Alignment by Maximization of Mutual Information

by

Paul A. Viola

Master of Science in Computer Science,
Massachusetts Institute of Technology (1990)

Bachelor of Science in Computer Science and Electrical Engineering,
Massachusetts Institute of Technology (1988)

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the
Massachusetts Institute of Technology
June, 1995

Signature of Author [illegible]

Paul A. Viola

Certified by [illegible]

Professor Tomás Lozano-Pérez, Thesis Co-Supervisor

Certified by

Professor Christopher G. Atkeson, Thesis Co-Supervisor

Accepted by [illegible]

Professor F. R. Morgenthaler
Chairman, Departmental Graduate Committee

Alignment by Maximization of Mutual Information

by

Paul A. Viola

Submitted to the Department of Electrical Engineering and Computer Science on June 1995, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Over the last 30 years the problems of image registration and recognition have proven more difficult than even the most pessimistic might have predicted. Progress has been hampered by the sheer complexity of the relationship between an image and an object, which involves the object's shape, surface properties, position, and illumination.

Changes in illumination can radically alter the intensity and shading of an image. Nevertheless, the human visual system can use shading both for recognition and image interpretation. We present a metric for comparing objects and images that uses shading information, yet is explicitly insensitive to changes in illumination. This metric is unique in that it compares 3D object models directly to raw images. No pre-processing or edge detection is required. The metric has been rigorously derived from information theory.

In our derivation few assumptions are made about the nature of the imaging process. As a result the algorithms are quite general and can be used in a wide variety of imaging situations. Experiments demonstrate this approach aligning a number of complex 3D object models to real images. In addition, we demonstrate that the same technique can be used to solve problems in medical registration.

Alignment is accomplished by adjusting the pose of an object until the mutual information between image and object is maximized. We will present a gradient descent alignment procedure based on stochastic approximation that has an efficient implementation. Stochastic approximation affords a speed up of at least a factor of 500 in this application. Though stochastic approximation is 40 years old it is not widely used in computer vision. There are a number of vision problems to which it is applicable. We will demonstrate how stochastic approximation can be used to accelerate an existing vision application by a factor of 30.

EMMA may also prove useful for a wide variety other tasks. Experiments will demonstrate that EMMA can be used in medical image processing and machine learning.

Thesis Committee:	Prof. Tomas Lozano-Perez (Co-Supervisor)
	Prof. Christopher G. Atkeson (Co-Supervisor)
	Prof. W. Eric L. Grimson
	Prof. Berthold K. P. Horn

Acknowledgments

I would like to acknowledge the MIT Artificial Intelligence Laboratory for being a haven of intellectual freedom. Professors like Tomás Lozano-Pérez, Christopher Atkeson, Eric Grimson, Rodney Brooks, and others have worked to build an environment where every resource is unfettered. Chris and Tomas have supported me through thick and through thin. They've taught me how truly valuable unconditional support can be.

The students are the AI lab's most precious resource and I benefited from time with Phil Agre, Davi Geiger, David Chapman, Jose Robles, Tao Alter, Misha Bolotski, Jonathan Connel, Karen Sarachik, Maja Mataric, Ian Horswill, Colin Angle, Cynthia Ferrel, Henry Minsky, Saed Younis, Rick Lathrope, Barbara Moore, and many others. Willliam Wells has stood out among all those I've met at MIT. His unique approach to vision, his care in research and his advice as a friend have proven invaluable. I owe much to John Shewchuk, then of Brown University, for introducing me to the field of statistical learning. Memories of his irrepressible intellectual curiosity serve to continually motivate my own thinking.

Outside of MIT I have spent many productive months at the Salk Institute in the Computational Neurobiology Laboratory of Terrence Sejnowski. Terry is the most tirelessly devoted scientist that I have ever met. In his lab I learned that science is an uncompromising pursuit of truth. Science is about building on the work of others, and that to build one must first understand. In Terry's lab I have had the pleasure of working with David Lawrence, Rich Zemel, Nici Shraudolph, Tony Bell, and Peter Dayan. Each of them has had an effect on some part of this thesis.

Most importantly I must recognize the technical contributions of Sara Billey. She worked tirelessly with me to understand the most cryptic mathematics and clear up my own cryptic thinking. Without her this thesis would not exist.

To

Sara Billey for being the love of my life.

My parents Mary Ancona-Viola and Alfredo Viola for making it all possible.

Contents

Chapter 1

Introduction

Over the last 30 years the problems of image registration and recognition have proven more difficult than even the most pessimistic might have predicted. Though human visual perception is both effortless and robust, computational vision is costly and unreliable. Progress in computer vision has been hampered by the sheer complexity of the relationship between image and object, which involves the object's shape, surface properties, position, and illumination.

A computer vision program is faced with the task of interpreting an image of intensities. Because changes in illumination can radically alter the intensity and the shading in an image, raw image intensities have been widely considered unusable. Many existing vision systems throw out intensity and shading information in an effort to obtain "illumination invariance". Nevertheless, the human visual system can use shading both for recognition and image interpretation. We will present a measure for comparing objects and images that uses shading information, yet is explicitly insensitive to changes in illumination. This measure is unique in that it compares 3D object models directly to raw images. No pre-processing or edge detection is required.

This image/model comparison measure has been rigorously derived from information theory. The theory and algorithms involved are new, at least within the field of computer vision. We will present an overview of the requisite theory including probability, statistics, and entropy. This review will culminate in the presentation of a concrete and efficient scheme for evaluating mutual information called EMMA[1]. We will demonstrate that mutual information is an effective measure of image/model alignment.

In our derivation of mutual information based alignment, few assumptions are made about the nature of the imaging process. As a result the algorithms are quite general and can be used in a wide variety of imaging situations. Experiments demonstrate this approach aligning a number of complex 3D object models to real images. In addition, we demonstrate that the same technique can be used to solve problems in medical registration.

Alignment is accomplished by adjusting the pose of an object until the mutual information between image and object is maximized. We will present a gradient descent alignment procedure based on stochastic approximation that has an efficient implementation. Stochastic approximation affords a speed up of at least a factor of 500 in this application. Though stochastic approximation is 40 years old it is not widely used in computer vision. There are a number of vision problems to which it is applicable. We will demonstrate how stochastic approximation can be used to accelerate an existing vision application by a factor of 50.

EMMA may also prove useful for a wide variety other tasks. Experiments will demonstrate that EMMA can be used in medical image processing and machine learning.

1.1 An Introduction to Alignment

The general problem of alignment entails comparing a predicted image of an object with an actual image. Given an object model and a pose (coordinate transformation), a model for the imaging process could be used to predict the image that will result. This is typically a difficult problem. If we had a good imaging model then deciding whether an image contained a particular model at a given pose is straightforward: compute the predicted image and compare it to the actual image directly. Given a perfect imaging model the two images will be identical, or close

[1]EMMA is a random but pronouncible subset of the letters in the words "EMpirical entropy Manipulation and Analysis".

to it. Of course finding the correct alignment is still a remaining challenge.

The relationship between an object model (no matter how accurate) and the object's image is a complex one. The appearance of a small patch of a surface is a function of the surface properties, the patch's orientation, the position of the lights and the position of the observer. In the part of the scene containing an image of the object, we can formulate an imaging equation

$$v(T(x)) = F(u(x), q) \ . \tag{1.1}$$

The imaging equation in in principal separable into two distinct components. The first component is called a transformation, or pose, denoted $T(x)$. It relates the coordinate frame of the model to the coordinate frame of the image. The transformation tells us which point in the model is responsible for a particular point in the image. The second component is the imaging function, $F(u(x), q)$. The imaging function determines the value of image point $v(T(x))$. In general a pixel's value may be a function both of the model and other exogenous factors. For example an image of a model depends not only on the model but also on the lighting. The parameter, q, collects all of the exogenous influences into a single vector.

One reason that it is, in principle, possible to define F is that the image does convey information about the model. Clearly if there were no mutual information between u and v, there could be no meaningful F. We propose to finesse the problem of finding and computing F by dealing with this mutual information directly. Such a technique would attempt to find the alignment of the model in an image by maximizing the information that the image provides about the model. We will present an algorithm that does just this. It requires no a priori model of the relationship between surface properties and scene intensities – it only assumes that the model tells more about the scene when it is correctly aligned.

1.1.1 An Alignment Example

One of the alignment problems that we will address involves finding the pose of a three-dimensional object that appears in a video image. This problem involves comparing two very different kinds of representations: a three-dimensional model of the shape of the object and a video image of that object. For example, Figure 1.1

contains a video image of an example object on the left and a depth map of that same object on the right (the object in question is a person's head: RON). A depth map is an image that displays the depth from the camera to every visible point on the object model. A depth map like this is a complete description of the shape of the object, at least the visible parts.

From the depth map alone it might be difficult to see that the image and the model are aligned. The task can be made much easier, at least for us, if we simulate the imaging process and construct an image from the 3D model. Figure 1.2 contains two computer graphics renderings of the object model. These synthetic images are constructed assuming that the 3D model has a Lambertian surface and that the lighting comes from the right. It is almost immediately obvious that the model on the left is more closely aligned to the true image than the model on the right. Unfortunately, what we find trivial is very difficult for a computer. The intensities of the true video image and the synthetic images are very different. The true image and the correct model image are in fact uncorrelated. Yet any person can glance at these images and decide that both are images of a head and that both heads are looking in roughly the same direction. The human visual system is capable of ignoring the superficial differences that arise from changes in illumination and surface properties.

It is not easy to build an automated alignment procedure that can make this kind of comparison. It is harder still to construct a system that can find the correct the model pose. We have built such a system. That system selected the pose of the model shown at left in Figure 1.2.

As mentioned above, the synthetic images of RON were generated under the assumption the model surface is Lambertian and the lighting is from the right. The Lambertian model is perhaps the simplest model of surface reflectivity. It is an accurate model of the reflectance of a matte or non-shiny surface. The model states that the visible intensity of a surface patch is related to the dot product between the surface normal and the lighting. For the Lambertian model the imaging equation is:

$$v(Tx) = \sum_i \alpha_i \vec{l_i} \cdot u(x) \ , \tag{1.2}$$

where the model value $u(x)$ is the normal vector of a surface patch on the object, l_i is a vector pointing toward light source i, and α_i is proportional to the intensity

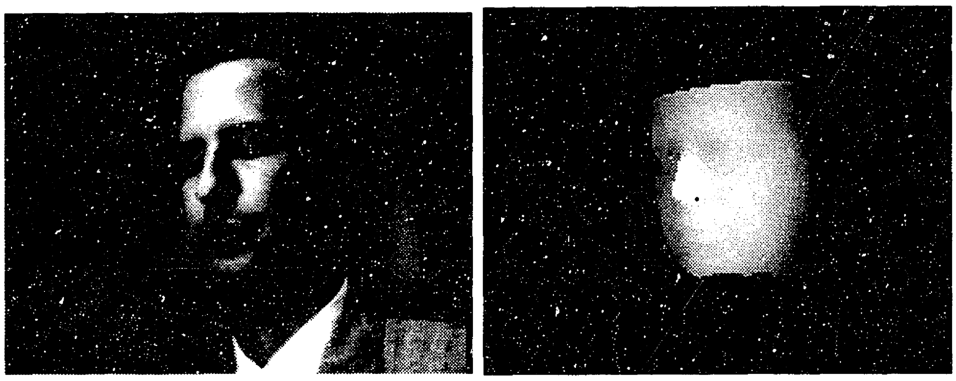


Figure 1.1: Two different views of RON. On the left is a video image. On the right is a depth map of a model of RON. A depth map describes the distance to each of the visible points of the model. White denotes points that are closer, black further.

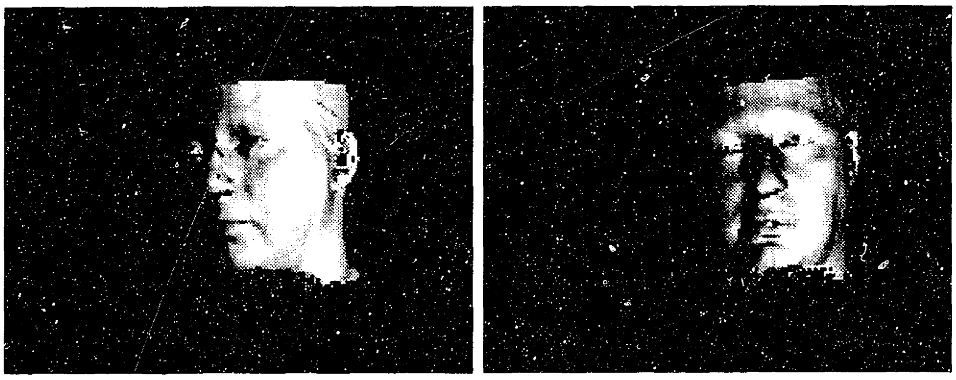


Figure 1.2: At left is a computer graphics rendering of a 3D model of RON. The position of the model is the same as the position of the actual head. At right is a rendering of the head model in an incorrect pose.

of that light source ((Horn, 1986) contains an excellent review of imaging and its relationship to vision). So we can see that even for this simple model, the image is not a function of the model alone. The lighting must be known before an image of an object can be predicted. As the illumination changes the relationship between model and image will change. There will be a different imaging function $F(\cdot)$ for each illumination condition.

Since we can not know beforehand what the imaging function will be, aligning a model and image can be quite difficult. These difficulties are only compounded if the surface properties of the object are not well understood. For example, many objects can not be modeled as having a Lambertian surface. Different surface finishes will have different reflectance functions. In general reflectance is a function of lighting direction, surface normal and viewing direction. The intensity of an observed patch is then:

$$v(Tx) = \sum_i R(\alpha_i, \vec{l_i}, \vec{o}, u(x)) \ , \tag{1.3}$$

where $\vec{o}$ is a vector pointing toward the observer from the patch. For an unknown material a great deal of experimentation is necessary to completely categorize the reflectance function. Since a general vision system should work with a variety of objects and under general illumination conditions, overly constraining assumptions about reflectance or illumination should be avoided.

Let us examine the relationship between a real image and model. This will allow us to build intuition about the alignment process. Data from the real reflectance function can be obtained by aligning a model to a real image. An alignment associates points from the image with points from the model. If the alignment is correct, each pixel of the image can be interpreted as a sample of the imaging function $R(\cdot)$. The imaging function could be displayed by plotting intensity against lighting direction, viewing direction and surface normal. Unfortunately, because intensity is a function of so many different parameters the resulting plot can be prohibitively complex and impossible to visualize. Significant simplification will be necessary if we are to detect any structure in this data.

In a wide variety of real images we can assume that the light sources are far from the object (at least in terms of the dimensions of the object). When this is true and there are no shadows, each patch of the object will be illuminated in the same way. Furthermore, we will assume that the observer is far from the object, and that the viewing direction is therefore constant throughout the image. The resulting relationship between normal and intensity is three dimensional: the normal vector has unit length and is determined by two parameters, x component and y component; the intensity is a third parameter. A three dimensional scatter plot of normal versus intensity is really a slice through the high dimensional space in which $R(\cdot)$ is defined. Though this graph is much simpler than the original, three dimensional plots are still quite difficult to interpret. We will slice the data once again so that all of the points have a single value for the y component of the normal.

Figure 1.1 contains a graph of the intensities along a single scan-line of the image of RON. These intensities come from the region indicated in Figure 1.1. On the left in Figure 1.2 is the RON model aligned with the image. Marked on the model is the same scan-line. Data from this scan-line is displayed in two graphs: the first shows the x component of the normal while the second shows the y component of the normal. Notice that we have chosen this portion of the model so that the y component of the normal is almost constant. As a result the relationship between normal and intensity can be visualized in only two dimensions. Figure 1.3 shows the intensities in the image plotted against the x component of the normal in the model. Notice that this relationship appears both consistent and functional. Points from the model with similar surface normals have very similar intensities. The data in this graph could be well approximated by a smooth curve. We will call an imaging function like this one *consistent*. Interestingly, we did not need any information about the illumination or surface properties of the object to determine that the resulting alignment led to a consistent relationship between model normal and image intensity.

Conversely we could look at the relationship between normal and intensity when the model and image are no longer aligned. Figure 1.3 shows such a graph. The only difference between this graph and the first is that the intensities come from a scan-line 3 centimeters below the correct alignment (i.e. the model is no longer aligned with the image, it is 3 centimeters too low). The normals used are the same. The resulting graph is no longer *consistent*. It does not look as though a simple smooth curve would fit this data well.

In summary, when the model and the image are aligned they are consistent.

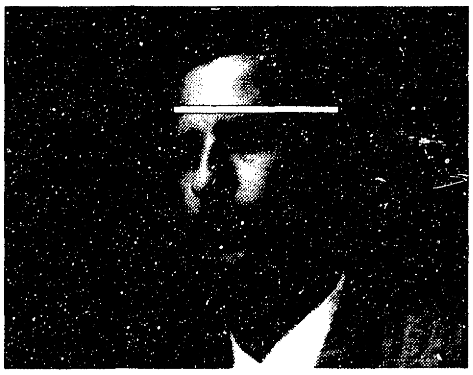
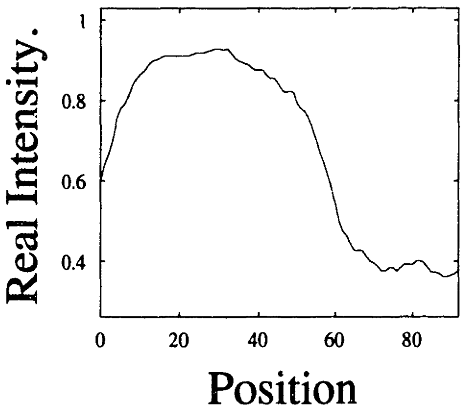
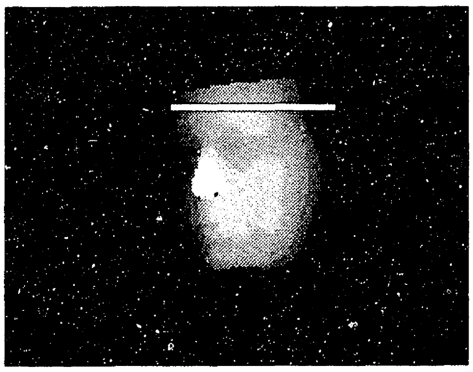
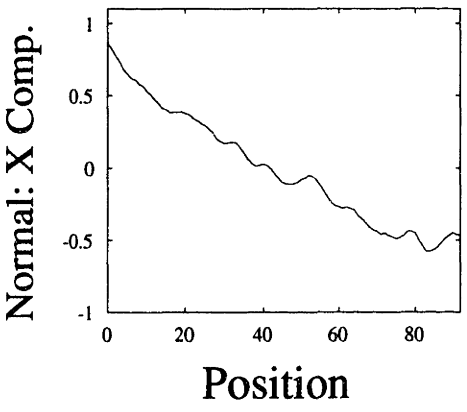



Table 1.1: On the left is a video image of Ron with the single scan-line highlighted. On the right is a graph of the intensities observed along this scan line.

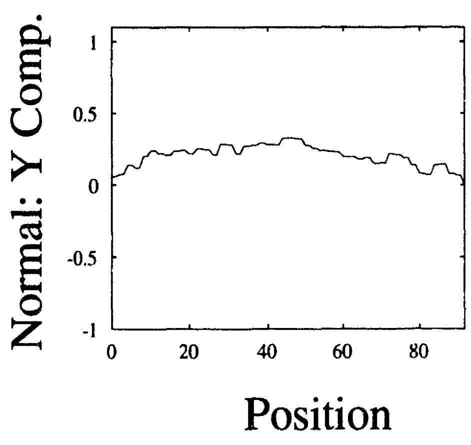


Table 1.2: On the left is a depth map of Ron with the single scan-line highlighted. At center is a graph of the x component of the surface normal. On the right is the y component of the normal.

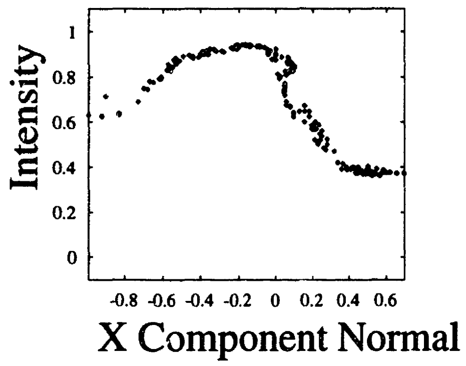


Figure 1.3: THE ALIGNED CASE: A scatter plot the intensity of the video image versus the x component of the surface normal from the model. The image and model a correctly aligned.

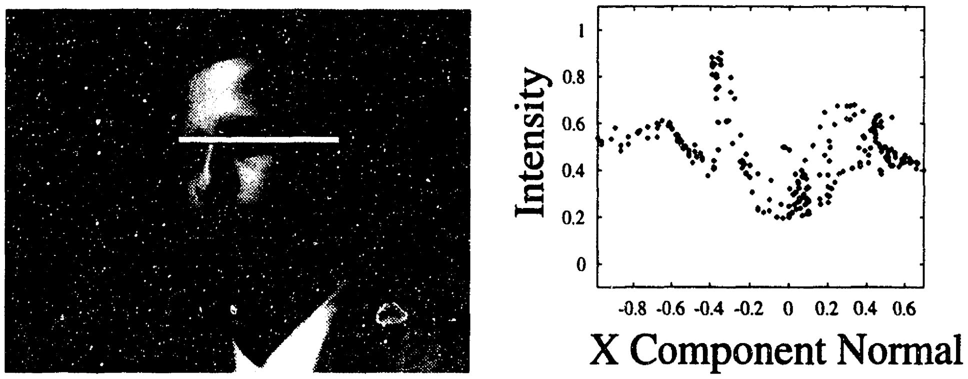


Table 1.3: THE MISALIGNED CASE: On the left is the mis-aligned scan-line from the video image of RON. On the right is a scatter plot the intensity of this part of the video image versus the x component of the surface normal from the model.

When they are misaligned they are inconsistent. As the lighting or surface properties change the imaging function can change radically, but it is always consistent.

A major contribution of this thesis is the derivation of a formal technique that delivers a principled estimate of "consistency". We will show that when the mutual information between an image and a model is high they are likely to be aligned. Toward making this technique a reality we have defined a new approach for evaluating entropy and information called EMMA. We have also defined an efficient scheme for adjusting a set of parameters so that the entropy can be optimized. We will use EMMA to effectively evaluate and adjust the alignment of three dimensional models and two dimensional images.

This same alignment technology can be used for the alignment of other types of images. We will show that EMMA can be used when there is a need to align images from two different sensors, the so-called "sensor fusion" problem.

Though developed for alignment, the EMMA estimates of entropy and mutual information can be used in other applications. We will show that EMMA can be used to correct inhomogeneities in MRI scans. In addition, we will derive a new approach for dimensionality reduction based on entropy. Similar to principal components analysis, our technique can find low dimensional projections of higher dimensional data that preserve the most possible information.

1.2 Overview of the Thesis

The second chapter contains an overview of the probability theory necessary to understand what EMMA is doing and how it does it. The third chapter discusses estimation of entropy from samples. While a number of techniques for manipulating entropy currently exist, EMMA combines computational efficiency with the flexibility to model a wide variety of distributions. The fourth chapter returns to our discussion of alignment. We show here that EMMA is capable of aligning signals where simpler techniques cannot. This chapter will present the basic equations that underly alignment by maximization of mutual information. The fifth chapter contains a wide variety of alignment experiments designed both to validate our approach and explore the scope possible application. In chapter six we will describe applications besides alignment to which we have applied EMMA. For example, our scheme for efficiently manipulating entropy includes a stochastic form of gradient descent. We will describe a flow estimation problem in which stochastic gradient descent speeds convergence by a factor of thirty. Chapter seven will include a discussion of our results and a comparison with related work.

Chapter 2

Probability and Entropy

One of the key insights in this thesis is that many of the techniques that are common in computer vision, such as correlation, are easily interpreted as statistics of random variables. Once this is done we can use a broad range of tools from probability to analyze the behavior of these statistics. The theory of probability will help us determine how statistics converge, what they converge to, and more importantly how alternative statistics might be more appropriate.

In this chapter we will introduce the basic mathematics that underly probability. In subsequent chapters we will assume that the reader has a fairly thorough knowledge of probability, statistics, entropy, and coding. This chapter is intended both as a review of the required techniques and theorems and as a bridge for a reader unfamiliar with these topics. The final sections of this chapter contain a new analysis and discussion of Parzen density estimation. Parzen density estimation will play an important role in the estimation of entropy in subsequent chapters.

We will sometimes use a simplified, or looser, definition of concepts like events and random variables than is typical. If you get overly confused reading this chapter, any good book on probability should clear things up (Papoulis, 1991; Baclawski et al., 1990). In general we will leave out the proofs of anything that is easily looked up, and of course most of the theory presented is cited here without reference. Unfortunately probability and statistics seems to have many conflicting standard notations. In our own definitions we will try to be consistent with the prevailing conventions.

2.1 Random Variables

In many cases an algebraic model of a physical system allows us to accurately predict its behavior. For instance, circuit theory can be used to analyze a particular circuit and predict that when a switch is closed current will flow. The physics of many circuits can be modeled as equations where unknown quantities are recorded as variables. In the case of a switched circuit, we can model the resistance of a switch as a variable that can take on one of two values: zero when closed or infinity when open. The current that flows through that resistor can then be predicted from algebraic manipulations. Conversely, knowing the value of the current allows us to predict whether the switch is open or closed. The equivalence of a circuit and a circuit model is fundamental within the fields of physics and engineering.

In a wide variety of physical systems the behavior of particular measurements cannot be easily predicted. The voltage of a wire may be a complex function of the circuit and the thermal noise in a resistor. Even when all of the other circuit variables are known, the voltage cannot be predicted accurately. Luckily, all hope is not lost. We may not know the actual voltage but we may know that it will be "near" V_0, and that it is never, in our experience, higher than V_{max}. Probability, random processes, and random variables provide the tools to quantify the intuitive concepts of "near" and "never".

A random variable, or RV, is a variable whose value is unpredictable. Recall that a variable is a symbol X and a set of values Ω_X over which the variable can range. For example, X could range over the real numbers between 1 and 10. In this thesis we will assume Ω_X will always be a subset of the real numbers. A random variable X is a variable together with a function $P_X : \Omega_X \rightarrow [0,1]$ called a probability distribution. For example we can construct an RV that models the roll of a six sided die. If the die is "fair" we cannot in advance know what its value will be, but we do know that its value will be one of 6 integers from 1 to 6, and that each will appear roughly one sixth of the time. The RV that describes this die includes the variable symbol X, a sample space $\Omega_X = \{1,2,3,4,5,6\}$ of possible outcomes, and a probability distribution function $P_X(n)$ which tells us the probability that X will take on the value n. A particular value of an RV is called a *trial*, for example from a die roll. A collection of trials is called a *sample*. An *event* is a set A such that $A \subset \Omega_X$. The probability of an event, $P_X(X \in A)$ is the proportion of times that you expect to see event A in a large sample. The sum over the sample space of the probability distribution equals one:

$$\sum_{x_i \in \Omega_X} P(X \in \{x_i\}) = 1 \ .$$

Here we denote the elements of the sample space Ω_X with the lower case letter x. In many cases we will write $P_X(x_i)$, $P(X = x_i)$ or $P(x_i)$ for $P_X(X \in \{x_i\})$[1]. An RV which takes on a finite or discrete set of values is known as a *discrete random variable.* An RV whose range includes some infinite set of continuous values is known as a *continuous random variable.*

A bit of thought leads one to a conundrum regarding continuous RV's—since there are an infinite number of possible outcomes the probability of almost every outcome will be zero. This will in fact be a continuing annoyance to us as we move toward the definition of entropy. Instead of probability distributions for continuous RV's we use probability densities:

$$p_X(x_0) = \lim_{\delta \to 0} \frac{P(x_0 < X < x_0 + \delta)}{\delta} \ .$$

The probability of an event can just as easily be defined from the density by

$$P(x_{low} < X < x_{high}) = \int_{x_{low}}^{x_{high}} p_X(x)dx \ .$$

The probability density of an RV always integrates to 1,

$$\int_{-\infty}^{\infty} p_X(x)dx = P_X(-\infty < X < \infty) = 1 \ .$$

It is not true, however, that $0 \leq p_X(x) \leq 1$. Probability densities are always non-negative, but can have arbitrarily large values. Often densities can be manipulated in the same way that distributions are. In subsequent discussion we will avoid duplication whenever definitions and theorems are the same for both distributions

[1]Typically probability books say that this is only done when there is no chance for confusion. We know better.

and densities.

Simple Statistics

A random variable model of a process allows us to answer a variety of quantitative questions about the behavior of the process. Though the voltage across a resistor is unpredictable, its long term average is not. Let us define the intuitive notion of "long term average" as the *expected value* or *mean* of an RV. The expected value $E_X[X]$ is defined as:

$$E_X[X] \equiv \sum_{x_i \in \Omega_X} x_i P(X = x_i) \ , \tag{2.1}$$

or

$$E_X[X] \equiv \int x p(X = x) dx \ . \tag{2.2}$$

For notational convenience we will sometimes refer to the expectation of X as $E[X]$. The mean of a random variable is a deterministic function of its distribution. Intuitively $E[X]$ is the average of the RV's value over a large sample. We will denote a sample a, somewhat non-standardly, by an ordered collection of trials x_a,

$$a = [...x_a...] \ .$$

The size of a sample $\|a\|$ we will refer to as N_a. In a small abuse of notation we will write

$$E_a[X] \equiv \frac{1}{N_a} \sum x_a \ ,$$

for the average over the sample a. Unlike the mean, the sample mean is a random variable. The law of large numbers allows us to prove that in the limit the sample mean equals the expectation:

$$E_X[X] = \lim_{N_a \to \infty} E_a[X] = \lim_{N_a \to \infty} \frac{1}{N_a} \sum x_a \ . \tag{2.3}$$

The mean is an example of a *statistic*. Statistics are deterministic values computed from an RV that sum up its gross or long term behavior. Statistics of X are defined as the expectation of functions of X or possibly $P(X)$.

By itself, $E[X]$ does not tell us much about X. For example, the average lottery number does not help us guess what the next lottery number will be. In addition to knowing the mean, we would like to know on average how close samples of X will be to the mean. We can tell that the average lottery number is a useless statistic by the fact that the variation in lottery numbers is huge. One measure of expected variation is called *variance* and is defined as:

$$Var(x) \equiv E_X[(X - E_X[X])^2] = E_X[X^2] - E_X[X]^2 \ . \tag{2.4}$$

The square root of variance is the *standard deviation*, $\sigma(X)$. The standard deviation is a measure of how far the samples of X will be from $E[X]$.

Though the expectation of an RV is equal to the infinite average, as in (2.3), we have not explored its relation to a finite average. Is the expectation a good estimate for the mean of a sample? In a qualified sense the answer is yes. The expectation of the sample mean is the same as the expectation:

$$E[E_a[X]] = E[\frac{1}{N_a}\sum x_a] = \frac{1}{N_a}\sum E[x_a] = E[X] \ .$$

Expectation, because it is defined as an integral, is linear and can be moved inside the summation. The sample mean is often called an *unbiased* estimator of the true mean. But, how close on average will the sample mean be to the true mean? The average squared difference from the mean is:

$$E[(E_a[X] - E[X])^2] = E[(E_a[X] - E[E_a[X]])^2] = Var(E_a[X]) \ .$$

Under the assumption that the different trials of X are independent and identically distributed, the standard deviation of the sample mean is

$$\sigma(E_a[X]) = \frac{\sigma(X)}{\sqrt{N_a}} \ .$$

Therefore, the standard deviation of the sample mean approaches 0 as N_a approaches infinity. We can conclude that the sample mean is an unbiased estimate for the true mean, and that the quality of the estimate is better for larger samples.

The mean and variance are the zeroth and first elements of an infinite class of moment statistics. These statistics can used to classify the behavior of an RV with ever increasing accuracy.

The Algebra of Random Variables

Random variables are useful descriptions of processes that occur in the real world. RV's can be used in algebraic equations just as variables are. The value of an equation that includes an RV is another random process. A new RV, Y, can be defined from X as $Y = F(X)$. For discrete RV's, the probability distribution of Y is easily defined as: $P_Y(F(n)) = P_X(n)$. For continuous RV's it is not quite as simple,

$$p_Y(F(x)) = \frac{p_X(x)}{\left|\frac{dF(x)}{dx}\right|} .$$

Intuitively this equation tells us to scale down the density at points where the derivative of F is large. In regions where $\left|\frac{\partial F(x)}{\partial x}\right|$ is large F acts to stretch out X. The density $p_X(x)$ gets diluted by this stretching.

With this new theory of random variables, and many identities that can only really be hinted at here, we can begin to analyze systems such as the noisy circuit described above. We can answer questions like, "If there is random noise in the voltage from a power supply, how much variation will there be in the current across a resistor on the other side of the circuit?" In general this kind of analysis starts from a description of the distribution of one RV and derives the distribution of other functionally related RV's in the system.

Joint and Conditional Distributions

When one RV is a function of another RV, as in $Y = F(X)$, Y and X are said to be dependent. Measuring X allows us to predict Y. It is also possible that two RV's are related but not directly predictable from each other. An example is a noisy voltage source that is powering a noisy current source. Actually measuring voltage tells you something about current, but it doesn't tell you everything. There is still unpredictability that arises from the current source itself. Finally, it is possible that two RV's are completely independent. For example, two different rolls of a fair die are considered independent.

Dependency can be formalized by examining the joint distribution of two RV's, $P(X,Y)$. The joint distribution tells us about the co-occurrence of events from the RVs X and Y. It is a complete description of the random behavior of both X and Y. The joint distribution allows one to compute the *marginal distributions*,

$$P(X) = \sum_{y \in \Omega_Y} P(X, Y = y) \ .$$

Two variables are *independent* if

$$P(X,Y) = P(X) \cdot P(Y) \ .$$

They are considered dependent when the joint distribution is not the product of marginal distributions. A closely related distribution, the conditional distribution, $P(Y \mid X)$, is the probability of Y if we knew X. It is defined as:

$$P(Y \mid X) = \frac{P(X,Y)}{P(X)} \ .$$

Complete, functional dependence can be determined from conditional probability when it is the case that for all $x \in \Omega_X$ that

$$P(Y = F(x) \mid X = x) = 1 \ .$$

What is known as Bayes' Law can be concluded from the following equation:

$$P(X \mid Y) = \frac{P(X,Y)}{P(X)} \frac{P(X)}{P(Y)} = \frac{P(Y \mid X) P(X)}{P(Y)} \ .$$

Bayes' Law is quite useful in situations where one would like to conclude the distribution of X from a measurement of Y, but in principle all that is known is $P(Y \mid X)$.

2.2 Entropy

Entropy is a statistic that summarizes randomness. The definition of a random variable makes no mention of how random the variable is. Is a lottery number more or less random than the roll of a die? Entropy helps us answer this question. As we will see, the more random a variable is the more entropy it will have. The theory described in this section can be found in the excellent textbook by Cover and Thomas (Cover and Thomas, 1991).

Entropy in one form or another is a very old concept. Its origins clearly date back to the first work on thermodynamics in the last century. But a lot of the credit for defining entropy and bringing it into statistics and engineering falls to Shannon (Shannon, 1948). The most straightforward definition of entropy is as an expectation:

$$H(X) = -E_X[\log(P(X))] = -\sum_{x_i \in \Omega_X} \log(P(X = x_i))P(X = x_i) \ .$$

where we define $0 \ \log(0) = 0$ here and elsewhere in the thesis. The classical definition of entropy applies only to discrete random variables. We will present the definition of continuous entropy, known as differential entropy, later. Entropy is typically defined in terms of the logarithm base 2. In that case entropy is given in units of bits.

Entropy is Code Length

One way of measuring the randomness of a signal is to ask how long a message is required to encode one trial. For a fair coin it takes one bit of information: a 1 for heads and a 0 for tails. As long as you are sending only one trial, it takes one bit no matter what kind of coin it is[2]. This restriction does not apply to a message that describes a sample of many coin flips. If you knew that the coin in question came up tails only one time in a thousand, there are a number of straightforward schemes for encoding a sample that require less than one bit per trial. For instance one could send the length of the sample a and then the position of the zeros. It would take $\log(N_a)$ bits to send the length and $\log(N_a)$ bits to send the position of each zero. The length of a message describing N_a flips of our coin would on average be

$$E[length(a)] = \log(N_a) + P(X = 0)N_a \log(N_a) \ .$$

In this coding scheme a message describing one thousand trials will on average take about 20 bits. The number of bits it takes to encode a sample is dependent both on the number of events and the distribution of the random variable.

[2]Provided it doesn't have two heads.

A discussion and comparison of coding schemes could take up quite a lot of space. Luckily, using the Kraft inequality it can be proven that on average one needs to send $H(X)$ bits to communicate a trial of the random variable X. Furthermore, Shannon showed that is possible to construct a code that will take at most $H(X) + 1$ bits on average. A simple algorithm discovered by Huffman can construct the shortest possible codes for any random variable. Because entropy is a bound on the code length that is required to transmit a trial, entropy is often called *information.*

Conditional Entropy, Joint Entropy, and Mutual Information

The concept of mutual information plays a critical role in this thesis. One of the key problems that we will need to solve is, "How likely is it that the random variable Y is functionally dependent on X?" Functional dependence is related to dependence. In Section 2.1 we defined the notion of independence and dependence, but were unable to say more. Entropy allows us to quantify the notion of dependence.

Quantifying dependence is very much like quantifying randomness. Total dependence implies that a measurement of one RV completely determines the other (i.e. knowing X removes any unpredictability from Y). Independence is just the opposite, knowing X does not help you predict Y. Just as joint and conditional distributions relate the co-occurrences of two RV's, entropy can be used to relate the predictability of two RV's. Conditional entropy and joint entropy are defined as:

$$H(Y \mid X) \equiv E_X[E_Y[\log(P(Y \mid X))]]$$

and

$$H(Y, X) \equiv E_X[E_Y[\log(P(Y, X))]] \ .$$

Conditional entropy is a measure of the randomness of Y given that you know X. Note that it is an expectation over the different events of X, so it measures on average how much randomness there would be in Y given X. $H(Y \mid X = x)$ is the randomness one expects from Y if X takes on a particular value x. Random variables are considered independent when

$$H(Y \mid X) = H(Y)$$

or,

$$H(X,Y) = H(X) + H(Y) \ .$$

As Y becomes more and more dependent on X, $H(Y \mid X)$ gets smaller. Conditional entropy by itself is not a measure of dependency. A small value for $H(Y|X)$ may not imply dependence, it may only imply that $H(Y)$ is small. The *mutual information*, MI, between two random variables is given by

$$I(X,Y) = H(Y) - H(Y \mid X) \ . \tag{2.5}$$

$I(X,Y)$ is a measure of the reduction in the entropy of Y given X.

A number of simple logarithm equalities can be used to prove relations between conditional and joint entropy. For instance, conditional entropy can be expressed in terms of marginal and joint entropies:

$$H(Y \mid X) = H(X,Y) - H(X) \ .$$

This allows us to provide three equivalent expression for mutual information (and a useful identity):

$$\begin{aligned} I(X,Y) &= H(Y) - H(Y \mid X) && (2.6)\\ &= H(X) + H(Y) - H(X,Y) && (2.7)\\ &= H(X) - H(X \mid Y) && (2.8)\\ &= I(Y,X) \ . && (2.9) \end{aligned}$$

An extremely useful inequality on expectations, known as Jensen's inequality, allows us to prove that for any concave function F that

$$E[F(X))] \leq F(E[X]) \ .$$

A function is concave when its second derivative is negative everywhere. Using the fact that the logarithm function is concave, Jensen's inequality allows us to prove the following useful inequalities:

$$H(X) \geq 0 \tag{2.10}$$

$$H(Y) \geq H(Y \mid X) \tag{2.11}$$
$$I(X,Y) \geq 0 \tag{2.12}$$

2.2.1 Differential Entropy

While a number of the main theorems of entropy apply both to continuous and discrete distributions, a number of other theorems change significantly. The continuous version of entropy is called differential entropy, and is defined as:

$$h(X) \equiv -E_X[\log(p(X))] = -\int_{-\infty}^{\infty} p(x)\log(p(x))dx \ . \tag{2.13}$$

For the most part differential entropies can be manipulated, and obey the same identities, as entropy. In fact all of the equalities and inequalities of the previous section hold except for (2.10). Throughout the thesis, when entropy is mentioned, it is to be understood as the applicable form of entropy. When the difference matters we will be explicit.

The most perplexing difference between entropy and differential entropy is that there is no longer a direct relationship between $h(X)$ and code length. It is possible to construct examples where differential entropy is negative. The is an implication of the fact that $p(X)$ can take on values greater than 1. Code length, however, is never negative.

Differential entropy does not provide an absolute measure of randomness. Discouragingly, it is even the case that a density with a differential entropy of negative infinity may still be unpredictable. Examples of this sort can be constructed by embedding a discrete process into a continuous space. For example one could model the roll of a die as a continuous RV. The density would then be made up of a series of delta functions centered at the points one through six. A delta function, often called a Dirac delta function, can be defined from a box car function,

$$b(x, x_{low}, x_{high}) = \begin{cases} \frac{1}{x_{high} - x_{low}} & \text{if } x_{low} < x < x_{high} \\ 0 & \text{otherwise} \end{cases} \ . \tag{2.14}$$

The box car function is defined so that it integrates to one. The delta function is a box car function in the limit as it approaches zero width,

$$\delta(x) = \lim_{\Delta \to 0} b(x, 0, \Delta) \ . \tag{2.15}$$

The delta function, because it is a box car, integrates to one. It can be shown that

$$f(x_0) = \int_{-\infty}^{\infty} \delta(x_0 - x) f(x) dx \tag{2.16}$$

Furthermore, from the definition of convolution,

$$(f * g)(x) \equiv \int_{-\infty}^{\infty} f(x - x') g(x') dx' \tag{2.17}$$

we can see that the delta function is the identity operator,

$$(\delta * f)(x) = \int_{-\infty}^{\infty} \delta(x - x') f(x') dx' = f(x) \ . \tag{2.18}$$

The density of the continuous model of die rolling can be formulated as

$$p(x) = \sum_{i=1}^{6} \frac{1}{6} \delta(x - i) \tag{2.19}$$

which will integrate to 1. Furthermore if we define the probability of an event as

$$P(x) = \lim_{\Delta \to 0} \int_{x-\Delta}^{x+\Delta} p(x) \ , \tag{2.20}$$

then $P(X = 1) = \frac{1}{6}$, as will the probability of the other events. Finally we can show that the entropy of X is negative infinity,

$$h(X) = -\int_{-\infty}^{\infty} p(x') \log(p(x')) dx' = -\sum_{i=1}^{6} \frac{1}{6} \log(\infty) = -\infty \ , \tag{2.21}$$

yet X is pretty clearly random.

Though differential entropy does not provide an absolute measure of randomness or code length, it does provide a relative measure of these properties. A random variable X is less predictable than Y whenever $h(X) > h(Y)$. Similarly an event from X requires more bits on average to encode than an event from Y.

2.3 Samples versus Distributions

A random variable is a mathematical structure that is intended to model the behavior of a physical process. In some cases there are excellent physical reasons to believe that an RV is an accurate model of a process. In many other cases the properties of a random physical process may well be unknown. In these cases we may still wish to use the theory of probability to analyze a system. The first step is to find an accurate RV model of our data.

In order to insure that our probabilistic inferences will be correct, our model must be as accurate as possible. While it is possible to model every coin as a fair coin, we do so at our peril. It makes much more sense to perform a large number of experiments intended to test the hypothesis, "Is the coin fair?" There are two important intuitions behind finding an accurate model of a random process. First and foremost, you want a model that seems to explain the data well. It might not make sense to guess that a coin is fair if after 500 flips heads has come up 400 times. But it is also important that your model be plausible. If after a lifetime of experience you realize that most coins are pretty fair, than perhaps it makes more sense to assume that 400 heads is unusual but not sufficiently unusual to assume that the coin is biased.

2.3.1 Model Selection; Likelihood and Cross Entropy

The field of statistics provides many tools for testing the validity of random models. A lot of this work shares a particular form, called *maximum likelihood* model selection. The goal is to select the most probable model given a large sample of measurements. Maximum likelihood selection proceeds in steps: (1) guess the definition of a random variable that might model the process; (2) evaluate the "goodness" of the model by computing the probability that the data observed could have been generated by the model, (3) after evaluating many models retain the model that makes the data most probable. The probability of a sample a under the RV X is the probability of the co-occurrence of the trials in a,

$$\ell(a) = P_X(a) = P_X(x_1 = x_{a_1}, x_2 = x_{a_2}, \ldots) \ . \tag{2.22}$$

The probability of a sample is usually called its *likelihood* and is denoted $\ell(a)$.

Justification for maximum likelihood model selection is based on Bayes' law. The likelihood of a sample is really a conditional probability, $P(a \mid X)$. Bayes' law allows us to turn the conditional around and find the most likely model given the sample,

$$P(X \mid a) = P(a \mid X)\frac{P(X)}{P(a)} \; . \tag{2.23}$$

In order to compute the model likelihood one must multiply the sample likelihood by the correcting factor $\frac{P(X)}{P(a)}$. The unconditioned probability of the sample $P(a)$ could well be arbitrary, because the sample is the same for all of the models we will evaluate. The prior probability of the model, $P(X)$, poses more problems. Maximum likelihood model selection assumes that all of the models that are to be evaluated are equally likely to have occured, i.e. $P(X)$ is constant. As a result $\frac{P(X)}{P(a)}$ is the same for all models, and the most probable model is the one that makes the data most probable.

When reliable information about the prior probability of a model is available we can use Bayes' law directly. This technique is known as *maximum a posteriori* model selection. For instance, over a wide variety of experiments, we may have observed that fair coins are far more common than unfair coins. It is very implausible that any particular coin would be unfair, but not impossible. While our prior knowledge should bias us toward the conclusion that a new coin is fair, it does not determine our conclusion. The likelihood of the model together with the prior model determine which model has the highest probability of explaining the data. We want a model that both explains the data well and is plausible.

In general, evaluating joint probability over a large number of random variables is intractable. In practice most maximum likelihood schemes assume that the different trials of X are independent. The probability of co-occurrence is then the product of the independent RVs,

$$\ell(a) = \prod_{x_a \in a} P_X(x_a) \; .$$

Maximizing $\ell(a)$ is still a daunting process. Significant simplification can be obtained by maximizing the logarithm of ℓ,

$$\log \ell(a) = \sum_{x_a \in a} \log P_X(x_a) \ .$$

Log likelihood has the same maximum as $\ell(a)$, but has a much simpler derivative.

There is an interesting parallel between log likelihood and entropy. Recall that entropy is a statistic of X. The finite sample average of entropy, or empirical entropy, which will figure strongly later in the thesis, is

$$h_a(X) = -E_a[\log P_X(X)] = -\frac{1}{N_a} \sum_{x_a \in a} \log P_X(x_a) \ . \tag{2.24}$$

We can therefore conclude that

$$h_a(X) = -\frac{1}{N_a} \log(\ell(a)) \ .$$

This provides us with an interpretation of model selection in terms of entropy. Instead of finding the model that makes the data most likely, we could instead find the model that has the lowest empirical entropy. Conversely, we could present a new interpretation of entropy: a distribution has low entropy if the probability of a sample drawn from that distribution is high, and a distribution has high entropy if the sample has low likelihood. A density with a very narrow peak has low entropy because most of the samples will fall in the region where the density is large. A very broad, low density has high entropy because the samples are spread out and fall where the density is low.

The close relationship between entropy and log likelihood is well known, but is often overlooked by students of probability. As result the parallels between research on entropy and maximum likelihood can be easily missed. The fact that a system manipulates "entropy" does not make it necessarily any better, or different, than one based on likelihood. For instance, the log likelihood of a density can be derived from cross entropy. The cross entropy $D(p_X \| p_{\hat{X}})$, or asymmetric divergence is a measure of the difference between two distributions:

$$D(p_X \parallel p_{\hat{X}}) = E_X \left[\log(\frac{p_X(X)}{p_{\hat{X}}(X)}) \right] \tag{2.25}$$

$$= \int_{-\infty}^{\infty} \log\left(\frac{p_X(x')}{p_{\tilde{X}}(x')}\right) p_X(x')dx \tag{2.26}$$

$$= \int_{-\infty}^{\infty} \log(p_X(x'))p_X(x')dx - \int_{-\infty}^{\infty} \log(p_{\tilde{X}}(x'))p_X(x')dx \tag{2.27}$$

$$= -h(X) - E_X[\log(p_{\tilde{X}}(X))] \tag{2.28}$$

$$\approx -h(X) - E_a[\log(p_{\tilde{X}}(X))] \ . \tag{2.29}$$

Cross entropy is non-negative, reaching zero if and only if p_X and $p_{\tilde{X}}$ are identical. Where maximum likelihood model selection searches for the model that makes the sample most likely, cross entropy selection searches for the model that is closest, in the cross entropy sense, to the true distribution. If we use the approximation in (2.29), the two procedures are in fact identical. In (2.29) $h(X)$ is a constant and does not play a role in the selection. The second term is the log likelihood of a sample drawn from X under the density $p(\tilde{x})$.

Though the maximum likelihood procedure might appear effective, it is important to point out that the space of possible models is infinite. Any naive search would last forever. In practice the search is often restricted to a parameterized class of possible models, where there are between 1 and 50 parameters. The parameter space is then searched non-exhaustively with any of a number of greedy search procedures such as gradient descent.

2.4 Modeling Densities

In this section we will describe a number of techniques for estimating densities from data. Understanding the process by which this is done is an important prerequisite for understanding the main algorithms in this thesis. We will begin with a discussion of the most widely observed continuous density: the Gaussian density[3]. We will then derive an closed form expression for the most likely Gaussian given a sample. This section will also include a discussion of other parametric density functions and finally a non-parametric technique for estimating densities known as Parzen window density estimation.

[3]For the sake of brevity we will sometimes refer to a Gaussian density as simply a Gaussian.

2.4.1 The Gaussian Density

The most ubiquitous of all random processes is the Gaussian or normal density. It literally appears everywhere. The most common justification arises from the "central limit theorem", which shows that the density of the sum of a very large number of independent random variables will tend toward Gaussian. An equally important justification is that the mathematics of the Gaussian density are quite simple because it is an exponential. Moreover, since any linear function of a Gaussian density is itself Gaussian, they are widely used in linear systems theory. It is almost certainly the case that the majority of continuous random processes have been modeled as Gaussians; whether they are or not. A *Gaussian density* is defined as:

$$g_\psi(x-\mu) \equiv \frac{1}{\sqrt{2\pi\psi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\psi}} \quad . \tag{2.30}$$

The parameters ψ and μ are the variance and mean of the density. One can demonstrate this with some clever integration.

The Gaussian density can also be defined in higher dimensions:

$$g_\psi(x-\mu) \equiv \frac{1}{(2\pi)^{\frac{n}{2}}|\psi|^{\frac{1}{2}}} \exp(-\frac{1}{2}(x-\mu)^T\psi^{-1}(x-\mu)) \quad . \tag{2.31}$$

In a d dimensional space, the mean μ is a d-vector. The variance is replaced by a covariance matrix, a d-by-d matrix ($|\psi|$ is the determinant of ψ). Recall that variance is defined as the expected square of the difference from the mean; covariance is somewhat more complex:

$$\psi_{ij} = E[(X_i - \overline{X_i})(X_j - \overline{X_j})] \quad , \tag{2.32}$$

here X_i is the i'th component of the RV X. The diagonal entries of ψ contain the variances of the components. The off-diagonal entries measure the expected co-variation.

Equation (2.30) defines an infinite family of density functions. Any one member of this family is determined by its mean and its variance. Before we can attempt to model an unknown density with a density from this family, we must first decide if the density is Gaussian. Maximum likelihood model selection can then be used to estimate μ and ψ from a sample. The form of the Gaussian density makes finding the maximum likelihood parameters very easy. The log likelihood of a sample of a Gaussian density is

$$\log(\ell(a)) = \sum_{x_a \in a} \log(P_X(x_a)) \tag{2.33}$$

$$= \sum_{x_a \in a} \log(g_\psi(x_a - \mu)) \tag{2.34}$$

$$= \sum_{x_a \in a} \log(\frac{1}{\sqrt{2\pi}}\psi) - \frac{1}{2}\frac{(x_a - \mu)^2}{\psi} \ . \tag{2.35}$$

The most likely μ minimizes

$$\sum_{x_a \in a} (x_a - \mu)^2 \ ,$$

a quadratic function of μ. Differentiating and solving for zeroes we find that the most likely μ is

$$\mu = \frac{1}{N_a} \sum_{x_a \in a} x_a \ .$$

This is a very satisfying result. We have proven that the most likely estimate for the mean, μ, is the mean of the sample. A very similar argument can be used to prove that the maximum likelihood estimate for the variance, ψ, is the sample variance:

$$\psi = \frac{1}{N_a} \sum_{x_a \in a} (x_a - \mu)^2 \ .$$

Figure 2.1 displays a 100 point sample drawn from a Gaussian density. The true density is shown together with the most likely model. Because the sample mean and sample variance are not perfect measures of the true mean and variance, the most likely model is not perfect. The accuracy of the estimated mean and variance gets better as the sample size increases. Even for a sample of 100 points there is significant variability in the estimated model for different samples. Figure 2.2 shows ten different estimates from ten different samples of the same density.

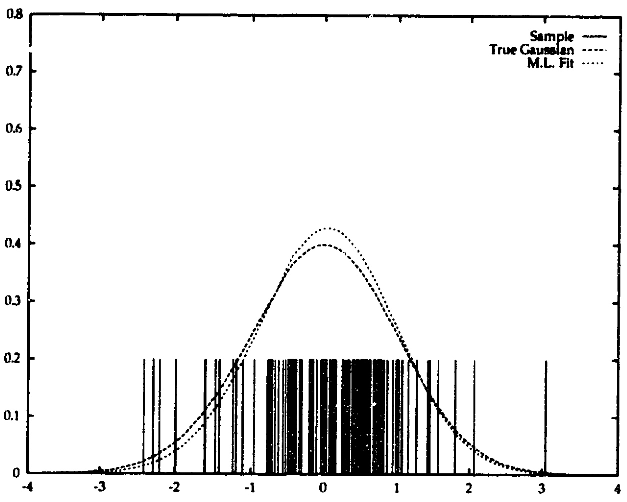


Figure 2.1: Three plots of a sample drawn from a Gaussian density with a mean of 0.0 and a variance 1.0. The first is a sample of 100 points drawn from the density. Each point is is represented a vertical black line. The second plot is the density of the true Gaussian. The third plot is the density of the Gaussian estimated from the sample (mean = 0.045, variance = 0.869).

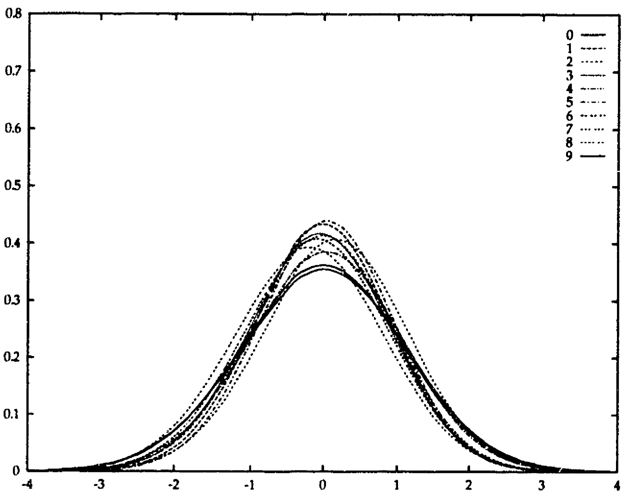


Figure 2.2: The maximum likelihood density estimates for ten different samples of 100 points drawn from the same Gaussian.

2.4.2 Other Parametric Densities

Finding the most likely Gaussian model for a sample is a very efficient operation. The mean and the variance are trivially computable in linear time. Efficient estimation is a property shared by all of the exponential densities, a class of densities which include the Gaussian density, the Gamma density, and others. For all other types of densities it is not possible to find maximum likelihood parameter estimates directly from statistics of the density. The most likely set of parameters must be determined by a search process. Since there are an infinite number of possible parameter values, finding values for these parameters that are optimal is not straightforward. Generally problems of this sort are solved using an iterative refinement process known as gradient descent. The gradient descent procedure is described in Appendix A.1.

The Gaussian density has many advantages. Why not use it to model every sample? The simple answer is that not all real densities are Gaussian. In fact, many real densities are far from Gaussian. One of the strongest limitations of the Gaussian, and of all the exponential densities, is that they are unimodal (they have a single peak). Modeling densities that have multiple peaks as if they had a single peak is foolhardy. Figure 2.3 shows an attempt to fit a two peaked function with a single Gaussian. In many situations it may seem as though the simplicity and efficiency that arises from using a Gaussian density outweigh the added accuracy that arises from using a more accurate model. As we will see, this is a temptation to which many have succumbed.

Once the decision to use a more complex model has been made, the set of possible model densities is literally infinite. In terms of accuracy this is an unambiguous advantage. Density can be modeled by any function that can be guaranteed to integrate to one. The most common model after the simple Gaussian is a mixture of Gaussians:

$$M(x, W) = \sum_{i=1}^{N} c_i \; g_{\psi_i}(x - \mu_i) \ , \tag{2.36}$$

here W represents the collection of parameters $(N, \{\mu_i\}, \{\psi_i\}, \{c_i\})$. When $\sum c_i = 1$, the mixture model is guaranteed to integrate to one. A mixture density need not be uni-modal; it may have as many as N peaks. Figure 2.3 contains a graph of a mixture of Gaussian density with two equal components. Given a large number

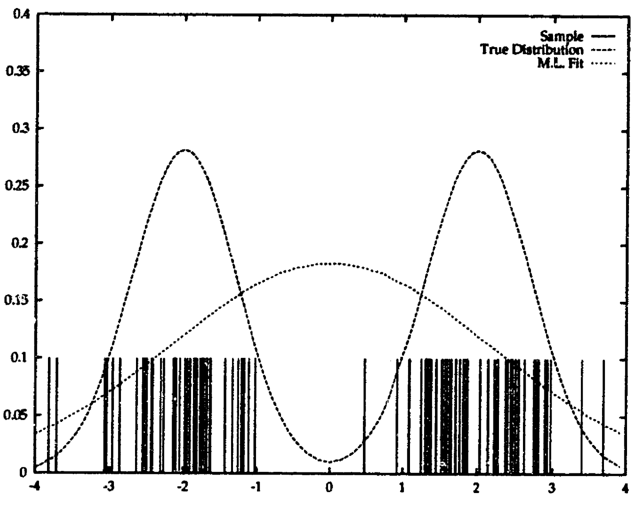


Figure 2.3: Three plots of a sample drawn from a combination of two Gaussians. The Gaussians have variance 0.3 and means of 2.0 and −2.0 respectively. The sample has 100 points. The maximum likelihood Gaussian has mean 0.213 and variance 3.824.

of Gaussians, almost any density can be modeled accurately. As before, maximum likelihood can be used to select the best set of parameters for a given sample a. It is possible to search for the correct parameter vector using gradient ascent, but for Gaussian mixture models there is a more efficient technique known as Expectation Maximization (A. P. Dempster and Rubin, 1977). In either case finding the best parameter vector can involve a lengthy search process.

While mixture models are fairly popular, almost any parameterized function can be used as a density estimate. Within the neural networks literature some have trained back-prop neural networks to approximate densities (Jacobs et al., 1991) (see also (Haykin, 1994) for an excellent review of neural network research). There is nothing terribly special about using a neural network for this purpose. It is just another form of parametric density estimation.

Now that we have some feel for the density estimation process, it is critical that important limitations be pointed out. Whenever one estimates a density from a sample a very important first step is required: assumptions must be made about the form of the density. The space of possible functions is so large that for any sample there are an infinite number of density functions that fit it equally well. Continuous density can defy intuition. For instance it is always possible to define a density that makes a given sample infinitely likely. Take for example a density function made up of delta functions. As we did in section 2.2 we could make up a function with a delta function for each trial:

$$p(X) = \frac{1}{N_a} \sum_{x_a \in a} \delta(x - x_a) \ . \tag{2.37}$$

The likelihood of this model density is then ∞, and is guaranteed to be bigger than any other density's likelihood. Wouldn't this imply that fitting any other density is sub-optimal? While intuition argues against such an artificial density, there is no principled scheme for dealing with this dilemma.

Much has been written about this problem in the machine learning literature, where it is called function approximation. There simply is not enough information in a finite sample to uniquely determine which of the infinite number of possible functions fit it. The only solution is to make strong assumptions about the function: for example that it is Gaussian, that it is polynomial, or that it is smooth. These assumptions provide a strong prior probability over the space of possible functions. Together the likelihood and the prior can often uniquely determine a solution.

Maximum likelihood model selection is not guaranteed to do a good job of density estimation. There are three reasons why the most likely model may fail to be an accurate model. The first reason is that the set of evaluated models may not contain the correct model. This is called *inductive inadequacy* and it arises when the underlying assumptions about the density are wrong. The second reason is that maximum likelihood can be fooled. An especially unlikely sample, as in our example of the unbiased coin, can lead to a model estimate that is not correct. This is a question of confidence. A larger sample is less likely to be unusual, and gives us more confidence in our model. The third reason is that the search through parameter space may fail. A good solution may exist but cannot be found, for example if there are local minima.

2.4.3 Parzen Window Density Estimation

The final class of density functions we will discuss are called non-parametric density estimators. For these models no search for parameters is needed. While parametric methods use the parameters as the model, non-parametric models use the sample

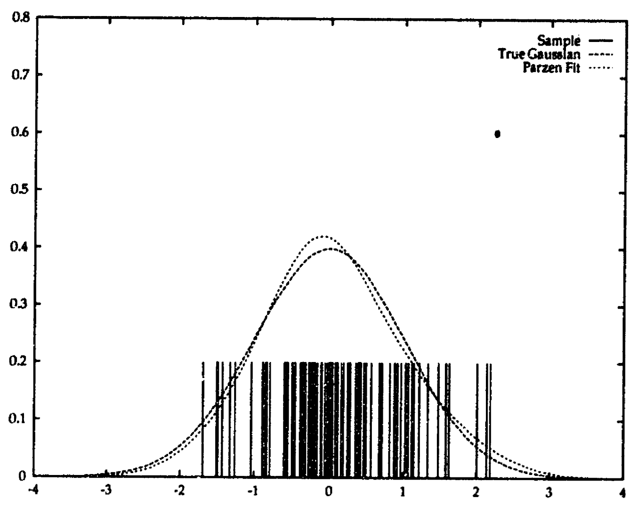


Figure 2.4: Three plots of a sample drawn from a Gaussian density with a mean of 0.0 and a variance 1.0. The first is a sample of 100 points drawn from the density. Each point is is represented a vertical black line. The second plot is the density of the true Gaussian. The third plot is the Parzen density estimate constructed with Gaussians of variance 0.35.

to directly define the model.

The non-parametric scheme on which we will focus is known as Parzen window density estimation. The general form of the density is:

$$P^*(x,a) \equiv \frac{1}{N_a} \sum_{x_a \in a} R(x - x_a) \ , \tag{2.38}$$

where a is a sample and R is a valid density function. The function R is often called the smoothing or window function. The choice of R is important to the quality of the approximation. Different functions will lead to very different density estimates. The Gaussian density is a common selection for R, making the density estimate a mixture of Gaussians. The Parzen estimate has a Gaussian centered at each sample, whose mean is given by that sample. Figure 2.4 contains a graph of a density, a sample, and the Parzen estimate. Figure 2.5 contains a graph of ten different Parzen estimates from ten different samples. The different Parzen estimates do not show significantly more variation than the Gaussian estimates shown in Figure 2.2.

The Parzen window estimator works by making local density estimates. Evaluating the Parzen density estimate at a point x involves computing the weighted

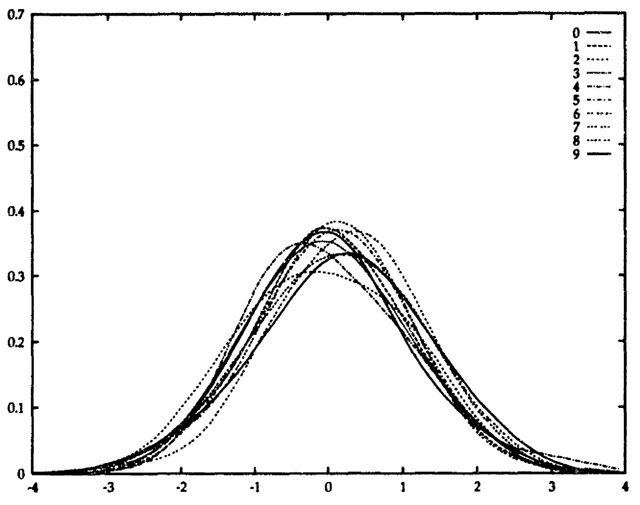


Figure 2.5: The Parzen density estimates for ten different samples of 100 points drawn from the same Gaussian density.

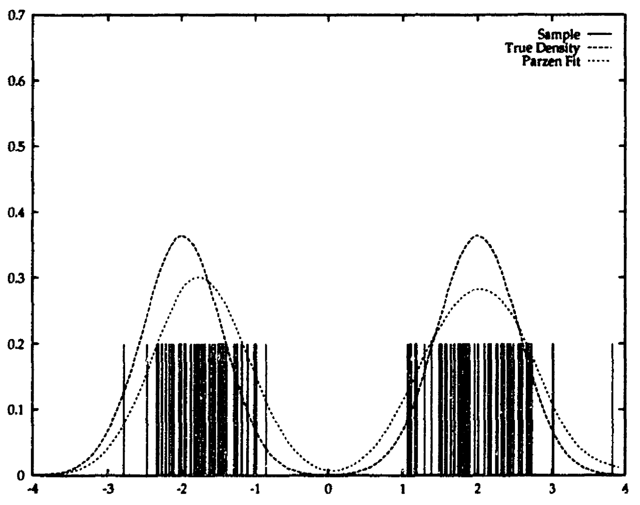


Figure 2.6: Three plots of a sample drawn from a combination of two Gaussians. The Gaussians have variance 0.3 and means of 2.0 and −2.0 respectively. The sample has 100 points. The Parzen estimate is constructed with Gaussians of variance 0.20.

sum over the sample. The weighting is determined by the smoothing function. The most common smoothing functions are monotonically decreasing, and are strongly peaked near the origin and fall off quickly to zero. In effect, the smoothing function defines a window centered on x in which samples points contribute to the density estimate. Points that fall outside of this window do not contribute. The density estimate of x is the ratio of the number of weighted sample points within the window divided by the total number of sample points. Getting a reliable estimate of this ratio involves having a reasonable number of points fall into the window of support of the smoothing function. The variance of the density estimate goes down as the number of sample points available increases.

The balance of computation required by Parzen window density estimation is qualitatively very different from parametric schemes. Constructing a parametric model involves a lengthy search through parameter space that takes more time for larger samples. Constructing a Parzen model is cheap. One need only memorize the sample. Evaluating a parametric model is usually efficient. Once the parameters are known the number of operations required is usually very small and does not grow with the size of the sample. Evaluating $P^*(x, a)$ is more expensive; requiring time proportional to the size of the sample. The overall computational complexity of either technique is a function of how they are used.

The Parzen estimate is not the maximum likelihood mixture model. The maximum likelihood model would not necessarily place the Gaussians at the sample points. There is however an asymptotic proof of Parzen convergence that relies on the law of large numbers. The Parzen estimate can be written as a sample mean:

$$P^*(x_0, a) = \frac{1}{N_a} \sum_{x_a \in a} R(x_0 - x_a) = E_a[R(x_0 - X)] \ .$$

In the limit this equals the true expectation which in turn is a convolution

$$\begin{aligned} \lim_{N_a \to \infty} P^*(x, a) &= E[R(x - X)] && (2.39) \\ &= \int_{-\infty}^{\infty} R(x - x')p(x')dx' && (2.40) \\ &= (R * p)(x) \ . && (2.41) \end{aligned}$$

So $P^*(x, a)$ converges to $p(x)$ if and only if $p(x) = (R * p)(x)$. There are two distinct conditions under which equality holds. The first is that R tends toward the delta function when the sample size approaches infinity. The second occurs when convolution by R does not change $p(x)$. In theory this could be achieved when $p(x)$ has bounded frequency content and R is a perfect low pass filter. In practice approximate equality holds whenever $p(x)$ has low frequency content and R is primarily a low pass filter, for example when $p(x)$ is a smooth function and R is a Gaussian. Furthermore, whenever equality holds $P^*(x, a)$ is an unbiased estimator of $p(x)$.

There are other conditions under which the Parzen estimate will converge to the correct estimate. This proof assumes that the samples are corrupted by measurement noise of a known density. Instead of X a corrupted random variable, $\widetilde{X} = X + \eta$, is observed. If η were known the probability of X would be,

$$p(X = x|\widetilde{X} = \tilde{x}, \eta) = \delta(x - \tilde{x} - \eta) \ .$$

Without knowledge of η we must integrate over all its possible values,

$$p(X = x|\widetilde{X} = \tilde{x}) = \int_{-\infty}^{\infty} p(X = x|\widetilde{X} = \tilde{x}, \eta')p_\eta(\eta')d\eta' \tag{2.42}$$

$$= \int_{-\infty}^{\infty} \delta(x - \tilde{x} - \eta)p_\eta(\eta)d\eta' \tag{2.43}$$

$$= p_\eta(x - \tilde{x}) \ . \tag{2.44}$$

To compute $p(X)$ we must integrate over $\widetilde{X}$. We show that the integral is approximated by the Parzen estimator,

$$p(X = x) = \int_{-\infty}^{\infty} p(X = x|\widetilde{X} = \widetilde{X}')p_{\widetilde{X}}(\widetilde{X}')d\widetilde{X}' \tag{2.45}$$

$$= \int_{-\infty}^{\infty} p_\eta(x - \widetilde{X}')p_{\widetilde{X}}(\widetilde{X}')d\widetilde{X}' \tag{2.46}$$

$$= E_{\widetilde{X}}[p_\eta(x - \widetilde{X})] \tag{2.47}$$

$$\approx E_a[p_\eta(x - \tilde{x}_a)] \tag{2.48}$$

$$= \frac{1}{N_a} \sum_{\tilde{x}_a \in a} p_\eta(x_0 - \tilde{x}_a) \ , \tag{2.49}$$

where a is a sample of $\widetilde{X}$. The probability of the uncorrupted random variable X is approximated by the Parzen estimate constructed from the samples of $\widetilde{X}$ where the smoothing function is the density function of the noise. The probability of the

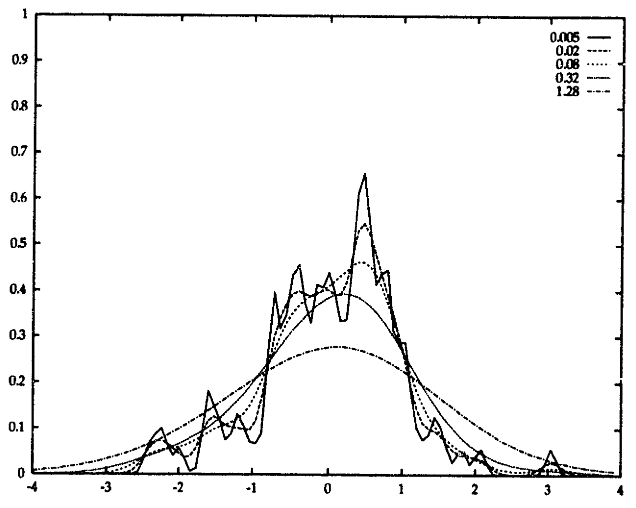


Figure 2.7: Five plots of the Parzen density estimates derived from a 100 point sample of a Gaussian. The Gaussian has variance 1.0 and mean 0.0. The different estimates use a different value for the variance of the component smoothing functions. The variances used range over a factor of 256, from 0.005 to 1.28.

corrupted random variable can be derived from a very similar argument,

$$p(\widetilde{X} = \tilde{x}) \approx \frac{1}{N_a} \sum_{\tilde{x}_a \in a} (p_\eta * p_\eta)(x_0 - \tilde{x}_a) \ .$$

The probability of a noise corrupted random variable $\widetilde{X}$ is approximated by the Parzen estimate using the smoothing function $(p_\eta * p_\eta)(x)$. This result is independent of the density of X. Often η is Gaussian noise, a very common assumption that we will return to in our discussions of entropy. The smoothing function is then a Gaussian density that has twice the standard deviation of η.

Finding the Best Smoothing Functions

As we have seen, when a priori information about the density is available Parzen estimation will converge to the correct density. Moreover, when we know either that the density is smooth or that it has been perturbed by noise it is possible to find the correct smoothing function. In the absence of a priori information, the quality of the Parzen estimate is dependent on the variance ψ of the smoothing functions. Figures 2.7 and 2.8 display the dependence of the density estimate on ψ. Each shows the Parzen estimates computed from a 100 point sample as ψ is

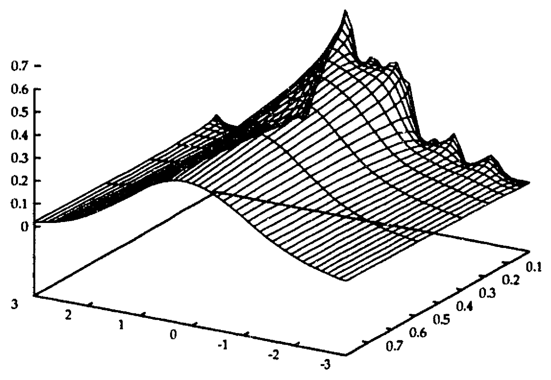


Figure 2.8: A parametric surface plot of the Parzen density versus variance. The horizontal and vertical axes are the space versus density. Variance changes with depth in the graph. Here variance changes over a factor of 80, from 0.80 to 0.01.

changed. When the variance of the smoothing function is small, less that 0.1, the density is very dependent smoothing variance. As the smoothing variance grows larger this dependence is reduced.

Maximum likelihood, or cross entropy, can be used to select the variance of the smoothing functions. Much in the same way that cross entropy can be used to find the parameters of a Gaussian to fit a sample, cross entropy can be used to find the variance of the Gaussians that make up the Parzen estimate. Since we wish to preserve the simplicity of the Parzen estimate, only a single variance will be estimated. The same variance will be used for each of the smoothing functions. Figure 2.9 graphs the cross entropy of the sample versus variance (recall that sample cross entropy is $\frac{-1}{N_a}$ times log likelihood). The sample used in this graph is the same used to estimate the densities in the above two graphs. The broad minimum around 0.25 matches the intuition gained from Figure 2.8. Both graphs tell us that the estimate is not terribly sensitive to changes in variance up to a factor of 10.

The true entropy of a Gaussian with variance 1.0 is 1.419. The optimal Parzen density estimate has a sample entropy of 1.47. This is not coincidence. We will show in the next chapter that empirical entropy is an effective method for estimating entropy.

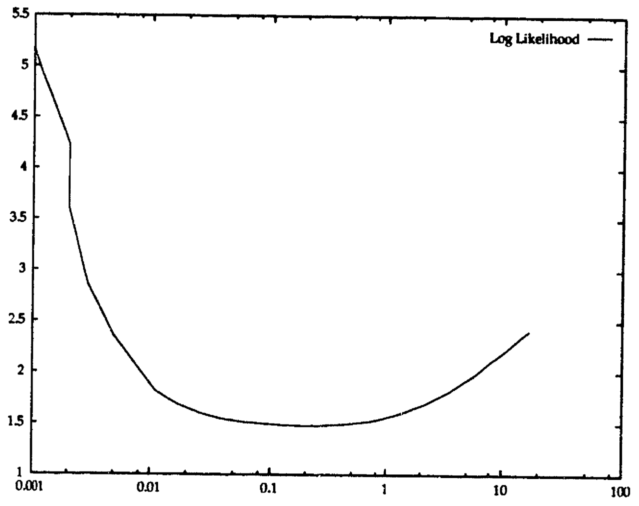


Figure 2.9: A log plot of cross entropy versus ψ. Near the minimum, the cross entropy is not terribly sensitive to the ψ. Values within a factor of 10 are all about the same.

There is a small technical note that should not be overlooked. We must be careful whenever the same sample that we use to construct the Parzen estimate is used to estimate cross entropy. Recall that the most likely, or lowest cross entropy, density estimate for a sample is a collection of delta functions centered at each point from the sample (see 2.2.1). We also know that this delta function density will have a cross entropy of negative infinity. The Parzen density is very similar in form to the delta function density. It too centers a function at each point from the sample. In the limit as the variance of the smoothing functions tends towards zero, each smoothing function approximates a delta function. Therefore the minimum cross entropy should be obtained when the variance is zero. This difficulty only arises when the sample from which the density is estimated is the same as the sample with which the cross entropy is estimated. If these two samples are different, and the density is not degenerate in some way, then no point should appear in both samples. As the variance of the smoothing functions tends to zero the density at all points that are not in the Parzen sample tends to zero. As a result the cross entropy should tend toward positive infinity as the variance tends to zero. This effectively precludes the solution where the variance of the smoothing functions is zero.

We can simulate having two different samples by a process called *cross-validation*. Cross-validation splits a single sample a into two samples. One sample has a single point $\{x\}$ and the second contains the remaining points $a - \{x\}$. There are N_a

different ways to split the sample in two parts. Rather than draw two different samples, we use the N_a different split samples. In each case the larger sample, of size $N_a - 1$, is used for the Parzen estimate, and the smaller sample is used to estimate the entropy. Estimating log likelihood or empirical entropy with two samples a and b yields

$$\log(\ell(b)) = -N_b h_b(X) = -N_b E_b[\log P^*(X, a)] \ , \tag{2.50}$$

versus

$$-N_a E_a[\log(P^*(x_a, a - \{x_a\}))] \ , \tag{2.51}$$

using cross validation. The cross validated cross entropy is an unbiased estimate of the two sample cross entropy.

The Quality of the Parzen Estimate

One way to evaluate the quality of the Parzen estimate is to evaluate the standard deviation of its estimate. Another, perhaps more useful statistic is the standard deviation normalized by the mean

$$\frac{\sigma(X)}{E[X]} \ . \tag{2.52}$$

The normalized standard deviation measures expected deviation from the mean as a function of the overall scale. When the mean is large, small deviations are not usually important. When the mean is very small, a small deviation can make a big difference. Normalized standard deviation is a good measure to use when the log of a variable is important (like log likelihood and entropy). Using the constant and linear terms of the Taylor expansion of logarithm and assuming that the standard deviation of X is small,

$$\sigma(\log(X)) \approx \frac{\sigma(X)}{E[X]} \ . \tag{2.53}$$

The standard deviation of the Parzen density estimate at a point x is a function of the total number of sample points used to estimate the density. The normalized standard deviation of a Parzen estimate is

$$\frac{\sigma(P^*(x,a))}{E[P^*(x,a)]} = \frac{\sigma(P^*(x,a))}{p(X)} \ , \tag{2.54}$$

where both the standard deviation and the expectation are taken over the space of possible samples. The two equations are equal whenever $P^*(x,a)$ is an unbiased estimator for $p(X)$.

The standard deviation of the Parzen estimate can be computed exactly when the smoothing functions are box car functions. The Parzen estimate is then the number of sample points that fall into the box car window divided by the total number of sample points,

$$P^*(x,a) \equiv \frac{1}{N_a} \sum_{x_a \in a} R(x - x_a) = \frac{kN_{in}}{N_a} \ . \tag{2.55}$$

The proportionality constant $k = R(0)$ turns the number of points that fall into a window into a density. The standard deviation is then

$$\sigma(P^*(x,a)) = \frac{k}{N_a} \sigma(N_{in}) \ . \tag{2.56}$$

The standard deviation of N_{in} is the standard deviation in the number of points that will fall into the window around x. Assuming each point in the sample is independent then

$$\sigma(P^*(x,a)) = \frac{k}{N_a} \sqrt{N_a P^*(x,a) \left(1 - \frac{P^*(x,a)}{k}\right)} \ . \tag{2.57}$$

The normalized standard deviation of the Parzen estimate is then

$$\frac{\sigma(P^*(x,a))}{E[P^*(x,a)]} = \frac{1}{E[P^*(x,a)]} \frac{k}{N_a} \sqrt{N_a P^*(x,a) \left(1 - \frac{P^*(x,a)}{k}\right)} \tag{2.58}$$

$$\approx \frac{1}{P^*(x,a)} \frac{k}{N_a} \sqrt{N_a P^*(x,a)(1 - \frac{P^*(x,a)}{k})} \tag{2.59}$$

$$\approx \frac{k}{\sqrt{N_a}} \sqrt{\frac{(1 - \frac{P^*(x,a)}{k})}{P^*(x,a)}} \ . \tag{2.60}$$

The Parzen estimate has a larger normalized standard deviation at points where the estimated probability is small. The Parzen density estimate converges to the true estimate at a rate proportional to $\frac{1}{\sqrt{N_a}}$.

The definition of Parzen window estimation can be generalized to higher dimensions by replacing the one dimensional smoothing functions by their d dimensional counterparts (see Section 2.4 for a d dimensional Gaussian). Though the definition of Parzen estimation is the same for any number of dimensions, the behavior of the algorithm can be very different. As the number of dimensions grows the number of data points required rapidly increases. In d dimensions, the window of support of a Gaussian smoothing function is an d dimensional sphere whose radius r is a function of its standard deviation. The volume of a d dimensional sphere of radius r is $V_d r^d$, where V_d is a constant dependent only on the dimension. Assuming that all of the sample data is contained in a sphere of radius R, on average $N_a \left(\frac{r}{R}\right)^d$ data points will fall in a randomly chosen window. Generally, r is selected so that $\frac{r}{R} < 1$. As a result with increased dimension the number of points falling in a randomly chosen window drops exponentially. While in theory this could be remedied by increasing the size of the sample exponentially, things rapidly get out of hand. Empirical evidence argues against using Parzen estimation in many more than six dimensions.

Chapter 3

Empirical Entropy Manipulation and Stochastic Gradient Descent

This chapter presents a novel technique for evaluating and manipulating the empirical entropy of a distribution, called EMMA[1]. The theory of entropy manipulation plays a critical role in the rest of this thesis and forms the algorithmic core in all of the applications.

There are a number of existing techniques that manipulate the entropy of a density. They each have significant theoretical and practical limitations that make them unsuitable for our purposes. We will begin with a description of these techniques, and two simple applications. The second part of the chapter describes a new procedure for evaluating empirical entropy, EMMA. We will present an efficient stochastic gradient scheme for extremizing the EMMA estimates. This scheme has applications outside of entropy manipulation.

In the final section of the chapter we will present a tutorial application of EMMA. We will show how EMMA can be used to derive an information theoretic version of principal components analysis.

[1]EMMA is a random but pronouncible subset of the letters in the words "EMpirical entropy Manipulation and Analysis".

3.1 Empirical Entropy

As we saw in the previous chapter, in many cases the true density of a random variable is not known. Instead one must make do with an estimate of the density obtained from a sample of the variable. Likewise, there is no direct procedure for evaluating entropy from a sample. A common approach is to first model the density from the sample, and then estimate the entropy from the density. This divides the problem into manageable parts which can be solved separately.

By far the most popular density model for entropy calculations is the Gaussian. This is based on two considerations: (1) finding the Gaussian that fits the data best is very easy (see Section 2.3.1) and (2) the entropy of the Gaussian can be directly calculated from its variance. The entropy of a Gaussian distribution is

$$\begin{aligned}
h(X) &= -\int_{-\infty}^{\infty} g_\psi(x-\mu)\log(g_\psi(x-\mu))dx \\
&= -\int_{-\infty}^{\infty} g_\psi(x-\mu)\left[\log\left(\frac{1}{\sqrt{2\pi\psi}}\right) - \frac{1}{2}\frac{(x-\mu)^2}{\psi}\right]dx \\
&= E\left[\frac{1}{2}\frac{(x-\mu)^2}{\psi}\right] + \frac{1}{2}\log(2\pi\psi) \\
&= \frac{1}{2} + \frac{1}{2}\log(2\pi\psi) \\
&= \frac{1}{2}\log(e) + \frac{1}{2}\log(2\pi\psi) \\
&= \frac{1}{2}\log(2e\pi\psi) \ .
\end{aligned}$$

The entropy of a Gaussian is a function of its variance and is not a function of its mean. Wider Gaussians have more entropy and narrower Gaussians have less. There is a simple procedure for finding the empirical entropy of a Gaussian distribution: compute the variance of the sample and evaluate (3.1).

The equivalence between the log of variance and entropy can be used to reformulate well known signal and image processing problems as entropy problems. Since the logarithm is a monotonically increasing function, any technique that maximizes or minimizes the variance of a signal can be viewed as an entropy technique. Examples include principal components analysis, where variance is maximized, and least square solutions to matching problems, where variance is minimized. There is however one significant caveat. Variance maximization is only equivalent to entropy maximization if the density of the signal involved is Gaussian. When this assumption is violated it is possible to reduce entropy as variance is increased. All but a very few of the techniques that manipulate the entropy of a signal assume that the signals are Gaussian or another exponential distribution (Linsker, (Becker and Hinton, 1992), Zemel). We will discuss these techniques in Section 7.1.

Principal Components Analysis

There are a number of signal processing and learning problems that can be formulated as entropy maximization problems. One well known example is *principal component analysis.* The principal component of a d dimensional distribution is a d dimensional vector. Given a density X every vector v defines a new random variable, $Y_v = X \cdot v$. The variance along an axis v is the variance of this new variable:

$$Var(Y_v) = E_X[(X \cdot v - E_X(X \cdot v))^2] \ . \tag{3.1}$$

The principal component V is the vector for which $Var(Y_v)$ is maximized.

In practice neither the density of X nor Y_v is known. The projection variance is computed from a sample a of points from X,

$$Var(Y_v) \approx Var_a(Y_v) \equiv E_a[(X \cdot v - E_a[X \cdot v])^2] \ . \tag{3.2}$$

We can then define a vector cost function,

$$C(v) = -Var_a(Y_v) \ , \tag{3.3}$$

which is minimized by the principal component vector. There are many schemes for finding the principal component of a density. One of the more elegant accomplishments of linear algebra is the proof that the first eigenvector of the covariance matrix of X, ψ_X, is the principal component vector.

Under the assumption that X is Gaussian we can prove that the principal component is the projection with maximum entropy. First, every projection of a Gaussian is also Gaussian. Second, the entropy of a Gaussian is monotonically related to variance. Therefore, the Y_v corresponding the axis of highest variance is a Gaussian with the highest possible entropy. Moreover, principal components analysis finds the axis that contains the most information about X. The mutual information between X and Y_v is

$$I(X, Y_v) = h(Y_v) - h(Y_v|X) \ . \tag{3.4}$$

This equation has two components. The first implies that Y_v will give you more information about X when Y_v has a lot of entropy. The second can be misleading. Since knowing X removes all of the randomness from Y_v, $h(Y_v|X)$ is negative infinity. This is not particularly bothersome precisely because relative entropy is relative. Only the differences between relative entropies are significant. The variable Y_{v_1} yields more information about X than Y_{v_2} when

$$I(X, Y_{v_1}) > I(X, Y_{v_2}) \tag{3.5}$$
$$h(Y_{v_1}) - h(Y_{v_1}|X) > h(Y_{v_2}) + h(Y_{v_2}|X) \tag{3.6}$$
$$h(Y_{v_1}) > h(Y_{v_2}) \ . \tag{3.7}$$

We can conclude that the principal component axis carries more information about a distribution than any other axis.

Function Learning

There are other well known problems that can be formulated within the entropy framework. Let us analyze a simple learning problem. Given a random variable X we can define a functionally dependent RV, $Y = F(X \cdot v) + \eta$, which we assume has been perturbed by measurement noise. We are given samples of the joint occurrences of the RVs: $a = [...\{x_a, y_a\}...]$. How can we estimate v? Typically this is formulated as a least squares problem. A cost is defined as the sum of the squares of the differences between predicted $\hat{y}_a = F(x_a \cdot \hat{v})$ and the actual samples of Y,

$$C(\hat{v}) = \sum_{\{x_a, y_a\} \in a} (y_a - F(x_a \cdot \hat{v}))^2 \ . \tag{3.8}$$

The cost is a function of the estimated parameter vector $\hat{v}$. The sum of squared difference can be justified from a log likelihood perspective (see Section 2.3.1). If we assume that the noise added to Y is Gaussian and independent of Y then the log likelihood of $\hat{v}$ is:

$$
\begin{aligned}
log(\ell(\hat{v})) &= \sum_{\{x_a,y_a\}\in a} log(p(Y = y_a|\hat{Y} = F(x_a \cdot \hat{v}))) && (3.9) \\
&= \sum_{\{x_a,y_a\}\in a} log(g_{\psi}(y_a - F(x_a \cdot \hat{v}))) && (3.10) \\
&= - \sum_{\{x_a,y_a\}\in a} (y_a - F(x_a \cdot \hat{v}))^2 + C \ , && (3.11)
\end{aligned}
$$

where C is a constant value independent of $\hat{v}$. The $\hat{v}$ that minimizes the sum of the squares is also the $\hat{v}$ that makes the data observed most likely. For most problems like this, gradient descent is used to search for $\hat{v}$.

Finally we can show that minimizing cost maximizes mutual information. We showed in Section 2.3.1 that log likelihood is related to sample entropy:

$$
log(\ell(\hat{v})) = -\frac{1}{N_a} h_a(Y|\hat{Y}) \approx -\frac{1}{N_a} h(Y|\hat{Y}) \ . \tag{3.12}
$$

The mutual information between Y and $\hat{Y}$ is

$$
I(Y, \hat{Y}) = h(Y) - h(Y|\hat{Y}) \ . \tag{3.13}
$$

The first term is not a function of $\hat{v}$. The second is an approximation of log likelihood—minimizing $C(\hat{v})$ maximizes $I(Y, \hat{Y})$.

Non-Gaussian Densities

There is a commonly held misconception that mutual information techniques are all equivalent to simple well known algorithms. Contrary to the impression that the above two examples may give, this is far from the truth. Entropy is *only* equivalent to least squares when the data is assumed to be Gaussian. The approach to alignment and bias correction that we will describe in the next chapters does not and could not assume that distributions were Gaussian. We will show that if the data were Gaussian our alignment technique would reduce to correlation.

There are a number of non-Gaussian problems that can be solved using entropy or mutual information. Bell has shown that signal separation and de-correlation can be thought of as entropy problems (Bell, 1995). Bell's technique can be derived both for Gaussian and non-Gaussian distributions. Bell shows that the Gaussian assumption leads to a well known and ineffective algorithm. When the signals are presumed to be non-Gaussian, the resulting algorithms are much more effective. Many compression and image processing problems clearly involve non-Gaussian distributions.

In theory, empirical entropy estimation can be used with any type of density model. The procedure is the same: estimate the density from a sample and compute the entropy from the density. In practice, the process can be computationally intensive. The first part, maximum likelihood density estimation, is an iterative search through parameter space. The second, evaluating the entropy integral, may well be impossible. For example, there is no known closed form solution for the entropy of a mixture of Gaussians. The entropy integral can however be approximated as a sample mean:

$$h(X) \approx h_b(X) = E_b[log(\hat{p}(X))] \ , \tag{3.14}$$

where $E_b[\,]$ is the sample mean taken over the sample b, $\hat{p}()$ is the estimate for the sample density and $h_b(X)$ is the sample entropy first introduced in Section 2.3.1. The sample mean converges toward the true mean at a rate proportional to $1/\sqrt{N_b}$ (N_b is the size of b). Based on this insight, two samples can be used to estimate the entropy of a distribution: the first is used to estimate the density, the second is used to estimate the entropy.

While the two sample approach can be used to estimate entropy, it is not a practical algorithm for entropy manipulation. In the two applications above, changes in the parameter vector effect the densities that are being approximated. As the search through parameter space adjusts the parameter vector a new sample must be drawn, a new density estimated, and the derivative of entropy evaluated. If estimating the density is itself a complex search process, the search for the correct parameter vector can take an unbearably long time.

3.2 Estimating Entropy with Parzen Densities

In this section we will describe a technique that can effectively estimate and manipulate the entropy of non-Gaussian distributions. The basic insight is that rather than use maximum likelihood to estimate the density of a sample, we will instead use Parzen window density estimation (see Section 2.4.3). The Parzen scheme for estimating densities has two significant advantages over maximum likelihood: (1) since the Parzen estimate is computed directly from the sample, there is no search for parameters; (2) the derivative of the entropy of the Parzen estimate is simple to compute.

In the following general derivation we will assume that we have samples of a random variable X, and we would like to manipulate the entropy of the random variable $Y = F(X, v)$. The entropy, $h(Y)$, is now a function of v and can be manipulated by changing v. Since there is no direct technique for finding the parameters that will extremize $h(Y)$ we will search the parameter space using gradient descent. The following derivation assumes that Y is a vector random variable. The joint entropy of a two random variables, $h(Y_1, Y_2)$, can be evaluated by constructing the vector RV, $W = [Y_1, Y_2]^T$ and evaluated $h(W)$.

The form of the Parzen estimate constructed from a sample a is

$$P^*(y, a) = \frac{1}{N_a} \sum_{y_a \in a} g_\psi(y - y_a) \ , \tag{3.15}$$

where the Parzen estimator is constructed with Gaussian smoothing functions. Given $P^*(y, a)$ we can approximate entropy as the sample mean,

$$\begin{aligned} h^*(Y) &\equiv E_b[log(P^*(Y, a))] && (3.16) \\ &= \frac{1}{N_b} \sum_{y_b \in b} log(P^*(y_b, a)) \ , && (3.17) \end{aligned}$$

computed over a second sample b ($h^*(X)$ is the EMMA estimate of empirical entropy).

In order to extremize entropy we must calculate the derivative of entropy with respect to v. This may be expressed as

$$\frac{d}{dv}h^*(Y) = \frac{-1}{N_B}\sum_{y_b \in b} \frac{\sum_{y_a \in a} \frac{d}{dv} g_\psi(y_b - y_a)}{\sum_{y_a \in a} g_\psi(y_b - y_a)} \ , \tag{3.18}$$

and, after differentiating the Gaussian,

$$\frac{d}{dv}h^*(Y)) = \frac{1}{N_b}\sum_{y_b \in b} \frac{\sum_{y_a \in a} g_\psi(y_b - y_a)\ (y_b - y_a)^T \psi^{-1} \frac{d}{dv}(y_b - y_a)}{\sum_{y_a \in a} g_\psi(y_b - y_a)} \ . \tag{3.19}$$

This expression may be written more compactly as follows,

$$\frac{d}{dv}h^*(Y) = \frac{1}{N_b}\sum_{y_b \in b}\sum_{y_a \in a} W_y(y_b, y_a)\ (y_b - y_a)^T \psi^{-1} \frac{d}{dv}(y_b - y_a) \ , \tag{3.20}$$

using the following definition:

$$W_y(y_b, y_a) \equiv \frac{g_\psi(y_b - y_a)}{\sum_{y_a \in a} g_\psi(y_b - y_a)} \ . \tag{3.21}$$

$W_y(y_b, y_a)$ takes on values between zero and one. It will approach one if y_b is significantly closer to y_a than any other element of a. It will be near zero if some other element of a is significantly closer to y_b. Distance is interpreted with respect to the squared Mahalonobis distance (see (Duda and Hart, 1973))

$$D_\psi(y) \equiv y^T \psi^{-1} y \ .$$

Thus, $W_y(y_b, y_a)$ is an indicator of the degree of match between its arguments, in a "soft" sense. It is equivalent to using the "softmax" function of neural networks (Bridle, 1989) on the negative of the Mahalonobis distance to indicate correspondence between y_b and elements of a.

Equation 3.20 may also be expressed as

$$\frac{d}{dv}h^*(Y) = \frac{1}{N_B}\sum_{y_b \in b}\sum_{y_a \in a} W_y(y_b, y_a)\ \frac{d}{dv}\frac{1}{2}D_\psi(y_b - y_a) \ . \tag{3.22}$$

In this form it is apparent that to reduce entropy, the parameters v should be adjusted such that there is a reduction in the average squared distance between points which W indicates are nearby.

Before moving on, it is worth reemphasizing that for most density models $\frac{d}{dv}h(Y)$ is very difficult to compute. The general derivation of the derivative of entropy is much more complex than the Parzen derivation:

$$\frac{\partial h(Y)}{\partial v} \approx \frac{d}{dv}\frac{1}{N_b}\sum_{y_b \in b} log p(y_b, a) \tag{3.23}$$

$$= \frac{1}{N_b}\sum_{y_b \in b} \frac{\frac{d}{dv}p(y_b, a)}{p(y_b, a)} \tag{3.24}$$

$$= \frac{1}{N_b}\sum_{y_b \in b} \frac{\frac{d}{dy_b}p(y_b, a)\cdot\frac{dy_b}{dv} + \frac{d}{da}p(y_b, a)\cdot\frac{da}{dv}}{p(y_b, a)} \ . \tag{3.25}$$

The numerator of the derivative has two components. The first, $\frac{d}{dy_b}p(y_b, a)\cdot\frac{dy_b}{dv}$, is the change in entropy that results from changes in the sample b. The second, $\frac{d}{da}p(y_b, a)\cdot\frac{da}{dv}$, is far more problematic. The second component is a measure of the change in the density estimate that results from changes in the sample a. In the Parzen framework the two components of the derivative collapse into a single term and can be directly computed from the samples. In the maximum likelihood framework $p(y_b, a)$ is a complex function of the sample. Since there is no closed form function that computes the density estimate from the sample, computing its derivative can be very difficult.

3.3 Stochastic Maximization Algorithm

The variance maximization/minimization applications described above (principal components analysis and learning) are deterministic procedures. Starting from an initial guess, gradient descent uses the derivative of cost to repeatedly update the parameter vector. Two different runs that start from the same initial parameters will end up with the same final parameters. Our justification for using probability and entropy to analyze these problems is purely convenience. There is nothing random about these problems once the samples are drawn. One of the benefits of understanding the probabilistic interpretation of these problems is that we can introduce randomness into our solutions and understand its effect. Here is a simple example: we want to know the average of a large sample of data. Without knowing anything else, it would make sense to sum over the entire sample. But, if we needed only a rough estimate of the average, significant computation could be saved by averaging over a subset of the sample. Furthermore, knowledge of the sample variance would allow us to compute the size of the subsample needed to estimate the mean to a given precision.

A similar analysis can be applied to principal components analysis or function learning. The cost of a particular parameter vector is computed by summing over an entire sample as we did in Equation 3.8. But, when that sample is very large this expectation can be approximated by a smaller random sample. The same argument applies to the gradient. Since the gradient is defined as an average over a very large sample it may make sense to approximate it over a smaller random sample. When we use random samples, both the error estimate and the gradient estimate are now truly random. For very large samples, accurate error/gradient estimates can be made without averaging over the entire sample. For problems where the gradient needs to be evaluated often, this can save significant computation.

Though a random estimate of the gradient is cheaper to compute, it could be useless. Under what conditions does it make sense to use a random gradient estimate? The theory of stochastic approximation tells us that stochastic estimates of the gradient can be used instead of the true gradient when the following conditions hold: (1) the gradient estimate is unbiased; (2) the parameter update rate asymptotically converges to zero; (3) the error surface is quadratic in the parameters ((Robbins and Munroe, 1951), (Haykin, 1994), (Ljung and Söderström, 1983)). The first condition requires that on average the estimate for the gradient is the true gradient. The second insures that the search will eventually stop moving about randomly in parameter space. In practice the third condition can be relaxed to include most smooth non-linear error surfaces; though there is no guarantee that the parameters will end up in any particular minimum.

Returning our attention to equations (3.16) and (3.20), notice that the both the calculation of the EMMA entropy estimate and its derivative involve a double summation. One summation is over the points in sample a and another over the points in b. As a result the cost of evaluation is quadratic in sample size: $O(N_a N_b)$. We will present an experiment where the derivative of entropy for an image containing 60,000 pixels is evaluated. While the true derivative of empirical entropy could be obtained by exhaustively sampling the data, a random estimate of the entropy can be obtained with much less computation. This is especially critical in entropy manipulation problems, where the derivative of entropy is evaluated many thousands of times. Without the quadratic savings that arise from using smaller samples entropy manipulation would be impossible.

For entropy manipulation problems involving large samples we will use stochastic gradient descent. Stochastic gradient descent seeks a local maximum of entropy by using a stochastic estimate of the gradient instead of the true gradient. Steps are repeatedly taken that are proportional to the approximation of the derivative of the mutual information with respect to the parameters:

```
Repeat:

        $a \leftarrow \{N_a \text{ samples drawn from } y\}$
        $b \leftarrow \{N_b \text{ samples drawn from } y\}$
        $v \leftarrow v + \lambda \frac{dh}{dv}$
```

where $\frac{dh}{dv}$ is the derivative of entropy evaluated over samples a and b, v is the parameter to be estimated, and the parameter λ is called the "learning rate". The above procedure is repeated a fixed number of times or until convergence is detected. In most problems the initial value of λ is reduced during the search. In subsequent chapters we will describe experiments where samples of 50 or less can be used to effectively find entropy maxima.

Stochastic approximation does not seem to be well known in the computer vision community. We believe that it is applicable to number of cost minimization problems that arise in computer vision. Stochastic gradient descent is most appropriate for tasks where evaluation of the true gradient is expensive, but an unbiased estimate of the gradient is easy to compute. Examples include problems where the derivative is a sum over all of the pixels in an image. In these cases, stochastic gradient search can be orders of magnitude faster than even the most complex second order gradient search schemes. In the experimental section (Chapter 6) of this thesis we will briefly describe joint work with Mike Jones and Tomaso Poggio where an existing vision application was sped up by a factor of fifty using stochastic approximation.

Convergence of Stochastic EMMA

Most of the conditions that insure convergence of stochastic gradient descent are easy to obtain in practice. For example, it is not really necessary for the learning rate to asymptotically converge to zero. At non-zero learning rates the parameter vector will move randomly about the minimum/maximum endlessly. Smaller learning rates make for smaller excursions from the true answer. An effective way to terminate the search is to detect when on average the parameter is not changing and then reduce the learning rate. The learning rate only needs to approach zero if your goal is zero error, something that no practical system can achieve anyway. A better idea is to reduce the learning rate until the parameters have a reasonable variance and then take the average parameters.

The first proofs of stochastic approximation required that the error be quadratic in the parameters. More modern proofs are more general. For convergence to a particular optimum, the parameter vector must be guaranteed to enter that optimum's basin of attraction infinitely often. A basin of attraction of an optimum is defined with respect to true gradient descent. Each basin is a set of points from which true gradient descent will converge to the same optimum. Quadratic error surfaces have a single optimum and the basin of attraction is all of parameter space. Non-linear error spaces may have many optima, and the parameter space is partitioned into many basins of attraction. When there are a finite number of optima, we can prove that stochastic gradient descent will converge to one of them. The proof proceeds by contradiction. Assume that the parameter vector never converges. Instead it wanders about parameter space forever. Since parameter space in partitioned into basins of attraction it is always in some basin. But since there are a finite number of basins it must be in one basin infinitely often. So it must converge to the optima of that basin.

One condition will give us more trouble than the others. The stochastic estimate of the gradient must be unbiased. It is *not* true that the sample approximation for empirical entropy is unbiased. Moreover, we have been able to prove that it has a consistent bias: it is too large. In Section 2.4.3 we described the conditions under which the Parzen density estimate is unbiased. When these conditions are met, a number of equalities hold:

$$\begin{aligned} p(X = x) &= \lim_{N_a \to \infty} P^*(x,a) & (3.26) \\ &= E_{\{a \in \{X\}\}}[P^*(x,a)] & (3.27) \\ &= E_{\{a \in \{X\}\}}\left[\frac{1}{N_a} \sum_{x_a \in a} R(x - x_a)\right] & (3.28) \\ &= E_X[R(X - x)] \ . & (3.29) \end{aligned}$$

Here $E_{\{a \in \{X\}\}}[P^*(x,a)]$ denotes the expectation over all possible random samples of size N_a drawn from the random variable X. Assuming the different samples of X are independent allows us to move the expectation inside the summation. The true entropy of the RV X can be expressed as

$$h(X) = -E_X[log\ E_{\{a \in \{X\}\}}[P^*(x,a)]] \ . \tag{3.30}$$

We can define a similar statistic:

$$\begin{aligned} \overline{h^*(X)} &\equiv E_{\{b \in \{X\}, a \in \{X\}\}}[h^*(X)] & (3.31) \\ &= -E_{\{b \in \{X\}, a \in \{X\}\}}[E_b[log E_a[R(x_b - x_a)]]] & (3.32) \\ &= -E_X[E_{\{a \in \{X\}\}}[log(P^*(x,a))]] \ . & (3.33) \end{aligned}$$

$\overline{h^*(X)}$ is the expected value of $h^*(X)$. Therefore, $h^*(X)$ provides an unbiased estimate of $\overline{h^*(X)}$. Jensen's inequality allows us to move the logarithm inside the expectation:

$$\begin{aligned} h(X) &= -E_X[log E_{\{a \in \{X\}\}}[P^*(x,a)]] & (3.34) \\ &\leq -E_X[E_{\{a \in \{X\}\}}[log(P^*(x,a))]] & (3.35) \\ &\leq \overline{h^*(X)} \ . & (3.36) \end{aligned}$$

The stochastic EMMA estimate is an unbiased estimator of a statistic that is provably larger than the true entropy. Intuitively, overly large estimates arise when elements of b fall in regions where $P^*(x,a)$ is too small. For these points the log of $P^*(x,a)$ is much smaller than it should be.

How then might we patch the definition of EMMA to remedy the bias? Another statistic that is similar to entropy is

$$\overline{\hat{h}(X)} = -E_X[p(X)] = -\int_{-\infty}^{\infty} p(x)^2 dx \ . \tag{3.37}$$

The definition of $\overline{\hat{h}(X)}$ bears a resemblance to the ***** Fisher information (we will define $\hat{h}(X)$ shortly). $\overline{\hat{h}(X)}$ is a measure of the randomness of X. Strongly peaked distributions have large negative values for $\overline{\hat{h}}$. For widely spread uniform distributions $\overline{\hat{h}}$ approaches zero. Using the well known inequality, $x - 1 \geq log(x)$, we can show,

$$\overline{\hat{h}(X)} = -\int_{-\infty}^{\infty} p(x)^2 dx \tag{3.38}$$

$$\leq -\int_{-\infty}^{\infty} p(x) log(p(x)) dx \tag{3.39}$$

$$\leq h(X) \leq \overline{h^*(X)} \tag{3.40}$$

Parzen window density estimation can be used to construct a stochastic measure

$$\hat{h}(X) \equiv -E_b[P^*(x, a)] \ . \tag{3.41}$$

The expectation of $\hat{h}(X)$ is $\overline{\hat{h}(X)}$:

$$-E_{\{b \in \{X\}, a \in \{X\}\}}[\hat{h}(X)] = -E_{\{b \in \{X\}, a \in \{X\}\}}[E_b[P^*(x_b, a)]] \tag{3.42}$$

$$= -E_X[E_{\{a \in \{X\}\}}[P^*(X, a)]] \tag{3.43}$$

$$= -E_X[p(X)] \tag{3.44}$$

$$= \overline{\hat{h}(X)} \ . \tag{3.45}$$

Further simplifying,

$$\overline{\hat{h}(X)} = -E_{x=X}[E_{\tilde{x}=X}[R(x - \tilde{x})]] \tag{3.46}$$

is an expectation over a pair of events from the RV X. On average $\hat{h}$ is far too negative when $p(X)$ is large.

We have now defined two alternative statistics for which inexpensive unbiased estimates are available. h^* is on average too large, and $\hat{h}$ is on average too small. Instead of using either, we have had good success using a third estimate,

$$\tilde{h}(X) = -E_b[\Upsilon(x_b, a)] \ , \tag{3.47}$$

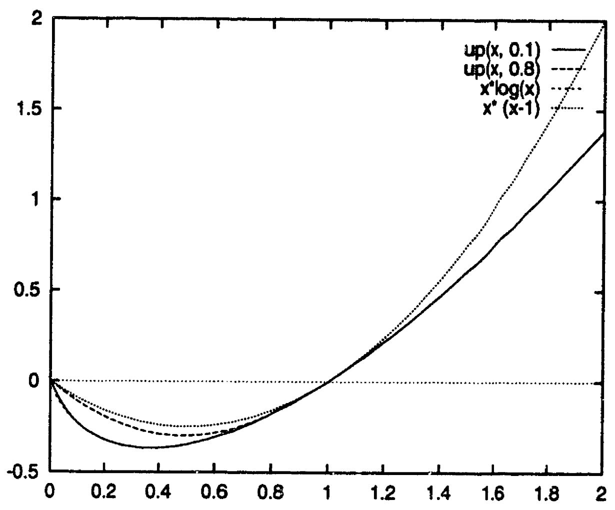


Figure 3.1: Plot of the functions $x\ \Upsilon(x)$, $x\ log(x)$ and $x\ (x-1)$. Two different values for p_{min} are plotted: 0.1 and 0.8. Notice that smaller values of p_{min} cause the approximation of $x\ log(x)$ to be very good. The difference between the two functions when $p_{min} = 0.1$ is almost unnoticeable.

where

$$\Upsilon(x,a) = \begin{cases} log(P^*(x,a)) & \text{if } P^*(x,a) < p_{min} \\ \frac{P^*(x,a)}{p_{min}} + log(p_{min}) - 1 & \text{otherwise} \end{cases} . \tag{3.48}$$

The Υ function is designed so that both $\Upsilon(x,a)$ and $\frac{d\Upsilon(x,a)}{dx}$ are continuous. See Figure 3.1 for a plot of the Υ function. The intuition behind $\tilde{h}$ is that we use h^* whenever possible, and for those sample points where $P^*(x,a)$ has a large standard deviation, we use $\hat{h}$ instead. The standard deviation of the Parzen density estimate is highest at points where the probability density is lowest, so we use $\hat{h}$ where $P^*(x,a)$ is below p_{min} (see Section 2.4.3). The variable p_{min} allows us to continuously vary the $\tilde{h}$ estimate from the two extremes of $\hat{h}$ and h^*.

Include table of empirical vs true entropies for a number of different distribution and description

Other Stochastic Search Techniques

Non-linear stochastic gradient descent is commonly used in the neural network literature, where it is often called the LMS rule. It was introduced there by Widrow and Hoff (Widrow and Hoff, 1960) and has been used extensively with good results. It has been observed that the variation introduced by the sampling can effectively penetrate small local minima. Such local minima are often characteristic of continuous alignment schemes, and we too have found that local minima can be overcome in this manner. The textbook by Haykin (Haykin, 1994) discusses the use of such algorithms by the neural network community. An excellent discussion of stochastic approximation appears in the textbook by Ljung and Söderström (Ljung and Söderström, 1983).

Simulated annealing is a related method that has been used in optimization problems which have local minima (Kirkpatrick et al., 1983). Simulated annealing performs a random, though usually local, search through parameter space. At each step a random modification to the parameters is proposed and the new cost is evaluated. If the new cost is lower than the previous cost the parameter modification is accepted. If the difference in cost is positive, the modification is accepted probabilistically. The probability of acceptance is proportional to the negative exponential of the difference,

$$p_{accept}(d) = \frac{exp(-\frac{d}{t})}{t} \ , \tag{3.49}$$

where d is the difference between the new and the old cost and t is a temperature that controls the likelihood that a bad modification is accepted. Simulated annealing is based on the insight that physical systems, like iron, invariably find good energy minima when heated and then cooled slowly. The process of physical annealing is basically a gradient search perturbed by thermal noise. The thermal noise provides the energy to kick physical systems out of unfavorable local minima. The parameter t is an analog of physical temperature, it is initially set to large values and is gradually "cooled" during learning. In some cases it can be proven that with the right annealing schedule simulated annealing will converge to the global optimum.

We believe that stochastic approximation serendipitously combines efficient computation with effective escape from local minima.

3.3.1 Estimating the Covariance

In addition to the learning rate λ, the covariance matrices of the smoothing functions densities R are important parameters of EMMA. These parameters may be chosen so that they are optimal in the maximum likelihood sense. This is equivalent to minimizing the cross entropy of the estimated distribution with the true distribution (see Section 2.4.3). Our goal is to find the parameters that minimize empirical entropy.

For simplicity, we assume that the covariance matrices are diagonal,

$$\psi = \mathrm{DIAG}(\sigma_1^2, \sigma_2^2, \ldots) \ . \tag{3.50}$$

Following a derivation almost identical to the one described in Section 3.2 we can derive an equation analogous to (3.20),

$$\frac{d}{d\sigma_k} h^*(Y) = \frac{1}{N_b} \sum_{y_b \in b} \sum_{y_a \in a} W_y(y_b, y_a) \left(\frac{1}{\sigma_k}\right) \left(\frac{[y]_k^2}{\sigma_k^2} - \frac{1}{2}\right) \tag{3.51}$$

where $[y]_k$ is the kth component of the vector y. This equation forms the basis for a method of stochastic maximization of likelihood.

```
Repeat:

        $A \leftarrow \{N_a \text{ points drawn from } y\}$
        $B \leftarrow \{N_b \text{ points drawn from } y\}$
        $\sigma_k \leftarrow \sigma_k + \lambda' \frac{d}{d\sigma_k} h^*(Y) \quad \forall k$
```

The above procedure is very similar to the one described in Section 3.3. During entropy manipulation, it is possible to interleave covariance updates with parameter updates.

3.4 Principal Components Analysis and Information

We can now rederive a parameter estimation rule akin to principal components analysis that truly maximizes information. Principal components is used in compression and dimensionality reduction. This new EMMA based component analysis (ECA) should have applications in these areas.

ECA manipulates the entropy of the random variable $Y_v = X \cdot v$ under the constraint that $|v| = 1$. Given a trial of the RV X, we can compute a trial for the RV Y_v as $y = x \cdot v$. For any sample X a sample of Y_v can be computed. For any given value of v the entropy of Y_v can be estimated from a sample of X as:

$$h^*(Y_v) = \frac{1}{N_b} \sum_{y_b \in b} log \left(\frac{1}{N_a} \sum_{y_a \in a} g_\psi(y_b - y_a) \right) \tag{3.52}$$

$$= \frac{1}{N_b} \sum_{x_b \in b} log \left(\frac{1}{N_a} \sum_{x_a \in a} g_\psi(x_b \cdot v - x_a \cdot v) \right) \tag{3.53}$$

where ψ is the variance of the Parzen smoothing function. Moreover we can estimate the derivative of entropy:

$$\frac{d}{dv} h^*(Y_v) = \frac{1}{N_b} \sum_{y_b \in b} \sum_{y_a \in a} W_y(y_b, y_a) \, \frac{1}{\psi}(y_b - y_a) \frac{d}{dv}(y_b - y_a) \tag{3.54}$$

$$= \frac{1}{N_b} \sum_b \sum_a W_y(y_b, y_a) \, \frac{1}{\psi}(y_b - y_a)(x_b - x_a) \ , \tag{3.55}$$

Let us decompose the derivative into parts which can be understood more easily. We will first analyze the second part of the summand: $(y_b - y_a)(x_b - x_a)$. Ignoring the weighting function W_y we could construct the derivative of some mystery function:

$$\frac{d}{dv} f(Y_v) = \sum_b \sum_a (y_b - y_a)(x_b - x_a) \tag{3.56}$$

$$= N_b N_a E_b[E_a[(y_b - y_a)(x_b - x_a)]] \ . \tag{3.57}$$

What then is $f(Y_v)$? The derivative of the squared difference between samples is:

$$\frac{d}{dv}(y_b - y_a)^2 = 2(y_b - y_a)\frac{d}{dv}(x_b \cdot v - x_a \cdot v) \tag{3.58}$$

$$= 2(y_b - y_a)\frac{d}{dv}(x_b - x_a \cdot v) \tag{3.59}$$

$$= 2(y_b - y_a)(x_b - x_a) \ . \ . \tag{3.60}$$

So we can see that

$$f(Y_v) = N_b N_a E_b[E_a[(y_b - y_a)^2]] \tag{3.61}$$

is the expectation of the squared difference between pairs of trials of Y_v.

Recall that PCA searches for the RV Y_v that has the largest variance: $E_a[(y_a - E_a[y_a])^2] = Var_a(Y_v)$. Interestingly the expected squared difference between a pair of trials is precisely twice the variance:

$$C(v) = E_b[E_a[(y_b - y_a)^2]] \tag{3.62}$$

$$= E_b[E_a[y_b^2 - 2y_a y_b + y_a^2]] \tag{3.63}$$

$$= E_b[y_b^2 + E_a[2y_a y_b + y_a^2]] \tag{3.64}$$

$$= E_b[y_b^2 - 2y_b E_a[y_a] + E_a[y_a^2]] \tag{3.65}$$

$$= E_b[y_b^2] - 2E_b[y_b]E_a[y_a] + E_a[y_a^2]] \tag{3.66}$$

$$= E_b[Y^2] - 2E_b[Y]E_a[Y] + E_a[Y^2] \tag{3.67}$$

$$= 2Var_a(Y) \ , \tag{3.68}$$

Without the W_y ECA would find exactly the same vector that PCA does: the maximum variance projection vector. However, because of $W_y(y_b, y_a)$, the derivative of ECA does not force all pairs of points in Y apart equally. Recall that W_y is a measure of the distance between y_b and y_a. It is large when y_b is significantly closer to y_a than any other element of a. As a result ECA maximizes variance in a local way. Points that are very far apart are forced no further apart.

When a density is Gaussian the maximum entropy projection is the maximum variance projection. This first test of ECA is to insure that for Gaussian distributions it finds the same principal axis as PCA. Figure 3.2 shows a sample of data and the PCA and ECA principal components. Since this density has a larger

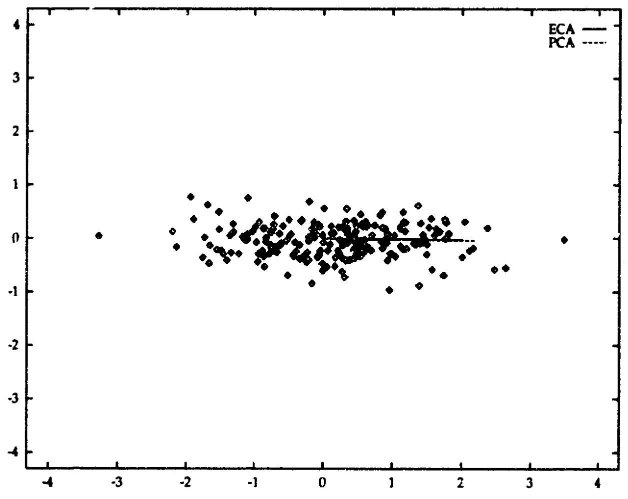


Figure 3.2: A scatter plot of a sample from a two dimensional Gaussian density. The sample contains 200 points. The principal axis and the ECA axis are also plotted as vectors from the origin. The vectors are nearly identical. To help differentiate them the PCA axis is plotted slightly longer.

variance along the horizontal axis, both the ECA and PCA axes point along the horizontal axis. Our ECA code take roughly 10 seconds to run on Sun Sparc5 workstation. This is comparable to the time it takes to run PCA.

In general, PCA does not find the highest entropy projection of non-Gaussian densities. Figure 3.3 shows a density for which the PCA and ECA axes are very different. The PCA axis, which is vertical, spreads the points in the sample as far apart as possible. The ECA axis, which is oblique, spreads nearby points in the sample as far apart as possible. The resulting densities, Y_{PCA} and Y_{ECA}, are graphed in Figure 3.4. The PCA density is very tightly peaked, the ECA density is broadly spread out. Though the final variance of Y_{PCA} is larger, 2.005 vs. 1.626, the entropy of the Y_{ECA} distribution is much higher, $h^*(Y_{PCA}) = -0.17$ and $h^*(Y_{ECA}) = 1.61$.

Linsker has argued that the PCA axis separates the clusters of a distribution (Linsker, 1988). To justify this claim, he uses figures much like Figure 3.3 and Figure 3.4. These graphs show the PCA axis projecting points from separated clusters so that they remain separate. It is then proposed that the PCA axis is useful for cluster classification of high dimensional data. In other words, that high dimensional data can be projected down into a low dimensional space without perturbing the cluster structure. In general this is *not* true. PCA only separates

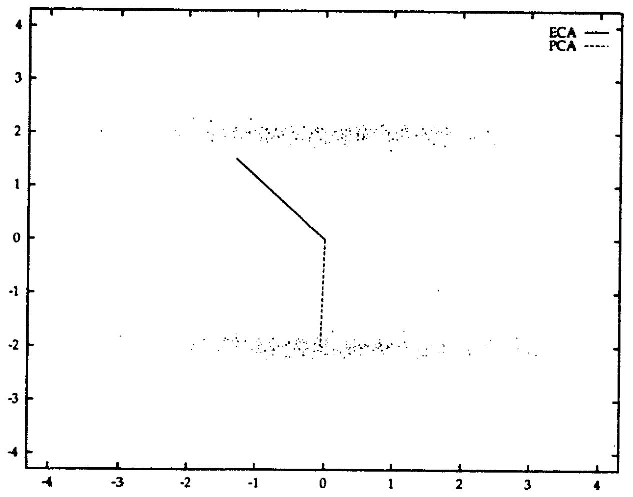


Figure 3.3: A scatter plot of a 400 point sample from a two dimensional density. The density is a mixture of two horizontally stretched Gaussians. The PCA and ECA principal axes are also plotted as vectors from the origin.

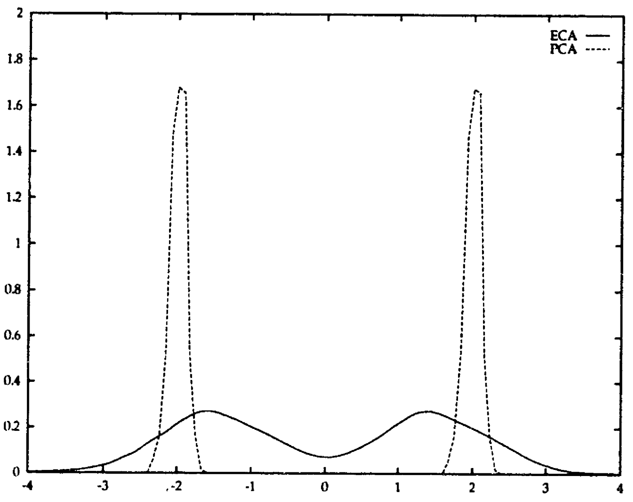


Figure 3.4: The Parzen density estimates of Y_{PCA} and Y_{ECA}.

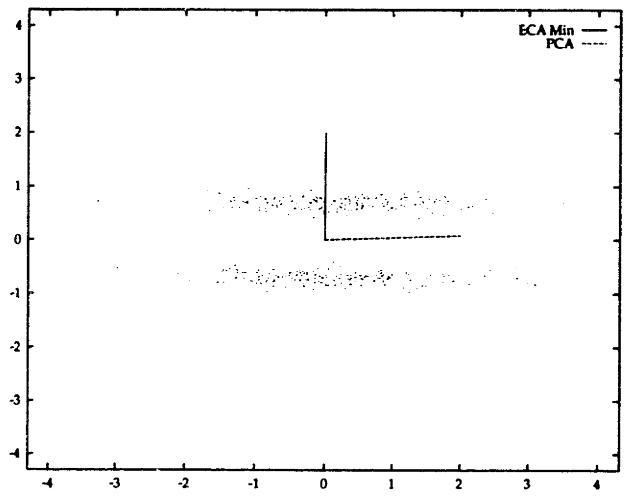


Figure 3.5: A scatter plot of a 400 point sample from a two dimensional density. The density is a mixture of two horizontally stretched Gaussians. The PCA and ECA minimum entropy axes are also plotted as vectors from the origin.

clusters when the variance between clusters is higher than the variance within clusters.

Ironically, it is the *minimum* entropy projection that should separate clusters well. Let us assume that each cluster is generated from a prototypical point that has been perturbed by random noise. If there were no noise, the sample points associated with a prototype would be clustered together very tightly. The entropy of the distribution would be very low. Noise acts to spread out the clusters and adds entropy. A projection that minimizes entropy will, if possible, be perpendicular to the noise. Figure 3.5 shows a distribution where the PCA axis does not separate the clusters at all. In this example, the within variance within the clusters is high compared with the variance between the clusters. The ECA axis shown is the minimum entropy axis (which is obtained by running the EMMA algorithm with a negative learning rate). The ECA axis separates the clusters much better than the PCA axis (see Figure 3.6).

3.5 Conclusion

This chapter has presented a new technique for estimating the entropy of a distribution called EMMA. Provided the density being approximated is smooth, we have

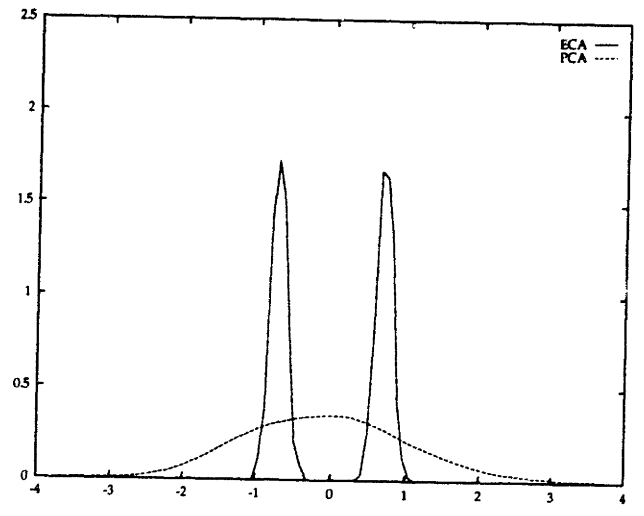


Figure 3.6: The Parzen density estimates of Y_{PCA} and Y_{ECA} from the previous graph.

proven that the technique will converge to the correct entropy estimate. Moreover we have presented an computationally efficient stochastic technique for manipulating entropy. For reasonable sample sizes, the technique is not guaranteed to optimize true entropy. Instead it optimizes a very similar statistic that retains all of the salient characteristics of entropy.

We have also described a simple application of EMMA. EMMA enables us to find low dimensional projections of higher dimensional that minimize or maximize entropy.

Chapter 4

Matching and Alignment

This chapter is perhaps the most important in this thesis. Previous chapters have presented the mathematics and algorithms that underly the computation of empirical entropy. We have already seen that empirical entropy can be used to define a new algorithm for finding the most informative projection of a distribution. This chapter will show that matching and alignment can also be formulated as an entropy problem. In addition, we will discuss the intuition behind our framework and suggest some simplified schemes that reflect these intuitions.

A number of simple synthetic alignment problems will drive our discussion of existing matching techniques like correlation. We will start out with a rederivation of correlation as a maximum likelihood method. This derivation will make clear the assumptions under which correlation will work, and when it may fail. We will then attempt to generalize correlation so that it will work with a wider set of inputs. While this generalization is theoretically straightforward it will prove computationally intractable.

Dropping our focus on correlation, we will define an intuitive approach to alignment which is efficiently computable. Using this intuition we will then define an approximation to a maximum likelihood technique that is both concrete and computable. Finally we will draw a parallel between this technique and mutual information. Experimental data from synthetic alignment problems will help us evaluate the proposed alignment techniques.

This chapter will conclude with an entirely different motivation for the use of mutual information in alignment. We will show how the alignment problem can be thought of as a *Minimum Description Length* problem (Rissanen, 1978; Leclerc, 1988). This formulation will naturally focus on the task of coding efficiency and entropy minimization. A very similar set of alignment equations will arise from these considerations.

4.1 Alignment

We are given two signals of time or space: $u(x)$ and $v(y)$. We will call $u(x)$ the *model*. Often it is an description of a physical object that has been computed with great care. For example, in one of our experiments the model is an accurate three dimensional description of a skull. The second signal, $v(y)$, is an *image* of the model. In general the form and even the coordinate systems of the model and image can be very different. For example, one of our 3D models is a collection of three dimensional points and normals; the corresponding image is a two dimensional array of intensities. It is assumed that $v(y)$ is an observation of $u(x)$, for example that $v(y)$ is a picture of the skull $u(x)$.

The relationship between $u(x)$ and $v(y)$ is based on the physics of imaging. The process of constructing an observation has two separate components. The first component is called a transformation, or pose, denoted $T(x)$. It relates the coordinate frame of the model to the coordinate frame of the image. The transformation tells us which part of the model is responsible for a particular pixel of the image. The second component is the imaging function, $F(u(x), q)$. The imaging function determines the value of image point $v(T(x))$. In general a pixel's value may be a function both of the model and other exogenous factors. For example, an image of an object depends not only on the object but also on the lighting. The parameter, q, collects all of the exogenous influences into a single vector. The complete imaging model is then:

$$v(T(x)) = F(u(x), q) + \eta \ , \tag{4.1}$$

where η is a random variable that models noise in the imaging process.

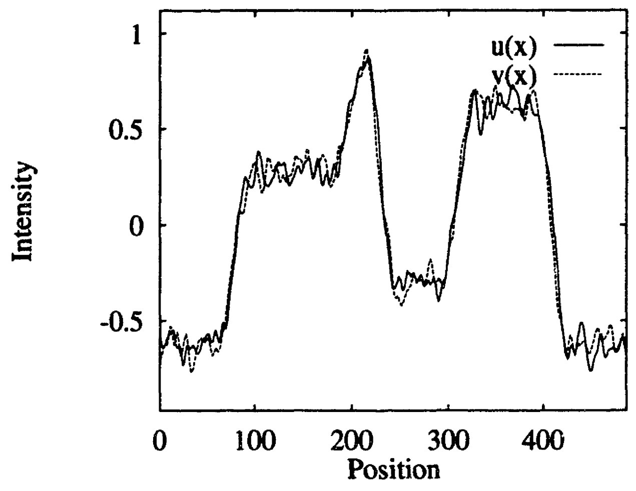


Figure 4.1: Graph of $u(x)$ and $v(x) = u(x) + \eta$ versus x.

For a number of practical problems, the transformation between a model and an image is not known. Alignment is the process by which the correct transformation is extracted. Alignment can be a difficult problem for a number of reasons:

- The imaging function F of the physical world can be difficult to model.
- The exogenous parameters q are not necessarily known and can be difficult to find. For example computing the lighting in an image is not an easy problem.
- The space of transformations can be difficult to search. Transformation space can have many dimensions. Rigid objects often have a 6 dimensional transformation space. Non-rigid objects can in principal have an unbounded number of pose parameters.

A simple example can lend intuition to these definitions. Let $u(x)$ and $v(y)$ be one dimensional signals. Let the transformation space be the space of all possible translations

$$T(x) = x - \beta \ . \tag{4.2}$$

Let the imaging function F be the identity function. Choosing $\beta = 0$ leads to

$$v(x) = u(x) + \eta \ . \tag{4.3}$$

Figure 4.1 contains a graph of two signals that obey this relationship. Though we show the image and model aligned, the correct alignment of v to u is not known.

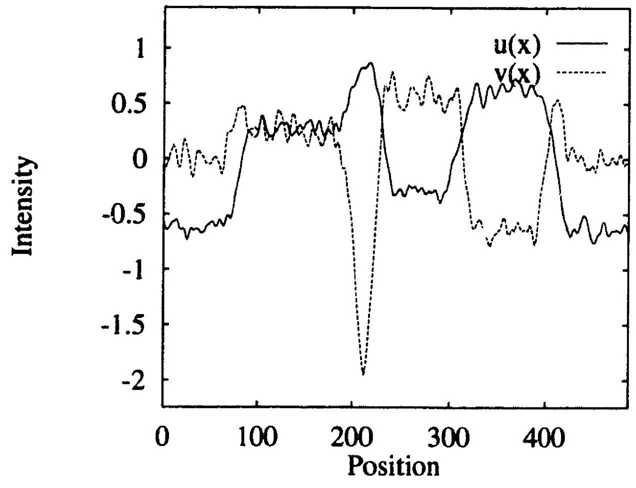


Figure 4.2: Graph of $u(x)$ and $v(x) = (u(x) - 2)^2$ versus x.

In all of our synthetic experiments 10% random noise has been added to v. Noise is of course an unavoidable reality of any real system. But, more importantly the addition of noise demonstrates that the algorithms presented are numerically stable.

More complex imaging functions are possible. For example, F might be non-linear

$$F(u) = -u^2 \ . \tag{4.4}$$

Figure 4.2 contains a graph of the $u(x)$ and $v(T(x)) = F(u(x))$.

4.1.1 Correlation as a Maximum Likelihood Technique

The search for the correct alignment can be cast as a maximum likelihood or variance minimization problem (see Section 2.3.1). The probability of an image given the model, the transformation, the noise distribution, the exogenous parameters, and the imaging function is:

$$p(v \mid u, T, \eta, q, F) = \prod_{x_a \in a} p(\eta = v(T(x_a)) - F(u(x_a), q)) \ . \tag{4.5}$$

In the above equation we have assumed that each pixel of v is conditionally independent. Conditional independence does not imply that the pixels are independent, just that if u, T, η, q, and F are known the pixels are independent. Assuming that the noise is Gaussian, we can then compute the log likelihood of a transformation as

$$
\begin{aligned}
log(\ell(T)) &= log\ p(v \mid u, T, \eta, q, F) && (4.6)\\
&= \sum_{x_a \in a} log\ p(\eta = v(T(x_a)) - F(u(x_a), q)) && (4.7)\\
&= -k_1 \sum_{x_a \in a} (v(T(x_a)) - F(u(x_a), q))^2 && (4.8)\\
&\approx -k_2 E\left[(v(T(X)) - F(u(X), q))^2\right] && (4.9)\\
&\approx -k_2 E\left[v(T(X))^2\right] - 2E\left[v(T(X))F(u(X), q)\right] + E\left[F(u(X), q)^2\right] && (4.10)
\end{aligned}
$$

where k_1 and k_2 are constants computed from the variance of the noise and the number of sample points. They play no role in the maximization. Search can be used to find the transformation that makes the image most likely. Intuitively, the most likely transformation is the one that causes the model to match the image "best". Alternatively one can define the cost of a transformation,

$$
C(T) = -log(\ell(T)) \ , \qquad (4.11)
$$

as the negative of log likelihood. In (4.10) we have expanded the square of the differences to show that the cost of a transformation has three components: one that arises from the variance of the model; a second that arises from the correlation between the image and the predicted image; and a third that arises from the variance of the predicted image. For problems where the variance of the image and predicted image is fixed, the best transformation is the one that maximizes the correlation between the actual and predicted image.

As we did in the analysis of principal components and function learning, we have "invented" random variables: X, $v(T(X))$ and $u(X)$. The random variable X ranges over points from the coordinate system of u where $u(X)$ is defined. The random variables $u(X)$ and $v(T(X))$ range over the values in the model and image. In reality there are no random processes involved in matching and alignment. The model and image are pre-determined and fixed. Alignment could proceed deterministically; the cost of a transformation being evaluated directly from all of the points in the model and image. We have chosen to interpret the summation over pixels that arises in correlation as an expectation over a set of random variables. As a result the insights of probability and statistics can be brought to bear on these problems.

4.1.2 Correlation Maximizes Mutual Information

Alignment is very similar to the problem of function learning that we encountered in Section 3.1. Equation (3.9) is almost identical to (4.6). In both problems we are looking for a set of parameters that cause the inputs to model the outputs. Function learning attempts to find the best parameter vector v; alignment attempts to find the best transformation T. With some judicious manipulation the two problems are identical. As we did for function learning, we can draw an analogy with sample entropy,

$$log(\ell(T)) = -\frac{1}{N_a} h_a(v(T(X)) \mid u(X), T, \eta, q, F) \ . \tag{4.12}$$

This is not the EMMA estimate of entropy, but the conditional entropy of v under the assumption that v is conditionally Gaussian. We cannot claim however that maximizing log likelihood is equivalent to maximizing the mutual information between v and u. The mutual information

$$I(v(T(X)), u(X)) = h(v(T(X))) - h(v(T(X)) \mid u(X), T, \eta, q, F) \ , \tag{4.13}$$

has both a conditioned and unconditioned entropy. For some types of transformations the unconditioned entropy may not be constant as T is varied. In these cases minimizing conditional entropy is not equivalent to maximizing mutual information. One must also maximize the unconditioned entropy. In our simple example, where only translation is varied and the signals are periodic, unconditioned entropy does not change as T is varied.

Returning to the first synthetic example, we can plot $C(T)$ versus translation. Figure 4.3 graphs $C(T)$ for the two signals from Figure 4.1 (we have assumed periodic boundary conditions on the signals). We can see that cost has a very strong minima at the true translation of 0 pixels.

Correlation works very well at matching together u and v when the imaging model and exogenous parameters are known. In many cases however we may be faced with a situation where F and q are *unknown*. In some cases alignment

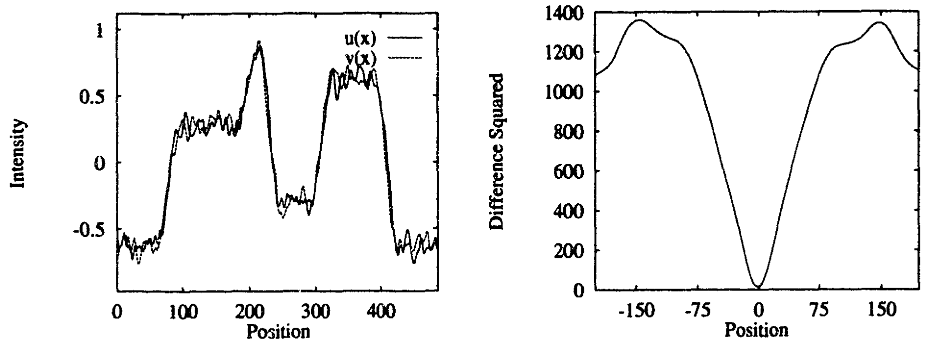


Figure 4.3: On the left is a plot of image and model that are identical except for noise. On the right is a plot of $C(T)$ versus translation. There is a very significant minimum at the correct aligning translation 0 pixels.

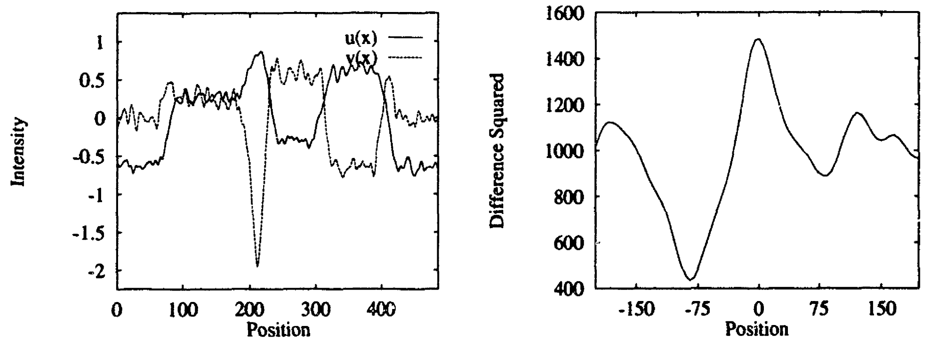


Figure 4.4: On the left is a plot of image and model that are related non-linearly. On the right is a plot of $C(T)$ versus translation. There is no minima at the aligning translation 0 pixels. In fact minima exist at incorrect translations.

problems can still be solved by assuming that the imaging function is the identity function. This assumption is not effective when aligning the non-monotonically related signals shown in Figure 4.2. Figure 4.4 graphs $C(T)$ versus translation for these two signals. Notice that there is no convincing minima anywhere and that each of the actual minima are at incorrect translations.

In general $C(T)$ cannot be used to align signals that have been transformed by unknown non-linear transformations of signals. $C(T)$ can however be generalized to work with signals that have been transformed linearly. Rather than minimize the squared difference between signals, we can instead minimize the squared difference between signals that have been normalized. A normalized signal is one with a mean of zero and a standard deviation of one and can be computed as

$$\hat{u}(x) = \frac{u(x) - E[u(X)]}{\sigma(u(X))} \ . \tag{4.14}$$

The normalized version of a signal is invariant to multiplicative and additive changes to the original. The sum of the squared differences between the normalized signals, $NC(T)$, can be computed directly as one minus the *Normalized correlation* between the signals u and v. *Normalized correlation* is defined as:

$$N(u,v) = \frac{E_a[u(X)v(T(X))] - E_a[u(X)]E_a[v(T(X))]}{\sigma_a(u(X))\sigma_a(v(T(X)))} \ . \tag{4.15}$$

As a shorthand we have abbreviated sums over the coordinates x as expectations and variances.

Normalized cost can be used on signals like the ones shown in Figure 4.5 ($F(u) = 3(u-2)$). A plot of $NC(T)$ versus translation has the same form as Figure 4.3. In some cases, normalized cost can be applied to signals transformed by non-linear *monotonic* functions. The two signals shown in Figure 4.2 are related by a non-monotonic function and cannot be accommodated in this way. For this simplified problem of finding the correct aligning translation, normalized cost can never produce a convincing minimum where cost alone does not. Since translation does not effect the mean or the standard deviation of the model or the image, normalized cost is just a shifted and scaled version of cost. No amount of shifting and scaling will create a strong minimum where one does not exist.

We may still wish to align models and images that are related by non-monotonic

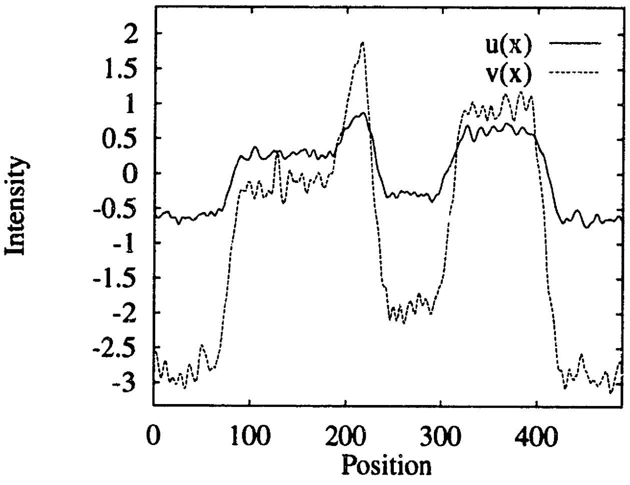


Figure 4.5: Graph of $u(x)$ and $v(x) = 3(u(x) - 2)$ versus x.

functions. In this case alignment can be performed by jointly searching over the space of possible imaging functions, exogenous parameters and transformations. Probability can be used to motivate this procedure. We can evaluate $p(v \mid u, T, N, q, F)$ when F and q are unknown by integrating out the unknown variables. The probability of the image would then be,

$$p(v \mid u, T, \eta) = \int \int \prod_{x_a \in a} p(\eta = v(T(x_a)) - F(u(x_a), q))\, p(F)\, p(q)\, dF\, dq \ . \quad (4.16)$$

This equation directs us to integrate over all possible imaging functions and all possible sets of exogenous variables. We are aware of no approach that has come close to evaluating such an integral. It may not be feasible. Another possible approach is to find the imaging function and exogenous variables that make the image most likely,

$$p(v \mid u, T, N) \approx \max_{F,q} \prod_{x_a \in a} p(\eta = v(T(x_a)) - F(u(x_a), q))\, p(F)\, p(q) \ . \quad (4.17)$$

Here we have assumed that the integral in Equation 4.16 is approximated by the component of the integrand that is maximal. The approximation is a good one when a particular F and q are much more likely than any other. This equation directs us to repeat a pair of nested searches: (1) given an estimate for the transformation, find F and q that make the image most likely; (2) given new estimates for F and q, find a new transformation that makes the image most likely. Terminate when the transformation has stabilized.

In summary, a transformation associates points from the model with points in the image; for every $u(x)$ there is a corresponding $v(T(x))$. A function F and parameter vector q are sought that best model the relationship between $u(x)$ and $v(T(x))$. This can be accomplished by "training" a function to fit the collection of pairs $\{v(T(x_a)), u(x_a)\}$. Algorithms for finding F and q are very similar to the those for density approximation and learning described in Chapter 3. Once F and q are found they can be used to further optimize the transformation.

The pitfalls of density approximation as described in Chapter 2 as apply to function approximation. Before we can hope to learn the function F we must first make a set of assumptions about the form of F. Without these assumptions outrageous estimates for F can prevent convergence or lead to incorrect answers. One way to prevent, or discourage, this behavior is to formulate a strong prior over the space of functions, $p(F)$.

In many cases the search for imaging functions and exogenous parameters can be combined. For any particular F and q, another function $F_q(u(x)) = F(u(x), q)$ could be defined. Combining functions like this is common in shape from shading and photometric stereo. Both techniques compute the shape of an object from the shading that is present in an image or images. Rather than independently model the exogenous variable (the lighting direction) and imaging function (the reflectance function) a combined function is represented and manipulated. The combined function is called a *reflectance map* (Horn, 1986). It maps the normals of an object directly into intensities. The three dimensional alignment procedure we will describe manipulates a similar combined function. By simplifying the search and focusing on F_q we can see that alignment without an imaging model is very similar to entropy maximization. Entropy maximization is a nested search for a density estimate and parameters. Alignment is a nested search for an imaging model and a transformation. We will return to this analogy shortly.

How might Equation 4.17 be approximated efficiently? It seems reasonable to assume that for most real imaging functions similar inputs yield similar outputs. In other words, that the unknown imaging function is continuous and piecewise smooth. An efficient scheme for alignment could skip the step of approximating the imaging function and attempt to directly evaluate the *consistency* of a transformation. A transformation is considered consistent if points that have similar values in the model project to similar values in the image. By similar we do not mean similar in physical location, as in $|x_j - x_i|$, but similar in value, $|u(x_j) - u(x_i)|$ and $|v(T(x_j) - v(T(x_i))|$. One ad-hoc technique for estimating consistency is to pick a similarity constant ϵ and evaluate the following sum:

$$\text{Consistency}_1(T) = -\sum(v(T(x_i)) - v(T(x_j)))^2 \ , \tag{4.18}$$

where the sum is over i and j such that $|u(x_i) - u(x_j)| < \epsilon$. Consistency is flawed in a number of ways. For instance, there are no obvious clues of picking ϵ. We can replace the "all or nothing" nature of the ϵ test with a more gradual discrimination:

$$\text{Consistency}_2(T) = -\sum_{i,j} g_\psi(u(x_i) - u(x_j))(v(T(x_i)) - v(T(x_j)))^2 \ , \tag{4.19}$$

where g_ψ is a Gaussian with standard deviation, ψ. In order to minimize this measure points that are close together must be more consistent, and those further apart less so. Another problem with any consistency measure is that it is too aggressive; consistency is maximized by constancy. The most consistent transformation projects the points of the model onto a constant region of the image. For example, if scale is one of the transformation parameters, one entirely consistent transformation projects all of the points of the model down to a single point of the image. Though there are a number of problems with consistency that need to be addressed, it will serve as source of intuition as we analyze different approaches.

We now have two alternatives for alignment in the absence of an imaging function: a theoretical technique that may be intractable, and an outwardly efficient technique that has a number of important difficulties. Perhaps we can construct a technique with elements of both. The intuition of consistency could be applied to the task of function approximation. One type of function estimator that maximizes consistency is known as a nearest neighbor function approximation(Duda and Hart, 1973):

$$F_q(u(x_i), u, v) = v(T(x_j)) \ \text{ such that } j = \operatorname*{argmin}_{k \neq i} |u(x_i) - u(x_k)| \ . \tag{4.20}$$

There is no longer any need to search for F_q; the model, image and transformation define it directly. In this sense, the nearest neighbor approximation is very much like the Parzen estimate (see Section 2.4.3). Unlike the Parzen estimate, the nearest neighbor approximation is not continuously differentiable. A differentiable version is called *weighted neighbor* or *basis function* approximation:

$$F^*(u(x_i), u, v) = \sum_j \frac{R(u(x_i) - u(x_j))v(T(x_j))}{\sum_k R(u(x_i) - u(x_k))} \ . \tag{4.21}$$

The weighting function R usually has a maximum at zero, and falls off asymptotically away from zero. A common choice for R is the Gaussian density function g_ψ with standard deviation ψ. We can re-write F^* as,

$$F^*(u(x_i), u, v) = \sum_j W_u(u(x_i), u(x_j))v(T(x_j)) \ . \tag{4.22}$$

where W is the soft nearest neighbor function first defined in Section 3.2,

$$W_u(u(x_i), u(x_j)) \equiv \frac{g_\psi(u(x_i) - u(x_j))}{\sum_k g_\psi(u(x_i) - u(x_k))} \ .$$

An estimate for the log likelihood is then,

$$\begin{aligned}
log(\ell(T)) &= \sum_i [v(T(x_i)) - F^*(u(x_i), u, v)]^2 &(4.23)\\
&= \sum_i \left[v(T(x_i)) - \sum_j W_u(u(x_i), u(x_j))v(T(x_j)) \right]^2 &(4.24)\\
&= \sum_i \left[\sum_j W_u(u(x_i), u(x_j))v(T(x_i)) - \sum_j W_u(u(x_i), u(x_j))v(T(x_j)) \right]^2 &(4.25)\\
&= \sum_i \left[\sum_j W_u(u(x_i), u(x_j))(v(T(x_i)) - v(T(x_j))) \right]^2 \ . &(4.26)
\end{aligned}$$

Step (4.25) relies on the fact that

$$\sum_j W_u(u(x_i), u(x_j)) = 1 \ .$$

The log likelihood of a transformation using a weighted neighbor function approximation is very similar to the intuitive consistency measure (4.19). In addition its derivative bears a striking resemblance to the derivative of the EMMA estimate

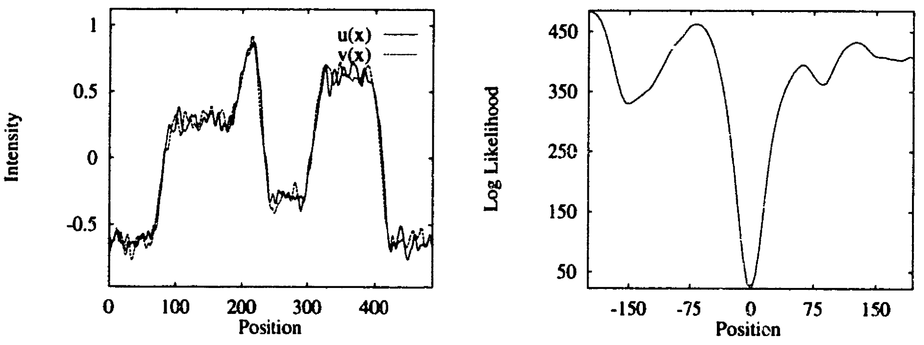


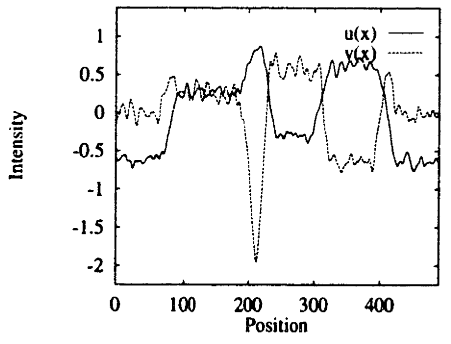
Figure 4.6: On the left is a plot of image and model that are identical except for noise. On the right is a plot of the logarithm of weighted neighbor likelihood versus translation.

(see (3.20)):

$$
\begin{aligned}
&\frac{d}{dT} log(\ell(T)) && (4.27)\\
&= \sum_i \frac{\sum_j \left(g_\psi(u(x_i) - u(x_j))^2 (v(T(x_i)) - v(T(x_j)))(\frac{d}{dT} v(T(x_i)) - \frac{d}{dT} v(T(x_j)))\right)}{\left(\sum_j g_\psi(u(x_i) - u(x_j))\right)^2} && (4.28)\\
&= \sum_i \sum_j W_a(u(x_i), u(x_j))^2 \left((v(T(x_i)) - v(T(x_j))) \frac{d}{dT}(v(T(x_i)) - v(T(x_j))\right) && (4.29)
\end{aligned}
$$

Weighted neighbor likelihood can be used to evaluate the cost of different translations. Figure 4.6 shows a graph of weighted neighbor likelihood versus translation for the initial pair of signals, $u(x)$ and $v(x) = u(x) + \eta$. Figure 4.7 contains a similar graph for the second, non-linear experiment, $u(x)$ and $v(x) = (u(x) - 2)^2 + \eta$. Both graphs show a strong minimum at the correct alignment of the signals. We can conclude that weighted neighbor likelihood can be used in situations where neither cost nor normalized cost would work.

The parallel between EMMA and weighted neighbor likelihood is more than structural. Algorithmically, EMMA and weighted neighbor alignment are very similar: EMMA estimates the density of the sample directly and uses it to compute the derivative of entropy with respect to the parameter vector; weighted neighbor likelihood estimates the imaging function directly and uses it to compute the derivative of log likelihood with respect to the transformation. More importantly both techniques manipulate the entropy of the of the joint distribution u and v. EMMA can be used to evaluate the joint entropy $h^*(v(T(x)), u(X))$;



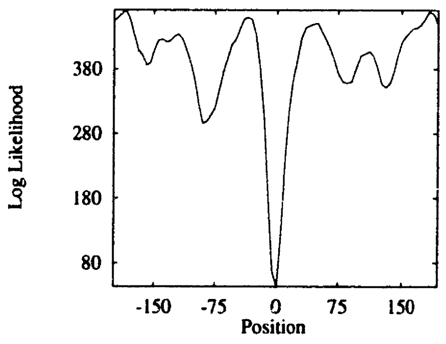


Figure 4.7: On the left is a plot of image and model that are related non-linearly. On the right is a plot of the logarithm of weighted neighbor likelihood versus translation.

weighted neighbor likelihood evaluates the conditional entropy $h(v(T(x)) \mid u(X))$ under the assumption that $p(v(T(x))|u(X), u, v) = g_\psi(v(T(x)) - F^*(u(x), u, v))$ (see Sections 2.3.1 and 4.1.2 commentary on the equivalence of log likelihood and sample entropy).

We can relax the constraint that v be conditionally Gaussian by using EMMA to estimate the conditional entropy:

$$h^*(v(T(x))|u(X)) = h^*(u(x)) - h^*(v(T(x)), u(X)) \ . \tag{4.30}$$

The equation above implies that an estimate for the likelihood that v and u are functionally related can be computed from entropies. The first term is the entropy of the model. It is not a function of the transformation. The joint entropy $h^*(v(T(x)), u(X))$ is equal to the entropy of $h^*(w)$, where $w = [v(T(x)), u(X)]^T$.

Why are these two seemingly unrelated concepts, weighted neighbor likelihood and conditional entropy so closely related? We can gain some intuition about this equivalence by looking at the joint distribution of u and v. For any particular transformation we can sample points $[u(x_i), v(T(x_i))]^T$ and plot them. Figure 4.8 shows the joint samples of the signals from Figure 4.1 when aligned. There are several points to note about this graph: the thin line in the plot is the weighted neighbor function estimation of this data, it is a good fit to the data; there is a noticeable clumping, or clustering, in the data. These clumps arise from the regions of almost constant intensity in the signals. There are four large regions of constant intensity and four clusters.

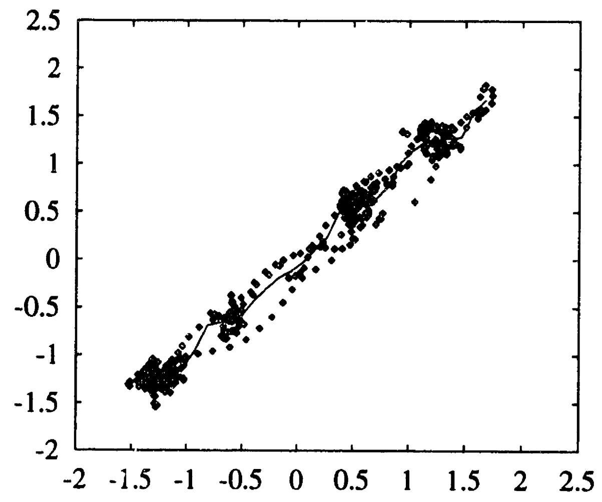


Figure 4.8: Samples from the joint space of $u(x)$ and $v(x) = u(x) + \eta$. A small black square is plotted for every pixel in the signals. The X-axis is the value of $u(x)$. The Y-axis is the value of $v(x)$. The clumping of points in clusters is caused by the regions of almost constant intensity in the images. The thin line plotted through the data is the weighted neighbor function estimate.

When these almost identical signals are aligned they are strongly correlated. Large values in one signal corresponds to large values of the other. Conversely, small values in one correspond to small values in the other. Correlation measures the tendency of the data to lie along the line $x = y$ (normalized correlation measures the tendency of the data to lie along some line of positive slope). Figure 4.9 shows the joint samples of two signals shown in Figure 4.2. These signals are not linearly related or even correlated, but they are functionally related.

Weighted neighbor likelihood measures the quality of the weighted neighbor function approximation. In both of these graphs the points of the sample lie near the weighted neighbor function approximation. Moreover, in both of these graphs the joint distribution of samples is tightly packed together. Points are not distributed throughout the space, but lie instead in a small part of the joint space. This is the hallmark of a low entropy distribution.

We can generate similar graphs for signals that are not aligned. Figures 4.10 and 4.11 show the same signals except for the fact that the image has been shifted 30 units. For these shifted signals the structure of the joint distribution is destroyed. There is no good way to fit these distributions with a function. We can see that the weighted neighbor function approximation is a terrible fit to the data. As a

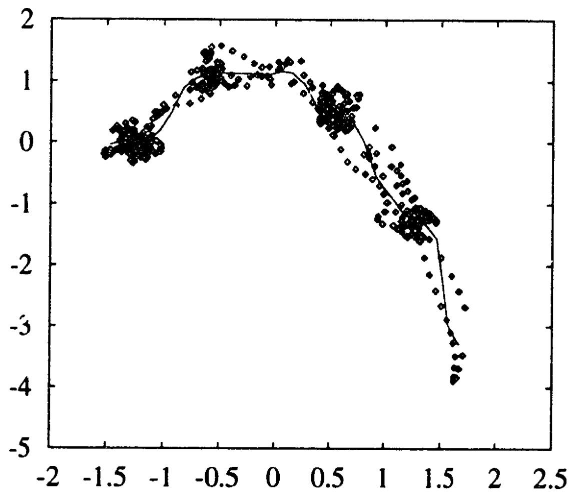


Figure 4.9: Samples from the joint space of $u(x)$ and $v(x) = u(x)^2 + \eta$.

result the weighted neighbor likelihood of these signals is low.

Alternatively we could look directly at the distributions. When the signals are aligned the distributions are compact. When they are misaligned the distributions are spread out and haphazard. Or in other words, aligned signals have low joint entropy and misaligned signals have high joint entropy.

This suggests an alternative to weighted neighbor likelihood: the EMMA approximation of joint entropy. Graphed below are the EMMA estimates of joint entropy, $h^*(w)$, versus translation for each signal alignment problems discussed. Figure 4.12 shows a graph of joint entropy for the two signals that are nearly identical. Figure 4.13 shows a graph of joint entropy for the model and the non-linearly transformed image. In both case the graphs show strong minima at the true translation of 0 units.

4.2 Weighted Neighbor Likelihood vs. EMMA

Weighted neighbor likelihood and EMMA are both smoothly differentiable functions that can be used to align signals when the imaging function is unknown. Qualitatively the EMMA estimate of joint entropy seems better. Joint entropy seems to have a wider basin in these synthetic experiments.

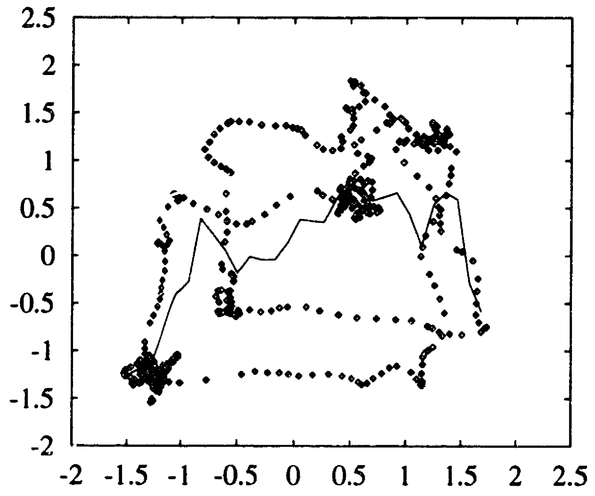


Figure 4.10: Samples from the joint space of $u(x)$ and $v(x) = u(x+30)+\eta$. Unlike the previous graph these two signal are no longer aligned.

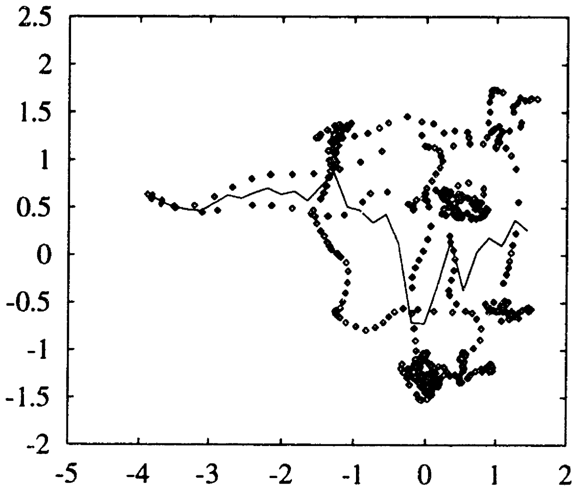


Figure 4.11: Samples from the joint space of $u(x)$ and $v(x) = -u(x+30)^2 + \eta$. The two signals are not aligned.

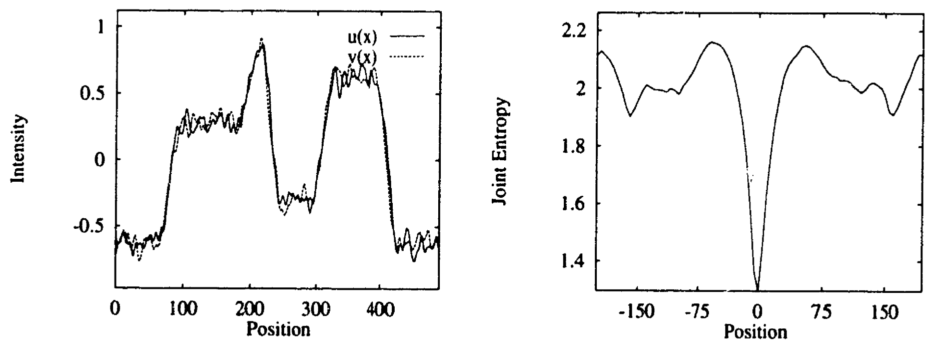


Figure 4.12: On the left is a plot of image and model that are identical except for noise. On the right is a plot of the estimated joint entropy versus translation.

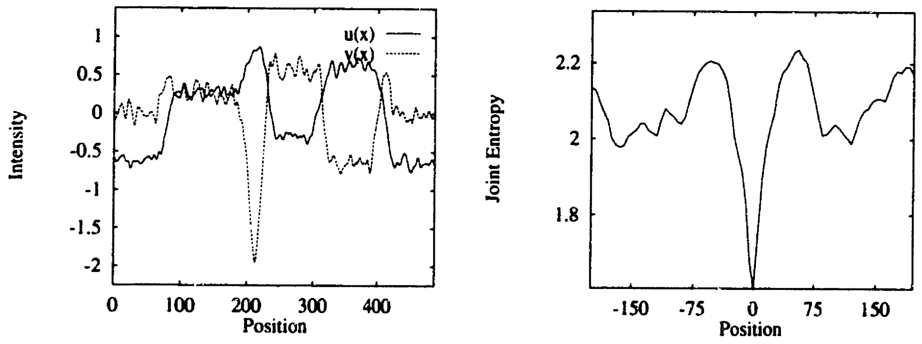


Figure 4.13: On the left is a plot of image and model that are related non-linearly. On the right is a plot of estimated joint entropy versus translation.

If weighted neighbor likelihood and EMMA are so similar, why is there a difference? Recall that weighted neighbor likelihood measures the conditional entropy of the image given the model. It does this under the assumption that the conditional distribution of the image is Gaussian. In other words, the local data is used to estimate the mean of a Gaussian. The log likelihood of a point is proportional to the squared difference from this mean. In general log likelihood calculations are very sensitive to *outliers*. Outliers are points that are, because of noise or measurement error, perturbed and land far from where they should have. Recall that the log likelihood of a sample is the sum of the log likelihoods of each point in the sample. As a result a single outlier can ruin a sample that would otherwise have had a high likelihood.

A more reasonable measure might introduce a bound on the penalty for a single point. Once a single point moved beyond a certain distance from the local mean the cost would no longer increase. Calculating likelihood in this way is closely related to the concept of a *robust* statistic ***reference here***.

EMMA on the other hand does not assume that the conditional distribution of the image given the model is Gaussian. It approximates the density non-parametrically. EMMA can handle situations where there are multiple peaks in the conditional distribution. While there is a likelihood penalty if a group of points are perturbed away from the local mean, it is not a function of the distance from the mean. Once these points move outside the effective range of the smoothing function there is no additional penalty.

In the next sections we will describe a number of situations where EMMA is better than weighted neighbor likelihood.

4.2.1 Non-functional Signals

Up until this point our analysis of alignment has assumed that there exists an imaging function that relates the model and the image. For at least two classes of problems there will be no imaging function at all. The first arises from a common situation in computer vision: occlusion. The second arises when the model does not contain all of the information required to predict the image. In both cases no single function, regardless of exogenous variables, can be used to predict the image from the model.

Figure 4.14 shows a graph of our original pair of signals, except that $v(x)$ has now been corrupted by an occlusion. Occlusion proves to be particularly bothersome for the alignment techniques we have proposed. The basic assumption behind normalized cost have been violated. The occluded signal is not a linearly transformed version of the model. A quick glance at the joint space shows that the assumption behind weighted neighbor likelihood has also been violated (see Figure 4.15). Even when the signals are aligned, there is no longer any function that relates $u(x)$ and $v(x)$. Figure 4.16 show a graph of weighted neighbor likelihood versus translation. There is no convincing minimum when the signals are correctly aligned. In some cases joint entropy can be used to align partially occluded signals. Joint entropy does not suffer from the strong assumption that the signals are functionally related. Though part of the signal may be corrupted the remaining parts retain their low entropy alignment. Figure 4.17 show a plot of joint entropy for the occluded pair of signals.

The simplest example of non-functional signals often arises when the model and the image are swapped. Whenever the function between the model and the image is non-monotonic, the relationship between the image and the model is non-functional. The non-monotonically related signals shown in Figure 4.2 are an example. Figure 4.18 shows the joint space of the swapped signals and a weighted neighbor function approximation. The function fit to this joint space is a terrible approximation of the data. The quality of the function approximation points out an important limitation of weighted neighbor likelihood. While normalized cost is

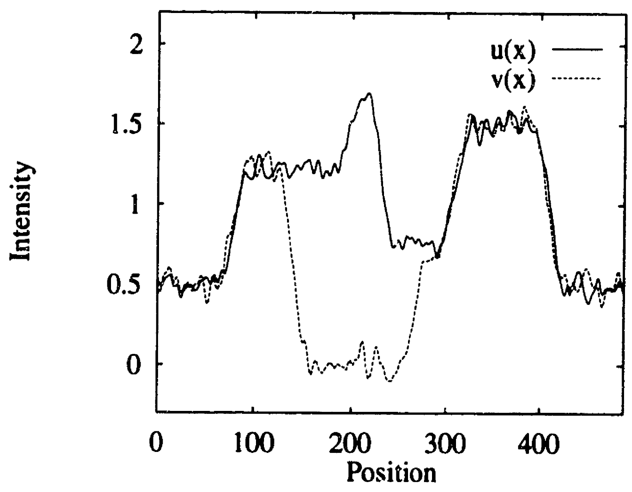


Figure 4.14: Graph of $u(x)$ and $v(x)$ where $v(x)$ has been perturbed by noise, and a portion of it occluded.

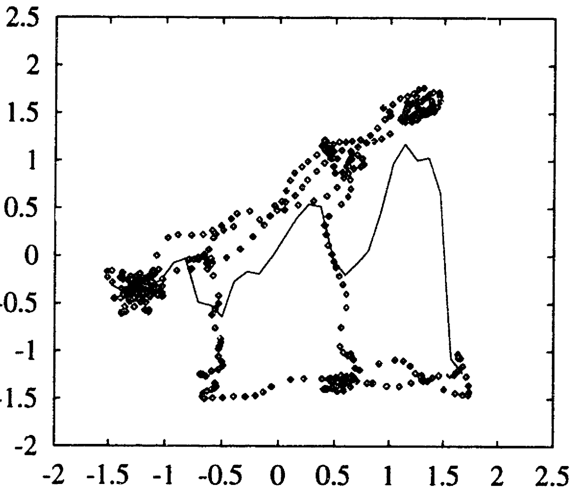


Figure 4.15: Samples from the joint space of $u(x)$ and $v(x)$ where the image has been occluded. Though these signals are aligned the the weighted neighbor function is a terrible fit to the data. The data is segregated into two parts: the linearly related part that arise from the non-occluded signal, and the constant part that projects to the occluded part.

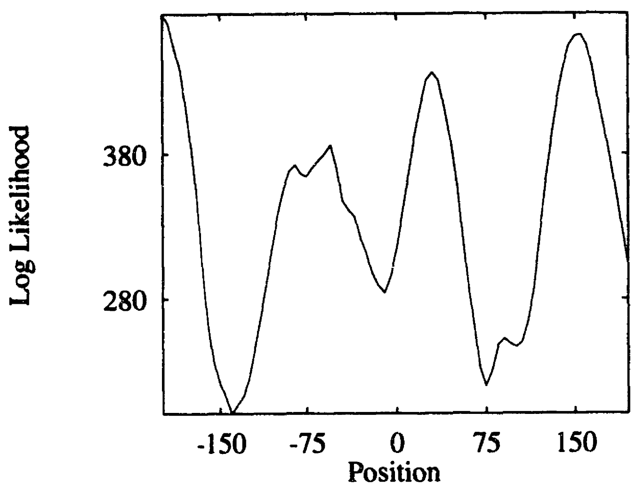


Figure 4.16: Graph of weighted neighbor likelihood versus translation where $v(x)$ has been occluded.

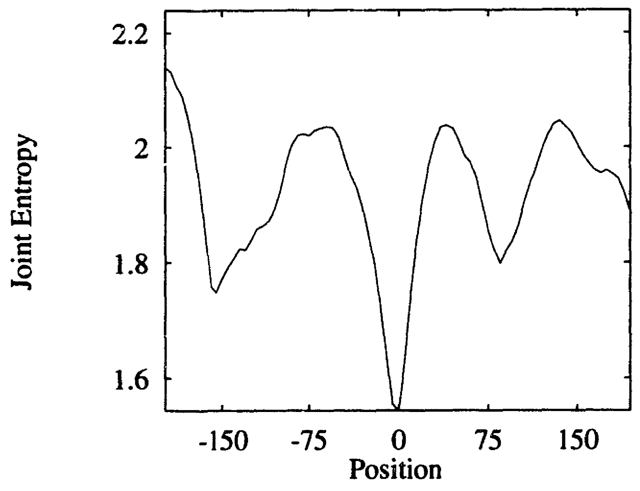


Figure 4.17: Graph of EMMA joint entropy estimate versus translation for the for the occluded pair of signals.

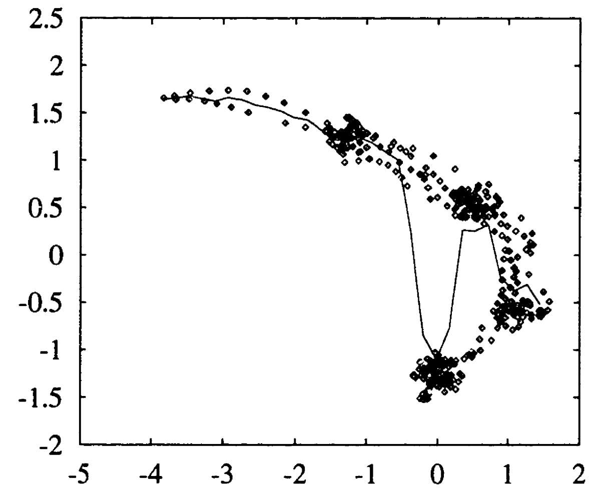


Figure 4.18: Samples from the joint space of $u(x)$ and $v(x) = (u(x) - 2)^2 + \eta$. The roles of the model and the image have been reversed.

a *symmetric* comparison metric, weighted neighbor likelihood is *not*. It may seem at first that this is an unimportant distinction. It is not. Symmetric measures allow us to match images to models as well as models to images. This can be critical when it is not possible to construct a detailed model.

Both joint entropy and mutual information are symmetric measures. A plot of EMMA entropy for the swapped signals is identical to Figure 4.13.

A more complex example of non-functional signals arises when both the model and the image are functions of some third unmeasurable signal. Call this signal $z(x)$. There are now two imaging functions, one that creates the model $u(x) = F_u(z(x), q_u)$, and another that creates the image $v(T(x)) = F_v(z(x), q_v)$. The "two sensor" problem is fairly common for real data. Though a model is constructed to represent an object very accurately, it is often impossible to build an absolutely correct model. The sensor that observes the object is designed to deliver the most information possible, but no sensor is perfect. The problem of medical registration, which we describe in some detail in the next chapter, is a clear example of two sensor problem. In medical registration one seeks an alignment of signals from two types of sensors (for example a CT scan and an MRI scan). Neither gives perfect information about the object, and neither is completely predictable from the other.

The two sensor problem can be simulated by transforming our original signal by two different non-linear transforms. Using the original signal from Figure 4.1

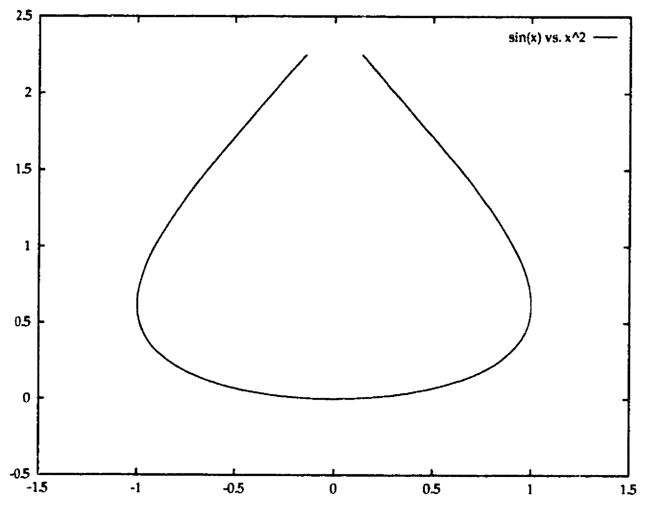


Figure 4.19: Graph of the function defined by $u = sin(2z)$ and $v = z^2$ as z varies from -1.5 to 1.5.

we can define $F_u(z) = sin(2z)$ and $F_v(z) = z^2$. The resulting aligned distribution should fall approximately along a line that looks like Figure 4.19. The actual distribution is shown in Figure 4.20. Here too EMMA shows a strong minimum at the correct alignment of model and image (see Figure 4.21).

Mutual Information versus Joint Entropy

In these simple examples joint entropy or conditional entropy can be used as a measure of alignment. In more complex examples, where the model can change scale or project on a limited part of the image, the entropy of the image itself must be taken into account.

Recall that conditional entropy is not a measure of the dependence between two signals (see Section 2.2). Conditional entropy $h(v|u)$ can be small for two different reasons: $h(v)$ is itself small, or v is dependent on u. Mutual information is a better measure of dependence. In the simple examples above the entropy of the image does not change as the transformation is varied. In other problems the entropy could well vary. We will solve alignment problems by maximizing mutual information. This involves not only minimizing conditional entropy but also maximizing the entropy of the image.

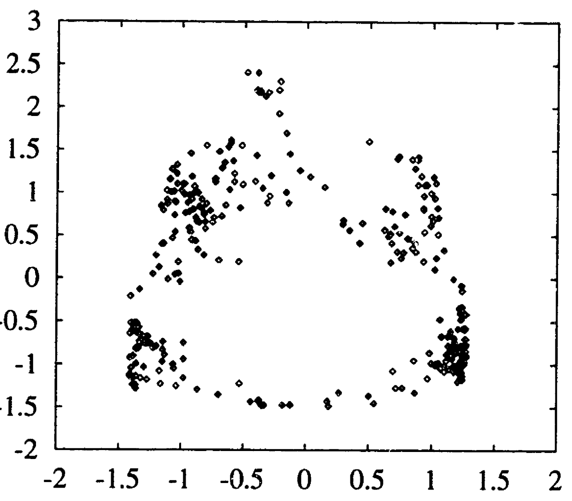


Figure 4.20: Samples from the joint space of from the simulated two sensory data.

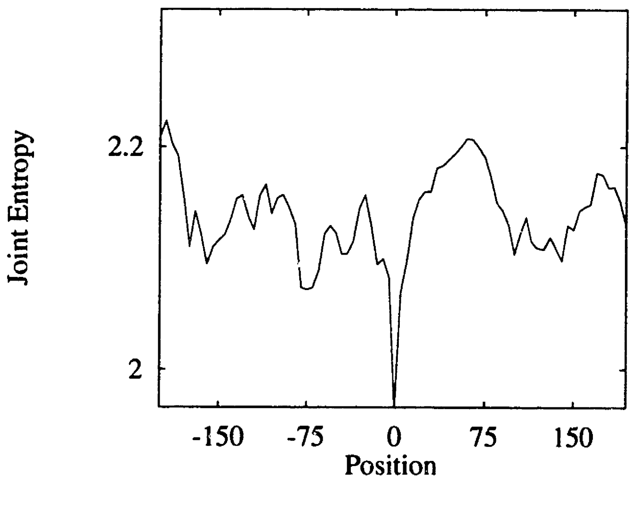


Figure 4.21: Graph of EMMA joint entropy estimate versus translation for the for the non-functional pair of signals.

4.3 Alignment Derivation

Let us derive the equations and algorithms used for alignment by maximization of mutual information. We wish to maximize the mutual information between model and image signals. This requires a search of the aligning transformation space. We will use the stochastic gradient descent algorithm described in Section 3.3.

The derivative of mutual information is

$$\frac{d}{dT}I(u(x),v(T(x))) = \frac{d}{dT}h(u(x)) + \frac{d}{dT}h(v(T(x))) - \frac{d}{dT}h(u(x),v(T(x))) \ .$$

Since $h(u(x))$ is not a function of T, it drops from our calculations. Using the EMMA entropy estimate,

$$\frac{d}{dT}I(u(x),v(T(x)) \approx \frac{1}{N_B}\sum_{x_i \in B}\sum_{x_j \in A} W_v(v_i,v_j)\ (v_i - v_j)^T \psi_v^{-1} \frac{d}{dT}(v_i - v_j) \quad (4.31)$$

$$-\frac{1}{N_B}\sum_{x_i \in B}\sum_{x_j \in A} W_{uv}(w_i,w_j)\ (w_i - w_j)^T \psi_{uv}^{-1} \frac{d}{dT}(w_i - w_j) \quad (4.32)$$

The following definitions have been used:

$$W_v(v_i,v_j) \equiv \frac{g_{\psi_v}(v_i - v_j)}{\sum_{x_k \in A} g_{\psi_v}(v_i - v_k)} \ ,$$

$$W_{uv}(w_i,w_j) \equiv \frac{g_{\psi_{uv}}(w_i - w_j)}{\sum_{x_k \in A} g_{\psi_{uv}}(w_i - w_k)} \ ,$$

$$u_i \equiv u(x_i) \ , \ u_j \equiv u(x_j) \ , \ u_k \equiv u(x_k) \ ,$$

$$v_i \equiv v(T(x_i)) \ , \ v_j \equiv v(T(x_j)) \ , \ v_k \equiv v(T(x_k)) \ ,$$

$$w_i \equiv [u_i, v_i]^T \ , \ w_j \equiv [u_j, v_j]^T \ , \text{ and } w_k \equiv [u_k, v_k]^T \ .$$

We assume that the covariance matrices of the component densities used in the approximation scheme for the joint density are block diagonal,

$$\psi_{uv}^{-1} = \mathrm{DIAG}(\psi_{uu}^{-1}, \psi_{vv}^{-1}) \ ,$$

and we obtain an estimate for the derivative of the mutual information as follows

$$\widehat{\frac{dI}{dT}} = \frac{1}{N_B} \sum_{x_i \in B} \sum_{x_j \in A} (v_i, v_j)^T [W_v(v_i, v_j)\psi_v^{-1} - W_{uv}(w_i, w_j)\psi_{vv}^{-1}] \frac{d}{dT}(v_i - v_j) \ .$$

If we are to increase the mutual information, then the effect of the first term in the brackets may be interpreted as acting to increase the squared distance between pairs of samples that are nearby in image intensity, while the second term acts to decrease the squared distance between pairs of samples that are nearby in *both* image intensity and the model properties. It is important to emphasize that distances are in the space of values (intensities, brightness, or surface properties), rather than coordinate locations.

The term $\frac{d}{dT}(v_i - v_j)$ will generally involve gradients of the image intensities and the derivative of transformed coordinates with respect to the transformation. In the simple case that T is a linear operator, we obtain the following outer product expression,

$$\frac{d}{dT} v(T(x_i)) = \nabla v(T(x_i)) x_i^T \ .$$

4.4 Matching and Minimum Description Length

There is another entirely different motivation for using mutual information as a alignment metric. Alignment, and many other vision problems, can be reformulated as *minimum description length* (MDL) problems (Rissanen, 1978; Leclerc, 1988). MDL can provide us with some new insight into the problem of alignment and help us derive a missing and often useful term in the alignment equations.

The standard framework of MDL involves a sender and a receiver communicating descriptions of images. Given that the sender and the receiver have an agreed upon language for describing images, the sender's goal is to find a message that will accurately describe an image in the fewest bits. The concept of description length is clearly related to the code length introduces in Section 2.2.

For the problem of alignment we will assume that the sender and the receiver share the same set of object models. The sender's goal is to communicate an image of one of these models. Knowing nothing else, the sender could simply ignore the models and send a message describing the entire image. This would require a message at least as long as the entropy of the image. However, whenever the image is an observation of a model a more efficient approach is possible. For example the sender could send the pose of the model and a description for how to render it. From these the receiver can reconstruct the part of the original image in which the model lies. To send the entire image, the sender need only encode the errors in this reconstruction, if there are any, and any part of the image that is unexplained by the model. Alignment can be thought of as the process by which the sender attempts to find the model pose that minimizes the code length of the overall message.

The overall message has several parts: (1) a message describing the pose; (2) a message describing the imaging function; (3) a message describing the errors in the reconstruction; and (4) a message describing the parts of the image unexplained by the model. The length of each part of the message is proportional to its entropy. We can assume that poses are uniformly distributed, and that sending a pose incurs some small uniform cost. The length of part (4) is the entropy of the image that is unexplained. Parts (2) and (3) can be interpreted in two ways. We can assume that the imaging function can be sent with a fixed or small cost. Part (3) is then proportional to the conditional entropy of the image given the model and imaging function. This is precisely what was estimated and minimized with weighted neighbor alignment. A second interpretation comes from EMMA. EMMA estimates the joint entropy of the model and image, $h(u,v)$. The conditional entropy of the image given the model can be computed as $h(v|u) = h(u,v) - h(u)$. Since the entropy of the model is fixed, minimizing the joint entropy minimizes the conditional entropy. In both cases entropy based alignment minimizes the cost of sending parts (1), (2) and (3). MDL suggests that we must also minimize the entropy of the unmodeled part of the image.

Are mutual information and MDL equivalent? Not quite. The entropy of v is proportional to the length of a message describing a single sample of v. The entropy of an image is the entropy of a collection of samples. If we wanted to send a pixel of a binary image where half of the pixels where one and the other half zero, it would take a single bit. Clearly the cost of sending the entire image is much higher. Naively we might say the cost of sending the whole image is equal to the number of pixels times the cost of sending each pixel. As we saw in Section 2.2, this naive intuition can be wrong. The pixels and the image are both random variables over which we can compute entropies. An event of the image RV is a collection of pixel events. The entropy of the pixel RV is proportional to the length of the message that would be required to communicate a single pixel in isolation. The entropy of the image random variable is proportional to the length of the message that describes the whole image. The two are only directly related when the pixels are independent. The image entropy is then n times the pixel entropy. Otherwise one must take the conditional probabilities between the pixels into account.

A quick thought experiment makes this much more clear. One could take an image and compute the entropy of the pixels. This could be done by taking statistics of the pixels. A message that describes a particular pixel is proportional to that entropy. Now let's assume that all of the pixels in the image but one have already been transmitted. How much unpredictability, and therefore entropy, is left about the untransmitted pixel? If our image tended to be smoothly varying, as most images are, we could guess that the untransmitted pixel was a smoothed version of the neighboring pixels. If our receiver did this, and the sender knew that she had, then the sender need only send the difference between the smoothed estimate and the actual value. In practice such a scheme works very well and underlies the process of pyramid compression (Burt and Adelson, 1983).

In our previous information theoretic formulation we had no concept of pixels or of the proportion of the image explained by the model. In fact, in the previous formulation the entropy of the explained part of the image could get larger as the model shrunk. For example, let us assume that our model covers a contiguous region of an image where most of the pixels have constant value. At the center of this region is a small patch containing varied pixels. Recall that the image is sampled at points that are projected from the model. Most of the model points will project into the region of constant intensity and a few will project onto the varied patch. The resulting distribution of image pixels, because it has many samples of the same value, has fairly low entropy. If we were to shrink the model to cover only the varied patch, then all of the points from the model would fall in the varied region. The new distribution of pixel values will have higher entropy.

What if we were to measure the mutual information between the model and the entire image? This would be a more direct parallel to the MDL framework. MDL directs the sender to send the model and pose that will produce an encoding for the entire image that is shortest. As the model explains more, less of the image needs to be encoded directly. We can derive an approximation to mutual information between the model and the entire image. Let us define two new variables, U and V that are the whole model and whole image respectively. We can now rewrite the mutual information equation,

$$I(U,V) = h(U) + h(V) - h(U,V) \ . \tag{4.33}$$

The dependence of $h(U,V)$ on T is implicit in the above equation.

We can split the image into two parts: V_u and V_m. V_m is the part of the image in which the model lies. V_u is the unmodeled part of the image. We can define them as

$$V_m \equiv \{v(y) \text{ such that } u(T^{-1}(y)) \text{ is defined}\} \ , \tag{4.34}$$

and,

$$V_{\bar{m}} \equiv V - V_m \ . \tag{4.35}$$

We will assume that the two parts of the image are independent. This allows us to split the entropies that involve V into two distinct parts,

$$\begin{aligned} I(U,V) &= h(U) + h(V) - h(U,V) && (4.36)\\ &= h(U) + h(V_m, V_{\bar{m}}) - h(U, V_m, V_{\bar{m}}) && (4.37)\\ &= h(U) + h(V_m) + h(V_{\bar{m}}) - h(U, V_m) - h(V_{\bar{m}}) && (4.38)\\ &= h(U) + h(V_m) - h(U, V_m) \ . && (4.39) \end{aligned}$$

In (4.38) we have assumed that $V_{\bar{m}}$ is independent of V_m and U (i.e. that the background is independent both of the image of the object and the object). This allows us to replace split out this term from the joint entropy. Equation 4.39 directs us to maximize the entropy of the modeled part of the image, $h(V_m)$, as well as minimizing the joint entropy of model and the image.

EMMA can estimate the entropy of a signal. How can we estimate the entropy of an image? The short answer is that we can't. The entropy of an image is a function of the distribution of a vector random variable with between ten thousand and a million dimensions. We can however model the entropy of an image as the the sum of the pixel entropies. This is guaranteed to be an overestimate of the image entropy.

One of the problems with mutual information alignment is that it does not give us an idea if the object is present in the image or not. We can find the best pose of the object, but we do not know if it is really present. The principle of minimum description length should allow us to derive a decision threshold for the presence of an object. The object is present if the image is easier to encode with it than without it. Unfortunately this decision is highly dependent on the estimate of the likelihood, or code length, of the explained part of the image. The naive overestimate of image entropy—that simply sums the entropies of the pixels—is not tight enough to determine the decision threshold correctly. An important area of open research is deriving a more reasonable estimate of the code length of an image.

In our previous derivations we sampled points from the model and projected them into the image. Since we are now explicitly modeling the entropy of the image, we will sample from the image and project back into the model. The joint entropy is then,

$$\begin{aligned} h(v(y), u(T^{-1}\, y)) &= E_Y[log\, p(v(y), u(T^{-1}\, y))] && (4.40)\\ &= \int p(v(y), u(T^{-1}\, y)) log\, p(v(y), u(T^{-1}\, y)) dy && (4.41)\\ &= \int p(v(T\, x), u(x)) log\, p(v(T\, x), u(x)) A(T, x) dx && (4.42)\\ &= E_X[A(T, x) log\, p(v(T\, x), u(x))]\ . && (4.43) \end{aligned}$$

Equation (4.42) is a change of variables from y back into x. The correcting term $A(T, x)$ plays the role of the Jacobian of the transformation, measuring the ratio of the area in the model as it projects into the image. For affine transformations the projected area is related to the model area by the determinant of the transformation. For projective transformations there will also be a term that arises from foreshortening. The mutual information is then

$$I(U, V) \approx E_b \left[A(T, x_b) \begin{pmatrix} +log(P^*(u(x_b), a)) \\ +log(P^*(v(T\, x_b), a)) \\ -log(P^*(\{v(T\, x_b), u(x_b)\}, a)) \end{pmatrix} \right] . \qquad (4.44)$$

The mutual information between the whole model and image is the EMMA estimate of the mutual information between the model and image signals weighted by the projected area of the model. This new formulation of mutual information includes a term which encourages the model to explain as much of the image as possible. The derivative

$$\frac{d}{dT}I(U,V) \approx E_b \left[\begin{array}{c} A(T,x_b)\frac{d}{dT}\left(\begin{array}{l} +log(P^*(u(x_b),a)) \\ +log(P^*(v(T\ x_b),a)) \\ -log(P^*(\{v(T\ x_b),u(x_b)\},a)) \end{array} \right) \\ +\left(\begin{array}{l} +log(P^*(u(x_b),a)) \\ +log(P^*(v(T\ x_b),a)) \\ -log(P^*(\{v(T\ x_b),u(x_b)\},a)) \end{array} \right)\frac{d}{dT}A(T,x_b) \end{array} \right] , \quad (4.45)$$

encourages the model to grow as large as possible.

4.5 Summary

In this chapter several motivations for the selection of mutual information as a measure of alignment have been presented. The primary existing measure of alignment, correlation, has been rederived as a maximum likelihood technique which has a number of important weaknesses. The concept of maximum likelihood matching is then extended to define more general measure of alignment. This measure, called weighted neighborhood likelihood, can be interpreted as estimating the conditional entropy of an image given a model. Weighted neighbor likelihood is still somewhat limited however. From a set of more general assumption a related technique known as EMMA alignment has been derived. This technique explicitly estimates the mutual information between an image and model. A number of synthetic experiments demonstrate that mutual information is a very flexible measure of alignment.

In the final section of this chapter the concept of minimum description length is used as yet another motivation for mutual information as an alignment measure. This alternative framework encourages the model to explain as much of the image as possible.

Chapter 5

Alignment Experiments

This chapter contains a number of experiments designed to demonstrate that alignment by maximization of mutual information is a practical technique. The previous chapter contained a very general definition of alignment. Though a procedure for adjusting the pose parameters during alignment was derived, many of the details regarding representations and implementations were left out. Each experiment will include both these needed details and a general discussion of the experimental framework.

By the end of this chapter we will have developed some familiarity with the application of EMMA alignment. The chapter will conclude with a section describing explicit limitations of this approach. In addition to describing problems for which EMMA alignment is poorly suited, it will be emphasized that EMMA alignment is not a complete object recognition system.

5.1 The Basic Framework

The basic framework for all of the experiments described in this chapter is the same. The framework is easily broken down into discrete steps that can be applied to a wide variety of alignment problems. Table 5.1 describes the general procedure that is used to set up each experiment. With each experiment we will include a

1.	Choose a model and image representation (i. e. define $u()$ and $v()$). Define an interpolation scheme for sampling $v()$ at non-integral coordinates.
2.	Choose a scheme for sampling the model (i.e. define x).
3.	Determine the space of possible aligning transformations and its concrete representation (i.e. define T). The definition of the random variables $u(x)$ and $v(T(x))$ is now complete.
4.	Derive an expression for $dv(y)/dy$.
5.	Pick a metric for computing distances between pairs of samples of $u(x)$, $v(T(x))$ and $\{u(x), v(T(x))\}$.
6.	Pick the variance for the component densities: ψ.
7.	Choose a value for p_{min}.
8.	Determine the number of samples used to estimate the distribution, and the number used to estimate the entropy.
9.	Pick a parameter update rate, λ. Since the alignment procedure is stochastic, a large update rate causes a lot of bouncing around in transform space. A small one never gets to the minimum. In general the update rate should decrease with time.

Table 5.1: The process for setting up an alignment.

similar table with a specific realization for each of these requirements.

5.2 Alignment of 3D Objects to Video

In our first experiment we will return to the example described in the introduction: alignment of a three-dimensional object to a video image. In order to apply the EMMA alignment framework we must first determine that there is mutual information between the model and the image.

The appearance of a small patch of the object's surface is a function of the surface properties, the patch orientation, the position of the lights and the position of the observer. The relationship between the image and the model can be summarized in an imaging equation,

$$v(T\,x) = F(u(x), s(x), q) \ , \tag{5.1}$$

where $u(x)$ is the normal vector of a surface patch, $s(x)$ are the surface properties of the patch, q is a description of the lighting, and F is the imaging function. We can conclude that the model contains information about the image because the image is a function of the model. EMMA allows us to align a model and image even when the the imaging function and the lighting is unknown.

In all of our alignment experiments we will assume that the entire object has the same surface properties. We can then treat surface property as yet another exogenous variable.

Following Table 5.6:

1. Models are a collection of points that lie on the surface of the object. We chose this representation because it is capable of representing any shape including smoothly curved or irregular forms. It is equally capable of representing objects with flat faces such as polyhedra. The models have been constructed so the distribution of surface points is as close to uniform as possible. Associated with each surface point is the local surface normal, a unit vector perpendicular to the surface. The models used have between 7000 and 65,000 points. Video images are represented as simple two dimensional arrays.

2. The random variable x that is used to sample the model and image is defined from the model. A trial of x is a randomly selected model point. The value of the trial is the 3D location of that model point. We sample the points of the model uniformly.

3. The transformation space is the space of rigid translations and rotations followed by perspective projection. The overall transformation is a concatenation of rotations and translations each acting on the pre-defined "center" of the object.

   Because of self-occlusion, not every point on the model is visible. Visibility is determined by a Z-buffer rendering of the model. Z-buffer rendering takes each point in the model and projects it into the image. When multiple points fall onto the same pixel, only the point that is nearest is considered visible. As pose changes, some points become visible and others become invisible. In theory Z-buffering needs be repeated every time the pose of an object changes. Unfortunately, Z-buffering takes time proportional to the size of the model. This cost is far larger than the cost of computing an estimate for the derivative of entropy. Since pose does not change much between iterations of gradient descent, it has proven sufficient to Z-buffer every 300 iterations.

4. The derivative $dv(y)/dy$ is the spatial gradient of the image.

5. The metric used for comparing points sampled from the image and model is Euclidean distance. The metric for the joint space of the image and the model must measure distances between joint events $w = \{v(T\ x), u(x)\}$. We will represent only two dimensions of the normal vector: the x and y components. Since the normal is always a unit vector, the z component is redundant. The joint events are therefore three dimensional vectors, two components from the model and one from the image. We will use Euclidean distance to measure the distance between joint events.

6. Since we will be using diagonal covariance matrices for the smoothing functions, four variances are required. Three for the joint entropy and one for the image entropy. Based on maximum likelihood estimates from aligned objects, we have settled on a single set of smoothing parameters that we will use for all of our 3D alignment experiments. For the joint entropy, the variance of x and y components of the normal are both 0.3 and the variance for image intensity is 0.2. For the image entropy, the variance for image intensity is 0.15. Having a single set of parameters is for every experiment is possible in part because we have pre-normalized the image so that its variance is 1.0.

7. We will use a value of 0.01 for p_{min}. The alignment process shows very little sensitivity to p_{min}. We have repeated a number of experiments with a p_{min} value of 0.1 and 1.0. Our results are not significantly different. Values that are more than a factor of 10 smaller than 0.01 cause the derivative of estimated entropy to be too noisy (see Section 3.3). This noise can prevent convergence to the correct pose.

8. Rather than draw two different samples, we will use the cross-validation approximation (see Section 2.4.3). In all of our experiments we use a sample size of 25.

9. Finally, we must choose a parameter update rate, λ. Actually, since the units of rotation and translation are very different two update rates are necessary. Internally we represent rotations in radians and translations in millimeters. For an object with a 100 millimeter radius a rotation of 0.01 radians about the center of mass can translate a model point up to 1 millimeter. A translation of 0.01 can at most translate a model point 0.01 millimeters. The derivative of mutual information with respect to a model point's position is a combination of a rotation and translation. A small step in the direction of the derivative will move the model point up to 100 times further by rotation than translation. If there is only a single update rate a poor compromise must be made between the rapid changes that arise from the rotation and the slow changes that arise from translation. If the rotation update rate is reduced by a factor of 100 the model point will move approximately as far by rotation as it does by translation. Scale issues such as these do not arise when more complex gradient descent techniques are used, for example conjugate gradient descent or Levenberg-Marquardt. Unfortunately, neither of these techniques can use stochastic estimates of the gradient. Since our models have a radius that is on the order of 100 millimeters, we have chosen rotation update rates are 100 times smaller than translation rates. Most of our 3D alignment experiments proceed in two stages. In the first stage the rotation update rate is 0.0005 and the translation update rate is 0.05. After a number of iterations the update rates are then reduced to 0.0001 and 0.01 respectively. We have chosen a simple automatic descent procedure in an effort to simplify subsequent analysis of convergence.

The realization of the basic framework is summarized in Table 5.2.

5.2.1 Alignment of Skull Model

In our first experiment we will align a 3D skull model to a number of different images. The skull model was produced automatically from a Computed Tomography (CT) scan of a plastic skull[1]. The same plastic skull was then photographed in a

[1]Thanks to Gil Ettinger and Ron Kikinis for providing the skull model. Their work on medical registration using this model is described in (Grimson et al., 1994).

1.	Define the model and image $u()$ and $v()$: $u()$ contains points distributed on the surface of the object. Each point has an associated normal. $v()$ is the image of intensities.
2.	Sampling x: The sampling is determined by the distribution of surface points which is close to uniform.
3.	Transformation space T: The space of rigid spatial transformations followed by perspective projection using an estimated for the camera parameters.
4.	Definition of $dv(y)/dy$: This is the intensity gradient.
5.	Distance metric: Euclidean distance.
6.	Variance, ψ: Assuming diagonal covariance matrices, four different variance are necessary, three for the joint entropy estimate and one for the image entropy estimate. The variances were 0.3, 0.3, 0.2, and 0.15 for the x component of the normal, y component of the normal, image intensity and image intensity respectively.
7.	Minimum probability, p_{min}: 0.01.
8.	Number of samples: One sample of 25 using cross-validation.
9.	Update rate, λ: Rotation rate: 0.0005 for 3200 steps and then 0.0001 for 3200 steps. Translation rate: 0.05 and then 0.01.

Table 5.2: Summary of 3D to video alignment.

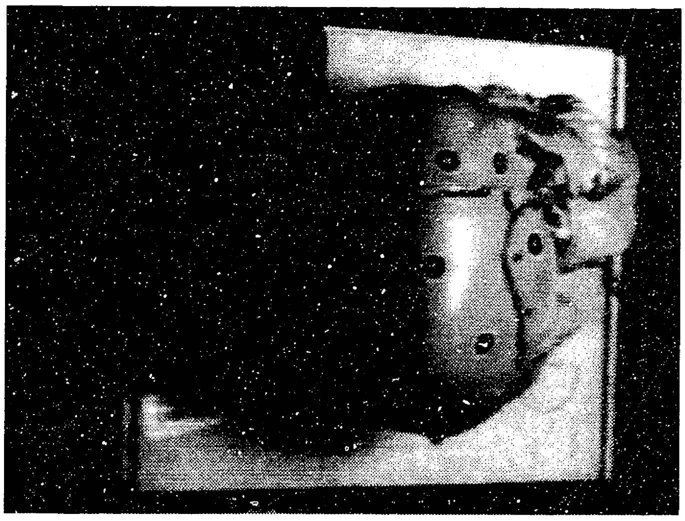


Figure 5.1: A typical image of the skull object.

number of different poses under natural lighting[2]. The skull model contains 65000 points. The video images are 240 by 320 pixels. Figure 5.1 is an example video image of the skull.

Figure 5.2 contains a representation of the shape of the skull model. It is an image displaying the distance from the camera to the visible points on the skull model. White is further and black nearer. This image is computed by projecting each model point into the image plane. The pixel to which the model point projects records the distance of the model point from the camera. There may, however, be a number of model points that project to the same image pixel. In this case, the depth of the model point which is nearest the camera is used. Since the model is constructed from a collection of points, it is not dense. As a result there are some pixels to which no model point projects. A few of these pixels, which remain white, appear throughout the model.

Alternatively we can use a Lambertian reflectance model to render a graphical picture of the skull. A Lambertian model relates the model normal and image intensity:

$$v(Tx) = \sum_i \alpha_i \vec{l_i} \cdot u(x) \ , \tag{5.2}$$

where the model value $u(x)$ is the normal vector of a surface patch, l_i is a vector

[2]Thanks to J. P. Mellor for providing the skull images. His work on registration is described in (Mellor, 1995).

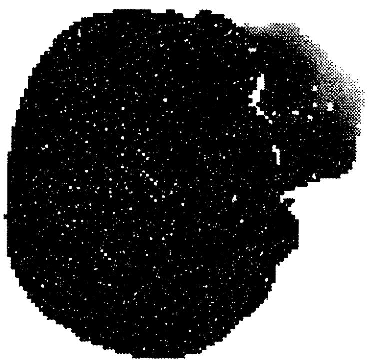


Figure 5.2: A depth map of the skull model. See text for description.

pointing toward light source i, and α_i is proportional to the intensity of that light source ((Horn, 1986) contains an excellent review of imaging and its relationship to vision). Figure 5.3 shows a rendered version of the model in the same pose as Figure 5.1. To the human eye this sort of image is more readily interpretable than a depth map. We can bring to bear our substantial visual competence when the shape of an object is rendered as an image. From Figure 5.3 it is almost immediately clear that the pose of the object model is close to correct. There is however no simple relationship between the intensities of the video image and the rendered image.

The goals of this first experiment is to answer three questions. (1) Can EMMA align a complex 3D object model to a number of different images taken under uncontrolled lighting? (2) How long does EMMA alignment take to run? (3) What is the range of poses from which a "correct" alignment can be obtained. Regarding this third point, we do not have true information about either the pose of the object nor the camera parameters of the video camera. The "correct" pose has been determined by inspection of the alignment results. We can however ask a related question about reliability. How far can the the object be perturbed away from the "correct" pose and have EMMA alignment reliably re-align it?

To answer our first question, we must establish that in the six dimensional space of rigid transformations there is a maximum of mutual information at a plausible alignment pose. For each image the object model was initially adjusted so that it's

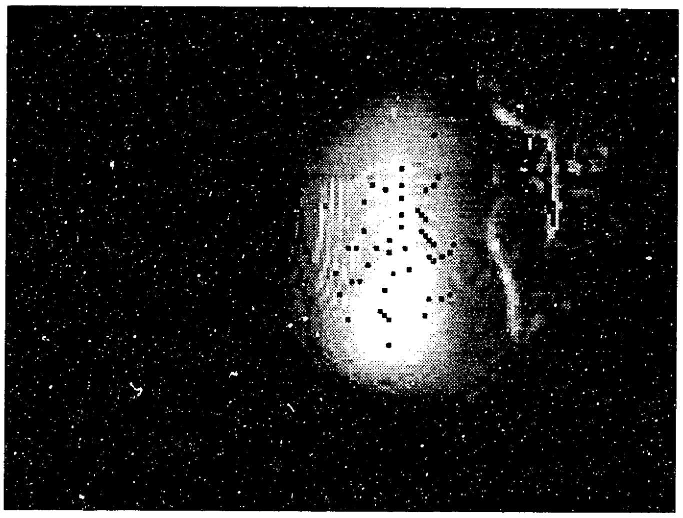


Figure 5.3: A rendered image of the skull model.

pose was close to correct. This was done by eye. EMMA alignment was then used to pull the object into a "correct" pose. We can determine whether a maxima of mutual information coincides with the correct pose if: (1) EMMA alignment pulls the initial pose toward the correct pose, and (2) EMMA alignment does not force the initial pose away from correct pose.

One scheme for assessing the quality of an alignment is to display the model pose in the video image. To do this, we take a random collection of model points and project them into the image. At the point of projection we set the video image pixel to white. When the image and model are aligned no white points will fall on the background and the occluding contour of the object will be closely approximated by the model points. Figure 5.4 shows an initial incorrect pose in this way. Figure 5.5 shows the final pose obtained after running EMMA alignment. For this final pose there is close agreement between the model and the image.

Figures 5.6, 5.7, and 5.8 show the final alignment obtained for three other images. Notice that in each of these images the boundaries of the skull are in close agreement with the outline of the model points.

We would like to emphasize that in none of these experiments have we pre-segmented the image. The initial poses often project the model into regions of the image that contain a significant amount of clutter. EMMA reliably settles on a pose where few if any of the model points project onto the background.

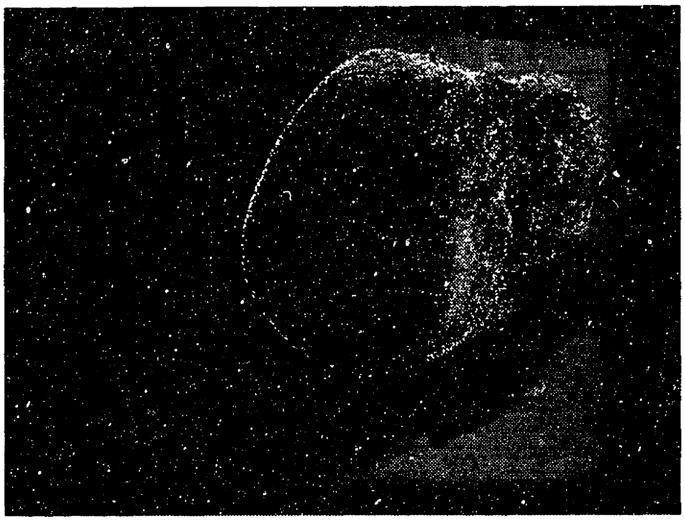


Figure 5.4: Initial pose of the skull model before alignment.

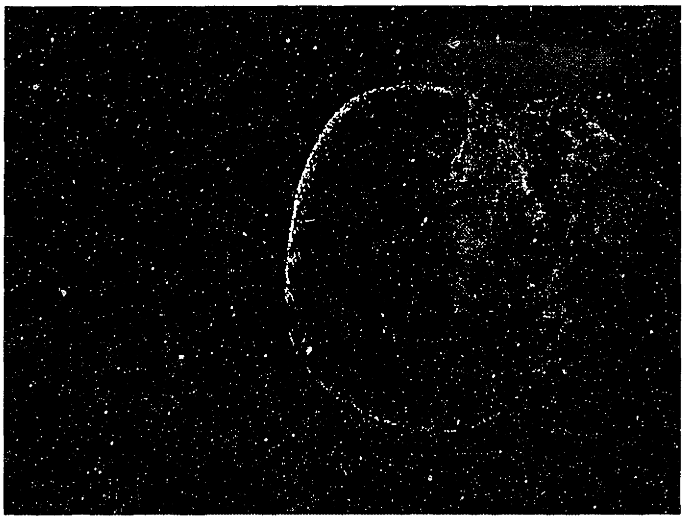


Figure 5.5: Final pose of the skull model after alignment.

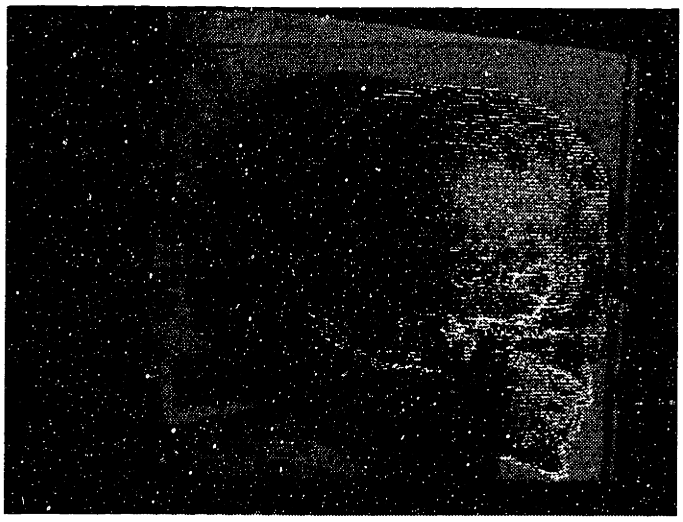


Figure 5.6: Final pose of the skull model after alignment.

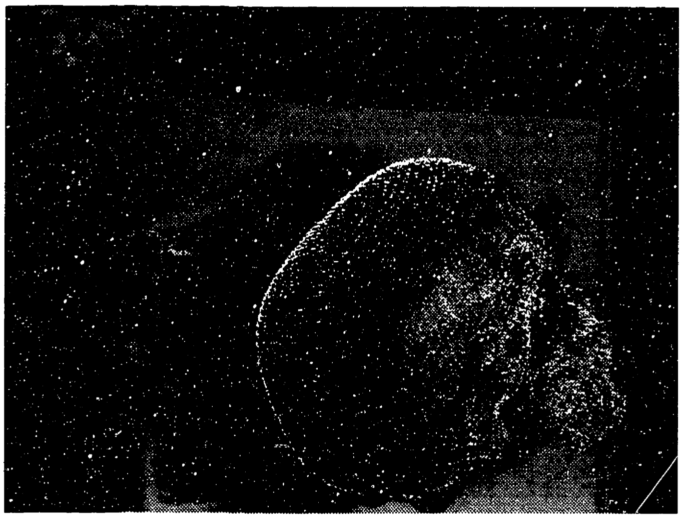


Figure 5.7: Final pose of the skull model after alignment.

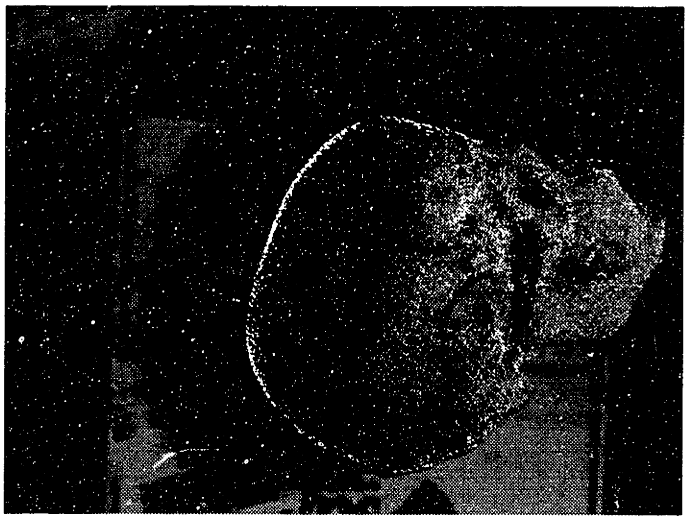


Figure 5.8: Final pose of the skull model after alignment.

In answer to the second question, EMMA requires roughly 35 seconds on a Sun SparcStation5 for each of the alignments shown above. Run times are identical because we have chosen to use a fixed number of update iterations for each alignment experiment. In some cases an accurate alignment was obtained well before the full number of iterations had been completed. In others it appeared that the final alignment could have been improved if the number of iterations were increased.

There are few if any principled results on the convergence of stochastic approximation. Convergence detection is a subtle issue. For example, EMMA does not make a direct estimate of the mutual information between model and image. During alignment only a stochastic estimate of the gradient is available. It may be possible to construct an ad hoc procedure that would be able detect convergence. Alignment could then be continued until the pose estimate had converged.

From an analysis of the program's memory access and computation patterns, we conclude that an implementation on a digital signal processor would be as much as 100 times faster than our current implementation. One major issue is cache performance. Because EMMA randomly accesses each of the points in the image and model, much time is wasted flushing and refilling the cache. The cache on a general purpose processor is often fairly limited. Most digital signal processors include a large quantity of fast SRAM, eliminating the need for a cache. For random memory accesses a digital signal processor should be approximately 5 times faster than a conventional computer. The inner loop of the EMMA derivative estimation

ΔT	$\Delta\theta$	INITIAL				FINAL				
X Y Z		σ_X	σ_Y	σ_Z	$\overline{\mid\Delta\theta\mid}$	σ_X	σ_Y	σ_Z	$\overline{\mid\Delta\theta\mid}$	
±mm	°	mm			°	mm			°	%
10	10	5.94	5.56	6.11	5.11	.61	.53	5.49	3.22	100
30	10	16.53	18.00	16.82	5.88	1.80	.81	14.56	2.77	96
20	20	10.12	12.04	10.77	11.56	1.11	.41	9.18	3.31	96
[10, 20]	[20, 40]	14.83	15.46	14.466	28.70	1.87	2.22	14.19	3.05	78

Table 5.3: Skull Results Table. The final column contains the percentage of poses that successfully converged to a pose near the correct pose.

procedure is dominated by simple floating point operations. Modern digital signal processors can execute these instructions 10 to 20 times faster than conventional computers. Together these advantages should lead to an overall improvement in speed of between 50 and 100.

A number of randomized experiments were performed to determine the reliability, accuracy and repeatability of alignment. This data is reported in Table 5.2.1. An initial alignment was performed to establish a base pose. This pose, shown in Figure 5.5, is used as a point of reference. A set of randomized experiments were performed where the base pose is first perturbed, and then EMMA is used to re-align the image and model. The perturbation is computed as follows: a random uniformly distributed offset is added to each translational axis (labeled ΔT) and then the model is rotated about a randomly selected axis by a random uniformly selected angle ($\Delta\theta$). There were four experiments including 50 random initial poses. The distribution of the final and initial poses can be compared by comparing the variance of the location of the centroid, computed separately in X, Y and Z. Furthermore, the average angular rotation from the true pose is computed (labeled $\overline{\mid\Delta\theta\mid}$). Finally, the number of poses that failed to converge near the correct solution is reported. The final statistics are only evaluated over the poses that converged near the correct solution.

These experiments demonstrate that the alignment procedure is reliable when the initial pose is close to the "correct" pose. Outside of this range gradient descent, by itself, is not capable of converging to the correct solution. Our empirical experience has led us to conclude that alignment is most sensitive to rotation in depth. Translations as large as half the diameter of the skull can be accommodated,

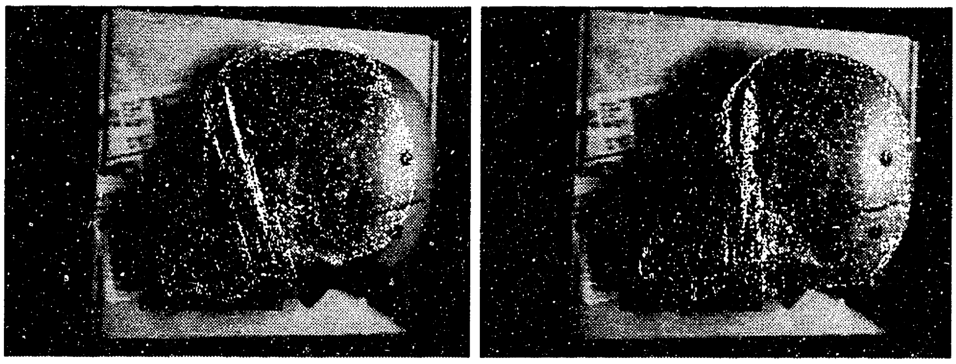


Figure 5.9: An image including an artificial occlusion. White spots denote the pose of the model. On the left is the initial pose, on the right is the final pose.

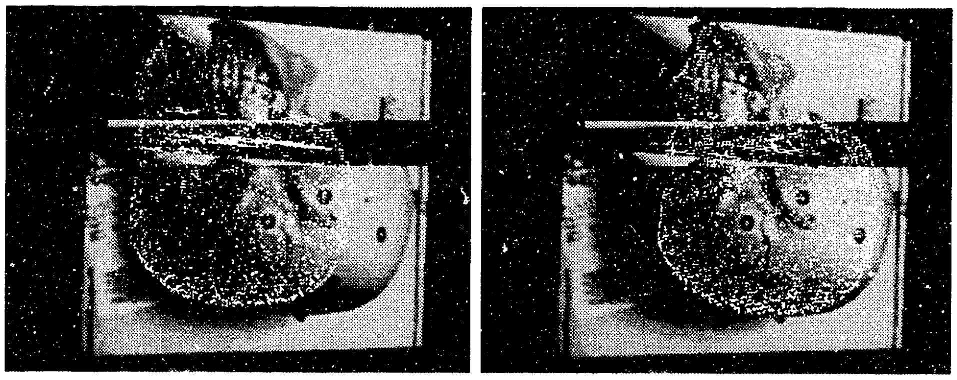


Figure 5.10: An image including an artificial occlusion. White spots denote the pose of the model. On the left is the initial pose, on the right is the final pose.

as can rotations in the plane of up to 45 degrees. Rotation in depth, however, can act to hide portions of model through self-occlusion. Only points that are visible play a role in the calculation of the derivative. As a result, when the chin is hidden the derivative gives you no information about how move the chin out from behind the rest of the skull.

Finally, we have done a number of experiments to demonstrate that EMMA alignment can deal with occlusion. Figure 5.9 shows an initial and final alignment for an image that includes an artificial occlusion that covers the entire chin area. The final alignment is very close to the correct one despite the occlusion. Figure 5.10 shows an initial and final pose for a more complex occlusion. In this image we have replaced a rectangular window with another randomly chosen window of the image. The source of the rectangle is near the bottom of the image. In a number of experiments, we have found that alignment to occluded images can require more time for convergence.

5.2.2 Alignment of Head Model

We have repeated many of the skull experiments with a three dimensional model of a human head. This model was obtained from a Cyberware scan of the subject that was taken approximately two years before the video images[3]. A Cyberware scan is a complete three dimensional representation of the shape of the subject's head in polar coordinates. The surface normals were computed from the surface by smoothing and differencing neighboring surface points.

The experiments in this section are designed to answer two questions: (1) Will the same techniques and parameters work with two different types of models and images? (2) Is it possible to use the pose refinement procedure to track a moving object in a video sequence? Figure 5.11 shows an image of the head and a rendering of the model.

How are the face experiments different from the skull experiments? Firstly, the face model is much smoother than the skull model. There really aren't any creases or points of high curvature. As a result it is much less likely that an edge-based system could construct a representation either of the image or the model that would be stable under changes in illumination. Secondly, the albedo of the actual object is not exactly constant. The face contains eyebrows, lips and other regions where the albedo is not the same. As a result this is a test of EMMA's ability to handle objects where the assumption of constant albedo is violated. Thirdly, not all of the occluding contours of the object are present in the model. The model is truncated both at the chin and the forehead. As a result experiments with this model demonstrate that EMMA can work even when the occluding contours of the image and model are not in agreement.

Since the model is very smooth and some occluding contours are missing simply projecting the model points into the image is not sufficient to determine the quality of an alignment. For our experiments with the head model we will display the

[3]Thanks to Ron Kikinis for providing the Cyberware scan and for allowing me to take the images of him.

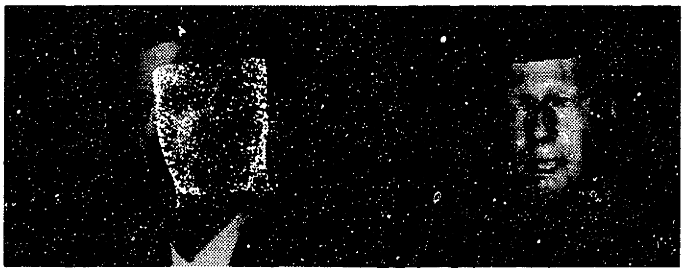


Figure 5.11: An initial incorrect pose. The model has been rotated 30 degrees about the vertical and translated 40 millimeters to the right. On the left is an image of the head along with a collection of points projected from the model. On the right is rendering of the model is the same pose.

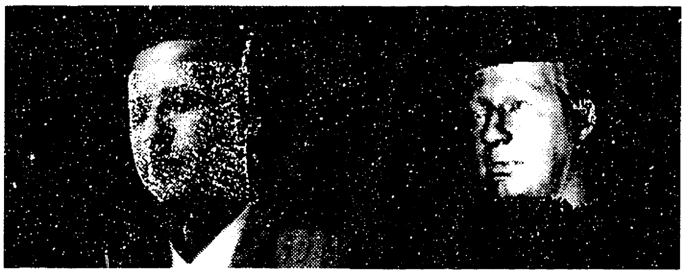


Figure 5.12: The final aligned poses. On the left is an image of the head along with a collection of points projected from the model. On the right is rendering of the model is the ~ame pose.

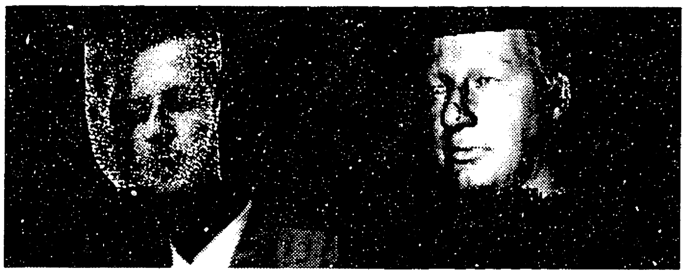


Figure 5.13: An initial incorrect pose. The model has been moved toward the camera 150 millimeters toward the camera.

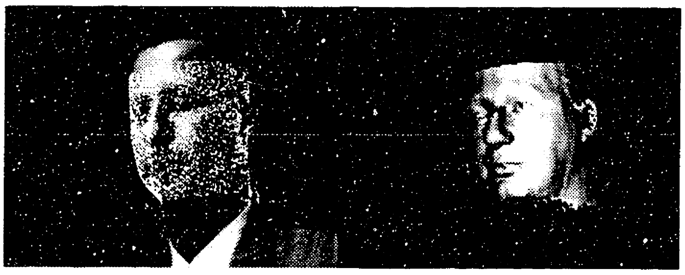


Figure 5.14: The final aligned poses.

original image, augmented with model points, along side a rendered image of the model. Figures 5.11 and 5.12 show the model before and after alignment. In this experiment the model has been rotated 30 degrees around the vertical and translated 40 millimeters to the right. Figures 5.13 and 5.14 show another experiment where EMMA alignment corrects for a 150 millimeter translation in depth.

We have also tested EMMA alignment on a video sequence digitized from a video tape. The sequence was taken at the same time as the other images, though the camera and the lens were different. Ten frames were acquired from a video tape at 3 frames per second. The quality of the resulting images is very low. The images were degraded both by their storage on video tape and by the frame grabber that was used. It was somewhat surprising that these images worked nearly as well as the higher quality still frames.

Motion in the video sequence was tracked by sequentially aligning the model to each of the frames. The initial pose for each frame was obtained by using the aligned pose from the previous frame. The initial model pose was hand selected so that EMMA alignment could acquire a good initial alignment. The sequence and pose estimates are displayed in Figure 5.4.

5.2.3 Alignment of Curved Surfaces

The third experiment is designed to explore the nature of the information that EMMA alignment uses to detect the correct pose. The previous two experiments, because they are based on real data, can be difficult to analyze. We would like to determine which component of the information in the image and model is critical to alignment. For example, it could be the case that EMMA alignment relies implicitly on intensity edges to match model to image. Or, it could be the case that the occluding contours of the object are of critical importance.

For this experiment we created a very simple, almost pathological, synthetic example. The object is a set of three Gaussian shaped bumps in a flat patch of surface. This object in has no sharp edges and does not have a well defined occluding contour. Figure 5.15 shows ten different images of this object. Each image uses the same Lambertian reflectance model but has different illumination.

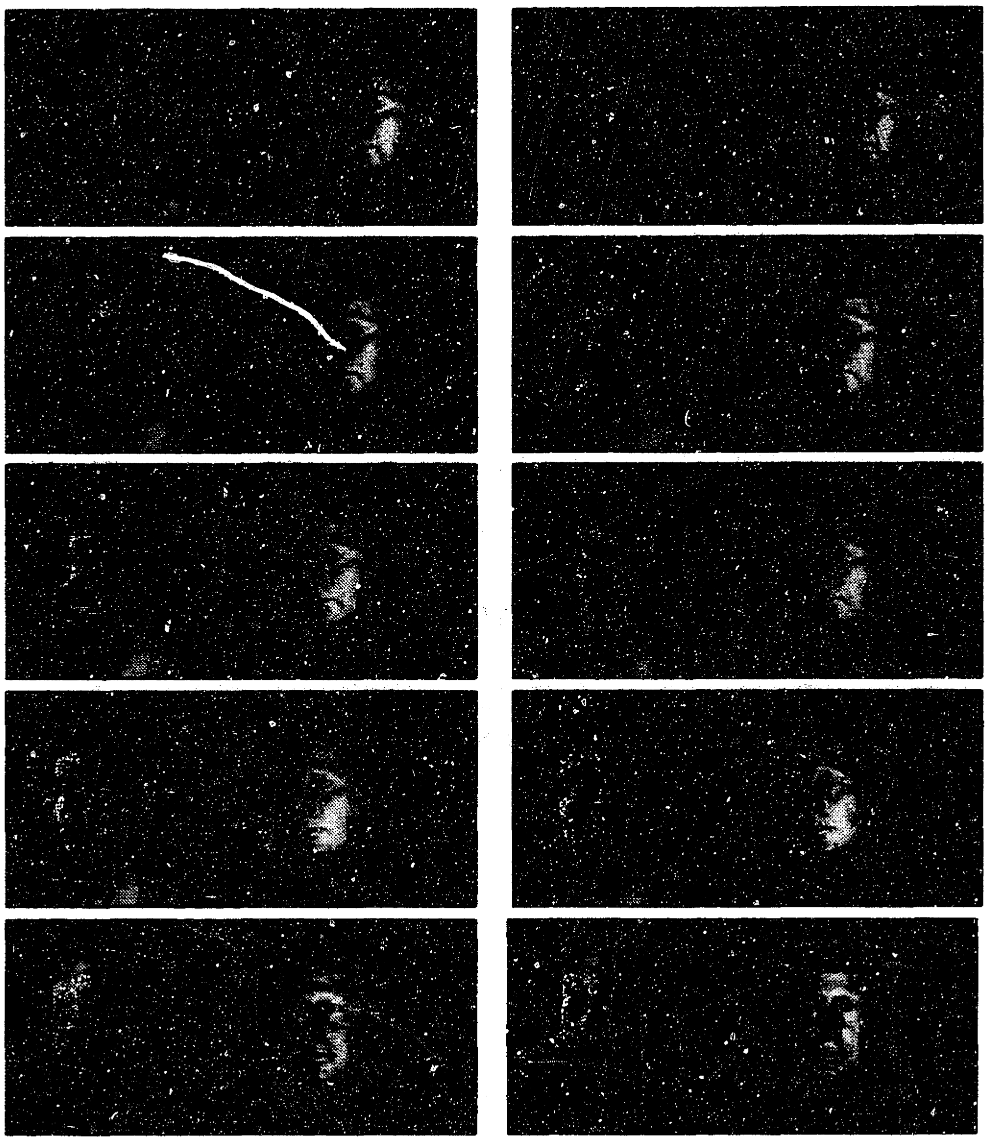


Table 5.4: Ten frames from a video sequence of RON's head.

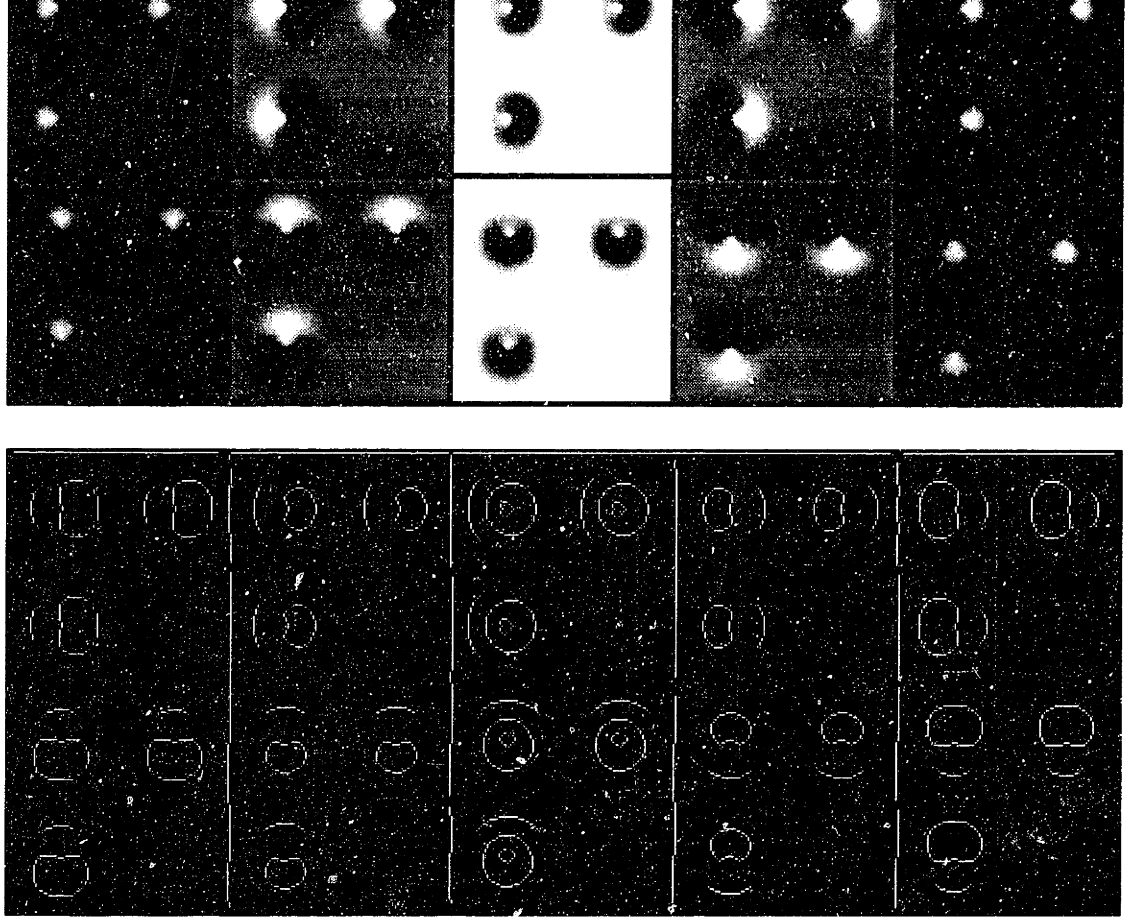


Figure 5.15: Bumps with Different Lighting, and their Edges

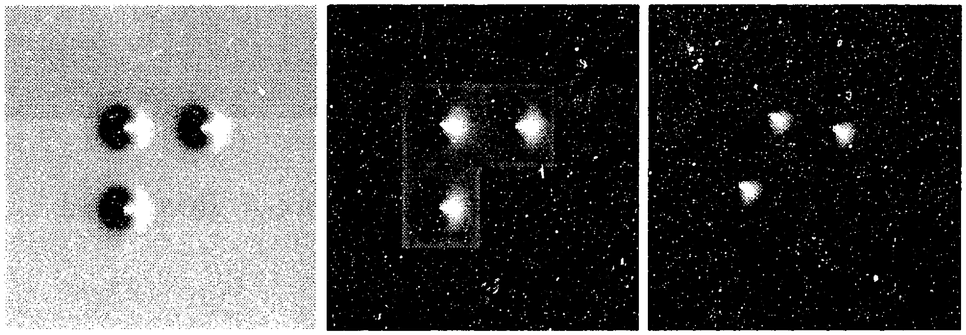


Figure 5.16: Target Image, Final Model Pose, and Initial Model Pose

ΔT			$\Delta\theta$	INITIAL				FINAL				SUCCESS
X	Y	Z		σ_X	σ_Y	σ_Z	$\overline{\lvert\Delta\theta\rvert}$	σ_X	σ_Y	σ_Z	$\overline{\lvert\Delta\theta\rvert}$	
±mm			°	mm			°	mm			°	%
10	10	25	30	5.21	5.82	19.97	15.41	.56	.45	9.73	5.68	100
15	15	25	20	8.75	8.95	14.61	9.15	.82	.39	8.95	3.23	100
20	20	25	20	11.72	11.07	13.85	9.22	.65	.38	8.35	3.12	80

Table 5.5: Curved Surface Alignment Data

Across the top row the light source moves gradually from left to right. In the second row the light source moves from top to bottom. Even for a simple Lambertian surface, image variation can be significant. Below this we show the output of a Canny edge detector run on the ten different images (Canny, 1986). The variation between the different edges extracted is quite striking.

Figure 5.16 shows the target image on the left. Here the bumps are inserted into an infinite plane. Also shown is a rendered version of a typical final and initial pose of the model. The rendered images of the model are made using a particular surface and lighting scheme. Neither is part of the alignment process. The model is smaller than the target image, and points outside the model are rendered in black. Notice that the model's boundaries do not coincide with any discontinuities in the target image. Alignment proceeds using only shading information.

The bumps each have a sigma of 7mm (the bump is about 3 sigma or 21mm wide). The bump height is 20mm. Lying together in the same plane they take up an L shaped region that is 100mm by 100mm. The true pose is 1000mm away from the camera and perpendicular to the camera axis. The camera has a viewing angle of 18 degrees. This experiment proceeds exactly as the previous two three dimensional experiments.

As we did with the skull model we performed an analysis of the reliability of the maxima of mutual information. These experiments summarized in Table 5.2.3.

5.3 Medical Registration Experiments

EMMA alignment of three dimensional objects relies on the fact that the image is a function of the model and the lighting. It is not necessary that we know the exact nature of this function. In medical imaging we are faced with a different though related task. We are given two different observations of the same object. For example we may be given a *Computed Tomography* (CT) scan and *Magnetic Resonance Image* of the same patient. Two scans are often obtained because neither gives perfect information about the patient. CT is good at finding bone. MRI is good for distinguishing soft tissue. These measurements are frequently taken at different times with different machines. A clinician that would like to have information about bone and soft tissue must integrate these two scans into a single self-consistent picture. Once this is done the spatial relationships between structures in the two different scans become apparent. For example the distance between a tumor and a bone can be measured.

In Chapter 4 we argued that EMMA alignment should be able to align two signals whenever there is mutual information between them. This experiment is designed to demonstrate that this is in fact feasible. We are given two different MR images of the same head (see Figure 5.17). They comprise the proton density and T2-weighted images of a double-echo MR scan. It is clear that the two images share a great deal of information, while they are not identical. They are taken simultaneously in the same machine. The correct alignment should be close to the identity transformation. Because we know ground truth, we can evaluate the accuracy of the EMMA alignment procedure.

A typical initial alignment appears in Figure 5.18. Notice that this image is a scaled, sheared, rotated and translated version of the original. The final alignment is displayed as a checkerboard. Here every other 20x20 pixel block is taken either from the model image or aligned target image. Notice that the boundary of the brain in the two images is in close agreement.

We represent transformation space as a 6 element affine matrix that is used to project two dimensional points from the image into the model. This scheme can represent any combination of scaling, shearing, rotation and translation. The remaining algorithmic details are summarized in Table 5.3.

A set of 50 randomized trials were performed. The initial transformations had a randomly select translation of up to 32 pixels (this is about one third of the width of the head), an initial rotation of up to 28 degrees, . and an initial scaling of up to 20%. The correct alignment was obtained in 100% of the experiments. After alignment the affine transformations had an average translation error of 0.1 pixels. The remaining affine parameters represent a mixing of rotation, scale and shearing. They are somewhat more difficult to interpret. The correct matrix is the identity matrix. On average the coefficients of the final matrix were in error by 0.02. It can be concluded that EMMA alignment finds the correct alignment of these two medical images with high precision and reliability.

These two MRI images are fairly similar. Good alignment could probably have been obtained with a normalized correlation metric. Normalized correlation assumes, at least locally, that one signal is a scaled and offset version of the other. Our technique makes no such assumption. In fact, it will work across a wide variety of non-linear transformations. More difficult alignment problems are easily simulated. In Figure 5.19 we show the model image after a non-monotonic non-linear function has been applied. Recall that initially the image lies in the range [0,1]. We subtract 0.5, square the result, and renormalize to [0,1]. This operation is shown at the right of the figure. After applying this non-linear transformation the two images are anti-correlated. EMMA alignment performance, however, is not affected.

5.4 View Based Recognition Experiments

In the previous two vision experiments we used knowledge of the physics of imaging to show that the surface normal of an object should be predictive of the intensity observed in an image. In many experimental situations a three dimensional model is not available from which which we can compute the surface normal. In these situations it is frequently the case that the only information available about the object is series of images taken under different conditions.

One framework for solving problems like this is to collect a set of model images. We will call such a technique a "view based" approach. Given a novel image of some object, each model image is compared to it in turn. If some model image is

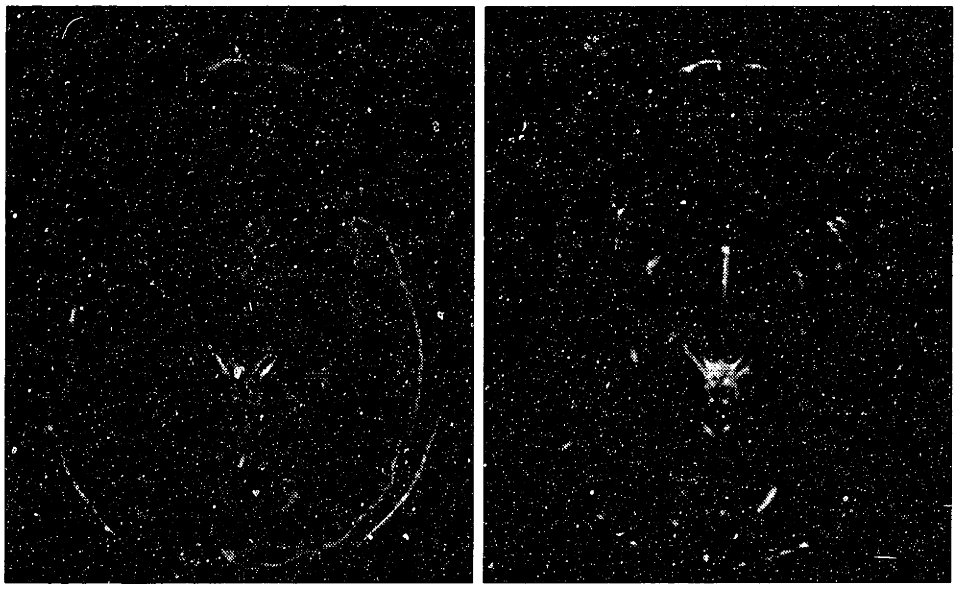


Figure 5.17: MR Images

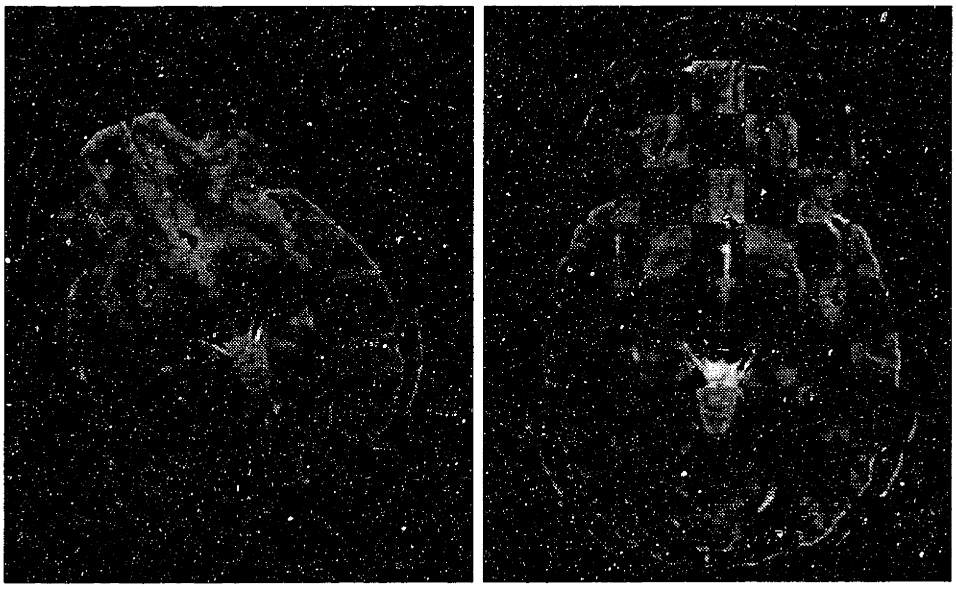


Figure 5.18: Initial Pose and Display of Result

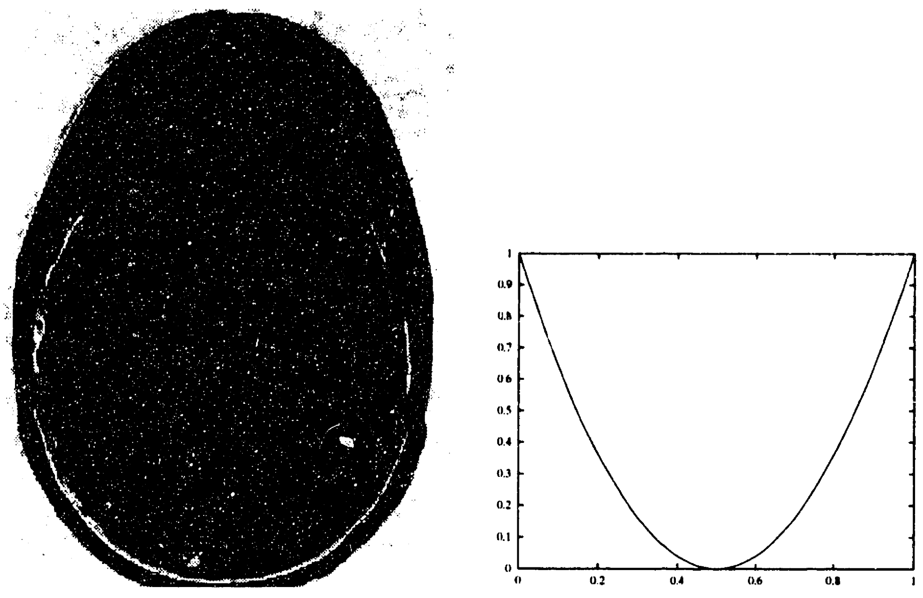


Figure 5.19: Transformed Model and the Transformation

1.	Define the model and image $u()$ and $v()$: $u()$ is the model image, $v()$ is the target image.
2.	Sampling x: Sample the pixels of the model image uniformly.
3.	Transformation space T: The space of affine transformations mapping pixels locations from the model into pixel locations in the image.
4.	Definition of $dv(y)/dy$: This is the intensity gradient.
5.	Distance metric: Euclidean distance.
6.	Variance, ψ: Assuming diagonal covariance matrices, three different variance are necessary: two for the joint entropy and one for the image entropy. The variances were 0.1 in all cases.
7.	Minimum probability, p_{min}: 0.01.
8.	Number of samples: One sample of 20 using cross-validation.
9.	Update rate, λ: 0.02 for 500 steps and then 0.005 for 500 steps.

Table 5.6: Summary of MRI alignment experiments.

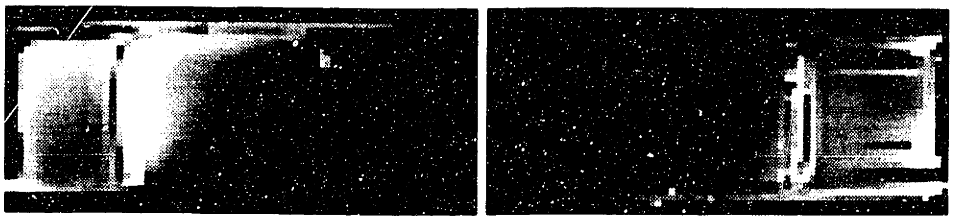


Figure 5.20: Car Model Images

"close enough" to the novel image, the model and image are considered aligned (or recognized). One can significantly reduce the number of model images required by adding an affine transformation to the comparison process. The novel image is compared to the model images under a set of affine transformations. The most commonly used comparison metric is correlation. As we saw in Section 4.1.1, correlation makes the assumption that the model and the image are identical (or possibly related by linear function).

Comparing images can be quite difficult. The set of images that can arise from a single object under varying illumination is very broad. Figure 5.20 shows two different images of the same object under different illumination conditions. These images are anti-correlated. Bright pixels in the left image correspond to dark pixels in the right image. Dark pixels in the left image correspond to bright pixels in the right image. No variant of correlation could match these images together.

Anti-correlation implies that there is some consistent relationship between images. It is not difficult to find two images of the same object for which there is no consistent mapping between the intensities in one image and the intensities in the other. Figure 5.21 shows a novel image which is aligned with the other model images. Figure 5.22 contains scatter plots of the pixels in the two model images versus the pixels in the novel image. There is no consistent relationship between in either of these graphs. No technique that depends on there being a relationship between model and image, like EMMA, could match these images together.

When multiple images of an object are available a technique called photometric stereo can be used to estimate the shape of the object (Horn, 1986). Photometric stereo assumes that the images are taken from the same location but under different illumination conditions. Typical implementations of photometric stereo require that the surface properties of the object be constant throughout. This insures that the reflectance properties of the object are constant. Furthermore, the direction

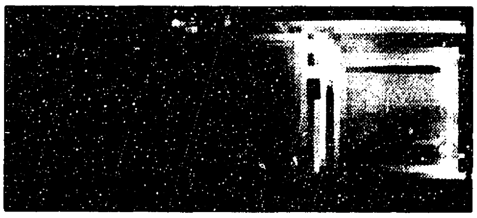


Figure 5.21: A novel image of the car model.

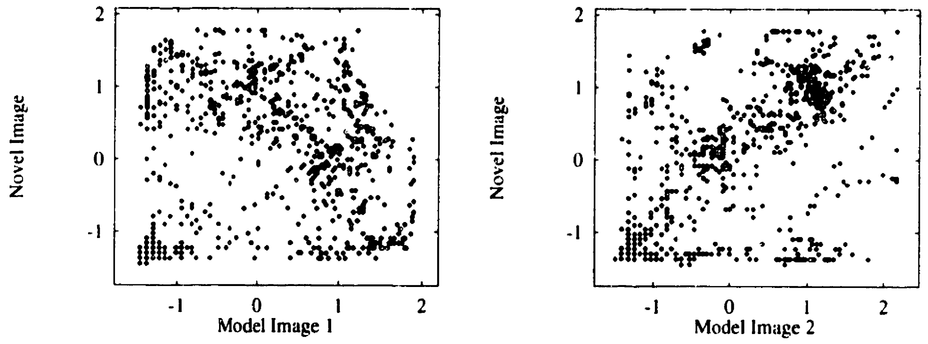


Figure 5.22: The relationship between pixels in the novel image and each of the model images.

and magnitude of the light sources must be accurately known. Knowledge of the light sources and surface material allows for the construction of a single reflectance map for each image.

Given an image and a reflectance map a set of constraints for the normals can be computed. From the reflectance a set of allowable normals for every pixel can be computed. The allowable normals usually lie along a closed curve on the unit circle. Using a second image and reflectance map another set of normal constraints can be computed. By intersecting these constraints, two images are sufficient to constrain the surface normal at each pixel to one of two values. From the normals the shape can be obtained through integration.

Using the shape extracted by photometric stereo we could now perform the three dimensional version of EMMA alignment. As we've seen before the shape of an object, plus some exogenous variables, are sufficient to predict any novel image. Prediction of the intensity in a novel image is the concatenation of the photometric stereo process followed by the imaging process:

$$I(T(x_i)) = F(G(u_1(x_i), r_1, u_2(x_i), r_2), q)$$

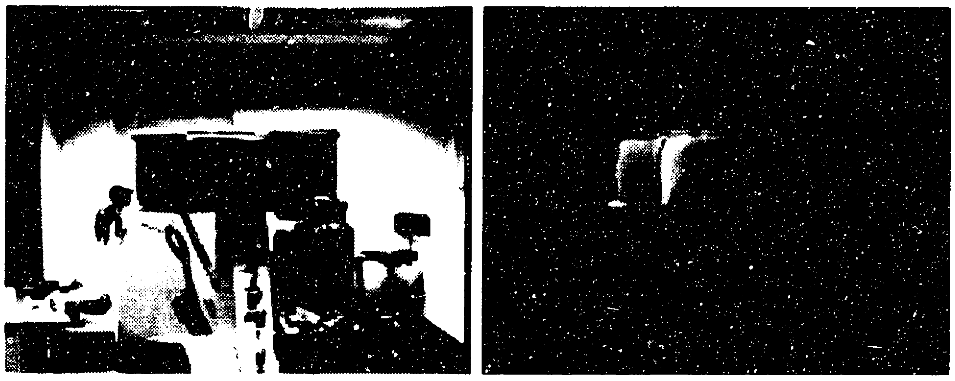


Figure 5.23: Car Image and Final Pose

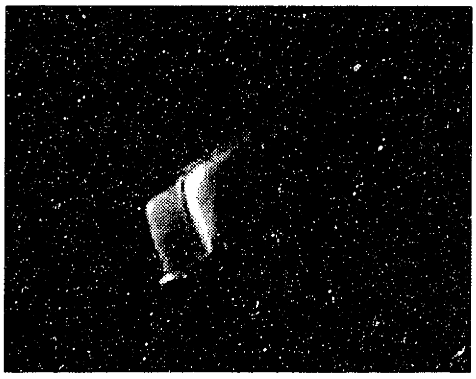


Figure 5.24: Initial Pose of Car Model

where $G()$ is the photometric stereo function that takes two images and two reflectance maps, and $F()$ predicts image intensities.

In practice, however, performing photometric stereo requires detailed metric information about illumination that is only available under very controlled circumstances. One cannot use natural images where lighting is uncontrolled. Luckily, we need not actually know $G()$, r_1, r_2, $F()$, or q. If these functions and parameters exist there will be high mutual information between any novel image and a pair of model images. This is the essence of view based EMMA alignment. We don't actually perform photometric stereo, we simply assume that it is possible. As a result a pair of images should give information about any third image.

To demonstrate this approach we have built a model using the two images in Figure 5.20. Figure 5.23 shows the target image, and the final pose obtained after alignment. Figure 5.24 shows the initial pose of the model.

Technically this experiment is very similar to the MRI alignment experiment. The main difference is that the model $u() = \{u_1(), u_2()\}$ contains a pair of model images. A sample of the model $u(x) = [u_1(x), u_2(x)]^T$ is a two dimensional vector containing the intensity of the two images at location x. This is similar to the two component representation of normal used in the three dimensional alignment experiments. For this experiment $\sigma = 0.1$ and the learning rate was 0.002 for 1000. Experiments demonstrate a capture range of about 40% of the length and width of the car, and rotations of up to 35 degrees.

5.5 Limitations of EMMA Alignment

Several important caveats must be emphasized here. EMMA alignment is *not* a recognition procedure. Though EMMA could well play a role in recognition there are two major missing components. The first missing component is an indexing scheme. EMMA alignment only works when the initial hypothetical pose is "close" to the true pose. A number of experiments have been performed in which an empirical estimate of "close" is determined (see Tables 5.2.1 and 5.2.3). The capture range of alignment is not large enough to expect that a randomly chosen transformation will converge to the true solution. As a result, some sort of additional procedure is required that can rapidly propose possible poses. Such a procedure is typically called an "indexing" scheme.

The second missing component is recognition itself. A recognition task requires that a decision be made about whether the model object is really present in the image. This aspect of object recognition is very similar to the theory of pattern classification (Duda and Hart, 1973) (Fukunaga, 1990). Pattern classification can be formulated as a maximum likelihood or maximum a posteori problem. Each possible model is evaluated in turn. If one model is much more likely then any other, then that model is considered present in the image. It is particularly important that the most likely model be more likely than the null hypothesis, the image does not contain any model. The likelihood of the null hypothesis is proportional of the unconditioned likelihood or entropy of an image. As we have seen log likelihood is closely related to entropy. What is missing from EMMA alignment is a reliable measure of the unconditioned entropy of an image. Though EMMA alignment does estimate the entropy of the part of the image explained by the model, the estimate is a naive one. An assumption that the pixels are independent, underlies the EMMA estimate of image entropy. Though it leads to inaccuracy, the independence assumption has proven sufficient for alignment. This is primarily because alignment is a relative procedure. The model is adjusted so that the image is best explained. Recognition, however, is an absolute procedure. Different models in different alignments must be compared. A more accurate estimate of image entropy will be required before EMMA can be used for object recognition.

Chapter 6

Other Applications of EMMA

The ideas in this thesis have had a number of other applications outside of alignment. This chapter is devoted to a description of these problems and their solutions. The first section is devoted to the correction of images that have been "corrupted" by a bias field. Examples include: MRI corruption that arises from non-uniformity in magnetic field, and lightness correction in visual images. The second section of the chapter is devoted to an application of stochastic gradient descent outside of entropy manipulation. Jones and Poggio have a system that attempts to align line drawings of faces with novel line drawings. Their published work uses a complex second order gradient descent technique known as Levenberg-Marquardt (see (William H. Press and Veterling, 1992) for a discussion of this algorithm and a sample implementation). We will show that similar if not better results can be obtained with stochastic gradient descent, and it operates roughly 25 times as fast.

6.1 Bias Compensation

MRI signals, at least of the head and possibly elsewhere, are made up of distinct values for the distinct tissue classes. In the head there are a number of tissue classes including: bone, water, white matter, grey matter, and fat. In a perfect world the distribution of values should be clustered, with one cluster for each tissue class. This kind of clustering should make classification of MRI pixels easy, just classify a pixel as being of the tissue type of the nearest class (with a correction for prior probabilities). In reality MRI signals are corrupted by something called a bias field which is a additive offset of low spatial frequency. The bias field results from almost unavoidable variations in magnetic fields. Luckily, because of our very powerful visual systems, bias fields are almost invisible to the human eye. We can easily correct for these kind of slow changes. But a naive classifier can't. The pixels no longer fall into to distinct classes.

Wells et al. have built a system where they explicitly modeled the distribution of MRI pixels as a Gaussian mixture, with one Gaussian for each class (Wells III et al., 1994). The first step in their algorithm is to construct an accurate model of the different tissue classes. This model contains an estimated mean, variance and prior probability for each type of tissue. From this density model the likelihood of the pixels in an image can be calculated. They then search for a bias field that makes the data most likely. In other words, find an estimated bias field that when subtracted from the data makes it look like tightly clustered multi-class data. Subtracting the bias field sharpens up the classes and makes automatic classification easy again. This isn't quite enough though. We don't just want any bias field that sharpens up the data, we want low-frequency bias fields. A high frequency, rapidly varying bias field could force some of the data into the wrong classes. Some assumptions about the nature of the bias field is needed to side-step this problem. Wells et al. assume that the bias field is smooth.

Using entropy we can proceed in a much less model-based way. Since Wells et al.'s technique has proven to be quite effective, we can safely assume that the pixel values must be clustered into distinct classes. Any distribution where the density is concentrated at a few values will have low entropy. Corruption from the bias field adds an offset to some pixels and subtracts from others. This spreads out the clusters. The bias field acts like noise, adding entropy to the pixel distribution. This is the central idea behind our approach. We attempt to find the low-frequency bias field that when subtracted from the image, makes the pixel distribution have a lower entropy. The resulting "bias corrected" image will have a tighter clustering than the original distribution. Once again we must be careful to include smoothness constraints.

Insuring the smoothness of the bias field can be tricky. Wells et al. estimate a dense field with one estimate for every pixel and therefore the same frequency response as the image. They insure smoothness by periodically, at every iteration, smoothing the bias field estimates. Another approach would be to represent the bias field as smooth in the first place. This can be done parametrically by finding a smooth parameterized function, or it can be done using a low-frequency bias image, with say 1 pixel for every 10 in the image. Another approach, one which is guaranteed to be better when it is possible, is to represent the bias fields with an explicit physical model. For MRI the bias field should be a low order polynomial of location (M. Tincher and Williams, 1993). We take this approach for correcting MRI scans, representing the bias field as a third degree polynomial in the x and y coordinates of the scan.

The code that minimizes the entropy of the sum of the data and the bias field is very similar to the other entropy manipulation programs we have described. Once again we sample points from the image, x, where each point now has a value, $v(x)$, and a current estimate for the bias field, $b(x)$. Proceeding as we have before, we approximate the entropy of the bias compensated image, $c(x) = v(x) - b(x)$ (see Equation (3.16)). The bias field is adjusted to minimize $h^*(c(x))$ by taking steps in the direction of the derivative as approximated in Equation 3.18. In this case the parameters over which we are minimizing entropy are the coefficients of the bias field polynomial. The derivatives of the bias field, since they are polynomial, are easy to compute.

One can also combine the image and the bias field in other ways. If the image is corrupted by a multiplicative bias field one could instead minimize the entropy of $c(x) = v(x) * b(x)$. The two implementations are close to identical, differing in only two places. Since multiplicative bias fields can be concatenated to form higher order polynomials, multiplicative bias field estimation is strictly more powerful (i.e. $c(x) = v(x) * b_1(x) * b_2(x)$). Additive bias field estimation does not have this property. The concatenation of two polynomial additive bias fields is just a third polynomial bias field of the same order (i.e. $c(x) = v(x) + b_1(x) + b_2(x) = v(x)+b_3(x)$). The bias field of an MRI scan is best approximated as a multiplicative field.

Our initial experiment is a synthetic experiment performed by Wells et al. A binary checkerboard is used as a prototypical example of a two class image. Half of the pixels belong to the black class, the other half to the white class. The pixel

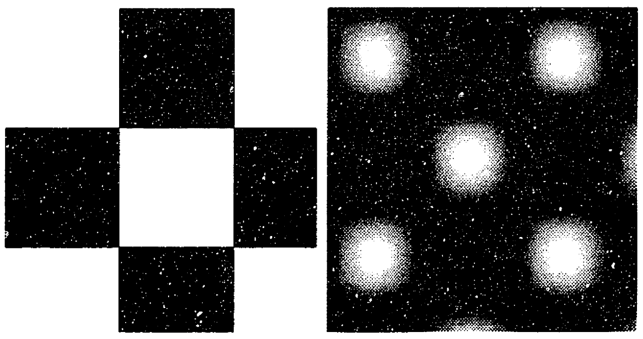


Figure 6.1: The original checkerboard image and the bias field that corrupts it.

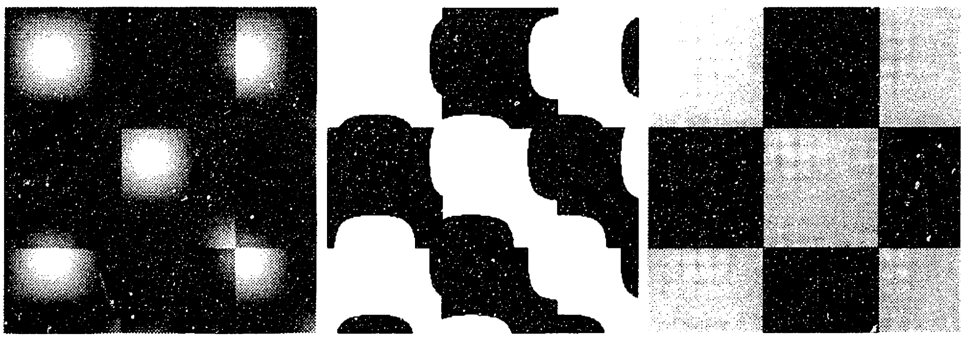


Figure 6.2: Left: the corrupted checkerboard. Center: a thresholded version of the corrupted image. Right: the corrected checkerboard.

entropy of a checkerboard is very low. The checkerboard is then corrupted by a large unknown bias field (see Figures 6.1 and 6.2). The corruption is so large that any cluster structure in the data has disappeared. This is apparent both from the distribution of pixels and a thresholding of the corrupted image (see Figures 6.3 and 6.2). Entropy minimization comes very close to exactly compensating for the bias field. Figure 6.2 show the compensated image. Its distribution is also shown in Figure 6.3. The image is not perfectly compensated because we use a different bias field representation that Wells et al.

One of the goals of the work by Wells et al. is to correctly classify white versus grey matter in the brain (see (Bezdek et al., 1993) for a comprehensive overview of MRI segmentation). They show that classification is much easier if the bias field of a scan is known beforehand. We have performed a number of experiments on real scans of heads with the goal of finding the correcting bias. While their system is designed to give a tissue classification for the corrected scan, ours is not. Instead we will examine the distribution of the corrected scan and attempt to determine if the scan has been corrected in a way that would make classification of white

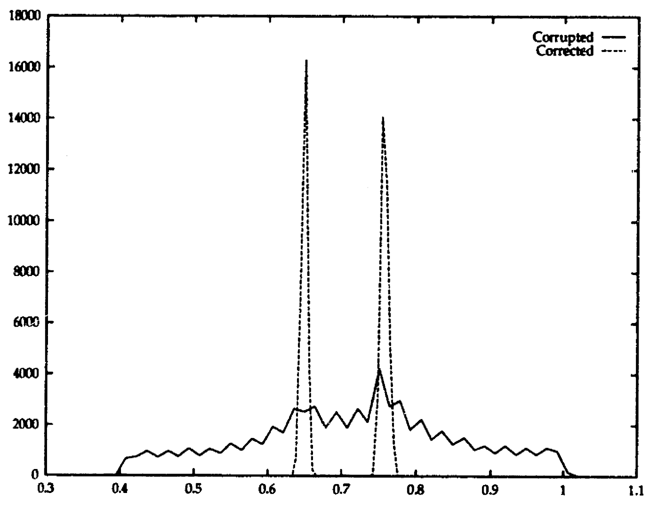


Figure 6.3: The pixel density of the corrupted checkerboard and the compensated checkerboard.

and grey matter easier. Figure 6.4 shows a slice of MRI data taken from a brain. Figure 6.5 shows the histogram of the scan before and after correction. To the left is a very large number of black pixels that results from the surrounding air. In the middle of a graph is the distribution of pixel values from the brain. In the uncorrected image the distribution seems like an undistinguished lump. In the corrected image the distribution is sharpened, showing two peaks, at about 0.48 and 0.58. We can highlight the pixels in this distribution by mapping all the pixels below the first peak to black, all the pixels above the second peak to white, and linearly scaling between (white matter appears darker than grey matter in this MRI scan). Figure 6.6 shows the original and corrected scans in this manner. Notice that the inhomogeneity in the original image becomes immediately apparent. The lower left hand portion of the original scan is dark. The corrected scan does not show this inhomogeneity. The white and grey matter of the corrected scan appear very distinct.

A Second Experiment

The procedure for bias field correction has been repeated for a number of different scans. Figures 6.7, 6.8 and 6.9 show the results of an experiment performed on a coronal slice of an MRI scan.

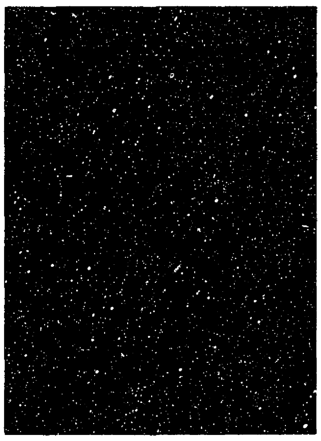


Figure 6.4: A slice from an MRI scan of a head.

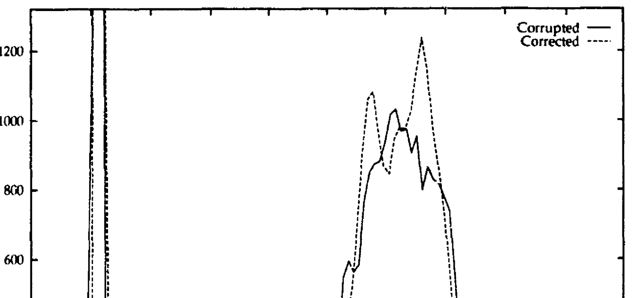


Figure 6.5: The distribution of pixel value in the MRI scan before and after correction.

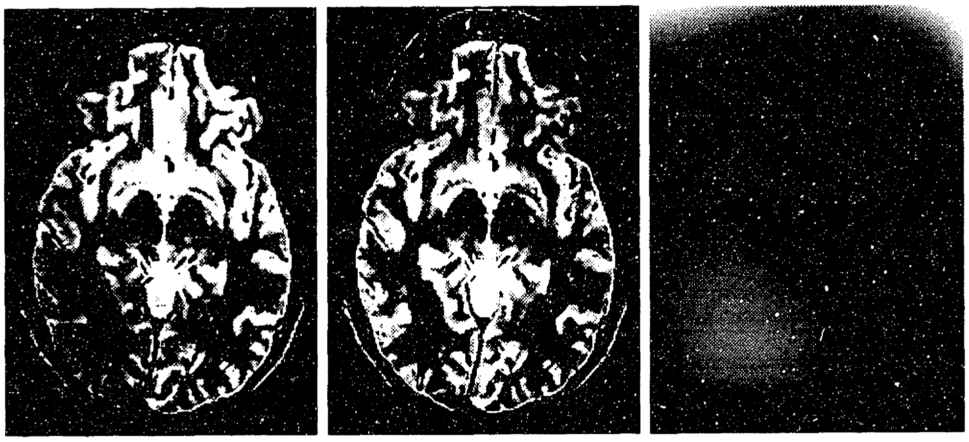


Figure 6.6: The same scan but with a modified intensity scale. Points above the intensity for grey matter appear white. Points below the intensity for white matter appear black. There is a linear scale between. On the right is the estimated bias field.

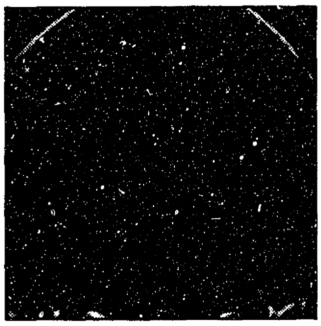


Figure 6.7: A coronal slice from an MRI scan of a head.

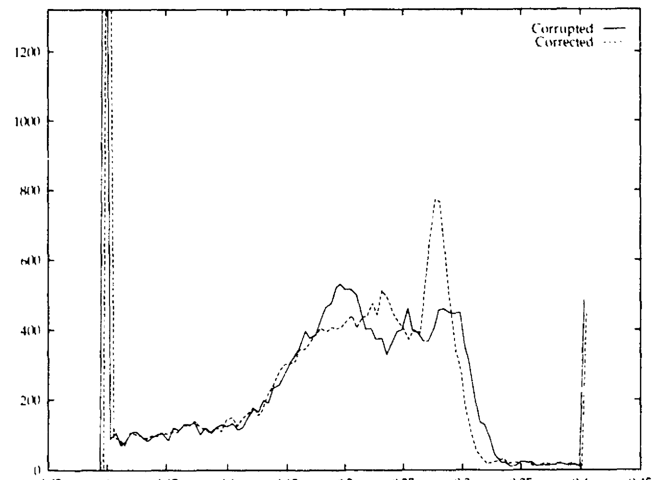


Figure 6.8: The distribution of pixel value in the MRI scan before and after correction. Solid line is the original distribution. The dashed line is the corrected distribution.

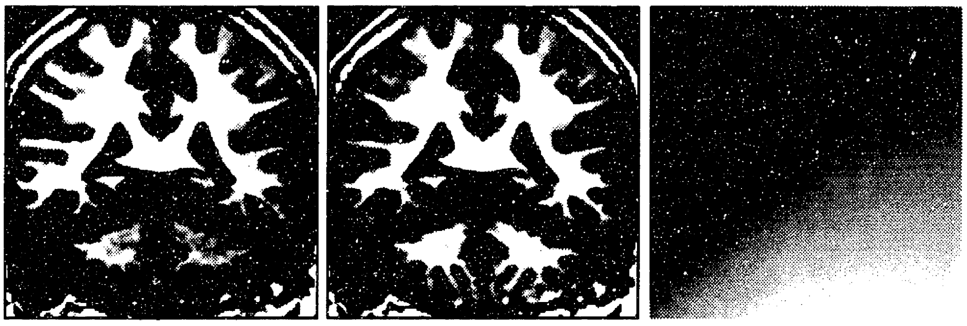


Figure 6.9: The same scan but with a modified intensity scale. Points above the intensity for grey matter appear white. Points below the intensity for white matter appear black. There is a linear scale between. Notice that the lower part of the image is darker in the uncorrected image. On the right is the estimated bias.

Experimental Details

All of the experiments use a smoothing function variance of 0.01. The learning rate was -0.003. The sample size was 30. Every run was concluded after 8000 parameter updates. The p_{min} value was 0.01.

The checkerboard image is 256 by 256 pixels. The checkerboard experiment uses a bias field which is a 20 by 20 pixel low resolution image. This 20 by 20 image is then bilinearly interpolated to create a continuous bias offset for each pixel in the checkerboard. The 400 parameters in this bias field are the most ever simultaneously approximated by an EMMA based technique.

The other images varied in size from 100 square to 256 square. The bias field for the MRI experiments was a 3rd order polynomial in x and y location.

6.2 Alignment of Line Drawings

Jones and Poggio have constructed a system that automatically analyzes hand drawn faces (Jones and Poggio, 1995). Their system works by constructing a non-rigid transformation that maps a novel drawing onto a hand drawn "neutral" face. They use a representation for non-rigid transformations that is called *flow*. A flow is an image of displacement vectors. For each pixel location it records the displacement of that pixel's location. A flow can be applied to an image to create a new image where the pixels have been displaced as directed by the flow.

Jones and Poggio search for a flow that minimizes the difference between the base image $b(x)$ and the novel image $n(x)$,

$$C(W) = \sum (n(x) - b(x + f(x)))^2 \ . \tag{6.1}$$

The problem of non-rigidly transforming one image into another is quite common in computer vision. Flows appear in motion processing and stereo. The general problem usually involves a search over all possible smooth flow fields. Jones and Poggio decompose flow into a linear combination of component flows,

$$C(W) = \sum \left(n(x) - b(x + \sum_i \alpha_i f_i(x)) \right)^2 \ . \tag{6.2}$$

The search for flow becomes a search for the parameters α_i.

Each component flow represents a different type of emotion. The component flows are defined beforehand from a set example drawings. Figure 6.10 contains the average face and some of the emotion examples. The emotional content of a novel face can be determined from the α's required to map it to the base face. Figure 6.11 show several novel images and the best reconstruction obtained by transforming the base face.

In their paper Jones and Poggio use Levenberg-Marquardt, a second order gradient descent procedure, to determine the flow parameters. Together we have replaced this technique with a much simpler stochastic gradient descent procedure. Over the course of many experiments the stochastic technique has improved the speed of the experiments by a factor of 20 to 30. Running times have gone down from about a minute to a second or two. The quality of the minima found with stochastic gradient descent is equivalent to, if not better, than Levenberg-Marquardt. Moreover, stochastic gradient descent rarely if ever gets stuck far from a good estimate for flow. There were a number of cases where Levenberg-Marquardt converged far from the the true solution. On these same problems, stochastic gradient descent finds a plausible solution in almost every case.

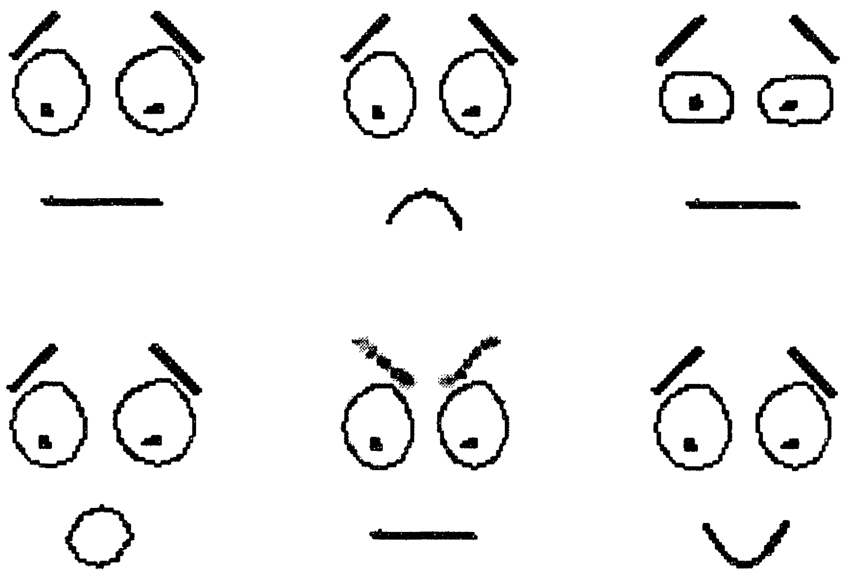


Figure 6.10: The first face is the "neutral" face. The others are "frown", "narrow eyes", "surprise", "eyebrows", and "smile".

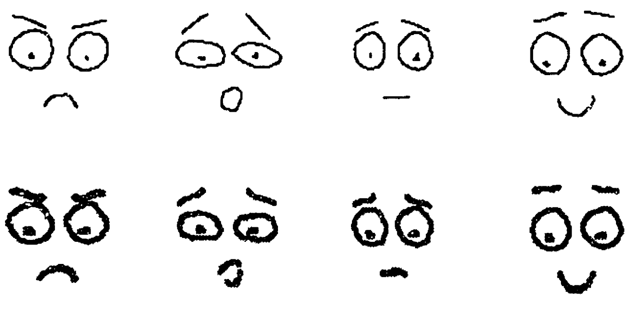


Figure 6.11: A series of novel faces with the best reconstruction displayed below.

Chapter 7

Conclusion

7.1 Discussion

Maximization of mutual information appears to be a new and powerful means of performing local alignment of objects and images. In a typical vision application it is an intensity-based, rather than feature based method. While intensity based, it is more robust than traditional correlation, as shown by the insensitivity to lighting demonstrated in the experiment of Section 5.2.3. In addition, the method is insensitive to negating the image data, as well as a variety of non-linear transformations, which would defeat conventional intensity-based correlation.

The sensitivity of intensity correlation may be corrected, to some extent, by performing correlations on the magnitude of the brightness gradient. This, as well as edge-based matching techniques, can perform well on objects having discontinuous surface properties, or useful silhouettes. These approaches work because the image counterparts of these discontinuities are reasonably stable with respect to illumination.

Gradient magnitude correlation, as well as edge-based methods can have serious difficulties in domains lacking discontinuities, such as the example shown in Section 5.2.3, because neither edges, nor their precursor, gradient magnitude, are stable in image position with respect to lighting changes (see Figure 5.15). While our technique works well using only shading, it also works well in domains having surface property discontinuities and silhouette information (see Section 5.2.2).

Alignment by extremizing properties of the joint signal has been used by Hill and Hawkes (Hill et al., 1994) to align MRI, CT, and other medical image modalities. They use third order moments to characterize the clustering of the joint data. We believe that mutual information is perhaps a more direct measure of the salient property of the joint data at alignment, and demonstrate an efficient means of estimating and extremizing it.

The method we have used for evaluating and adjusting the joint information bears some similarity to methods used for evaluating and adjusting *geometrical* alignment. These similarities may be seen by revisiting the entropy derivative of Equation 3.22, and comparing it to the derivative of the following construct.

We define $\mathcal{D}$, half the averaged Mahalonobis distance between values in B and their nearest correspondences in A,

$$\mathcal{D}(T) \equiv \frac{1}{N_B} \sum_{z_i \in B} \min_{z_j \in A} \frac{1}{2} D(z_i - z_j) \ . \tag{7.1}$$

Locally (away from discontinuities), the derivative of the above expression is

$$\frac{d}{dT}\mathcal{D}(T) \equiv \frac{1}{N_B} \sum_{z_i \in B} \min_{z_j \in A} \frac{d}{dT}(\frac{1}{2} D(z_i - z_j)) \ .$$

Comparing the above expression with Equation 3.22, we see the following analogy. If the transformation T is adjusted to reduce the averaged squared "differences" between points in B and their counterparts from A that are nearest in *signal value*, then a reduction in entropy is obtained. This is intuitive, in that entropy will be lower if clusters in "signal value" are tighter so that nearby signal differences will be smaller. The approximation of this analogy is due to the dissimilarity between max and softmax.

Equation 7.1 is essentially the measure used in chamfer matching techniques, such as the method described by Borgefors (Borgefors, 1988). Huttenlocher (Huttenlocher et al., 1991) has used a related measure in feature matching applications, the Hausdorff distance, which uses maximum instead of the sum that appears in

Equation 7.1. The similarity between geometrical matching and entropy becomes even stronger if one uses the softmax operation to weight the closest element rather than simply selecting the closest, as Wells has (Wells III, 1992b) (Wells III, 1992a).

We reiterate that in vision applications, these methods have typically been used to measure aggregate geometrical distance, while here we are measuring aggregate distances among signal values (typically intensities, brightnesses, or surface properties).

7.2 Related Work

There are many schemes that represent models and images by collections of edges and define a distance metric between them that is proportional to the number of edges that coincide (see the excellent survey articles: (Besl and Jain, 1985; Chin and Dyer, 1986)). A smooth, optimizable version of this metric can be defined by introducing a penalty both for unmatched edges and for the distance between those that are matched (Lowe, 1985; Wells III, 1992b; Huttenlocher et al., 1991). This metric can then be used both for image/model comparison and for pose refinement. Edge based metrics can work under a variety of different lighting conditions, but they make two very strong assumptions: the edges that arise are stable under changes in lighting, and the models are well described as a collection of edges. Clearly smoothly curved objects are a real problem for these techniques. As we alluded before, Wells has performed a number of experiments where he attempts to match edges that are extracted under varying lighting. In general for even moderately curved objects, the number of unstable and therefore unreliable edges is problematic. Faces, cars, fruit and a myriad of other objects have proven to be very difficult to model using edges.

Others use more direct techniques to build models. Generally these approaches revolve around the use of the image itself as an object model. Objects need not have edges to be well represented in this way, but care must be taken to deal with changes in lighting and pose. Turk and Pentland have used a large collection of face images to train a system to construct representations that are invariant to some changes in lighting and pose (Turk and Pentland, 1991). These representations are a projection onto the largest eigenvectors of the distribution of images within the collection. Their system addresses the problem of recognition rather than alignment, and as a result much of the emphasis and many of the results are different. For instance, it is not clear how much variation in pose can be handled by their system. We do not see a straightforward extension of this or similar eigenspace work to the problem of pose refinement. On a related note Shashua has shown that all of the images, under different lighting, of a Lambertian surface are a linear combination of any three of the images (Shashua, 1992). This bears a clear relation to the work of Pentland in that the eigenvectors of a set of images of same object should span this three dimensional space.

Entropy is playing an ever increasing role within the field of neural networks. We know of no work on the alignment of models and images, but there has been work using entropy and information in vision problems. None of these technique uses a non-parametric scheme for density/entropy estimation as we do. In most cases the distributions are assumed to be either binomial or Gaussian. This both simplifies and limits such approaches.

Linsker has used the concept of information maximization to motivate a theory of development in the primary visual cortex (Linsker, 1986). He has been able to predict the development of receptive fields that are very reminiscent of the ones found in primate visual cortex. He uses a Gaussian model both for the signal and the noise.

Becker and Hinton have used the maximization of mutual information as a framework for learning different low-level processing algorithms such as disparity estimation and curvature estimation (Becker and Hinton, 1992). They assume that the signals whose mutual information is to be maximized are Gaussian. In addition, they assume that the only joint information between images is the information that they wish to extract (i.e. the train their disparity detectors on random dot stereograms).

Finally, Bell has used a measure of information to separate signals that have been linearly mixed together (Bell, 1995). His technique assumes that the different mixed signals carry little mutual information. While he does not assume that the distribution has a particular functional form, he does assume that the distribution is well matched to a pre-selected transfer function. For example, a Gaussian is well matched to the logistic function because applying a correctly positioned and scaled logistic function results in a uniform distribution.

Appendix A

Appendix

A.1 Gradient Descent

In a number problems described in this thesis one must find a set of parameters that extremizes an evaluation function. Examples include: (1) finding the parameters of density so that the likelihood of sample is maximized; (2) finding the pose parameters that align a model and an image best; and (3) finding the weights of a neural network so that it approximates a function best. In each case there is a function of a parameters set $F(p)$, whose value is to be either maximized or minimized. The parameters are continuous variables, and we are therefore faced with an infinite number of possible solutions. The gradient descent procedure is an effective though greedy technique for searching such a space.

There are many closely related gradient descent algorithms. Here we will describe the simplest: steepest descent or hill climbing. Starting from an initial guess for the parameters, steepest descent is an iterative procedure that uses the partial derivatives of a function to construct an improved estimate for its parameters. Each parameter is updated by

$$p \leftarrow p + \lambda \frac{\partial F(p)}{\partial p} \; .$$

The update rate λ (which is also known as the learning rate) must be chosen carefully. When λ is sufficiently small one can use a Taylor expansion of $F()$ to prove that

$$F(p+\lambda\frac{\partial log(\ell(a))}{\partial p}) \geq F(p) \ .$$

When λ is too small p might take arbitrarily long to approach a maximum. If λ is chosen correctly p will converge toward the maximum relatively rapidly.

There are many gradient based techniques that attempt to speed the rate of convergence of p. Second order techniques such as Levenberg-Marquart and Newton-Raphson use the second derivatives of $F(p)$ to re-estimate λ. Conjugate gradient techniques attempt to find better directions than the gradient of $F()$. In every case one must be careful that the theoretical advantages of the algorithm are not outweighed by the costs of computing it. Researchers in neural networks have found that for many problems it is difficult to realize any actual improvement in convergence speed. The problems for which steepest descent works as well as more complex techniques include functions where there are a large number of parameters – this makes computing the second derivatives quite expensive.

Bibliography

A. P. Dempster, N. L. and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Baclawski, K., Rota, G.-C., and Billey, S. (1990). Introduction to the theory of probability. MIT course notes for 18.313.

Becker, S. and Hinton, G. E. (1992). Learning to make coherent predictions in domains with discontinuities. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing*, volume 4, Denver 1991. Morgan Kaufmann, San Mateo.

Bell, A. J. (1995). An information-maximisation approach to blind separation. In *Advances in Neural Information Processing*, volume 7, Denver 1994. Morgan Kaufmann, San Francisco. To appear.

Besl, P. and Jain, R. (1985). Three-Dimensional Object Recognition. *Computing Surveys*, 17:75–145.

Bezdek, J., Hall, L., and Clarke, L. (1993). Review of MR Image Segmentation Techniques using Pattern Recognition. *Medical Physics*, 20(4):1033 - 1048.

Borgefors, G. (1988). Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. *IEEE Transactions PAMI*, 10(6):849–865.

Bridle, J. S. (1989). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In Touretzky, D. S., editor, *Advances in Neural Information Processing 2*, pages 211–217. Morgan Kaufman.

Burt, P. and Adelson, E. H. (1983). The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions PAMI*, PAMI-8(6):679–698.

Chin, R. and Dyer, C. (1986). Model-Based Recognition in Robot Vision. *Computing Surveys*, 18:67–108.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory.* John Wiley and Sons.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis.* John Wiley and Sons.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* Harcourt Brace Jovanovich.

Grimson, W., Lozano-Pérez, T., Wells, W., et al. (1994). An Automatic Registration Method for Frameless Stereotaxy, Image Guided Surgery, and Enhanced Realigy Visualization. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, WA. IEEE.

Haykin, S. (1994). *Neural Networks: A comprehensive foundation.* Macmillan College Publishing.

Hill, D. L., Studholme, C., and Hawkes, D. J. (1994). Voxel Similarity Measures for Automated Image Registration. In *Proceedings of the Third Conference on Visualization in Biomedical Computing*, pages 205 - 216. SPIE.

Horn, B. (1986). *Robot Vision.* McGraw-Hill, New York.

Huttenlocher, D., Kedem, K., Sharir, K., and Sharir, M. (1991). The Upper Envelope of Voronoi Surfaces and its Applications. In *Proceedings of the Seventh ACM Symposium on Computational Geometry*, pages 194–293.

Jacobs, R., Jordan, M., Nowlan, S., and Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Jones, M. and Poggio, T. (1995). Model-based matching of line drawings by linear combinations of prototypes. *Proceedings of the International Conference on Computer Vision.*

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Leclerc, Y. (1988). Constructing simple stable descriptions for image partitioning. *Proceedings Image Understanding Workshop*, 1:365–382.

Linsker, R. (1986). From basic network principles to neural architecture. *Proceedings of the National Academy of Sciences, USA*, 83:7508–7512, 8390–8394, 8779–8783.

Linsker, R. (1988). Self-organization in a perceptual network. *IEEE Computer*, pages 105–117.

Ljung, L. and Söderström, T. (1983). *Theory and Practice of Recursive Identification.* MIT Press.

Lowe, D. (1985). *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers.

M. Tincher, C. R. Meyer, R. G. and Williams, D. M. (1993). Polynomial modeling and reduction of rf body coil spatial inhomogeneity in mri. *IEEE Trans. Med. Imaging*, 12(2):361–365.

Mellor, J. (1995). Realtime camera calibration for enhanced reality visualization. In *Computer Vision, Virtual Reality and Robotics in Medicine*, pages 471–475. Nice, France.

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, Inc., third edition.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

Robbins, H. and Munroe, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423 and 623–656.

Shashua, A. (1992). *Geometry and Photometry in 3D Visual Recognition.* PhD thesis, M.I.T Artificial Intelligence Laboratory, AI-TR-1401.

Turk, M. and Pentland, A. (1991). Face Recognition using Eigenfaces. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, Lahaina, Maui, Hawaii. IEEE.

Wells III, W. (1992a). Posterior Marginal Pose Estimation. In *Proceedings: Image Understanding Workshop*, pages 745 - 751. Morgan Kaufmann.

Wells III, W. (1992b). *Statistical Object Recognition.* PhD thesis, MIT Department Electrical Engineering and Computer Science, Cambridge, Mass. MIT AI Laboratory TR 1398.

Wells III, W., Grimson, W., Kikinis, R., and Jolesz, F. (1994). Statistical Gain Correction and Segmentation of MRI Data. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, Wash. IEEE , *Submitted.*

Widrow, B. and Hoff, M. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record*, volume 4, pages 96–104. IRE, New York.

William H. Press, Brian P. Flannery, S. A. T. and Veterling, W. T. (1992). *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, England, second edition edition.

THESIS PROCESSING SLIP

FIXED FIELD: ill. ______ name ______

index ______ biblio ______

▶ COPIES: [✓ Archives] Aero Dewey [✓ Eng] Hum

Lindgren Music Rotch Science

TITLE VARIES: ▶☐ ______

NAME VARIES: ▶☒ Alfred

IMPRINT: (COPYRIGHT) ______

▶COLLATION: 157 p

▶ADD. DEGREE: ______ ▶ DEPT.: ______

SUPERVISORS: ______

NOTES:

cat'r: date:

page: ▶J143

▶DEPT: E.E.

▶YEAR: 1995 ▶DEGREE: Ph.D.

▶NAME: VIOLA, Paul A.