7-15-2013

Use of support vector machines and fabry-perot interferometry to classify states of a laser

John Motley McKinnon

Use of Support Vector Machines and Fabry-Perot Interferometry to Classify States of a Laser

by

John McKinnon

Thesis

Submitted to the Department of Physics and Astronomy

Eastern Michigan University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Physics

Thesis Committee:

David Pawlowski, Ph.D, Chair

David Johnson, Ph.D. (Michigan Aerospace)

Ernest Behringer, Ph.D.

Marshall Thomsen, Ph.D.

July 15, 2013

Ypsilanti, Michigan

Dedication

I dedicate this document to Abigail Cosgriff. I do not know if I would have made it this far without her love and support.

Acknowledgements

I am deeply indebted to all of my family and friends, who have constantly shown their support to see me succeed. This journey would have been much more difficult without them.

I would like to thank Dr. David Johnson for giving me the invaluable opportunity to work as an intern at Michigan Aerospace and for his neverending patience and encouragement. He has acted as a mentor to me and is the reason that I have decided to pursue further studies in applied mathematics. I would also like to thank Dr. Matthew Lewis for helping me to understand support vector machines and David Kryskowski for sharing his knowledge in signal processing with me.

Abstract

This thesis develops an algorithm that can determine if a laser is functioning correctly over a long period of time. A Fourier fit is created to model fringe profiles from a Fabry-Perot interferometer, and singular value decomposition is used to reduce noise in each signal. Levenberg-Marquardt gradient descent is performed to correctly locate the center of each image and to optimize each fit with respect to the spatial frequency. The Fourier fit is used to extract important information from each image to be used for separating the image types from one another. Principal component analysis is used to reduce the dimensionality of the data set and to plot a projection of the data using its first two principal components. It is determined that the image data are not linearly separable and require a non-linear support vector network to complete the classification of each image type.

Table of Contents

List of Figures

Introduction

Fabry-Perot interferometers have wide-reaching applications. For example, we use them to measure the wavelengths of light and to study the fine structure of spectrum lines (Fowles, 1989). They can also can be used as optical resonators to select the mode of oscillation desired for a laser application (Saleh, 2007). Their main purpose in this study will be in reference to atmospheric lidar (light detection and ranging) applications.

Lidar systems are very accurate instruments that are used extensively in atmospheric studies (Wandiger, n.d.) such as for measuring wind velocity, temperature, pressure, and humidity, as well as the concentration of atmospheric gases and the visibility and height of clouds. Typically lidar systems have very demanding requirements on instrumentation including laser power, wavelength of light, and beam width. Because of these requirements lasers used in these experiments must exhibit stable and reliable behavior over long periods of time. Otherwise there can be significant inaccuracy in the lidar system.

When a laser is stable, it has a relatively low fluctuation in amplitude and frequency over a period of time. These fluctuations will occur naturally as a result of thermal changes in the laser chamber, vibrations (Melles Griot, n.d.), or possible mode beating. Mode beating is the process by which more than one laser mode is oscillating simultaneously, and can be caused by backscattered radiation. When the laser is not exhibiting stable behavior, then it may be experiencing problems with its reliability, which refers to its lifetime before it experiences failure.

Many methods exist to analyze stability for lasers. A standard method to measure frequency stability is to heterodyne the laser with a second laser with equal or greater

stability to itself (Melles Griot, n.d.). The method of heterodyning is to create a new frequency by mixing two frequencies. By examining the fluctuation of the beat frequencies of the system, the stability of the laser can be determined relative to the system since it must be no worse than the sum of the two frequency instabilities. To analyze the amplitude, the fluctuations of laser light can be monitored by a photodetector. Another useful way to determine frequency stability, as discussed by Hrabina et al., 2013, is to use a Fabry-Perot cavity as a frequency discriminator so that frequency fluctuations can be measured. Amplitude and noise fluctuations are typically measured as noise and are compared to a standard accepted value to determine their influence. For example, the cesium beam frequency standards are discussed in Lutwak, 2011. Methods such as these to analyze laser stability are useful in determining their reliability because the two concepts are closely related.

Laser failure occurs when there is a sudden shift in a laser parameter which is away from the operating requirements so that the laser does not function normally (Eliseev, 1991). More specifically, lasing will not occur when there is a failure of the laser diode. Diode failure can occur due to a vast number of reasons, many of which are very fast and readily apparent and others occur very slowly and are harder to identify. A description of these diode failure and degradation mechanisms is provided in Eliseev, 1991.

For example, according to Eliseev, the properties of diode crystals can be altered by extended exposure to radiation and can cause the formation of defects. Defects appearing as dark spots or bands can also form due to thermal or optical damage to the mirror faces of the diode resulting from optical or thermal overload. These overloads can be uncontrolled such as overloads caused by power instabilities from poor equipment or

leakage current. A controlled overload could be due to improper wiring or other varieties of operational mistakes. These forms of degradation of the laser diode result in fairly immediate reduction in laser power or laser characteristic, so they are easier to identify than longer term effects.

Natural aging of the laser diode can result in many different forms of degradation which are often difficult to identify because they occur slowly and can be mistaken for other forms of degradation (Eliseev, 1991). These are the forms of laser degradation that we are most concerned in for this study. This is because they have more direct relevance to the long term reliability for the laser. For instance, mechanical stress can occur due to heating, impact, or vibrations. Additionally, oxidation of diode facets can occur over a long period of time and because these oxide films are not uniform, the facet can become rough (Fakuda, 1991), causing lasing threshold to increase. These are only a few of the degradation mechanisms that occur due to age and wear on the diode. These degradation mechanisms and their effects on laser characteristics are shown by the production of self-sustained pulses, or self-modulation pulses. This causes a certain percent of laser pulses to result in broadening of the lineshape, and can disrupt the single mode operation of the laser (Eliseev, 1991).

Current methods to analyze laser reliability are based on the statistical prediction of laser failure. Failure rates are based upon long term product testing at many different temperatures. For example, Bao et al., 2010 discuss reliability analysis of high-performance 9xx nm single-emitter diodes. The article discusses how laser lifetimes are predicted based upon the behavior of an exponential reliability curve that depends on several parameters including the current, power, junction temperature, activation energy, and current and

power acceleration factors. Some of these parameters are determined based upon extrapolation of lifetime failure data for several different levels of current. Laser failures are described according to whether they are an early lifetime failure, wearout failure (long-lifetime failure), or a random failure (a failure that occurs in the intermediate stage of laser life).

Failures for the 9xx nm lasers investigated by Bao et al. were mainly due to catastrophic optical damage (COD) and COD resulting from bulk wear of the diode. Parameters of current, power, and so on that are required for predicting these failures were based upon limited data because the reliability analysis is so costly. In an attempt to find a more precise method of analyzing laser reliability which could possibly have accurate results based upon less data, this thesis investigates a different approach to the problem. It analyzes the stability and reliability of lasers by using techniques in pattern recognition and image classification to investigate the behavior of Fabry-Perot fringe patterns over a long period of time. By investigating patterns in the laser lineshapes, it could be worthwhile to investigate whether a certain form of laser degradation will correspond to a particular lineshape. For instance, could a phase-shifted Fabry-Perot pattern correspond to a lineshape that experienced broadening as the result of long-term laser degradation?

The etalon system used in the experiment was set up on an optical bench at Michigan Aerospace Corporation. In this investigation, we analyze fringe profiles produced by a Fabry-Perot air-gap etalon and a 266 nm neodymium-doped yttrium aluminum garnet (Nd:YAG) laser which are intended for future use in a lidar system. We wish to answer the following questions regarding the functionality and stability of the laser: Are there areas that the laser becomes phase-shifted? Are there dark images where the laser is not

producing output? While it is possible to visually sift through each profile by the naked eye in order to distinguish between these images, it becomes increasingly cumbersome and time consuming to do so for large data sets. The purpose of this paper is to lay the groundwork in developing the ability to identify Fabry-Perot fringe profiles by the process of machine learning. We are able to determine stability and reliability of the laser by identifying when and how frequently the laser does not produce good fringe profiles over a large data set.

Methodology

The theory of Fabry-Perot etalons, Fourier series, singular value decomposition, Levenberg-Marquardt gradient descent, the density based clustering algorithm entitled DBSCAN (Density Based Spatial Clustering of Applications with Noise), and support vector machines are discussed in Methodology.

To help the reader to visualize the algorithm which is developed, a flow chart is provided in Figure 1 to show a step by step procedure for performing the algorithm. The reader should refer back to this chart each time he begins reading a new section of material.

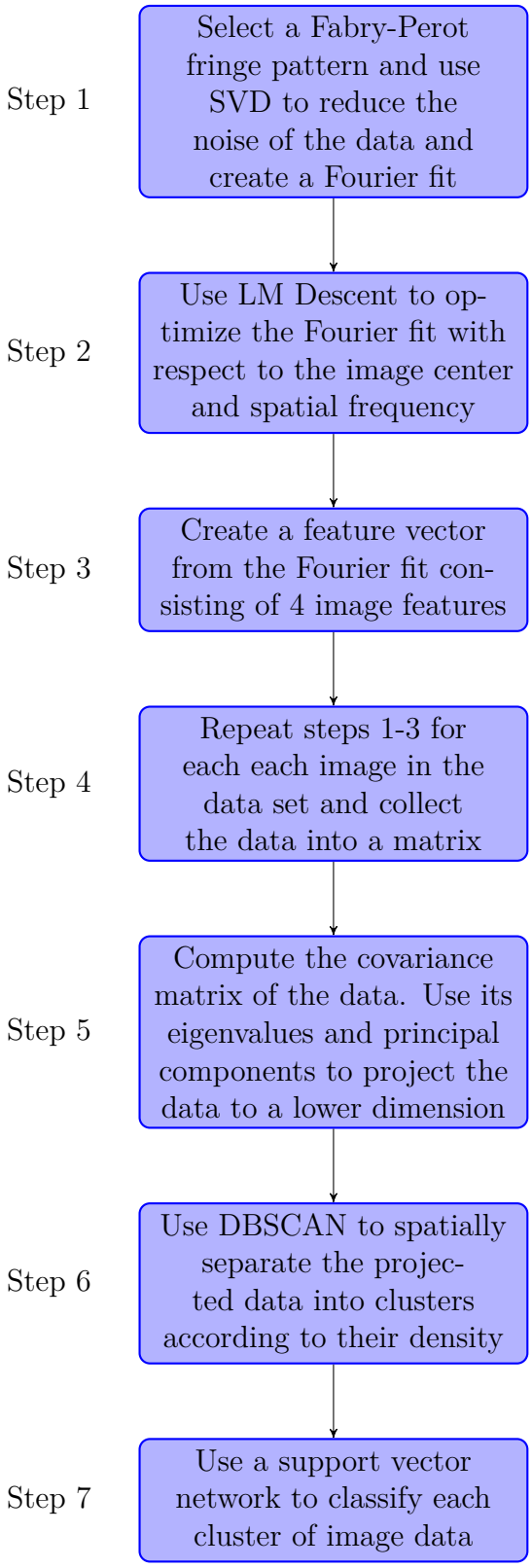

Figure 1. A flow chart showing how to execute the algorithm in a step-by-step procedure

The Fabry-Perot Interferometer

In this section we describe the Fabry-Perot intensity distribution and phase difference for multiple beam interference and provide a diagram to show the setup of a Fabry-Perot etalon. We format the distribution as an approximate function of ρ^2 where ρ is the radial distance away from the center of the image plane. This will allow Matlab to produce a periodic function.

Description of phase difference and intensity distribution. First, we derive the phase difference ϕ. Figure 2 shows a ray diagram for light incident on a Fabry-Perot etalon. Assume that the beam has been initially collimated by a lens, and then enters at an external angle θ_0 into an etalon consisting of two identical partially-reflecting mirrors, M_1 and M_2, each with reflectivity r separated by a distance d. The transmitted beam inside the etalon travels a path of length L at an angle θ_1, hits point A where it partially transmits through the etalon as T_1 before reflecting on a path to B, reflects to C, and then transmits through the etalon again as T_2. The phase difference between the two transmitted rays T_1 and T_2 is given by $\phi = \phi(ABC) - \phi(AD)$ where the focusing lens is located at point D.

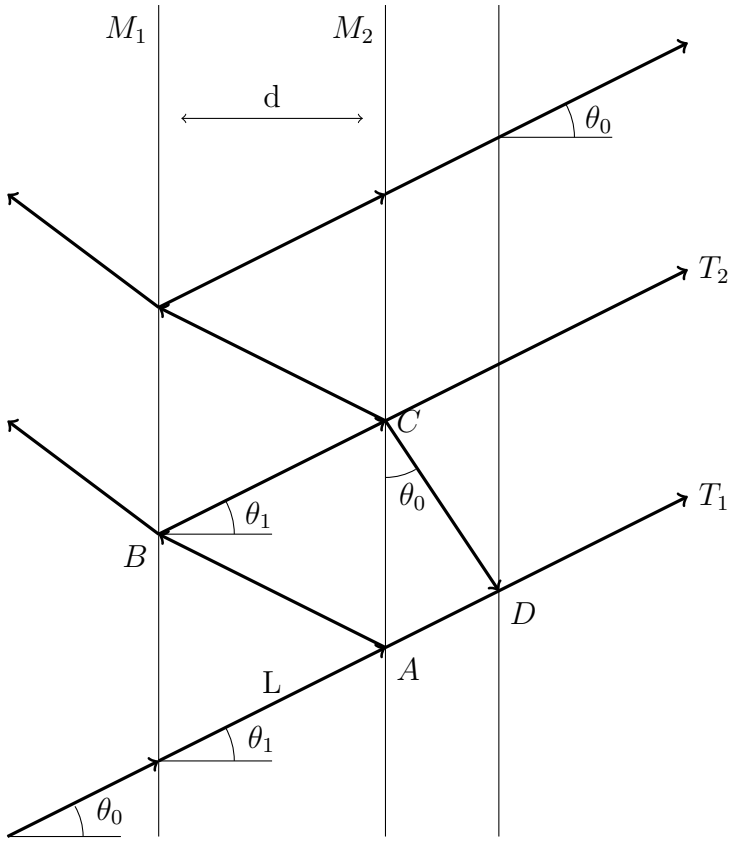

Figure 2. A diagram showing the setup of a Fabry-Perot etalon.

If the etalon has an index μ_1 and λ_0 is the laser central wavelength, then the phase difference can be derived from the geometry of Figure 2 as (refer to Appendix A):

$$\phi = \frac{4\pi\mu_1 d}{\lambda_0} cos(\theta_1). \tag{1}$$

We now define ρ as:

$$\rho(x, y) = \sqrt{(x - x_0)^2 + (y - y_0)^2}, \tag{2}$$

where the image center of each fringe profile is (x_0, y_0). The phase difference can be shown

to be an approximate function of ρ^2 (refer to Appendix A):

$$\phi(x,y) \approx \frac{4\pi\mu_1 d}{\lambda_0}(1 - \frac{\mu_0^2}{2\mu_1^2}\frac{\rho(x,y)^2}{f^2}), \tag{3}$$

assuming $f >> \rho$ where f is the focal length of the focusing lens. Now, we define the central order number M_0 to be:

$$M_0 = \frac{2\mu_1 d}{\lambda_0}. \tag{4}$$

Using $\mu_0 = 1$ for air, (3) can be written as:

$$\phi = 2\pi M_0(1 - \frac{\rho^2}{2\mu_1^2 f^2}), \tag{5}$$

so the phase becomes:

$$\phi(x,y) = \phi_0 - 2\pi\alpha[\rho(x,y)]^2, \tag{6}$$

where the spatial frequency α (the number of waves per square pixel) is defined as:

$$\alpha = \frac{M_0}{2\mu_1^2}\frac{(\Delta x_{pix})^2}{f^2}. \tag{7}$$

ϕ_0 is the phase at the minimum intensity which is given by:

$$\phi_0 = 2\pi M_0. \tag{8}$$

The factor Δx_{pix} is the pixel size in meters for the camera. This ensures that when we measure ρ in pixels, ϕ is dimensionless.

If we assume that the initial intensity of the incoming light is I_0, the final intensity is I_f, the transmittance is T and the reflectance is R, then I_f can be written as an approximate function of ρ^2 (refer to Appendix A) as follows:

$$I_T(\rho^2) \approx I_0 \frac{T^2}{1 - 2Rcos(\phi_0 - 2\pi\alpha\rho^2) + R^2}, \tag{9}$$

which is the Fabry-Perot intensity distribution.

Identification of Image Features

Each image type in the data set is described and a discussion is provided on how to select a suitable set of features that can be used distinguish between the three possible image types: good images, phase-shifted images, and dark images.

If the laser is functioning correctly, it will continually produce images that have a central fringe that is aligned in the center of the image plane. These are referred to as good images. A shifted image will correspond to a fringe pattern that has a central fringe that is not aligned with the center of the image plane, and thus its fringes are phase shifted in comparison to those of a good image. A Fourier series in complex form is a simple and effective way to keep track of the phase ϕ. Thus, in order to determine how much a given image is phase shifted away from a good image, an effective strategy is to determine how similar the Fourier coefficients of the selected image are to a good image which is used as reference. If the coefficients for the given image are not similar to the reference image, then the image has likely been phase shifted.

The fringe profile for a Fabry-Perot etalon is characterized by its *lineshape function* (Saleh et al., 2007), which is a plot of the intensity of the laser as a function of the phase (or frequency). The lineshape depends on several parameters, including its linewidth and peak value. The linewidth is usually defined as the full width of the lineshape function at half of its maximum value, or FWHM. If the laser is exhibiting stable behavior, then the fringe profile will have a stable linewidth and peak value. If we observe variation in the linewidth and peak value of the lineshape function in each image over time, it suggests that the laser is unstable. Thus, the FWHM and peak values of the lineshape function are two

important features that should be computed for each image in the data set.

The final image type that we wish to identify are dark images. These are images where the laser is simply not producing any output and we do not observe a fringe profile at all. If we plot the lineshape profiles for these dark images, it can be seen that they have very little variation in their peak and minimum values, which suggests that a key image feature that can be used to identify this image type is the minimum value of intensity (and also the maximum, which has already been discussed as an important feature).

For each image, our goal is to construct a feature vector that consists of the four features that have been described: the minimum and maximum intensity, the measure of similarity in the Fourier coefficients, and the FWHM, and to record all of the feature vectors into a data matrix.

Fourier Projection and Least Squares

In this section we describe how a Fourier series can be used to calculate the similarity between Fourier coefficients for two given images. This important image feature will help us to distinguish between each image type in the data set. The method to solve for the Fourier coefficients is formulated in terms of the method of least squares. We describe how singular value decomposition can be used to reduce the noise of each Fourier fit. In the final section we show how principal component analysis (PCA) is able to create a projection of our data set so that it can be visualized.

Complex form of Fourier series - phase shifting and similarity. We recall that a function $f(x)$ of period 1 can be expanded in a Fourier series with complex coefficients given by $c_n = a_n + ib_n$ in the following way:

$$f(x) \approx \sum_{n=-N}^{N} c_n exp(i2\pi nx). \tag{10}$$

If we wish to compare the phase of a series that has been shifted τ units away from that of the original series, then we find a simple relation between the set of Fourier coefficients $\{c_n\} = \{c_{-N}, ...c_N\}$ and the set of the shifted series $\{d_n\} = \{d_{-N}, ...d_N\}$. If we let our shifted series be $g(x)$, then we have:

$$g(x) = f(x+\tau) = \sum_{n=-N}^{N} d_n exp(i2\pi nx) = \sum_{n=-N}^{N} c_n exp(i2\pi n(x+\tau)) \tag{11}$$

$$= \sum_{n=-N}^{N} c_n exp(i2\pi n\tau) exp(i2\pi nx) \tag{12}$$

and we see that the relationship between the coefficients is given by:

$$d_n = c_n exp(i2\pi n\tau). \tag{13}$$

This inspires the following strategy to be employed on the set of image data. Take a good image and expand it in a Fourier series with the set of coefficients $\{c_n\}$. Then take an image that has been phase shifted and expand it as a new Fourier series with the set of coefficients $\{d_n\}$. If the image of the laser is showing a normal stable evolution over time, then the Fourier coefficients of the new image should have a phase shift very similar to that of the good image. We see that if each coefficient is equal so that $c_n = d_n$, then $\tau = 0$, so there is no phase shift. If there is a large phase shift, then we will see differences in magnitude and phase when comparing c_n and d_n.

While it is likely possible to numerically estimate the values of τ and compare them for each image in order to determine those which have become phase shifted from the good image, this thesis does not take that approach to the problem. We use an idea based on the definition of the inner product between two vectors to determine how similar the shifted images are to a good image. We first recall that the angle θ between two vectors u and v is given by:

$$cos(\theta) = \frac{< u, v >}{\|u\|\|v\|}. \tag{14}$$

We will now define the variable $S = cos(\theta)$ to be the similarity between the vectors of

Fourier coefficients given by $c = (c_{-N}, ...c_N)$ and $d = (d_{-N}, ..., d_N)$ so we will have:

$$S = \frac{< c, d >}{\|c\|\|d\|} = \frac{c^* d}{\|c\|\|d\|}, \tag{15}$$

where * is the conjugate transpose. We also note that we need $0 \leq S \leq 1$ because c and d are complex. By defining S in this way, we see that a value of $S = 0$ corresponds to a situation where c and d are orthogonal to each other, so the two images are entirely dissimilar and so there may be a substantial phase shift. Similarly, a value of $S = 1$ corresponds to the situation that was discussed earlier when $c_n = d_n$ and the images are identical, so there will be zero phase shift.

The Fabry-Perot profile using the method of least squares. The method of solving for the Fourier coefficients is equivalent to solving the linear system:

$$Ax = b, \tag{16}$$

where A is an $m\,x\,n$ matrix consisting of m observations and N Fourier terms corresponding to $n = 2N + 1$ coefficients, x is an $n\,x\,1$ vector of coefficients in the expansion, and b is an $m\,x\,1$ data vector; in our case it is the ρ^2 values from the Fabry-Perot intensity profile. In the Fourier sense, we can write the matrix equation as:

$$Ax = g, \tag{17}$$

and we can write the m^{th} row of the equation as a function of ρ^2 in the following format:

$$g_m = \sum_{n=-N}^{N} c_n exp(i2\pi n\alpha\rho_m^2), \tag{18}$$

where α is the spatial frequency that we found in (7). We can simplify this in terms of (6) and write the equation as:

$$g_m = \sum_{n=-N}^{N} c_n exp(in\phi_m^2). \tag{19}$$

A term in our Fourier matrix A in row m and column n is given by:

$$A_{mn} = exp(in\phi_m^2) \quad n = -N \cdots N. \tag{20}$$

Since A is a matrix with $m > n$, it is overdetermined and we will solve for the coefficients of best fit by the method of least squares, which seeks to minimize the sum of the squares of the errors (SSE) where we can define "error" as being:

$$E = \|Ax - b\|. \tag{21}$$

It can be shown (refer to Appendix B) that the least squares estimate that minimizes the SSE is given by:

$$\hat{x} = (A^*A)^{-1}A^*b. \tag{22}$$

Unfortunately, finding the Fourier fit in this manner is not very helpful if the data which we wish to use is very noisy, which is the case with our images. However, the language that we use in least squares is useful for solving (16) when we discuss the process

of singular value decomposition in our next section. It will also be useful to us when we discuss optimization of the Fourier fit in section 4.3.

Singular value decomposition. Computing each image feature from the Fabry-Perot lineshape function cannot be accurately done without first reducing the noise in each image profile. Singular value decomposition (SVD) is used to determine the Fourier coefficients required to produce a Fourier fit for each image as in (16) and is used to reduce noise of each fit by computing the pseudoinverse of the Fourier basis matrix.

A reader who is not familiar with techniques in linear algebra, or would like a brief review of the key concepts needed to understand SVD, should refer to Appendix C before reading further. We first state two very important properties of symmetric matrices (Schlens, 2009) which we will need to understand the construction of the SVD. The first property is that if A is a symmetric matrix, then it has orthogonal eigenvectors. If λ_1 and λ_2 are distinct eigenvalues of the symmetric matrix A with corresponding eigenvectors e_1 and e_2, then we can use the eigenvalue equation (Appendix C) to show that (Schlens, 2009) $(\lambda_1 e_1) \cdot e_2 = (\lambda_2 e_1) \cdot e_2$ and hence $(\lambda_1 - \lambda_2) e_1 \cdot e_2 = 0$. Since $\lambda_1 \neq \lambda_2$ then it must be true that $e_1 \cdot e_2 = 0$, so the eigenvectors are orthogonal. We can choose the eigenvectors to be normalized so that A has an orthonormal basis. Next, we note that if A is a symmetric matrix, then it can be *orthogonally diagonalized*. This means that A can be written in the form:

$$A = EDE^T, \tag{23}$$

where E is the matrix of eigenvectors of A so that each eigenvector e_i is a column of E, and D is a diagonal matrix where $D_{ii} = \lambda_i$ so that each entry is an eigenvalue of A. The

proof of this fact as discussed by Schlens is provided in Appendix C.

We can now describe the SVD for a matrix. If A is an $m\,x\,n$ matrix, then we see that A^TA and AA^T are both symmetric matrices because $(A^TA)^T = A^TA$ and similarly $(AA^T)^T = AA^T$. Thus, we can diagonalize the matrices such that:

$$A^TA = VDV^T, \quad AA^T = UDU^T. \tag{24}$$

Because both of these conditions are true no matter what the original matrix A was (it can be shown that the eigenvalues of A^TA are the same as the eigenvalues of AA^T), then it must also be true that there is a decomposition for the original matrix A which satisfies them simultaneously. Trial and error shows us that the choice of A which satisfies the conditions (24) is given by:

$$A = USV^T, \tag{25}$$

where $S^2 = D$. Equation (25) is the SVD for the matrix A. If A is an $m\,x\,n$ matrix, then it can be written as the product of an $m\,x\,m$ orthonormal matrix U, an $m\,x\,n$ diagonal matrix S, and the transpose of a second $n\,x\,n$ orthonormal matrix V. The columns of the matrix V consist of the orthonormal eigenvectors of the symmetric matrix A^TA and the columns of the matrix U consist of the orthonormal eigenvectors of the (also symmetric) matrix AA^T. The first r diagonal entries of S are called the singular values and are arranged in decreasing order as $s_1 > s_2 > ...s_r$ where $s_i = \sqrt{\lambda_i}$ and where λ_i is the eigenvalue of either A^TA or AA^T. The value r is the rank of A (the number of linearly independent columns of A). If it turns out that $r < m$ then we have to add in an extra $(m - r)$ orthonormal vectors to the matrix U, and similarly if $r < n$ we have to add in an

extra $(n-r)$ orthonormal vectors to the matrix V. This is because we need to make sure that both U and V are truly orthonormal. We now show a computation to verify the claim that $\lambda_i = s_i^2$ is an eigenvalue of A^TA. We start with:

$$A^TA = VSU^TUSV^T = VS^2V^T \tag{26}$$

and then multiplying by V we get:

$$(A^TA)V = VS^2, \tag{27}$$

so if v_i is a column vector of V then we have:

$$(A^TA)v_i = s_i^2 v_i \tag{28}$$

and thus we have verified that $\lambda_i = s_i^2$ is an eigenvalue of A^TA with eigenvector v_i.

It is important to note that although one possible way to compute the SVD of a matrix is by finding the eigenvectors and eigenvalues of A^TA or AA^T, this is typically not done in practice for larger matrices (Sauer, 2006). This is because substantial loss in accuracy occurs as the matrix increasingly approximates a singular matrix (one which is not invertible). In general, special techniques are often used to form new matrices out of the matrix A in order to compute the SVD. In order to explain the application of SVD to reduce noise of a signal, we provide the SVD for a simple matrix. For this example, there is no harm in simply computing A^TA and AA^T to find their eigenvectors. The SVD of the

matrix $\begin{bmatrix} 6 & 8 \\ 8 & 6 \end{bmatrix}$ is given as:

$$\begin{aligned} \begin{bmatrix} 6 & 8 \\ 8 & 6 \end{bmatrix} &= \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 14 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \\ &= \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \left(\begin{bmatrix} 14 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \right) \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \\ &= \begin{bmatrix} 7 & 7 \\ 7 & 7 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}. \end{aligned} \tag{29}$$

We see that we can write our original matrix as the sum of two other matrices, one for each singular value, and that the matrix corresponding to the larger singular value accounts for a larger portion of the data in our original matrix than the matrix corresponding to the smaller singular value. Large singular values will account for the portion of the data with a large amount of variance and the small singular values with account for the portion of the data which has very small variance, which we can categorize as being noise. Thus, if we compute the SVD of a data matrix and throw away singular values which are small in magnitude and keep the values with large magnitudes, we will have filtered the data in a sense and removed a substantial portion of the noise. There is a name for this kind of procedure. If we have a matrix A with rank r, and we choose to keep $p \leq r$ singular values, then we have created the best rank p approximation to A in the least squares sense. Ideally, we would like to perform SVD on our Fourier matrix A (with

$m > n$), and then just solve the least squares system (16) as:

$$\begin{aligned} (A^TA)^{-1} =& (VSU^TUSV^T)^{-1} \\ =& (VS^2V^T)^{-1} = VS^{-2}V^T \\ x = (A^TA)^{-1}A^Tb =& VS^{-2}V^TVSU^Tb \\ x =& (VS^{-1}U^T)b, \end{aligned} \tag{30}$$

but unfortunately, the matrix S is not square! We can however use *Reduced* SVD to compute what is called the *pseudoinverse* of A, which is denoted as $A^\dagger$. The pseudoinverse is defined to be (Clark, 2007):

$$A^\dagger = VS^\dagger U^T, \tag{31}$$

where V is an $n\,x\,n$ matrix, $S^\dagger$ is $n\,x\,n$, and U^T is $n\,x\,m$ so that $A^\dagger$ is then $n\,x\,m$. The definition for U in reduced SVD is slightly different than before, so that now we only include the first n columns from the U as defined in (25). The diagonals entries of $S^\dagger$ are given by $s_i^\dagger = 1/s_i$. Also, we throw away the last $(m-n)$ rows from S in order to ensure that the matrix $S^\dagger$ is an $n\,x\,n$.

Thus, the least squares solution to (16) using SVD on the Fourier matrix A, is that the Fourier coefficients x of the filtered signal will be given by:

$$x = A^\dagger b = (VS^\dagger U^T)b, \tag{32}$$

where we remember that b is our data from the fringe profile.

The method of using SVD to create a fit for each fringe profile relies on the

assumption that we are capable of correctly estimating the center of each image prior to producing the fit. One possible method to make a fairly good estimate for the center (x_0, y_0) of a given image is to use an image tool that produces individual pixel values. Unfortunately, we have to manually perform this procedure on an individual image in order to produce the correct center. We can still identify which images appear to have been phase shifted. If the center of a shifted image is incorrect (and we just used the center for the good image) then its Fourier fit will also be incorrect, so the image features that we observe will be highly different from that of the good one. The main disadvantage (a fairly large one) to using this approach is that we are not guaranteed that images which appear to be phase shifted are actually representing bad images. It is possible that they were good images, but were produced by the camera drifting off center instead of by a problem with the laser. If we are able to estimate the correct center for each image arbitrarily, and the resulting fringe profiles are substantially different from a good image that we use for reference, then we will have much more substantial evidence to support our belief that we have identified images that have truly been phase-shifted. This estimation can be performed with Levenberg-Marquardt gradient descent, and is the topic of the next chapter.

Principal component analysis. We now describe some of the basic ideas of principal component analysis (PCA) and explain its relevance to our problem of image classification. Principal component analysis provides a method for taking a data set and representing it in a new set of orthogonal basis vectors that point in the directions of maximal variance of the data (Shlens, 2009). This is the method that we use to reduce the dimensionality of our data set so that it can be visualized. The description of PCA that is presented here is very similar to the tutorial on performing PCA by Schlens, 2009.

PCA provides a simple and fairly efficient method of taking a problem that is seemingly very complex with observations that depend on a large number of parameters, and then greatly reduces the dimensionality and redundancy of the data while eliminating noise in the process. To accomplish this goal, PCA provides a reinterpretation of the SVD for how we can project our data to a lower dimensional space.

The general strategy in PCA is to take a set of data that has been expressed in some basis, and then reexpress it in a new basis which is a linear combination of the original basis. To make these ideas more clear, we start with a data matrix X constructed from m observations and n features. In order to perform PCA, it is often assumed (Schlens, 2009) that each column of X has zero mean to ensure that each feature has the same weight. In the case that the n features of X are measured in different units, then we may choose to instead divide by the standard deviation. We now let P be an $n\,x\,n$ linear transformation which will project our data set as the new $m\,x\,n$ representation Y, so that (Schlens, 2009):

$$XP = Y. \tag{33}$$

We define p_i to be the columns of P and x_i to be the rows of X. Thus, the columns of P will represent a new basis for expressing the rows of X. We can see this as:

$$XP = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} p_1 & p_2 & \cdots & p_n \end{bmatrix}$$

$$Y = \begin{bmatrix} x_1 \cdot p_1 & \cdots & x_1 \cdot p_n \\ \vdots & \ddots & \vdots \\ x_m \cdot p_1 & \cdots & x_m \cdot p_n \end{bmatrix}. \tag{34}$$

The row vectors y_i of Y are given by:

$$y_i = \begin{bmatrix} x_i \cdot p_1 & \cdots & x_i \cdot p_n \end{bmatrix}. \tag{35}$$

The row vectors x_i of the original data matrix X are expressed in terms of the columns of P. The new representation for a row vector y_i is found by projecting the i^{th} row of x_i onto the basis $\{p_1, p_2, \cdots, p_n\}$.

This new basis should be the *clearest* representation of the data set so that it can be most easily understood by the experimenter. A way of selecting the new basis is to require that it produce data with the highest *variance* (Schlens, 2009). The covariance matrix of the data C_X is defined to be the $n\,x\,n$ matrix:

$$C_X = \frac{1}{n-1} X^T X. \tag{36}$$

If we divide the matrix X by the variance so that it is dimensionless, then it is referred to as a correlation matrix. We see that the ij^{th} entry of the matrix is found by taking the dot product of the i^{th} data type with the j^{th} data type. The variance in each direction is represented by each diagonal term, and each covariance term is represented by the off-diagonal. The terms of variance which are large represent the important information in the signal, and the small values correspond to noise. The off-diagonal terms represent information which is redundant. These terms represent areas where the information from one data type can be used to predict the information from another. Thus, to perform PCA, we should define a procedure that will *diagonalize* the covariance matrix of the data since we want to minimize redundancy (covariance) and maximize the signal (variance) (Schlens, 2009).

In order to accomplish this, we wish to project the data onto the basis for which the variance is maximized. Thus, we will choose each normalized basis vector p_i to be in a direction where the variance of X is maximized, and such that each basis vector is perpendicular to the previous ones. Each column of P which is chosen in this way is referred to as a *principal component*. Our assumption that each principal component is orthonormal is important, because it allows PCA to be performed by the use of SVD.

To make our goal more specific, we wish to find the orthonormal matrix P with $XP = Y$ such that covariance of the projected data C_Y with:

$$C_Y = \frac{1}{n-1} Y^T Y \tag{37}$$

becomes diagonalized. A fact that is verified by Schlens (refer to Appendix D) is that

selecting the matrix P so that each column is an eigenvector of X^TX will ensure that:

$$C_Y = \frac{1}{n-1}D, \tag{38}$$

so that our goal is met and C_Y is diagonalized. Thus, the principal components p_i of X are orthonormal eigenvectors of X^TX, and are the columns of P. How does this model fit SVD? The SVD of the matrix X is written as:

$$X = USV^T, \tag{39}$$

where the columns of V are the orthonormal eigenvectors of X^TX. We can multiply X by V to write this as:

$$XV = Y, \tag{40}$$

where $Y = US$ is the projection of the data. This is the exact model that was described in (33) with $P = V$. Iif we compute the SVD of $\frac{1}{\sqrt{n-1}}X$, then the columns of V are the eigenvectors of C_X, so they are the principal components of X (Schlens, 2009). The matrix combination US represents the *best* representation of X, which was found by projecting it onto V. How does this let us reduce dimensionality and noise in the data set? We compute the $n\,x\,n$ matrix V and then use the first k columns with $k < n$, which correspond to the largest k principal components so that V is then an $n\,x\,k$. Our projection Y will then be an $m\,x\,k$ data matrix which retains the data with the largest variation, but has a lower dimensional representation than X.

To decide how many principal components we should keep, we plot the magnitudes of

the individual eigenvalues of C_X as a fraction of the total sum of eigenvalues. This will tell us how much variance is contained in each principal component. This is the method that we will use to reduce the dimensionality of our image data, so that it can be plotted and then classified by support vector machines.

Optimization of the Fourier Fit

We now describe how to optimize the Fourier fit to each image in the data set using gradient descent techniques. First we describe a basic gradient descent technique, and then describe the more advanced Levenberg-Marquardt gradient descent. It is by using this method that we can estimate each image center arbitrarily. In addition, the algorithm optimizes each Fourier fit with respect to the spatial frequency α as defined in (7), given that we begin with a starting guess. The presentation of each descent method follows closely with ideas presented by Ranganathan, 2004.

Levenberg-Marquardt gradient descent. Levenberg-Marquardt gradient descent (LM algorithm) is one of the most widely used algorithms in optimization theory. Specifically, it provides a fast solution to the Non-Linear Least Squares Minimization Problem (Ranganathan, 2004). We wish to minimize the quantity:

$$E(x) = \sum_{j=1}^{m} \frac{1}{2} \|R_j(x)\|^2, \tag{41}$$

where $x = (x_1, x_2, x_3, ..., x_n)$ is a vector of n parameters, $E(x)$ is called an error function, and each $R_j(x)$ is a non-linear function of x called a residual. This can be rewritten in terms of a residual vector $R(x)$ of length m so that:

$$E(x) = \frac{1}{2} \|R(x)\|^2 \tag{42}$$

and $R(x) = (R_1(x), R_2(x), ..., R_m(x))$. We now use the derivatives of E to define the $m\,x\,n$ Jacobian matrix J where:

$$J = \frac{\partial R_j}{\partial x_i} \;\; 1 \leq j \leq m, 1 \leq i \leq n. \tag{43}$$

Now, we will describe the simple gradient approach to solve the problem. First, take an initial guess $x = x_0$ for the value for x which $\nabla E(x) = 0$. We will use the following update rule (Ranganathan, 2004):

$$x_{i+1} = x_i - \gamma \nabla E(x_i). \tag{44}$$

Thus we will take the current value of x and subtract from it the gradient of E which is scaled by a factor of γ where we have:

$$\nabla E(x) = \sum_{j=1}^{m} R_j(x) \nabla R_j(x) = J(x)^* R(x). \tag{45}$$

The factor of γ is set to be a value so that $E(x_{i+1}) < E(x_i)$, and so the error function $E(x)$ decreases at every step, and gets closer to its minimum value. Since $\nabla E(x)$ points in the direction of maximum increase, we see that $-\nabla E(x)$ will point in the direction of maximum decrease of $E(x)$. This method of gradient descent is fairly slow and can miss the value of the minimum, so it will not always converge. Using this rule, the stepsize used to update our parameter x is proportional to the present size of the gradient. This means that where the gradient is large and the function has a steep slope we end up taking a large step, and where the gradient is small and the function has a shallow valley we end up

taking a small step. In reality, we would like to do the exact opposite procedure in order to converge quickly. Thus, simple gradient descent is slow because it forces us to take a small stepsize in order to avoid rattling out of the minimum (Ranganathan, 2004).

This method can be greatly improved by using the curvature of the function in addition to the gradient. First, we describe the Gauss-Newton update. Then we describe the Levenberg update and finally the Levenberg-Marquardt update. The latter is the most useful for our problem.

Newton's method is used in order to approximate the solution to the equation $\nabla E(x) = 0$ (since this is where our minimum occurs). If $x = x_0$ is the current value of our parameter vector, then we can use Taylor's formula with a center x_0 to expand $\nabla E(x)$ as follows:

$$\nabla E(x) = \nabla E(x_0) + (x - x_0)^* \nabla^2 E(x_0) + \textit{higher order terms}. \tag{46}$$

If we assume that E is *quadratic* around the value x_0, then the minimum of the function will be where $\nabla E(x) = 0$ and can be found by solving the equation (Ranganathan, 2004):

$$\nabla E(x_0) + (x - x_0)^* \nabla^2 E(x_0) = 0. \tag{47}$$

If we set $x_0 = x_i$ and $x = x_{i+1}$ then the update rule in Newton's method will be given by (Ranganathan, 2004):

$$x_{i+1} = x_i - (\nabla^2 E(x_i))^{-1} \nabla E(x_i). \tag{48}$$

Using the definition in (43) we see that:

$$\nabla^2 E(x) = J(x)^* J(x) + \sum_{j=1}^{m} R_j(x) \nabla^2 R_j(x). \tag{49}$$

A common approximation is to assume that for the function E the R_i's are nearly linear around the solution, which means that the quantity $\nabla^2 R_j(x)$ is small. Thus, the *Hessian* matrix $H = \nabla^2 E(x)$ (the matrix of second partial derivatives) is approximately given by:

$$H = J(x)^* J(x). \tag{50}$$

The Newton update (actually, it is now referred to as the Gauss-Newton update) can be written as (Ranganathan, 2004):

$$x_{i+1} = x_i - H^{-1} \nabla E(x_i), \tag{51}$$

where H must be evaluated at x_i.

The advantage of using this update is that it will converge to the minimum much faster than simple gradient descent. Unfortunately, it relies on the assumption that E around the starting guess x_0 is fairly linear in $R(x)$, which may or not be the case. Thus, the tradeoff we have is that convergence is sensitive to our choice of x_0. This is exactly the opposite situation that we saw with simple gradient descent.

It is beneficial to combine simple gradient descent and the Gauss-Newton update into a new rule which we call the Levenberg update. When we are far away from the minimum where E is not very quadratic around x_0, then we will use simple gradient descent. When

we get closer to the minimum, then our quadratic assumption should work better and we will implement the Gauss-Newton update. We will define this new update rule as follows (Ranganathan, 2004):

$$x_{i+1} = x_i - (H + \gamma I)^{-1} \nabla E(x_i), \tag{52}$$

where I is the identity matrix. So the smaller the value of γ the more that this update will look like (51), while large values of γ will make the update rule much closer to:

$$x_{i+1} = x_i - \gamma^{-1} \nabla E(x_i), \tag{53}$$

which is just gradient descent. The Levenberg algorithm can be described as follows (Ranganathan, 2004):

1. Perform the update rule in (52) to find the new parameter vector x_{i+1}.
2. Calculate the error at the new parameter vector $E(x_{i+1})$.
3. If $E(x_{i+1}) - E(x_i) < 0$ so the error decreases, it means our quadratic assumption is correct so we accept the update in (52). We *decrease* γ (generally by a factor of abound 10) to reduce the amount that we rely on on the gradient.
4. If $E(x_{i+1}) - E(x_i) > 0$ so the error increases, then we undo the update and reset the parameter vector to its previous value. We *increase* γ (by a factor of about 10) so that we increase our use of the gradient and then we try the update again.

While this new update rule is much better than both simple gradient descent and the Gauss-Newton algorithm, it still has the disadvantage that using large values of γ results in

us ignoring the Hessian, so the curvature provides us with no additional information. We would like an update rule that should take large steps in directions of high curvature and small steps in directions of low curvature. The Levenberg-Marquardt algorithm will do exactly this and uses an update rule defined by:

$$x_{i+1} = x_i - (H + \gamma diag[H])^{-1} \nabla E(x_i), \tag{54}$$

so the identity matrix in (52) is replaced by the diagonal of H, which scales each component of the gradient according to the curvature. Since the Hessian is proportional to the curvature, we see that where the curvature is large the stepsize $(H + \gamma diag[H])^{-1}$ becomes small, and similarly where the curvature is small the stepsize is large (Ranganathan, 2004).

The Levenberg-Marquardt algorithm (LM algorithm) works very well to optimize a system which has a moderate number of parameters. When the system begins to have on the order of a few thousand parameters the method becomes too difficult to implement in practice because it requires us to use matrix inversion and requires too many operations to be efficient (Ranganathan, 2004).

Implementation of the LM algorithm with image data. We now put the LM algorithm in the context of our image data. The algorithm works by taking an initial estimate for the image center (x_0, y_0) and spatial frequency α, and then iteratively arrives at values for (x_0, y_0) and α which *minimize* the sum of the squares of the errors (SSE) between the Fourier fit that was created using SVD and the image data. The error function $E(x)$ that we wish to minimize will be:

$$E(\alpha, x_0, y_0) = \frac{1}{2}\|b - g(\alpha, x_0, y_0))\|^2, \tag{55}$$

where we have:

$$g(\alpha, x_0, y_0) = \sum_{j=-N}^{N} c_j exp(in\phi(\alpha, x_0, y_0)) \tag{56}$$

and

$$\phi(\alpha, x_0, y_0) = 2\pi\alpha\rho^2 = 2\pi\alpha[(x - x_0)^2 + (y - y_0)^2]. \tag{57}$$

The residual vector which has length m is defined as:

$$R(\alpha, x_0, y_0) = b - g(\alpha, x_0, y_0). \tag{58}$$

The gradient of the error function is found to be:

$$\nabla E = \langle b - g, -\nabla g\rangle = (-\nabla g)^*(b - g), \tag{59}$$

where the Jacobian matrix that we need is given by:

$$\nabla g = \begin{pmatrix} \frac{\partial g}{\partial \alpha} & \frac{\partial g}{\partial x_0} & \frac{\partial g}{\partial y_0} \end{pmatrix} = \begin{pmatrix} \frac{\partial g}{\partial \phi}\frac{\partial \phi}{\partial \alpha} & \frac{\partial g}{\partial \phi}\frac{\partial \phi}{\partial x_0} & \frac{\partial g}{\partial \phi}\frac{\partial \phi}{\partial y_0} \end{pmatrix}. \tag{60}$$

Each of the derivatives that we must compute are as follows:

$$\frac{\partial g}{\partial \phi} = in \sum_{j=-N}^{N} a_j exp(in\phi(\alpha, x_0, y_0)) \tag{61}$$

$$\frac{\partial \phi}{\partial \alpha} = 2\pi[(x - x_0)^2 + (y - y_0)^2] \tag{62}$$

$$\frac{\partial \phi}{\partial x_0} = -4\pi\alpha(x - x_0) \tag{63}$$

$$\frac{\partial \phi}{\partial y_0} = -4\pi\alpha(y - y_0), \tag{64}$$

where the Hessian is found using (50):

$$H = (\nabla g)^* \nabla g. \tag{65}$$

In order to compute the LM update as defined in (54), the Jacobian and Hessian must be evaluated at the current value of the parameter vector $x = (\alpha, x_0, y_0)$.

Thus, for each image we use LM Gradient Descent to create a Fourier fit with minimal noise which is produced using accurate values for its image center and spatial frequency. Each Fourier fit is used to compute a vector consisting of four features. We then collect the feature information for the entire set of images into a matrix so that PCA can be performed on it.

Density Based Clustering (DBSCAN)

Clustering algorithms are extremely useful for *class identification*, which is the process of taking a set of data and then dividing it into a number of subclasses. DBSCAN (Density Based Spatial Clustering of Applications with Noise) is a clustering algorithm which takes a data set and separates it into subclasses according to how spatially dense they are (Ester et al, 1996). DBSCAN has been accepted as a very effective method for discovering data clusters since it can identify clusters which have arbitrary shape, will work on databases with several thousand objects, and it will significantly outperform other clustering algorithms, such as CLARANS (Clustering Large Applications Based on Randomized Search) (Ester et al, 1996). While we do not execute DBSCAN to obtain our results, we discuss its implications for our results and refer to it in our conclusions when discussing future research.

The basics of DBSCAN are discussed here, using definitions as presented by Ester et al, 1996. The main idea behind DBSCAN is to observe that data clusters can be identified by the fact that inside of a data cluster there is a substantially higher density of data points than there is outside of the cluster, and that places of noise in the data are identified by an area of lower density. Every data point of a cluster should have a minimum number of points within a neighborhood around it. Thus, if D is our database, we define the $\epsilon - neighborhood$ of a point p $N_\epsilon(p)$ to be (Ester et al, 1996):

$$N_\epsilon(p) = \{q \in D \mid d(p,q) \leq \epsilon\}, \tag{66}$$

where $d(p,q)$ is the normal Euclidean distance between two points. There are two kinds of

data points of a data cluster. The points which are inside of the cluster are called core points, and the points are on the outside of the cluster are called border points. A core point p is easy to identify because it should satisfy the condition that (Ester et al, 1996) $|N_\epsilon(p)| \geq MinPts$ where $MinPts$ is a threshold number of points that we choose. This is sometimes, but not necessarily, true for a border point. To identify all points of a data cluster, we need several new definitions.

First, we say that a point p is *directly density-reachable* from a point q with respect to the parameters ϵ and $MinPts$ if p is contained in $N_\epsilon(q)$ and $|N_\epsilon(q)| \geq MinPts$ (Ester et al, 1996).

This relation between two points is not necessarily symmetric. Two core points are always directly density-reachable from each other, but if a border point p is directly density-reachable from q, then q is not necessarily directly density-reachable from p.

We say that a point p is *density-reachable* from a point q with respect to ϵ and $MinPts$ if there is a chain of points $p_1, \cdots, p_n$ where $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i (Ester et al, 1996).

This definition is a simple extension of direct density reachability. It will allow us to state the final definition that we need before we define a cluster and how to find one.

We say that a point p is *density-connected* to a point q with respect to ϵ and $MinPts$ if there is a point o such that both p and q are density-reachable from o with respect to ϵ and $MinPts$ then q is in C (Ester et al, 1996).

Thus, we see that even though two border points in the same cluster are not density-reachable from each other that they must be density connected. We now state the definition of a cluster.

If D is a database of points, then a cluster C with respect to ϵ and $MinPts$ is a non-empty subset of D which satisfies the following conditions (Ester et al, 1996):

1. For every p and q in C, p is density-connected to q with respect to ϵ and $MinPts$.

2. For every p and q, if p is in C and q is density-reachable from p with respect to ϵ and $MinPts$, then q is in C.

We can then define noise as being the collection of points which do not belong to any cluster of D. The algorithm DBSCAN identifies a cluster by first choosing an arbitrary core point from the data set, and then returning the set of data points which are density-reachable from that core point. In order to run, it first requires a suitable choice for ϵ and $MinPts$. Details for executing the algorithm and the process for determining ϵ and $MinPts$ are both included in Ester et al, 1996.

Support Vector Machines

Support vector machines (SVMs) are a widely used tool in pattern recognition. They have been used with high accuracy to classify data in biological sequence analysis (Rätsch, n.d.) as well as in handwriting (Cortes et al., 1995) and face detection in images (Osuna et al., 1997). This section is not meant to be a tutorial on how to use SVMs, but instead to give a brief introduction to SVMs and to summarize some of their underlying theory. In this section we describe the fundamental concepts of SVMs and state a formal definition of the linear SVM. Note that the notation and much of the discussion presented here is very similar to Ben-Hur et al., 2010. Some excellent step by step tutorials on how to correctly implement SVMs can be found in Hsu et al., 2010 and Ben-Hur et al., 2010. A much more extensive guide to the theory of SVMs can be found in Burges, 1998.

Support vector machines are two-class classifiers which allow us to take a set of data and separate it into two categories. For instance, one class of data might be a set of images that are good Fabry-Perot fringe patterns and another class might be phase shifted patterns. We say that the i^{th} observation vector x_i from a data set with n observations will be given a label y_i where $y_i \in \{+1, -1\}$, so that we can classify each object as being either "positive" for one class or "negative" for the other (Burges, 1998).

SVMs, whether they are linear or non-linear rely on a separating hypersurface in order to correctly classify observation points. For linear SVMs, these hypersurfaces are simply hyperplanes. In order to decide which class an object will belong to, we need to know which side of the hyperplane the observation lies on. If we are able to find a suitable linear decision boundary that is able to separate the data into positive and negative

classes, then we say that the data is linearly separable. In order to define what our linear decision boundary is, we will introduce the concept of a discriminant function $f(x)$ such that (Ben-Hur et al., 2010):

$$f(x) = w^T x - \gamma, \tag{67}$$

where we call w the weight vector, and γ is the bias which determines how far from the origin the hyperplane is translated. Our decision boundary is where $f(x) = 0$, so we have (Ben-Hur et al., 2010):

$$w^T x - \gamma = 0. \tag{68}$$

In the training process the parameters w and γ are generated and used to create the decision boundary. This process allows the SVM to read in part of a given data set in order to recognize the characteristics of each class. The method to compute these parameters is discussed later in this section after more definitions have been stated. In the classification process, the sign of f(x) is computed for data that the SVM has not encountered. We say that a label y_i is given to a data point x_i according to which side of the decision boundary it is on (Burges, 1998):

$$w^T x_i - \gamma \geq +1 \;\; if \;\; y_i = +1 \tag{69}$$

$$w^T x_i - \gamma \leq -1 \;\; if \;\; y_i = -1. \tag{70}$$

This set of inequalities can be combined to give (Burges, 1998):

$$y_i(w^T x_i - \gamma) \geq 1 \;\; i = 1, ..., n. \tag{71}$$

The support vectors are the data points which are the closest to the decision boundary. The geometric margin m, is defined as the distance from a support vector to the decision boundary. If we let x_+ and x_- be support vectors that are equidistant from the decision boundary and on the positive and negative sides respectively, then if $\hat{w}$ is a unit vector in the direction of w, we can define the margin to be as follows (Ben-Hur et al., 2010):

$$m = \frac{1}{2}\hat{w}^T(x_+ - x_-). \tag{72}$$

If we further require $f(x_+) = 1$ and $f(x_-) = -1$, then we obtain two equations that can be subtracted to yield (Ben-Hur et al., 2010):

$$w^T(x_+ - x_-) = 2 \tag{73}$$

Then dividing by $2\|w\|$ we obtain a different way to express the margin (Ben-Hur et al., 2010):

$$m = \frac{1}{2}\hat{w}^T(x_+ - x_-) = \frac{1}{\|w\|}. \tag{74}$$

Figure 3 below shows a linearly separable data set along with the separating surface, the support vectors, and the geometric margin.

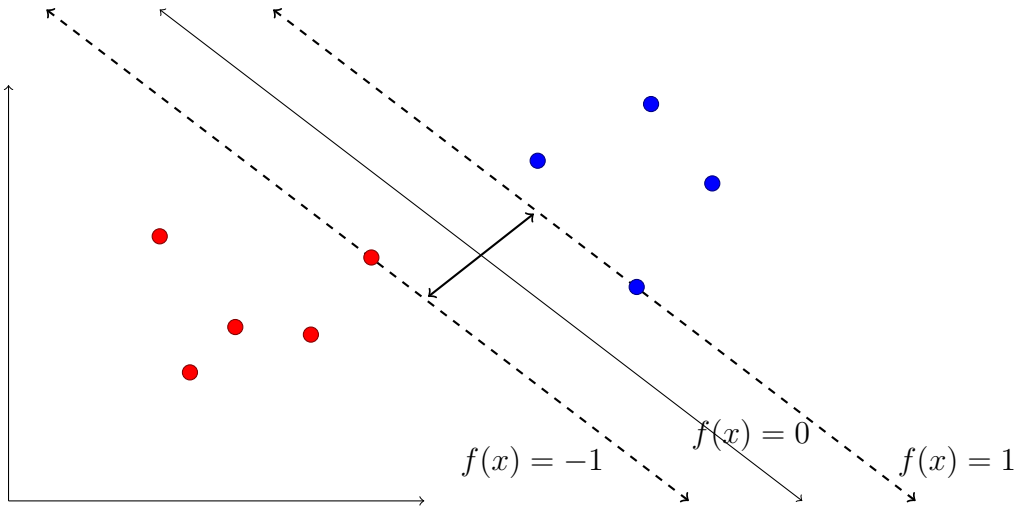

Figure 3. A linearly separable data set. The line $f(x) = 0$ represents the separating surface. The support vectors are located along $f(x) = 1$ and $f(x) = -1$. The black line drawn between the support vectors represents twice the distance of the geometric margin.

We are now in a position to more precisely define support vector machines. A support vector machine is a discriminant function that maximizes the geometric margin. As such, we also refer to a SVM as being a maximum margin classifier. We will talk about the hard-margin SVM first, which is a maximum margin classifier that assumes our data is linearly separable. In order to maximize the geometric margin $m = 1/\|w\|$ we will minimize $\frac{1}{2}\|w\|^2$, which is an equivalent task. Note that the factor of $1/2$ appears purely for convenience. Thus, in creating a hard margin SVM we solve the following optimization problem (Ben-Hur et al., 2010):

$$\begin{aligned} &\underset{w,\gamma,\xi}{\text{minimize}} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to:} \quad y_i(w^T x_i - \gamma) \geq 1 \;\; i = 1, ..., n. \end{aligned} \tag{75}$$

A large number of data sets are linearly separable and therefore can be classified with 100 percent accuracy, but many are not. In the case that our data sets are not linearly

separable, we can still have a fair degree of accuracy by allowing the classifier to have room for error. This new classifier which allows for misclassified data points is referred to as a soft-margin SVM, which we will now discuss.

We introduce positive slack variables ξ_i which correspond to each data point x_i and alter our previous constraints to read (Ben-Hur et al., 2010):

$$y_i(w^T x_i - \gamma) \geq 1 - \xi_i \;\; i = 1, ..., n. \tag{76}$$

These new constraints allow data points that lie within the geometric margin to have $0 \leq \xi_i \leq 1$ and data points which are misclassified (and lie on the opposite side of the margin than we would hope) to have $\xi_i > 1$. In the spirit of reducing the number of misclassified points, we now seek to solve the following optimization problem (Ben-Hur et al., 2010):

$$\begin{aligned} \underset{w,\gamma,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + \nu \sum_{i=0}^{n} \xi_i \\ \text{subject to:} \quad & y_i(w^T x_i - \gamma) \geq 1 - \xi_i \quad \xi_i \geq 0. \end{aligned} \tag{77}$$

This is the soft-margin SVM. Our minimization condition now provides a term $\sum_{i=0}^{n} \xi_i$ which represents an upper bound on the number of misclassified data points. The factor of ν in (77) allows us to choose how much we wish to penalize ourselves for misclassifying points, and serves as a regularization parameter (Ben-Hur et al., 2010).

In order to implement the SVM algorithm, we use the approach as described by Mangasarian et al., 2001. A more optimal version of (77), where the constraint equation is written with matrices to make interpretation in Matlab easier (Mangasarian et al., 2001):

$$\begin{aligned} \underset{w,\gamma,\xi}{\text{minimize}} \quad & \frac{1}{2}(\|w\|^2 + \gamma^2) + \nu\frac{\|\xi\|^2}{2} \\ \text{subject to:} \quad & D(Aw - e\gamma) \geq e - \xi. \end{aligned} \tag{78}$$

The added factor of γ^2 in the quantity to be minimized allows for the margin between the separating planes to be optimized with respect to both the orientation w as well as the location γ (instead of only with respect to w) (Mangasarian et al., 1999). The term involving ξ was reformulated to $\|\xi\|^2$ in order to make the constraint $\xi \geq 0$ redundant. The diagonal matrix D is the $m\,x\,m$ matrix where m is the number of observations to be classified and has either either $D_{ii} = +1$ or $D_{ii} = -1$ depending on the membership of the observation point A_i (Mangasarian et al, 2001). The constant 1 is now written as the column vector e consisting of all ones.

Because this is a convex quadratic programming problem (Boyd et al., 2004), the minimization problem (78) is reformulated in terms of the single variable u, which is the much simpler *dual* of the problem, and is ideal for computation. The *dual* is (Mangasarian, 1994):

$$\underset{u\geq 0}{\text{minimize}} \quad \frac{1}{2}u'(\frac{I}{\nu} + D(AA' + ee')D)u - e'u., \tag{79}$$

where the values for w and γ that define the separating plane are found from the relations (Mangasarian et al., 2001):

$$w = A'Du, \quad \xi = \frac{u}{\nu}, \quad \gamma = -e'Du. \tag{80}$$

To simplify notation, the following matrices are introduced (Mangasarian et al., 2001):

$$H = D[A - e], \quad Q = \frac{I}{\nu} + HH', \tag{81}$$

so that (79) is written as:

$$\underset{u \geq 0}{\text{minimize}} \quad \frac{1}{2}u'Qu - e'u. \tag{82}$$

By using the Karush-Kuhn-Tucker (KKT) necessary and sufficient optimality conditions for (82) (Mangasarian, 1994):

$$u \geq 0 \quad and \quad u \perp (Qu - e) \geq 0 \tag{83}$$

the solution to the minimization problem is then provided by the iterative scheme (Mangasarian et al., 2001):

$$u^{i+1} = Q^{-1}(e + ((Qu^i - e) - \alpha u^i)) \quad \alpha \geq 0, \quad i = 0, 1, ... \tag{84}$$

and is referred to as the Lagrangian support vector machine algorithm (LSVM algorithm). Values for the parameters ν and α are chosen as (Mangasarian, 1994) $\nu = 2.0, \alpha = 1.9/\nu$ in order to ensure convergence. The inverse of the matrix Q is computed by a special case of the Sherman-Morrison-Woodbury (SMW) identity (Golub et al., 1996):

$$(\frac{I}{\nu} + HH')^{-1} = \nu(I - H(\frac{I}{\nu} + H'H)^{-1}H'). \tag{85}$$

As a final note, we state that material regarding non-linear SVMs is not needed for

understanding the rest of this paper, but provides useful background information. As such, topics regarding non-linear SVMs are discussed in Appendix E.

Results and Discussion

The results consist of four sections. First, specific examples of the three image types are shown and the etalon system is described. In the next section two Fourier fits are produced to show how the LM algorithm is able to optimize the fit by correctly guessing the image center and spatial frequency for each image. We provide two additional plots that give valuable information about the stability of the laser by showing the drift of the image center and spatial frequency over time. We then create a feature vector for each image and collect the information into a data matrix. PCA is used to compute the principal components of the data set, and the two components with the largest variance are used to create an easily visualized projection. A linear support vector network is created to show how to classify a simple set of three Gaussian clusters. Because the projection of our data is not Gaussian, the linear support vector network will not classify our data set without misclassification.

Discussion of Image Data

In this section we very briefly describe the image data. This project uses images taken from a pulsed, diode-pumped, 266 nm Nd:YAG laser sent through an air gap etalon. The etalon uses two fused silica mirrors each with index ≈ 1.49 that are separated by a distance of roughly d $\approx$ 6.7mm. The beam is focused onto a CCD image sensor with a pixel size of 16 microns and an image plane that is $512\,x\,512$ pixels. The data set consists of a total of 6780 images which are taken over a period of several hours. Our goal is to separate the images into the three classes that we described in the introduction: images in which the laser produces a "good" pattern so that the fringes are centered in the middle of

the image plane; images which appear to be phase-shifted; and dark images in which the laser does not produce output. Examples of Fabry-Perot fringe patterns for each of these three image types are shown below in Figures 4, 5, and 6 respectively. Although it can be difficult to see a difference between the good image in Figure 4 and the image in Figure 5, if the reader investigates closely, it appears that the entire fringe pattern in Figure 5 has been phase shifted downward by several pixels in comparison to Figure 4. In Figure 6 we see that the image plane of the CCD is completely dark so it does not display any fringes at all. This indicates that the laser is not producing output.

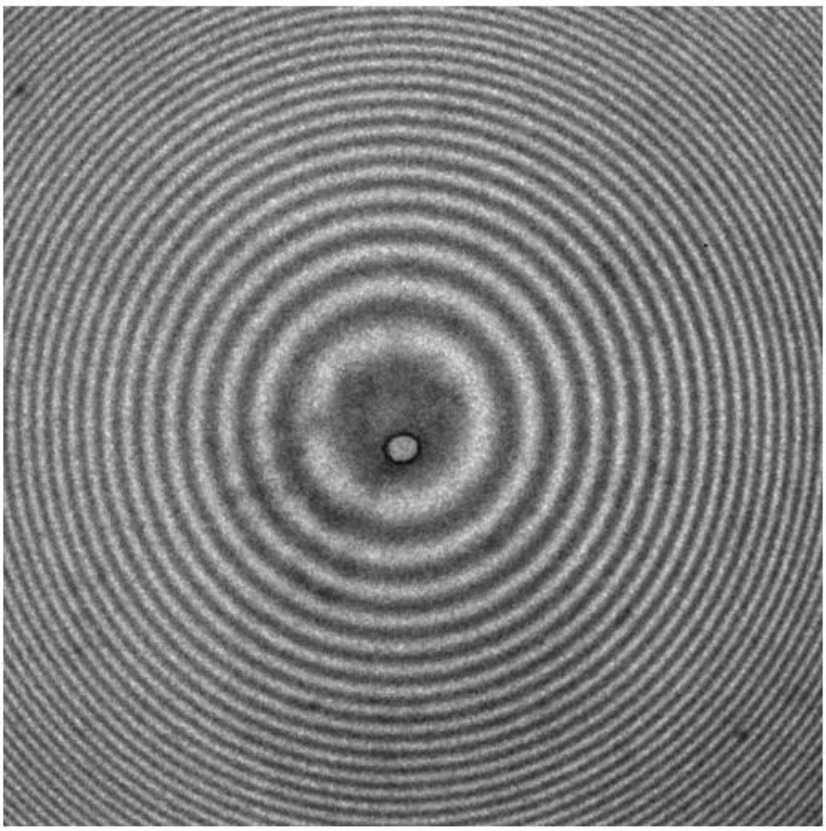

Figure 4. A fringe pattern that is aligned to the center of the CCD frame.

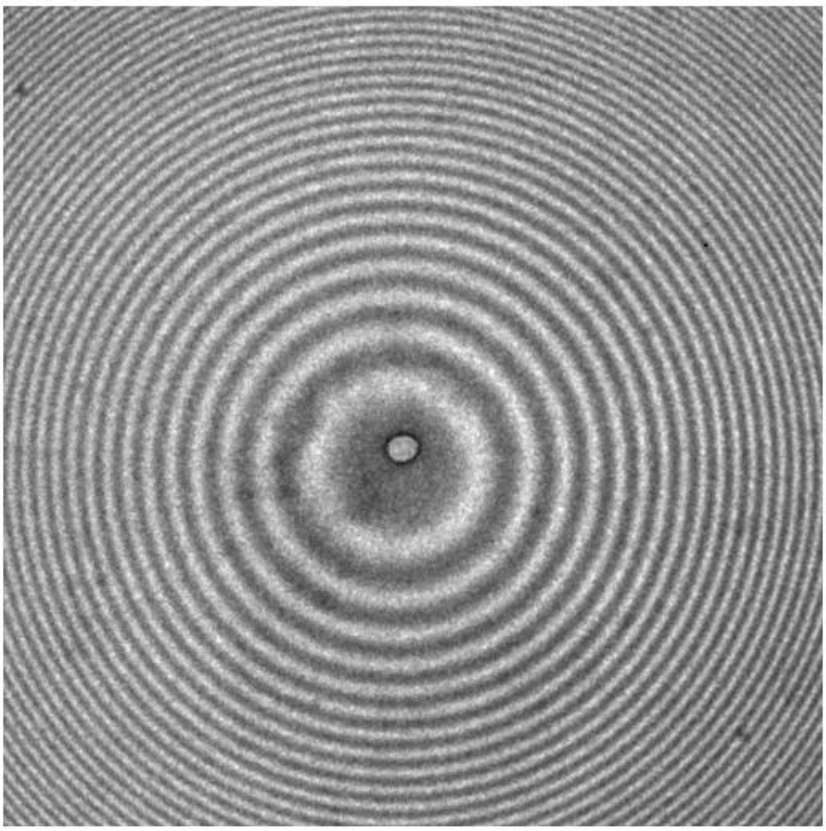

Figure 5. A shifted fringe pattern. This image appears shifted downward by several pixels in comparison to our good image.

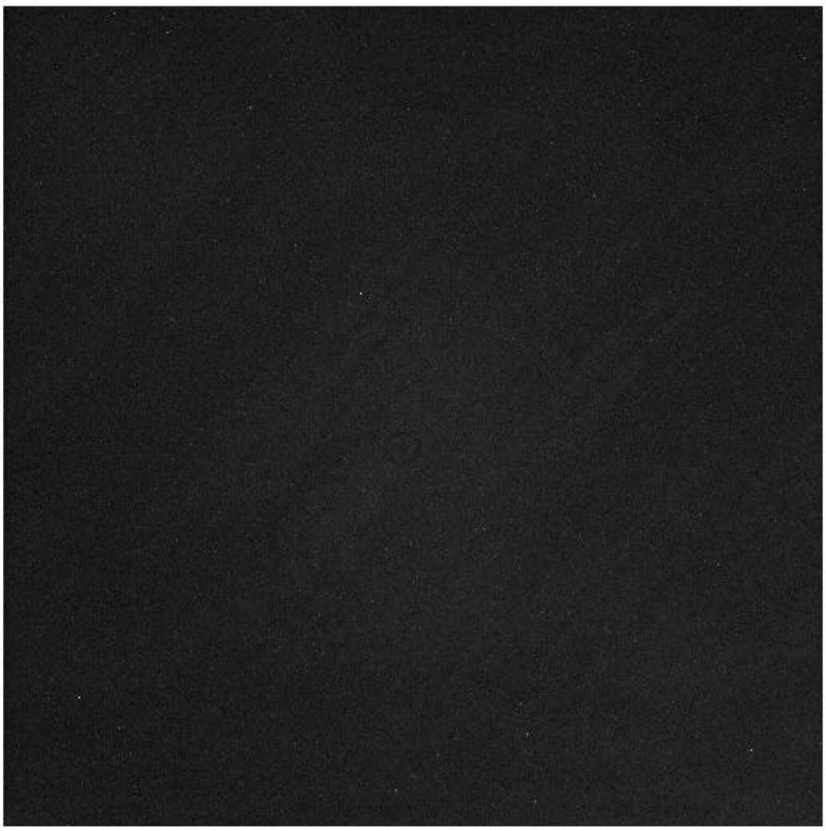

Figure 6. A dark image.

Optimization of the Fourier Fit Using Levenberg-Marquardt Algorithm

In this section, an example is shown of how the LM algorithm is able to use an image and an initial guess for the image center to produce the optimal image center and Fourier fit. We begin with an initial estimate for the center (x_0,y_0) which is somewhat arbitrary, but is chosen in an attempt to minimize deviation away from the true center for each image in the set in order to ensure convergence of the method. The initial estimate for the spatial frequency, α, is based upon the spacing of the peaks in the intensity profile. This is done by simply counting a set number of fringes that we have moved away from the image center and dividing by an estimate for the number of square pixels it took to get there. In the process of producing the Fourier fit, we also perform singular value decomposition (SVD) on the data in order to reduce the noise of our signal.

If we use the Matlab image processing tool "impixelregion" on the image in Figure 5, we might use the values $x_0 = 246.0$ and $y_0 = 287.0$ as an approximation to the correct center. However, when we use a poor initial guess $x_0 = 250.0$, $y_0 = 265.0$, we see that an incorrect center produces a very poor fringe profile. We do not see a fringe pattern that travels smoothly between a distinct minimum and maximum intensity, we simply see an incoherent pattern of data. The following fit in Figure 7 is produced between the first and second fringe orders, so that the phase is plotted over $2\pi \leq \phi \leq 4\pi$ where ϕ is calculated using (6). The blue points in the plot represent the total intensity of the light collected by the CCD. The width of this cluster of points shows that the data has considerable noise, thus making it difficult to choose individual data points for a particular intensity. The red line is the Fourier fit that is produced by performing SVD on the matrix A of Fourier basis

vectors, where A is defined in (20) and where the Fourier coefficients are found by solving (32). This fit is a reconstruction of the data that has significantly less noise than before.

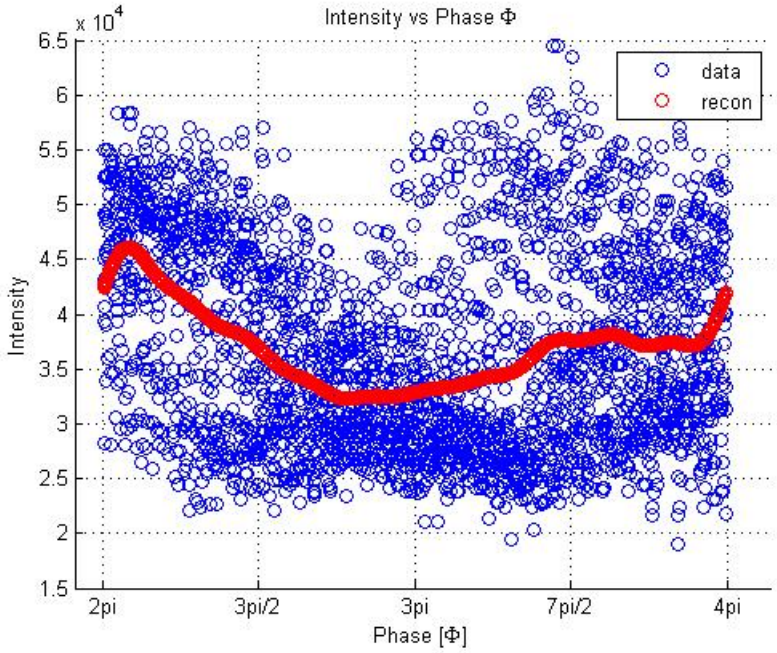

Figure 7. A Fourier fit that uses an incorrect guess for the image center.

When we perform the LM algorithm using the update rule as defined in (54), we produce the much more accurate values $x_0 = 245.9$ $y_0 = 287.8$ for the center of the image, which is comparable to our initial guess using "impixelregion". In addition, when using an initial estimation of $\alpha = 2.17 \times 10^{-4}$, we produce an optimal value for the spatial frequency $\alpha = 3.17 \times 10^{-4}$. For this particular image, the sum of squares of the errors for the poor fit was found to be $E = 1.94 \times 10^{11}$, but after using the optimal values this sum has been reduced to $E = 1.72 \times 10^{10}$, a factor of more than 10. Thus, we see the LM algorithm allows us to correctly guess the center and spatial frequency for each image in an arbitrary fashion. These values are used to produce the *best* Fourier fit in the sense that the sum of the squares of the errors has been minimized. The optimized fit is shown below in Figure 6.

Because it uses an accurate estimate for the image center, a smooth fringe profile is produced.

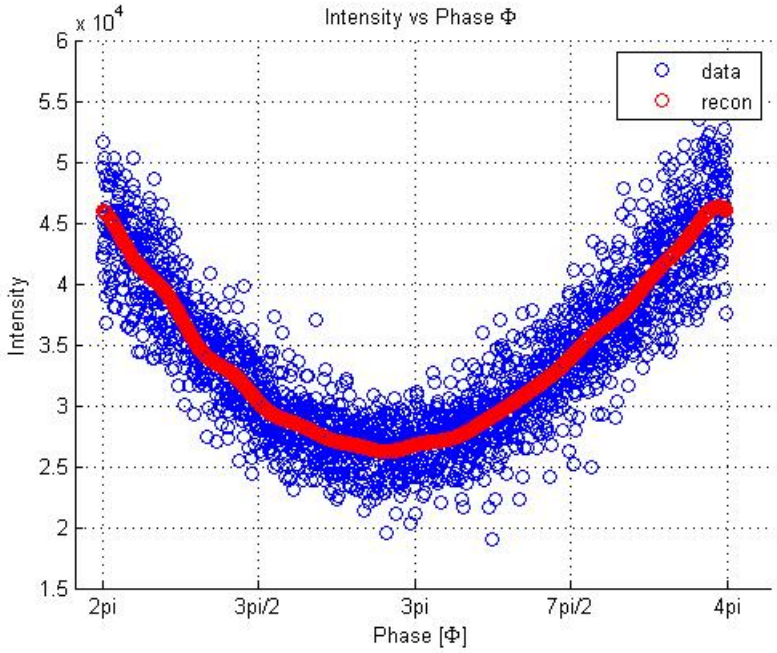

Figure 8. A Fourier fit that uses an optimized guess for the image center.

The LM algorithm also allows two very useful plots to be produced which provide insight into the stability of the laser. Figure 9 shows the evolution of the image center over time. The x and y locations of the image center (in pixels) are plotted against the total number of images which have elapsed.

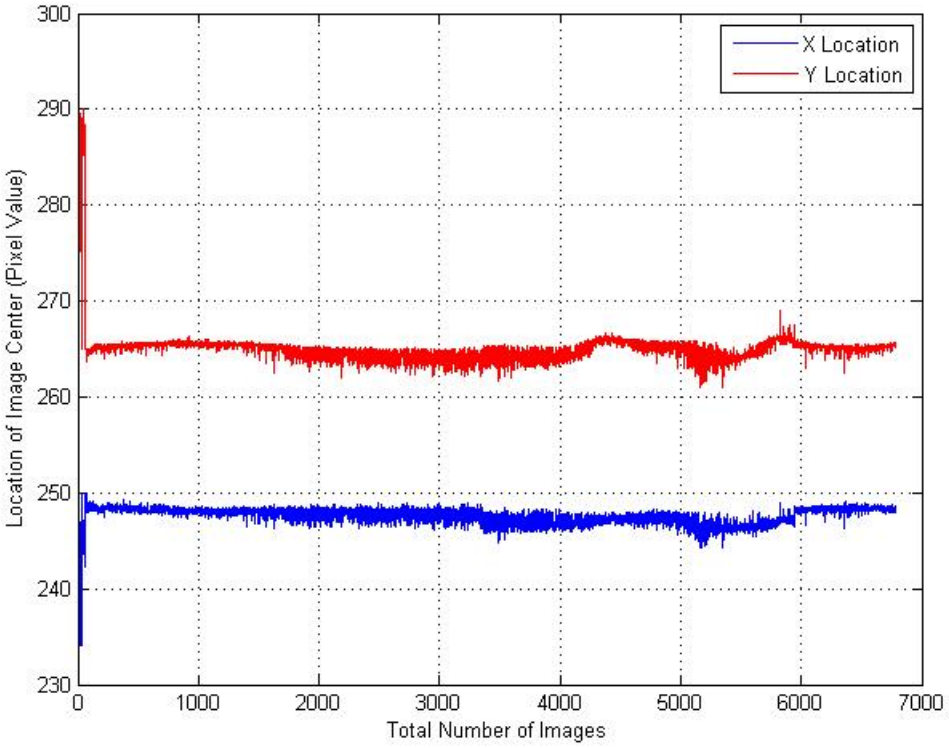

Figure 9. A plot of the x and y locations of the image center as a function of total time elapsed.

Figure 9 indicates that while there is substantial drift in the image center in the first hundred images or so, there is not much drift over long periods of time. Most of this observed drift is likely due to thermal noise. Figure 10 shows the evolution of the spatial frequency over time.

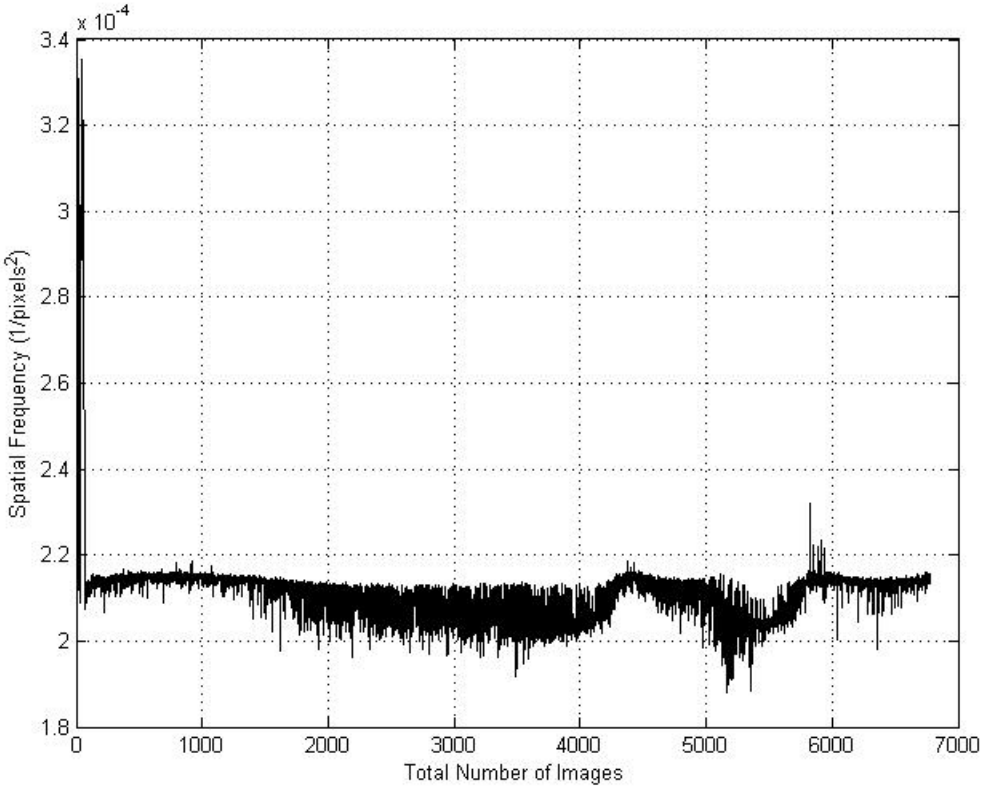

Figure 10. A plot of the spatial frequency α as a function of total time elapsed.

Figure 10 shows that there is substantial change in the spatial frequency over long periods of time indicating that the fringes are expanding and contracting fairly rapidly and thus becoming unstable.

Creation of The Feature Vector and Dimensionality Reduction with PCA

For each image, we construct an optimized Fourier fit using the LM algorithm . The DC term of each fit is set to zero prior to calculating each image feature in order to avoid scaling problems with the correlation matrix used in PCA. Each feature vector is given by $x = [max(f), min(f), S, \delta]$ where $max(f)$ and $min(f)$ are the maximum and minimum values of the Fourier fit, S is the measure of similarity as described in (15), and δ is the FWHM of each fit. We use each vector to form a $6780\,x\,4$ matrix consisting of all of the feature information in our data set, so that each feature vector is a row. We then divide each column by its variance to ensure that each column of observations is dimensionless. We form the correlation matrix C_X of the data matrix X as described in (36) and compute its eigenvectors, which are the principal components.

We now create a plot which shows us the amount of variance in the data that is present in each eigenvalue. Figure 11 is produced by plotting the amount of variance in the data against the sum of the first k eigenvalues for $k \leq 4$. As we increase k, the total amount of variance that we account for increases. Note that the eigenvalues are plotted in ascending order of magnitude. Our plot is shown below:

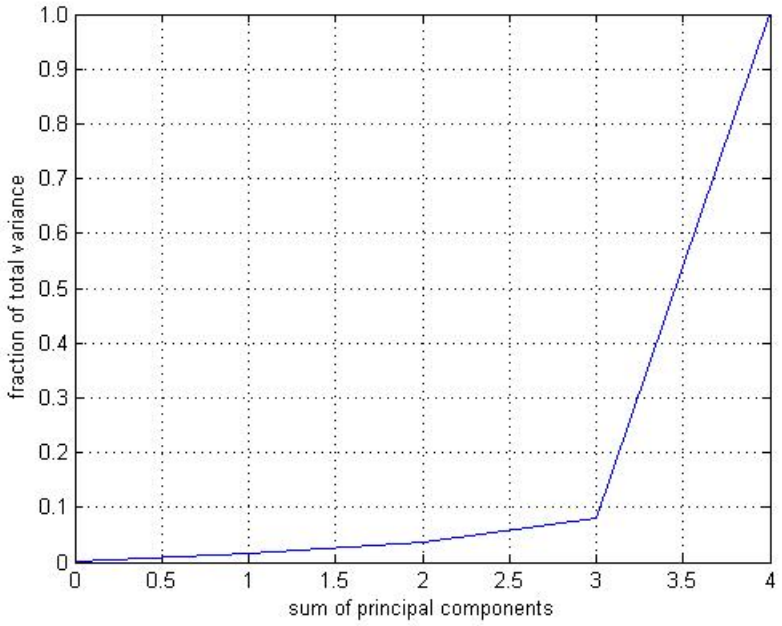

Figure 11. A plot of the fraction of total variance in the data as a function of the cumulative sum of the eigenvalues.

We see that more than 95% of the variance in the data is contained in the last two eigenvalues. Thus, we remove the first two principal components and use the remaining two components as input to (40) to create a projection of our data, which is shown in Figure 12 below:

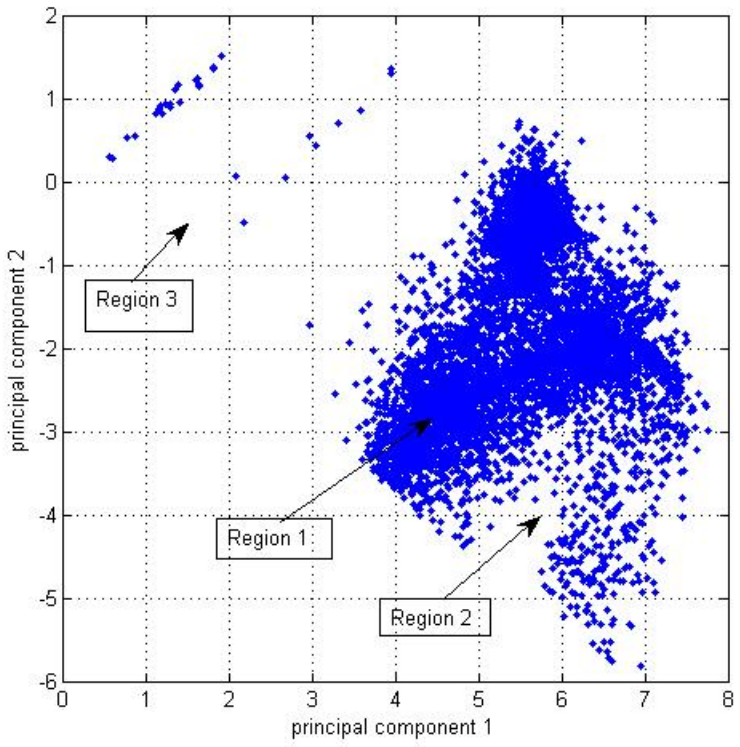

Figure 12. A projection of the data using the two principal components with the maximum variance.

Figure 12 provides evidence that the data set can be clustered into three distinct categories, each according to their density by using the DBSCAN algorithm. The densest region of data points seem to be present in region 1. Region 2 is substantially less dense. The least dense points seem to be present in region 3. The right side of region 3 seems to have mostly isolated points which suggests that they correspond to noise (the same can be said about some of the data points on the edges of region 1). In order to verify this conjecture and complete the clustering process DBSCAN must be performed on the data. The shape of these data clusters are not Gaussian, which (as we will see in the next section) suggests that a linear SVM will likely not classify the set of image data without substantial

error and thus a non-linear SVM must be used to perform the classification process.

Classifying Sets of Gaussian Data with Linear SVMs

In this section we describe how we can use a network of linear SVMs to classify three clusters of Gaussian data. Even though our set of data is not Gaussian, this demonstration provides a clear example of how a support vector network is able to correctly classify three data clusters. It also provides a simple framework for understanding how a non-linear SVM would function in a similar way.

First, three Gaussian data classes are generated by using the the Matlab command "mvnrnd" which creates a vector of multivariate normal random numbers. It requires us to specify the mean and covariance matrix for the data set.

In order to implement a support vector network which will classify a total of N classes of data, we create a support vector machine for each of the $\frac{N(N-1)}{2}$ unique class pairings. Thus for each pairing (i, j) we create a support vector machine that will classify a given data vector into one of two classes:i or j where we have $i = \{1, 2, ..., N-1\}$ and $j = \{i+1, i+2, ..., N\}$. In this example we use a total of three support vector machines to classify our three data types. The first half of the data set is trained in order to generate the decision boundary for each support vector machine. The code used to train the data is the LSVM algorithm that was taken (and simplified) from Mangasarian et al., 2001, and performs the update rule for u in (84). By training a particular pairing (i, j) of data clusters we produce the parameter w which is the normal vector to the decision boundary, and the parameter γ. These parameters are then used to create each of the three decision boundaries as in (68).

We then classify the second half of the data by taking each pairing (i, j), and

producing the output of the corresponding decision function as defined in (67) for every data vector. The output for a given data vector is either positive or negative and indicates into which class the decision function has classified the data vector. If the output is negative, then the decision function categorizes the vector as belonging to class i, and if it is positive, then the decision function categorizes the vector as belonging to class j. A given data vector will receive one vote from each support vector machine to indicate which class it belongs to. Because a given data vector will always be misclassified by one of the support vector machines, we determine which class has received the most votes. If the winning index is k, then we determine that the data vector ultimately belongs to class k.

Figure 13 shows a plot for each decision boundary and the data set which has been classified using the support vector network. Because the data set is linearly separable, it is classified by the support vector machine with 100% accuracy. We can clearly see this because each decision boundary perfectly separates two of the data classes. The plot is shown below:

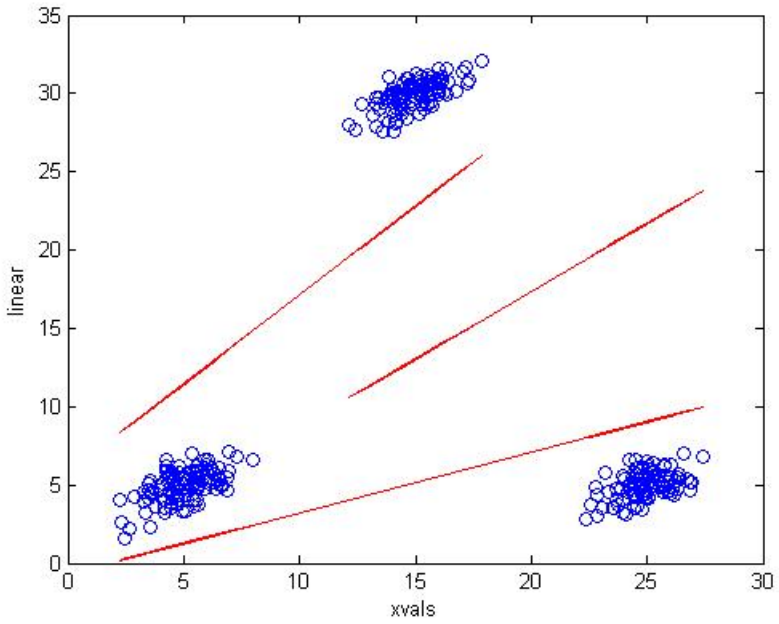

Figure 13. A plot of the three decision boundaries and the gaussian clusters that have been correctly classified.

Next, three different Gaussian clusters are generated, which cannot be perfectly separated by linear decision boundaries. This is because many of the data points are much closer together and thus more difficult to separate. In Figure 14, each of the data classes is represented by a different color. For argument's sake, we will label the red data points class 1, the green data points class 2, and the blue data points class 3.

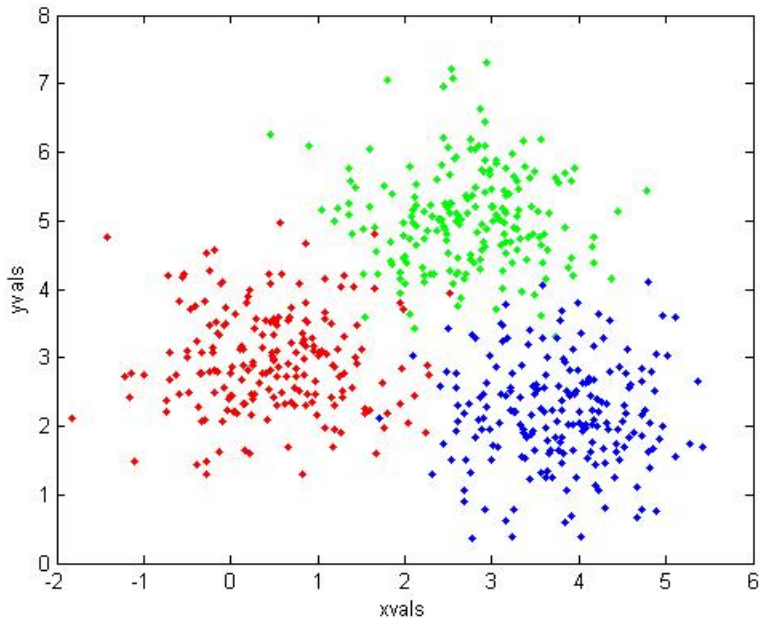

Figure 14. A gaussian data set that has classification error when using a linear support vector network.

To further illustrate how a linear support vector network will misclassify some of the data, the first half of the data belonging to classes 1 and 2 is trained in order to generate a decision boundary. The second half of the data is then used for classification and it is shown that the boundary does not completely separate the two data classes.

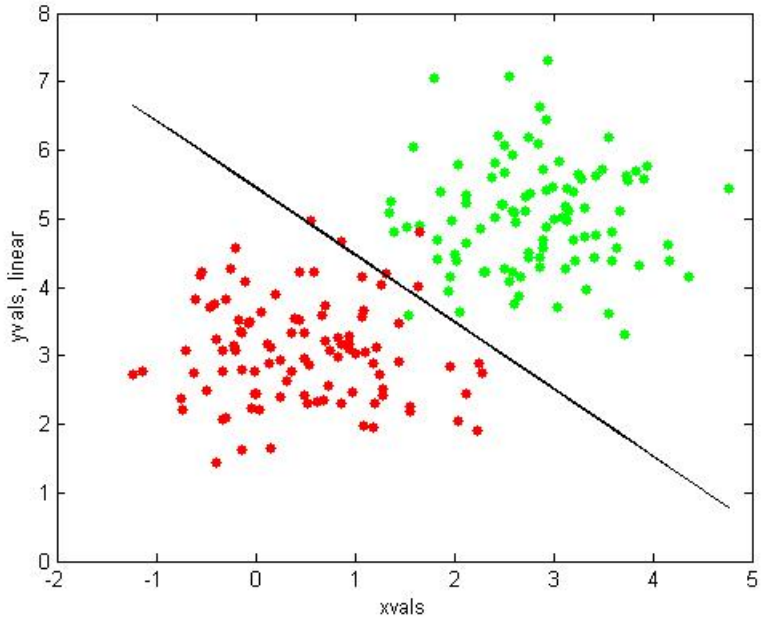

Figure 15. A plot of two of the data classes and the decision boundary used to separate them. The boundary cannot perfectly separate each data set, which indicates classification error.

Figure 15 shows even though some data points are misclassified, a linear support vector network is able to classify the data set with 94.3% accuracy. Thus, we see that linear support vector machines do well to classify Gaussian data clusters.

Conclusions

We have made progress toward the creation of an algorithm which is able to classify Fabry-Perot fringe profiles according to their image types. We have determined that singular value decomposition is an effective method to reduce the noise present in each fringe profile while simultaneously generating a Fourier series to model the data. Use of the LM algorithm has allowed us to acquire an accurate guess for the center of every image in our data profile and to find the value of the spatial frequency that produces a Fourier fit that minimizes the sum of errors. We are ensured that the phase shift calculated for each profile is independent of the choice of the image center, so we eliminate the possibility of the shift being caused by a drift in the camera. Additionally, our plots of the image center and spatial frequency as a function of the total time elapsed provide evidence that the laser used in this investigation shows unstable behavior.

Principal component analysis has served as an effective method to reduce the dimensionality of our feature data so it can be visualized. Our projection of the data set shows us that there appear to be three different regions of data, each sorted according to their density. If our projected data did not produce data clusters which could be spatially separated from one another based on their density, it would be evidence to suggest that PCA may have been inadequate. In this case, we could have proceeded by attempting a transformation of the data set prior to performing PCA, as suggested in Schlens, 2009.

In order to verify that we can sort the data into the three regions, we must perform DBSCAN as described by Ester et al., 1996 for future research. This will assign each image to a particular data cluster, so that they can be fed into a support vector network and then

classified. Because of the shape of each proposed cluster in the data set, we see that a non-linear SVM should perform much better than using a linear SVM. Alterations to our linear SVM should be made as discussed in Mangasarian et al., 2001 to produce a non-linear SVM.

After classification is complete, we can perform a more detailed stability analysis than the plots provided by the LM algorithm. Stability of the laser can be determined based upon how many images belong to each class. If we determine that more than a very small percentage of the images are either dark or show substantial phase shift, then we would conclude that the laser is not producing stable output and must undergo work before it can be used to take data involved in future experiments.

References

Bao, L., Wang, J., DeVito, M., Xu, D., Wise, D., Leisher, P. & Martinsen, R. (2010). Reliability of High Performance 9xx-nm Single Emitter Diode Lasers. Retrieved from http://www.nlight.net/nlight-files/file/technical_papers/PW10/Jan%2030%20Reliability%20of%20High%20Performance%209xx-nm%20Single%20Emitter%20Laser%20Diodes.pdf

Ben-Hur, A. & Weston, J. (2010). A User's Guide to Support Vector Machines. *Methods in Molecular Biology*, *609*, 223–239. Retrieved from http://www.cs.colostate.edu/~asa/pdfs/howto.pdf

Boyd, S. & Vandenberghe, L. (2004). *Convex Optimization*. New York: Cambridge University Press. Retrieved from http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf

Burges, C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, *20*, 121–167. Retrieved from http://research.microsoft.com/en-us/um/people/cburges/papers/svmtutorial.pdf

Clark, D. (2007). A Note on the Pseudoinverse. Retrieved from http://faculty.rmc.edu/davidclark/math/papers/pseudoinverse.pdf

Cortez, C. & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*, 273–297. Retrieved from http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf

Eliseev, P. G. (1991). Reliability Problems of Semiconductor Lasers. In *Proc. of the Lebedev Physics Institute Academy of Sciences of the USSR* (Vol. 197). New York: Nova Science Publishers.

Ester, M., Kriegel, H., Sander, J. & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-96)*. München : Institute for Computer Science, University of Munich. Retrieved from http://dns2.icar.cnr.it/manco/Teaching/2005/datamining/articoli/KDD-96.final.frame.pdf

Fakuda, M. (1991). *Reliability and Degradation of Semiconductor Lasers and LEDs*. Norwood: Artech House.

Fowles, G. (1989). *Introduction to Modern Optics*. Mineola: Dover Publications. (Original work published 1975)

Golub, G. H. & Van Loan, C. F. (1996). *Matrix Computations*. Baltimore: The John Hopkins University Press. Retrieved from http://eduunix.ccut.edu.cn/index2/math/202.38.126.65/mathdoc/Computing/Matrix%20Compute.pdf

Hofmann, T., Schölkopf, B. & Smola, A. (2008). Kernel Methods in Machine Learning. *The Annals of Statistics*, *36*(3), 1171–1220. Retrieved from http://alex.smola.org/papers/2008/HofSchSmo08.pdf

HPFSA Fused Silica Standard Grade. (2003, September). Retrieved from http://www.corning.com/docs/specialtymaterials/pisheets/H0607_hpfs_Standard_ProductSheet.pdf

Hrabina, J., Lazar, J., Holá, M. & Čip, O. (2013). Investigation of Short-term Amplitude and Frequency Flucuations of Lasers for Interferometry. *Measurement Science Review*, *13*(2). Retrieved from http://www.measurement.sk/2013/Hrabina.pdf

Hsu, C., Chang, C. & Lin, C. (2010). A Practical Guide to Support Vector Classification. Taiwan: Department of Computer Science, Information Engineering and National Taiwan University. Retrieved from http://www.ee.columbia.edu/~sfchang/course/spr/papers/svm-practical-guide.pdf

Joliffe, I. T. (2002). *Principal Component Analysis*. New York: Springer-Verlag New York. Retrieved from http://www.ucl.ac.uk/~rmjbale/Stat/PCA.pdf

Lutwak, R. (2011). EFTF-IFCS 2011 Tutorial. San Francisco: Symmetricom - Technology Realization Center. Retrieved from http://www.ifcs-eftf2011.org/sites/ifcs-eftf2011.org/files/editor-files/Slides_Lutwak.pdf

Mangasarian, O. L. (1994). *Nonlinear Programming*. Philadelphia: SIAM.

Mangasarian, O. L. & Musicant, D. R. (1999). Successive Overrelaxation for Support Vector Machines. *IEEE Transactions on Neural Networks*, *10*(5). Retrieved from http://www.cse.iitb.ac.in/~ganesh/cs705-guestLec/supplementary-reading/SORSVM.pdf

Mangasarian, O. L. & Musicant, D. R. (2001). Lagrangian Support Vector Machines. *Journal of Machine Learning Research*, *1*, 161–177. Retrieved from http://machinelearning.wustl.edu/mlpapers/paper_files/MangasarianM01.pdf

Optics, C. L. & Griot, M. (2009). Introduction to Laser Technology. Retrieved from https://www.cvimellesgriot.com/Products/Documents/TechnicalGuide/Laser-Guide.pdf

Osuna, E., Freund, R. & Girosi, F. (1997). Support Vector Machines: an Application to Face Detection. In *Proc. of Computer Vision and Pattern Recognition*. Cambridge: Center for Biological & Computational Learning and Operations Research Center,

Massachusetts Institute of Technology. Retrieved from http://www.cmlab.csie.ntu.edu.tw/~cyy/learning/papers/SVM_FaceCVPR1997.pdf

Ranganathan, A. (2004, June). The Levenberg-Marquardt Algorithm. doi:10.1.1.135.865.

Rätsch, G. & Soon Ong, C. (n.d.). Kernal Methods for Predictive Sequence Analysis. Tübingen: Friedrich Miescher Laboratory, Max Planck Society and Max Planck Institute for Biological Cybernetics. Retrieved from http://www.raetschlab.org/lectures/gcbtutorial/Tutorial-OngRaetsch-Summary.pdf

Saleh, B. & Teich, M. (2007). *Fundamentals of Photonics*. Hoboken: John Wiley & Sons.

Sauer, T. (2006). *Numerical Analysis*. Boston: Pearson Education.

Schlens, J. (2009). A Tutorial on Principal Component Analysis. New York City, La Jolla: Center for Neural Science, New York University and Systems Neurobiology Laboratory, Salk Insitute for Biological Studies. Retrieved from http://www.snl.salk.edu/~shlens/pca.pdf

Storey, B. (n.d.). Computing Fourier Series and Power Spectrum with MATLAB. Retrieved from http://www2.warwick.ac.uk/fac/sci/physics/research/cfsa/people/sandrac/lectures/basicfouriermatlab.pdf

Wandiger, U. (n.d.). Introduction to Lidar. Retrieved from http://superlidar.colorado.edu/Classes/Lidar2011/ReadingMaterials/IntroLidar.pdf

APPENDICES

Appendix A - Derivation of Fabry-Perot Phase Difference and Intensity Distribution

Derivation of phase difference. In this section we provide the derivation for the Fabry-Perot phase difference (1) as presented by Fowles, 1989. We can see from the geometry of Figure 12 that we need to have:

$$\overline{AB} = \overline{BC} = L = \frac{d}{cos(\theta_1)}$$

$$\overline{AC} = 2Lsin(\theta_1) = 2dtan(\theta_1)$$

$$\overline{AD} = \overline{AC}sin(\theta_0) = 2dtan(\theta_1)sin(\theta_0).$$

If the external index of refraction is μ_0, the etalon has index of refraction μ_1, and λ_0 is the laser central wavelength we can find the phase difference as follows:

$$\phi(ABC) = \frac{2\pi\mu_1}{\lambda_0}(\overline{AB} + \overline{BC}) = \frac{4\pi\mu_1 d}{\lambda_0}\frac{1}{cos(\theta_1)}$$

$$\phi(AD) = \frac{2\pi\mu_0}{\lambda_0}(\overline{AD}) = \frac{4\pi\mu_0 d}{\lambda_0}tan(\theta_1)sin(\theta_0)$$

$$\phi = \phi(ABC) - \phi(AD)$$

$$\phi = \frac{4\pi\mu_1 d}{\lambda_0}(\frac{1}{cos(\theta_1} - \frac{\mu_0}{\mu_1}tan(\theta_1)sin(\theta_0))$$

$$\phi = \frac{4\pi\mu_1 d}{\lambda_0}(\frac{1 - sin^2(\theta_1)}{cos(\theta_1)})$$

$$\phi = \frac{4\pi\mu_1 d}{\lambda_0} cos(\theta_1),$$

where we have used Snell's Law in (7) to write $\frac{\mu_0}{\mu_1} sin(\theta_0) = sin(\theta_1)$.

Next, we show the derivation for the phase difference in (3) which is written as an approximate function of ρ^2. First, Snell's law is used to write the cosine of the angle through the etalon as a function of the external angle:

$$cos(\theta_1) = (1 - [\frac{\mu_0}{\mu_1} sin(\theta_0)]^2)^{\frac{1}{2}}.$$

We now use the binomial expansion:

$$(1 + x)^n \approx 1 + nx + ...,$$

with $x = -(\frac{\mu_0}{\mu_1} sin(\theta_0))^2$ and $n = \frac{1}{2}$ to write:

$$(1 - (\frac{\mu_0}{\mu_1} sin(\theta_0))^2)^{\frac{1}{2}} \approx 1 - \frac{1}{2}(\frac{\mu_0}{\mu_1} sin(\theta_0))^2,$$

where the external angle θ_0 is assumed to be small. For a point on the image plane that is a radius ρ away from the optical axis, we can write:

$$sin(\theta_1) = \frac{\rho}{\rho + f}$$

where f is the focal length of the focusing lens. By assuming $f >> \rho$, we obtain the

expression:

$$1 - \frac{1}{2}(\frac{\mu_0}{\mu_1} sin(\theta_0))^2 \approx 1 - \frac{\mu_0^2}{2\mu_1^2}\frac{\rho^2}{f^2},$$

so the phase shift written as a function of pixel location becomes:

$$\phi(x,y) = \frac{4\pi\mu_1 d}{\lambda_0} cos(\theta_1) \approx \frac{4\pi\mu_1 d}{\lambda_0}(1 - \frac{\mu_0^2}{2\mu_1^2}\frac{\rho(x,y)^2}{f^2}).$$

Derivation of intensity distribution. In this section, we provide a derivation of the Fabry-Perot intensity distribution as presented by Fowles, 1989. Let us first derive the expression for the electric field. Assume that the transmission coefficient is t, the reflection coefficient is r, and that the ingoing electric field has amplitude E_0. Looking at Figure 2 once more, we see that the transmitted fields are $E_{T1} = E_0 t^2$ and $E_{T2} = E_0 t^2 r^2 e^{i\phi}$. If we continue in this way we see that the final transmitted electric field can be written as the following sum:

$$E_T = E_0 t^2 + E_0 t^2 r^2 e^{i\phi} + E_0 t^2 r^4 e^{2i\phi} + ...$$

$$= E_0 t^2 (1 + r^2 e^{i\phi} + r^4 e^{2i\phi} + ...).$$

If we use the geometric series given by:

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x} \; with \; |x| < 1$$

and let $x = r^2 e^{i\phi}$ then we obtain:

$$E_T = \frac{E_0 t^2}{1 - r^2 e^{i\phi}}.$$

If we square the magnitude of the electric fields and use the transmittance $T = |t^2|$ and the reflectance $R = |r^2|$ then we can write the intensity as follows:

$$I_T = I_0 \frac{T^2}{|1 - Re^{i\phi}|^2}$$

and we have:

$$|1 - Re^{i\phi}|^2 = (1 - Re^{i\phi})(1 - Re^{-i\phi}) = 1 - 2Rcos(\phi) + R^2,$$

so the expression becomes:

$$I_T = I_0 \frac{T^2}{1 - 2Rcos(\phi) + R^2}.$$

When we use the approximation (6) for the phase shift ϕ written in terms of ρ^2 we obtain:

$$I_T(\rho^2) \approx I_0 \frac{T^2}{1 - 2Rcos(\phi_0 - 2\pi\alpha\rho^2) + R^2},$$

so we have derived the intensity distribution for Fabry-Perot and shown that it is an approximate function of ρ^2.

Appendix B - Derivation of Least Squares Estimate

In this Appendix, we derive the least squares estimate (22). In terms of the calculus of variations, we seek to minimize the cost functional $J(x)$ where:

$$J(x) = \|Ax - b\|_2 = \langle Ax - b, Ax - b \rangle$$

and the subscript denotes that we are taking an l_2 norm. If we increment J by an amount proportional to h, then we obtain:

$$J(x; h) = \langle Ax - b, A(x + \beta h) - b \rangle.$$

By minimizing the variation in J with respect to the parameter β we obtain:

$$\delta J = \frac{\delta}{\delta \beta} \langle Ax - b, A(x + \beta h) - b \rangle |_{\beta=0} = 0$$

which yields:

$$\langle Ax - b, Ah \rangle = 0 \ \forall \ h$$

and since l_2 is a Hilbert Space, we can write this as:

$$\langle A^*(Ax - b), h \rangle = 0 \ \forall \ h,$$

where * denotes the complex conjugate. Since this equation holds for all h, then we arrive at:

$$A^*Ax = A^*b,$$

so the least squares estimate that minimizes the SSE is given by:

$$\hat{x} = (A^*A)^{-1}A^*b.$$

Appendix C - Linear Algebra

Review of basic concepts. In this section, we review some basic definitions from linear algebra which are needed to understand singular value decomposition (SVD). First, we define the transpose of a matrix, the inverse of a matrix, and the notion of a symmetric matrix. We also define eigenvectors and eigenvalues. If A is an $m\,x\,n$ matrix, we denote its transpose by A^T and define it as the matrix where we exchange every row with every column, so that $a_{ij}^T = a_{ji}$ for all i, j. Thus since A is an $m\,x\,n$ matrix, then A^T is an $n\,x\,m$ matrix. We denote the inverse of a matrix A to be A^{-1}, and define it to be the matrix which satisfies the equation $AA^{-1} = I$ where I is the identity matrix consisting of all 1's across the diagonal. We say that a a matrix A is symmetric if $A^T = A$, or written another way $a_{ij} = a_{ji}$ for all i, j. A vector e and a scalar λ which provide a solution to the equation:

$$Ae = \lambda e$$

are called an eigenvector e and an associated eigenvalue λ. We find the entire set of eigenvalues of a matrix A (Sauer, 2006) by solving for the roots λ of the characteristic polynomial defined by:

$$det(A - \lambda I) = 0.$$

The associated eigenvectors are found by taking scalar multiples of the nonzero vectors e which are in the nullspace of $A - \lambda I$.

Next we will define orthonormal vectors, the idea of an orthonormal matrix, and we state a property about orthonormal matrices. We say that two vectors u and v are

orthonormal if they are unit vectors and are orthogonal so that their inner product (dot product) is zero:

$$u \cdot v = u^T v = u_1 v_1 + u_2 v_2 + ... = 0.$$

A set of vectors $\{u_n\}$ forms an orthonormal set if every vector in the set is a unit vector and is mutually orthogonal. Similarly, a matrix Q is orthonormal if its column vectors form an orthonormal set. A very useful property of orthonormal matrices is that $Q^T = Q^{-1}$, which is equivalent to saying that $Q^T Q = I$. This is easily shown (Schlens, 2009) since the ij^{th} element of $Q^T Q$ is

$$(Q^T Q)_{ij} = q_i^T q_j = \delta_{ij},$$

where δ_{ij} is the Kroenecker delta, so $\delta_{ij} = 1$ when i = j, and $\delta_{ij} = 0$ when i $\neq$ j. So we must have $Q^T Q = I$ and thus $Q^T = Q^{-1}$.

A proof about symmetric matrices. In this section, we provide a proof to equation (23) regarding a symmetric matrix A as discussed by Schlens, 2009. To show that (23) is true for symmetric matrices, we first show that any matrix A with independent eigenvectors can be written as:

$$AE = ED.$$

If we assume that E is defined as in the above equation, then we see that the right hand side means that:

$$AE = A\begin{bmatrix} e_1 & e_2 & ... & e_n \end{bmatrix}$$
$$AE = \begin{bmatrix} Ae_1 & Ae_2 & ... & Ae_n \end{bmatrix}$$

and then the eigenvalue equation gives us:

$$\begin{bmatrix} Ae_1 & Ae_2 & ... & Ae_n \end{bmatrix} = \begin{bmatrix} \lambda_1 e_1 & \lambda_2 e_2 & ... & \lambda_n e_n \end{bmatrix} = ED$$

and thus we have seen that $AE = ED$. If this equation is written as $A = EDE^{-1}$ and if A is also symmetric, then we have finally shown that we can write $A = EDE^T$ for a symmetric matrix A.

Appendix D - Selection of the Projection Matrix in PCA

In this Appendix, we provide a proof (as discussed by Schlens) of the claim that choosing the matrix P as defined in (33) such that each column is an eigenvector of X^TX will diagonalize the covariance matrix C_Y. To verify this, we first write C_Y in terms of P as:

$$\begin{aligned} C_Y =& \frac{1}{n-1} Y^T Y \\ =& \frac{1}{n-1} (XP)^T (XP) \\ =& \frac{1}{n-1} P^T (X^T X) P. \end{aligned}$$

Because X^TX is symmetric, we can write it as in (53) so that:

$$C_Y = \frac{1}{n-1} P^T (EDE^T) P$$

and our choice of P ensures that $P = E$ so that:

$$\begin{aligned} C_Y =& \frac{1}{n-1} (P^T P) D (P^T P) \\ C_Y =& \frac{1}{n-1} D, \end{aligned}$$

which shows us that C_Y has been diagonalized.

Appendix E - Lagrangian SVMs and Non-Linear SVMs

Linear SVMs using the Lagrangian. Now that we have discussed the basics of linear SVMs, we provide a reformulation of the problem using Lagrange multipliers. This will be an extremely useful interpretation of the problem, since it will allow us to very easily begin a discussion about kernel methods and non-linear SVMs. First we will create the Lagrangian of our problem (Burges, 1998). We take each constraint c_i which has been written in the form $c_i \geq 0$ and multiply it by a positive Lagrange multiplier α_i, then subtract the objective function which is to be minimized. Our primal Lagrangian for the soft-margin SVM will then become (Burges, 1998):

$$L_P = \frac{1}{2}||w||^2 + \nu \sum_{i=0}^{n} \xi_i - \sum_{i=0}^{n} \alpha_i \{y_i(w^T x_i - \gamma) - 1 + \xi_i\} - \sum_{i=0}^{n} \mu_i \xi_i,$$

so we seek to solve the primal optimization problem (Burges, 1998):

$$\begin{aligned} &\underset{w,\gamma,\xi}{\text{minimize}} \quad L_P \\ &\text{subject to:} \quad \frac{\partial L_P}{\partial \alpha_i} = 0, \quad \frac{\partial L_P}{\partial \mu_i} = 0, \quad \alpha_i \geq 0 \quad \mu_i \geq 0. \end{aligned}$$

We can solve the equivalent dual problem which is formulated as follows (Burges, 1998):

$$\begin{aligned} &\underset{w,\gamma,\xi}{\text{maximize}} \quad L_P \\ &\text{subject to:} \quad \nabla_w L_P = 0, \quad \nabla_\gamma L_P = 0, \quad \alpha_i \geq 0 \quad \mu_i \geq 0 \end{aligned}$$

and our constraints for the dual yield the conditions (Burges, 1998):

$$w = \sum_{i=0}^{n} \alpha_i y_i x_i$$

$$\sum_{i=0}^{n} \alpha_i y_i = 0.$$

It is important to note that the data points x_i which correspond to Lagrange multipliers $\alpha_i > 0$ are support vectors which either lie on or within the margin. Because these constraints for the dual must also be true for the primal Lagrangian, we will substitute them in order to obtain (Burges, 1998):

$$L_D = \sum_{i=i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i^T x_j,$$

which is the dual formulation of the Lagrangian. So our new formulation of our optimization problem is as follows (Burges, 1998):

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & L_D \\ \text{subject to:} \quad & \sum_{i=0}^{n} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq \nu. \end{aligned}$$

This Lagrangian formulation will be the starting point for us to begin discussing kernel-based methods and non-linear SVMs, which is the topic of the next section.

Kernel methods and non-linear SVMs. As we have stated in the previous sections, many data sets are not linearly separable, and cannot be accurately classified by linear decision boundaries. Instead we need to rely on non-linear decision boundaries, and

hence we must create an interpretation for a non-linear SVM. Even if the data is linearly separable and has a fair degree of accuracy by using a linear SVM, we can oftentimes substantially increase the accuracy by using non-linear SVMs. Thus, in this section we will describe the basic concepts of non-linear SVMs using arguments presented by (Ben-Hur et al., 2010).

Let's start by defining a non-linear decision function, by replacing our variable x, with some non-linear function $\phi(x)$ with $\phi : X \to F$ which will map our original feature space X into some new and possibly higher dimensional space F (Ben-Hur et al., 2010):

$$f(x) = w^T \phi(x) + b.$$

We can immediately see that we have some difficulty in utilizing such a decision function. We started with a problem that had a fairly high dimensional input space, and we are now trying to compute in a space which is quadratic, or even higher dimensional with respect to our data vectors. The number of operations in performing the computation required becomes increasingly cumbersome as we increase the dimension of our original input space (Ben-Hur et al., 2010). How can we create a non-linear decision function while avoiding a problem with scaling its computational complexity?

Let us broaden our definition of support vector machines, to lump in SVMs as part of a broad category of algorithms that are called kernel methods. Kernel methods do not directly depend on a given data set; instead they indirectly depend on the data through dot products. If we examine the dual Lagrangian L_D, we see that the data set *only* appears

as the dot product $x_i^T x_j$. We will now write the Lagrangian as:

$$L_D = \sum_{i=i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j k(x_i, x_j),$$

where the *linear* kernel function is (Ben-Hur et al., 2010):

$$k(x_i, x_j) = x_i^T x_j.$$

If we return to our linear discriminant function (67), and if we assume the weight vector w can be expressed as a linear combination of the training samples x_i then we have (Ben-Hur et al., 2010):

$$f(x) = \sum_{i=1}^{n} \alpha_i x_i^T x - \gamma,$$

which we express in terms of the kernel function as (Ben-Hur et al., 2010):

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x) - \gamma.$$

Rewriting the decision function this way for the linear case is not extremely helpful, but if we perform the same procedure for a non-linear case, we will see that this approach will have considerable advantage. If we use the decision function defined above along with a non-linear kernel function (Ben-Hur et al., 2010):

$$k(x, y) = \phi(x)^T \phi(y),$$

then we will be able to avoid directly mapping our data into a complex feature space. By

picking some simple functions as possibilities for ϕ, we can reason that we can write our kernel function $k(x, y)$ in terms of the dot product $x^T y$. We are able to compute the function $k(x, y)$ without ever having to make explicit reference to the function ϕ in our computation. We can now see that our method involving the kernel function requires substantially less effort to compute $f(x)$ than our original formulation of the problem.

We now mention some of the kernel functions that are prominent throughout the literature. The degree d polynomial kernel is a simple extension of the linear kernel and can be defined as (Hsu et al., 2010):

$$k(x_i, x_j) = (\gamma x_i^T x_j + r)^d,$$

where $\gamma > 0$ and r are kernel parameters. In general, as the degree of the polynomial kernel is increased we add curvature to the decision boundary. One of the most popular and effective kernel functions is the radial basis function (RBF) kernel, oftentimes referred to as a Gaussian kernel (Hsu et al., 2010):

$$k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2},$$

where $\gamma > 0$. This kernel has the effect of creating decision boundaries that have several bumps in them. As the factor of γ is increased, so does the curvature and irregularity of these bumps.

Before kernel functions are computed in order to construct a non-linear SVM, it is generally a good idea to perform basic scaling of the data. This can help reduce numerical

difficulties when choosing a range for kernel values. In order to search for the best penalty term C and appropriate kernel parameters, a cross-validation search is often performed. Details on these procedures are discussed in Hsu et al., 2010.