Exploration, Processing and Visualization of Physiological Signals from the ICU

By

Carlos A. Renjifo

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 19, 2005

© 2005 MIT. All rights reserved.

Author_____
Department of Electrical Engineering and Computer Science
May 19, 2005

Certified by_____
George C. Verghese
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Exploration, Processing and Visualization of Physiological Signals from the ICU

By
Carlos A. Renjifo

Submitted to the
Department of Electrical Engineering and Computer Science

May 19, 2005

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This report studies physiological signals measured from patients in the Intensive Care Unit (ICU). The signals explored include heart rate, arterial blood pressure, pulmonary artery pressure, and central venous pressure measurements. Following an introduction to these signals, several methods are proposed for visualizing the data using time and frequency domain techniques. By way of a patient case study we motivate a novel method for data clustering based on the singular value decomposition and present some potential applications based on this method for use within the ICU setting.

Thesis Supervisor: George C. Verghese
Title: Professor of Electrical Engineering and Computer Science

Table of Contents

List of Figures

List of Tables

Acknowledgements

Over the last year and a half there have been many people who have helped me along with this project. Without your help, this project would have been an impossible task.

Special thanks to George Verghese, my thesis advisor, for helping me formulate a thesis topic, giving me advice and challenging me along the way and for helping edit this thesis report.

Thanks also to Dr. Roger Mark, the P.I. for the BRP project. Your dedication to this project has helped establish the infrastructure within which the modeling group operates.

I would also like to offer a word of thanks to members of the modeling group (Thomas Heldt, Tushar Parlikar, Zaid Samar, and Willie Sanchez). Thomas and Tushar, thanks a lot for all the advice about how to get a thesis written, for giving me ideas about how to interpret my work and what to do next, and for innumerable great conversations to help me take my mind away from thesis work for at least a brief period of time. Also, thank you Thomas for giving me the idea to add the flip-book movie feature to this thesis. Zaid, we both worked hard to get our research done and thesis written and in the end things worked out. I wish you the best in all that the future holds for you. Willie: thanks for your help with some of the heart rate variability visualization.

Thanks to members of the laboratory for physiological computation (LCP). A big thank you to Gari Clifford for feeding me many ideas for my work, particularly those related to phase space analysis. Thanks also to Mohammed Saeed and Omar Abdala for help in understanding the workings of the annotation station and helping resolve data formatting issues. Finally, another big thank you to Brian Janz for spending countless hours with me in front of the annotation station to help annotate and understand the patient presented in this thesis.

To my friends (Matt, Howard, Suzanne, Tina, Jenny, Leeann, Pam and Harold): thanks for making these past five years at MIT amazing. I wish you all the best as you continue your lives at MIT and elsewhere.

I would also like to thank my family (Mom, Dad, Jorge, and Ale) for the unconditional support, comfort, and love that they have given me. I am truly blessed to be part of such a great family. Mom and Dad, thank you for always encouraging me to learn and for sacrificing so much to provide me with the best education possible. Jorge and Ale, you are the best friends anyone could have.

Last, but definitely not least, I would like to give praise to God for bringing me to where I am this day, instilling within me patience and comfort when things got stressful, and for keeping me honest in everything I've done in my life.

This work was supported in part through the National Aeronautics and Space Administration (NASA) through the NASA Cooperative Agreement NCC 9-58 with the National Space Biomedical Research Institute and in part by Grant RO1 EB001659 from the National Institute of Biomedical Imaging and Bioengineering.

Chapter 1

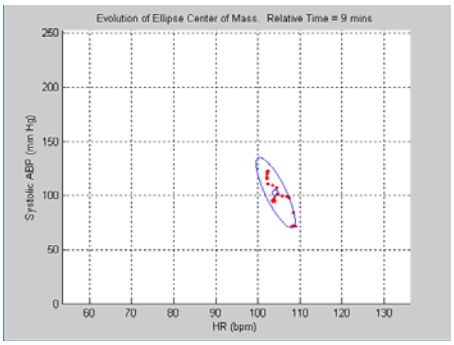
Introduction

1.1 Motivation

The use of physiological signals in the clinical setting has been of invaluable importance over the last hundred years. The signals, ranging from electrocardiogram tracings to blood pressure measurements, provide physicians with a wealth of information concerning the ongoing processes within the human body. Based on these signals, patient conditions are assessed and lives are saved on a daily basis.

One of the most important settings for physiological signals is the intensive care unit. The intensive care unit, or ICU, is the hospital unit responsible for patients with life-threatening conditions. These patients are kept under close watch to ensure their return to good health.

In the past much of patient monitoring within the ICU was done by doctors and nurses, who manually recorded vital signs such as pulse rate, body temperature, respiration rate, and blood pressure with few electronic aides. As technology evolved, more tools became available, capable of recording an ever increasing amount of signals. With the advent of the microprocessor many of the measurements became automated, thus reducing the burden of data acquisition, storage, and computation on the doctors and nurses. Despite these improvements, a substantial bulk of the work still remains on the doctors' and nurses' shoulders, as it is their job to interpret the acquired data.



Because human resources are limited, there is a genuine demand to automate some of the monitoring tasks to aid physicians and nurses with their jobs [Shortliffe & Perreault, 2001, Chapter 13].

To this day, the major successes in the monitoring front have come in the area of arrhythmia monitoring, which, in the most general definition, involves the automatic detection and classification of irregular rhythms of the heart. The basis for this kind of monitoring is the electrocardiogram, or ECG, which describes the electrical activity of the heart[1]. Because the ECG is a relatively clean signal with a countable number of features, it is possible to detect and classify arrhythmias with a high level of confidence by extracting features from the ECG and matching them with previously established heart beat patterns [Weinfurt, 1990]. Even in the presence of noise, arrhythmia monitoring is quite successful because a redundant amount of information is available from the multiple leads used in the ECG [Shortliffe & Perreault, 2001, pp. 451-465].

Unfortunately, the success of arrhythmia monitoring has not yet been carried over to general-purpose physiological monitoring systems. This is mainly due to the fact that other signals measured in the ICU (such as blood pressures and blood oxygenation measurements) are much noisier than ECG signals and do not have clear-cut patterns that distinguish good segments of data from bad ones. The present solution for monitoring of these signals is to establish numerical thresholds that define the acceptable ranges for the data. Whenever the signals depart from the established region of normality, the nurses/doctors are notified. These alarms are often unreliable, resulting in false positives that call nurses over to the patient's bedside unnecessarily. Some studies have shown that false alarm rates in the ICU are as high as 80 to 90 percent [Tsien & Fackler, 1997]!

In addition to improving automated monitoring, there is a continual interest for developing better data visualization schemes. Current monitors use numerical displays for the measured signals and in some models also include a history of the signals displayed as a time-series. These monitors may even give additional information about the present trajectory of the signals (increasing or decreasing) along with the slope of the trajectory. With the ever increasing amount of information displayed to doctors and

[1] For more details on the electrocardiogram, see Section 2.3.1.

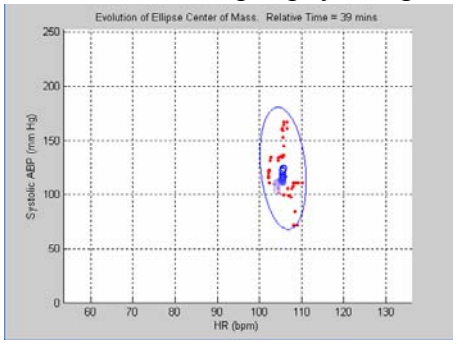
nurses it is important develop displays that convey more meaningful information to nurses and doctors in the ICU to reduce the amount of data cluttering the monitors.

In an effort to help address the necessity for more sophisticated monitoring systems, a bioengineering research partnership (BRP) made up of several MIT groups, Philips Medical Systems, and Beth Israel Deaconess Medical Center, was established to develop and test a patient monitoring system capable of improving the precision and efficiency of clinical decision-making in the ICU. Some of the major issues being addressed by the BRP project include data acquisition, database annotation and development, signal processing, signal modeling and estimation, data visualization, and clinical and expert reasoning systems. Recent BRP work can be found in [Abdala et al., 2004], [Saeed et al., 2002], [Saeed & Mark, 2000], [Douglass et al., 2004], and [Zong et al., 2004] as well as on the BRP website (http://mimic.mit.edu/).

1.2 Goals of Thesis

The work within this thesis is aimed to supplement the ongoing efforts within the signal modeling and estimation initiative, whose goal is to develop quasi-static and dynamic models of the cardiovascular system to better understand the underlying physiology of patients in the ICU. Furthermore, this thesis devotes time to the area of data visualization for data from the ICU. Specifically, this work had four major contributions:

(1) To present a concise description of the types of physiological signals available in the ICU and to explain what these can tell us about the patient.
(2) To investigate and present new methods for visualizing data from the ICU, both in the time and frequency domain.
(3) To develop, implement, and analyze a method to cluster multiple physiological variables in their phase space.
(4) To explore and offer some practical applications of the clustering method within the scope of the intensive care unit.



1.3 Thesis Outline

This thesis is divided into six chapters.

Chapter 2 gives a detailed description of the signals typically measured in the ICU. In addition to describing what the signals can tell us about the health of the patient, we discuss the nominal ranges of the signals and the different temporal resolutions of the data. We then go into a brief discussion about noise in data and highlight some methods for artifact removal and noise attenuation. The chapter concludes with a section on data visualization.

Chapter 3 explores the area of data visualization in more detail and also sets the stage for the analysis done in Chapter 4. By way of a case study we introduce the power and practicality of data visualization using two specific techniques. The first method, phase space plotting, looks at time-domain characteristics. Following a detailed description of phase space analysis, we introduce the application of power spectrum analysis as a useful frequency domain technique for variability analysis of physiological signals.

Motivated by phase space visualization from Chapter 3, Chapter 4 introduces a novel data clustering method based on the singular value decomposition (SVD). After reviewing the basics of the SVD, we describe in detail the clustering algorithm. We conclude with a discussion about the effectiveness of the clustering method and address some of its limitations.

Chapter 5 expands on SVD clustering and explores some potential applications of the clustering method within the scope of the intensive care unit. Finally, Chapter 6 concludes this thesis and presents directions for future work.

Chapter 2

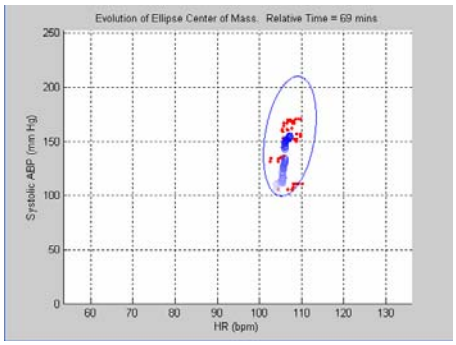
Physiological Signals from Intensive Care

2.1 Introduction

In this day and age, it is almost certain that the reader has come in contact with some device to measure signals emitted from his/her body. Perhaps the signal was body temperature to check for a fever. Maybe it was arterial blood pressure to check for high blood pressure. In either case, a measurement was taken of some signal generated by the body and an assessment about health was made based on the measurement.

If we think of the body as a large, complicated system, we can view the recorded measurements as samples from this system, giving insight into what is happening inside the body at a specific point in time. These signals are important because they provide a direct means of probing the body and observing its behavior under different conditions, such as illness or disease. With these measurements, changes in the body can be mapped to changes in signals. Ultimately the measured values can then be used to establish baselines for normal signal behavior and to make assessments about the current health of an individual.

Because the system governing our bodily functions is very complex, it is impossible to understand exactly how the whole body works together. Fortunately, it is often possible to make good assessments based on relatively few signals measured from the body. For this reason, bodily signals are invaluable for



understanding the conditions of human beings.

The use of physiological signals is not limited to evaluation of the current health of an individual. These signals can also be used to predict the future health of a person, to measure the effects of medications, and to develop models that mimic the behavior of certain bodily functions. All of these applications are important and are actively being researched.

One of the most important places where physiological signals are used on a daily basis is at the intensive care unit. In the ICU, signals are measured regularly to monitor the present condition of patients. Depending on the values of the measured signals, patients may be given treatment in order to improve their condition.

Although the exact collection of signals recorded in ICUs may vary, there are a few signals that almost always get measured. These typically include electrocardiogram (ECG) readings, heart rate measurements, blood pressure (both arterial and venal), respiration rate, and blood O_2 saturation. These signals, measured using specialized equipment, are by no means perfect (as will be described in more detail later in this chapter). However, even in the presence of noise, the signals can provide sufficient information for doctors to make decisions about what treatment is best for their patients.

This chapter gives a brief overview of the types of data used during the course of this project. After covering the background on the signals, we briefly discuss the issues of noise and signal artifacts. We present some techniques to reduce this noise and conclude with a brief look into data visualization techniques.

2.2 Signals on Different Time Scales

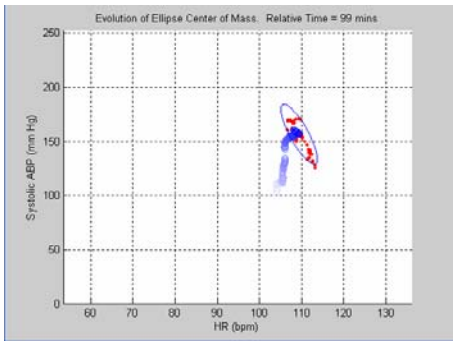
Before getting into the specifics about the signals explored in this thesis, it is first important to understand that patient data can contain a varying level of detail about the underlying processes occurring within the human body. This level of detail is primarily dictated by the rate at which the underlying processes are sampled and by the amount of averaging performed on the sampled signals. In other words, depending how often we take measurements of particular signals and how much of this data we average together over time, we can vary the type of detail that we capture about the signal.

In this work, three types of signal detail are explored: waveform, beat-by-beat averaged, and trend data. While waveform data consists of signals recorded at a rate of 125 samples per second (or 125Hz) and trend data captures signal information at a much lower rate of 1 sample per minute (or 1/60 Hz). More specifically, trend data is derived by averaging waveform data, or selected features thereof, over consecutive one minute intervals.

The waveforms examined in this thesis came from bedside monitors manufactured by Philips Medical Systems. Trend data was derived from the waveforms using algorithms implemented by Philips. Although the specific algorithms are proprietary and not at our disposal, we do not think that the signal quality is assessed prior to single-channel averaging of the waveforms or feature extraction. This in part may contribute to noise in the trend data, explained in more detail in Section 2.4.

In between these two ranges of signal resolution is beat-by-beat averaged data. This data is derived by averaging waveforms, or extracting features, over a single cardiac cycle—the sequence of events that occur over one heart beat—using open-source algorithms created by Zong et al. (2003) and Zong, Moody, and Jiang (2003). To derive beat-by-beat heart rate data, we use the algorithm *wqrs*, which detects and annotates the onsets of the QRS complexes in a single-channel ECG. Beat-by-beat systolic, mean, and diastolic arterial blood pressures are calculated using *wabp*, which detects and records the onset of each blood pressure pulse. Prior to applying these algorithms, artifacts are rejected by visual inspection. Finally, in order to make the signals more processing-friendly, they are then resampled to an integer sampling rate. In our case, we chose to resample at 2Hz.

Knowing that different levels of detail exist, it is also important to understand that one form of data is not necessarily better than another. Their use depends on the type of information one is trying to explore. For example if one is interested in developing an algorithm for accurately detecting ventricular fibrillation—characterized by irregular and uncoordinated activity of the ventricular muscle fibers leading to the inability to pump blood to the body [Berne &



Levy, 2001, pp.50]—it makes more sense to look at electrocardiogram (ECG) measurements, instead of trying to observe heart rate trend data. However, if one is more interested in detecting long-term trends—for example how heart rate and blood pressure of a patient vary over several hours—it makes more sense to look at trend data, which gets rid of the finer details but preserves the slow-acting processes governing patient condition.

Having discussed the details of data on different time scales, it is now possible to get into the specifics of the typical signals measured in the ICU.

2.3 Physiological Signals from the ICU

It should be clear that there is an enormous value in physiological signals. However, in order to derive the useful information from these signals, it is first important to understand what they tell us about the condition of the patient, how they are measured, and over what values they range. Excellent resources for more in-depth information are the texts written by Berne and Levy (2001), Martin (2004), and Marino (1998).

2.3.1 Electrocardiogram

The electrocardiogram (ECG) is a signal measuring the electrical activity of the heart. It is measured by placing a set of electrodes on the body and recording electrical impulses emitted by the heart.

At a bare minimum, the ECG is usually recorded by placing three bipolar electrodes (meaning they measure a change in electric potential between two points) on the left arm, right arm, and left leg. With this setup, the differentials between the left arm and right arm, left arm and left leg, and right arm and left leg can be measured. These three differentials form what is often called Einthoven's triangle, named after the Dutch doctor who first described the relationship between the differentials [Berne and Levy, 2001, pp. 44].

In addition to these 3 signals, 3 more measurements can be recorded by referencing the individual electrodes to a null point. The null point is calculated by the sum of the potentials from the other two electrodes. Because the ECG measures six

different signals, it is sometimes called a 6-lead ECG. Beyond 6-lead ECGs are 12-lead ECGs, which place another six unipolar electrodes over different locations of the chest. An example of the ECG is shown in Figure 2.1 [Yanowitz, 2005].

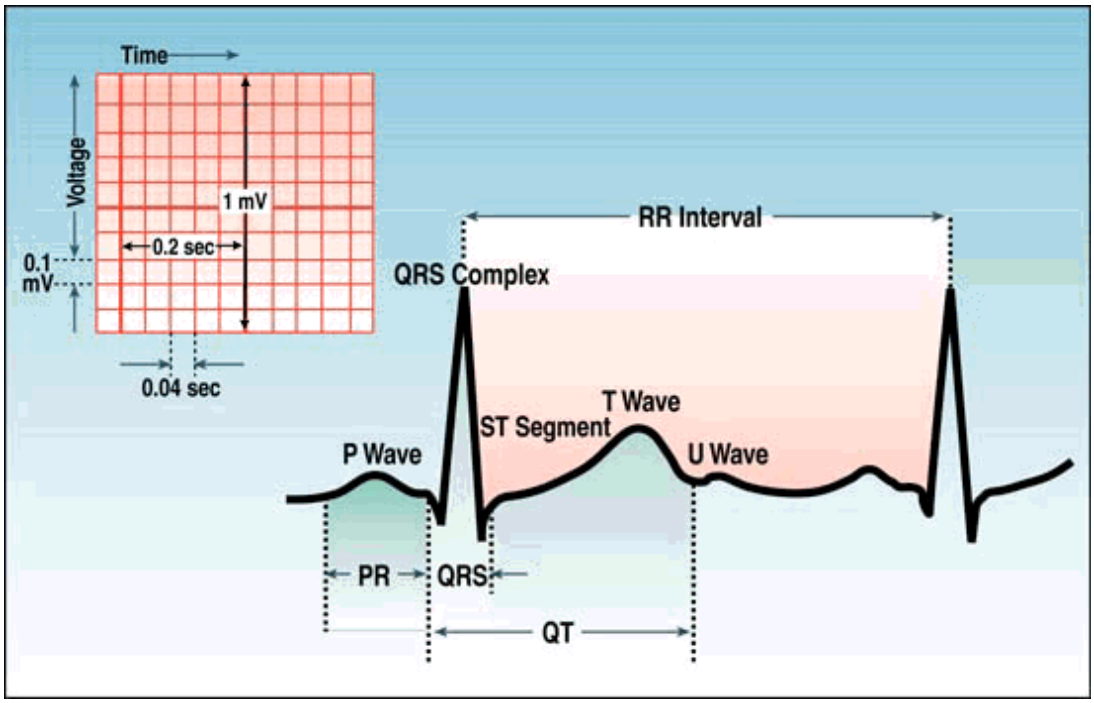
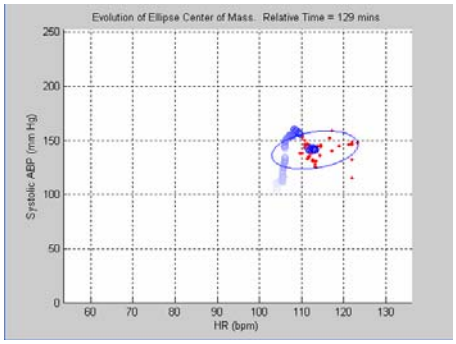


Figure 2.1: An illustration of the ECG waveform [Yanowitz].

As shown in the figure, the ECG waveform consists of several parts. These include the P wave, QRS complex, ST segment, T wave, and U wave.

The P wave describes the spread of the electrical activity across the atria of the heart, responsible for pumping blood into the ventricles. During the P wave the atria begin to contract, sending blood into the ventricles via the mitral and tricuspid valves (found at the exit of the left and right atria, respectively).

Following the P wave there is a period of time during which the ventricles fill up with the blood from the atria. This segment is then succeeded by the QRS complex, which describes the depolarization across the ventricles. A short time after the QRS complex, the right and left ventricles contract, pumping oxygen-rich blood to



the body through the aortic valve and oxygen-poor blood to the lungs via the pulmonary valve.

Blood continues to be ejected from the ventricles during the ST segment, representing the re-polarization of the ventricles. Some time after the T wave has occurred, the ventricular exit valves close and the heart enters its relaxation period (also known as diastole), allowing newly-circulated blood to enter the atria in preparation for the next contraction. Finally, the U wave, although not completely understood, is believed to represent some further depolarization occurring in the ventricles. This entire cycle is summarized in Figure 2.2 [From Berne and Levy, 2001].

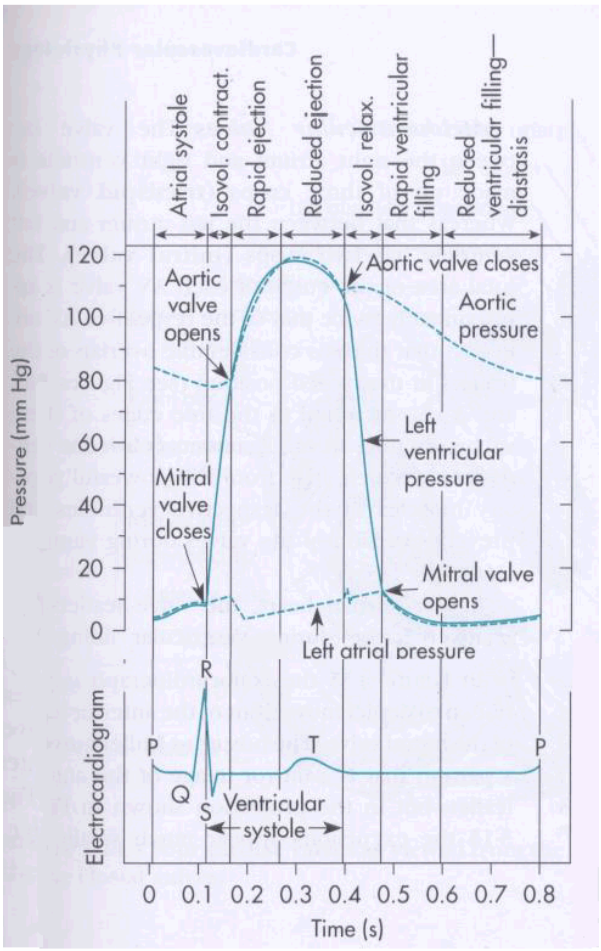


Figure 2.2: The link between the ECG and the mechanical activity of the heart. The top plot annotates the mechanical activity of the heart, reflected in the left atrial and ventricular pressure waveforms. The bottom plot shows the corresponding electrical activity of the heart (the ECG) [Berne and Levy, 2001].

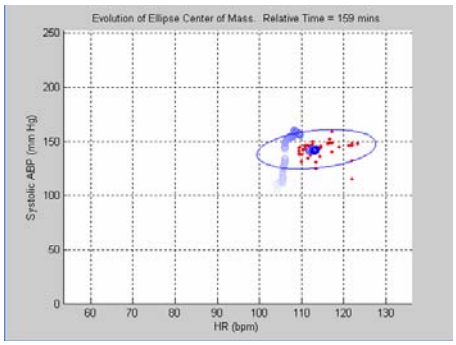
The ECG provides an incredible amount of information about the on goings of the heart. Because of its rich structure, the ECG is helpful for monitoring many cardiovascular conditions including heart arrhythmias[2] and for investigating heart disease.

2.3.2 Heart Rate

Heart rate is a description of the frequency at which the heart beats and is quoted in beats per minute (bpm). Although the values of heart rate can vary depending on physical fitness, current level of activity, illness, and many other factors, typical values of heart rate range from around 70 beats per minute at rest to values well over 100 (180 bpm is not an unreasonable value) during periods of activity or illness.

A typical heart rate signal for a patient in the ICU is shown in Figure 2.3. Because the heart rate signal is already in the form of one sample per minute, it is clearly trend data. The signal is typically derived by counting the number of QRS complexes in the patient's electrocardiogram waveform over the period of one minute [Bianco, 2004]. The reason for using the QRS is because it is the most prominent feature of the ECG, making it the easiest to detect.

Heart rate signals like the one shown in Figure 2.3 can give us a hint about the long-term variations of heart rate activity, cycles of activity and rest, and potential cases of certain heart arrhythmias.



[2] Arrhythmia – An irregularity in the rhythm of the heart.

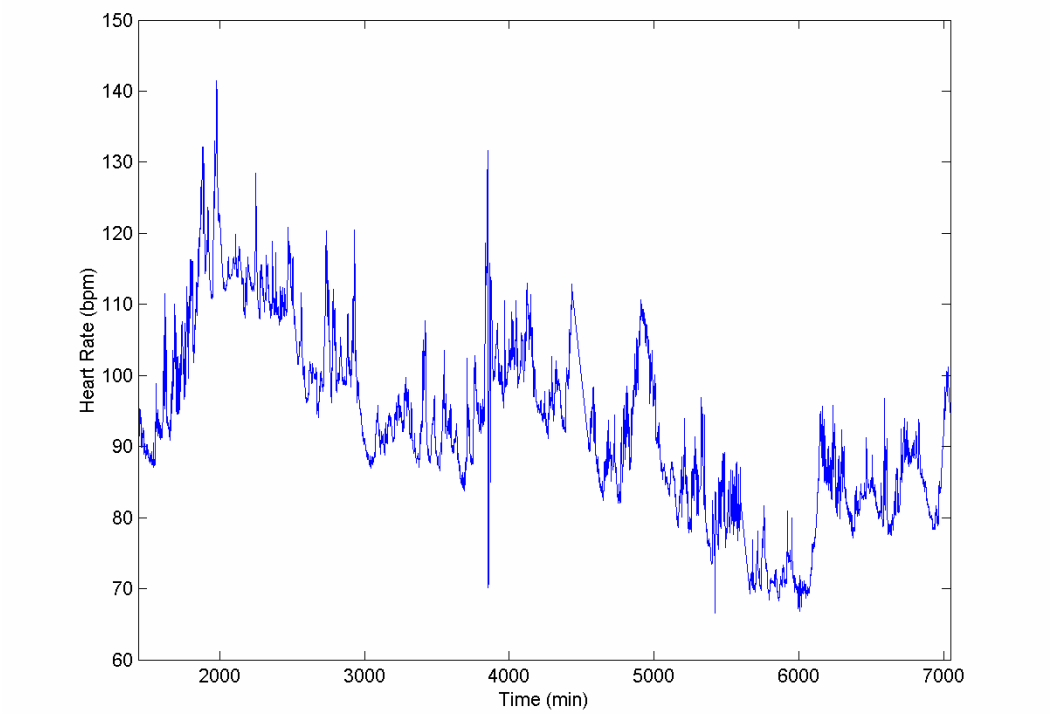


Figure 2.3: Typical heart rate signal for a patient in the ICU. This plot contains approximately 3.5 days of data.

2.3.3 Arterial Blood Pressure

Arterial blood pressure (ABP) is a measure of the effort required to pump blood from the heart into the aorta, the main trunk of the systemic arteries responsible for transmitting oxygen-rich blood from the left ventricle to the rest of the body. Measurements of ABP give insight into cardiovascular function. In particular, they can tell us how hard the heart is working to pump blood throughout the body; they also provide an indication of whether or not organ perfusion is adequate.

An example of the ABP pressure waveform is shown in Figure 2.4. This signal was measured by placing a fluid-filled tube inside one of the major arteries in the body (typically the radial artery) and bringing the fluid column in contact with a pressure transducer, which converts the pressure measurement into a voltage. ABP can also be measured non-invasively, but is typically not as accurate as invasive methods.

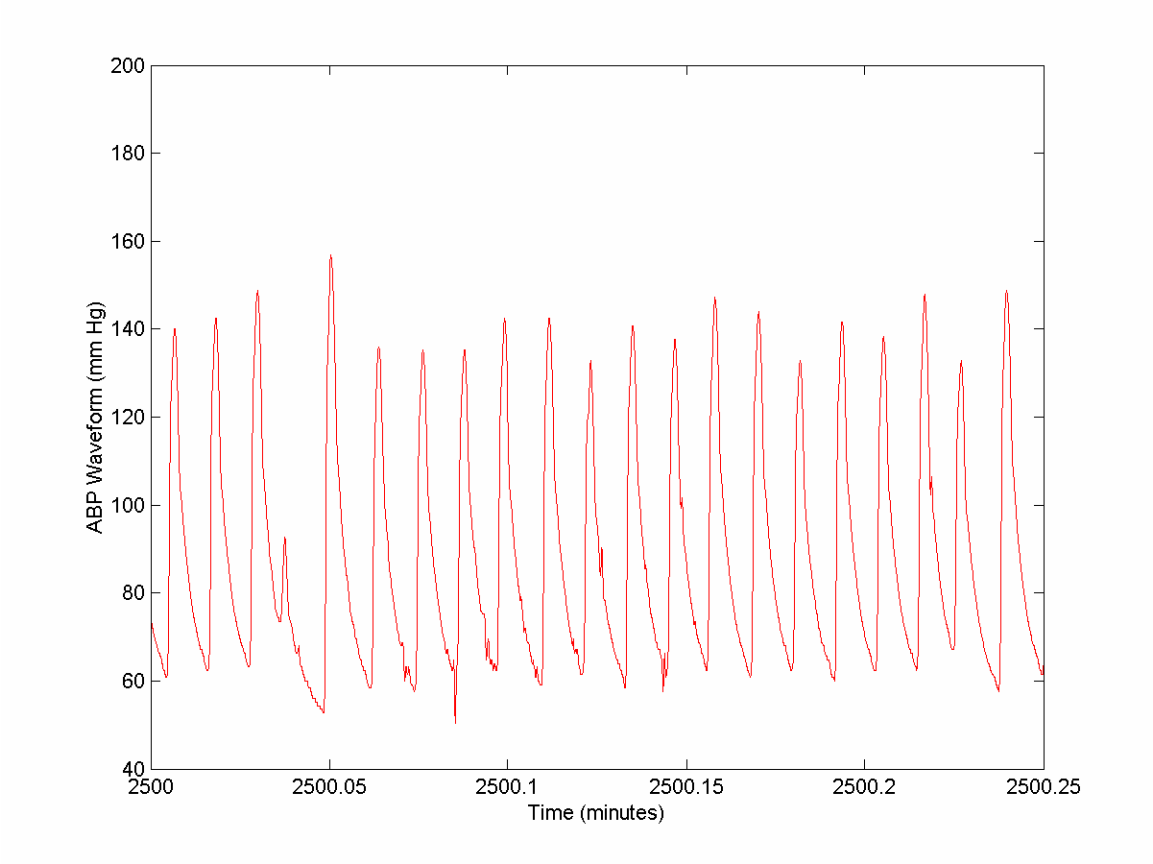
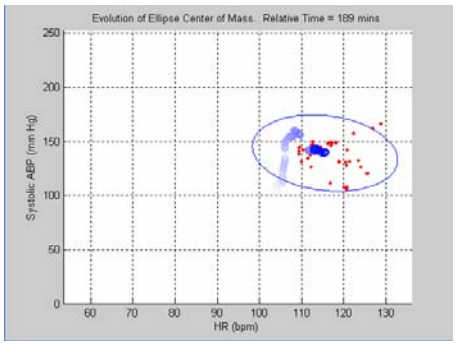


Figure 2.4: Example of an arterial blood pressure waveform.

Because the cardiac cycle is quasi-periodic, the pressure waveform is also quasi-periodic. For each cycle, the peak occurs during contraction of the heart muscle and the trough appears during cardiac relaxation. These maxima and minima are called the systolic and diastolic pressures, respectively. Typical values range from 100 to 140 millimeters of mercury (mm Hg) for systolic pressure, and 60 to 90 mm Hg for diastolic pressure. In individuals with obstructed arteries, these pressures can be significantly higher.

In addition to the systolic and diastolic pressures, two other derived blood pressures are pulse pressure and mean blood pressure.[3] Pulse



[3] Systolic, mean, diastolic, and pulse pressures are not unique to the ABP. They can also be found for the central venous and pulmonary artery pressure waveforms, which will be described later in this section.

pressure is defined as the difference between systolic and diastolic pressure. Mean blood pressure is calculated by finding the average value over one cycle of the blood pressure waveform. However, to a very good approximation, the mean of the arterial blood pressure waveform can also be calculated by the following equation,

$$ABPM = \frac{1}{3} ABPS + \frac{2}{3} ABPD$$

where ABPM, ABPS, and ABPD are the mean, systolic and diastolic arterial blood pressures [Berne & Levy, 2001, pp. 142]. Typical values for mean blood pressure are in the range of 70 to 105 mm Hg. Figure 2.5 shows an example of the systolic, mean, and diastolic ABP signals plotted together.

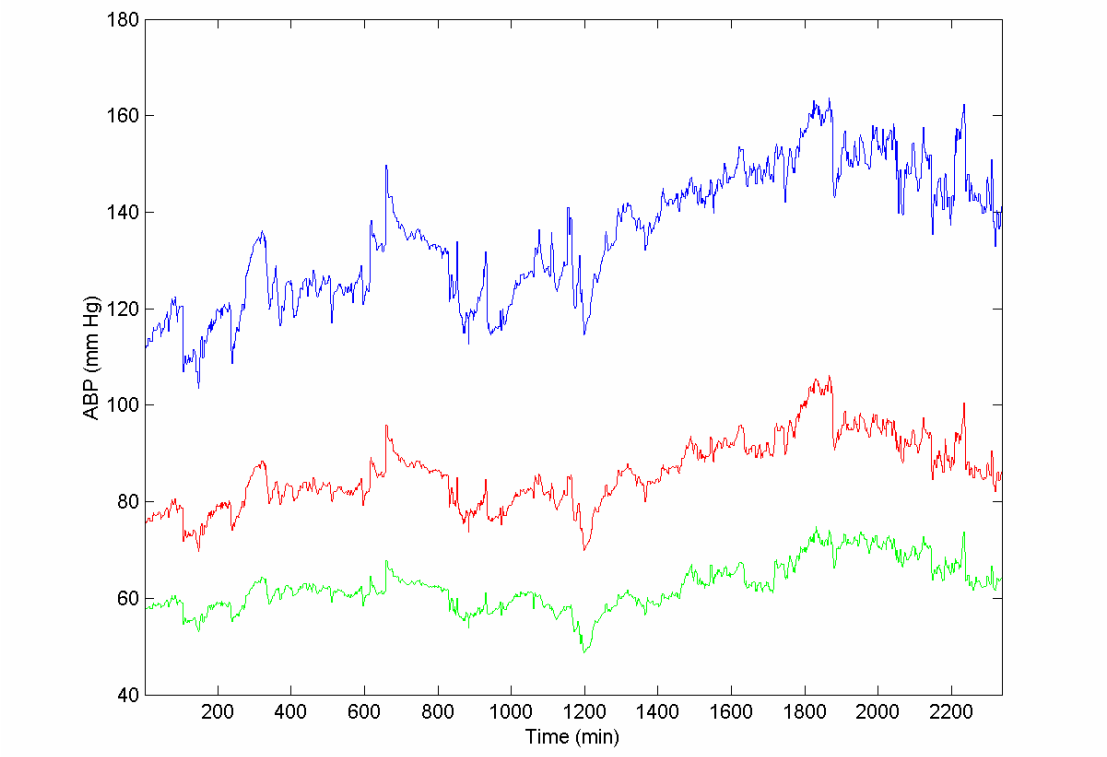


Figure 2.5: An example of trend data for ABP over a stretch of approximately 1.5 days. The trends (from top to bottom) denote systolic, mean, and diastolic pressures.

One thing to note about Figure 2.5 is that the systolic, mean, and diastolic pressures are highly correlated in the positive sense. That is, changes in one of the

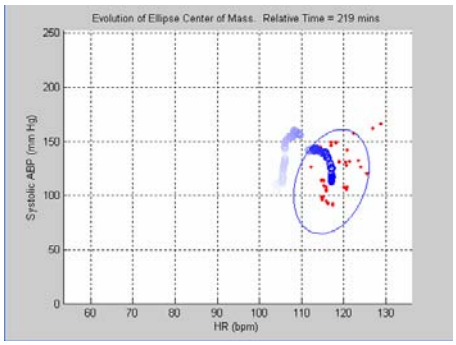
variables are reflected positively in the other two. If the systolic pressure goes up, for example, the diastolic and mean tend to increase also. Changes in the degree of correlations in the signals are good indicators that either (1) there is noise or artifacts present in the segment of data or (2) the patient state may be changing.

2.3.4 Diastolic Pulmonary Artery Pressure

While measurements of arterial blood pressure tell us about the load the left ventricle experiences, the pulmonary artery pressure measurements can be used to assess the pressure in the left atrium, which is responsible for pumping oxygen-rich blood from the lungs into the left ventricle. This measurement is useful, for example, for checking of left heart failure in patients.

Pulmonary artery pressure is measured by placing a catheter into one of the main pulmonary arteries of the heart. To reach its final destination, the catheter begins at the subclavian or internal jugular veins (near the neck), travels via the great veins into and through the right atrium and right ventricle, and is finally placed in one of the pulmonary arteries.

Because the pulmonary arterial resistances are much less than their systemic counterparts and because the right heart generates much lower pressures than the left heart, pulmonary artery pressure is much lower than arterial blood pressure. Typical values range from 15 to 28, 10 to 22, and 5 to 16 mm Hg for systolic, mean, and diastolic pressures, respectively. Figure 2.6 shows a plot of the diastolic PAP trend waveform.



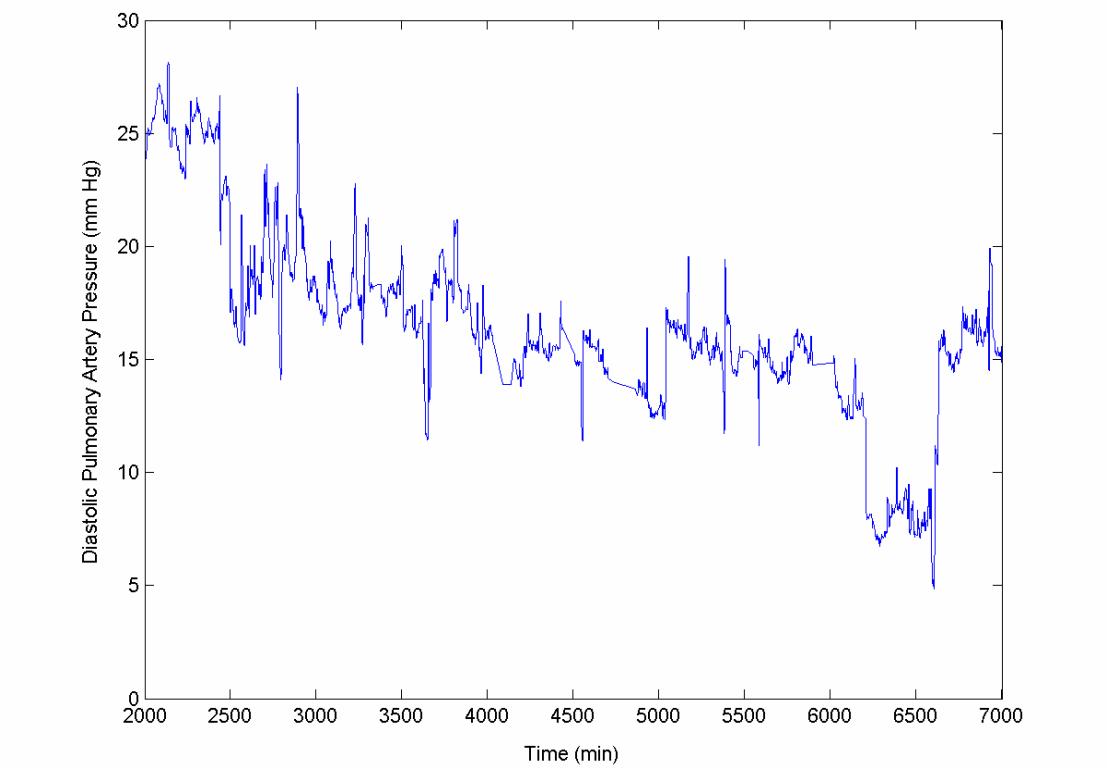


Figure 2.6: Diastolic pulmonary artery pressure trend data. The plot contains approximately 3.5 days of data.

2.3.5 Central Venous Pressure

Central venous pressure (CVP) is a measure of the pressure in the thoracic vena cava, which is near the right atrium. Like the other pressures discussed so far, the CVP is measured using a catheter which is typically guided into the body through one of the veins in the neck. An approximation of right ventricular end diastolic pressure[4], the CVP gives insight about right ventricular function (Recall that the right ventricle is responsible for sending oxygen-deficient blood to the lungs, where oxygen is picked up and carbon dioxide is released). Typical values for CVP are less than 10 mm Hg and greater than 1 mm Hg [Martin, 2004].

Figure 2.7 shows a plot of central venous pressure trend data. Note that although some of the data is within the typical range of values for CVP, there is a lot of data slightly over this range. This could be due to some genuine patient condition or a loss of

[4] Right ventricular end diastolic pressure – The pressure exerted on the right ventricle at the end of ventricular filling.

calibration in the measurement equipment. In general, CVP signals are not as reliable as other hemodynamic measurements because they are sensitive to, for example, movements by the patient. These changes introduce baseline offsets into the signal that must be corrected through regular recalibration of the equipment [Faucy et al., 1998].

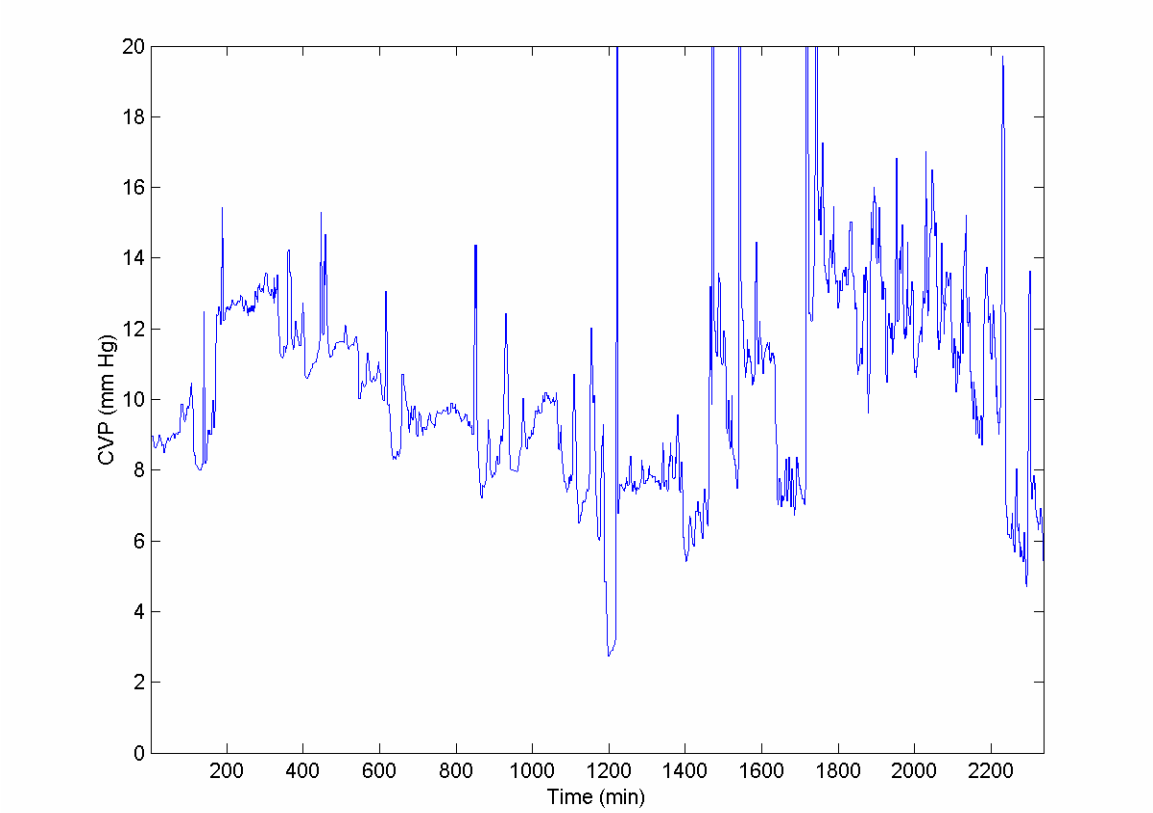
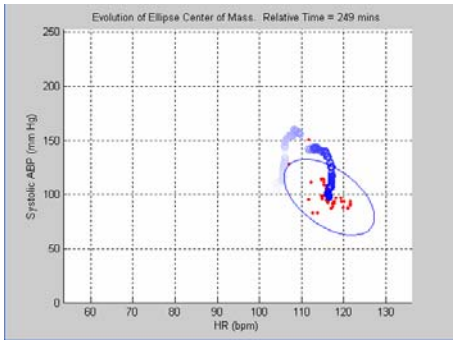


Figure 2.7: Central venous pressure trend data.

2.4 Signal Noise and Artifacts

As is evident from the plots in the previous section, physiological signal measurements are not perfect; they all contain some degree of noise as well as signal artifacts. Noise can be introduced into signals in many different ways. For example, imperfections in the measurement equipment (such as the probes and processing hardware) and improper use of the tools (for example: the poor placement of ECG electrodes or arterial lines) can lead to noisy measurements. Furthermore, noise can be



introduced by poor performance of signal processing software (such as false ABP detections), sudden movements by the patient, or even by the patient physically removing probes from his or her body.

There is an important distinction to be made between signal noise and signals artifacts. Whereas noise in the scope of this thesis refers to *any* undesired uncertainty of the signal, signal artifacts are noise instances in a signal, which can be confused for actual physiological events. In other words, signal artifacts are a subset of signal noise. In medicine we are most concerned with signal artifacts since these can lead to incorrect interpretations of the data. Any of the sources of noise mentioned above can potentially generate signal artifacts.

In some cases artifacts are easily detected. For example, if one sees the CVP signal jump to a value of 200 mm Hg, this is clearly an artifact. In other cases, however, noise artifacts can be much harder to spot. As an example, if we were to look at the heart rate trend data in Figure 2.8, it is hard to say for sure whether the spikes around minutes 2000 and 4000 are genuine or artifactual. The only way to check would be to look at the underlying ECG data.

2.5 Methods for Reducing Noise and Removing Signal Artifacts

In exploring the different types of signals, we experimented with several methods for detecting and eliminating some of the noise artifacts. The two methods we found most effective were threshold detection and median filtering.

2.5.1 Threshold Detection

For any patient signal, there are limits on the ranges of values that the signal can take. Any values exceeding these limits are likely due to noise artifacts. Care has to be taken to ensure that the limits are large enough to take into account extreme values that may arise in the population distribution.

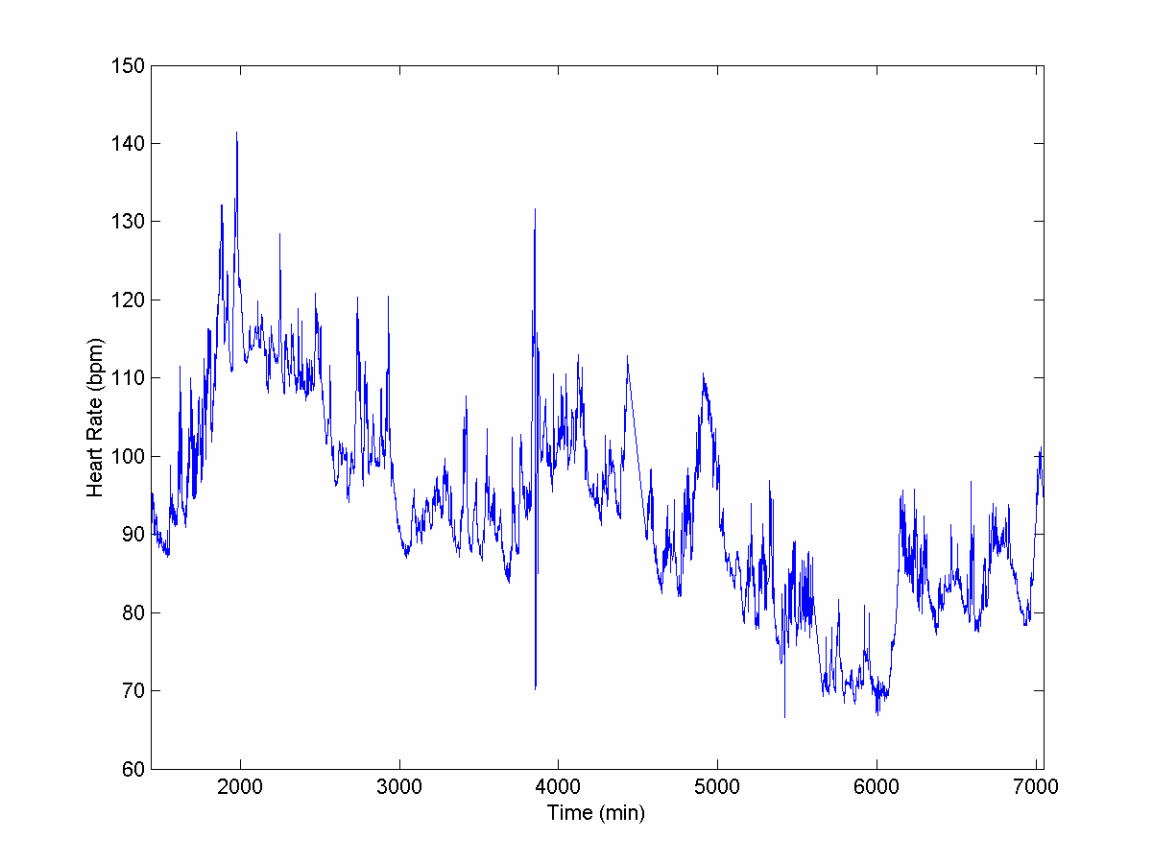
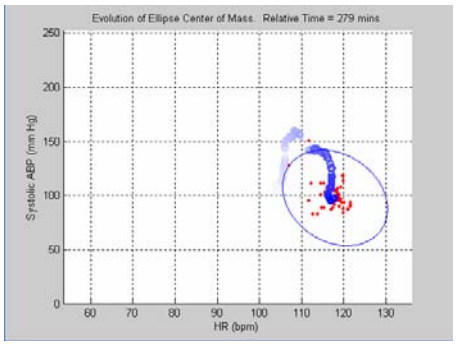


Figure 2.8: Example of heart rate trend data with potential noise artifacts.

A good example of where threshold detection comes into play is CVP trend data. As mentioned earlier in this chapter, the CVP measurement is very sensitive to noise. Even natural body functions such as breathing or movement can shift the baseline of the signal to unreasonably high or even negative values[5], as shown in Figure 2.9. Because CVP cannot be significantly less than zero and is usually less than 10 mm Hg, regions where the signal is exceeding these bounds are highly questionable. Therefore it is better to ignore these regions when any form of analysis is done.

Bounds for the signals, as suggested by Wong (2003), are shown in Table 2.1. These bounds are not very tight, but ensure that any values outside these ranges are artifactual.



[5] Referenced to atmospheric pressure.

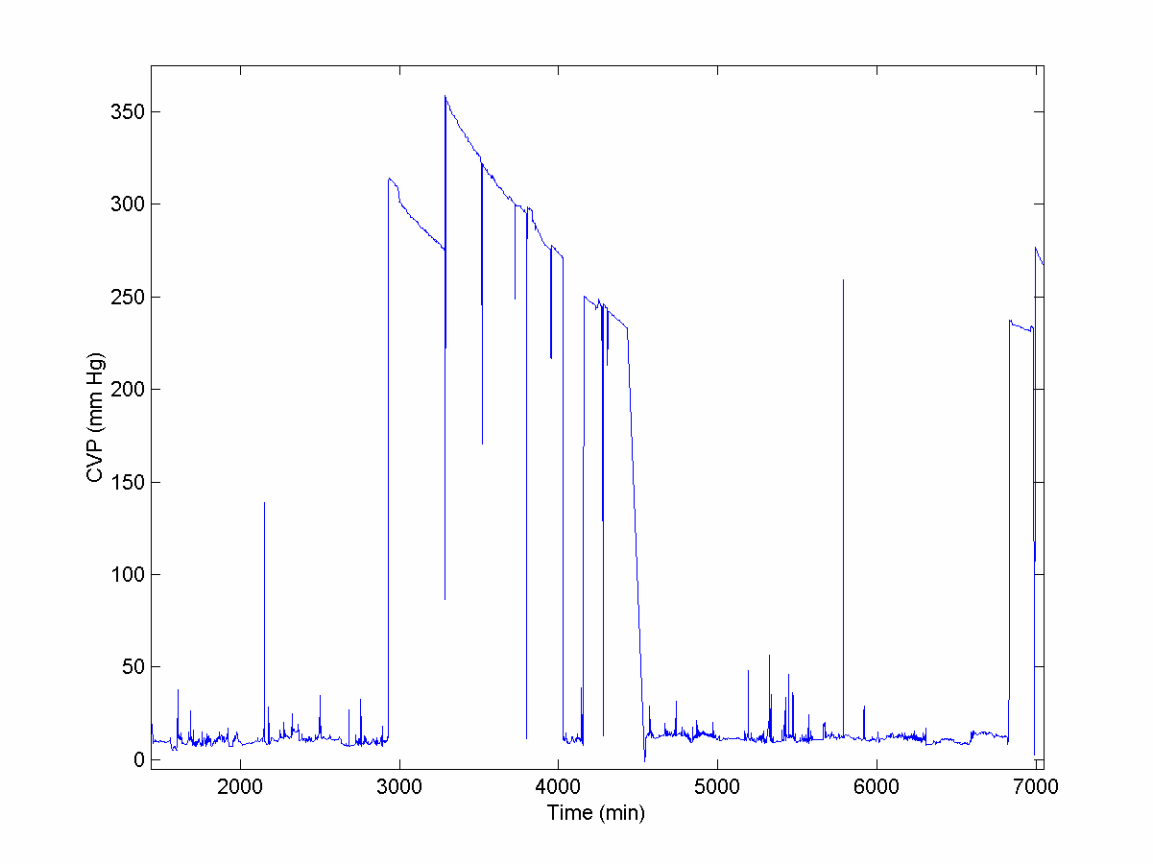


Figure 2.9: CVP trend data with many noise artifacts.

Table 2.1: Suggested limits for different physiological signals

Signal	Upper Bound	Lower Bound
Heart Rate	250 bpm	20 bpm
ABP (Sys, Mean, Dia)	300, 250, 200 mm Hg	0 mm Hg
PAPD	50 mm Hg	0 mm Hg
CVP	30 mm Hg	0 mm Hg

2.5.2 Median Filtering

Another useful method for removing some of the noise in the trend data is median filtering. This technique has been successfully applied in many signal processing and image processing applications.

Median filtering is a nonlinear signal processing technique, which looks at each point in a set of data along with a window of points around it. The median of the window is calculated and its result replaces the value of the point in the middle of the window. The key benefit of median filtering is its ability to remove outliers in the data while still preserving the structure of the signal.

Like any other type of filter, there is a tradeoff between noise removal and signal distortion. In the case of median filtering, this tradeoff is addressed in the choice of window length. Figure 2.10 illustrates an example of median filtering. In the top panel of the figure it is almost impossible to differentiate the original signal from its median-filtered version. However, if we take a closer look, as shown in the bottom panel of Figure 2.10, we see that outliers are removed from the data with minimal distortion of the original signal shape. Whether the spike in the bottom panel is due to an arrhythmic event can only be determined by analyzing the corresponding ECG.

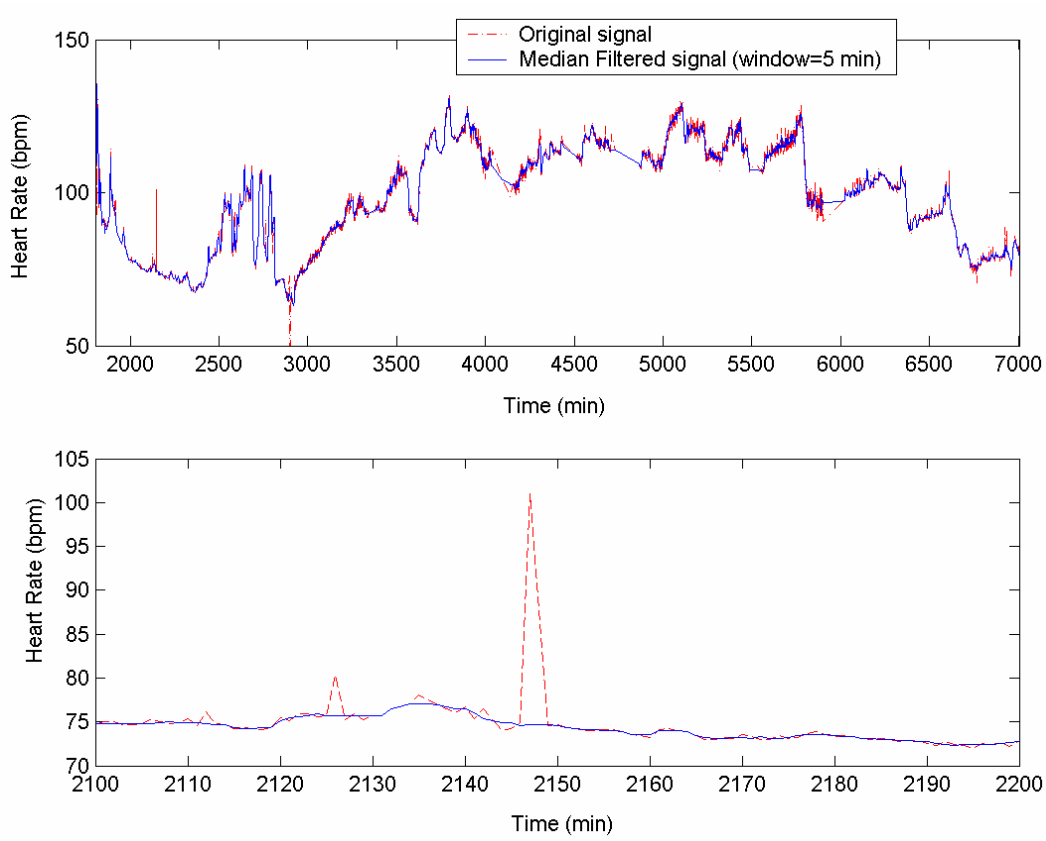
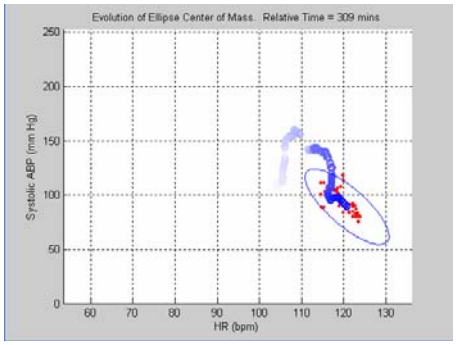


Figure 2.10: Effects of applying median filtering to a heart rate signal. The top plot shows the entire signal and the bottom plot shows the effects of filtering on a shorter time scale.



2.6 Data Visualization Techniques

With all the data recorded in the ICU, one question that arises naturally is how to best visualize the

recorded information. This section attempts to give some insight into techniques currently being explored in the field.

2.6.1 Time Series Visualization

The most commonly used method for visualizing ICU data is time series plotting. This method involves plotting a dependent variable measured at a specific time against an independent variable, time. Examples of time series can be found throughout this chapter.

Within the domain of time series, there exist different types of time series plots. Three of the most commonly used in the ICU are linear plots, stem plots, and sample-and-hold plots.

Linear plots are often used for continuously recorded variables, such as ECG and blood pressures (Figures 2.3 – 2.10 are all examples of linear plots). On the other hand, stem plots make more sense for measurements that are discrete in nature; such as fluid bolus administered to a patient in need of fluid resuscitation. Finally, in cases where a measurement is in the form of a rate, such as a saline drip or a drug given intravenously, it makes most sense to use a sample-and-hold plot, whose level changes when the drip rate changes. Examples of these three types of plots are shown in Figure 2.11.

2.6.2 Plotting of Functional Relationships

Although time series plotting is a nice way to emphasize temporal correlation among multiple data streams, it is of limited use when one is more interested in the dependencies between variables or the dynamics of the underlying process being observed. In these cases it is more useful to explore functional relationships between signals.

Some examples of functional plots include plots of one variable versus another, plots of one variable against a time-delayed version of itself, and plots of changes in one variable versus itself. Each of these plots have names associated with them (phase space plot, time-delay embedded phase portrait, and phase portrait, respectively). Figure 2.12 shows an example of each of these plots.

Many excellent references on phase space plotting, time-delay embedded phase portraits and phase portraits can be found in texts by Kantz and Schreiber (1997), Jackson (1991), and Abraham and Shaw (1989). Within the scope of ICU signals, work has been done exploiting some of these functional plotting methods ([Saeed & Mark, 2000], [Zimmerman et al., 2003], and [Narayan et al., 2004]).

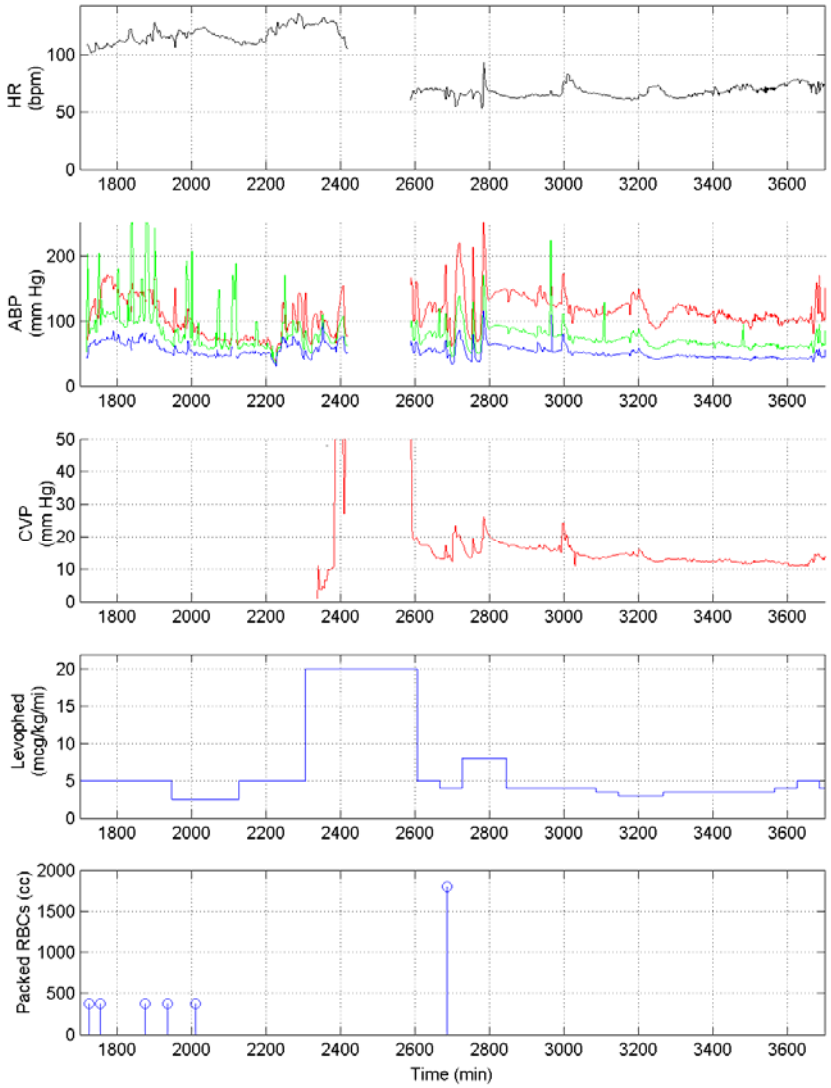
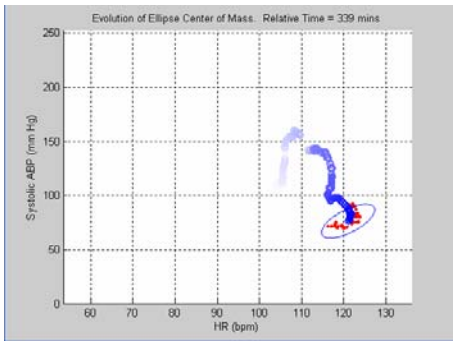


Figure 2.11: Examples of time-series data for a patient in intensive care. The first three plots display measured physiological variables and the last two plots show examples of two fluids (a drug and red blood cells, respectively) administered to the patient.



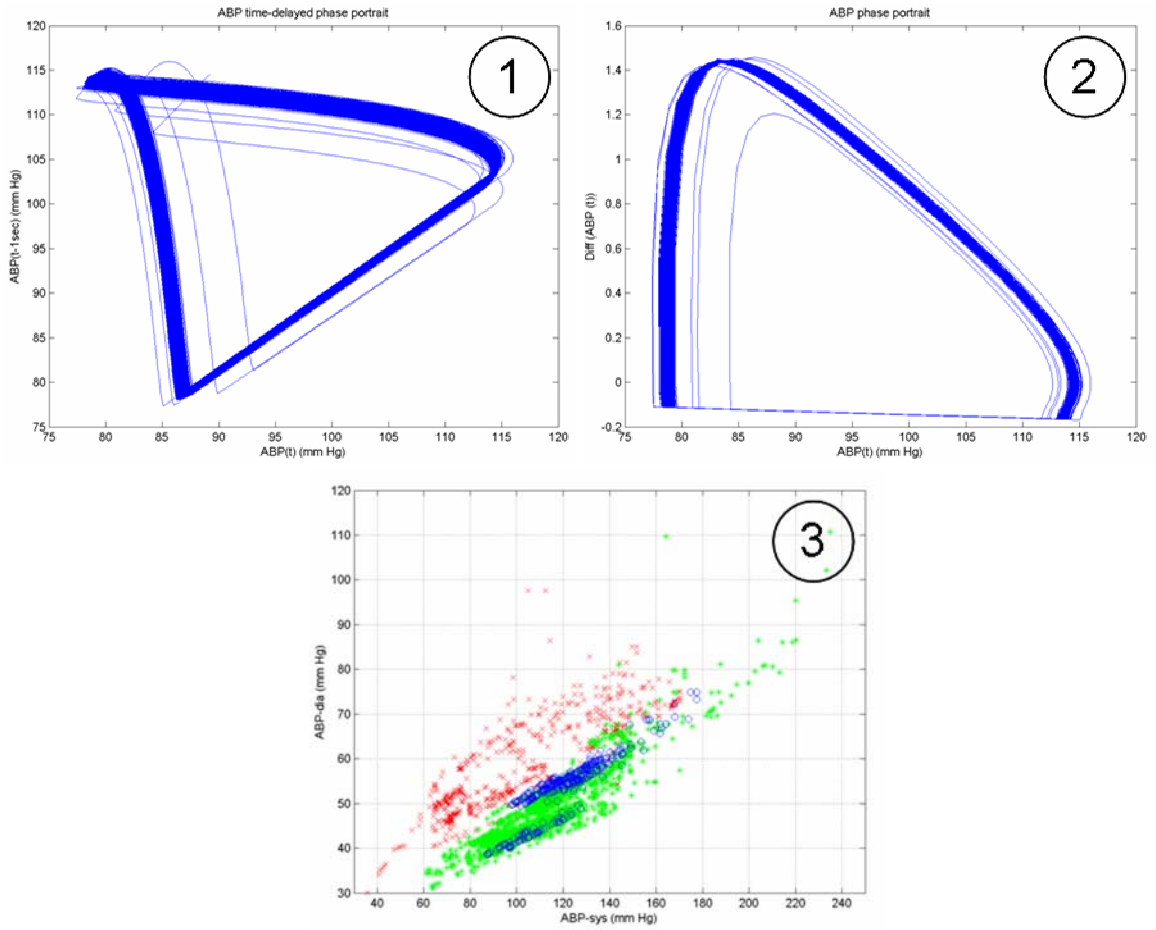


Figure 2.12: Examples of three types of functional plots. (1) A time-delayed reconstruction of a synthetic ABP waveform. (2) The same ABP waveform, but this time plotted versus its first difference. (3) A phase space plot of systolic and diastolic ABP trend data.

2.7 Conclusion

Having presented a brief overview of the signals used in the rest of the project, we are now ready to discuss how we can exploit these signals to make inferences about the condition of a patient. In the next chapter, we present a case study for a patient in the intensive care unit and then make use of two techniques, phase space plotting and power spectrum analysis, to observe and analyze the evolution of the patient's condition during her stay in the ICU.

Chapter 3

Visualization and Analysis of Patient Data

3.1 Introduction

This chapter explores the power of data visualization and analysis as a tool for formulating interpretations about patient data. Specifically, we focus on the application of phase space plotting and spectral analysis to cardiovascular data. With help from a real-world example, we explain the potential benefits of using these analysis techniques.

3.2 Analysis Tools

Before presenting the case study, we introduce the details of the two analysis tools we worked with.

3.2.1 Phase Space Analysis

Phase space analysis is not a new discovery. It has been in use for some time in the field of physics as a tool for viewing trajectories of dynamic systems [Jackson, 1991, Chapt. 2].

In its simplest definition, a phase space is a coordinate system whose axes represent the measured variables of some system. In other words, a phase space is created by taking a set of variables and using these as the axes of a plot.

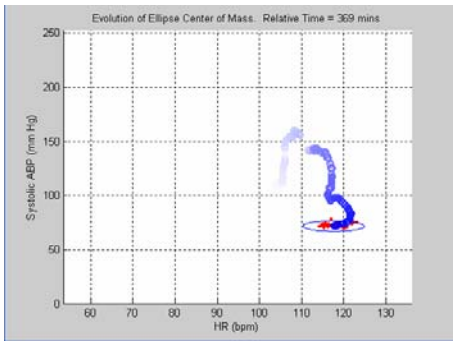


Figure 3.1 shows an example of a phase space formed by three variables.

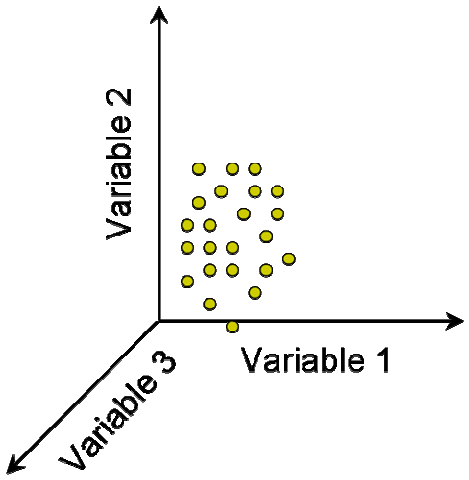


Figure 3.1: A simple example of a phase space formed by three variables.

In plotting a set of variables in phase space, there is an underlying assumption that the variables are all sampled at the same rate. If they are not, it is difficult to make an interpretation. To graph a set of variables in their phase space one must think of the variables sampled at each time step as the coordinates for a point in the space. For example, if the three variables are displacement, velocity, and acceleration, a point in phase space is defined by the coordinates (d(t),v(t),a(t)), where t is the time at which the variables were sampled.

An interesting aspect of the phase space is that by plotting the measured variables against each other, the explicit time dependency is lost in the process.[6] In loosing this time dependency, we gain a new type of information describing the spatial behavior of the variables corresponding to the underlying system. By further noting the sequence in which points are plotted, we gain a description of the trajectories in phase space. It is these features that make phase space plotting particularly attractive for visualization of dynamical systems.

Furthermore, as we will see in the sections that follow, phase space plotting provides a very succinct way of visualizing the dynamic history of the variables being

[6] Although the explicit time dependency disappears from the phase space plot, it is possible to encode time into the phase space by introducing color gradients (darker colors represent more recent data points, lighter shades are older time points), symbols, or numbers into the plot.

observed. Whereas a time series requires one to view multiple streams of variables over (possibly) hours of recorded data, phase space plotting condenses all this information into a single plot. Such a method is very powerful, allowing one to make assessments about changes in a set of variables over some period of time very quickly. Because of its ability to compress so much information into one plot, there is great potential for using phase space plotting in the ICU as a quick means for evaluating the medical history of a patient.

3.2.2 Power Spectrum Analysis

The power spectrum of a signal describes how the power of the signal is distributed over its frequency content. For example, the power spectrum for a cosine with a frequency of 60 Hz will be an infinitely-narrow pulse concentrated around 60 Hz in the frequency domain, as shown in the figure below.

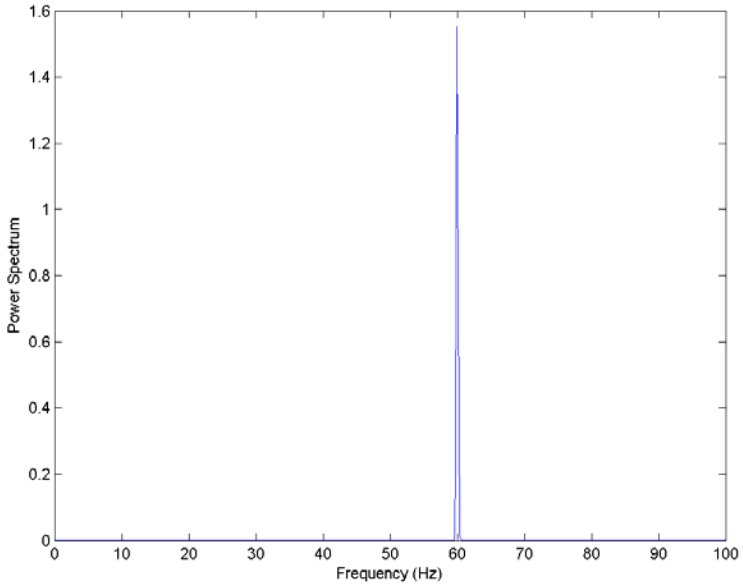
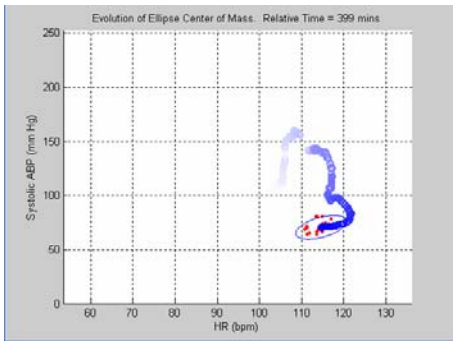


Figure 3.2: Power spectrum of a 60 Hz cosine.

For more complicated signals consisting of many (or possibly an infinite amount of) frequency components, the power of the signal will be spread out over a larger portion of the frequency domain. As we will show soon, power spectrum analysis is a



valuable tool for both modeling and visualization purposes, since it allows us to characterize the variability of physiological signals, like beat averaged heart rate and blood pressure data.

Many methods exist for power spectrum estimation. These can be divided into two major categories: parametric and nonparametric. Parametric methods rely on an underlying model, with parameters that can be adjusted to better estimate the power spectrum. Nonparametric methods do not have a fixed structure and simply require mathematical computation. While nonparametric algorithms are typically simple and computationally fast (since they often involve the FFT), parametric methods usually yield smoother results and can do a good job of estimating the power spectrum even with a small amount of data. One disadvantage of the nonparametric method is that it often requires a large dataset to get good results. Parametric methods, on the other hand, may not always be suitable for the spectra being modeled and can suffer from over-fitting[7] if the order of the model is too high.

In this work, the Welch method for power spectrum estimation was used. Welch's method is a nonparametric algorithm that takes a signal, divides it into multiple overlapping segments, computes the power spectrum for each segment, and then averages together the individual spectra to produce a single power spectrum. Specifics about the Welch algorithm we used can be found in the documentation for the Matlab signal processing toolbox [Mathworks, 2005, keyword: "pwelch"]. For more details about spectral analysis see [Oppenheim, 1999, pp. 730-742].

3.3 A Patient Case Study

To bring the data visualization techniques into the perspective of physiological signals, we now present a case study on a patient who spent some time in the intensive care unit.

The subject, an 83 year old female, first arrived at the hospital after complaining of pains in her left hip and knee caused by a fall she took at her nursing home. Upon arrival, her vitals were taken. All readings were normal (temperature was 97.9˚F; heart

[7] Over-fitting – A phenomena where a model begins to fit the data so well that it starts to capture the undesirable aspects (such as noise) of the signal or process being modeled.

rate was 64 beats/min; respiration rate was 16 breaths/min), with the exception of her blood pressure (158/96), which hinted at possible hypertension[8].

Subsequently, her knee and hip were examined more closely. There was some tenderness over the left knee and the range of motion for the hip was limited. An x-ray was taken of the hip area, revealing a loose acetabular shell[9], with a possible loose femoral head. Because the patient was on anticoagulants due to a recent aortic valve replacement, no immediate surgical action could be taken to correct the problem.

While waiting for the anticoagulants to wear off, the patient began complaining of pains in her lower-right abdomen. On examination, a 6-by-6 cm mass of fluid was found near the area the patient pointed out. Since the last examination the patient's vitals had changed significantly. Her temperature dropped to 94˚F – indicating hypothermia – and her hemotocrit[10] level, fell from 29.3 to 20 to 16 over the course of 2 days. Her blood pressure also fell significantly.

Worried by the low hematocrit levels, the doctors conducted a further examination, which uncovered that the patient was suffering from a bleed in her lumbar artery. She was rushed to interventional radiology to have the bleed embolized and then returned to the ICU for recovery.

During the recovery period, the patient was treated for hypovolemic shock[11]. She was given fluid resuscitation, including several units of packed red blood cells and fresh frozen plasma, to compensate for the blood loss. She was also given levophed, a vasoactive drug, to help raise her blood pressure by constricting the blood vessels.

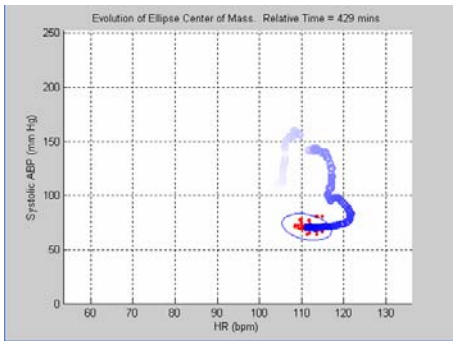
---

[8] Hypertension – Also known as high blood pressure.

[9] Acetabular shell – A cup-like socket in the pelvic bone where the head of the femur (the "thigh bone") articulates.

[10] Hematocrit – A measure of the percentage of red blood cells in the blood. Typical values range from 36 to 42. Because red blood cells are responsible for carrying oxygen in the body, a low hematocrit can lead to anemia – characterized by dizziness, low blood pressure, and shortness of breath – and even death.

[11] Hypovolemic Shock – A condition brought about by a decrease in the volume of blood in the body. Typical indicators of hypovolemia include: tachycardia (high heart rate), hypotension (low arterial blood pressure), low CVP, low pulmonary artery pressure, low hematocrit (if bleeding), high hematocrit (if dehydrated), hypothermia (body temperature below 95 degrees), and reduced pH levels in the body.



Eventually the patient was weaned from levophed and fluid resuscitation and stabilized on her own.

3.4 Plotting Patient Data in Phase Space

3.4.1 Analysis

With an understanding of the patient's background we proceeded to visualize some of the data measured during her stay in the ICU using phase space plotting. We focused on heart rate and arterial blood pressure trend data (1 sample/min), since the CVP and PAPD trends were unavailable for the majority of the stay.

Because we were interested in seeing how different states of the patient affected the trajectories in phase space, we segmented the data into regions. With help from a surgeon—who carefully examined the signals, along with other pieces of information, such as nurses notes, discharge summary information, input-output fluids, medications, and lab results—the data was separated into three states labeled as: "*hypovolemia*", "*intermediate recovery*", and "*steady state*."

The region labeled "*hypovolemia*" denoted the period of time during which the patient suffered blood loss through her lumbar artery. This segment began when the patient first entered intensive care and ended when she was removed from the ICU to have the bleed embolized. Upon return to the ICU, the patient entered the "*intermediate recovery*" stage, where she began recovering with aid from drugs and fluid resuscitation. Once all the fluids and drugs were weaned, the patient entered the "*steady state*", where she stabilized on her own. These states are labeled in Figure 3.3.

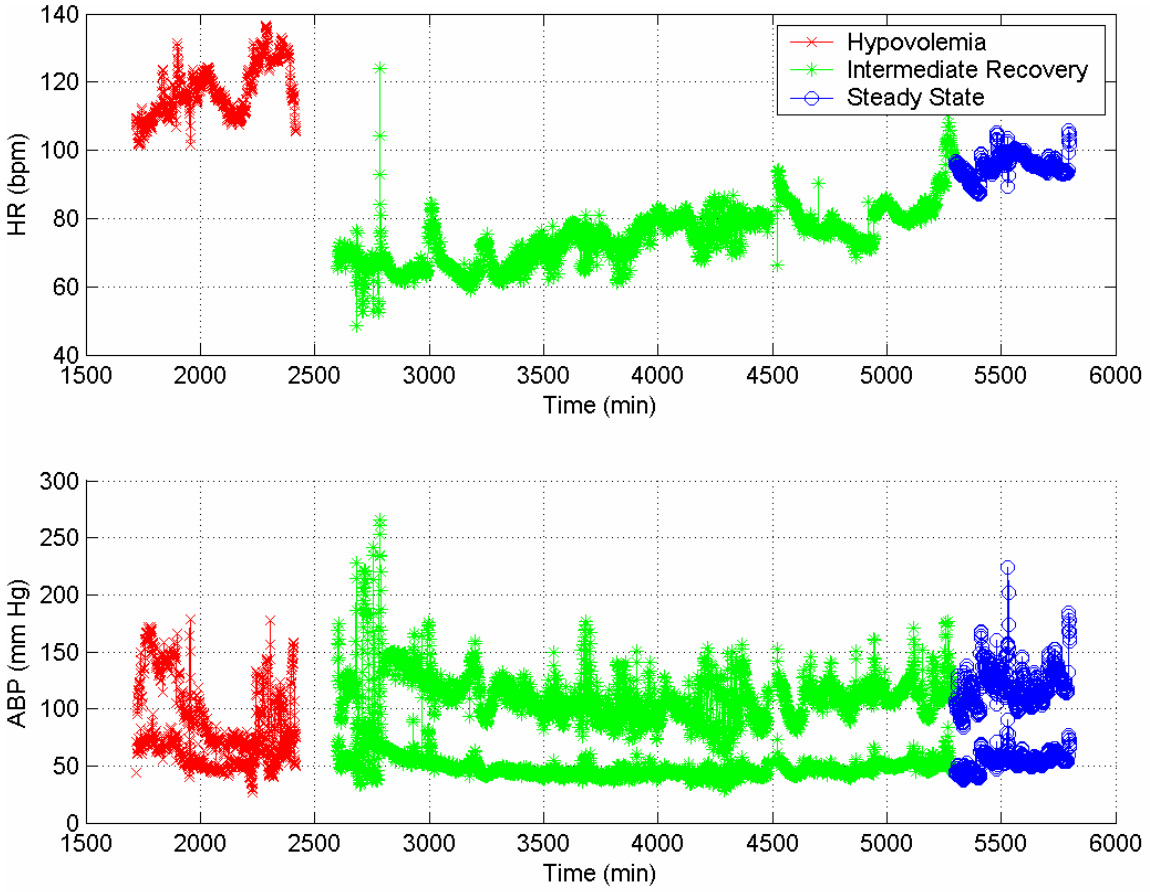
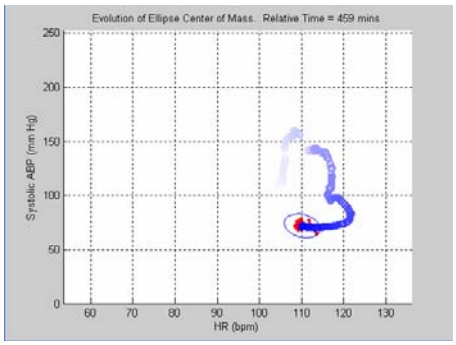


Figure 3.3: Trend data segmented into hypovolemia, intermediate recovery, and steady states.

Having segmented the data, we then plotted it in phase space. Figure 3.4 shows four combinations of the variables we plotted.

Some interesting observations can be made about the data in Figure 3.4. First, note how we have taken the 3 days of data from Figure 3.3 and condensed it all into a single plot. This illustrates the power of phase space plotting to concisely represent a set of data.

Looking at Figures 3.4.1 – 3.4.3, the different states are almost separable. In other words, if one were to draw a few lines, as shown in Figure 3.4, one could do a good job of separating the states. This suggests that states can be



separated by thresholding based on a linear combination of heart rate and blood pressure.

Furthermore, the variance of the data during the steady state is considerably less than during the hypovolemia and intermediate recovery states. This observation is comforting, since at this resolution (1 sample/min) it implies that the body is settling down to a more stable and better regulated state.

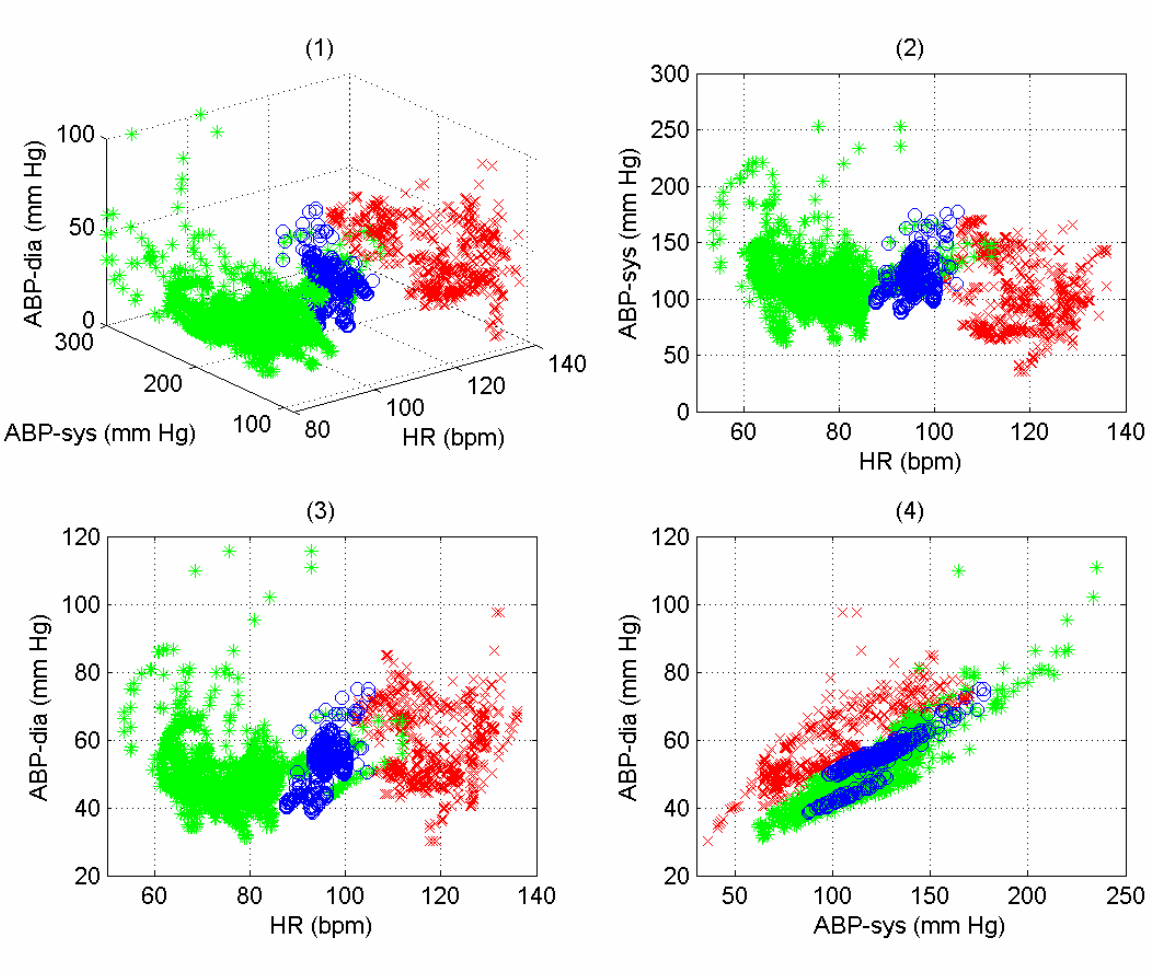


Figure 3.4: Four views of patient data in phase space. Red crosses denote the "*hypovolemia*" state, green asterisks denote the "*intermediate recovery*" state, and blue circles denote the "*steady state*".

Finally, observing Figure 3.4.4 (enlarged in Figure 3.6), we notice another nice characteristic of phase space plots. For signals that are inherently correlated, such as systolic and diastolic arterial pressures, it is possible to visually gauge the degree of correlation between two signals by observing how well a line approximates the

distribution of the data. As seen in Figure 3.6, the correlation between signals seems to improve as the patient approaches steady state.

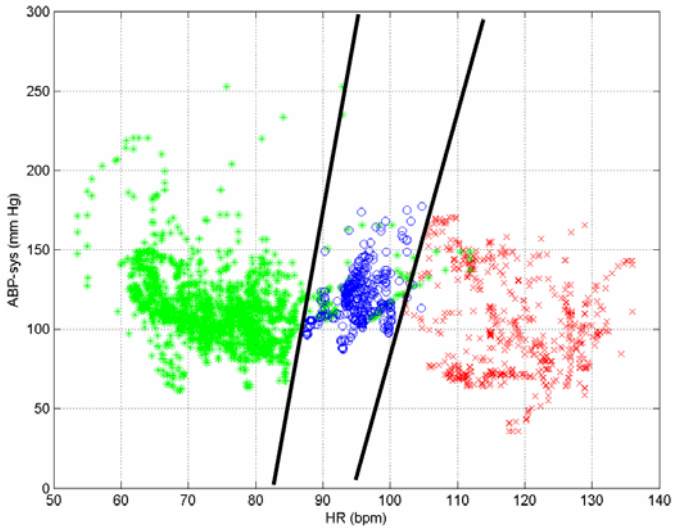


Figure 3.5: A good linear separation of states using HR and systolic ABP as the axes of the phase space plot.

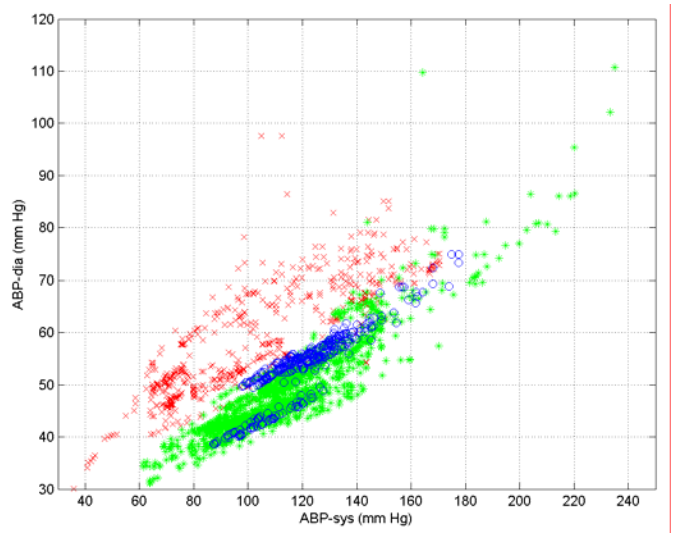
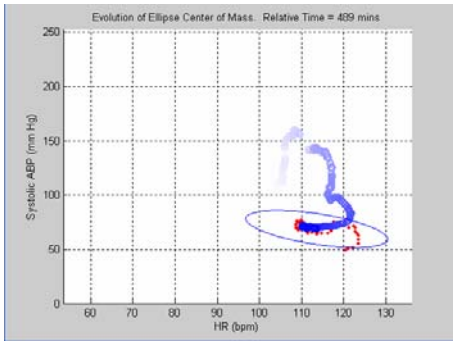


Figure 3.6: Phase space plot of the systolic and diastolic blood pressures showing how the degree of correlation between the signals increases as the patient moves from the hypovolemic state to the steady state. Red crosses denote the hypovolemic state, green asterisks denote the intermediate recovery state, and blue circles denote the steady state.



Based on these observations, we see that there is some added value to using phase space plots to visualize physiological data. In addition to providing a means to concisely summarize a history of physiological measurements, phase space plotting allows us to observe the trajectories of physiological variables, visually assess correlations between physiological signals, and easily inspect the level of variation in the data.

3.4.2 Conclusions

In this section we made use of phase space plotting to efficiently condense the information of multiple variables into a single plot. In addition to allowing us to make quick assessments about the evolution of the patient state, phase space plots revealed the possibility of detecting significant changes in patient health by observing an adequate set of variables.

Because only one patient has been studied in detail, it is hard to form any concrete conclusions. For the particular patient examined in this chapter, the heart rate and blood pressure were a good choice of variables to formulate the phase space. It is not quite clear if for different patients with different illnesses these variables would be the best ones to observe. In the future, more patients will be looked at to verify these conclusions.

Assuming that a proper set of variables can be chosen to distinguish between patient states, it might be possible to detect transitions between states automatically. Furthermore, by introducing physiology into the picture in the form of favorable/unfavorable directions in phase space (shown for the case of HR and systolic ABP shown in Figure 3.7), it may be possible to develop alarms that perform better than the current threshold-based alarms. In the following chapters we begin to explore some of these possibilities.

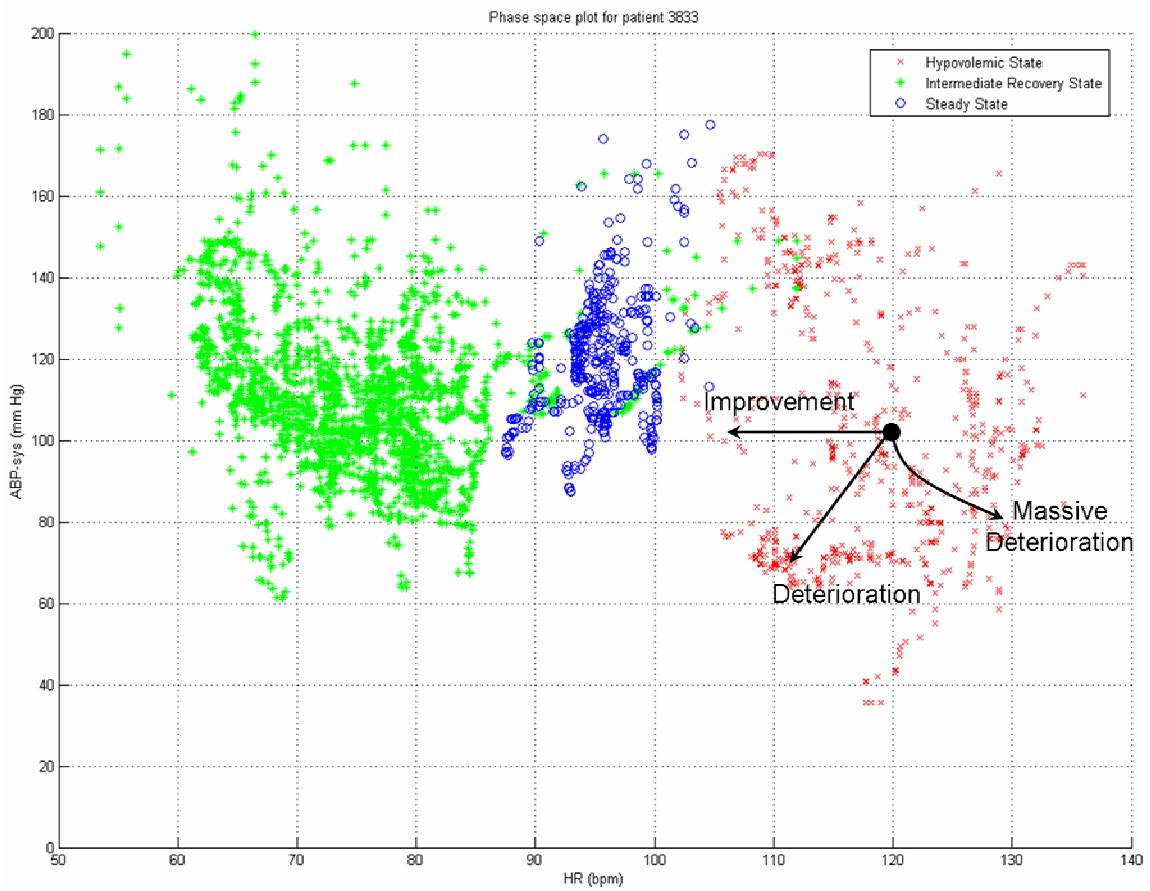
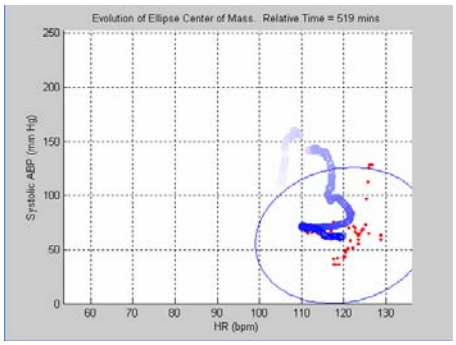


Figure 3.7: Physiologically favorable/unfavorable directions in phase space for systolic ABP and HR measurements.

3.5 Variability Analysis Using Power Spectra

3.5.1 Background

The idea of using spectral analysis to examine the variability of physiological signals, particularly heart rate, has been around for some time. However, it wasn't until the early 1980s that the scientific community began to understand how to interpret the signal spectra. Akselrod et al. (1981) were the first suggest that the variability in hemodynamic parameters could be explained by dynamic responses of the cardiovascular control systems in the body. Specifically, they concluded that the



spectra of heart rate fluctuations, averaged on a beat-to-beat basis, consisted on three major peaks: A low-frequency, mid-frequency, and high-frequency peak (see Figure 3.8). The presence of the mid and high-frequency peaks was attributed to parasympathetic control, while the low-frequency peaks were modulated by both the sympathetic and parasympathetic nervous systems as well as the renin-angiotensin system.[12]

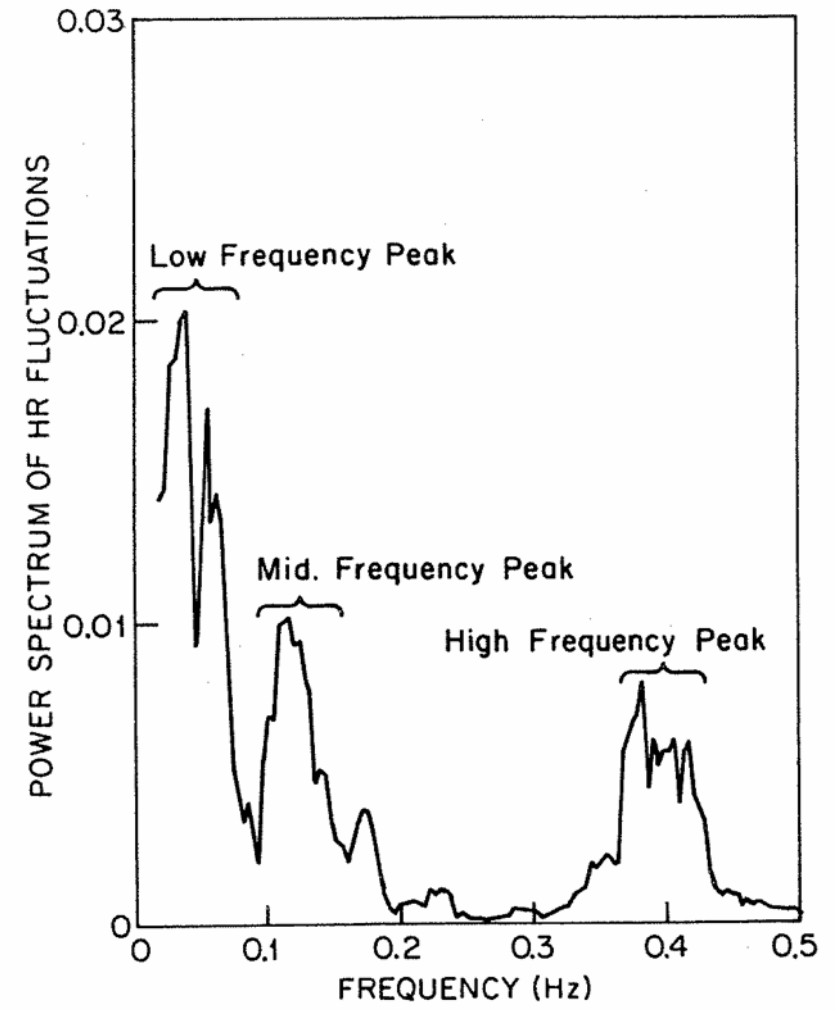


Figure 3.8: Structure of the HR power spectrum from a canine [Akselrod et al., 1981].

Since its pioneering, spectral analysis has been studied extensively and has produced many interesting findings ([Pomeranz et al., 1985], [Winchell & Hoyt, 1996], [Goldstein et al., 1998]). Much work has gone into establishing standards for heart-rate variability analysis in order for the scientific community to collaborate [Malik et al.,

[12] Details about the sympathetic, parasympathetic, and renin-angiotensin systems can be found in Fauci et al. (1998).

1996] and work has also been done to begin to understand the variability in other signals such as the ABP [Shi et al., 2003]. To this day, this area continues to attract substantial research interest.

3.5.2 Processing

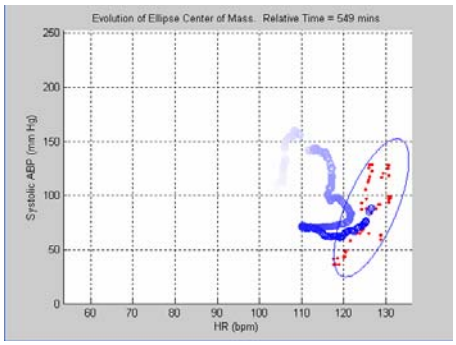
Our purpose for looking into spectral analysis of physiological signals was to explore the variability of our data. In particular, we focused on analysis of beat-to-beat averaged heart rate and arterial blood pressure data sampled at 2Hz.

The procedure for analyzing the data was as follows. We began by taking a portion of beat-averaged heart rate data from the patient described in Section 3.3. Since we were mainly interested in the higher frequency fluctuation of the signals, the low-frequency trends were removed by subtracting a best-fit linear approximation to the data over consecutive 30-second windows (see Figure 3.9). Next, the data was visually inspected to separate it into segments with different levels of variability. The power spectrum for each of these segments was computed using Welch's method and the results were plotted (see Figure 3.10).[13] These steps were repeated for the beat-averaged ABP data (see Figures 3.11 and 3.12).

3.5.3 Analysis and Conclusions

Examining the resulting spectra for the heart rate we see that with the exception of segment 1 – containing a substantial amount of high frequency noise – all the spectra have large amount of their frequency content concentrated around 1 or 2 frequencies. Furthermore, the spectral peaks are all at similar frequencies (roughly .25 Hz). This peak corresponds to the respiration rate of the patient, which based on the spectrum is approximately 15 breaths/min.

Finally, comparing the heart rate spectra with the mean blood pressure spectra of the same



[13] In computing the spectra via Welch's method we divided data into 8 segments, with a segment overlap of 50 percent. The underlying FFT contained 1024 points. Each data segment was approximately 15 minutes in length and the data sampling rate was 2Hz.

segments, we see that the two are very similar. This suggests that autonomic functions that affect variability in heart rate are also often reflected in the mean ABP. It may be interesting to investigate under what conditions, if any, these spectra will differ from each other.

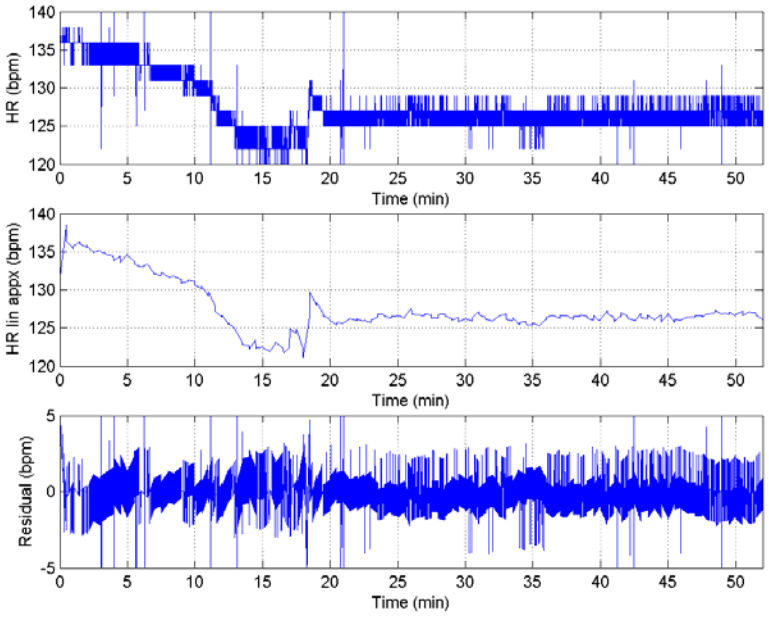


Figure 3.9: A linear approximation to the heart rate (middle plot) was subtracted from the original signal (top plot) to remove low frequency trends from the data.

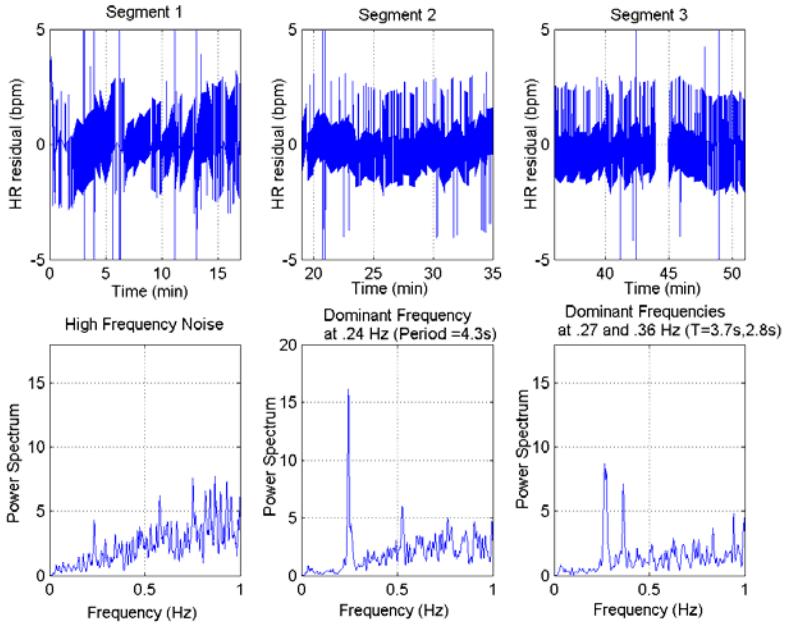


Figure 3.10: Spectral analysis of three segments of the heart rate signal. Each segment is approximately 15 minutes in length.

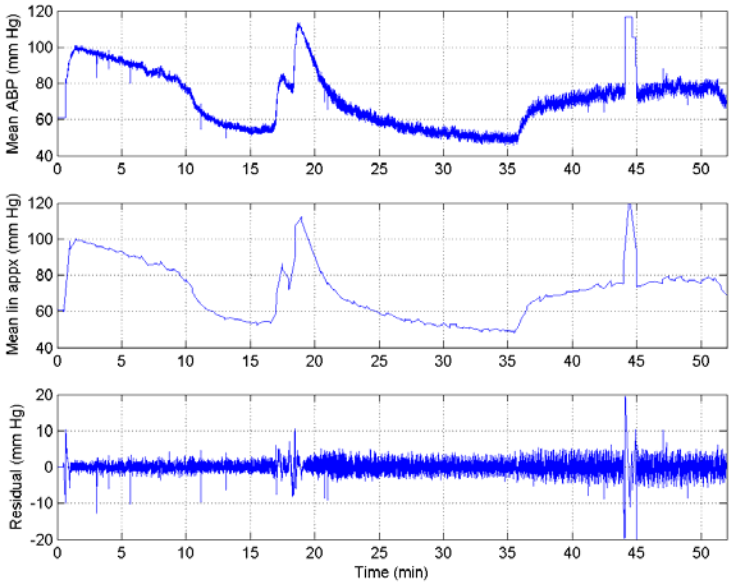


Figure 3.11: A linear approximation to the mean ABP (middle plot) was subtracted from the original signal (top plot) to remove low frequency trends from the data.

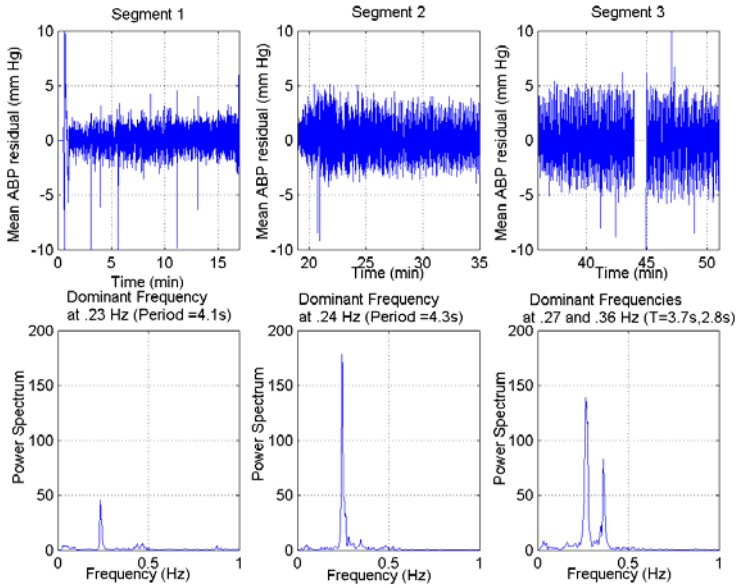
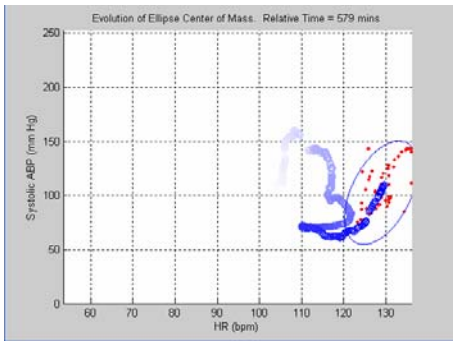


Figure 3.12: Spectral analysis of three segments of the mean ABP signal. Each segment is approximately 15 minutes in length.



The results of this study confirm that the variability in our data is consistent with that found

in heart rate variability studies by Pomeranz et al. (1985) and that our procedure for analyzing power spectra works properly. In the future we hope to use these methods to view the variability of the heart rate and ABP spectra as a function of time to see how the results correlate with the physician's interpretations of what is going on with the patient.

3.6 Conclusion

In this chapter we presented two methods to visualize and analyze physiological signals taken from a real patient. We introduced the power of phase space analysis to condense multi-parameter data into a single display and used spectral analysis to interpret variability of the physiological data. In the next chapter we build on some of the observations made using phase space analysis to expand on our set of visualization and analysis techniques.

Chapter 4

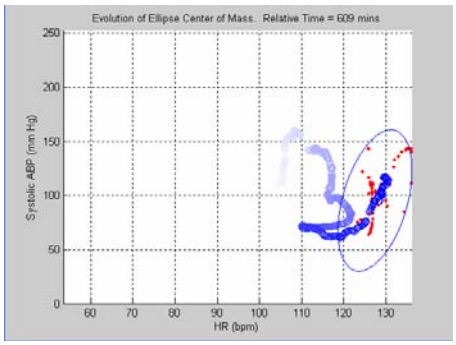
Data Clustering Using Singular Value Decomposition

4.1 Introduction

In the previous chapter, we showed that by observing an appropriate set of cardiovascular variables in phase space, it was possible to differentiate between significant patient states (summarized in Figure 4.1). Motivated by this observation, we took on the problem of characterizing a state in phase space, with the ultimate goal of developing an analytical method for tracking the evolution of patient state.

In this chapter, we propose a method for capturing the information of a particular state by geometrically clustering the data in phase space. The method is based on the singular value decomposition (SVD). We first introduce the basics of the SVD, followed by a detailed description of the application of the technique to clustering. Finally, we discuss limitations of the clustering method along with possible solutions to these limitations.

4.2 Defining a State



To begin tackling the problem of state characterization, we first need to develop the notion of what a state is. Unlike the definition of state often used in dynamic systems theory, we define a

state as a set of points residing within some region of an N-dimensional space, where N corresponds to the number of variables being observed. As long as the trajectory of points remains close to the designated region in phase space, the state is said to remain stable. If at some point in time the trajectory of points begins to drift outside this region, the state is said to have changed. Exactly how much deviation from the region is necessary for the state to change will be discussed in more detail later.

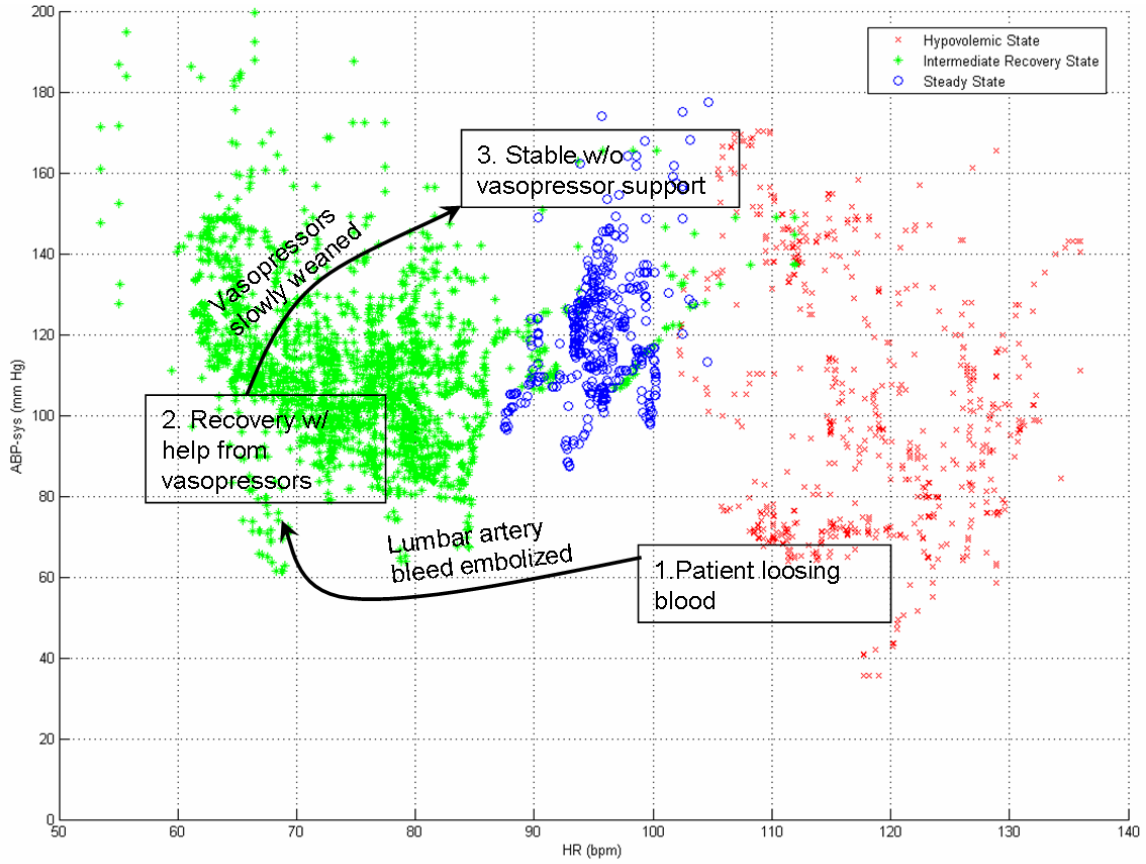


Figure 4.1: Summary of patient case study from Chapter 3. Notice how the hypovolemic, intermediate recovery, and steady states are well separated in phase space formed by heart rate and systolic ABP.

For the moment, the definition outlined above, depicted graphically in Figure 4.2 [from Castiglioni, 2004] is sufficient.

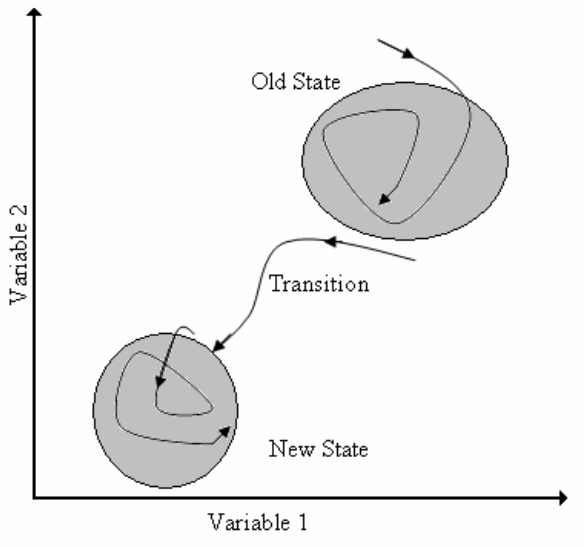
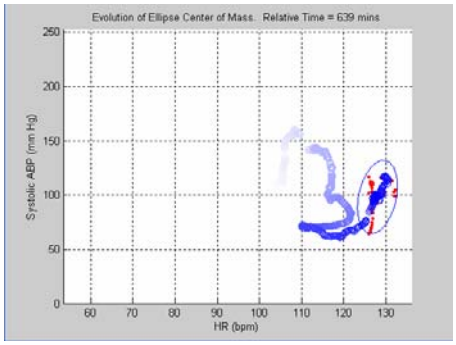


Figure 4.2: State representation in phase space for the case of two-dimensional data [Castiglioni, 2004].

Assuming we can determine a set of points that give a good representation of the range of values within a state, one way to approximate the region covered by a particular state is to geometrically cluster the data. Restricting the problem to the case of two-dimensional data for the time being, the cluster pattern could be as simple as a tightly fitting box or ellipse, as shown in Figure 4.3.[14]

Although either geometry could be used as the basis for clustering, we adopt the ellipse in our work because it ties in well with the geometric interpretations of the singular value decomposition. Because the SVD comprises the underlying groundwork for our clustering, we will review the basics of this technique before describing its application. A more thorough description of the SVD can be found in numerous texts, including those by Strang (1998), Trefethen and Bau (1997), and Golub and Van Loan (1996).



[14] This clustering can be generalized to N-dimensional data by replacing the 2-D geometrical cluster by one with N dimensions. However, beyond 3 dimensions, it is difficult to visualize the clustering.

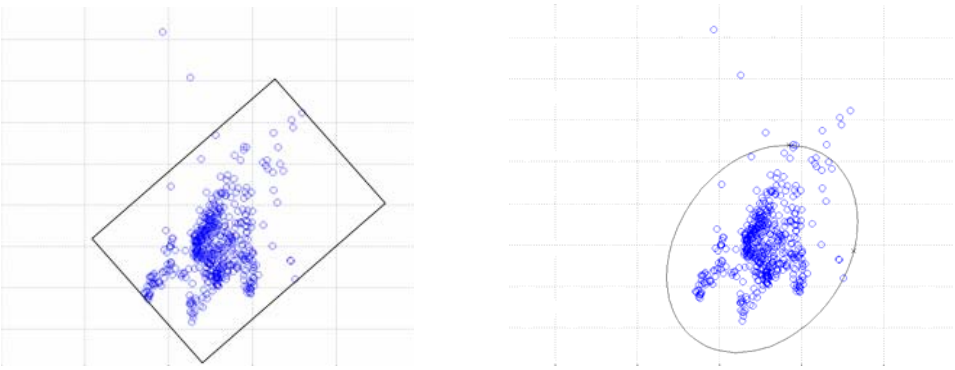


Figure 4.3: Two examples of how to geometrically cluster a set of data.

4.3 Singular Value Decomposition Basics

The singular value decomposition is one of the most powerful tools in linear algebra. It allows us to diagonalize any matrix, rectangular or square, by choosing an appropriate set of orthonormal bases. For real matrices, the factorization is typically written in the form[15]:

$$A = U\Sigma V^T \tag{4.1}$$

where U, Σ, and V are matrices whose properties are listed in the table below:

Table 4.1: SVD Matrix Properties

Matrix Name	*Dimensions*	*Special properties*
A	M-by-N	Real matrix
U	M-by-M	Orthogonal matrix ($U^T U = I$)
V	N-by-N	Orthogonal matrix ($V^T V = I$)
Σ	M-by-N	Diagonal matrix $\left\{\begin{matrix}\Sigma_{i,j} = 0; i \neq j \\ \Sigma_{i,j} = \sigma_i; i = j\end{matrix}\right\}$ Diagonal entries sorted by size (i.e. $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(M,N)} \geq 0$)

[15] In the case of complex matrices, the matrix properties are slightly different. For more information about SVD, see [Strang].

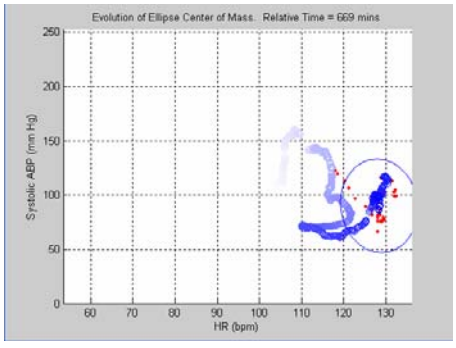
If the rank of the *A* matrix is r (i.e. rank (A) = r), the decomposition can be further reduced to the form:

$$A = U_* \Sigma_* V_*^T \tag{4.2}$$

where U_* and V_* are the first r columns of U and V, and Σ_* is an r x r diagonal matrix with values σ_1 to σ_r along its diagonal[16]. By design, the columns of the U_* and V_* matrices span the column and row spaces of *A*, respectively. Furthermore, we can show that the columns of U_* give the directions of maximum variation for the vectors formed by the columns of *A* and that Σ gives the standard deviations in the directions of U.[17] It is these two properties of the SVD that we will exploit in our data clustering.

Finally, to tie the idea of the ellipse with the SVD, we go to the geometric interpretation of the decomposition. In the context of linear transformations, the *A* matrix is a linear operator that maps a vector in M-dimensional space (or input space) to an N-dimensional space (or output space). Re-examining Equation 4.1, the singular value decomposition simply breaks down this transformation into simpler transformations, namely rotations (performed by the U and V matrices) and a scaling (performed by the Σ matrix). The geometric interpretation of these simpler transformations is illustrated for the 2D case in Figure 4.4 [from Muller, Magaia, and Herbst, 2004].

As shown in the figure, the image of a 2-dimensional sphere centered about the origin under any non-singular 2-by-2 matrix is an 2-dimensional ellipse. The sphere is first rotated such that the vectors defined by the columns of the V^T matrix ($\mathbf{v}_1$ and $\mathbf{v}_2$) line up with the coordinate axes. It is then scaled by the matrix of singular values Σ, which corresponds to scaling along the directions of $\mathbf{v}_1$ and $\mathbf{v}_2$ by σ_1 and σ_2, respectively. Finally, the resulting ellipse is rotated once more, such that the coordinate axes line up with the columns of the U matrix.



[16] Equation 4.2 is sometimes referred to as the thin SVD [Golub & Van Loan, 1996].

[17] Both of these statements are verified in Appendix A.

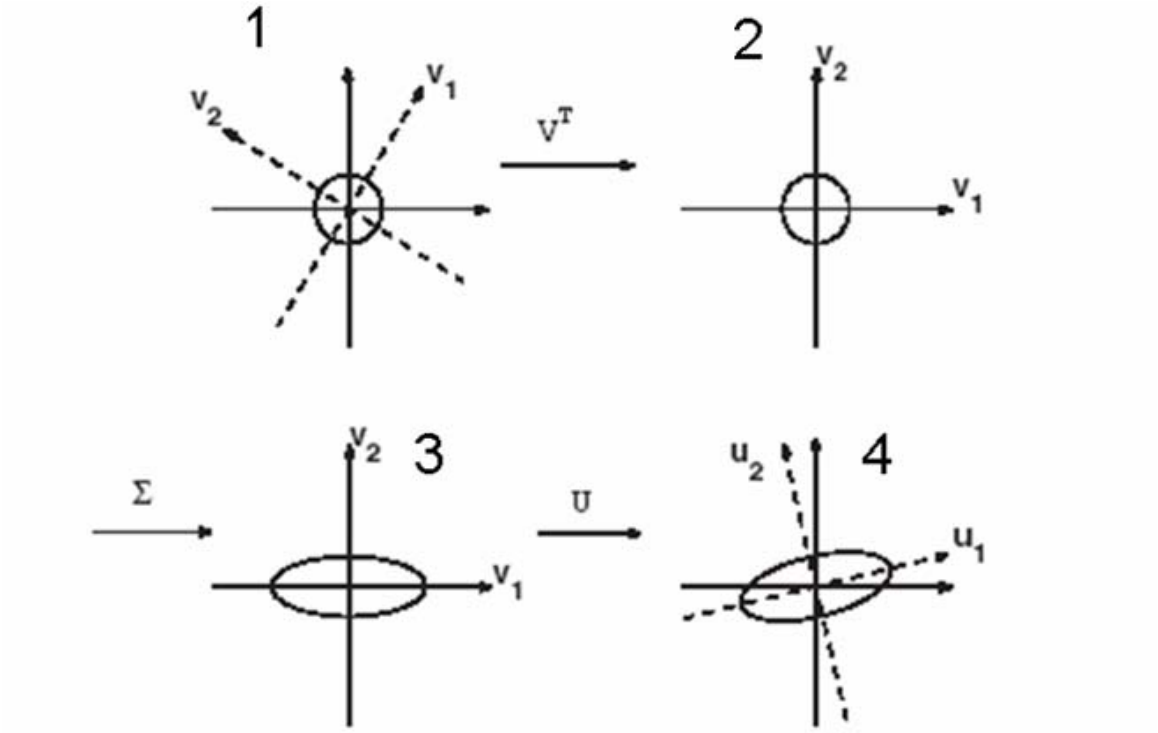


Figure 4.4: The geometrical interpretation of the SVD. The circle under matrix multiplication becomes an ellipse [image from Muller, Magaia, and Herbst, 2004].

A consequence of the geometrical interpretation is that the resulting ellipse characterizes the matrix *A*. The ellipse is aligned along the direction of $\mathbf{u}_1$ and $\mathbf{u}_2$ – which span the column space of *A* – and has principal semi axes of length σ_1 and σ_2, describing the magnitude of the spread of A in the directions of the $\mathbf{u}_i$. Note that the principal axis corresponds to the largest singular value σ_1 and that the semi-minor axis of the ellipse corresponds to the second-largest singular value. If this picture were generalized to M dimensions, the m^{th} largest singular value would describe the length of m^{th} principal axis of a hyper-ellipse.

Having established the basic ideas of the SVD along with its geometrical link to the ellipse, we are now ready to develop the clustering method.

4.4 Application of SVD to Data Clustering

Say we have a set N elements of 2-dimensional data $\mathbf{X}$ whose mean, defined as $\overline{\mathbf{X}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$, is the zero vector.[18] Each dimension of the data could, for example, correspond to a cardiovascular variable.

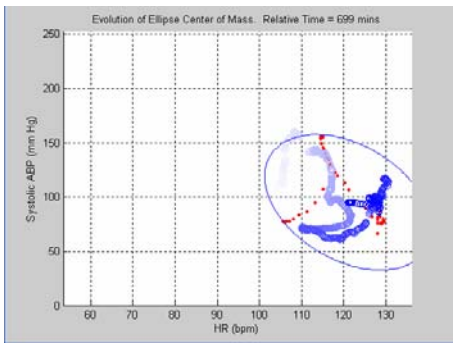
If we define a matrix *A* such that the columns of *A* are the set of points $\mathbf{x}_i$, then computing the SVD gives us a decomposition that looks like

$$\left[\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_N\right] = U\Sigma V^T = \left[\mathbf{u}_1\ \mathbf{u}_2\right]\begin{bmatrix}\sigma_1 & 0 & 0 & \ldots & 0\\ 0 & \sigma_2 & 0 & \ldots & 0\end{bmatrix}\begin{bmatrix}\mathbf{v}_1^T\\ \mathbf{v}_2^T\\ \ldots\\ \ldots\\ \mathbf{v}_n^T\end{bmatrix} = \left[\sigma_1\mathbf{u}_1\ \sigma_2\mathbf{u}_2\ 0\ldots 0\right]\begin{bmatrix}\mathbf{v}_1^T\\ \mathbf{v}_2^T\\ \ldots\\ \ldots\\ \mathbf{v}_n^T\end{bmatrix} \qquad (4.3)$$

where the U, Σ, V have dimensions 2-by-2, 2-by-N, and N-by-N, respectively.

The rightmost equality of Equation 4.3 tells us that $\sigma_1\mathbf{u}_1$and $\sigma_2\mathbf{u}_2$ span the space occupied by the $\mathbf{x}_i$; that is, any of the $\mathbf{x}_i$ can be recovered by the linear combination of the weighted $\sigma_1\mathbf{u}_1$and $\sigma_2\mathbf{u}_2$ vectors. For example, to recover $\mathbf{x}_1$ we simply scale $\sigma_1\mathbf{u}_1$ by $\mathbf{v}_1^T$, $\sigma_2\mathbf{u}_2$ by $\mathbf{v}_2^T$ and then add the two vectors.

Using the fact that the vectors $\sigma_1\mathbf{u}_1$ and $\sigma_2\mathbf{u}_2$ are the directions of maximum variation for the data $\mathbf{x}_i$, we know these vectors capture the directions that best approximate the distribution of points $\mathbf{x}_i$ in the mean-squared sense. By using the $\sigma_i\mathbf{u}_i$ as the principal semi-axes of an ellipse and then scaling the axes appropriately, we can generate an ellipse that fits tightly around the $\mathbf{x}_i$. Figure 4.5 graphically summarizes these steps.



[18] If $\overline{\mathbf{X}}$ does not have zero mean, we can define a new set of vectors $\hat{\mathbf{x}}_i = \mathbf{x}_i - \overline{\mathbf{X}}$, which does have zero mean.

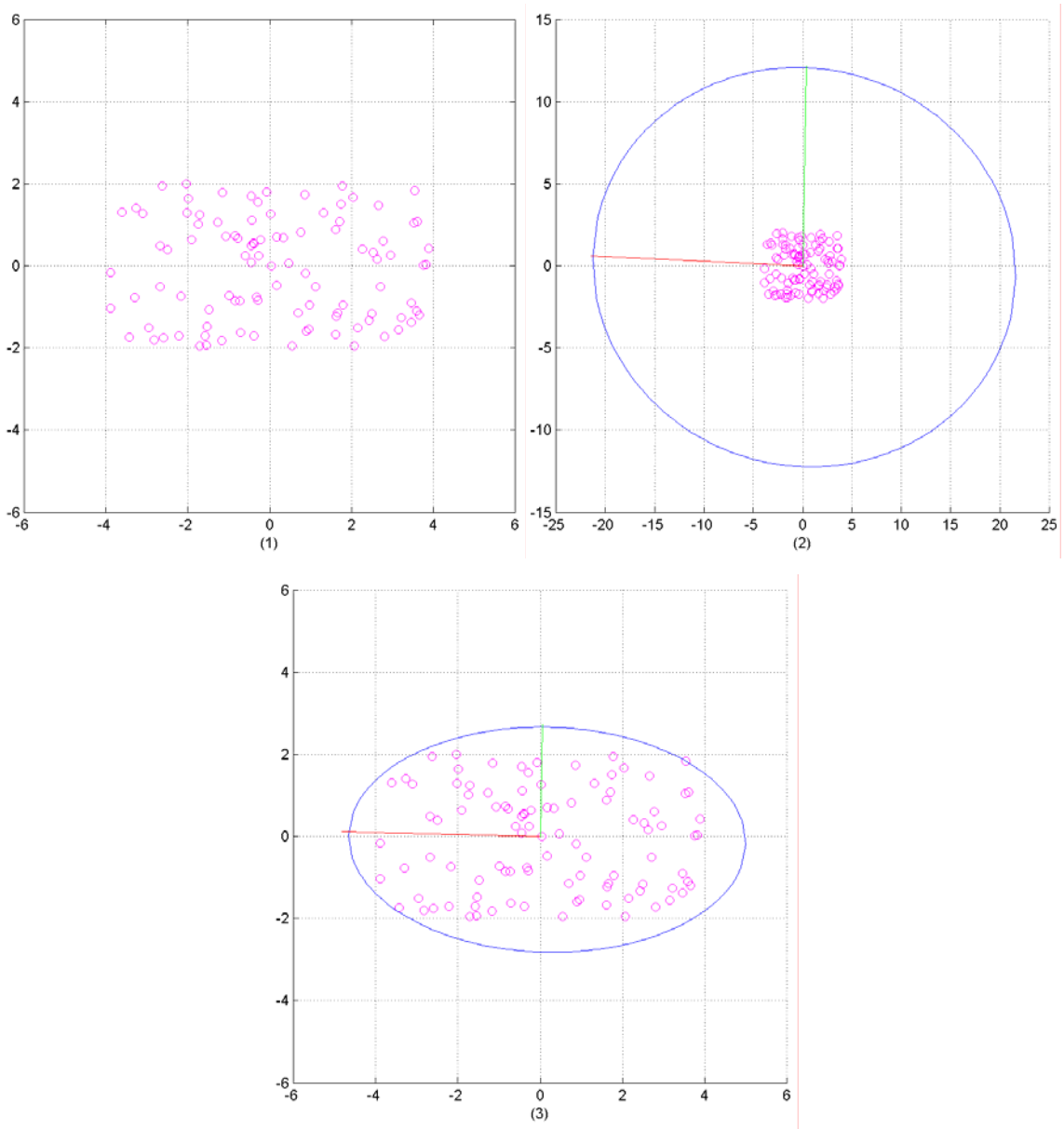


Figure 4.5: Graphical summary of SVD clustering. (4.5.1) We begin with a set of points $\mathbf{x}_i$. (4.5.2) We then calculate the SVD and use the $\sigma_i\mathbf{u}_i$ vectors as the principal axes of our ellipse. (4.5.3) Finally, the ellipse is rescaled for a tighter fit.

The final step of scaling the ellipse by a constant is necessary because the $\sigma_i\mathbf{u}_i$ increases with the number of points being clustered. This effect is shown in Figure 4.6 and has to do with the projection from an N-dimensional space onto an M-dimensional space, where N>M. Fortunately, because the SVD gives us the correct proportions for the deviations, we only need to find one scale factor by which to rescale all the axes.

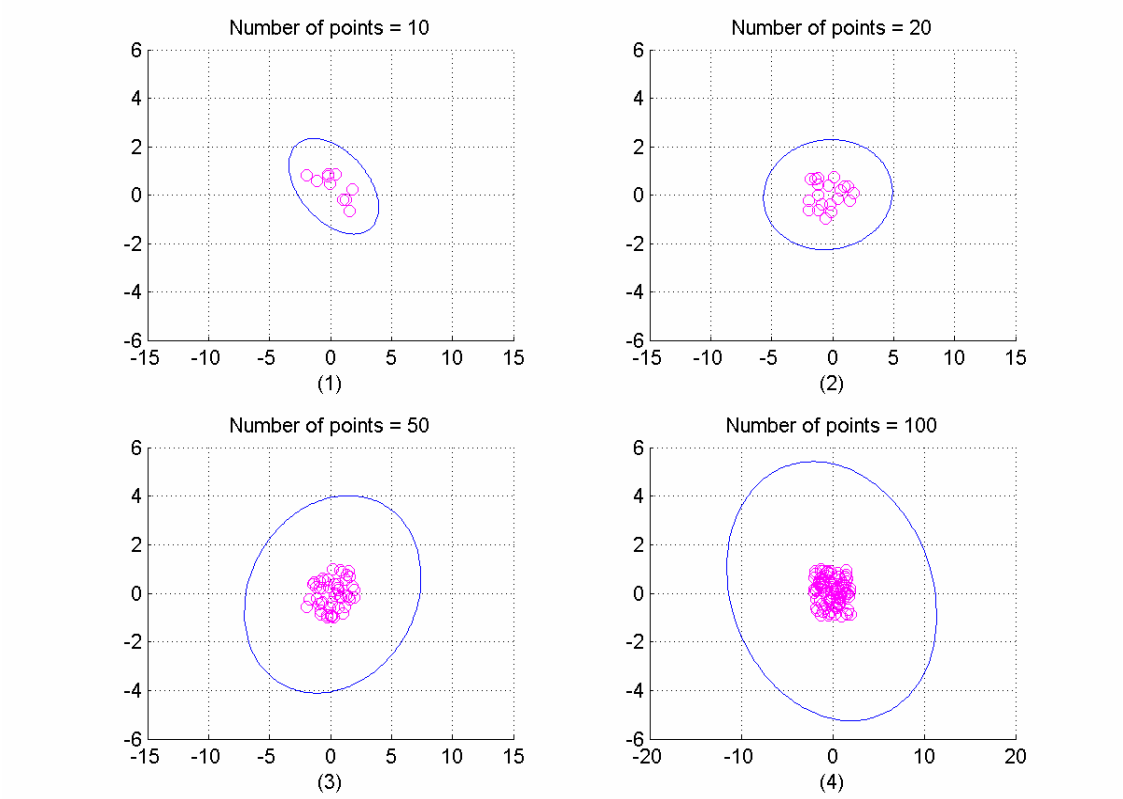
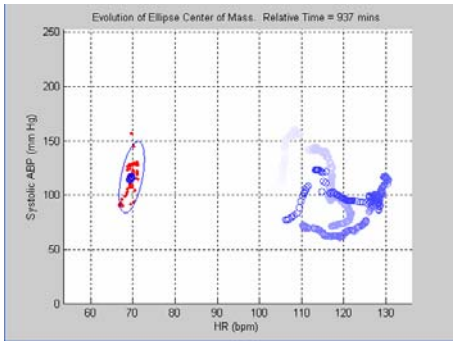


Figure 4.6: Effect of number of points on the size of the clustering ellipse. Points were generated using a 2D uniform distribution with the x axis varying between –2 and 2 and the y axis varying between –1 and 1.

To see how the problem can be corrected, we go back to Equation 4.3, rewritten and expanded below:

$$\begin{bmatrix}\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_N\end{bmatrix} = \begin{bmatrix}\sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & 0 & \dots & 0\end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & v_{13} & \dots & v_{1N} \\ v_{21} & v_{22} & v_{23} & \dots & v_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ v_{N1} & v_{N2} & v_{N3} & \dots & v_{NN} \end{bmatrix} \tag{4.4}$$

Looking at the right side of the equality, we see that only the v_{1i}'s and v_{2i}'s contribute to the $\mathbf{x}_i$'s. To capture all the points, we need rescale the principal directions of the ellipse such all the points lie within the 2-norm of the $\sigma_i\mathbf{u}_i$ vectors. The newly



scaled vectors $\sigma_i \tilde{\mathbf{u}}_i$ are defined by the following equations:

$$\sigma_i \tilde{\mathbf{u}}_i = (\sigma_i \mathbf{u}_i) * (K) \tag{4.5}$$

$$K = \sqrt{\max_i \sum_{k=1}^{2} (v_{k,i})^2} \tag{4.6}$$

4.5 Summary of SVD Clustering Algorithm

With all the major steps defined in the previous section, we now summarize the clustering algorithm:

1. Subtract center of mass from data and store this value.
2. Compute SVD and store U, Σ, and V matrices.
3. Rescale $\sigma_i \mathbf{u}_i$ vectors by factor defined in Equations (4.5) and (4.6)
4. Generate clustering ellipse using $\sigma_i \mathbf{u}_i$ vectors as principal semi-axes
5. Shift ellipse by center of mass computed in step 1.

Although most of the work presented to this point was for the two dimensional case, the algorithm can be easily extended to higher dimensional data.

The algorithm was implemented using MATLAB. The commented code can be found in Appendix B.

4.6 Limitations of SVD Clustering

In evaluating the SVD clustering method we observed several limitations. These are outlined in the sections below.

4.6.1 Sensitivity to Outliers

Because the clustering technique calculates an ellipse that captures all the points it is given, the method is very sensitive to outliers. As shown in Figure 4.7, even a single outlier will expand the ellipse significantly.

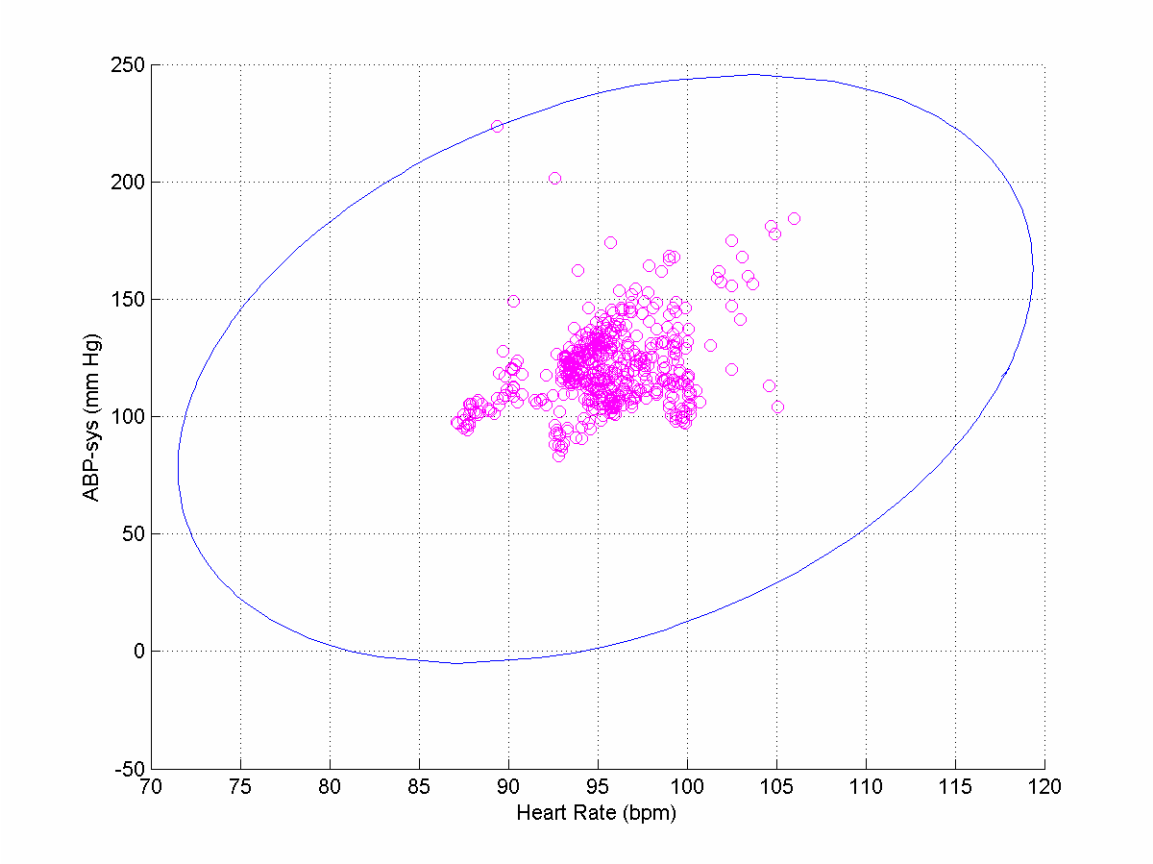
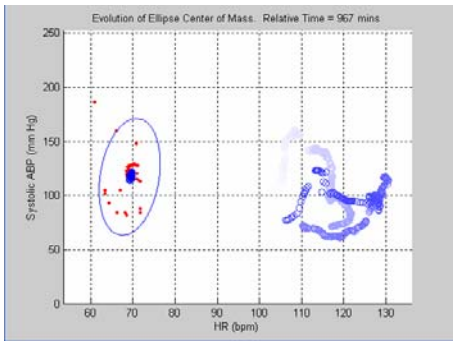


Figure 4.7: The effect of outliers on the clustering ellipse.

One way to address this problem is to alleviate the constraint that requires all points to be clustered by the ellipse. We incorporated this solution into our algorithm by adding an input parameter to specify the minimum percentage of points that must lie within the ellipse. This value was set to 95%.

Although this feature helps eliminate some of the sensitivity of the clustering technique to outliers (see Figure 4.8), it is not a perfect solution. The best way to address this issue is to preprocess the data prior to using the clustering algorithm to remove as many outliers as possible.



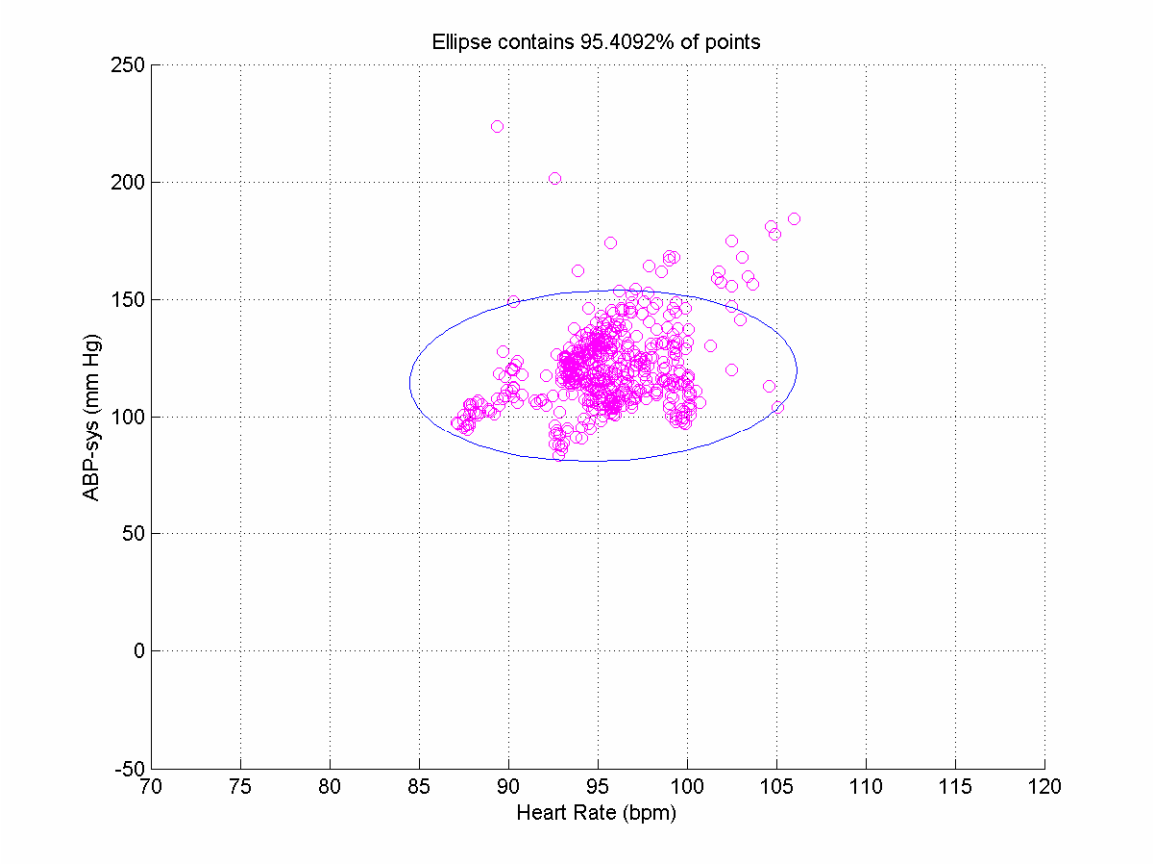


Figure 4.8: Reduced sensitivity to outliers by requiring only 95% of points to lie inside the ellipse.

4.6.2 Unevenly Distributed Data

Another drawback of this method is that it has difficulties when the data is not uniformly distributed in phase space. In this case, the ellipse is not centered on the data, as shown in Figure 4.9. This problem stems from the fact that we are subtracting the center of mass from the data. A more desirable fit for the data, yielding a tighter fit around the data, is shown in Figure 4.9 is shown as a dashed ellipse.

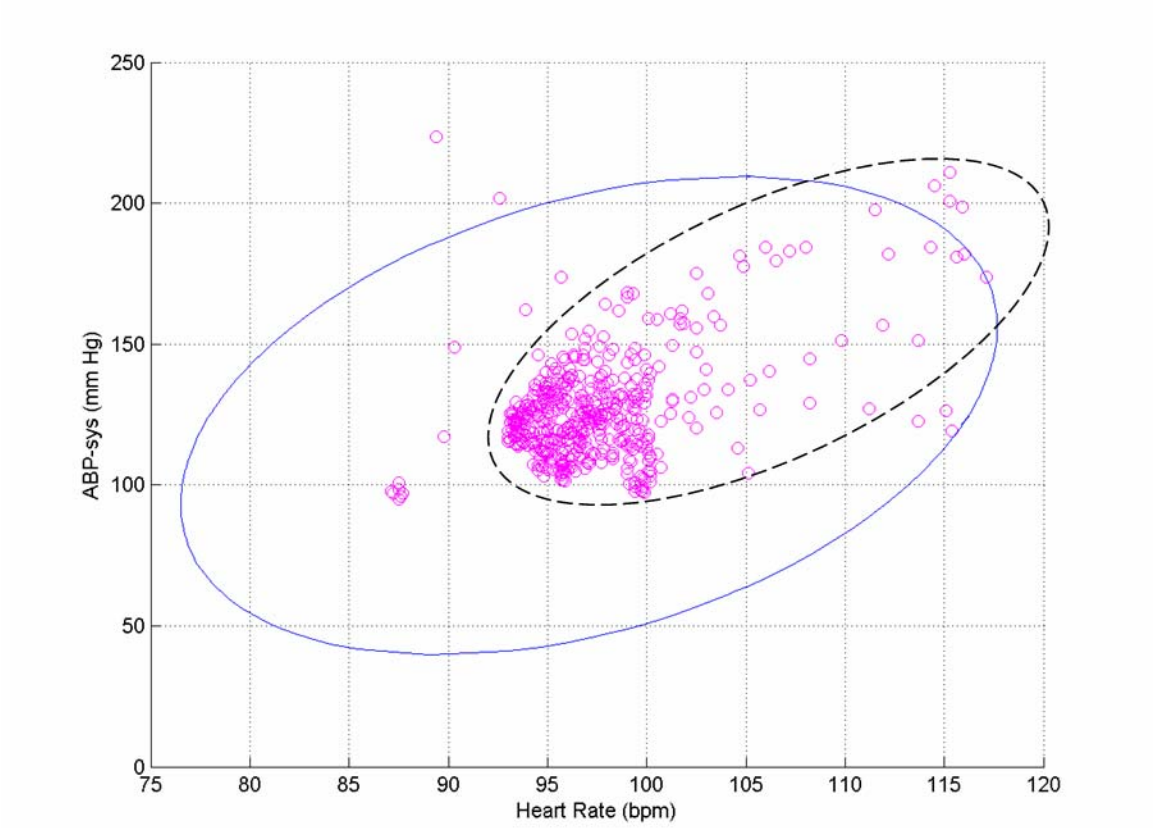
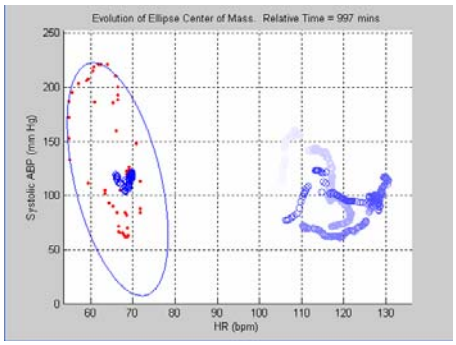


Figure 4.9: Effect of unevenly distributed data on the clustering ellipse. The solid ellipse is the one determined by the SVD clustering method. The dotted ellipse denotes a more desirable fit for this data.

4.6.3 Little Variation in One or More Variables

Although much less common than the previous two issues, the clustering method also has problems when one or more variables are nearly constant. In this case, a proper ellipse cannot be formed. Figure 4.10 shows an example of this scenario.



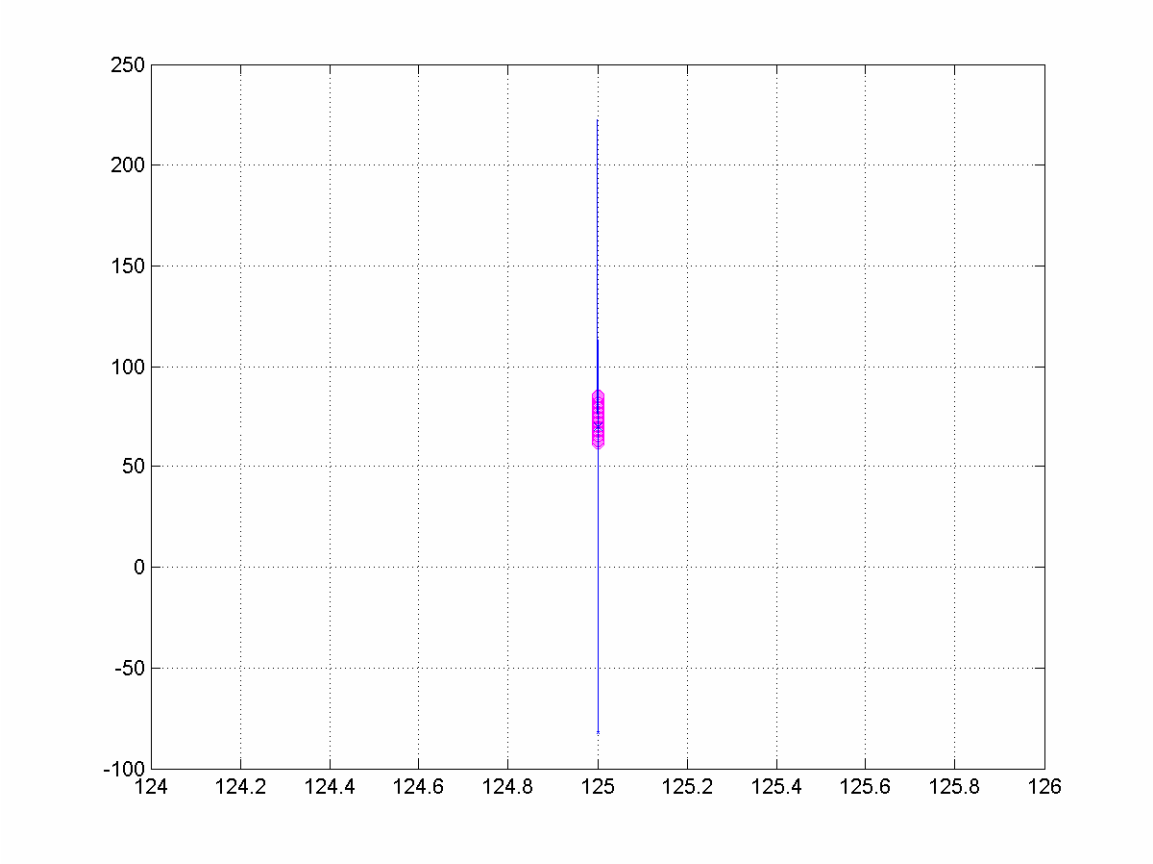


Figure 4.10: Effect of one variable being constant on clustering ellipse.

4.6.4 Clustering Data in Higher Dimensions

One final issue that arises with the SVD clustering is evident when we attempt to cluster 3-dimensional data. As an example, we take a set of samples from a Gaussian distribution described by the mean and covariance matrices:

$$\mu = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}; \mathrm{K} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

we can find a clustering ellipse that fits tightly around the data, as shown in the plots in Figure 4.11.

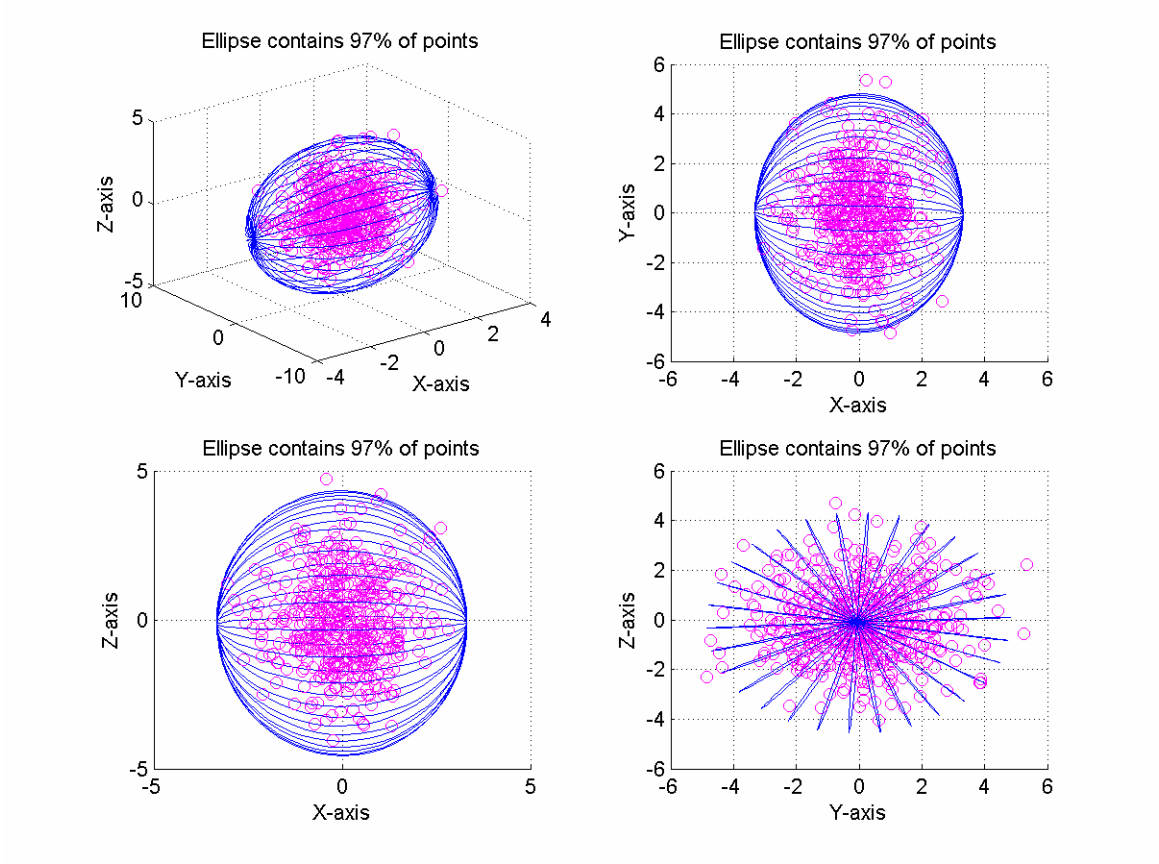
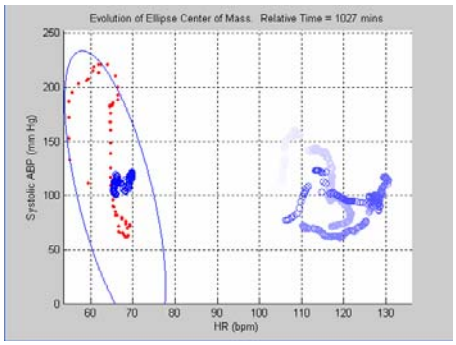


Figure 4.11: Clustering of 3D data derived from a Gaussian distribution with zero mean and covariance matrix $K=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$. The figure shows a 3-dimensional view of the data along with the 2-dimensional projections of the points onto the x-y, x-z, and y-z axes.

However, if we take a set of data, uniformly distributed from -1 to 1 in the x, y, and z direction, the ellipse does not fit as tightly anymore (that is, there is more empty space between the data points and the clustering ellipse (see Figure 4.12).



The reason for this "looser" fit is entirely due to the geometry of the data we are trying to cluster. Whereas the ellipse is well-suited for data with smooth, rounded edges in phase space, such as

a Gaussian distribution, it cannot do as good a job with data having sharp, flat edges, like the uniform distribution, below. The best job we can hope the ellipse to do is to cluster the data such that the cube of points is inscribed inside the ellipse, in which case the ellipse becomes a sphere. Because this problem arises from geometry, there is no way of improving the ability of the ellipse to fit the data. Fortunately, a uniform-like distribution is rather unlikely when we deal with real physiological signals.

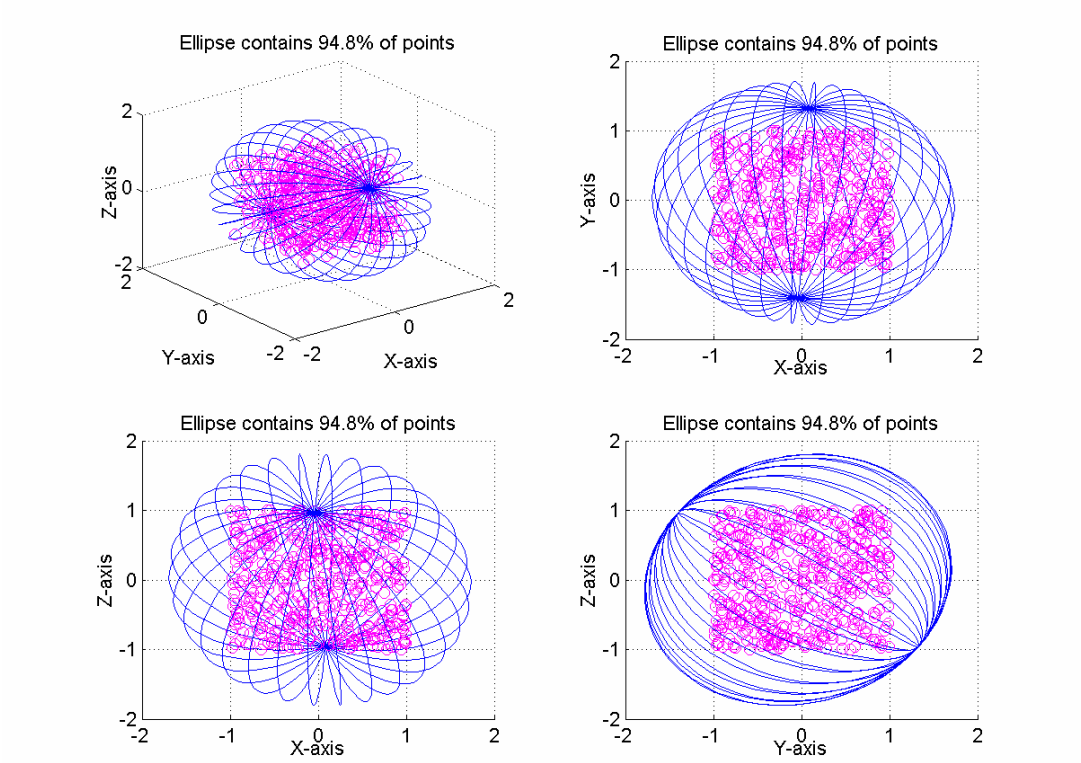


Figure 4.12: Clustering of data from a 3-dimensional uniform distribution. Notice how the ellipse fits much less tightly around the data.

4.7 Conclusion

This chapter introduced a novel method for clustering a set of data in phase space by making use of some of the properties of the singular value decomposition. In addition to explaining the structure of the algorithm, observations were made about the method's limitations and solutions were proposed to avoid these shortcomings whenever possible. Having established a basis for summarizing a set of data within phase space, it is now possible to extend the clustering algorithm.

Chapter 5

SVD Clustering Applications

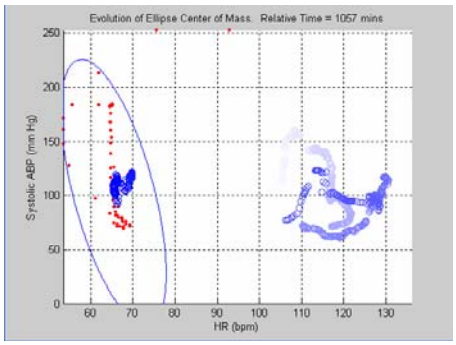
5.1 Introduction

This chapter focuses on the potential applications of SVD clustering within the ICU setting. We begin by presenting an extension to the clustering algorithm to track data in phase space. We then use the tracking algorithm to visualize trajectories of measured physiological variables as a function of time. Finally, we introduce similarity metrics based on clustering ellipse geometries to automatically detect changes in the state of a patient.

5.2 Data Tracking

5.2.1 Tracking Algorithm

In Chapter 4 we presented a method to summarize the second order statistics of a set of data by geometrically clustering it in phase space using a tight-fitting ellipse. One logical extension to this clustering technique is to track the evolution of a set of data as a function of time. This simply involves a repeated calculation of the clustering ellipse for a moving window of data. As new points are presented, the window of points discards older data and replaces it with the new samples. The result is a moving data window consisting of the last N samples for a set of variables, as shown in Figure 5.1.



To implement the tracking algorithm, the SVD clustering method was updated to compute ellipses for a moving data window. Input arguments were included to define the size of the sliding window and a plot interface was added to the algorithm to visualize the

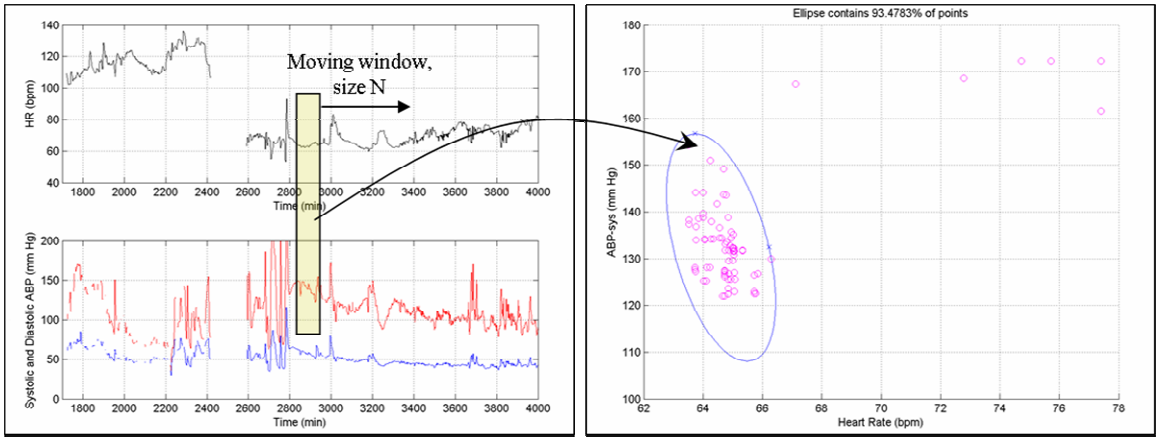


Figure 5.1: Extension of SVD clustering to tracking of physiological data.

trajectories of the ellipse as a function of the moving window. This latter feature was a good way of seeing how well the ellipse was clustering the data at each time step.

5.2.2 Performing Data Tracking on Cardiovascular Data

After completing the tracking algorithm, we decided to test it on some of the cardiovascular data from the ICU. The goals of this test were twofold. First, we wanted to observe how well the algorithm clustered the data. Second, we wished to examine how the clustering ellipse trajectories were influenced by the size of the moving window.

We initially tested the algorithm with trend data but quickly switched to higher resolution beat-to-beat averaged data, because the additional data points made the ellipse less sensitive to outliers in the data (see Section 4.6.1) and gave a richer description of the changes in the cardiovascular system.

The results of the first test were positive. In general, the ellipse did a good job of tracking the cluster of points. The only times when the ellipse did a poor job of clustering occurred when there were large changes in the data (reflected in the phase space plots as by large jumps in phase space). In these situations, the ellipse became

large and did not fit tightly around the data until the trends began to settle into a more stable trajectory.

In the second test, we found that the size of the window affected the robustness of the clustering ellipse to outliers. Setting the minimum percentage of points to be clustered at 95%[19], we observed that as the window size was increased, the sensitivity to outliers decreased. As a consequence of the increased window size, however, there was a decrease in the ability of the ellipse to respond to fast trends in the data. This result suggested that there was a tradeoff between sensitivity to outliers and the resolution at which the clustering ellipse could detect changes in the data.

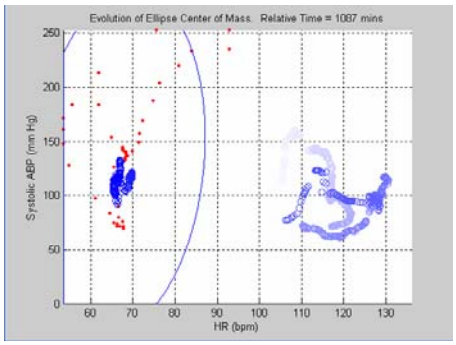
5.3 Data Tracking Applications

With the data tracking algorithm in place, we now present two of its applications. The first uses the tracking algorithm to display the trajectories of the physiological signals in phase space. The second modifies the algorithm to detect significant trends in the cardiovascular data.

5.3.1 Visualizing Trajectories of Physiological Signals

One useful application of the tracking algorithm is trajectory visualization of data in phase space. By plotting the center of mass of the clustering ellipse at each time step, we can get a sense of the average trajectory of the data.

To illustrate this application, we used the heart rate and systolic ABP trend data (1 sample/min) from the patient studied in Chapter 3. The results are summarized in Figure 5.2 and as a flip-book movie in bottom right-hand-corner of this thesis. In addition to plotting the centers of mass at each time step, we introduced a color gradient to make it easier to follow the trajectory of the patient data. Darker shades represent more recent data points.



Ideally, the information displayed in Figure 5.2 would be plotted in real time. That is, one would see the trajectory move as new trend

[19] See Section 4.6.1 for an explanation.

samples are recorded. Also, it may make sense to allow the user to decide how far back the trajectories should be plotted. For example, it may be sensible to limit the trajectory information to values recorded in the last 24 hours.

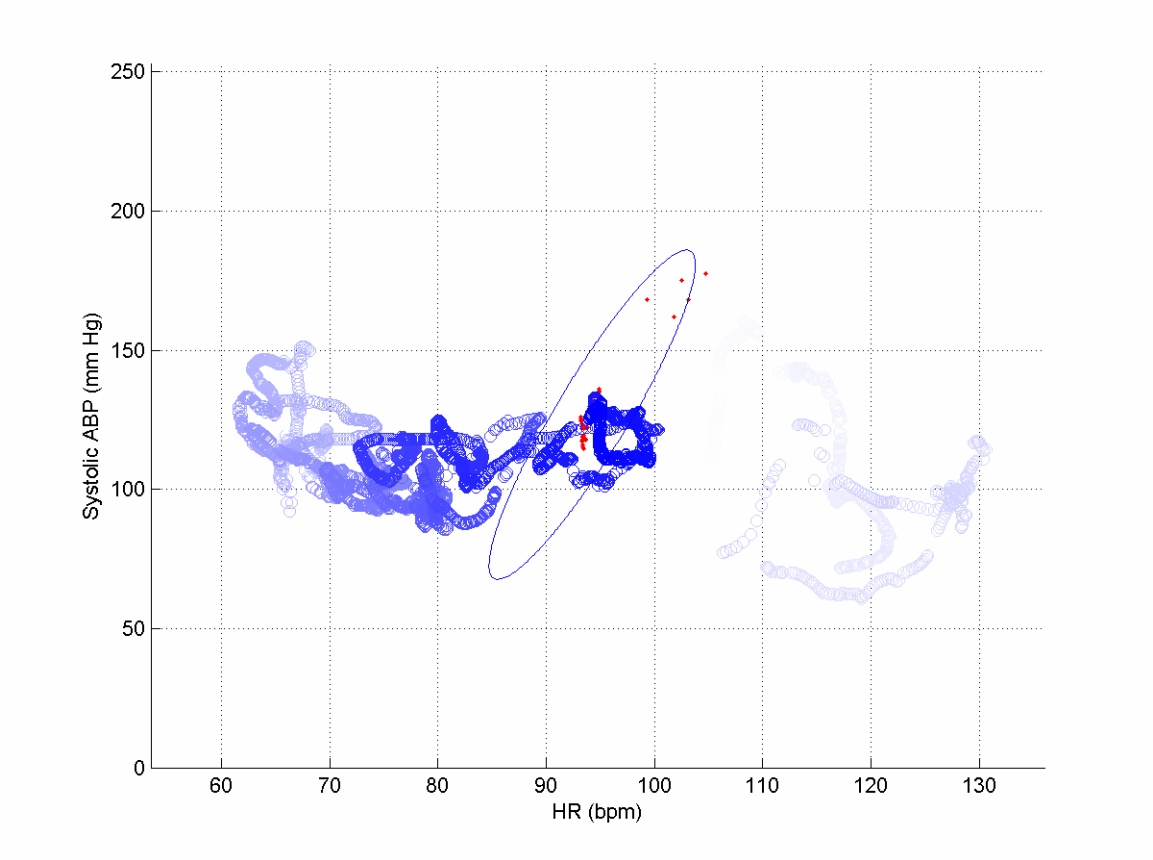


Figure 5.2: Phase space trajectory of heart rate and systolic ABP trend data (1 sample/min) lasting approximately 3 days. Darker shades of blue represent more recent data points. Lighter shades correspond to older time points. In addition to displaying the trajectory, the current clustering ellipse is shown along with the corresponding points being clustered.

5.3.2 Detecting Changes in Cardiovascular Data

The trajectories in Figure 5.2 allow us to see the evolving trajectories of data in phase space as a function of time. From these plots, it is possible to visually detect significant changes in phase space and determine whether these changes are good or bad. One reasonable question to pose is whether or not the detection of these changes can be automated using some of the tools we have already developed.

To simplify the problem, we ignored the physiological aspect of determining the significance of the changes (i.e. whether the change was good or bad) and focused on the

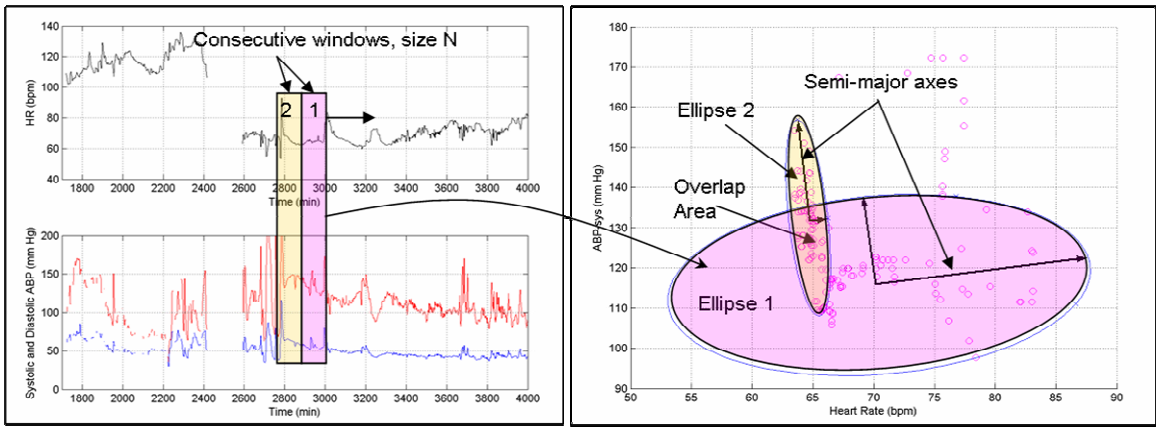


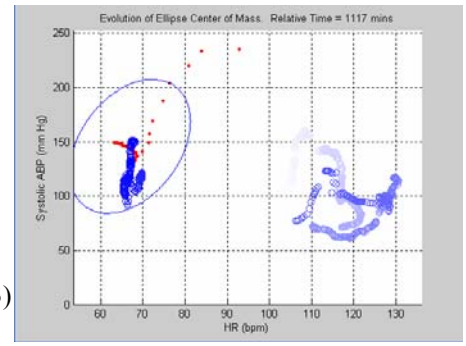
Figure 5.3: State change detection using two consecutive, moving windows.

problem of detecting significant changes in the data. Using some of our previously developed ideas, we proposed the scheme depicted in Figure 5.3.

Instead of having a single moving window, as we did for the data tracking problem, two consecutive moving windows were used. At every time step an ellipse was generated for each window and common features of and relationships between the ellipses were examined.

The reasoning for this scheme was that given that a proper window size is chosen, data in steady state should produce two ellipses with very similar features, while changes in the data would yield ellipses with different features. Thus by examining common features and the relationship between ellipses one could determine whether the data was changing or steady.

Although we considered a variety of similarity criteria, the two that provided the most meaningful results were the difference in the semi-major axis lengths and the distance between the ellipse centers of mass.[20] These similarity criteria were defined as follows:



[20] Other criteria we looked at included the difference in semi-minor axes, the angle between clustering ellipses, the percentage overlap between ellipses referenced to the ellipse generated by the lagging window (i.e. "window 2" in Figure 5.3) and the ratio of ellipse areas.

Distance metric : The distance between the centers of mass of the two ellipses.

Principal axis metric: The length of the major axis of ellipse 2 minus the length of the major axis of ellipse 1.

where the vertical bars in the first metric denote the Euclidean distance.

The distance metric gives a sense of how much the average of the data changes from one window of data to the next. A large value indicates a large change while a small value indicates that the average trajectory of the data hasn't changed much over a window of time.

The principal axis metric, gives a sense of how the average variation in the data has changed. A negative value indicates that the variation of the data in the leading window is less than the average variance of the lagging window. On the other hand, positive value indicates that the variance of the data in the leading window has increased relative to the previous window.

Ideally, for a set of data to be considered steady, we would expect the distance metric to be small and principal axis metric to be near zero. This indicates that the signals are constrained to a particular region of phase space and that the variation in the data is nearly constant. Large values in the distance metric indicate that the data is starting to move away from steady state.

Similarly, large deviations from zero in the principal axis metric indicate changes in the average fluctuations of the data. Of particular interest are the regions where the principal axis metric is positive, signifying that the variation in the new data is increasing relative to the previous data. A positive change can be indicative of either a large trend in the data (which is already captured by the distance metric) or an increase in the local variation of the data, if the average is not changing very much. In some cases, it may be useful to look at negative values in the principal axis metric, which indicate that the local variation of the data is decreasing. However, for the resolution of cardiovascular data we are using, decreases in variability are usually favorable, since they indicate that the patient is stabilizing[21].

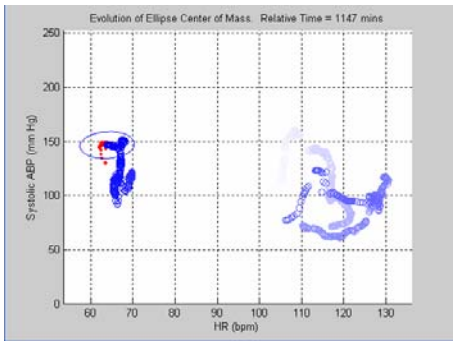
[21] One case where decreased variability has shown to be negative is at the level of beat-by-beat variability. Under this scenario it may be indicative of increased likelihood of mortality if this reduction in variability is preceded by an acute myocardial infarction (also known as a heart attack) [Malik et al., 1996].

The results of applying these metrics on a segment of beat-by-beat data from the patient studied in Chapter 3 using 60 second moving windows are shown in Figure 5.4 and Figure 5.5.

A quick inspection of Figure 5.4 indicates that the metrics seem capture all the significant trends in the data. Upon closer scrutiny we notice that there is a small lag between the metrics and the trends in the data. This lag, which is approximately 20 seconds, is due to the window length chosen.

It is also clear that the principal axis metric is more sensitive than the distance metric. This is more evident when the metrics are applied to noisier data (see Figure 5.5). The sensitivity is due to the influence of outliers on the ellipses.

One way to address this problem is to increase the window size. However, in doing this we face the tradeoff mentioned in Section 5.2, namely the exchange of outlier sensitivity for reduced ability to detect fast trends in the data. Furthermore, an increased window size will increase the delay of the metric in response to trends in the data. Another solution may be to do some filtering on the incoming data and/or smooth the metric using some sort of averaging. This issue and that of interpreting the metrics automatically are left as future work.



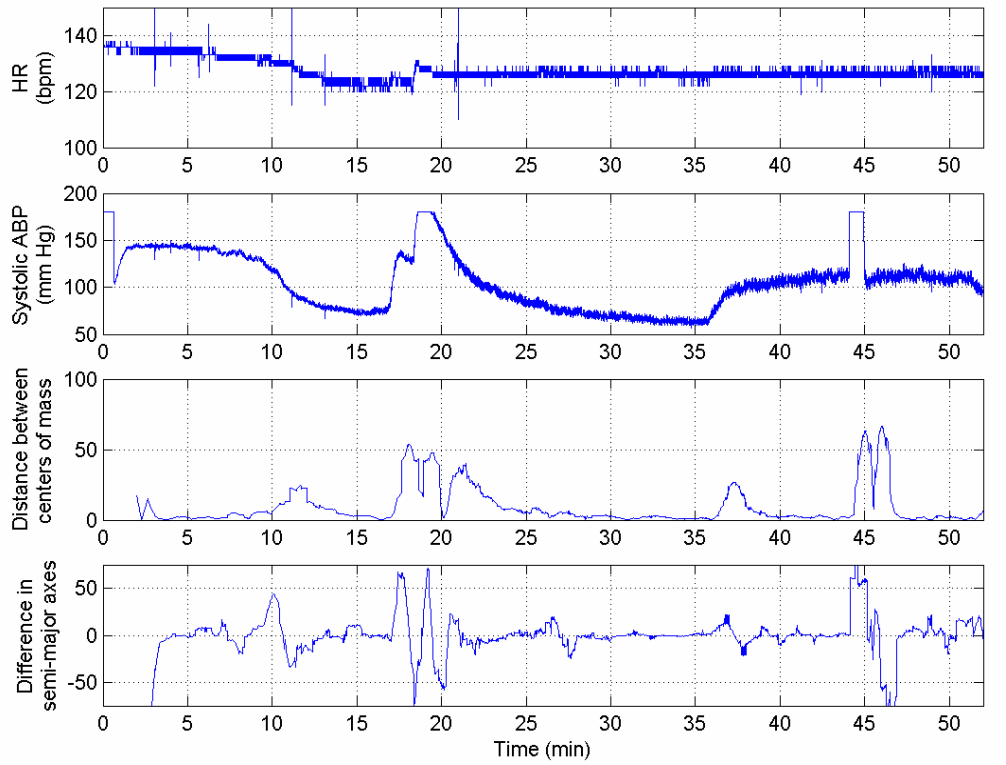


Figure 5.4: Similarity metrics to assess changes in steady state of cardiovascular data. The moving windows are each 1 minute long.

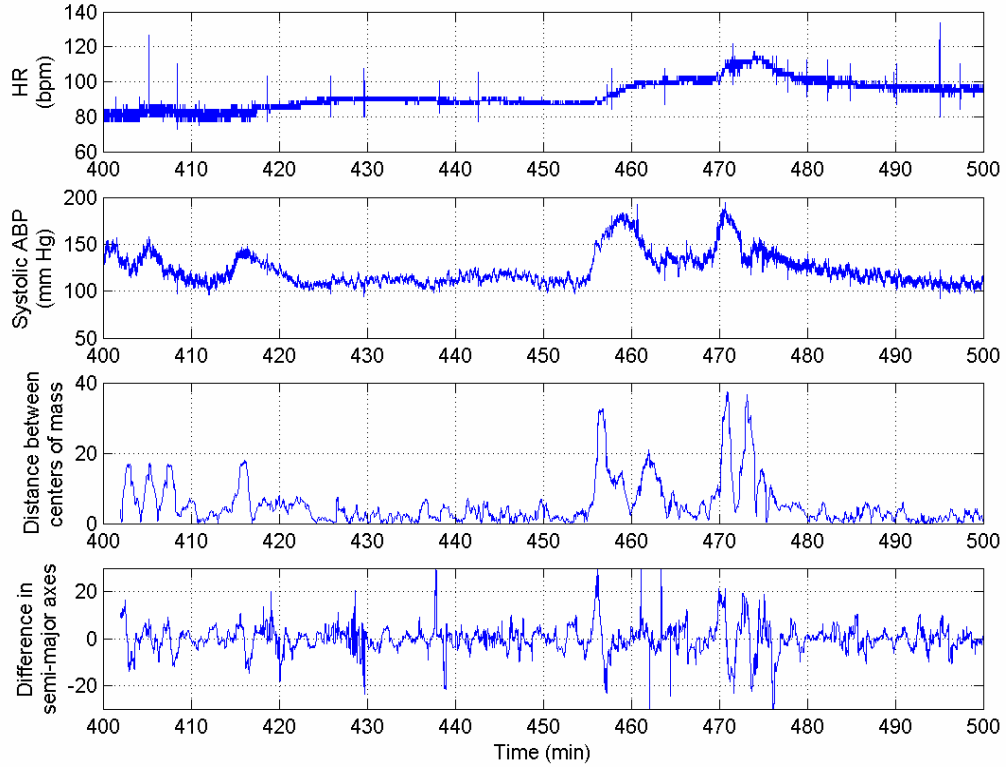


Figure 5.5: Another example of the similarity metrics on a different data segment.

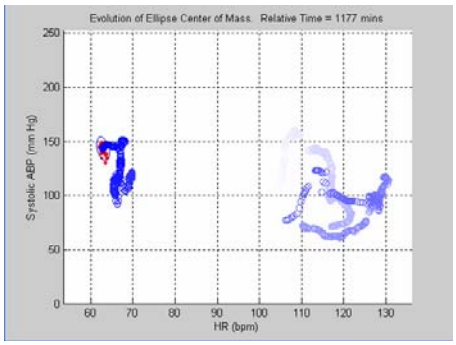
5.4 Summary and Remarks

This chapter presented several applications of SVD clustering. We extended the clustering algorithm to track data in phase space and then used the data tracking algorithm to visualize trajectories in phase space and develop metrics to detect significant changes in the data.

Although the trajectory visualization is pretty well developed, much work remains in the area of detecting changes in phase space. One fundamental problem that needs to be addressed is the choice of window size for the metrics. This choice dictates the tradeoff between metric sensitivity and the resolution at which changes can be detected.

Furthermore, there is still a need to develop a means to automatically interpret the metrics. The solution may involve some sort of thresholding of the metrics or perhaps something more elaborate.

There is also a need to generalize the metrics to other signals. In computing the metrics, we assumed that incremental changes in heart rate and arterial pressures were equally significant. Although this is a fairly good assumption for the HR and ABP signals, the assumption breaks down if we compare either of these signals with signals of low amplitude, such as CVP or PAP. Under this latter scenario, some type of weighting of the metric would be called for.



Chapter 6

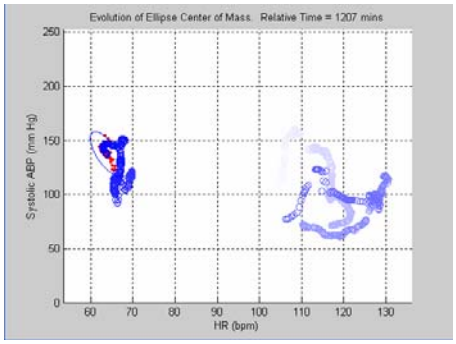
Conclusions and Future Work

The purpose of this thesis was to gain an understanding of some of the signals measured in the ICU. Furthermore, we sought out to explore and develop several techniques for visualizing and processing cardiovascular data. These techniques were tested on real signals recorded from a patient in intensive care. In this chapter, we summarize the main contributions of this effort and suggest directions for future work.

6.1 Summary and Contributions

Chapter 2 introduced the signals we made use of throughout this thesis. In addition to describing the importance of these signals, we discussed the significance of data on different time scales and presented some of the ways in which data is currently visualized in the ICU. We also touched on the problem of signal noise and suggested some methods for removing undesired features from the data.

By way of a real-world example, Chapter 3 illustrated the power of phase space and spectral analysis on cardiovascular data. Phase space provided a framework for concisely representing the medical history of a patient and allowed us to visualize trajectories of and correlations between different variables. Spectral analysis looked at the beat-by-beat variability of the signals and gave a sense of how the cardiovascular control systems mediated the variability of these signals.



Chapter 4 built on the concepts of phase space presented in Chapter 3. Specifically, we focused on the problem of tracking the dynamic changes of signals in phase space. As a first step for tracking these changes, we developed a novel data clustering method based on the singular value decomposition. The SVD clustering algorithm generated a tight fitting ellipse around the data, which succinctly summarized the distribution of data in phase space.

Finally, in Chapter 5 we extended the SVD clustering algorithm to track data in phase space. Subsequently, we introduced applications of SVD clustering for visualizing changes in phase space and detecting significant changes in the variables.

6.2 Suggestions for Future Work

Many interesting questions and ideas arose while conducting this research. As there was not nearly enough time to get to all of them, we list below some future research areas.

Phase space analysis During our studies using phase space, we focused primarily on heart rate and arterial blood pressure. It may be of interest to look at other variables, such as pulmonary artery pressure, central venous pressure, cardiac output, and arterial pulse pressure. The latter of these, which is a crude estimate of stroke volume[22], is of particular interest, since it is regularly used in the clinical setting to assess heart function.

Continued spectral analysis work In our preliminary exploration of heart rate and ABP variability we demonstrated that there is value in tracking these spectra as a function of time. In the future, it may be interesting to see if we can incorporate our understanding of physiology to understand if and/or how significant clinical events contribute to changes in the spectra.

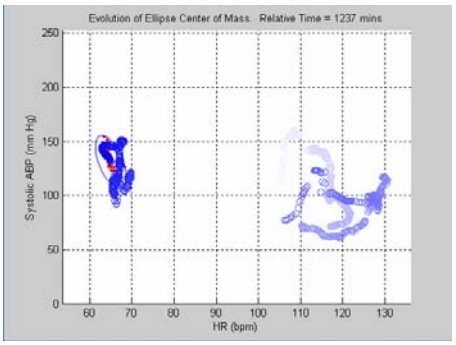
Clustering efficiency evaluation There is currently no metric for evaluating how well the SVD algorithm clusters a set of data. One metric that may be interesting to look at is the area of the tightest enclosing ellipse for a set of data. A good starting point may be to

[22] The amount of blood pumped per cardiac cycle.

use the algorithm designed by Gärtner and Schönherr (1998) to compute the tightest enclosing ellipse for a set of data, and then compare the area of this optimal ellipse with the area of the ellipse computed using SVD clustering.

Continued work with ellipse similarity metrics Work remains to be done to interpret the change metrics and address the sensitivity of the principal axis metric. There is also a need to generalize the metrics to signals with different dynamic ranges. One solution to this latter problem might be to introduce a weighting function for each signal. Once these issues are resolved, the next step is to look into alarm generation based on a combination of the change metrics and physiological interpretations of the data trajectories (see Figure 3.6). It may be interesting to evaluate how well these alarms perform compared to current alarm methods.

Examining more patients Lastly, there is also a need to look at a large population of patients. All the work done in this thesis focused on data from a single individual. If one is to make any claims about the effectiveness of the methods presented in this thesis, it is important to assess how well these techniques perform on a diverse class of patients.



Appendix A:

Properties of the SVD

A.1 Showing that the First R Columns of the U, V Matrices Span the Column Space and Row Space of the A Matrix

We are told that *A* is a matrix with rank r. To show that the row and column space of *A* are spanned by the first r columns of U and V matrix, we need to look at the product of $A^T A$ and AA^T.

$$A^T A = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma U^T U\Sigma V^T = V\Sigma^2 V^T \quad (7.1)$$

$$AA^T = U\Sigma V^T (U\Sigma V^T)^T = U\Sigma V^T V\Sigma U^T = U\Sigma^2 U^T \quad (7.2)$$

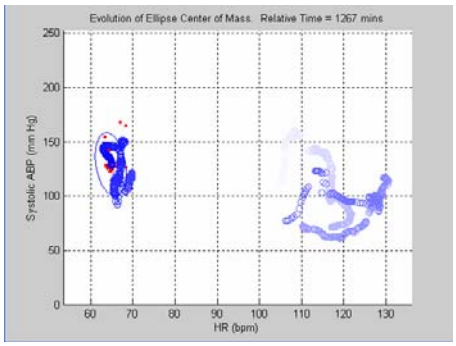
$$\Sigma^2 = (\Sigma_{i,j})^2 = \sigma_i^2; i = j \quad (7.3)$$

Note that AA^T is an MxM matrix and $A^T A$ is an NxN matrix, both with rank *r*. Rearranging Equations 7.1 and 7.2, we get:

$$(A^T A)V = V\Sigma^2 \quad (7.4)$$

$$(AA^T)U = U\Sigma^2 \quad (7.5)$$

This form shows us that the columns of the matrices of V and U are the eigenvectors of the $A^T A$ and AA^T, respectively. We also see that the matrix Σ (known as the matrix of singular values) is



related to the matrix of eigenvalues Λ by the square root. Finally, because $A^T A$ and AA^T have rank r, only the first r columns of U and V are linearly independent. Hence, the first r columns of V span the row space of A, $\Re^N$, and the first r columns of U span the column space of A, $\Re^M$.

A.2 Proof that the Columns of the U Matrix Capture the Directions of Maximum Variance in the Column Space of A [Adapted from Muller, Magaia, and Herbst (2004)]

The problem we are trying to solve involves finding an orthonormal basis, $\{\mathbf{a}_1, \ldots, \mathbf{a}_M\}$, whose basis vectors point in the directions of maximum variance for a set of vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ living in the column space of the matrix A, where $A = [\mathbf{x}_1\ \mathbf{x}_2 \ldots \mathbf{x}_N]$.

The basic strategy is to first find the direction in which the variance is largest. Then we find the second largest direction of variation, orthogonal to the first. We then solve for the next largest direction of variation, with the constraint that the direction be orthogonal to all the previous ones. This process continues until the m directions of variation are found, where m is the dimension of the $\mathbf{x}_k$'s.

A.2.1 Finding the Largest Direction of Variation

Mathematically, we are trying to solve the following:

$$\underset{\|\mathbf{a}\|=1}{\arg\max} \frac{1}{N}\sum_{k=1}^{N}(\mathbf{a}^T\mathbf{x}_k)^2 - \left(\frac{1}{N}\sum_{k=1}^{N}\mathbf{a}^T\mathbf{x}_k\right)^2 \tag{7.6}$$

To simplify the problem, we require that the $\mathbf{x}_k$'s have a zero mean ($\frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k = \vec{0}$). If this is not the case, we can simply define a new set of vectors, $\mathbf{y}_k = \mathbf{x}_k - \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k$. This simplifies Equation 7.6 to:

$$\arg\max_{\|\mathbf{a}\|=1} \frac{1}{N}\sum_{k=1}^{N}(\mathbf{a}^T\mathbf{x}_k)(\mathbf{a}^T\mathbf{x}_k)^T \tag{7.7}$$

Distributing out the contents inside the parentheses, and rewriting in terms of A= [$\mathbf{x}_1$ $\mathbf{x}_2$ … $\mathbf{x}_N$], we can re-express Equation 7.7 as:

$$\arg\max_{\|\mathbf{a}\|=1} \mathbf{a}^T AA^T\mathbf{a} \tag{7.8}$$

Next we make use of the SVD to rewrite AA^T as $(U\Sigma V^T)$ $(U\Sigma V^T)^T$= $U\Sigma\Sigma U^T$=$U\Lambda U^T$, where Λ is a diagonal matrix with the pivot entries corresponding to the variances in the directions of the columns of U. These variances are sorted by size, with the largest entry found at $\Lambda_{1,1}$ and the smallest entry at $\Lambda_{M,M}$.

$$\arg\max_{\|\mathbf{a}\|=1} \mathbf{a}^T U\Lambda U^T\mathbf{a} \tag{7.9}$$

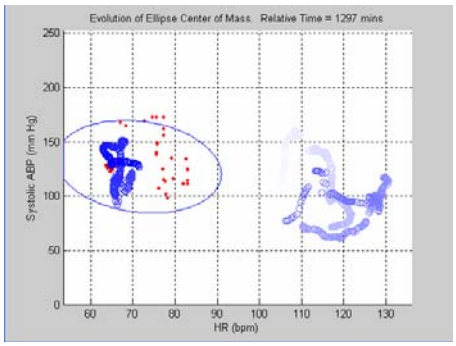
Letting $\mathbf{b}$=$U^T\mathbf{a}$ and noting that $\|\mathbf{b}\| = \|U^T\mathbf{a}\| = \|\mathbf{a}\| = 1$ we are then left with:

$$\arg\max_{\|\mathbf{b}\|=1} \mathbf{b}\Lambda\mathbf{b}^T = \arg\max_{\|\mathbf{b}\|=1} \sum_{k=1}^{M}\lambda_k\|b_k\|^2 \tag{7.10}$$

Since $\mathbf{b}$ must be of unit length and λ_1 is the largest eigenvalue, all we need to do to maximize Equation 7.10 is to choose $\mathbf{b} = \mathbf{e}_1 = [1\ 0\ \ldots\ 0]$. Substituting back into $\mathbf{a}$ we find $\mathbf{a} = U\mathbf{b} = \mathbf{u}_1$. This shows that $\mathbf{u}_1$ captures the direction of largest variation in the data and that the length in this direction is λ_1.

A.2.2 Finding the 2nd, 3rd, … , mth Largest Directions of Variation

Having found the largest direction of variation it is relatively straightforward to show how to find all the others. To find the second largest direction of variation we just need to solve Equation 7.10, with the added constraint that the new direction must be orthogonal to the first. Since we know that the U



matrix from the SVD is already orthogonal, this simplifies Equation 7.10 to

$$\arg\max_{\left\|\vec{b}\right\|=1} \sum_{k=2}^{M} \lambda_k \left\| b_k \right\| \tag{7.11}$$

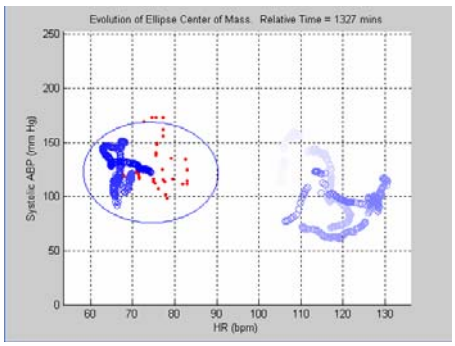
Using the same logic as before, we choose $\mathbf{b} = \mathbf{e}_2 = [0\ 1\ \ldots\ 0]$ and then solve for $\mathbf{a}$ to get we $\mathbf{a} = U\mathbf{b} = \mathbf{u}_2$. Thus $\mathbf{u}_2$ captures the direction of largest variation in the data and that the length in this direction is λ_2.This process can be repeated to find all the directions of variation, which turn out to be the $\mathbf{u}_i$'s, along with their respective lengths, which are the λ_i's.

Appendix B:

Commented MATLAB Code

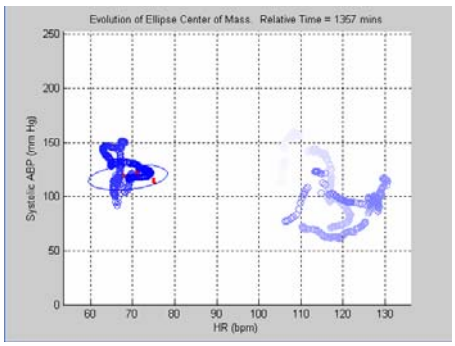
B.1 SVD_cluster.m

```
function [vectors,data_mean,pct_pts_in_ellipse]=svd_cluster(data,plot_number,remove_points)
%[vectors,data_mean,pct_pts_in_ellipse]=svd_cluster(data,plot_number,remove_points)
%Technique to cluster 2D and 3D data using the singular value decomposition.
%The data is clustered by the tightest fitting ellipse possible around the
%data.  To make the ellipse fit even tighter around the data, points in
%the dataset can be ignored in the ellipse calculation by assigning the
%variable "remove_points" with a number corresponding to the number of
% points to remove from the data set.  For the 2D and 3D case,
%results can be plotted by assigning a plot number to the
%"plot_number" variable.
%
%More specifically, the input-output information is as follows.
%
%INPUTS
%
%Data: An mxn data vector, where m is the dimension of the data and n is
%the number of points.  In other words, each column of "Data" represents a
%single point in the m-dimensional phase-space.
%
%Plot_number: By default, a plot is generated of how the data
is clustered
%for 2 and 3 dimensional data.  If one wishes to plot the data
in a
%particular figure, assign "plot_number" the number
```



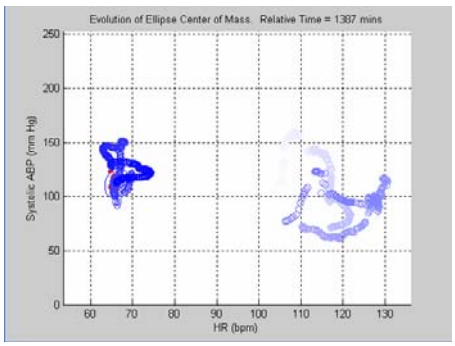
```
corresponding to the
%figure number you wish to generate.  Assigning "0" to this variable
%supresses the plot.
%
%Remove_points: When assigned an integer value N, N points are removed from
%the SVD calculation.  The points to remove are chose in an iterative manner.
%At each iteration, the point at the boundary of the tightest fitting ellipse is removed.
%The reason this point is removed is because it is the one that keeps the ellipse from
%shrinking any further.  By default, enough points are removed such that 95% of
%
%
%
%
%
%OUTPUTS
%
%Vectors: Stores the vectors (centered about the origin) corresponding to
%the principal semi-axes of the clustering ellipse.  The vectors are the
%columns of this matrix.
%
%Data mean: Values indicating the offset of the ellipse from the origin.
%
%Pct_pts_in_ellipse: Gives the percentage of points enclosed by the
%clustering ellipse.
%
%In case you would like to generate the plots yourself, here is some code
%to do this.
%
%To reconstruct a 2D clustering ellipse around a data set "data2D", type
%the following code into the MATLAB prompt.
%
%>>[vectors,data_mean,pct_pts_in_ellipse]=svd_cluster(data2D,0,0);
%>>vector_sizes=sqrt(sum(vectors.^2));
%>>[Ymax,I]=max(vector_sizes);
%>>[Ymin,J]=min(vector_sizes);
%>>theta=atan2(vectors(2,I),vectors(1,I));
%>>rotation_matrix=[cos(theta),-sin(theta);sin(theta),cos(theta)];
```

```
%>>half_ellipse=[linspace(-Ymax,Ymax);sqrt(1-(linspace(-Ymax,Ymax)/Ymax).^2)*Ymin];
%>>ellipse=[half_ellipse,-half_ellipse];
%>>rotated_ellipse=rotation_matrix*ellipse;
%>>rotated_ellipse=rotated_ellipse;
%>>hold on;
%>>figure(1);
%>>plot(data2D(1,:),data2D(2,:),'o');
%>>plot(rotated_ellipse(1,:)+data_mean(1),rotated_ellipse(2,:)+data_mean(2),'b');
%>>hold off;grid on;
%
%To the 3D clustering ellipse for a set of 3D data, "data3D" type:
%
%>>[vectors,data_mean,pct_pts_in_ellipse]=svd_cluster(data3D,0,0);
%>>vector_sizes=sqrt(sum(vectors.^2));
%>>x_rot=vectors;
%>>x_original=[vector_sizes(1) 0 0;0 vector_sizes(2) 0;0 0 vector_sizes(3)];
%>>rotation_matrix=x_rot*inv(x_original);
%>>[x,y,z]=ellipsoid(0,0,0,vector_sizes(1),vector_sizes(2),vector_sizes(3),30);
%>>x_rotated=x*rotation_matrix(1,1)+y*rotation_matrix(1,2)+z*rotation_matrix(1,3);
%>>y_rotated=x*rotation_matrix(2,1)+y*rotation_matrix(2,2)+z*rotation_matrix(2,3);
%>>z_rotated=x*rotation_matrix(3,1)+y*rotation_matrix(3,2)+z*rotation_matrix(3,3);
%>>hold on;
%>>plot3(x_rotated+data_mean(1),y_rotated+data_mean(2),z_rotated+data_mean(3),'b');
%>>plot3(data3D(1,:),data3D(2,:),data3D(3,:),'o');
%>>hold off;grid on;
%
%Created by Carlos Renjifo
%October 21, 2004
%Last modified: May 17, 2005
if nargin < 1
   load patients_complete
   data=A3833n(5400:5900,1:2)';
   remove_points=0;
elseif nargin < 3
   remove_points=0;
end
```



```
if remove_points > (size(data,2)-2)
   error('Number of points removed must be less than (number of points - 2)');
end

dimension=size(data,1);
original_num_pts=size(data,2);
original_data=data;
data_mean=mean(data,2);
points_to_remove=[];
%Remove the points that keep the clustering ellipse from forming a tighter
%fit around the data.  These are the points that touch the boundary of the
%ellipse.  The point removal must be done recursively to determine what the
%next constraining point is after the previous limiting data point is
%removed.  This portion of code takes the longest to run since the SVD must
%be recomputed at each iteration

for k=1:remove_points
   %Subtract out mean from data
   data_mean=mean(data,2);
   data_centered=data-data_mean*ones(1,size(data,2));
   %Take the SVD
   [U,S,V]=svd(data_centered);
   S=S(1:dimension,1:dimension);
   V=V';
   %Determine which point needs to be removed and remove it
   distances=sqrt(sum(V(1:dimension,:).^2));
   [max_distance,index]=max(distances);
   points_to_keep=setdiff([1:size(data,2)],index);
   points_to_remove=[points_to_remove,data(:,index)];
   data=data(:,points_to_keep);
end

%Take SVD of the reduced data (with k points removed, where k =
%remove_points)
data_mean=mean(data,2);
data_centered=data-data_mean*ones(1,size(data,2));
[U,S,V]=svd(data_centered);
```

```
S=S(1:dimension,1:dimension);
V=V';
distances=sqrt(sum(V(1:dimension,:).^2));
[max_distance,index]=max(distances);
scaling_factor=max_distance;

%Rescale principal axes of ellipse
vectors=U*S*scaling_factor;

%Determine percentage of points inside ellipse
thresh=1+1e-12; %The strange threshold is to circumvent a rounding issue in matlab
temp=inv(vectors)*(original_data-data_mean*ones(1,size(original_data,2)));
temp2=sum(temp.^2,1);
num_pts_in_ellipse=sum(sqrt(temp2)<=thresh);
pct_pts_in_ellipse=num_pts_in_ellipse/original_num_pts*100;

%Plot data
if nargin < 2
   figure;
elseif plot_number == 0
   return;
else
   figure(plot_number);
end

if dimension == 2
   %Generate clustering ellipse
   vector_sizes=sqrt(sum(vectors.^2));
   [Ymax,I]=max(vector_sizes);
   [Ymin,J]=min(vector_sizes);
   theta=atan2(vectors(2,I),vectors(1,I));
   rotation_matrix=[cos(theta),-
sin(theta);sin(theta),cos(theta)];
   half_ellipse=[linspace(-Ymax,Ymax,1e4);sqrt(1-(linspace(-
Ymax,Ymax,1e4)/Ymax).^2)*Ymin];
   ellipse=[half_ellipse,-half_ellipse];
   rotated_ellipse=rotation_matrix*ellipse;
```

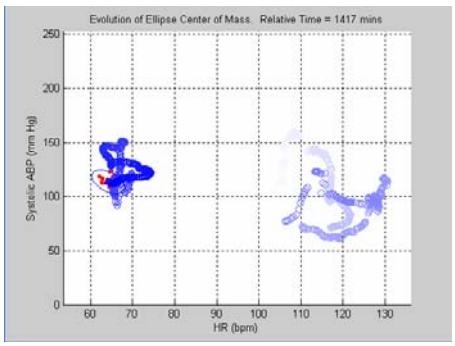


```
    rotated_ellipse=rotated_ellipse;
    hold on;
    %Plot data and clustering ellipse
    plot(original_data(1,:),original_data(2,:),'mo');
    if k>0
    plot(points_to_remove(1,:),points_to_remove(2,:),'xb');
    end
    plot(data_mean(1),data_mean(2),'ko','markerface','k');
    plot([data_mean(1),vectors(1,1)+data_mean(1)],[data_mean(2),vectors(2,1)+data_mean(2)]);
    plot([data_mean(1),vectors(1,2)+data_mean(1)],[data_mean(2),vectors(2,2)+data_mean(2)]);
    plot(vectors(1,:)+data_mean(1),vectors(2,:)+data_mean(2),'s','markerface','g');
    plot(rotated_ellipse(1,:)+data_mean(1),rotated_ellipse(2,:)+data_mean(2),'b');
    if k>0; legend('Original data','Points removed from SVD computation');end;hold off;
    title(cat(2,'Ellipse contains ',num2str(pct_pts_in_ellipse),'% of points.  ',num2str(size(original_data,2)-
num_pts_in_ellipse),' outside ellipse.'));
    xlabel('Heart Rate (bpm)');
    ylabel('ABP-sys (mm Hg)');
    axis equal
elseif dimension == 3
    %Generate 3D ellipsoid
    vector_sizes=sqrt(sum(vectors.^2));
    x_rot=vectors;
    x_original=[vector_sizes(1) 0 0;0 vector_sizes(2) 0;0 0 vector_sizes(3)];
    rotation_matrix=x_rot*inv(x_original);
    [x,y,z]=ellipsoid(0,0,0,vector_sizes(1),vector_sizes(2),vector_sizes(3),30);
    x_rotated=x*rotation_matrix(1,1)+y*rotation_matrix(1,2)+z*rotation_matrix(1,3);
    y_rotated=x*rotation_matrix(2,1)+y*rotation_matrix(2,2)+z*rotation_matrix(2,3);
    z_rotated=x*rotation_matrix(3,1)+y*rotation_matrix(3,2)+z*rotation_matrix(3,3);
    hold on;
    %Plot clustering ellipsoid and data
    plot3(original_data(1,:),original_data(2,:),original_data(3,:),'mo');
    if k>0
    plot3(points_to_remove(1,:),points_to_remove(2,:),points_to_remove(3,:),'xb');
    end
    plot3(x_rotated+data_mean(1),y_rotated+data_mean(2),z_rotated+data_mean(3),'b');
    if k>0; legend('Original data','Points removed from SVD computation');end;hold off;grid on;
    title(cat(2,'Ellipse contains ',num2str(pct_pts_in_ellipse),'% of points'));
```

```
    xlabel('Heart Rate (bpm)');
    ylabel('Mean ABP (mm Hg)');
    zlabel('Pulse Pressure (CO estimate) (mm Hg)');
end
```

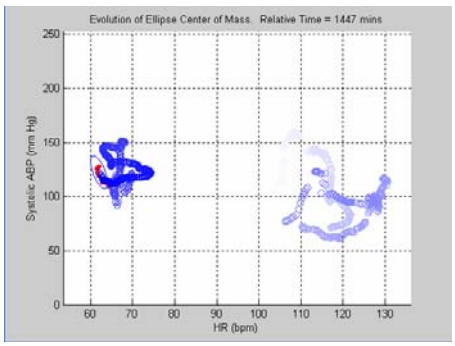
B.2 State_track_v2.m

```
function
[distance,difference_in_major_axis,trajectory_direction,cluster_movie]=state_track_v2(data,window_size,p
ct_in_ellipse,rec_movie,plot_data)
%function
[distance,difference_in_major_axis,trajectory_direction,cluster_movie]=state_track_v2(data,window_size,p
ct_in_ellipse,rec_movie,plot_data)
%State_track_v2 visually tracks the evolution of a set of 2D or 3D data by
%clustering data from two consecutive,non-overlapping windows of length N.
%Each window of data is clustered by an ellipse generated using the
%singular value decomposition.  In addition to visually tracking the data
%and clustering it with ellipses, changes in the data are assesed by
%calculating common metrics between the clustering ellipses
%
%EXTERNAL FUNCTION CALLS: This function calls on the function svd_cluster,
%which computes the principal directions of the clustering ellipses as well
%as the offset of these vectors from the origin.
%
%INPUTS:
%
%Data: 2xN or 3xN matrix of data points, where the rows of the matrix
%represent the dimensions of the data in phase-space.
%
%Window_size: Size of the moving windows of data.  The
maximum window size
%allowed is equal to half the number of points in the data set.
%
%Pct_in_ellipse: This variable specifies the percentage of
points in each
%window that are enclosed by the clustering ellipse.  The
```



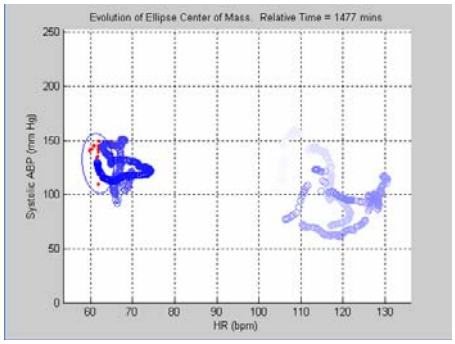
```
default value is
%95.
%
%Rec_movie: If set to 1, generates a movie of the state evolution and
%outputs it as a movie vector called "cluster_movie".  By default, no movie
%is recorded.
%
%Plot_data: When set to numbers 1 through 5, the script generates different
%types of plots.  The plot modes are:
%
%   Plot_data = 1 | Plot Mode 1: Plots the evolution of data and the
%                   clustering ellipse as a function of the moving window
%                   of size "window_size".
%   Plot_data = 2 | Plot Mode 2: Plots the evolution of center of mass of
%                   the clustering ellipse as a function of the moving
%                   window.
%   Plot_data = 3 | Plot Mode 3: Plots the ellipses generated by clustering
%                   two consecutive windows of data.  The red ellipse
%                   corresponds to the leading window and the blue ellipse
%                   corresponds to the lagging window.  These plots
%                   illustrate the two metrics we are currently exploiting
%                   (distance between centers of mass, ratio of semi-major
%                   axes) to evaluate significant changes in the data.
%   Plot_data = 4 | Plot mode 4: Plots evolution of state change metrics as
%                   a function of the moving windows.
%   Plot_data = 5 | Plot mode 5: Does estimation of values in the leading
%                   window based on statistics of lagging window & data
%                   from other variable during leading window.
%
%OUTPUTS:
%
%Distance: The distance between the centers of mass of the clustering
%ellipses at each time step.
%
%Difference in major axis: The difference in lengths of the principal axes
%(Major axis length of leading window clustering ellipse subtracted by major
%axis length of trailing window clustering ellipse).
```

```
%
%Trajectory direction: The direction of the ellipse trajectories at every
%time step.  These trajectories are computed by finding the difference
%between the center of mass of the ellipse generated by the leading window
%minus the center of mass of the ellipse of the trailing window.  The rows
%of the ellipse represent the vector direction of the trajectory at each
%time step.
%
%Cluster movie: MATLAB movie vector generated when a movie of the
%clustering video is recorded.
%
%See also: SVD_CLUSTER
%
%By Carlos Renjifo (carplos@mit.edu), Last Updated: May 17th,2005.

close all;
%Time how long the script runs
t0=clock;

if nargin < 2
   error('You must supply at least the first 2 arguments (data,window_size).  See "help state_track"')
elseif nargin < 3
   pct_in_ellipse=95;
   rec_movie=0;
   plot_data=0;
elseif nargin < 4
   rec_movie=0;
   plot_data=0;
elseif nargin < 5
   plot_data=0;
end

if rec_movie==0
   cluster_movie=0;
end

if window_size*2 > size(data,2)
```



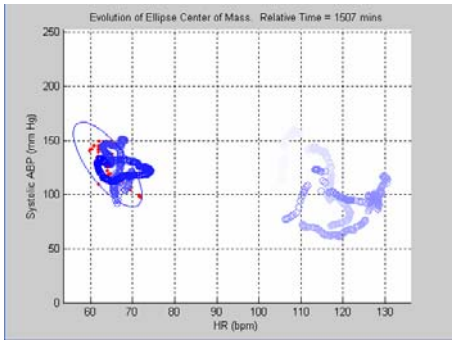
```
    error('Window size must be smaller than half the number of data points');
elseif size(data,1) > 3
    error('Data dimension (number of rows) must be less than or equal to 3');
end

%Median filter the data
medfilt_data=data';
medfilt_data=nanmedfilt1(medfilt_data,5);
data=medfilt_data';

if plot_data~=0
    fig1=figure(1); %Save handle to figure;
    winsize=get(fig1,'Position'); %Get window size of figure;
    winsize(1:2)=[0 0]; %Adjust size of the window so it includes everything, including the axes and title
    set(fig1,'NextPlot','replacechildren');%Fix the features in the plot window so each frame is the same size
    set(gcf,'doublebuffer','on');
end

%Initialize output and temporary variables for better performance
frame=1; %Keeps track of frames for movie
rec_rate=3; %Variable used to determine how often a frame is recorded
wait_time=rec_rate; %Variable to determine how long since the last recorded frame
num_dimensions=size(data,1);
max_data_vals=max(data,[],2);
min_data_vals=min(data,[],2);
size_k=size(data,2)-2*window_size;
distance=zeros(1,size_k);
difference_in_major_axis=zeros(1,size_k);
trajectory_direction=zeros(num_dimensions,size_k);
center_of_mass=zeros(num_dimensions,size_k);
if plot_data==5
    hr_mean_mse=zeros(1,size_k);
    abp_mean_mse=zeros(1,size_k);
    hr_lin_mse=zeros(1,size_k);
    abp_lin_mse=zeros(1,size_k);
end
```

```
for k=1:size_k
   clc;fprintf('k = %d',k); %Print progess
   %Create two windows of data
   data1=data(:,k:window_size+k-1);
   data2=data(:,window_size+k:2*window_size+k-1);
   %Remove NaNs from data
   data1=data1(:,isfinite(sum(data1)));
   data2=data2(:,isfinite(sum(data2)));

   %If either moving window has less than 4 data points, skip iteration
   %and set all outputs to NaN
   if or(size(data1,2) <= 4,size(data2,2) <= 4)
      distance(k)=NaN;
      difference_in_major_axis=NaN;
      trajectory_direction(:,k)=NaN;
      center_of_mass(k)=NaN;
      if plot_data ==5
         hr_mean_mse(k)=NaN;
         abp_mean_mse(k)=NaN;
         hr_lin_mse(k)=NaN;
         abp_lin_mse(k)=NaN;
      end
      continue;
   end

   %Determine the number of points to remove based on pct_in_ellipse
   pts_to_remove_1=round(size(data1,2)*(100-pct_in_ellipse)/100);
   pts_to_remove_2=round(size(data2,2)*(100-pct_in_ellipse)/100);

   %Calculate clustering ellipse features for each window
```



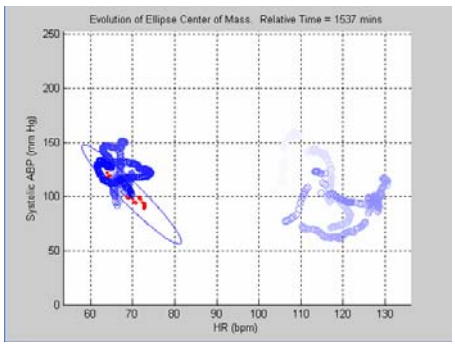
```
[vectors1,data_mean1,pct_pts_in_ellipse1]=svd_cluster(data1,0,pts_to_remove_1);
   [vectors2,data_mean2,pct_pts_in_ellipse2]=svd_cluster(data2,0,pts_to_remove_2);

   %Calculate similarity metrics
   trajectory_direction(1:num_dimensions,k)=data_mean2-data_mean1;
   distance(k)=norm(data_mean2-data_mean1);
   major_axis_length1=max([norm(vectors1(:,1)),norm(vectors1(:,2))]);
   major_axis_length2=max([norm(vectors2(:,1)),norm(vectors2(:,2))]);
   difference_in_major_axis(k)=major_axis_length2-major_axis_length1;
   center_of_mass(:,k)=data_mean2;

   %Plotting Data
   %Plot Mode 1: Plots the evolution of data and the clustering ellipse as
   %a function of the moving window of size "window_size"
   if plot_data == 1
      clf;hold on;
      if num_dimensions == 2
         %Leading Ellipse
         vector_sizes2=sqrt(sum(vectors2.^2));
         [Ymax2,I2]=max(vector_sizes2);
         [Ymin2,J2]=min(vector_sizes2);
         theta2=atan2(vectors2(2,I2),vectors2(1,I2));
         rotation_matrix2=[cos(theta2),-sin(theta2);...
            sin(theta2),cos(theta2)];
         half_ellipse2=[linspace(-Ymax2,Ymax2,200);...
            sqrt(1-(linspace(-Ymax2,Ymax2,200)/Ymax2).^2)*Ymin2];
         ellipse2=[half_ellipse2,-half_ellipse2];
         rotated_ellipse2=rotation_matrix2*ellipse2;
         plot(rotated_ellipse2(1,:)+data_mean2(1),...
             rotated_ellipse2(2,:)+data_mean2(2),'b');

         %Plot data
         plot(data2(1,:),data2(2,:),'bo');
         xlabel('HR (bpm)');ylabel('Systolic ABP (mm Hg)');grid on;
         axis([min_data_vals(1),max_data_vals(1),...
              min_data_vals(2),max_data_vals(2)]);
```

```
    elseif num_dimensions == 3
        %Leading Ellipse
        vector_sizes2=sqrt(sum(vectors2.^2));
        x2_rot=vectors2;
        x2_original=[vector_sizes2(1) 0 0;...
                     0 vector_sizes2(2) 0;...
                     0 0 vector_sizes2(3)];
        rotation_matrix2=x2_rot*inv(x2_original);
        [x2,y2,z2]=ellipsoid(0,0,0,vector_sizes2(1),...
                          vector_sizes2(2),vector_sizes2(3),30);
        x2_rotated=x2*rotation_matrix2(1,1)+...
                   y2*rotation_matrix2(1,2)+...
                   z2*rotation_matrix2(1,3);
        y2_rotated=x2*rotation_matrix2(2,1)+...
                   y2*rotation_matrix2(2,2)+...
                   z2*rotation_matrix2(2,3);
        z2_rotated=x2*rotation_matrix2(3,1)+...
                   y2*rotation_matrix2(3,2)+...
                   z2*rotation_matrix2(3,3);
        plot3(x2_rotated+data_mean2(1),y2_rotated+data_mean2(2),...
            z2_rotated+data_mean2(3),'b');

        %Plot data
        plot3(data2(1,:),data2(2,:),data2(3,:),'bo');
        grid on;view(3);
        axis([min_data_vals(1),max_data_vals(1),...
            min_data_vals(2),max_data_vals(2),...
            min_data_vals(3),max_data_vals(3)]);
        xlabel('Heart Rate (bpm)');
        ylabel('ABP mean (mm Hg)');
        zlabel('Pulse-Pressure (Est. CO)');
    end
    title(cat(2,'Data Tracking using SVD Clustering.
Relative Time = ',num2str(k),' mins'));

    %Movie Recording
    if rec_movie == 1
```



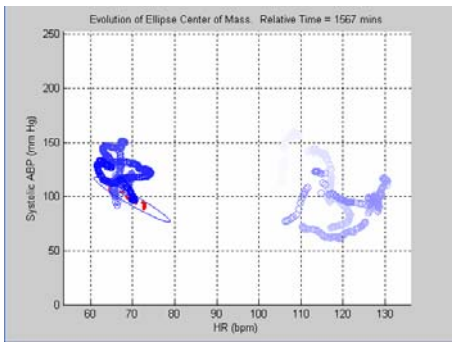
```
        if wait_time >= rec_rate
            wait_time=1;
            cluster_movie(frame)=getframe(fig1,winsize);
            frame=frame+1;
        else
            wait_time=wait_time+1;
        end
    end
    pause(0.05);

%Plot Mode 2: Plots the evolution of center of mass of the clustering
%ellipse as a function of the moving window
elseif plot_data == 2
    clf;hold on;
    if num_dimensions == 2
        %Plot data points in ellipse
        plot(data2(1,:),data2(2,:),'r.');
        %Plot center of mass
        c_map=[1-linspace(0,1,k)',1-linspace(0,1,k)',ones(k,1)];
        for w=1:k
            plot(center_of_mass(1,w),center_of_mass(2,w),'o','color',c_map(w,:));
        end
        %Plot clustering ellipse
        vector_sizes2=sqrt(sum(vectors2.^2));
        [Ymax2,I2]=max(vector_sizes2);
        [Ymin2,J2]=min(vector_sizes2);
        theta2=atan2(vectors2(2,I2),vectors2(1,I2));
        rotation_matrix2=[cos(theta2),-sin(theta2);...
            sin(theta2),cos(theta2)];
        half_ellipse2=[linspace(-Ymax2,Ymax2,200);...
            sqrt(1-(linspace(-Ymax2,Ymax2,200)/Ymax2).^2)*Ymin2];
        ellipse2=[half_ellipse2,-half_ellipse2];
        rotated_ellipse2=rotation_matrix2*ellipse2;
        plot(rotated_ellipse2(1,:)+data_mean2(1),...
             rotated_ellipse2(2,:)+data_mean2(2),'b');
        xlabel('HR (bpm)');ylabel('Systolic ABP (mm Hg)');grid on;
        axis([min_data_vals(1),max_data_vals(1),...
```

```
        min_data_vals(2),max_data_vals(2)]);

elseif num_dimensions == 3
    %Plot data points in ellipse
    plot3(data2(1,:),data2(2,:),data2(3,:),'r.');
    %Plot center of mass
    c_map=[1-linspace(0,1,k)',1-linspace(0,1,k)',ones(k,1)];
    for w=1:k
       plot3(data2(1,w),data2(2,w),data2(3,w),...
           'o','color',c_map(w,:));
    end
    %Plot clustering ellipse
    vector_sizes2=sqrt(sum(vectors2.^2));
    x2_rot=vectors2;
    x2_original=[vector_sizes2(1) 0 0;...
                 0 vector_sizes2(2) 0;...
                 0 0 vector_sizes2(3)];
    rotation_matrix2=x2_rot*inv(x2_original);
    [x2,y2,z2]=ellipsoid(0,0,0,vector_sizes2(1),...
                      vector_sizes2(2),vector_sizes2(3),30);
    x2_rotated=x2*rotation_matrix2(1,1)+...
             y2*rotation_matrix2(1,2)+...
             z2*rotation_matrix2(1,3);
    y2_rotated=x2*rotation_matrix2(2,1)+...
             y2*rotation_matrix2(2,2)+...
             z2*rotation_matrix2(2,3);
    z2_rotated=x2*rotation_matrix2(3,1)+...
             y2*rotation_matrix2(3,2)+...
             z2*rotation_matrix2(3,3);
    plot3(x2_rotated+data_mean2(1),y2_rotated+data_mean2(2),...
        z2_rotated+data_mean2(3),'b');
    grid on;view(3);
    axis([min_data_vals(1),max_data_vals(1),...
        min_data_vals(2),max_data_vals(2),...
        min_data_vals(3),max_data_vals(3)]);
    xlabel('Heart Rate (bpm)');
    ylabel('ABP mean (mm Hg)');
```



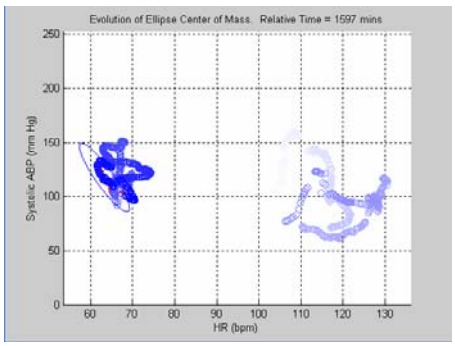
```
        zlabel('Pulse-Pressure (Est. CO)');
    end
    title(cat(2,'Evolution of Ellipse Center of Mass.  Relative Time = ',num2str(k),' mins'));
    %legend('Clustered points','Ellipse center of mass');
    %Movie Recording
    if rec_movie == 1
        if wait_time >= rec_rate
            wait_time=1;
            cluster_movie(frame)=getframe(fig1,winsize);
            frame=frame+1;
        else
            wait_time=wait_time+1;
        end
    end
    pause(0.05);

%Plot Mode 3: Plots the ellipses generated by clustering two consecutive
%windows of data.  The red ellipse corresponds to the leading window
%and the blue ellipse corresponds to the lagging window.  These plots
%illustrate the two metrics we are currently exploiting (distance
%between centers of mass, ratio of semi-major axes) to evaluate
%significant changes in the data.
elseif plot_data == 3
    clf;hold on;
    if num_dimensions == 2
        %%%%%%%%%%%%%%%%%%%%
        %Plotting 2D Data%
        %%%%%%%%%%%%%%%%%%%%
        %Create the colormaps for the data inside the ellipses.  Red
        %data corresponds to the leading window.  Blue data corresponds
        %to the trailing window.  The brighter the data point, the
        %closer it is to the end of its window.
%           size_dat1=size(data1,2);
%           size_dat2=size(data2,2);
%           c_map =[1-linspace(0,1,size_dat1)',1-linspace(0,1,size_dat1)',ones(size_dat1,1)];
%           c_map2=[ones(size_dat2,1),1-linspace(0,1,size_dat2)',1-linspace(0,1,size_dat2)'];
%
```

```
%          for w=1:size_dat1
%             plot(data1(1,w),data1(2,w),'o','color',c_map(w,:));
%          end
%          for w=1:size_dat2
%             plot(data2(1,w),data2(2,w),'o','color',c_map2(w,:));
%          end

          %Ellipse 1
          vector_sizes1=sqrt(sum(vectors1.^2));
          [Ymax1,I1]=max(vector_sizes1);
          [Ymin1,J1]=min(vector_sizes1);
          theta1=atan2(vectors1(2,I1),vectors1(1,I1));
          rotation_matrix1=[cos(theta1),-sin(theta1);...
             sin(theta1),cos(theta1)];
          half_ellipse1=[linspace(-Ymax1,Ymax1,200);...
             sqrt(1-(linspace(-Ymax1,Ymax1,200)/Ymax1).^2)*Ymin1];
          ellipse1=[half_ellipse1,-half_ellipse1];
          rotated_ellipse1=rotation_matrix1*ellipse1;
          plot(rotated_ellipse1(1,:)+data_mean1(1),...
             rotated_ellipse1(2,:)+data_mean1(2),'b');

          %Ellipse 2
          vector_sizes2=sqrt(sum(vectors2.^2));
          [Ymax2,I2]=max(vector_sizes2);
          [Ymin2,J2]=min(vector_sizes2);
          theta2=atan2(vectors2(2,I2),vectors2(1,I2));
          rotation_matrix2=[cos(theta2),-sin(theta2);...
             sin(theta2),cos(theta2)];
          half_ellipse2=[linspace(-Ymax2,Ymax2,200);...
             sqrt(1-(linspace(-Ymax2,Ymax2,200)/Ymax2).^2)*Ymin2];
          ellipse2=[half_ellipse2,-half_ellipse2];
          rotated_ellipse2=rotation_matrix2*ellipse2;
          plot(rotated_ellipse2(1,:)+data_mean2(1),...
             rotated_ellipse2(2,:)+data_mean2(2),'r');

          %Plot semi-major axes of ellipses
          plot([data_mean1(1),vectors1(1,I1)+data_mean1(1)],...
```



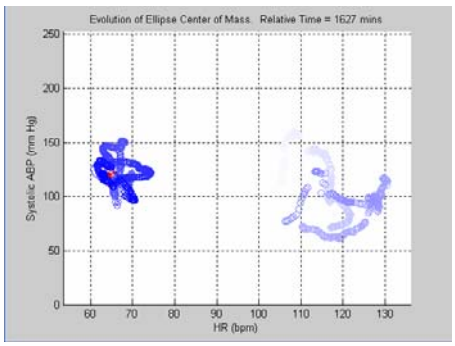
```
        [data_mean1(2),vectors1(2,I1)+data_mean1(2)],'b',...
        [data_mean2(1),vectors2(1,I2)+data_mean2(1)],...
        [data_mean2(2),vectors2(2,I2)+data_mean2(2)],'r');
    %Plot trajectory and distance metric
    plot(data_mean1(1),data_mean1(2),'marker','o','markerface','g');
    plot(data_mean2(1),data_mean2(2),'marker','s','markerface','r');
    plot([data_mean1(1),data_mean2(1)],...
       [data_mean1(2),data_mean2(2)],'k');
    hold off;grid on;
    xlabel('Heart Rate (bpm)');ylabel('Systolic ABP (mm Hg)');
    axis([min_data_vals(1),max_data_vals(1),...
         min_data_vals(2),max_data_vals(2)]);

elseif num_dimensions == 3
    %%%%%%%%%%%%%%%%%%
    %Plotting 3D Data%
    %%%%%%%%%%%%%%%%%%

    %Create the colormaps for the data inside the ellipses.  Red
    %data corresponds to the leading window.  Blue data corresponds
    %to the trailing window.  The brighter the data point, the
    %closer it is to the end of its window.
    size_dat1=size(data1,2);
    size_dat2=size(data2,2);
    c_map =[1-linspace(0,1,size_dat1)',1-linspace(0,1,size_dat1)',ones(size_dat1,1)];
    c_map2=[ones(size_dat2,1),1-linspace(0,1,size_dat2)',1-linspace(0,1,size_dat2)'];

    for w=1:size_dat1
       plot3(data1(1,w),data1(2,w),data1(3,w),...
          'o','color',c_map(w,:));
    end
    for w=1:size_dat2
       plot3(data2(1,w),data2(2,w),data2(3,w),...
          'o','color',c_map2(w,:));
    end

    %Ellipse 1
```

```
vector_sizes1=sqrt(sum(vectors1.^2));
x1_rot=vectors1;
x1_original=[vector_sizes1(1) 0 0;...
   0 vector_sizes1(2) 0;...
   0 0 vector_sizes1(3)];
rotation_matrix1=x1_rot*inv(x1_original);
[x1,y1,z1]=ellipsoid(0,0,0,vector_sizes1(1),...
   vector_sizes1(2),vector_sizes1(3),30);
x1_rotated=x1*rotation_matrix1(1,1)+...
   y1*rotation_matrix1(1,2)+...
   z1*rotation_matrix1(1,3);
y1_rotated=x1*rotation_matrix1(2,1)+...
   y1*rotation_matrix1(2,2)+...
   z1*rotation_matrix1(2,3);
z1_rotated=x1*rotation_matrix1(3,1)+...
   y1*rotation_matrix1(3,2)+...
   z1*rotation_matrix1(3,3);
plot3(x1_rotated+data_mean1(1),y1_rotated+data_mean1(2),...
   z1_rotated+data_mean1(3),'b');

%Ellipse 2
vector_sizes2=sqrt(sum(vectors2.^2));
x2_rot=vectors2;
x2_original=[vector_sizes2(1) 0 0;...
   0 vector_sizes2(2) 0;...
   0 0 vector_sizes2(3)];
rotation_matrix2=x2_rot*inv(x2_original);
[x2,y2,z2]=ellipsoid(0,0,0,vector_sizes2(1),...
   vector_sizes2(2),vector_sizes2(3),30);
x2_rotated=x2*rotation_matrix2(1,1)+...
   y2*rotation_matrix2(1,2)+...
   z2*rotation_matrix2(1,3);
y2_rotated=x2*rotation_matrix2(2,1)+...
   y2*rotation_matrix2(2,2)+...
   z2*rotation_matrix2(2,3);
z2_rotated=x2*rotation_matrix2(3,1)+...
   y2*rotation_matrix2(3,2)+...
```



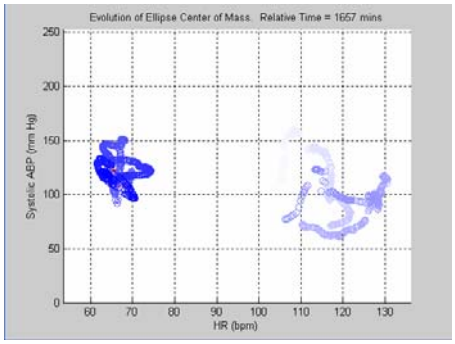
```
            z2*rotation_matrix2(3,3);
        plot3(x2_rotated+data_mean2(1),y2_rotated+data_mean2(2),...
            z2_rotated+data_mean2(3),'r');

        %Plot trajectory
        plot3(data_mean1(1),data_mean1(2),data_mean1(3),...
            'marker','o','markerface','g');
        plot3(data_mean2(1),data_mean2(2),data_mean2(3),...
            'marker','s','markerface','r')
        plot3([data_mean1(1),data_mean2(1)],...
            [data_mean1(2),data_mean2(2)],...
            [data_mean1(3),data_mean2(3)],'k');
        hold off;grid on;view(3);
        axis([min_data_vals(1),max_data_vals(1),...
            min_data_vals(2),max_data_vals(2),...
            min_data_vals(3),max_data_vals(3)]);
        xlabel('Heart Rate (bpm)');
        ylabel('ABP mean (mm Hg)');
        zlabel('Pulse-Pressure (Est. CO)');
    end
    title('State Similarity Metric Calculations Based on Ellipse Similarities');
    %legend('Red Ellipse: Cluster of leading window data.','Blue Ellipse: Cluster of trailing window
data.')

    %Movie Recording
    if rec_movie == 1
        if wait_time >= rec_rate
            wait_time=1;
            cluster_movie(frame)=getframe(fig1,winsize);
            frame=frame+1;
        else
            wait_time=wait_time+1;
        end
    end
    pause(0.05);

  %Plot mode 4: Plots evolution of state change metrics as a function of
```

```
%the moving windows
elseif plot_data == 4
   subplot(221);cla;hold on;grid on;ylim([min(data(1,:)),max(data(1,:))]);
   plot_lines([k:10:window_size+k-1],[],'b');
   plot_lines([window_size+k:10:2*window_size+k-1],[],'r');
   plot(data(1,:),'k');xlabel('Samples');ylabel('HR (bpm)');
   xlim([k-500 k+500]);

   subplot(222);hold on;grid on;
   if k~=1;plot([k-1,k],distance([k-1,k]));end
   xlabel('Samples');ylabel('Dist b/w centers of mass');
   xlim([k-500 k+500]);

   subplot(223);cla;hold on;grid on;ylim([min(data(2,:)),max(data(2,:))]);
   plot_lines([k:10:window_size+k-1],[],'b');
   plot_lines([window_size+k:10:2*window_size+k-1],[],'r');
   plot(data(2,:),'k');xlabel('samples');ylabel('ABP (mm Hg)');
   xlim([k-500 k+500]);

   subplot(224);hold on;grid on;
   if k~=1;plot([k-1,k],difference_in_major_axis([k-1,k]));end
   xlabel('Samples');ylabel('Ratio of principal axes');
   xlim([k-500 k+500]);

   %Movie Recording
   if rec_movie == 1
      if wait time >= rec_rate
         wait_time=1;
         cluster_movie(frame)=getframe(fig1,winsize);
         frame=frame+1;
      else
         wait_time=wait_time+1;
      end
   end
   pause(0.05);

%Plot mode 5: Does estimation of values in the leading
```



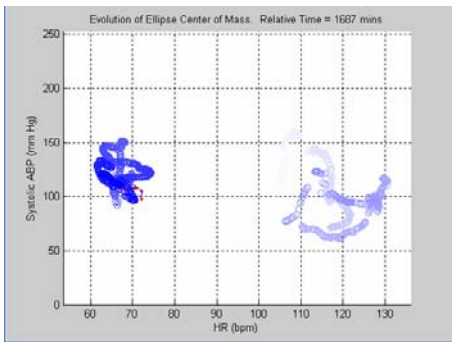
```
window based on
   %statistics of lagging window & data from other variable during leading
   %window.
   elseif plot_data == 5
      %Calculations for ellipse
      vector_sizes1=sqrt(sum(vectors1.^2));
      [Ymax1,I1]=max(vector_sizes1);
      [Ymin1,J1]=min(vector_sizes1);
      theta1=atan2(vectors1(2,I1),vectors1(1,I1));
      rotation_matrix1=[cos(theta1),-sin(theta1);...
                        sin(theta1),cos(theta1)];
      half_ellipse1=[linspace(-Ymax1,Ymax1,200);...
         sqrt(1-(linspace(-Ymax1,Ymax1,200)/Ymax1).^2)*Ymin1];
      ellipse1=[half_ellipse1,-half_ellipse1];
      rotated_ellipse1=rotation_matrix1*ellipse1;
      %Calculating HR estimate from Systolic ABP data
      hr_est=[vectors1(1,I1)/vectors1(2,I1)]*[data2(2,:)-data_mean2(2)]+data_mean1(1);
      %Calculating Systolic ABP estimate from HR data
      abp_est=[vectors1(2,I1)/vectors1(1,I1)]*[data2(1,:)-data_mean2(1)]+data_mean1(2);
      %Mean Squared Errors
      hr_mean_mse(k)=sum((data2(1,:)-mean(data2(1,:))*ones(1,window_size)).^2)/window_size;
      abp_mean_mse(k)=sum((data2(2,:)-mean(data2(2,:))*ones(1,window_size)).^2)/window_size;
      hr_lin_mse(k)=sum((data2(1,:)-hr_est).^2)/window_size;
      abp_lin_mse(k)=sum((data2(2,:)-abp_est).^2)/window_size;

      subplot(321);cla;hold on;grid on;
      plot([1:k,nan,2*window_size+k-1:size(data,2)],[data(1,1:k),nan,data(1,2*window_size+k-1:end)],'k');
      plot([k:window_size+k-1],data1(1,:),'b--');
      plot([window_size+k:2*window_size+k-1],mean(data2(1,:))*ones(1,window_size),'r--');
      plot([window_size+k:2*window_size+k-1],hr_est,'g.-');
      xlim([k-2*window_size,k+2*window_size]);
      xlabel('Samples');ylabel('HR (bpm)');
      title('Blue: Estimating Window. Red: Estimate using mean of HR. Green: Estimate based on ABP');

      subplot(323);cla;hold on;grid on;
      plot([1:k,nan,2*window_size+k-1:size(data,2)],[data(2,1:k),nan,data(2,2*window_size+k-1:end)],'k');
      plot([k:window_size+k-1],data1(2,:),'b--');
```

```
        plot([window_size+k:2*window_size+k-1],mean(data2(2,:))*ones(1,window_size),'r--');
        plot([window_size+k:2*window_size+k-1],abp_est,'g.-');
        xlim([k-2*window_size,k+2*window_size]);
        xlabel('Samples');ylabel('Sys-ABP (bpm)');
        title('Blue:Estimating Window. Red: Estimate using mean of Sys-ABP. Green: Estimate based on
HR');

        subplot(322);grid on;
        time_index=1:k;
        plot(time_index+window_size,hr_mean_mse(time_index),'r--
',time_index+window_size,hr_lin_mse(time_index),'g.-');
        xlim([k-2*window_size,k+2*window_size]);
        xlabel('Sample');ylabel('MSE for estimation of HR');
        title('Red:Estimate using mean of HR. Green:Estimate using Sys-ABP');

        subplot(324);grid on;
        plot(time_index+window_size,abp_mean_mse(time_index),'r--
',time_index+window_size,abp_lin_mse(time_index),'g.-');
        xlim([k-2*window_size,k+2*window_size]);
        xlabel('Sample');ylabel('MSE for estimation of Sys-ABP');
        title('Red:Estimate using mean of Sys-ABP. Green:Estimate using HR');

        subplot(313);cla;hold on;grid on;
        %Plot data used to estimate
        plot(data1(1,:),data1(2,:),'bo');
        %Plot data being estimated
        plot(data2(1,:),data2(2,:),'ro');
        %Plot ellipse corresponding to lagging window
        plot(rotated_ellipse1(1,:)+data_mean1(1),...
            rotated_ellipse1(2,:)+data_mean1(2),'b');
        %Highlight the principal components of the ellipse
```



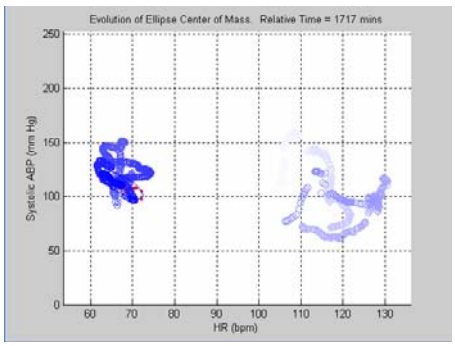
```
plot([data_mean1(1),vectors1(1,1)+data_mean1(1),nan,data_mean1(1),vectors1(1,2)+data_mean1(1)],[data
_mean1(2),vectors1(2,1)+data_mean1(2),nan,data_mean1(2),vectors1(2,2)+data_mean1(2)],'k');
      plot(data_mean1(1),data_mean1(2),'ko','markerface','k');
      plot(data_mean1(1)+vectors1(1,1),vectors1(2,1)+data_mean1(2),'gs','markerface','g');
      plot(data_mean1(1)+vectors1(1,2),vectors1(2,2)+data_mean1(2),'gs','markerface','g');
      xlabel('HR (bpm)');ylabel('Systolic ABP (mm Hg)');axis equal;
      title('Blue:Points used to estimate. Red:Points used to estimate');

      pause(0.01);
   end
end

total_time=etime(clock,t0);
minutes=floor(total_time/60);
seconds=(total_time/60-minutes)*60;
fprintf('\n Total processing time: %.0d minutes %.3f seconds',minutes,seconds);

%------------------------------------------------------------------------
%Function that plots a vertical line at the desired location on a graph.
%------------------------------------------------------------------------
function plot_lines(xin,mnmx,lntype);
%plot_lines(xin)   Plots vertical lines at points defined on the x-axis
%   using red solid lines and the current y-axis limits using a single line
%   handle
%plot_lines(xin,[a b])  Plots the lines from a to b
%
%plot_lines(xin,[],lntype)  Plots the lines using the current y-axis limits
%using the line type deined by lntype
%
%h = plot_lines(xin,[a b],lntype)  Returns the handle to the vertcal lines
if((nargin<2)|isempty(mnmx))
   mnmx = get(gca,'YLim');
end
if((nargin<3)|isempty(lntype))
   lntype = '-m';
end
xin = xin(:)';
```

```
L = length(xin);
x = reshape([xin;xin;ones(1,L)*nan;],L*3,1);

x = x(1:end-1);
x = x(:);
mnmx = mnmx(:)';

y = repmat([mnmx nan],1,L);
y = y(1:end-1);

plot(x,y,lntype);

%------------------------------------------------------------------------
%Nan-median filter
%------------------------------------------------------------------------
function y = nanmedfilt1(x,n,blksz,DIM)
%NANMEDFILT1  One dimensional median filter.
%   Y = NANMEDFILT1(X,N) returns the output of the order N, one dimensional
%   median filtering of X.  Y is the same size as X; for the edge points,
%   zeros are assumed to the left and right of X.  If X is a matrix,
%   then MEDFILT1 operates along the columns of X.
%
%   If you do not specify N, NANMEDFILT1 uses a default of N = 3.
%   For N odd, Y(k) is the median of X( k-(N-1)/2 : k+(N-1)/2 ).
%   For N even, Y(k) is the median of X( k-N/2 : k+N/2-1 ).
%
%   Y = NANMEDFILT1(X,N,BLKSZ) uses a for-loop to compute BLKSZ ("block size")
%   output samples at a time.  Use this option with BLKSZ << LENGTH(X) if
%   you are low on memory (NANMEDFILT1 uses a working matrix of size
%   N x BLKSZ).  By default, BLKSZ == LENGTH(X); this is the fastest
%   execution if you have the memory for it.
%
%   For matrices and N-D arrays, Y =
NANMEDFILT1(X,N,[],DIM) or
%   Y = NANMEDFILT1(X,N,BLKSZ,DIM) operates along
the dimension DIM.
%
```



```
%   See also MEDIAN, FILTER, SGOLAYFILT, and MEDFILT2 in the Image
%   Processing Toolbox and the NANSTATS tools.

%   Author(s): L. Shure and T. Krauss, 8-3-93
%   Copyright 1988-2002 The MathWorks, Inc.
%   $Revision: 1.8 $  $Date: 2002/03/28 17:28:51 $
%   ... updated to use nanmedian by G Clifford Mar 30th 3004

% Validate number of input arguments
error(nargchk(1,4,nargin));
if nargin < 2, n = []; end
if nargin < 3, blksz = []; end
if nargin < 4, DIM = []; end

% Check if the input arguments are valid
if isempty(n)
  n = 3;
end

if ~isempty(DIM) & DIM > ndims(x)
        error('Dimension specified exceeds the dimensions of X.')
end

% Reshape x into the right dimension.
if isempty(DIM)
        % Work along the first non-singleton dimension
        [x, nshifts] = shiftdim(x);
else
        % Put DIM in the first (row) dimension (this matches the order
        % that the built-in filter function uses)
        perm = [DIM,1:DIM-1,DIM+1:ndims(x)];
        x = permute(x,perm);
end

% Verify that the block size is valid.
siz = size(x);
if isempty(blksz),
```

```
        blksz = siz(1); % siz(1) is the number of rows of x (default)
else
        blksz = blksz(:);
end

% Initialize y with the correct dimension
y = zeros(siz);

% Call medfilt1D (vector)
for i = 1:prod(siz(2:end)),
        y(:,i) = medfilt1D(x(:,i),n,blksz);
end

% Convert y to the original shape of x
if isempty(DIM)
        y = shiftdim(y, -nshifts);
else
        y = ipermute(y,perm);
end



%-------------------------------------------------------------------
%                   Local Function
%-------------------------------------------------------------------
function y = medfilt1D(x,n,blksz)
%MEDFILT1D  One dimensional median filter.
%
% Inputs:
%   x     - vector
%   n     - order of the filter
%   blksz - block size

nx = length(x);
if rem(n,2)~=1    % n even
   m = n/2;
else
   m = (n-1)/2;
```

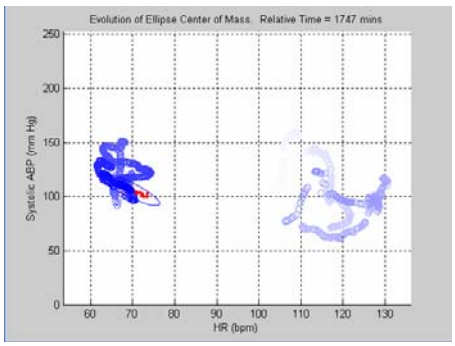


```
end
X = [zeros(m,1); x; zeros(m,1)];
y = zeros(nx,1);

% Work in chunks to save memory
indr = (0:n-1)';
indc = 1:nx;
for i=1:blksz:nx
   ind = indc(ones(1,n),i:min(i+blksz-1,nx)) + ...
        indr(:,ones(1,min(i+blksz-1,nx)-i+1));
   xx = reshape(X(ind),n,min(i+blksz-1,nx)-i+1);
   y(i:min(i+blksz-1,nx)) = nanmedian(xx);
end
```

B.3 Nanmedfilt1.m

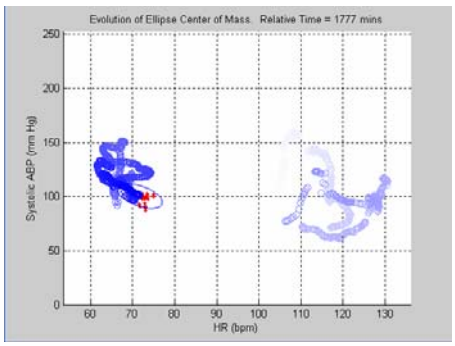
```
function y = nanmedfilt1(x,n,blksz,DIM)
%NANMEDFILT1  One dimensional median filter.
%   Y = NANMEDFILT1(X,N) returns the output of the order N, one dimensional
%   median filtering of X.  Y is the same size as X; for the edge points,
%   zeros are assumed to the left and right of X.  If X is a matrix,
%   then MEDFILT1 operates along the columns of X.
%
%   If you do not specify N, NANMEDFILT1 uses a default of N = 3.
%   For N odd, Y(k) is the median of X( k-(N-1)/2 : k+(N-1)/2 ).
%   For N even, Y(k) is the median of X( k-N/2 : k+N/2-1 ).
%
%   Y = NANMEDFILT1(X,N,BLKSZ) uses a for-loop to compute BLKSZ ("block size")
%   output samples at a time.  Use this option with BLKSZ << LENGTH(X) if
%   you are low on memory (NANMEDFILT1 uses a working matrix of size
%   N x BLKSZ).  By default, BLKSZ == LENGTH(X); this is the fastest
%   execution if you have the memory for it.
%
%   For matrices and N-D arrays, Y = NANMEDFILT1(X,N,[],DIM) or
%   Y = NANMEDFILT1(X,N,BLKSZ,DIM) operates along the dimension DIM.
%
%   See also MEDIAN, FILTER, SGOLAYFILT, and MEDFILT2 in the Image
```

```
%   Processing Toolbox and the NANSTATS tools.

%   Author(s): L. Shure and T. Krauss, 8-3-93
%   Copyright 1988-2002 The MathWorks, Inc.
%   $Revision: 1.8 $  $Date: 2002/03/28 17:28:51 $
%   ... updated to use nanmedian by G Clifford Mar 30th 2004

% Validate number of input arguments
error(nargchk(1,4,nargin));
if nargin < 2, n = []; end
if nargin < 3, blksz = []; end
if nargin < 4, DIM = []; end

% Check if the input arguments are valid
if isempty(n)
  n = 3;
end

if ~isempty(DIM) & DIM > ndims(x)
   error('Dimension specified exceeds the dimensions of X.')
end

% Reshape x into the right dimension.
if isempty(DIM)
   % Work along the first non-singleton dimension
   [x, nshifts] = shiftdim(x);
else
   % Put DIM in the first (row) dimension (this matches the order
   % that the built-in filter function uses)
   perm = [DIM,1:DIM-1,DIM+1:ndims(x)];
   x = permute(x,perm);
end

% Verify that the block size is valid.
siz = size(x);
if isempty(blksz),
   blksz = siz(1); % siz(1) is the number of rows of x (default)
```



```
else
   blksz = blksz(:);
end

% Initialize y with the correct dimension
y = zeros(siz);

% Call medfilt1D (vector)
for i = 1:prod(siz(2:end)),
   y(:,i) = medfilt1D(x(:,i),n,blksz);
end

% Convert y to the original shape of x
if isempty(DIM)
   y = shiftdim(y, -nshifts);
else
   y = ipermute(y,perm);
end

%-------------------------------------------------------------------
%                    Local Function
%-------------------------------------------------------------------
function y = medfilt1D(x,n,blksz)
%MEDFILT1D  One dimensional median filter.
%
% Inputs:
%   x     - vector
%   n     - order of the filter
%   blksz - block size

nx = length(x);
if rem(n,2)~=1    % n even
   m = n/2;
else
   m = (n-1)/2;
end
X = [zeros(m,1); x; zeros(m,1)];
```

```
y = zeros(nx,1);

% Work in chunks to save memory
indr = (0:n-1)';
indc = 1:nx;
for i=1:blksz:nx
   ind = indc(ones(1,n),i:min(i+blksz-1,nx)) + ...
         indr(:,ones(1,min(i+blksz-1,nx)-i+1));
   xx = reshape(X(ind),n,min(i+blksz-1,nx)-i+1);
   y(i:min(i+blksz-1,nx)) = nanmedian(xx);
end
```



References

Abdala O T, Clifford G, Saeed M, Reisner A, Moody G, Henry I, Mark R G (2004). "The Annotation Station: An Open-Source Technology for Annotating Large Biomedical Databases." *Computers in Cardiology*.

Abraham, R H and Shaw C D (1989). *Dynamics – The Geometry of Behavior, Part 1: Periodic Behavior*. Santa Cruz: Aerial Press Inc.

Akselrod S, Gordon D, Ubel F A, Shannon D C, Barger A C, Cohen R J (1981). "Power Spectrum Analysis of Heart Rate Fluctuation: A Quantitative Probe of Beat-to-Beat Cardiovascular Control." *Science*; 213:220-222.

Bendat, J S and Piersol, A G (1971). *Random Data: Analysis and Measurement Procedures*. New York: Wiley-Interscience.

Berne, R M and Levy, M N (2001). *Cardiovascular Physiology – Eighth Edition*. St Louis: Mosby.

Bianco, Carl (2004). *How Your Heart Works*. Available online: http//science.howstuffworks.com/heart.htm.

Castiglioni, P and Di Rienzo, M (2004). "How to check steady-state condition from cardiovascular time series." *Physiological Measurement*; 25:985-996.

Douglass M, Clifford G D, Reisner A, Moody G B, Mark R G (2004). "Computer-assisted de-identification of free text in the MIMIC II database." *Computers in Cardiology*.

Faucy AS, Braunwald E, Isselbacher KJ, Wilson JD, Martin JB, Kasper DL, Hauser SL, Longo DL (2001). *Harrison's Principles of Internal Medicine 14th Edition*. USA: McGraw-Hill Professional.

Gärtner B, Schönherr S (1998). "Smallest enclosing circles - an exact and generic implementation in C++." *Serie B - Informatik B 98-04*.

Goldstein B, Fiser D H, Kelly M M, et al (1998). "Decomplexification in critical illness and injury: Relationship between heart rate variability, severity of illness, and outcome. *Critical Care Medicine*; 26:352-357.

Golub, G H and Van Loan, C F (1996). *Matrix Computations*. Baltimore: Johns Hopkins University Press.

Jackson, E Atlee (1991). *Perspectives of Nonlinear Dynamics, Vol 1*. New York: Cambridge University Press.

Kantz, H and Schreiber, T (1997). *Nonlinear Time Series Analysis*. Cambrige,UK: Cambridge University Press.

Malik et al. (1996). "Standards of heart rate variability." *European Heart Journal*; 17:354-381.

Marino, Paul L (1998). *The ICU Book, Second Edition*. Baltimore: Lippincott Williams & Wilkins.

Martin, Lawrence (2004). *Pulmonary Physiology in Clinical Practice*. Avaliable online: http://www.mtsinai.org/pulmonary/books/physiology/.

Muller N, Magaia L, Herbst B M (2004). "Singular value decomposition, eigenfaces, and 3D reconstruction." *SIAM Review*; Vol 46 No 3:519-545.

"Mathworks signal processing toolbox documentation (2005)." Available Online: http://www.mathworks.com/access/helpdesk/help/toolbox/signal/.

Narayan, S M and Bhargava, V (2004). "Temporal and spatial phase analyses of the electrocardiogram stratify intra-atrial and intra-ventricular organization." *IEEE Transactions on Biomedical Engineering*; Vol 51, No 10.

Oppenheim A V, Schafer R W, Buck J R (1999). *Discrete-Time Signal Processing*. New Jersey: Prentice Hall.

Pomeranz B, Macaulay R J B, Caudill M A, Kutz I, Adam D, Gordon D, Kilborn K M, Barger A C, Shannon D C, Cohen R J, Benson H (1985). "Assessment of autonomic function in humans by heart rate spectral analysis." *American Journal of Physiology*; 248:H151-H153.

Saeed, M and Mark, R G (2000). "Multiparameter trend monitoring and intelligent displays using wavelet analysis." *Computers in Cardiology*; 27:797-800.

Saeed M, Lieu C, Raber G, Mark R G (2002). "MIMIC II: A massive temporal ICU patient database to support research in intelligent patient monitoring." *Computers in Cardiology*; 29:641−644.

Shi X, Huang G, Smith S A, Zhang R (2003). "Aging and Arterial Blood Pressure Variability during Orthostatic Challenge." *Gerontology*; 49:279-286.

Strang, Gilbert (1998). *Introduction to Linear Algebra, 2nd Edition*. Wellesley: Wellesley-Cambridge Press.

Shortliffe E H and Perreault L E (2001). *Medical Informatics: Computer Applications in Health Care and Biomedicine*. New York: Springer-Verlag.

Trefethen, L N and Bau, D. *Numerical Linear Algebra*. Philadelphia: SIAM, 1997.

Tsien, C L, Fackler J C (1997). "Poor prognosis for existing monitors in the intensive care unit." *Critical Care Medicine*; 25:614-619.

Yanowitz, Frank G (2005). "The Standard 12 Lead ECG." Available online: http://medstat.med.utah.edu/kw/ecg/ecg_outline/Lesson1/index.html.

Weinfurt, P T (1990). "Electrocardiographic monitoring: An overview." *Journal of Clinical Monitoring*, 6(2):132-138.

Winchell R J, Hoyt D B (1996). "Spectral analysis of heart rate variability in the ICU: A measure of autonomic function." *Journal of Surgical Research*; 63:11-16.

Wong, K H Hanson (2003). *Artifact Detection in Physiological Parameter Trend Data*. Master of Engineering in EECS Thesis. MIT.

Zimmerman M W, Povinelli R J, Johnson M T, and Ropella K M (2003). "A Reconstructed Phase Space Approach for Distinguishing Ischemic from Non-Ischemic ST Changes using Holter ECG Data." Computers in Cardiology; 30:243−246.

Zong W, Moody GB, Mark RG (2004). "Reduction of false arterial blood pressure alarms using signal quality assessment and relationships between the electrocardiogram and arterial blood pressure." *Medical and Biological Engineering and Computing*; 42:698-706.

Zong W, Heldt T, Moody GB, Mark RG (2003). "An open-source algorithm to detect onset of arterial blood pressure pulses." *Computers in Cardiology*; 30:259-262.

Zong W, Moody GB, Jiang D (2003). "A robust open-source algorithm to detect onset and duration of QRS complexes." *Computers in Cardiology*; 30:737-740.