx_param,:)
                  display('done')
            end

      end
      num_data_row(idx_param)=sum(num_data(idx_param,:));

end
num_data_row
num_data
num_evec_ion(:,:)
```

```
keyboard

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^

num_thr = 100
%%num_thr = 106
%%%%%%%%%%%%%%%%%%%%%%%%% Prob Detection / False Alarm

for (hr_real_thresh = [65,num_thr,120])

hr_min_thresh=30
%hr_real_thresh=100
hr_max_thresh=180
hr_step_thresh=1



for (j=1:3)


switch (j)

case 1
y_act = Y_test(num_train+1:num_time_col_cat_new);
case 2
y_act = Y_comp(num_train+1:num_time_col_cat_new);
case 3
y_act = Y_avg(num_train+1:num_time_col_cat_new);
end



y_detect = y_act;

y_dreal = y(num_train+1:num_time_col_cat_new);


      idx_detect = find(y_dreal <= hr_real_thresh);
y_dreal(idx_detect)=    0 ;

      idx_detect = find(y_dreal > 0);
y_dreal(idx_detect)=    1 ;





i=0;
for (num_thresh= hr_min_thresh:hr_step_thresh:hr_max_thresh)

      idx_detect = find(y_act <= num_thresh);
y_detect(idx_detect)=   0 ;

      idx_detect = find(y_act > num_thresh);
y_detect(idx_detect)=   1 ;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%% Prob Detection
i=i+1;

%%%                      sum of all that should be detected -


%  sum of num of no detection when there should have been = num. rightly
detected
% num_right/sum of all that should be detected  = prob. detection.

num_right = sum(y_detect(find(y_dreal>0)));

num_tachy = sum(y_dreal(find(y_dreal>0)));      % sum of all


if (num_tachy==0)
num_tachy = 1e-10;
end

arr_pdetect(j, i) = num_right / num_tachy;


%%%%%%%%%%%%%%%%%%%%%%%%  False Alarm
%                                    sum of all that should NOT be detected -
sum of num of YES detections when there should NOT have been = num. false
alarms




% sum of num of YES detections when there should NOT have been/ sum of all
that should NOT be detected  = prob. fa
num_no = length(find(y_dreal==0));  % sum of all (can't sum zeroes- would get
zero- so we take the length of the 'find')



if (num_no==0)
num_no = 1e-10;
end

arr_pfa(j,i) = sum(y_detect(find(y_dreal==0))) / num_no;




end







end
```

```
figure
plot(arr_pfa(1,:), arr_pdetect(1, :) , 'rx', arr_pfa(2,:), arr_pdetect(2, :),
'bo', arr_pfa(3,:), arr_pdetect(3, :),'k+')
legend('ION PCA-Based Predictor', 'Static Predictor', 'Average-Based
Predictor')
str_title = strcat('Number of Patients: ', num2str(num_pat), '  ')
str_x = 'P(False Alarm)'
str_y = 'P(Detection)'
%Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', str_y, '
vs. ', str_x ))
Title(strcat(str_y, ' vs. ', str_x,'.  Tachycardia Threshold:
',num2str(hr_real_thresh), ' BPM' ))
%%Title(strcat(str_y, ' vs. ', str_x,'.  Threshold:
',num2str(hr_real_thresh), ' mmHg' ))

xlabel(str_x)
ylabel(str_y)

end


%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^

hr_min_thresh=30
%hr_real_thresh=100
hr_max_thresh=180
hr_step_thresh=1



for (j=1:5)


switch (j)

case 1
y_act = Y_test(num_train+1:num_time_col_cat_new);
hr_real_thresh=60
case 2
hr_real_thresh=80
case 3
hr_real_thresh=100
case 4
hr_real_thresh=120
case 5
hr_real_thresh=140
end



y_detect = y_act;

y_dreal = y(num_train+1:num_time_col_cat_new);


      idx_detect = find(y_dreal <= hr_real_thresh);
```

```
y_dreal(idx_detect)=    0 ;

      idx_detect = find(y_dreal > 0);
y_dreal(idx_detect)=    1 ;




i=0;
for (num_thresh= hr_min_thresh:hr_step_thresh:hr_max_thresh)

      idx_detect = find(y_act <= num_thresh);
y_detect(idx_detect)=   0 ;

      idx_detect = find(y_act > num_thresh);
y_detect(idx_detect)=   1 ;

%%%%%%%%%%%%%%%%%%%%%%%%% Prob Detection
i=i+1;

%%%                     sum of all that should be detected -


%  sum of num of no detection when there should have been = num. rightly
detected
% num_right/sum of all that should be detected  = prob. detection.

num_right = sum(y_detect(find(y_dreal>0)));

num_tachy = sum(y_dreal(find(y_dreal>0)));      % sum of all


if (num_tachy==0)
num_tachy = 1e-10;
end

arr_pdetect(j, i) = num_right / num_tachy;


%%%%%%%%%%%%%%%%%%%%%%%%%  False Alarm
%                                     sum of all that should NOT be detected -
sum of num of YES detections when there should NOT have been = num. false
alarms




% sum of num of YES detections when there should NOT have been/ sum of all
that should NOT be detected  = prob. fa
num_no = length(find(y_dreal==0));  % sum of all (can't sum zeroes- would get
zero- so we take the length of the 'find')



if (num_no==0)
```

```
num_no = 1e-10;
end

arr_pfa(j,i) = sum(y_detect(find(y_dreal==0))) / num_no;




end







end


figure
plot(arr_pfa(1,:), arr_pdetect(1, :) , 'rx', arr_pfa(2,:), arr_pdetect(2, :),
'bo', arr_pfa(3,:), arr_pdetect(3, :),'k+',arr_pfa(4,:), arr_pdetect(4, :) ,
'gv', arr_pfa(5,:), arr_pdetect(5, :) , 'm*')
legend('Threshold: 70 BPM', 'Threshold: 90 BPM', 'Threshold: 100 BPM',
'Threshold: 110 BPM', 'Threshold: 130 BPM')
%%legend('Threshold: 70 mmHg', 'Threshold: 90 mmHg', 'Threshold: 106 mmHg',
'Threshold: 110 mmHg', 'Threshold: 130 mmHg')

%str_title = strcat('Number of Patients: ', num2str(num_pat), '  ')
str_x = 'P(False Alarm)'
str_y = 'P(Detection)'
%Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', str_y, '
vs. ', str_x ))
Title(strcat(str_y, ' vs. ', str_x,' of ION PCA-Based Predictor for Various
Thresholds'))

xlabel(str_x)
ylabel(str_y)



%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^


keyboard


x=x_ion;

% Plot PC1 vs. n (not necessarily evenly spaced)

% plot_graph(1:length(x(:,1)), x(:,1), strcat('Number of Patients: ',
num2str(num_pat), '  '), 'Samples', 'PC1')



str_title = strcat('Number of Patients: ', num2str(num_pat), '  ')
```

```
str_x = 'Samples';
str_y = ' PC1, PC2, PC3 and HR';


figure
Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', str_y, '
vs. ', str_x ))

for(num_pc=1:5)
subplot(11, 1, 2*num_pc-1), plot(  1:length(x(:, num_pc) )/3  ,
x(1:length(x(:, num_pc) )/3, num_pc) )

Title(strcat('PC', num2str(num_pc), ' vs. Time'))
end

% Param. #5 = PR = [Beats per min.]

subplot(11, 1, 11), plot(  1:length(all_data.npdata(PARAM_HR,:) )/3,
all_data.pdata(PARAM_HR,1:length(all_data.npdata(PARAM_HR,:) )/3) )
Title(strcat('Heart Rate vs. Time'))
%Title(strcat(datestr(now,2), ', ', datestr(now,14),str_title, '- ', str_y, '
vs. ', str_x ))




cur_spot = 1;

for i=1:num_pat
%                 [Hue,             Saturation, Value]      /Normalize to 1

%%keyboard
%length(  (([(i*NUM_COL/num_pat), (NUM_COL*.5), (NUM_COL*1/3)]/NUM_COL)' *
ones(1, idx_length_pat(i)) )')
%cur_spot:cur_spot+idx_length_pat(i)-1;

% 0.9 luminosity, ones() for columns!
if (i==32 | i==10 | i==24)
idx_pat=i;
else idx_pat=1;
end


hsv_col_map_pat(cur_spot:cur_spot+idx_length_pat(i)-1,:) =
(([(idx_pat*NUM_COL/num_pat), (NUM_COL*.9), (NUM_COL*2/3)]/NUM_COL)' *
ones(1,idx_length_pat(i) ))';

hsv_col_map_bar(i,:) = hsv_col_map_pat(cur_spot,:);

cur_spot = cur_spot+idx_length_pat(i)-1;

end

rgb_col_map_pat = hsv2rgb(hsv_col_map_pat);
rgb_col_map_bar = hsv2rgb(hsv_col_map_bar);
```

```
num_min = min(size(x,1), length(rgb_col_map_pat) )

% plot_graph(x(:,2), x(:,1), strcat('Number of Patients: ', num2str(num_pat),
'  '), 'PC2', 'PC1')

arr_x = x(1:num_min,2); arr_y = x(1:num_min,1);
scatter_graph(arr_x, arr_y, strcat('Number of Patients: ', num2str(num_pat),
'  '), 'PC2', 'PC1',[10]*ones(num_min,1)','','')

arr_x = x(1:num_min,1); arr_y = x(1:num_min,2); arr_z = x(1:num_min,3);
scatter3_graph(arr_x, arr_y, arr_z, strcat('Number of Patients: ',
num2str(num_pat), '  '), 'PC3', 'PC2', 'PC1', [10]*ones(num_min,1)','','')









%surface(1:num_pat,ones(1,num_pat),ones(1,num_pat),rgb_col_map_bar)

size(rgb_col_map_pat)
%size(arr_x)
figure
bar(ones(2,num_pat),'stacked'), colormap(rgb_col_map_bar)

arr_x = x(1:num_min,2);arr_y = x(1:num_min,1);
%keyboard
%scatter_graph(arr_x, arr_y, strcat('Number of Patients: ', num2str(num_pat),
'  '), 'PC2', 'PC1', [5]*ones(length(arr_x),1)',
rgb_col_map_pat(1:num_min,1:3), rgb_col_map_bar )
scatter_graph(arr_x, arr_y, strcat('Scatter Plot of: PC1 vs. PC2 (Colored by
Patient)'), 'PC2', 'PC1', [5]*ones(length(arr_x),1)',
rgb_col_map_pat(1:num_min,1:3), rgb_col_map_bar )


%% scatter(1:30,ones(1, num_pat), [200]*ones(num_pat,1)', rgb_col_map_bar,
'filled')
%keyboard


arr_x = x(1:num_min,3);arr_y = x(1:num_min,1);

scatter_graph(arr_x, arr_y, strcat('Scatter Plot of PC1 vs. PC3 (Colored by
Patient)'), 'PC3', 'PC1', [5]*ones(length(arr_x),1)',
rgb_col_map_pat(1:num_min,1:3), rgb_col_map_bar )


arr_x = x(1:num_min,3);arr_y = x(1:num_min,2);
```

```
scatter_graph(arr_x, arr_y, strcat('Scatter Plot of PC2 vs. PC3 (Colored by
Patient)'), 'PC3', 'PC2', [5]*ones(length(arr_x),1)',
rgb_col_map_pat(1:num_min,1:3), rgb_col_map_bar )


arr_x = x(1:num_min,1); arr_y = x(1:num_min,2); arr_z = x(1:num_min,3);
scatter3_graph(arr_x, arr_y, arr_z, strcat('3-D Scatter Plot of PC1, PC2, and
PC3 (Colored by Patient)'), 'PC1', 'PC2', 'PC3', [10]*ones(num_min,1)',
rgb_col_map_pat(1:num_min,1:3), rgb_col_map_bar )
% line([-40,40],[10,10],[8,0])
patch([-75,-75,75,75],[-10,10,10,-10],[10,12,-6,-8],'b')



arr_x = x(1:num_min,1); arr_y = x(1:num_min,2); arr_z = x(1:num_min,3);
scatter3_graph(arr_x, arr_y, arr_z, strcat('3-D Scatter Plot of PC1, PC2, and
PC3 (Colored by Patient)'), 'PC1', 'PC2', 'PC3', [10]*ones(num_min,1)',
rgb_col_map_pat(1:num_min,1:3), rgb_col_map_bar )
% line([-40,40],[10,10],[8,0])
patch([-75,-75,75,75],[-10,10,10,-10],[9,9,-6,-6],'g')
line([-30,30],[10,10],[-6,6])

scatter3_graph(arr_x(1:88,1), arr_y(1:88,1), arr_z(1:88,1), strcat('3-D
Scatter Plot of PC1, PC2, and PC3 (Colored by Patient)'), 'PC1', 'PC2', 'PC3',
[10]*ones(88,1)', rgb_col_map_pat(1:88,1:3), rgb_col_map_bar([1,2],:)  )



%keyboard








%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
display('3d time graphs')

keyboard
```

```
rgb_col_map_tim = hsv2rgb(hsv_col_map_tim);
rgb_col_map_tim_bar = hsv2rgb(hsv_col_map_tim_bar);

arr_x = x(1:num_min,1); arr_y = x(1:num_min,2); arr_z = x(1:num_min,3);
scatter3_graph(arr_x, arr_y, arr_z, strcat('3-D Scatter Plot of PC1, PC2, and
PC3 (Colored by Time From Induction of Inhaled Gas)'), 'PC1', 'PC2', 'PC3',
[10]*ones(num_min,1)', rgb_col_map_tim(1:num_min,1:3), rgb_col_map_tim_bar )
```

```
% dedicated to the sweet girl whose name I have yet to discover...
%figure

%bar(1:4, log(num_scr_plot(1:4) ), 1)
%Title(strcat(datestr(now,2), ', ', datestr(now,14),':      # Patients:
',num2str(num_pat),', Log(EigValue) vs. Index '))
%xlabel('Index')
%ylabel('Log(EigValue)')


%%str_title = strcat(datestr(now,2), ', ', datestr(now,14),':      #
Patients: ',num2str(num_pat),', Log(EigValue) vs. Index ');

%%eig_graph(num_scr_plot, '', 4, str_title)
%eig_graph(num_scr_plot, '', 16, str_title)
%%eig_graph(num_scr_plot, '', length(num_scr_plot), str_title)


%keyboard



%str_title = strcat(datestr(now,2), ', ', datestr(now,14),':      Patient
#',num2str(idx_pat),', Paramters vs. Time');
%image_graph(all_data.sel_npdata(:,:), str_title)
```

File 2: Pca_funct.m

```
function [num_evec_ret, x, num_scr_plot] = pca_funct(mat_data, arr_del)



num_cov = cov(mat_data(:,:)' ); % so each column= paramter and each row =
observation

% eval = eigenvalue
[num_evec, num_eval] = eig(num_cov);
```

```
(diag(num_eval));
% keyboard

[num_eval_sort, num_idx_sort] = sort(diag(num_eval));

% sort so largest is first.
num_eval_sort_descend = flipud(num_eval_sort);
num_idx_sort_descend = flipud(num_idx_sort);

% x = sorted in order descending


% length(num_idx_sort_descend)


num_evec_trunc = num_evec(:, num_idx_sort_descend(1:10)  );
num_evec_trunc(arr_del,:)=0;


num_evec_ret = num_evec(:, num_idx_sort_descend(1:10)  );



arr_pc = num_evec_trunc' * mat_data(:,:);
x=arr_pc';




num_scr_plot = flipud(sort(diag(diag(num_eval_sort_descend))));

if (~isempty(find(num_scr_plot <= 0) ) )
      display('Some eigenvalues <=0')
end

num_scr_plot(find(num_scr_plot<=0)) = 1e-10;
```

File 3: Pre_processing.m

```
function arr_all_pat = pre_processing(arr_all_pat, log_path, OS)
global PARAM_MAX_LINE
global MAX_NUM_PARAM
global PARAM_GOOD_PAT
global PARAM_LINE_START
global PARAM_LINE_STOP
global PARAM_AGTI_START

global FALSE
global TRUE



global idx_pat
```

```
MAX_NUM_NAN_ROWS = 10;

% clip off start
% take max line of parameters 1 or 5
max_pat=size(arr_all_pat.pdata, 1);



% put NaN's in remaining spots... before looking at NaN's!!!
      arr_all_pat.pdata(idx_pat,:,arr_all_pat.misc(idx_pat, PARAM_LINE_STOP)+1
:size(arr_all_pat.pdata, 3) ) = NaN;
%%% Strategy: replace all zeros in data to arbitrary number: -5555
%% then replace all -5555's with zeros after first replacing zeros with NaN's
in pre_processing

for idx_pat=1:max_pat

arr_all_pat.misc(idx_pat, PARAM_GOOD_PAT) = TRUE; % default = true

      for idx_param=1:MAX_NUM_PARAM

            idx_nans = find(arr_all_pat.pdata(idx_pat, idx_param,:) == 0);
            arr_all_pat.pdata(idx_pat, idx_param,idx_nans) = NaN;

% order important!!
            idx_zeros = find(arr_all_pat.pdata(idx_pat, idx_param,:) == -
5555);
            arr_all_pat.pdata(idx_pat, idx_param,idx_zeros) = 0;

      end
end



for idx_pat=1:max_pat

% [pat. #, param, line]

idx_pat;
max_pat;
arr_all_pat.pdata(idx_pat, 1, :);
arr_all_pat.pdata(idx_pat, 5, :);

min_param_1_line = min(find(~isnan( arr_all_pat.pdata(idx_pat, 1,
1:mac_max_line) )));
```

```
min_param_5_line = min(find(~isnan( arr_all_pat.pdata(idx_pat, 5,
1:mac_max_line) )));

arr_all_pat.misc(idx_pat, PARAM_LINE_START) = max(min_param_1_line,
min_param_5_line);


max_param_1_line = max(find(~isnan( arr_all_pat.pdata(idx_pat, 1,
1:mac_max_line) )));

max_param_5_line = max(find(~isnan( arr_all_pat.pdata(idx_pat, 5,
1:mac_max_line) )));

arr_all_pat.misc(idx_pat, PARAM_LINE_STOP) = min(max_param_1_line,
max_param_5_line);







% adjust max_length counter
%     arr_all_pat.misc(idx_pat,PARAM_MAX_LINE) = mac_max_line - start_line +
1;


%     arr_all_pat.misc(idx_pat,PARAM_LINE_STOP) = size(arr_all_pat.pdata, 3);









end



            line_start = arr_all_pat.misc(1, PARAM_LINE_START);
            line_stop = arr_all_pat.misc(1, PARAM_LINE_STOP);
            arr_all_pat.pdata(1,1,:);
%           input('2')




%pre-interpolation- start/end setting

for idx_pat=1:max_pat
      for idx_param=1:MAX_NUM_PARAM
            % ******* set start line to first non-NaN value- for interpolation
```

```
            line_start = arr_all_pat.misc(idx_pat, PARAM_LINE_START);
            line_stop = arr_all_pat.misc(idx_pat, PARAM_LINE_STOP);

            if (isnan(arr_all_pat.pdata(idx_pat, idx_param, line_start) ))


% determine value at line_stop- should be value at min. index



                  arr_tmp = min(find(~isnan(arr_all_pat.pdata(idx_pat,
idx_param, line_start:line_stop) )));
                  if (~isempty(arr_tmp))
                        arr_all_pat.pdata(idx_pat, idx_param, line_start) =
arr_all_pat.pdata(idx_pat, idx_param, line_start+arr_tmp-1);
                  end
            end




%display( arr_all_pat.pdata(idx_pat, idx_param, line_stop)  )
% input('arr at line_stop')

            % *** set max line to last non-NaN value- for interpolation
            if (isnan(arr_all_pat.pdata(idx_pat, idx_param, line_stop) ))
% keyboard
                  % input('isnan');
                  h = max(find(~isnan(arr_all_pat.pdata(idx_pat, idx_param,
line_start:line_stop) )));
                  if ~isempty(h)          % else it means all is NaN


% keyboard
                        arr_all_pat.pdata(idx_pat, idx_param, line_stop) =
arr_all_pat.pdata(idx_pat, idx_param, line_start+h-1);
% input('9')

                  % need to check later again if all is empty!!
                  end

            end


      end
end

% set start_line to have first non-NaN value
```

```
% interpolation = deleting NaN's

try

for idx_pat=1:max_pat
      for idx_param=1:MAX_NUM_PARAM
            line_start = arr_all_pat.misc(idx_pat, PARAM_LINE_START);
            line_stop = arr_all_pat.misc(idx_pat, PARAM_LINE_STOP);


            % find first NaN starting from the patient's start line

% rem to add line_start-1 offset!!!!!!!!!!!!
            idx_NaN =
find(isnan(arr_all_pat.pdata(idx_pat,idx_param,line_start:line_stop) ));
            if (
all(isnan(arr_all_pat.pdata(idx_pat,idx_param,line_start:line_stop) )))
                  idx_NaN = []; %don't search if all paramter is NaN!!!
            end

            if ~isempty(idx_NaN)
                  for k=(idx_NaN+line_start-1)
%                       idx_pat
%                       idx_param
%line_start
%line_stop
%k
                        num_search = k-1;
%input('1');
                        while (isnan(arr_all_pat.pdata(idx_pat, idx_param,
num_search)) & num_search > line_start)
                              num_search=num_search-1;
                        end

                        num_beg = num_search;



                        num_search = k+1;
                        while (isnan(arr_all_pat.pdata(idx_pat, idx_param,
num_search)) & num_search < line_stop)
                              num_search=num_search+1;
                        end
                              num_end = num_search;

                        num_nan_rows = (num_end - num_beg + 1);
%if (idx_param>17 )
%keyboard
%end
                        if (num_nan_rows > MAX_NUM_NAN_ROWS &
~isempty(find(~isnan(arr_all_pat.pdata(idx_pat, idx_param,
line_start:line_stop))))) & (num_end > line_start)

                              arr_all_pat.pdata(idx_pat, idx_param,
line_start) = arr_all_pat.pdata(idx_pat, idx_param, num_end+1);
```

```
                                if ( num_end > min(find
(~isnan(arr_all_pat.pdata(idx_pat, idx_param, 2:line_stop)) )))
                                      % we are beyond the min. number that is
not nan in the column => we do have string of nan's after the start => bad
                                      arr_all_pat.misc(idx_pat, PARAM_GOOD_PAT)
= FALSE;
arr_all_pat.pdata(idx_pat, idx_param, num_beg:num_end)

                                      display(strcat('Bad patient parameter
found: ',num2str(num_nan_rows), ' NaN rows for patient #',num2str(idx_pat), '
in parameter #', num2str(idx_param), ', row #', num2str(k), ' num_end: ',
num2str(num_end), ' line_start: ', num2str(line_start) , ' min:',
num2str(min(find (~isnan(arr_all_pat.pdata(idx_pat, idx_param, 1:line_stop)))
))  ) )
                                      % keyboard

                                      fid = fopen( strcat(log_path,
OS.SEP,'logfile.txt') ,'a');
                                      fprintf(fid, 'Bad patient parameter found:
%d NaN rows for patient #%d in parameter #%d, row #%d\n',num_nan_rows,
idx_pat, idx_param, k);
                                      fclose(fid);
                                else
                                      % we just have string of nan's at start-
BUT they are after the original line_start.  So, we init line_start value to
first non-nan vlue of column

 % we do this for both cases now% %arr_all_pat.pdata(idx_pat, idx_param,
line_start) = arr_all_pat.pdata(idx_pat, idx_param, num_end+1);

                                end
                          end




                          num_slope = (arr_all_pat.pdata(idx_pat, idx_param,
num_end) - arr_all_pat.pdata(idx_pat, idx_param, num_beg)) / (num_nan_rows -
1);  % (Differnece in patient data array values)  / (number of NaN's total in
space - 1)
                          for j=(num_beg+1):(num_end-1)
                                arr_all_pat.pdata(idx_pat, idx_param, j) =
arr_all_pat.pdata(idx_pat, idx_param, num_beg) + num_slope;

                          end
                    end
              end
%interp1(arr_all_pat.pdata(idx_pat,idx_param,1:mac_max_line), idx_NaN,
'*linear')
%input('1');
%           arr_all_pat.pdata(idx_pat,idx_param,1:mac_max_line) =
interp1(arr_all_pat.pdata(idx_pat,idx_param,1:mac_max_line), idx_NaN,
'*linear')

      end
```

```
PARAM_AGTI = 18;
PARAM_AGTI_START=7;

arr_all_pat.misc(idx_pat, PARAM_AGTI_START) = min(find(
arr_all_pat.pdata(idx_pat, PARAM_AGTI, line_start:line_stop)>0 )) +
line_start-1; % NOT DONE: - 4; % - 4 = 1 20 sec minute of samples added to
beginning

end









%for idx_pat=1:max_pat
%     for idx_param=1:MAX_NUM_PARAM
%           idx_NaN = find(isnan(arr_all_pat.pdata(idx_pat,idx_param,:) ));


%           if (
all(isnan(arr_all_pat.pdata(idx_pat,idx_param,line_start:line_stop) )))
%     end
%end







catch
display(strcat('idx_pat: ', num2str(idx_pat), '- idx_param: ',
num2str(idx_param), '- num_search: ', num2str(num_search), '- k: ',
num2str(k), '- line_start: ', num2str(line_start), '-error:', lasterr   ))
keyboard
end
```