February, 2002

Learning from One Example in Machine Vision by Sharing Probability Densities

Erik G Learned-Miller, *University of Massachusetts - Amherst*

Learning from One Example in Machine Vision by Sharing Probability Densities

by

Erik G. Miller

B.A. Psychology, Yale University, 1988
M.S. Electrical Engineering and Computer Science, M.I.T., 1997

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

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2002

Author .....
Department of Electrical Engineering and Computer Science
February 28, 2002

Certified by .....
Paul A. Viola
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Learning from One Example in Machine Vision by Sharing Probability Densities

by

Erik G. Miller

Submitted to the Department of Electrical Engineering and Computer Science
on February 28, 2002, in partial fulfillment of the requirements for the Degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Human beings exhibit rapid learning when presented with a small number of images of a new object. A person can identify an object under a wide variety of visual conditions after having seen only a single example of that object. This ability can be partly explained by the application of previously learned statistical knowledge to a new setting. This thesis presents an approach to acquiring knowledge in one setting and using it in another. Specifically, we develop probability densities over *common image changes*. Given a single image of a new object and a model of change learned from a *different* object, we form a model of the new object that can be used for synthesis, classification, and other visual tasks.

We start by modeling spatial changes. We develop a framework for learning statistical knowledge of spatial transformations in one task and using that knowledge in a new task. By sharing a probability density over spatial transformations learned from a sample of handwritten *letters*, we develop a handwritten *digit* classifier that achieves 88.6% accuracy using only a single hand-picked training example from each class.

The classification scheme includes a new algorithm, *congealing*, for the joint alignment of a set of images using an entropy minimization criterion. We investigate properties of this algorithm and compare it to other methods of addressing spatial variability in images. We illustrate its application to binary images, gray-scale images, and a set of 3-D neonatal magnetic resonance brain volumes.

Next, we extend the method of change modeling from spatial transformations to color transformations. By measuring statistically common joint color changes of a scene in an office environment, and then applying standard statistical techniques such as principal components analysis, we develop a probabilistic model of color change. We show that these color changes, which we call *color flows*, can be shared effectively between certain types of scenes. That is, a probability density over color change developed by observing one scene can provide useful information about the variability of another scene. We demonstrate a variety of applications including image synthesis, image matching, and shadow detection.

Thesis Supervisor: Paul A. Viola

Acknowledgements

Teachers are the great benefactors of humanity. Considering the respect they are due, but seldom receive, their sacrifice is immense. Many teachers have guided me, both in this work and in life. I thank Ray and Betty Cox for early encouragement. I thank Mrs. Ramey, who demonstrated that the carrot is more powerful than the stick. I thank Jin-Yi Cai for re-igniting my love of math. He said to me once, "AI is very interesting. But Hilbert, more interesting!"

Tom Mowery was my Socrates. His questions provoked a lifetime of thought. He donated a thousand unpaid afternoons to my education. This thesis is his legacy.

I thank Paul Viola, my advisor, for creating a magical and incredibly stimulating environment during my first years at MIT. I will never forget the lively Learning and Vision Group meetings which instilled in me and a whole generation of AI-Labbers the idea that learning and vision cannot be separated. I thank Tommi Jaakkola for great generosity with his time. Even at the busiest times of the semester, he readily listened for an hour or more to an overgrown graduate student struggling with the basics of machine learning. I enjoyed many lively discussions with Berthold Horn, from the good old days of Machine Vision classes through my masters degree and finally as a reader on this thesis. I thank him for his patience and for listening to my hare-brained ideas. I owe a great debt to Eric Grimson, who provided funding, advice, and support at the drop of a hat, while asking nothing in return. He has laid the foundation on which the MIT vision edifice rests.

In addition to these fabulous teachers, there are many others to whom I am indebted. I'll start with the guys at Pu Pu Hot Pot, who kept me going with their great food and friendly demeanor. They work harder than anyone else I know.

Thanks to Tomás Lozano-Pérez and Trevor Darrell for support in times of need. Thanks to Rod Brooks for bringing a lively vision to the AI Lab as director. Thanks to Alan Willsky for being a reader on my thesis, for a great SSG seminar, and for lectures of clock-like precision in 6.432.

I had a lot of help with this thesis in addition to those mentioned above. Mike Ross brought the Renaissance Man's touch to my thesis, correcting gaffes left and right. Kinh Tieu spent the time and effort to do a close mathematical reading. He caught more than a few errors! Polina Golland saved several chapters from total disaster with a massive reorganization effort. I am indebted to all of them. Many co-authors helped me extend my work in new directions: Paul Viola, Kinh Tieu, Nick Matsakis, Chris Stauffer, Sandy Wells, Simon Warfield, and Eric Grimson. It was a pleasure working with all of them. The biggest influences on my thinking probably came from the 10-foot radius around my office: Chris Stauffer, Kinh Tieu, and John Fisher.

The AI Lab has been a family for the past six and a half years. There are too many special people to name. I will miss you all dearly. It was so much fun that work and play were virtually indistinguishable. What a wonderful place to be!

Finally, the people who matter most. I want to thank my parents for being the best a boy could ever hope for. I couldn't have done any of it without you. And finally, the one who put up with all of the late nights, the complaints, and all the stresses of creating this document, the best and most dedicated teacher that I could ever imagine, my best friend, and my love, Carole.

I dedicate this thesis to my parents and to the teacher that started me on this path, my greatest influence, Tom Mowery.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is motivated by the ability of humans to learn a useful model of a class from a small number of examples, often just a single example (Moses et al., 1996). Consider the symbol for the new European currency, the "euro,", shown in Figure 1-1. This symbol was only recently conceived, and many people saw their first example of it during the last few years. Fortunately, after seeing a single example of this character, humans can recognize it in a wide variety of contexts, styles, and positions. It would certainly be inconvenient if people needed 1000 examples of such a new character in order to develop a good model of its appearance.

Since human beings are bound by the laws of probability and estimation, it would appear that we achieve this sparse-data learning by using prior[1] knowledge. Some of the most fundamental questions in machine learning and artificial intelligence concern prior knowledge, namely:

- What is its form?
- How is it obtained?
- How is it applied in new learning scenarios to improve the efficiency of learning?[2]
- Can prior knowledge be used to build a good model of a class from a single example?

The primary focus of this thesis is not to elucidate the exact mechanisms of human learning, but rather to emulate the dramatic learning efficiency of human beings by leveraging the knowledge gained in previously learned tasks.

1.1 Contributions

This thesis formalizes a general approach to acquiring prior knowledge and applying it to new learning problems to improve performance, specifically in the domain of machine vision and pattern recognition. We adopt an explicit form for prior knowledge in vision problems: probability densities over transformations of images. We develop probability densities on spatial transformations and on color transformations, both of which can be applied in a wide range of settings to produce useful object models from a small amount of data.

[1] By *prior* knowledge we mean knowledge obtained prior to a particular task.

[2] *Efficiency* can be thought of, informally, as the number of training examples required to achieve a certain test performance in a particular task.

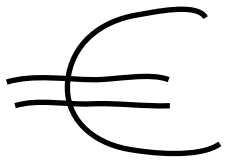


Figure 1-1: A handwritten version of the symbol for the new European currency, the euro.

We establish a novel methodology for modeling new classes of objects based only on samples of related classes (or *support classes*) and a single example of the new class. This is done for object classes that exhibit variability in shape (handwritten digits) and for object classes whose variability is due to lighting change (e.g. the same face under different lighting conditions).

The general approach is to use the support classes to form probabilistic models, not of images, but of *image change*. Changes due to spatial variations are represented by optical flow fields. Changes due to lighting variations and related phenomena are represented by a new type of structure: *color flow fields*. Combining a generic model of image change with a sample of a new object provides a model of the new object.

These object models are used to perform a variety of tasks. In the case of handwritten digit recognition, we develop a classifier based upon just a single randomly chosen example of each digit that achieves mean accuracy rates of 74.7%. This is the best known accuracy on such a task to date. Other classifiers typically score well below this mark even for training sets with several examples. By hand-selecting a single example of each digit for training rather than choosing the example randomly, this accuracy is boosted to 88.6%. This result stands in stark contrast to the usual practice of using thousands of training examples per character.

Another contribution is a new algorithm for the joint alignment of a set of one-dimensional signals, two-dimensional images, or three-dimensional volumes of the same class. This algorithm, called *congealing*, plays a central role in generating models of spatial variability from support sets. It is useful not only for generating these models, but also as a general tool for alignment and complexity reduction.

In the domain of lighting variability, we introduce a new probabilistic model of joint color change in objects due to lighting change and certain non-linear camera parameters. The model is learned from one object, and then can be transferred to a new object. We show how this model of color change, when combined with a single image of a new object, can be used to produce plausible alternative images of the new object. We demonstrate preliminary results in image matching as well as shadow detection and removal. The common theme in this work is that performance of a new task is improved by using knowledge gained from previous data sets or tasks.

1.2 Learning to learn

Using previously learned concepts to improve performance of the current task is not a new idea. The general problem of sharing arbitrary knowledge between tasks, also referred to as *learning to learn* has been considered by many authors. An introduction to the topic of learning to learn is given in (Thrun and Pratt, 1998) and summaries of much of the

relevant work can be found in (Pratt and Jennings, 1998). In this section, three techniques for sharing knowledge from previous tasks are reviewed. These include hierarchical Bayesian analysis, weight sharing in neural networks, and Bollacker and Ghosh's hierarchical classifier architecture.

Given a family of tasks, training experience for each of these tasks, and a family of performance measures (e.g., one for each task), Thrun and Pratt (Thrun and Pratt, 1998) define an algorithm that learns to learn as one for which the performance of each task improves with experience *and* with the number of tasks. Presumably, some kind of transfer must occur between tasks that has a positive impact on task performance if such an algorithm is to be developed. Suppose we identify a particular task of interest, which we shall call the *target task*. As mentioned above, data from sets not directly related to the target task, such as a set of handwritten *letters* provided when the target task is to classify handwritten *digits*, are called support sets. Support sets can provide a useful (or harmful) bias during the learning of a new task.

1.2.1 Hierarchical Bayes

One of the most common ways of leveraging support sets, or prior experience, is hierarchical Bayesian analysis (Berger, 1985; Baxter, 1998). Hierarchical Bayesian analysis is a simple extension of common Bayesian methods in which there is hierarchical structure in a set of data samples. The method is easily understood by comparing it to standard maximum likelihood (ML) and maximum a posteriori (MAP) estimation.

Taking an example from Berger (Berger, 1985), suppose one is trying to estimate the mean bushels of corn per acre for a farm X from a given county, and that we have data (the number of bushels) from several acres on farm X. In the ML approach, we would simply compute the most likely value for the mean given the data, which can be shown to be the sample mean. In this example, it would be the average of the bushel counts from the acres for which data was available on farm X.

In a (non-hierarchical) Bayesian analysis, we would also specify a prior probability distribution on the parameter of interest that incorporates (possibly subjective) prior knowledge. For example, we may have been told by an experienced farmer to expect the mean bushels of corn per acre to be approximately 100 with a standard deviation of about three bushels per acre, and to be distributed in an approximately Gaussian fashion. Along with the observed data, this prior would influence our final MAP estimate of the mean. An empirical mean (based only on the observed data) of substantially less than 100, say 90, based on a small sample of acres would thus be judged as erroneously low and adjusted upward to better fit the prior knowledge of the farmer.

Suppose, however, that we do not have trustworthy prior information about the expected number of bushels of corn from farm X, but that we have reason to believe that the statistics for other farms in the same county should be similar, although not identical, to farm X. In a hierarchical Bayesian approach, we can attempt to incorporate data gathered from other farms (assuming it is available) in an effort to estimate a "prior" distribution on the mean and variance of bushels per acre for all farms. A simple (but sub-optimal) way to do this is to calculate[3] the mean and variance of the bushel-per-acre means using the data from other farms, again assuming that these means are distributed in an approximately Gaussian

[3]Details of hierarchical Bayes calculations can be found in Berger (Berger, 1985), who provides abundant details.

fashion. These so-called *hyperparameters* play the same role as the information imparted to us by the experienced farmer, but this technique represents a more formal way of taking into account experience rather than merely "pulling a prior out of the air."

This type of approach represents a clear and principled way to incorporate data from prior experience, albeit by invoking certain assumptions about the data. The limitation of hierarchical Bayes is that for practical purposes, it requires a strict parametric formulation of the problem. It is often difficult to apply in artificial intelligence settings since most of the distributions that arise are not representable in simple parametric forms. There are, however, a variety of knowledge-sharing methods that have been developed for more complex scenarios.

1.2.2 Weight sharing in neural networks

Another method for learning from related tasks is to simultaneously train a learning machine on multiple related tasks. This approach has been applied to neural networks by many authors as in (Sharkey and Sharkey, 1993; Thrun, 1996; Baxter, 1995).

The idea is that the nodes of a multi-layer neural network can be considered as computing features of the data. Suppose one layer of units is task specific (say, the output layer) and another layer is shared among tasks. The intuition behind this scheme is that if tasks are "similar enough," then they may benefit from similar intermediate layers or representations. Performance can presumably be improved on a task with a small amount of training data by "borrowing" a well-estimated intermediate layer from a task or set of tasks that have abundant training data.

One problem with this approach is that there is no explicit evaluation of whether tasks are similar enough to benefit from shared nodes. Building a network from dissimilar tasks can degrade performance substantially by providing a worse-than-random bias for inference. A partial solution to this problem is addressed in the following scheme.

1.2.3 The Supra-Classifier architecture

Another approach to the problem of learning to learn is given in (Bollacker and Ghosh, 1998). In this work, classifiers developed previously for support classes are used as features in developing a classifier for a target class. That is, a set of positive and negative examples of a target class is evaluated by each of the previous classifiers, and the labels assigned by the previous classifiers are used as additional information about the target class.

The *mutual information*[4] between the training class labels and the output labels from the support set classifiers is used to determine which classifiers provide the most information about the target class.

A simple example serves to illustrate how mutual information can be used to apply knowledge from previous tasks only when it is most relevant. Suppose that we are trying to build classifiers for each handwritten digit from a small number of examples but that we have already built classifiers for each class of handwritten letters. Consider a set of training zeroes that have been labeled as zeroes and a set of additional images that have been labeled as non-zeroes. It is highly probable that the classifier for the letter "O" will generate nearly

[4]The mutual information between two random variables X and Y is defined to be $H(X)+H(Y)-H(X,Y)$, where $H(\cdot)$ is the discrete entropy function, and $H(\cdot,\cdot)$ is the discrete joint entropy function(Cover and Thomas, 1991). Mutual information can be thought of as the amount of information in a random variable X about Y, or vice versa.

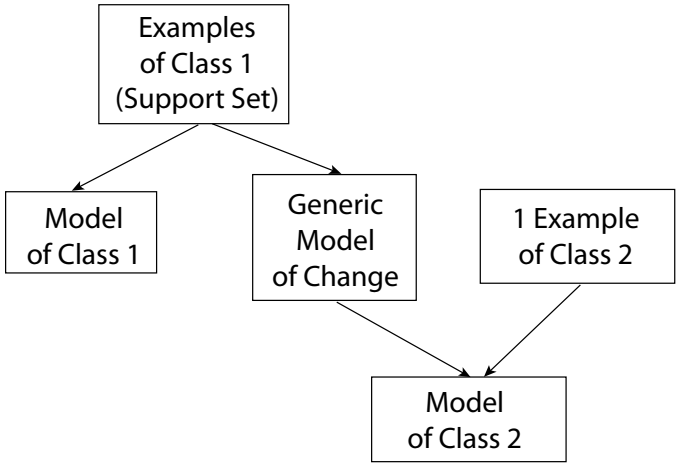


Figure 1-2: Using a model of change, derived from a support set, in creating a new object model.

the same set of labels (either "in class" or "not in class") for these training images of zeroes as they are labeled with, since zeroes and ohs are very similar in appearance. More formally, the "zeroness" of images and the "oh-ness" of images, viewed as random variables, have high mutual information. The supra-classifier scheme would thus heavily weight the vote of the "oh" classifier in determining whether any new image were in fact a zero.

Perhaps the most appealing aspect of this architecture is that there is an explicit scheme for using prior information only when it is relevant. This should greatly reduce the probability that performance in such a scheme will actually get worse when incorporating prior experience. Since the technique explicitly estimates the "relatedness" of tasks, no a priori assumptions about the relationship of tasks needs to be made. This is a highly desirable aspect of any method of learning to learn, and we shall retain it in our work.

1.2.4 A machine vision approach

All of these approaches are interesting, but are not necessarily well adapted to the domain of machine vision. As noted above, hierarchical Bayesian methods work well for simple parametric modeling, but are not practical for more complex distributions. Neural network approaches and the supra-classifier architecture, on the other hand, are so general that they fail to take advantage of the specific structure in machine vision problems. The methods of learning to learn proposed in this thesis are designed specifically to work with images (or other continuous signals). By implicitly taking advantage of the properties of images, we are able to make greater progress in learning to learn in this narrower domain.

1.3 Models of images vs. models of image change

It has been observed that the distribution of "natural" images, meaning images of common scenes and objects, is very different from the distribution of purely random independent pixel images. Machine vision researchers have tried to incorporate knowledge of natural images

in machine vision problems by producing universal models of natural images. Models of the principal components of natural images (Baddeley and Hancock, 1992), the independent components of natural images (Bell and Sejnowski, 1997), and many other properties of natural images have been proposed. These models, however, have proven very weak in providing a bias for specific image class models: there is simply too much variability in natural images to produce a powerful model in such a direct fashion. While it is true, for example, that many images exhibit a characteristic distribution of spatial frequencies (approximately), many other images do not. At best, we can put a weak prior on such generic features as spatial frequency distributions in a single image.

Implicitly, these natural image models seem to suggest that objects are somehow similar in appearance. In this thesis, we take a different viewpoint, as suggested by the work of Black, Fleet, and Yacoob (Black et al., 2000; Black et al., 1998; Black et al., 1997a). We suggest that the ways objects *change appearance* are similar. We hypothesize that people's success in object recognition is possible partly because a wide variety of objects *tend to change appearance in similar ways*. This property has been described and exploited in the domain of face recognition by Lando and Edelman(Lando and Edelman, 1995) and by Edelman and O'Toole (Edelman and O'Toole, 2001).

An almost trivial and yet essential example of a common mode of image change is the common spatial transformations that objects undergo from image to image. In different photographs of the same object, the object will appear in different positions, orientations, and sizes in the image. While this type of variability is hardly even noticed by human beings, it causes severe problems for machine vision programs.

It is well known in the machine vision community that a good object recognizer will have to address spatial and other types of variability. Many researchers deal with this variability by requiring the user to produce large numbers of training examples which include examples of this variability. Some have artificially created large training data sets by perturbing each training image according to known modes of variability (Simard et al., 1993; Beymer and Poggio, 1995). However, this approach has difficulty as the number of modes of variability grows large. In particular, the number of examples needed to model a class well tends to grow exponentially with the number of modes of variation one is trying to model.

In this work, we model common modes of change explicitly, rather than merely providing samples of that change. By representing *typical image changes* in certain ways, our models of image change can be shared among many different types of objects. Then, if we develop a good model of image change for one object, we can transfer it to a new object for which we may only have a single example image. This scheme is illustrated in Figure 1-2. As we shall see, the success of such a scheme depends critically upon the right representation of change.

1.4 Representations of image change

Suppose we wish to represent the difference between two images of an object that are the same except for a small rotation and magnification. The first row of Figure 1-3 shows two images of the digit "2" and then two representations of the change between the images.

1.4.1 Flow fields

The first representation of change is simply the pixelwise difference between the two images. The second representation of change (far right of first row in Figure 1-3) shows the change

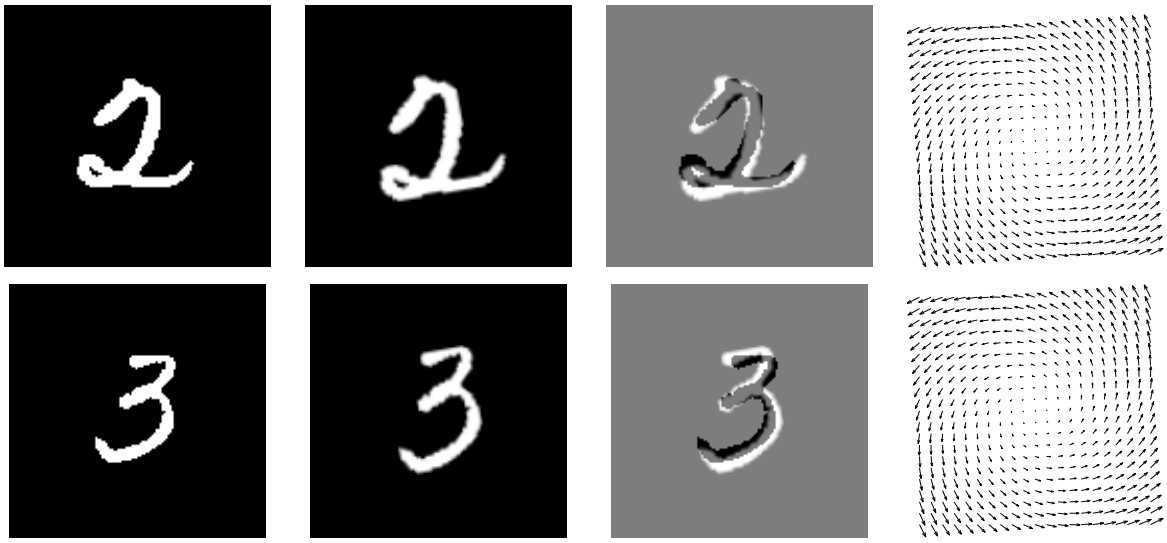


Figure 1-3: Two methods for representing *spatial* differences between images. In the first column are samples of various handwritten digits. In the second column are slight spatial distortions of those digits. In particular, they have been slightly rotated and magnified. In the third column is the result of subtracting the image in the second column from the image in the first column. Learning what such differences for one object look like tell us very little about what they might look like for other objects. In the fourth column is the flow field representation of the change. The fact that these have the same form for different classes enables us to share models of image change when they are represented in this fashion.

between the images of "two"s as an optical flow field. Each arrow in the optical flow field shows how a pixel should be transported in order to produce the new image. Thus taking the original image and the optical flow field, we can recover the second image. The key feature of this representation is that, unlike the difference image representation, it also works for any other type of image, namely, the "3" images. This suggests one of the guiding principles of this thesis: *Certain representations of change can be easily transferred among objects; others cannot.*

We use a similar idea to model lighting changes in a way that can be shared across objects. This will allow us to predict the appearance of a new object even when we have seen that object under only a single illumination.

Consider Figure 1-4 which has a structure parallel to that of Figure 1-3. Again, pairs of images are shown, but in this case it is the lighting that has changed, rather than the spatial positioning of the object. Again, a simple image difference (shown in the third column) is only meaningful for a single object. However, the representation in the fourth column is the same for both pairs of images. These *color flow fields* are to be interpreted as follows. The three-dimensional volume in which the vectors sit represents all possible colors, with axes of hue, brightness, and saturation. Each vector thus starts at one color and ends at another. When applied to a particular image, then, these vector fields describe how each color in one image is mapped to a color in the next image. By leveraging the fact that *color changes* have similar structure for different objects, we can develop a more general model

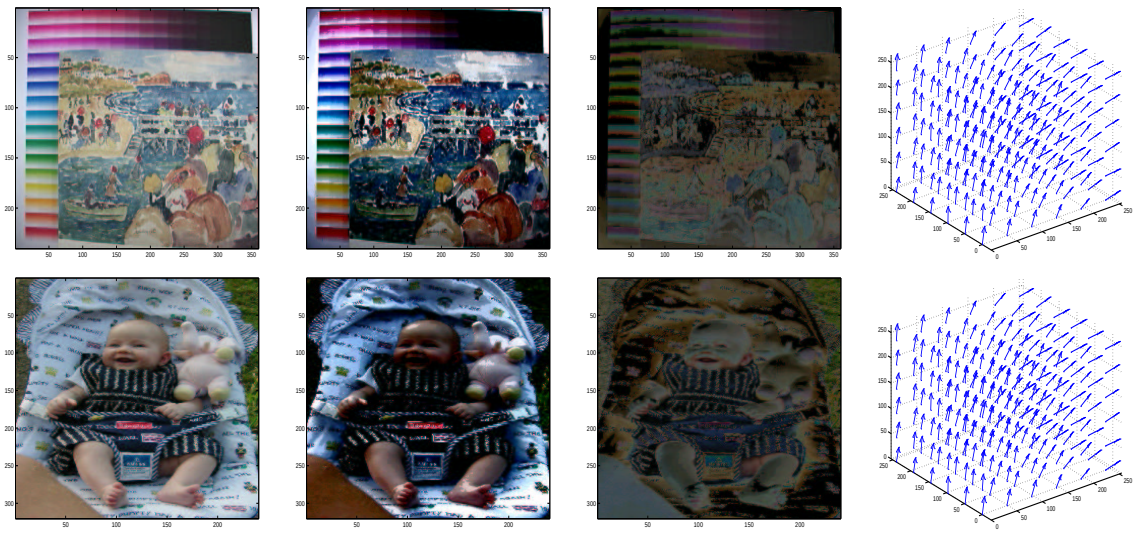


Figure 1-4: Two methods for representing *lighting* differences between images. Each image pair (in the first two columns) represents the same object but under different simulated lighting conditions. Representing these lighting changes as image differences (third column) shows no similarity between the changes. However, representing them as color flow fields (fourth column) reveals that they represent the same mapping of colors in one image to colors in the new image (see text). Such generic models of color change can be applied to a wide variety of object recognition problems (Miller and Tieu, 2001a).

of color change by using this type of representation.

Optical flow fields and color flow fields are both examples of what we call *feature flow fields*, collections of change vectors for features, represented in an appropriate space. Probability densities over feature flow fields will constitute the primary form of shared knowledge in this work.

1.4.2 Tangent vectors

Before describing models of change based on feature flow fields, we make a few more observations about difference image representations. Such representations have been used to model invariance in classes of digits, as in (Simard et al., 1993). While such a representation tells us how to create the second "2" image in Figure 1-3 from the first "2" image, (namely add the "2" difference image to the first image), the "2" difference image tells us little or nothing about how to change the image of the "3" to create another "3". Even for images of the same class, it is an accurate representation of the difference image only for a small portion of the image class.

Images as vectors

This difficulty with the difference image representation of image change can be understood by viewing the image as a vector in a high-dimensional space, where each component of the vector is the value of a particular pixel. This image representation is ubiquitous in the machine vision literature.

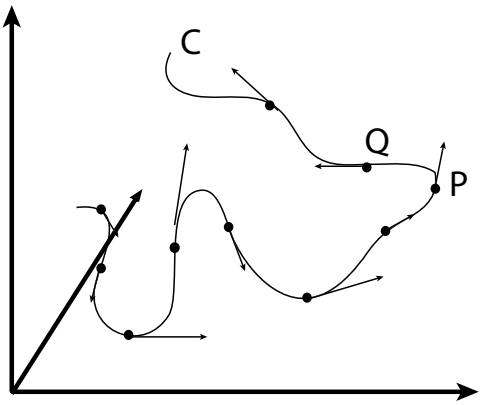


Figure 1-5: A one-dimensional image manifold parameterized by rotation. The three-axes represent the high-dimensional image space. A set of images parameterized by a single variable such as rotation defines a one-dimensional manifold in image space. Tangent spaces are not good approximations to a highly curved manifold. A manifold of an image with edges, under rotation or translation, has curvature too great to be well-represented by a small number of tangent spaces.

Consider the one-dimensional set of images created by changing an imaging parameter such as the rotation of the camera. Such a variation traces a one-dimensional manifold in the image vector space. This one-dimensional manifold of the rotation of an image can be schematically "visualized," as the curve C in Figure 1-5. The difference image then gives us an approximation to the tangents to the curve (shown as arrows in the figure). The ability to model another image on the manifold (Q for example) from an image P and the tangent at P is related to the curvature of the manifold. If the curvature is high, the approximation will be poor. If it is low, i.e. if the manifold is locally flat, the approximation will be good.

To determine over what range the manifold of a "2" image as it is rotated can be approximated as flat, we can examine the inner product between two nearby unit tangent vectors. If the inner product is near 1, the two tangent vectors point in nearly the same direction and the manifold is fairly flat, whereas if the inner product is near 0, the two tangents are nearly orthogonal and that the curve is rapidly curving away from its previous direction. For sample images such as those shown in Figure 1-3, the inner product between a tangent vector taken at the 0 rotation angle (represented abstractly by the point P in the Figure 1-5) versus a tangent vector taken at the 5 degree rotation angle (point Q) was only 0.0675. This means that even over the relatively small rotation of 5 degrees, the tangent vectors are almost completely orthogonal. This in turn suggests that tangent vector representations of change are not valid over a very large range of the relevant spatial parameters like rotation and translation.

This severe curvature can be reduced by blurring images. That is, the "rotation" manifold of a blurred image typically has lower average curvature than the rotation manifold of the original image. Blurring eliminates sharp edges, which in turn, reduces the curvature in manifolds parameterized by simple spatial transformations like translation, rotation, and scaling.

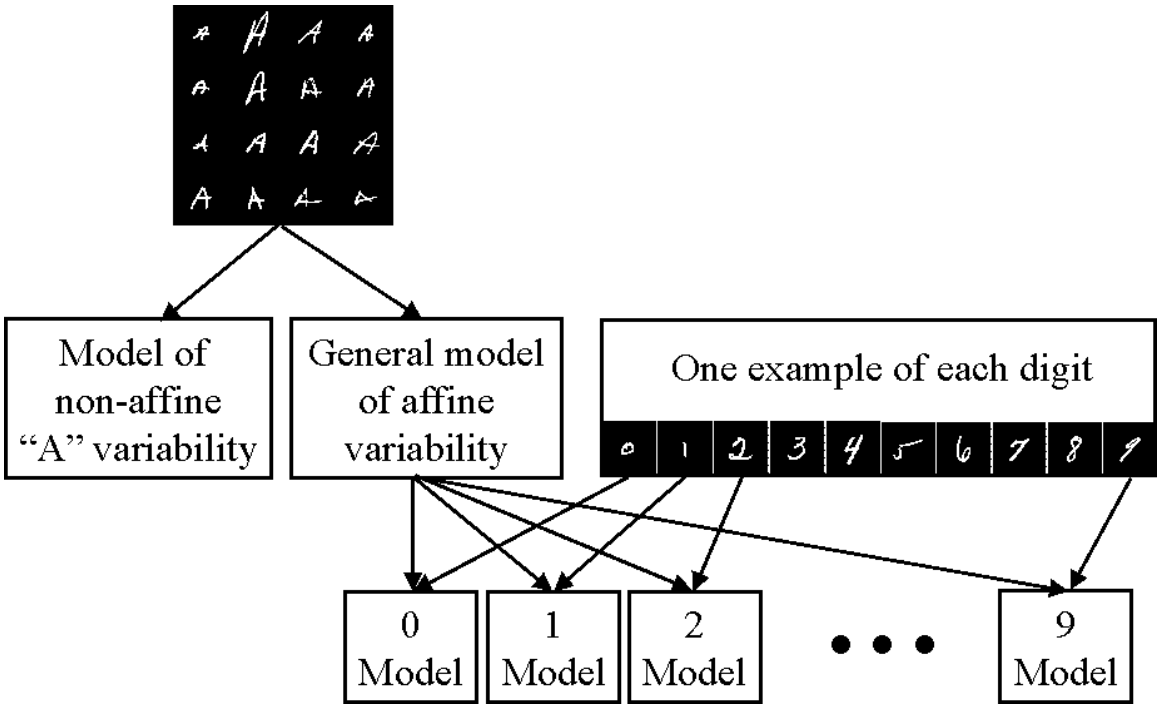


Figure 1-6: Diagram of our method for sharing models of affine variability. A support set (shown as a set of "A"s) is given to the learner. From this support set, a general model of affine variability is derived. Combining the model of affine variability with a single example of a handwritten digit, a model is made for the digit that incorporates the spatial variability.

This blurring approach is taken by Simard et al. in their use of tangent approximations of transformation manifolds (Simard et al., 1993). Such blurred image manifolds are represented much better by a small number of points and their tangents than are non-blurred image manifolds. However, since blurring destroys information about the original images, we favor an alternative approach to representing image change that preserves the information, feature flow fields.

1.5 Models of change from flow field data

The bulk of this thesis is about developing shareable models of change from flow field data. In this work, we focus on change due to spatial transformations and change due to lighting variability.

1.5.1 Models of spatial change

The first major application of shareable models is in developing a handwritten digit recognizer from one example of each digit class. In addition to the one example of each digit class, the classifier is provided with support sets consisting of large sets of handwritten letters.

The basic approach is to use the handwritten letter support sets to form models of

spatial variability, which are then combined with the single example of a digit to produce a probabilistic model for each digit class. This scheme is shown in Figure 1-6.

A classifier built using this scheme attains accuracy of about 75% on handwritten digits taken from the NIST database. This accuracy was improved to 88.6% by hand-picking the single training example to be "prototypical," emulating a teaching scenario. We are not aware of any reported performance on single training character classifiers, presumably because the performance is too low to be of interest.

1.5.2 Models of color change

The second major application of shareable models is in modeling color changes in images due to lighting changes and non-geometric camera parameters. In this part of the thesis, a statistical model of color changes is developed using one object, and transferred to a single example of a new object. This approach is diagrammed in Figure 1-7.

This technique can be used in a variety of useful vision tasks. It can be used to infer that two images do or do not represent the same scene or object, to identify shadows in images and remove them, and to synthesize new examples of an object for a variety of purposes. Examples of all of these applications are presented.

1.6 Reader's Guide

This thesis addresses a wide range of topics. Some readers may be interested in specific portions of the work, and not others. The first part of the thesis (Chapters 2, 3, and 4) focuses on developing a classifier for handwritten digits using only a single training example for each class and support sets consisting of handwritten letters. The classifier is built by forming a probabilistic model of spatial changes in images using the support sets. This model is then transferred to each handwritten digit example to form models for each digit class.

Specifically, Chapter 2 provides a brief introduction to supervised learning and the problem of handwritten digit recognition. It examines common criteria for evaluating classifier performance, and suggests that other criteria, such as maximizing performance on a small amount of training data, may be more appropriate in certain real-world scenarios. For the reader familiar with supervised learning, Chapter 2 can be skipped without loss of continuity.

Chapter 3 presents a factorized model for handwritten digits, where one factor represents the variability that can be represented with affine transformations and the other factor represents the remaining variability. A new algorithm for the joint alignment of images, *congealing*, is presented. The output of the congealing algorithm can easily be used to create a factorized model of images. It is argued that such factored representations, while not asymptotically optimal, may perform better in classification tasks than non-factored models for small amounts of data. Experimental results are reported for fully probabilistic classifiers and for variations on these classifiers based on the Hausdorff distance.

Chapter 4 is the culmination of the first half of the thesis. It shows how one of the factors in the models from the previous chapter can be borrowed from the model for a support set. That is, since modes of spatial variation tend to be the same from character to character, the probability density over spatial transformations for letters is similar to the probability density over spatial transformations for digits. After verifying the similarity of these distributions via certain statistical tests, models for handwritten digits are created by

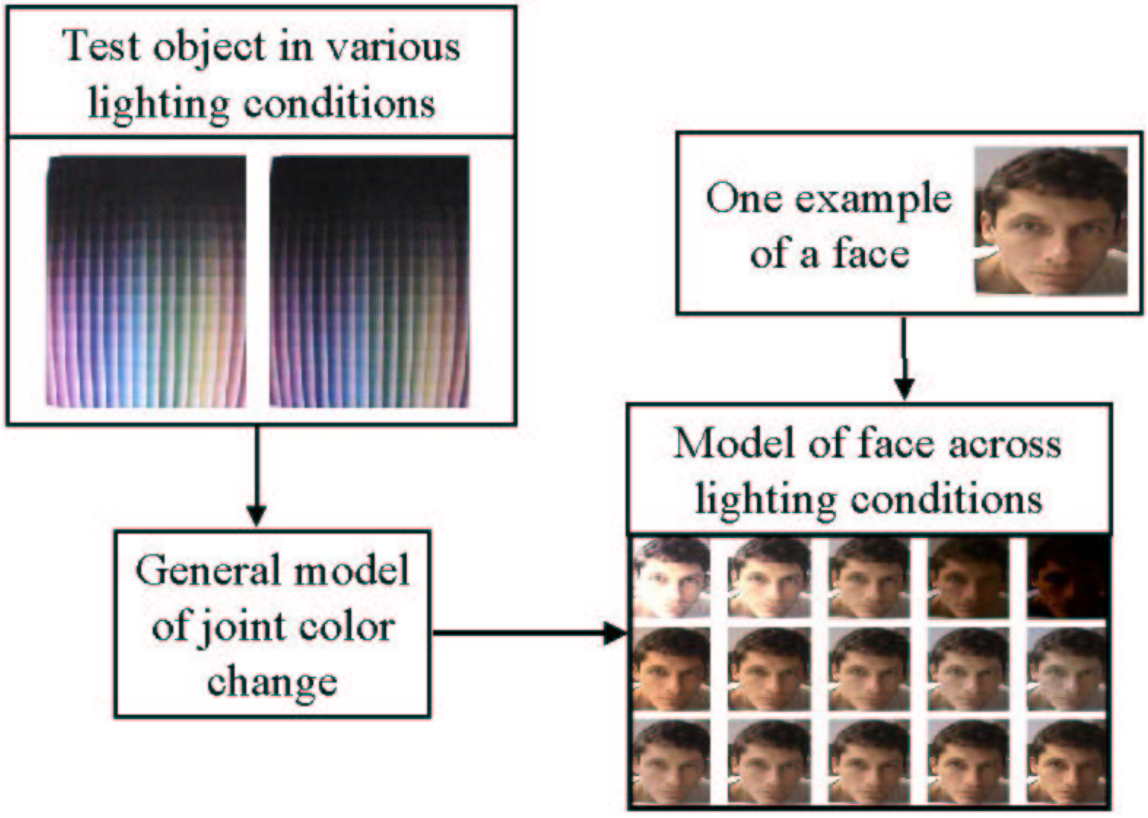


Figure 1-7: Diagram of method for sharing models of joint color variability. By representing typical joint color changes of a test object (upper left of figure), a general model of joint color change is developed. This model, combined with a single example of a new object (a face in this case) produces a model of the new object under common lighting changes (lower right).

borrowing transform densities from handwritten letter support sets. Classification results on a variety of experiments are reported, including 88.6% accuracy on a hand-picked single training example classifier.

Chapter 5 can be thought of as an appendix to the first part of thesis. It provides substantial additional detail about the joint alignment algorithm, congealing. It discusses the algorithm's robustness to noise, convergence properties, algorithmic complexity, and applicability to other types of signals such as EEG data and three-dimensional brain volumes of magnetic resonance images. This chapter is not directly related to the main theme of learning to learn, except that this technique is central to the implementations in the first half of the thesis. It can also be skipped without missing the main points in the thesis. However, it may be of interest to vision researchers interested in issues of invariance, correspondence, and alignment of images to models.

The second major part of the thesis, contained completely in Chapter 6, discusses color flow fields and probability densities over them. These models of joint color changes in images are acquired from video data of a natural setting as the lighting changes. The models are then transferred to new objects. Applications in synthesis, image matching, and shadow

detection are discussed. This chapter may be of particular interest to those interested in color constancy and lighting invariance issues.

Chapter 7 is a short chapter whose goal is to unify spatial change models and color change models under the framework of feature flow fields. It discusses common properties of feature flow fields and possible extensions of them to other domains.

Chapter 2

Background for Classification

There has been such great progress in the development of classifiers for problems such as handwritten digit recognition that many researchers consider the problem solved. Classifiers such as Belongie, Malik and Puzicha's shape matching method (Belongie et al., 2001) and LeCun et al.'s convolutional networks (LeCun et al., 1995) achieve performance very close to that of human test subjects, a commonly assumed benchmark of "optimal" performance. It seems that performance cannot get much better. However, these methods require large amounts of training data. For example, LeCun et al. use 6,000 samples of each character in training. This leaves open the question of whether these methods are appropriate in cases for which large amounts of training data are not available.

In this short chapter, we introduce the general problem of supervised learning and one of the most common instantiations of it in the field of pattern recognition: handwritten digit classification. We consider alternatives to the common criterion of judging a classifier by the minimization of test error in the presence of large training sets. We explore four criteria for evaluating the performance of handwritten digit classifiers: asymptotic expected error, error rates using large training sets, robustness to assumptions, and error rates using small training sets. Each of these criteria is important in certain domains, and may be unimportant or even entirely irrelevant in other domains. If we examine the performance of classifiers using only a few examples per class, there is still a large gap between the capabilities of machines and humans. This is particularly pronounced in visual pattern recognition, where adult human beings have a vast amount of prior experience. We argue that learning to classify from a small number of examples is both highly relevant as a general issue in artificial intelligence and that it is relatively unexplored by the machine learning community.

2.1 The classical problem of handwritten digit recognition

The recognition of handwritten digits produced by different writers has become a standard problem in the field of visual pattern recognition.[1] This is probably because the problem is easy to understand, data is readily available (there are standard training and testing databases), humans perform the task almost without error (on the standard databases), and there are clear applications (sorting mail, reading checks (LeCun et al., 1997)).

[1] There is some disagreement as to whether handwritten digit recognition constitutes a problem in *machine vision*.

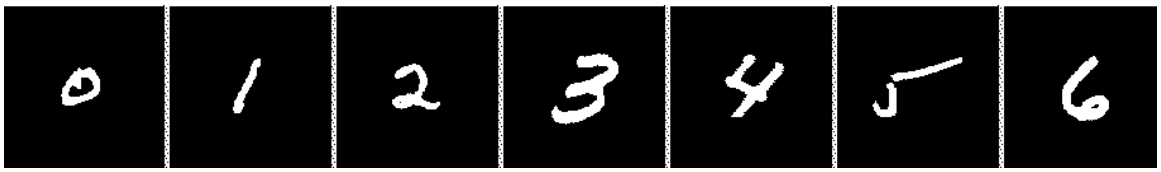


Figure 2-1: Typical samples of handwritten digits from the NIST database.

2.1.1 Supervised learning

Handwritten digit recognition can be cast as a supervised learning problem. The standard supervised learning problem is commonly formulated as follows. During a training phase, the learner is given a set of labeled examples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ with $\mathbf{x}_i \in \mathcal{X}$, the set of possible examples, and $y_i \in \mathcal{Y}$, the set of labels. The goal is to learn a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$ from examples to labels so as to minimize the sum (or integral) of some loss function $l(y, f(\mathbf{x}))$ over some unknown test set of corresponding pairs also drawn from $\mathcal{X} \times \mathcal{Y}$.

In a typical case of handwritten digit recognition, the training data are a collection of labeled digits, usually represented as binary or gray-scale images. The test data are a set of unlabeled digits, disjoint from the training set. And the loss function is simply 1 if the learner's guess of the digit is incorrect and 0 if the learner's guess is correct. (This is known as the *symmetric* loss function, since all types of errors are penalized equally.) Many algorithms have been developed and used for handwritten digit recognition (see (LeCun et al., 1995) for a comparison of several common algorithms). Figure 2-1 shows a typical sample of handwritten digits from the National Institutes of Standards and Technology (NIST) Special Database 19 (Grother, 1995).

2.2 What is a "good" recognition algorithm?

While it may seem straightforward to define the "best" character recognition algorithm corresponding to a particular loss function, the issue is not as simple as it first appears. For the symmetric loss function, the intuitive desire is to have an algorithm that correctly classifies as many characters as possible from an unknown test set. This leads researchers to simply try to produce a classifier that has minimum expected error on a test distribution, using all available training data. The vast majority of research on the handwritten digit recognition problem uses this criterion.

However, upon looking closer, we see that there is a wide range of potentially appropriate criteria for the problem in different settings, even given the same loss function. We examine four quite different criteria below, assuming the symmetric loss function in each case.

2.2.1 Criterion 1: Asymptotic expected error

A commonly evaluated theoretical criterion of classifier performance is whether the expectation of the error approaches the minimum possible value as the number of training examples approaches infinity. Suppose that the probability distribution of each class j, $1 \leq j \leq K$, in a classification problem is given by the class conditional probability density

$p^j(I) = Prob(I|C = j)$, where C is the class label.[2] If these class densities have overlapping support (of non-zero measure) and each class has non-zero prior probability, then it is impossible to produce a classifier with 0 expected error. The *Bayes error rate*, or simply the Bayes rate, is the minimum achievable expected error rate for a classification problem, where the expectation is taken with respect to the joint density over examples and class labels $p_{\mathbf{x},y}(\mathbf{x}, y)$. If we are seeking a classifier that achieves the Bayes error rate as the training set size goes to infinity, then we need look no further. There are many examples of such classification schemes.

The k-nearest-neighbor classifier

For example, the k-nearest-neighbor rule approaches the Bayes error rate as the training set size N approaches infinity, provided that we let k, the number of neighbors examined, also grow at a rate slower than the rate at which the data set grows, for example, at $k = N^{\frac{1}{2}}$ (Duda and Hart, 1973; Devroye et al., 1996).

An optimal maximum *a posteriori* classifier

Another well-known approach to producing an asymptotically minimum error rate classifier, assuming that the densities from which the characters are drawn obey certain smoothness conditions, is to estimate $p^j(I)$ and $p(C = j)$ and build a maximum *a posteriori* classifier using

$$c^*(I) = \arg\max_{c \in \mathcal{C}} p(I|c)p(c). \tag{2.1}$$

To do this, simply estimate the class conditional image densities $p^j(I)$ using a convergent non-parametric density estimator, such as the Parzen kernel estimator (see (Parzen, 1962)). To do this, we simply estimate each probability density as

$$\hat{p}^j(I) = \frac{1}{N} \sum_{i=1}^{N} K(I_i, I), \tag{2.2}$$

where $K(I_i, I)$ is a unit-mass kernel. A common kernel is the spherical $P-$dimensional Gaussian density $\mathcal{N}(I; I_i, \sigma^2)$, with mean I_i and variance σ^2. Here, and throughout the thesis, P is the number of pixels in the image. As the number of data points gets large, we can afford to make the kernel variance σ^2 smaller and smaller. In the limit of infinite data and a kernel which approaches a delta function, the estimate $\hat{p}^j(I)$ converges to the true probability density $p^j(I)$.

Since the density estimates will converge to the true class conditional densities, we can classify any test sample with minimum probability of error by simply selecting the class with maximum likelihood (ML). That is, a classifier based upon these estimates will converge to a "perfect" classifier asymptotically in the sense of Bayes minimum error.

Of course, from a practical point of view, this asymptotic optimality may not help us produce a good classifier[3] for a particular task since it is often impractical or impossible

[2] When discussing supervised learning and other learning scenarios in a general context, we will use $(\mathbf{x}, y)$ to refer to the (sample,label) pair. In the context of visual pattern recognition, we use the alternative variables (I, c) to denote the (image,class) pair for their heuristic value.

[3] There are many ways to define the "goodness" of a classifier. We can define an $\epsilon-$*good* classifier as a classifier whose expected error is within ϵ of the Bayes rate.

to collect enough training examples to reduce our error probability to within some ϵ of the smallest possible value. In fact, in most visual pattern recognition problems, the class conditional densities are too high-dimensional and complex to be represented in such a general form in any current digital computer, even if enough training data were available to estimate these densities well. Hence, while asymptotic optimality is a desirable property of a classifier, it is certainly not a sufficient property to satisfy practical requirements in most cases.

2.2.2 Criterion 2: Error rate for large training sets

Rather than focusing on theoretical properties of classifiers, the most common approach in building handwritten digit classifiers has been to optimize performance on a test distribution using large training sets. Typically, when classification results are reported for standard data sets, a fixed number of training samples are available, and all of the samples are used. While there are some exceptions, particularly recently as in (Simard et al., 1998), most papers report results only for the full training set.

For the problem of handwritten digit recognition, several standard databases of handwritten digits have been created. They contain standard training sets and test sets. Three of the most common databases for digit recognition are the United States Postal Service (USPS) Database, the NIST database, and the Modified NIST (MNIST) database. The total number of training samples available in these three data sets (summed over all digits) is 7291, 344307, and 60000 respectively. These are the training sets for which results are most commonly reported.

There are many scenarios in which Criterion 2 is the appropriate criterion with which to evaluate a classifier. Suppose that one is engineering a system for use in real world applications, and that furthermore, one can make a reasonable argument that the training data, test data, and the data on which the system is to be applied in the field all come from the same distribution. In this scenario, it makes sense to gather training data until the cost of acquiring it outweighs the additional benefit derived from it. For many classifiers, the more training data we gather, the closer we can get to the Bayes error rate in our real world application. When this data is inexpensive to gather, the task may warrant collecting large training sets. These are the conditions under which Criterion 2 should be used.

However, the goal of building a classifier based on a standard data set is not always to discriminate among the points in a test distribution. Often the goal is to compare one's own classifier to other classifiers that have been developed. That is, often the goal is to discriminate between the performance of different *classification schemes*. This is especially true if we plan to use the classifier we are studying in a new, but related scenario in which we may not have as much training data.

Suppose that we are comparing two classifiers, each of which is known to converge to the Bayes rate as the number of training samples goes to infinity. We know *a priori* that the classifiers cannot be distinguished if the training data sets are very large. Thus, if our goal is to compare classifiers, we should choose a training set size that maximizes the performance difference rather than training set sizes that maximize performance of each classifier, which may in fact *minimize* the performance difference. Many classifier comparisons now take place with the aforementioned large data sets, and classifiers can only distinguish themselves by reducing error by a tenth of a percent or less. This slim margin makes it difficult to understand performance differences between various techniques.

We stress that building classifiers with large training sets has important applications,

among which is the deployment of working handwriting recognition systems. However, for distinguishing between general approaches to building classifiers, there may be more information at the other end of the spectrum of training set sizes. Recently, some researchers (Simard et al., 1998) have begun looking at smaller training sets to better discriminate among classification methods. We believe this trend should continue.

2.2.3 Criterion 3: Robustness to assumptions and adaptability

Our ultimate aim in this work is to investigate classification and other learning problems based upon small training sets. However, before addressing the criterion of performance on small training sets, we briefly mention one other set of issues in the evaluation of classifiers: the sensitivity to assumptions about the training and test sets. Traditional cost functions for supervised learning typically do not include such a sensitivity analysis.

"Non-stationarity" of distributions

Suppose that the distribution of each class that a classifier was trained on is perturbed slightly after classifier deployment. Alternatively, suppose that data is drawn from a non-stationary distribution, i.e. one which is changing as a function of time. For example, this can arise in handwriting recognition systems when data is collected from one user, but the system is used by another user. One type of model (e.g. generative models) may be easier to adapt quickly than another type of model (e.g. discriminative models). While we will not delve into this issue in this thesis, it is an important issue to consider in evaluating classifiers.

Changing the set of classes to be distinguished

Another simple example of adaptability is the addition or subtraction of new classes to a current classifier, which amounts to a change in the task to be performed. For example, a check reader in the United States might be required to read the "$" symbol, whereas in Great Britain, the "£" symbol would need to be recognized. We may wish to evaluate how easily the classifier is adapted from one environment to the next. A fine-tuned discriminative classifier that has only modeled the border between classes may exhibit terrible performance when a class is added or removed from the set of possible classes. On the other hand, a minimum expected error ML classifier based on perfect probability density estimates remains a minimum expected error ML classifier when any subset of classes are removed from the model. Hence, such a classifier exhibits a particular type of robustness not seen in all classifiers. This suggests yet another axis along which classifiers can be compared.

2.2.4 Criterion 4: Best performance on small training sets

The particulars of the task at hand may bias us heavily in our evaluation of classifiers. From an artificial intelligence viewpoint, one can argue that decisions based upon only a single example are often critical. Certainly, a classification scheme whose performance evaluation depends upon N training examples may be irrelevant if that many training examples are never available.

Being eaten

Consider the following scenario. Suppose squirrel A knows nothing about cats until it witnesses a cat eating squirrel B. It would behoove squirrel A to recognize the class of cats from this single example, rather than waiting to see 1000 examples of such an event. The nature of survival suggests that learning from one example is often very important.

In addition, whenever learning is a competitive endeavor, the efficiency of learning from a small number of examples is important. Learning to locate food or shelter from a single stimulus may result in an agent winning out over another less efficient learner. Recent results (Ng and Jordan, 2002) show interesting cases in which classifiers with inferior asymptotic performance may nevertheless give lower error rates if the training sets are small.

2.3 Conclusions

We therefore suggest that learning from a single example is more relevant in many scenarios than asymptotic learning rates or learning from large data sets. However, we can infer very little from a single example[4] without prior knowledge. In Chapter 3, we describe a factorized image distribution that will enable us to share knowledge from previous tasks. This shared knowledge allows us to build the one training example classifier described in Chapter 4.

[4]Bounds on the probability of error in supervised learning scenarios such as the Vapnik-Chervonenkis (VC) bounds have been developed that are a function of the training set size. However, these bounds are usually uninformative for small training set sizes. They tend to be of the form $Prob(error) < \gamma$, where $\gamma > 1$, so they provide no new information. See (Vapnik, 1998) for detailed discussion of such bounds.

Chapter 3

Factored Density Estimation for Robust Classification

3.1 Introduction

This chapter describes a variety of handwritten digit classification algorithms.[1] In particular, it describes classifiers built from factorized probability models for images. Generative factorized probability models make the assumption that an observation was generated by at least two statistically independent processes. Since such an assumption represents an approximation to the true generative process, factorized probability densities cannot be perfectly accurate even with infinite data for estimation. Nevertheless, if the independence assumption holds approximately, then it may be possible to estimate a "good" factorized probability model much more rapidly (based upon a smaller sample) than a general non-factorized model. This can result in better classifiers when the amount of training data is limited.

The chapter begins with a description of a generative image model for handwritten characters. It models each observed character as having two unseen parts (factors): a character in a canonical form (the *latent image*), and a deformation away from this canonical form. One can think of the deformation as a type of spatial "noise" that has been added to an otherwise "centered" character. In the next section (3.3), we describe how classifiers can be built either directly from observed training images, or from estimates of these parts of the training images. We discuss the possible benefits of building classifiers from these constituent parts.

In Section 3.4, we then describe a new algorithm for generating these image factors from a set of training images for each class. This *congealing* algorithm simultaneously aligns a set of images of a class, producing a set of latent images and the transforms that would produce the observed images from these latent images.

With these "image parts" in hand, Section 3.5 describes probability density estimators for latent images and transforms. Then in Section 3.6, we report experimental results for the three classification algorithms described in Section 3.3 using the density estimates of Section 3.5. We confirm that a classifier based on a factorized model of the images outperforms a classifier that models the observed images directly.

Finally, in Section 3.7, we modify the classifiers to use an implicit rather than explicit

[1] Much of the content of Chapters 3, 4, and 5 appeared in (Miller et al., 2000).

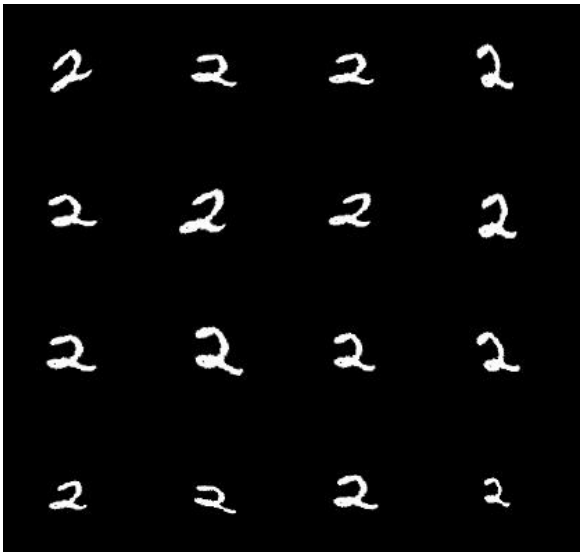


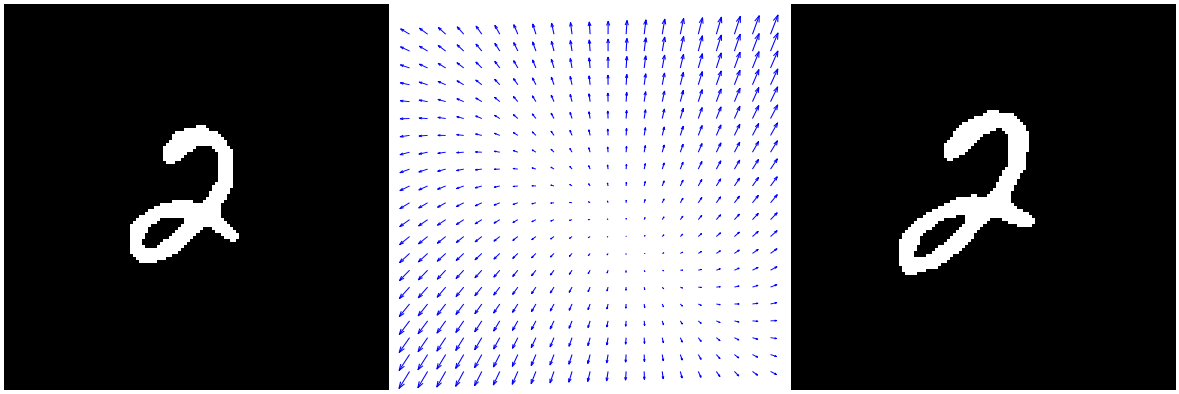
Figure 3-1: Images of "2"s that are the same up to an affine transformation. One of these is arbitrarily picked to represent the "latent image".

probability density over images. We adopt the Hausdorff measure as a distance between images, and use a nearest neighbor scheme to define a new set of classifiers. These classifiers substantially improve upon the results from Section 3.6, but still show the same relative performance. That is, the classifier based on the factorized model still outperforms the classifier based directly on the observed images.

3.2 A generative image model

We start with a few simple observations about handwritten characters. Suppose one is presented with the image of a single handwritten character such as a "2". A small rotation of that image is still likely to be considered a "2". A small magnification, shift, or shear of the image (i.e. any *affine* transformation) is also likely to be identified as a "2".[2] Consider the set of "2"s shown in Figure 3-1. These images are equivalent up to an affine transformation. If we choose one such image arbitrarily and call it the *latent image* associated with this set,

[2]Many researchers in machine vision and machine learning interpret this quality to be an *invariant* of handwritten characters. That is, it is said that handwritten characters are invariant to some set of affine transformations. There are at least two problems with this terminology. First, not every character can undergo the same transformations without changing identity. For example, an upright "6" can easily undergo 50 degrees of rotation in either direction and still be identified as a "6". However, a "6" that is already oriented at 45 degrees cannot undergo such major transformations without starting to be confused with a "9". Thus it is not clear with respect to which set of transformations invariance should be defined. Second, and perhaps more importantly, even a small transformation of a character that does not necessarily change the outcome of a simple classifier may alter the *likelihood* of each class label for that character, under a particular generative model (like the one presented in this chapter). This can have critical implications when contextual effects are taken into account. Thus, we believe the term *invariant* is misleading in that it suggests that there is essentially no effect whatsoever to affine transforming a character, which is not the case.



a **b** **c**

Figure 3-2: An illustration of a typical latent image (**a**), an affine transformation represented as a vector field (**b**), and the resulting observed image (**c**).

then every other image in the set can be created by transforming the latent image according to some affine transformation.

These observations suggest that we can build a compact model of the set of all handwritten "2"s by defining a set of different latent images and then describing each "2" as a pair consisting of a latent image and a transform away from that latent image. We adopt a generative image model similar to those used in papers on deformable templates as in (Amit et al., 1991). As described above, the basic idea is that the image be understood as *texture* and *shape* as described in (Vetter et al., 1997), or as a *latent image* that has undergone a *transformation*, as presented in (Frey and Jojic, 1999a). We adopt the latter terminology in this work.

Figure 3-2 shows a typical latent image (**a**), a transformation represented as a vector field (**b**), and the observed image (**c**) resulting from the action of the transformation on the latent image. Other researchers have recently had successes in using similar models to handle spatial variability (Frey and Jojic, 1999a; Vetter et al., 1997). While such factorizations are interesting and useful in their own right, the primary motivation for using them in this thesis is that they will enable the development of classifiers from a single example. In particular, they will provide a vehicle for us to share information learned from previous tasks.

Figure 3-3 represents the image model as a directed graph.[3] Let $\mathcal{I}_L$ be the set of possible latent images and $\mathcal{T}$ the set of possible transforms. The graph shows that we expect the latent image $I_L \in \mathcal{I}_L$ and transform $T \in \mathcal{T}$ to be dependent upon the character class. In addition the graph indicates our assumption that the transform and latent image are conditionally independent given the class. That is,

$$p(I_L, T|C) = p(I_L|C)p(T|C). \tag{3.1}$$

An additional assumption is made that the observed image I is a deterministic function of the transform T and the latent image I_L. Since the transform can be thought of as acting

[3]For a discussion of directed graphical models, or *Bayesian networks*, see the introduction in (Jensen, 1996).

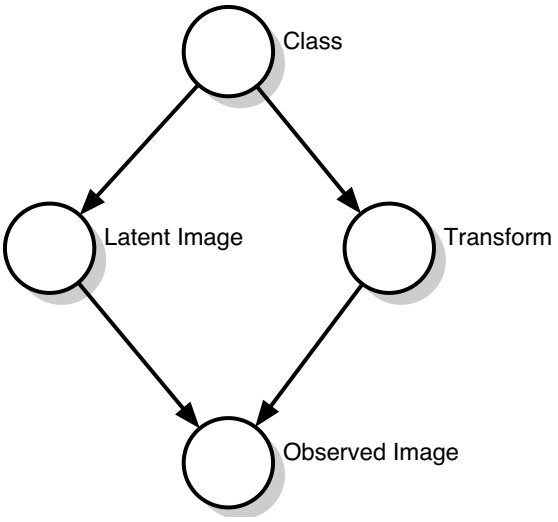


Figure 3-3: A directed graphical model representing a general generative image model. The latent image and transform are both dependent on class, and they are conditionally independent given the class.

on the latent image, we write

$$I = T(I_L). \tag{3.2}$$

We shall assume that the transformation is a one-to-one mapping between images and is hence invertible, so we may also write

$$I_L = T^{-1}(I). \tag{3.3}$$

Given an observed image I, however, we cannot uniquely determine a corresponding latent image-transform pair. This follows since for each transform $T \in \mathcal{T}$, there exists a latent image-transform pair $(T^{-1}(I), T)$ that explains the observed image. Below, we shall try to find the transform that converts an observed image back into the most likely latent image, where the likelihood is defined relative to other images in the same class. Hence, from an observed image, we will try to estimate T^{-1}, which for convenience we will denote U. In the most general setting, $T(\cdot)$ could be any smooth and invertible map from the coordinates of one image (values in $\mathcal{R}^2$) to the coordinates in the new image. For our initial development, however, we will assume that the transforms are affine.[4] Extensions to other transform families are discussed in Chapter 5.

3.3 Classification

We now consider three classifiers based upon the generative model from the previous section. The first is based upon the observed image density, the second upon the latent image density, and the third upon both the latent image and transform densities. Let $\mathcal{C}$ be the set of class

[4]To handle boundary effects (parts of the image being transformed beyond the boundaries of the image) we assume that the constant 0-valued background of the image extends infinitely in all directions, so that transformations of backgrounds pixels onto and off of the image have no effect. The transformation of foreground pixels off of the image is explicitly disallowed in the program implementation.

labels and let K be the size of this set. It will be assumed for simplicity that the prior probability distribution over class labels is uniform.

3.3.1 Maximum likelihood classification based on the factored image model

Suppose we have a set of training images for each class. Let the set of N training images for a single class be denoted I^j, $1 \leq j \leq N$. Assume for the moment that we are able to convert this set of observed training images for a class into a set of estimated latent image-transform pairs $(\hat{I}_L^j, \hat{T}^j)$. (The method for doing this is discussed in the next section.) This section addresses the question of how such a decomposition could be used to improve over a handwritten digit classifier that did not use such a decomposition.

Classifier 1: The observed image classifier

In the first classifier, a probability density $p_{\mathbf{I}|C}(I|c)$ is estimated directly from the training examples I^j for a class. A maximum likelihood classifier for a test image I is then defined by:

$$c^*(I) = \arg\max_{c \in \mathcal{C}} p_{\mathbf{I}|C}(I|c). \tag{3.4}$$

This is the simplest classifier as it makes no assumptions about independent processes in the generative process. It shall be referred to as the *observed image classifier*.

Classifier 2: The latent image classifier

The second type of classifier we wish to consider is one that maximizes the probability of the latent image. Hence, a probability density $p_{\mathbf{I}_L|C}(I_L|c)$ for latent images needs to be developed for each class. It is built from the set of estimated latent images $\hat{I}_L^j$ from the training data for each class. Again, we defer until the next section the discussion of how these latent image estimates are obtained.

To evaluate a test image I using this classifier, we also need to estimate a latent image for the test image. We assume again that we have a method for estimating a latent image from the test image, conditioned on the class label. Note that in this case, however, the class label for a test image is unknown. Because the latent image distribution is dependent upon the class, there will be a different latent image estimate for each possible class label. For example, if an observed test image looked like this: "9", then the latent image estimate under the nine model would be very similar to the observed image ("9"), whereas the latent image estimate under the six model would be the observed image rotated 180 degrees ("6"). Since there are K possible class labels for each test image, there will be K latent image estimates for every test image.

To estimate the "best" latent image $I_{L_{MAX}}$ for a test image I, under a particular class c, we maximize the probability of the latent image under $p_{\mathbf{I}_L|C}(\cdot|c)$, that is

$$I_{L_{MAX}}(I, c) = \arg\max_{T \in \mathcal{T}} p_{\mathbf{I}_L|C}\left(T^{-1}(I)|c\right). \tag{3.5}$$

Maximizing over the complete set $\mathcal{T}$ of transforms considers all possible latent images for a given observed image I.

The classifier then maximizes the likelihood of the class for each latent image:

$$c^*(I) = \arg\max_{c \in \mathcal{C}} p_{\mathbf{I}_L|C}\left(I_{L_{MAX}}(I, c)|c\right). \tag{3.6}$$

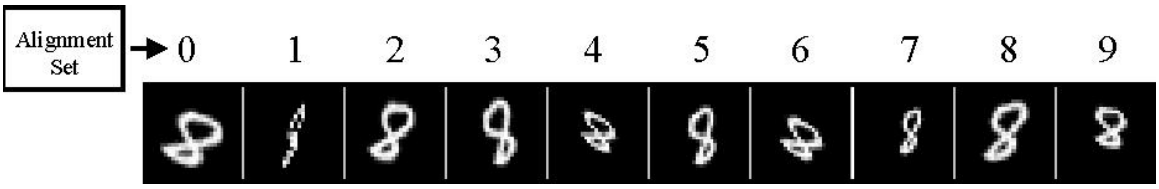


Figure 3-4: A set of latent image estimates for an observed "8" image. The eight is aligned with each model in turn. Notice that alignment to the "1" model produces a latent image that is a plausible "1". Rejecting this model will depend upon penalizing the highly unlikely transformation that would be paired with such a latent image to produce the observed image.

This classifier is similar in spirit to many classifiers that "preprocess" data by trying to remove variability before classifying. However, it differs in that it removes the variability in a different manner for each class. We shall refer to this classifier as the *latent image classifier*.

Classifier 3: The LT classifier

The third classifier is based upon the conditional probability of the latent image estimate for a test image *and* the conditional probability of the paired transform estimate:

$$c^*(I) = \arg\max_{c \in \mathcal{C}} p_{\mathbf{I}_L|C}\left(I_{L_{MAX}}(I,c)|c\right) p_{\mathbf{T}|C}\left(T_{MAX}(I,c)|c\right). \tag{3.7}$$

$I_{L_{MAX}}$ is computed as for the latent image classifier. However, we define T_{MAX} as the transform such that $T_{MAX}(I_{L_{MAX}}) = I$. In particular, T_{MAX} is *not* the most likely transform in $\mathcal{T}$, which would often be approximately equal to the identity matrix.[5] We shall refer to this classifier as the *latent image-transform classifier*, or the *LT classifier*, for short.

The behavior of the LT classifier is illustrated in Figure 3-4. The figure shows ten examples of a test digit "8" as it is aligned to each of the ten handwritten digit models. Thus, these ten images represent the latent image estimates of the observed image, one for each class. The number above each image denotes the class to which the observed image was aligned. It is easy to see that under the "1" model, the image of the eight has been squashed in the $x-$direction to match the model, i.e. to try to take on the shape of the typical "1" latent image. While the latent image estimate for the "8" in this case is a good match to the latent image model for the "1"s, the probability of the (unlikely) transform should be low enough to cause this model to be rejected.

The latent image under the "2" model has the opposite problem. While the transformation of the original test image is a common transformation, the latent image in this case represents a poor match to the "2" model. In fact, the only class for which the test image has a latent image and transform that match the model well is the class of eights, so in this case, the classifier should work correctly.

[5]It is possible that $p(I_{L_{MAX}})p(T_{MAX})$ is not the highest probability latent-image transform pair for a particular class, since it is defined only in terms of the maximum probability latent image. However, since the latent image probability dominates this product, we assume that it is a good approximation of the maximum of the product.

3.3.2 Asymptotic behavior

We start our analysis of the three classifiers by considering their relative performance in the limit as the amount of training data tends to infinity. With infinite training data for each class and appropriate density estimates, the observed image classifier will converge to the Bayes error rate[6] for this problem under mild assumptions.

Note that the probability density of the observed image given the class can be rewritten as follows:

$$\begin{aligned} p_{\mathbf{I}}(I) &= \int_{T\in\mathcal{T}} p_{\mathbf{I},\mathbf{T}}(I,T)\, dT && (3.8)\\ &= \int_{T\in\mathcal{T}} p_{\mathbf{I}|\mathbf{T}}(I|T)p_{\mathbf{T}}(T)\, dT && (3.9)\\ &= \int_{T\in\mathcal{T}} p_{\mathbf{I}_L}(I_L(I,T))p_{\mathbf{T}}(T)\, dT, && (3.10) \end{aligned}$$

where we have omitted the conditioning on the class variable for clarity. The last step follows from our earlier assumption that the latent image I_L is a deterministic function of the observed image I and the transform T (Equation 3.3).[7] Hence, maximizing the likelihood of the observed image I is equivalent to maximizing the *integral* of the likelihood of latent image-transform pairs that could explain the observed image.

If we compare the quantity being maximized in (3.7) to (3.10), we see that the LT classifier is not optimizing the correct quantity with respect to the generative model.[8] It maximizes the likelihood of the class based upon a *single* latent image-transform pair, ignoring all other possible combinations corresponding to the observed image. Even if we were given the true distributions over the latent image and transforms for each class instead of having to estimate them, in general we cannot achieve the Bayes rate using this method since it evaluates only part of the total probability of the observed image. The latent image classifier is even worse, since it has the same problem as the LT classifier, and also ignores the probability of the transform. Hence, in the limit of infinite training data, the observed image classifier will, on average, perform better than or equally to the other two.

We note, however, that the difference in the asymptotic classification performance between the observed image classifier and the LT classifier may be small in many cases. The class with maximum likelihood for a single latent image-transform pair is often the same as the class with the greatest integrated likelihood for *all* such pairs (the observed image classifier), since for distributions with a "peaky" shape, most of the mass from the integral in Equation 3.10 comes from a small neighborhood around the pair $(I_{L_{MAX}}, T_{MAX})$. If the class likelihoods are all similarly shaped, then $Prob(I_{L_{MAX}}, T_{MAX})$ multiplied by some fixed constant will make a good surrogate for the integrated likelihood of Equation 3.10.

In fact, from here forward, it is assumed that with high probability

$$\arg\max_{c\in\mathcal{C}} p_{\mathbf{I}|C}(I|c) = \arg\max_{c\in\mathcal{C}} p_{\mathbf{I}_L|C}\left(I_{L_{MAX}}(I,c)|c\right) p_{\mathbf{T}|C}\left(T_{MAX}(I,c)|c\right). \qquad (3.11)$$

[6] The Bayes error rate is the minimum possible error rate for a classifier given a set of distributions from which test samples are drawn. See (Duda and Hart, 1973) for a discussion.

[7] We also assume that $p_{\mathbf{I}|\mathbf{T}}(\cdot|\cdot)$ and $p_{\mathbf{I}_L}(\cdot)$ are densities with respect to the same measure.

[8] The exception to this is the degenerate case in which we can uniquely recover the latent image-transform pair from the observed image.

For the reason described above, this will be referred to as the *peakiness assumption*. Sometimes this assumption is justified; other times it is not. However, for the moment, we suggest that other factors in choosing a classifier are more important than the often small differences between the asymptotic performance of the observed image classifier and the LT classifier.

3.3.3 Efficiency

Since we do not have infinite training data, we may want to examine aspects of classifier performance other than asymptotic behavior. In particular, each of the classifiers above depends upon some density estimation process in the training phase. The quality of these density estimates for finite data directly affects the error rates of each classifier.

Under the assumption of 3.11, our new goal is to analyze which probability estimate converges to a good approximation of the true distribution faster, $p_{\mathbf{I}}(\cdot)$ or the product distribution $p_{\mathbf{I}_L}(\cdot)p_{\mathbf{T}}(\cdot)$. Our claim is that estimates of the second distribution converge much more rapidly, albeit to an approximation of the true distribution, than estimates of the first distribution do. This appears to be true based upon experiments (described below). While we cannot prove that this should be the case, we offer a series of intuitions why this may hold.

Consider estimating a joint distribution over two independent variables X and Y given a number of pairs (x^j, y^j) drawn *iid* from their joint density.. Two scenarios are addressed. In the first, it is known that the variables are independent. In the second, it is not known whether the variables are independent, and it must be assumed that we have a general non-factorizable joint distribution $p_{X,Y}(x, y)$.

The case where independence is known in advance has a great advantage in performing density estimates. This is because we can use all of the data pairs to estimate the two marginals $p_X(\cdot)$ and $p_Y(\cdot)$. In the case where independence is not known, it is not safe to assume that $p_{X|Y=y_1}$ equals $p_{X|Y=y_2}$, so each conditional distribution should be estimated separately. If X and Y are discrete random variables that take M possible values, then in the independent case, we have to estimate only $2M$ values, whereas in the agnostic case, we must estimate M^2 separate values. Of course, the convergence rates depend upon the specifics of the distribution and the particular estimator used, so this discussion is merely meant to provide intuition.

From the above discussion we may predict that the knowledge of approximate independence of $p_{\mathbf{I}_L}(\cdot)$ and $p_{\mathbf{T}}(\cdot)$ will help us converge quickly to an estimate of their joint distribution. But how does this help us in comparing the estimation of this joint, $p_{\mathbf{I}_L}(\cdot)p_{\mathbf{T}}(\cdot) = p_{\mathbf{I}_L,\mathbf{T}}(\cdot,\cdot)$, with the estimation of $p_{\mathbf{I}}(\cdot)$?

From the equivalence of 3.8 and 3.10, we see that $p_{\mathbf{I}}(\cdot)$ can also be viewed as a sort of marginal, where the integral is over all latent image-transform pairs that generate the same observed image. Since it too is a marginal, we may be tempted to assume that it can be approximated as rapidly as the latent image distribution, $p_{\mathbf{I}_L}(\cdot)$, especially since the sample spaces of these two distributions have the same dimensionality. However, *the benefit of marginalization is often due to the reduction of the number of states with significant probability mass*. That is, in the product distribution $p_X(\cdot)p_Y(\cdot)$ described above, the benefit of marginalization stems from reducing the number of states with significant probability mass from $O(M^2)$ to $O(M)$. When interpreting $p_{\mathbf{I}}(\cdot)$ as a marginal, however, our peakiness assumption from Section 3.3.2 suggests that there are not a great number of latent image-transform pairs that are condensed into a particular observed image. We claim that the

number of "plausible" (non-negligible probability) observed images is much higher than the number of plausible latent images, since, loosely speaking, most latent image-transform pairs with high probability generate a unique observed image. If the cardinality of the transform set is N_T and the cardinality of the latent image set is N_{I_L}, then our claim is that the cardinality of the observed image distribution is on the order of $N_T \times N_{I_L}$. This larger size for the observed image set makes it more difficult to estimate (without factoring) than the latent image or the transform distribution.

This analysis is clearly not rigorous, but hopefully provides some insight as to the reasons that the latent image-transform distribution can be estimated more efficiently than the observed image distribution. If this is true, then the LT classifier would be expected to have an advantage in the non-asymptotic range, which is supported by empirical findings described below.

3.3.4 Summary

To summarize the discussion of the three classifiers, the observed image classifier and the LT classifier are assumed to have comparable asymptotic performance by the peakiness assumption, but the latent image classifier is expected to perform less well asymptotically since it does not use information about the transform density. However, the LT classifier may outperform the observed image classifier for data sets in the non-asymptotic range, since the factorization of the distribution gives us an advantage in the estimation process, as long as reasonably good estimates of the latent image-transform pairs can be obtained for both the training and test images.

3.4 Congealing

To produce a factorized model as described above from the training data for each digit class, we must somehow estimate the latent image and transform associated with each training image.[9] We do this by developing a procedure to simultaneously align a set of images from the same class using affine transformations, as illustrated in Figure 3-5.[10]

Vetter et al. coined the term *vectorization* for the process of aligning (bringing into correspondence) an image with a reference image or a model. This can be thought of as trying to separate an observed image I (as in Figure 3-2**c**) into its constituent latent image I_L (Figure 3-2**a**) and transform T (Figure 3-2**b**). We introduce the term *congealing* which we define as the simultaneous alignment of a set of images to each other. Thus, congealing can be seen as the simultaneous vectorization of a set of images from a class. This is similar in spirit to Vetter et al.'s bootstrap vectorization procedure (Vetter et al., 1997) and Jojic and Frey's Transformed Mixtures of Gaussians (Frey and Jojic, 1999a), although it differs in several details, discussed in Chapter 5.

Figure 3-5 shows a set of handwritten zeroes and a set of handwritten twos, both before and after congealing. Roughly speaking, given a set of training images for a particular class, the process of congealing transforms each image through an iterative process so as

[9] Since there are multiple latent image-transform pairs that can produce a given observed image, this is an ill-posed problem without further constraints.

[10] The congealing algorithm was partly inspired by previous information theoretic methods of image alignment (Viola and Wells III., 1997).

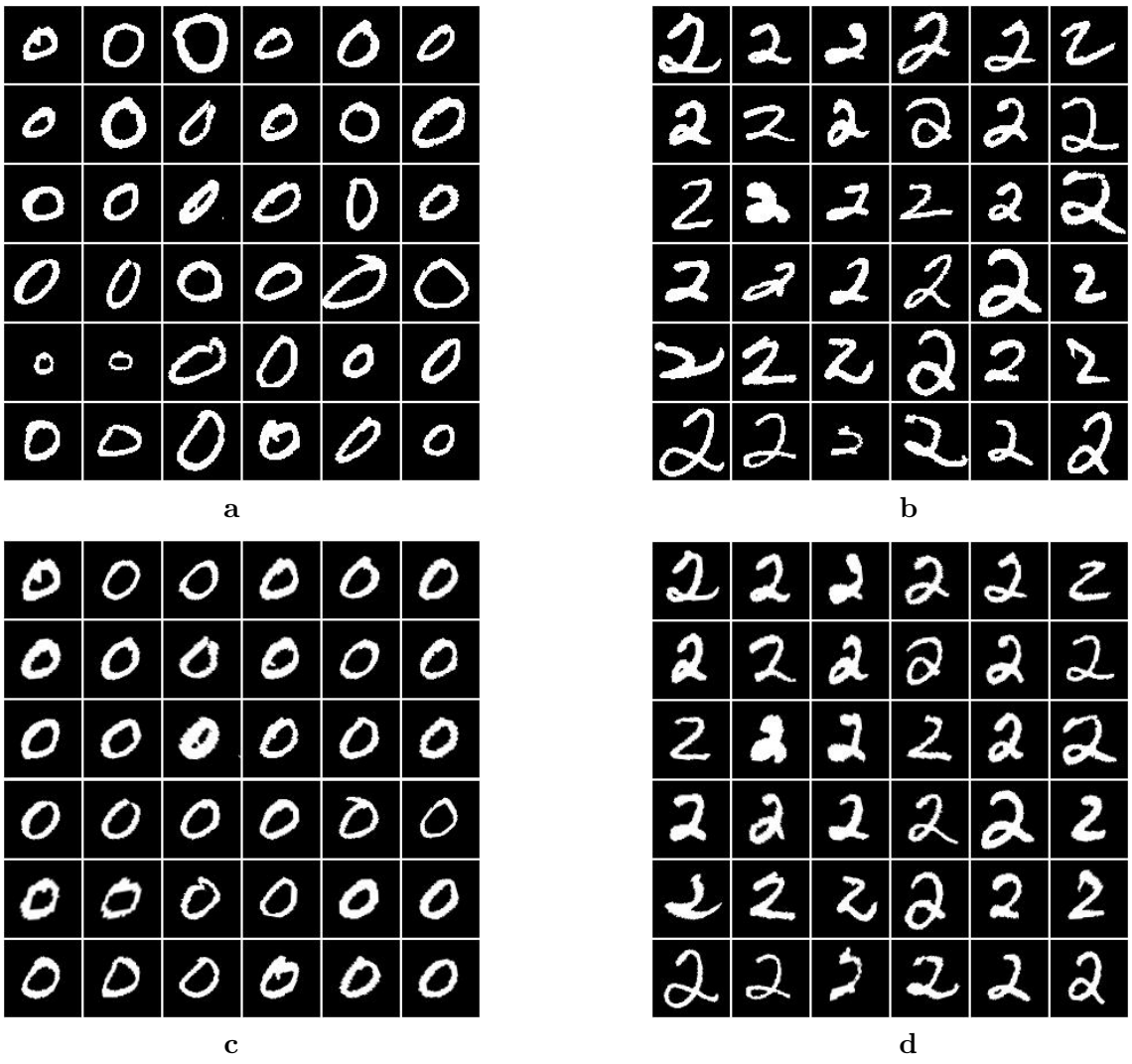


Figure 3-5: (a) Samples of zeroes from the NIST database. (b) Samples of twos from the NIST database. (c) The zeroes after congealing. (d) The twos after congealing.

to minimize the summed pixelwise entropies of the resulting images. The result is a set of estimated latent images and the transforms which produced them from the observed images.

3.4.1 Transform parameterization

Congealing is defined with respect to a set of transformations. In this work, we congeal using affine transformations. Before we can describe the algorithm in detail, we need to define our parameterization of affine transforms.

The affine transformation of a coordinate pair (x, y) may be represented as a linear transformation with an additional translational offset as in

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}, \tag{3.12}$$

or alternatively in homogeneous coordinates as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{3.13}$$

This second representation has the advantage of making the composition of two affine operations equivalent to a simple matrix multiplication transformations using the homogeneous representation. We further require of the component transformations. In this work we represent affine that the transformations be non-singular and non-reflecting (i.e. that they have positive determinants).[11]

We parameterize this set of transforms $\mathcal{T}$ by composing a transform from the following component transforms: x-translation, y-translation, rotation, x-log-scale, y-log-scale, x-shear, and y-shear. Thus, given a parameter vector $(t_x, t_y, \theta, s_x, s_y, h_x, h_y)$, a transformation matrix U is formed by multiplying the constituent matrices *in a fixed order* (needed to ensure a unique mapping between parameter vectors and the resulting matrices, since matrix multiplication is non-commutative):

$$\begin{aligned} U &= F(t_x, t_y, \theta, s_x, s_y, h_x, h_y) \\ &= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \\ &\quad \begin{bmatrix} e^{s_x} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & h_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3.14}$$

Note that this is an overcomplete parameterization, since there are seven parameters and only six degrees of freedom in the set of transforms. The reason for using an overcomplete parameterization concerns efficiency and will be addressed in Chapter 5. The current goal is to investigate how to make a set of images more similar to each other by independently transforming each one of them in an affine manner. We now describe an objective function that allows us to achieve this goal.

3.4.2 Entropy estimation

Consider a set of N *observed* binary images of a particular class, each image having P pixels. Let the value of the ith pixel in the jth image be denoted x_i^j. Let the image created by the transformation of the jth image by transform U^j be denoted $I^{j'}$. We assume for the moment that this new image is still binary. Let the ith pixel in this transformed image be denoted $x_i^{j'}$. Consider the set of N pixels at a particular image location after each image

[11] This set of transformations arises naturally in vision problems from the orthogonal projections of one flat surface onto another. Reflections (negativity of the matrix determinant) cannot occur unless we allow projection "through" a surface, as in a projection through a transparent object like a window. However, the vast majority of transformations due to viewing angle do not include reflections. Our congealing algorithm is parameterized so that it cannot produce a reflecting transformation. Although perspective transformations are a better approximation to the set of transformations actually seen by human beings and video cameras, affine transformations make a good approximation of common distortions for flat objects viewed from a distance and have better computational properties.

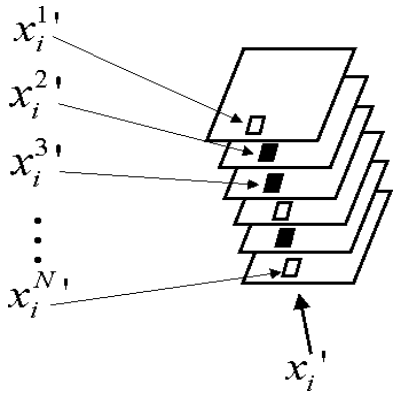


Figure 3-6: A *pixel stack* is a collection of pixels drawn from the same location in each of a set of N images. Here, the ith pixel from each of six images forms a pixel stack. Since half of the pixels are black and half are white, this corresponds to a Bernoulli random variable with parameter $p = 0.5$. The entropy of such a random variable is $-(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$ bit.

has undergone some transformation: $\{x_i^{1\prime}, x_i^{2\prime}, x_i^{3\prime}, ..., x_i^{N\prime}\}$. We call this set of N pixels across all of the images a *pixel stack*.[12] We denote the ith pixel stack of the original image set x_i and the ith pixel stack of a transformed image set x_i'. A pixel stack is illustrated in Figure 3-6.

This pixel stack can be viewed as a sample from a random variable or pixel process. We can estimate the entropy, or disorder, of this pixel process by calculating the entropy of the empirical distribution of values in the pixel stack. This is also referred to as the *empirical entropy* (see (Cover and Thomas, 1991), page 195) of the set of values in the pixel stack. For brevity, we shall refer to this quantity as simply the entropy of the pixel stack:[13]

$$H(x_i) = -\left(\frac{N_0}{N}\log_2\frac{N_0}{N} + \frac{N_1}{N}\log_2\frac{N_1}{N}\right), \tag{3.15}$$

where N_0 and N_1 are the number of occurrences of 0 (black) and 1 (white) in the binary-valued pixel stack.[14] The entropy of the pixel stack shown in Figure 3-6 is $H(x_i') = 1$ bit, since there are equal numbers of black and white pixels in the stack. We also refer to pixel stack entropies as pixelwise entropies.

3.4.3 Congealing for model formation

Congealing simply minimizes the quantity

$$E = \sum_{i=1}^{P} H(x_i') + \sum_{j=1}^{N} |v^j|. \tag{3.16}$$

[12]This term was coined by Kinh Tieu in the MIT Artificial Intelligence Laboratory.

[13]In this formula, $0 \log_2 0$ is interpreted to have value 0.

[14]A pixel whose value is between 0 and 1 is interpreted as a mixture of underlying 0 and 1 "subpixels". To extend Equation 3.15 to handle these pixels, we merely increment each count by the fraction of background and foreground in the mixture. For example, for a 50% gray value pixel, we would increment both N_0 and N_1 by 0.5.

by transforming the training images for a class. The v^j's are vectors of transformation parameters for each image and $|\cdot|$ is some norm on these vectors. This norm keeps the images from shrinking to size zero or undergoing other extreme transformations.[15] We shall refer to this quantity as the penalized pixelwise entropy.

There are two major applications of congealing in this work. The first is estimating a set of latent-image transform pairs for a training set of a known class. We refer to this process as *congealing for model formation*. A second somewhat different application is in determining the best latent image-transform pair for a test image of unknown class relative to a class model. We refer to this as *test congealing*.

The basic congealing for model formation algorithm proceeds as follows. It takes as input a set of N binary images of the same class. It makes small adjustments in the affine transforms applied to each image to reduce the penalized pixelwise entropy defined in Equation 3.16. More formally we have:

Algorithm 1: congealing for model formation

1. Maintain a transform parameter vector $v^j = (t_x, t_y, \theta, s_x, s_y, h_x, h_y), 1 \leq j \leq N$ for each image. Each parameter vector will specify a transformation matrix $\hat{U}^j = F(v^j)$ according to Equation 3.14. Initialize all v^j to zero vectors. This has the effect of initializing all of the transformation matrices $\hat{U}^j$ to the identity matrix.

2. Compute the penalized pixelwise entropy E for the current set of images from Equation 3.16.

3. Repeat until convergence:

   (a) For each image I^j,

      i. For each affine parameter (x-translation, y-translation, rotation, x-log-scale, y-log-scale, x-shear, and y-shear)

         A. Increment the current affine parameter for the current image by some small amount ϵ which may depend upon the parameter. This creates a new transform parameter vector v^j_{new} for the current image.

         B. Let $\hat{U}^j_{new} = F(v^j_{new})$.

         C. Let $I^j_{new} = \hat{U}^j_{new}(I^j)$.

         D. Recompute the penalized pixelwise entropy E of the entire image set according to Equation 3.16.

         E. If E has been reduced, accept the new parameter vector $v^j \leftarrow v^j_{new}$ and return to step 3.a.i. Otherwise:

         F. Decrement the current affine parameter for the current image by ϵ, recompute $\hat{U}^j_{new}$, I^j_{new}, and E. If E has been reduced, accept the new parameter vector: $v^j \leftarrow v^j_{new}$. Otherwise revert to the original parameter vector for that image v^j.

[15] Although the congealing algorithm itself is simple, its analysis is complicated by the fact that images must be spatially discretized to be represented in a digital computer. This discretization allows the affine transformations, which continuously deform an image rather than merely permuting the pixels, to reduce the summed pixelwise entropies without necessarily aligning the images, for example by shrinking all of the images simultaneously. This detail, however, will not prevent us from understanding the basic congealing algorithm while temporarily ignoring this issue. We shall fully address it at the beginning of Chapter 5.

a **b** **c** **d**

Figure 3-7: Mean images during the congealing process. **a.** The initial mean image for the set of zeroes. The blurriness is due to the misalignment of the images. **b.** The final mean image for the set of zeroes. The coherence of the aligned images is indicated by the increased sharpness of the image. **c.** The initial mean image for the set of twos. **d.** The final mean image for the set of twos.

(b) After each cycle through the image set, compute the mean parameter vector $\bar{v} = \frac{1}{N}\sum_{j=1}^{N} v^j$, and subtract it from each parameter vector: $v^j \leftarrow v^j - \bar{v}$.

(c) After each cycle through the image set, store the current mean image, $M_k = \frac{1}{N}\sum_{j=1}^{N} I^{j\prime}$, for later use with test character congealing.

4. At convergence, the algorithm terminates.

Congealing does coordinate descent of the entropy function. To visualize the entropy of the transformed image set for a class after each iteration of the algorithm, one can construct an image in which each pixel is the mean of its corresponding pixel stack. Such images are shown in Figure 3-7. Images **a** and **c** show the mean "0" and mean "2" image at the beginning of the algorithm. The relative abundance of intermediate gray values indicates that many pixel stacks have high entropy, since a middle gray value represents a pixel stack composed of half white and half black pixels, with maximum possible entropy. Images **b** and **d** show the pixel stack mean images at the end of the algorithm. Here, we can see that the pixels have distributions that are skewed much more heavily to pure black or pure white, and hence are of lower entropy. Notice that there is greater entropy represented in the final mean "2" image than in the "0" image due to the fact that zeroes can be better aligned through affine transforms than twos can. Actual numerical values of the entropies will be discussed in Chapter 5 in the context of other alignment algorithms.

After convergence to a minimum value of the summed pixel entropies, our hope is that each of the $\hat{U}^j$ is the transform such that

$$I^{j\prime} = \hat{U}^j(I^j) \approx I_L^j. \tag{3.17}$$

That is, $\hat{U}^j$ is our best guess of the transformation that maps the observed image into the corresponding latent image. Hence, we can identify the final transforms $\hat{U}^j$ from congealing as approximate samples of T^{-1} from Equation 3.3. Alternatively, we can recover estimates of the original T^j's, since

$$T^j = U^{j^{-1}}. \tag{3.18}$$

These estimated samples of T, $\hat{T}^j = (\hat{U}^j)^{-1}$, will be used below to form a density on

transforms.

Interpretation of pixel values for entropy computations

Our functions of pixel values, such as the entropy function discussed below, will depend upon our interpretation of these values. We shall start by assuming that images are binary valued functions (either 0 or 1) over some region of R^2, and that each pixel represents the mean value of the function over the region defined by the pixel. If two pixels are combined during a transformation (for example, by a translation of half of a pixel,[16]) then some pixels may be given values between 0 and 1. These are to be interpreted as pixels that are a (convex) mixture of foreground and background, rather than as a single scalar value. This has implications for our entropy computations below. In particular, we shall still treat the images as the result of an underlying binary process rather than as a process that generates scalars between 0 and 1.

3.4.4 Test congealing

To make a classifier based upon latent images, we also need a method for estimating a latent image from a test character. In Section 3.3, the optimal latent image for a test image was defined as the latent image with maximum probability over all inverse transformations of the observed image (Equation 3.5). But the practical question of how to find this maximum remains.

There are many possible ways to find the $I_{L_{MAX}}$ of Equation 3.5. When dealing with only a finite number of transforms as in (Frey and Jojic, 1999a), this becomes just an iterative computation, linear in the number of allowed transformations. An enumeration of all possible transforms is not possible when they are continuously parameterized, as in this work.

One option is to perform gradient ascent on the probability of the latent image in the space of transforms, effectively maximizing the probability of the latent image with respect to the latent image density defined by the congealed training images. The problem with doing this directly is that the probability function may have local minima or areas of zero gradient that cannot be circumvented easily, especially if the congealing process produced a low entropy model, as represented by Figures 3-7(b) and (d). That is, we may end up with local minima or zero-gradient problems, making traditional gradient ascent difficult. This is exactly the problem that occurs in trying to align a pair of images via gradient descent of an objective function, a ubiquitous difficulty in machine vision.

However, elements of the training ensemble rarely get stuck in local minima during the congealing process for the training set. (Statistics on the number of images reaching local minima are given in Chapter 5.) This fact can be leveraged in the alignment of test images to models. In order to improve convergence of a test example, it can be *inserted into the original training ensemble*. Congealing can then be performed as before. This shall be referred to as *test congealing*. If the test sample was drawn from the same distribution as a particular training class (e.g. congealing a test "2" with the training "2"s), then the test sample should also successfully reach a local minimum which is, on average, as good as the local minimum achieved by the training data for that class. If the test sample is drawn from a different distribution than the current congealing ensemble (e.g. congealing a test

[16]This is implemented with simple bilinear interpolation.

"2" with the training "5"s), the test character may not reach a good local minimum of the probability function. But this is exactly what is desired, since the test "2" should not match the "5"s anyway.

Of course, recongealing all of the training data for each class, for each test character presented to the system, would represent a great deal of redundant computation. Two insights allow for a significant reduction in computation. First, if the number of training examples in a class is large, then the addition of one image to an ensemble of characters for congealing should, with high probability, have a negligible impact on the convergence of the ensemble. Second, the intermediate state of the congealing training ensemble can be summarized sufficiently (for the purposes of entropy computations) by the mean image, which is stored after each iteration of Algorithm 1 during model formation. As a result the convergence behavior of a new example added to the ensemble is only a function of the sequence of mean training images (M_k), and the behavior of the entire ensemble need not be saved. Alignment of a new test example to a model proceeds by reducing the change in entropy of the test example with respect to a sequence of probabilistic models (represented as mean images), which are saved during step 3.c of Algorithm 1.

The following algorithm congeals a test image I to a particular set of training images for a class. The class of the training images does not necessarily match the class of the test image I.

Algorithm 2: test congealing

1. Maintain a transform parameter vector $v = (t_x, t_y, \theta, s_x, s_y, h_x, h_y)$ for the test image. As in Algorithm 1, the parameter vector will specify a transformation matrix $\hat{U} = F(v)$ according to Equation 3.14. Initialize v to the zero vector. This initializes $\hat{U}$ to the identity matrix.

2. Using mean image M_0 saved in step 3.c of Algorithm 1 as a sufficient statistic for the original training images, compute the penalized pixelwise entropy E for the training set from Equation 3.16.

3. Set $k = 0$. While $k < iter$, the number of iterations required to congeal the training set for this class:

   (a) For each affine parameter (x-translation, y-translation, rotation, x-log-scale, y-log-scale, x-shear, and y-shear)

      i. Increment the current affine parameter for the test image by some small amount ϵ which may depend upon the parameter. This creates a new transform parameter vector v_{new}.

      ii. Let $\hat{U}_{new} = F(v_{new})$.

      iii. Let $I_{new} = \hat{U}_{new}(I)$.

      iv. Using the mean image M_k as a sufficient representation of the partially congealed training set, recompute the penalized pixelwise entropy E of the entire image set, including the test image, according to Equation 3.16.

      v. If E has been reduced, accept the new parameter vector $v \leftarrow v_{new}$ and return to step 3.a.i. Otherwise:

      vi. Decrement the current affine parameter for the current image by ϵ, recompute $\hat{U}_{new}$, I_{new}, and E. If E has been reduced, accept the new parameter

vector: $v \leftarrow v_{new}$. Otherwise revert to the original parameter vector for that image v.

(b) Increment k.

(c) Load the next mean image M_k for use in the next iteration.

3.4.5 Summary

This section has described how congealing can be used to separate a set of images of a class into latent images and transforms, a process referred to as congealing for model formation. It has also described how, by saving intermediate results from congealing for model formation, a test image may be brought into correspondence with a model as though it were congealed with the original training images. This process is called test congealing. Next we describe the other half of the model formation process: estimating probability densities on images, using observed images or the latent image estimates resulting from congealing, and probability densities on the transforms resulting from congealing.

3.5 Density estimation for classification

In this section we describe the density estimators used for images (both latent images and observed images) and for transformations. Armed with these density estimates, we can implement each of the maximum likelihood classifiers described in Section 3.3.

3.5.1 Densities on images

For the first set of experiments performed, the probability of a test image with P pixels is evaluated simply as the product of the pixel probabilities in the image. That is,

$$p(I) = \prod_{i=1}^{P} p(x_i), \tag{3.19}$$

where x_i is the ith pixel in the image I.

The probability mass function for the ith pixel is treated as a Bernoulli random variable with the parameter a_i estimated from a set of N training images I^j, $1 \leq j \leq N$ as

$$a_i = \frac{N_0 + 1}{N_0 + N_1 + 2}, \tag{3.20}$$

where N_0 and N_1 represent the number of black and white pixels in the ith pixel stack. The addition of one in the numerator and two in the denominator smooths the probability mass function so that subsequent pixels (from test images) are never assigned zero probability. Thus, pixel i in a test image is assigned probability a_i if black and $(1 - a_i)$ if white. The probabilities for gray pixels are linearly interpolated between these two values. Densities for latent images are computed in an identical fashion except that the latent image estimates derived from congealing, rather than the original observed images, are used as training data.

This method for estimating probability densities on images makes the implicit assumption that the pixels are independent given the class, and it shall be referred to as the *independent pixel model* for images. It is a crude model, and hence results in relatively poor

density estimates. In Section 3.7, we replace this density estimate with an implicit one by adopting a nearest neighbor scheme using the Hausdorff distance.

3.5.2 Densities on transforms

It remains only to produce probability density estimates for affine transformations, given a set of transformations from congealing. (The reader interested only in implementation may wish to skip this subsection, referring to the formulas for generating densities (up-to-scale) for affine transformations. These formulas are given in Equation 3.49 and Equation 3.47.)

We would like to produce a probability density for an affine transformation T that is a function of its "closeness" to the training transforms T_i from congealing. In particular, if T is close, according to some measure, to many of the training transforms, then it should be assigned a relatively high probability density and if it is far from all of them it should be assigned a low probability density.

Suppose one treats the six components of an affine transformation matrix as a vector. Let $\mathbf{v}$ be the vector representation of the affine matrix T and let $\mathbf{v}_i$ be the vector representation of the N matrices T_i. A Gaussian kernel based probability density[17] can then be defined

$$p(\mathbf{v}) = \sum_{i=1}^{N} \frac{1}{(2\pi)^3} \exp(-\frac{1}{2}(\mathbf{v} - \mathbf{v}_i)^T(\mathbf{v} - \mathbf{v}_i)), \tag{3.21}$$

or with a non-spherical Gaussian kernel, as in

$$p(\mathbf{v}) = \sum_{i=1}^{N} \frac{1}{(2\pi|\Sigma|)^3} \exp(-\frac{1}{2}(\mathbf{v} - \mathbf{v}_i)^T\Sigma^{-1}(\mathbf{v} - \mathbf{v}_i)). \tag{3.22}$$

There is a serious problem with these estimators, however, that can be seen with an example. Suppose for a moment that

$$T = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

and that

$$T_1 = \begin{bmatrix} 7.07 & 7.07 & 0 \\ -7.07 & 7.07 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.24}$$

Here, T_1 and T are related by a rotation matrix of 45 degrees. That is, if they each represent the transformation of the same object, then the only "difference" between them is that T_1 represents an additional transformation by a rotation of 45 degrees.

We ask the question, what is the contribution to the probability of T due to T_1? In Equation 3.21, the contribution is a function of the summed squared differences in components, which in this case has a value of about 117.

[17] See (Duda and Hart, 1973) for a discussion of kernel estimators.

Now consider a similar scenario in which

$$T = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.25}$$

and

$$T_1 = \begin{bmatrix} 0.0707 & 0.0707 & 0 \\ -0.0707 & 0.0707 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.26}$$

In this case again, the matrices "differ" only by the multiplication of a 45 degree rotation matrix. Yet the contribution to the probability of T based on Equation 3.21 will be only 0.0117, a factor of 10,000 less than in the previous example.

The implication of this analysis is that matrices that represent shrinking transforms (the second example) will be considered to be closer together than matrices that represent magnifying transforms (the first example) even if they have the same transformational relationship. This will bias our estimators toward assigning higher probabilities to transforms with small components, since they will tend to be closer, in the vector difference sense, to other transforms. This analysis suggests searching for a density over transformations that does not have such a bias, or at least finding one with a smaller bias.

In a Lie group[18] such as the set of affine transforms, one frequently defines the "difference" between two elements A and B as

$$A^{-1}B, \tag{3.27}$$

where multiplication here represents the group operation. For affine matrices, this has the nice property that biases such as those described above are eliminated. In the two examples above, the product $T^{-1}T_i$ gives the same numerical result in both cases. It would be nice to define a probability density that is a function of this group difference operation, rather than the vector difference operation of Equations 3.21 and 3.22.

Another way to state this problem is that we would like to create a density function f with the training transforms acting as parameters $T_1, T_2, ..., T_N$, that has the property

$$f(T; T_1, T_2, ..., T_N) = f(AT; AT_1, AT_2, ..., AT_N). \tag{3.28}$$

Such a function is called *affine invariant* (see (Amari, 1998) for a discussion).

Unfortunately, while general affine invariant functions may be easily produced, affine invariant probability densities cannot be produced in general. To see this, assume that we have two functions f_1 and f_2 such that for all T, $f_1(T) = f_2(AT)$ for a fixed matrix A. For example, f_1 could be the function f above but with fixed parameters $T_1, T_2, ..., T_N$, and f_2 could be the function f but with parameters $AT_1, AT_2, ..., AT_N$. Let $U = AT$. We show

[18] See (Spivak, 1999a), Volume 1, for a discussion of Lie groups.

that f_1 and f_2 cannot both be probability densities.

$$\begin{aligned} \int f_1(T)\, dT &= \int f_2(AT)\, dT & (3.29)\\ &= \int f_2(U)\frac{1}{|A|}\, dU & (3.30)\\ &= \frac{1}{|A|}\int f_2(U)\, dU & (3.31)\\ &= \frac{1}{|A|}\int f_2(T)\, dT. & (3.32) \end{aligned}$$

Since the integrals $\int f_1(T)dT$ and $\int f_2(T)dT$ differ by a factor of $\frac{1}{|A|}$, they cannot both be equal to one in general. Thus, they cannot both be probability densities.

We choose instead to construct true probability densities that are affine invariant up to scale, or *pseudo-affine invariant*. By this we mean that $p(T; T_1, T_2, ..., T_N) = C(A)p(AT; AT_1, AT_2, ..., AT_N)$ for all affine matrices A. In other words, while the probability density assigned to any two matrices may not be equivalent for a transformation of the samples by a matrix A, the *ratio* of these densities is guaranteed to be the same.

To define such a pseudo-affine invariant density from a set of sample transforms $T_1, T_2, ..., T_N$, one may proceed as follows. Define a kernel function, parameterized by a training transform T_i, that *does not integrate to one*[19], as

$$K(T; T_i) = e^{-\frac{1}{2\sigma^2}\|T_i^{-1}T - I\|_F^2}, \quad (3.33)$$

where $\|\cdot\|_F$ is the Frobenius norm and I is the identity matrix. The square of the Frobenius norm merely sums the squared components of a matrix. Note that 3.33 has maximum value for a fixed T_i when $T = T_i$. This satisfies the criterion described above that the estimator have higher value when T is close to a training sample. Also, when $T_i = I$, the identity matrix, the resulting density-up-to-scale over T is proportional to a Gaussian distribution in the matrix component space, centered at the identity matrix.

We then define a probability density $p(\cdot)$ based upon this kernel to be

$$p(T) = \frac{\sum_{i=1}^{N} K(T; T_i)}{\sum_{i=1}^{N}\int K(T; T_i)\, dT}. \quad (3.34)$$

First we note that $p(T)$ is a valid density since it integrates to 1. Second, note that Equation 3.33 is invariant to the application of the same affine transform to both arguments:

$$\begin{aligned} K(AT; AT_i) &= e^{-\frac{1}{2\sigma^2}\|(AT_i)^{-1}AT - I\|_F^2} & (3.35)\\ &= e^{-\frac{1}{2\sigma^2}\|T_i^{-1}T - I\|_F^2}. & (3.36) \end{aligned}$$

Hence, the numerator in Equation 3.34 is also invariant to such affine transforms. Thus, the density is invariant up to scale as desired.

Finally, to evaluate the density one must evaluate the integrals in the denominator of Equation 3.34 for arbitrary T_i. Taking $\mathbf{v}$ again as the vector of the components in T, I_v the vector of corresponding components in an identity matrix I, letting A be the 6x6

[19] In fact, normalizing this kernel would break the desired invariance property.

matrix representation of T_i^{-1} that acts on the vector $\mathbf{v}$, and setting $\mathbf{u} = A\mathbf{v} - I_v$, we have $d\mathbf{u} = |\det A| d\mathbf{v}$. Then, letting d be the dimension of the set of transforms, we have

$$\begin{aligned} 1 &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \int e^{-\frac{1}{2\sigma^2}\mathbf{u}^T\mathbf{u}} \, d\mathbf{u} && (3.37) \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \int e^{-\frac{1}{2\sigma^2}(A\mathbf{v}-I_v)^T(A\mathbf{v}-I_v)} \, d\mathbf{u} && (3.38) \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \int e^{-\frac{1}{2\sigma^2}(A\mathbf{v}-I_v)^T(A\mathbf{v}-I_v)} |\det A| \, d\mathbf{v}. && (3.39) \end{aligned}$$

Therefore

$$\begin{aligned} & \int K(T;T_i) \, dT && (3.40) \\ &= \int e^{-\frac{1}{2\sigma^2}(A\mathbf{v}-I_v)^T(A\mathbf{v}-I_v)} \, d\mathbf{v} && (3.41) \\ &= \frac{(2\pi\sigma^2)^{\frac{d}{2}}}{|\det A|} && (3.42) \\ &= \frac{(2\pi\sigma^2)^{\frac{d}{2}}}{(\det(T_i^{-1}))^2} && (3.43) \\ &= (2\pi\sigma^2)^{\frac{d}{2}} (\det T_i)^2. && (3.44) \end{aligned}$$

This allows us to compute the denominator of Equation 3.34 in time linear in the number of training transforms.

In summary, we have exhibited the construction of a probability density on affine transformations that is invariant up to scale. It has the desirable property that the maximum contribution to the probability density of a transform T is made when it matches one of the training transforms, and that the amount of this contribution is dependent not upon where T and T_i are in the space of transforms, but only on their group difference $T_i^{-1}T$. This is the property desired of the pseudo-affine-invariant estimate.

There is still a problem however. There is a second type of bias in the function p. The view adopted in this work is that a transform that shrinks by some factor should be the same distance from the origin (the identity matrix) as one that expands by the same factor. The kernel defined in Equation 3.33, however, does not obey this property. For example, if we let

$$T_{double} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.45)$$

and

$$T_{half} = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.46)$$

then $K(T_{double}, I) \neq K(T_{half}, I)$. The kernel given in Equation 3.33 will tend to give lower probabilities to matrices with large components (expanding matrices) and higher probabilities to the inverses of those matrices. To mitigate this effect, we alter the kernel

by approximately *symmetrizing* it in the following way:

$$K_{Sym}(T;T_i) = \exp\left(-\frac{1}{2\sigma^2}\frac{\|T_i^{-1}T - I\|_F^2 + \|T^{-1}T_i - I\|_F^2}{2}\right). \tag{3.47}$$

This is achieved without losing the pseudo-invariance property, as can be easily verified.

Equation 3.47 represents a close approximation to the final form used for the unnormalized kernel in our experiments.[20] Unfortunately, unlike the kernel in Equation 3.33, it is not clear how to find the correct normalization constant for this kernel in closed form. It turns out that this will not be of concern, since we will be adjusting this unnormalized density with a scalar when combining it with the image density.

Our final form for the unnormalized density on affine transforms is thus

$$\tilde{p}(T) = \sum_{i=1}^{N} K_{Sym}(T;T_i). \tag{3.49}$$

The tilde notation indicates that $\tilde{p}$ is not a true density, but only a density up to scale. With a probability density for images (Equation 5.12), and an unnormalized probability density for affine transformations (Equation 3.49), experiments can be performed.

3.6 Experimental results

In the first experiment, the observed image classifier was implemented using the independent pixel model. 1000 training images, randomly selected from the NIST database, were used to estimate a density for each class as described in Section 3.5.1. 100 test samples of each class, *also from the training portion of the NIST database, but distinct from the training examples*,[21] were classified. The full resolution (128x128) binary NIST images were used for both training and testing. The confusion matrix is shown in Table 3.1. The accuracy was 59.7%.

Using the same training data (and the congealing algorithm), the latent image classifier was run. The accuracy dropped precipitously, to 48.4%. Results are shown in Table 3.2. The number of characters misclassified as "1"s can be explained by the fact that many characters can be well matched to the one model by severely squashing them horizontally, as illustrated in Figure 3-4. Since the likelihood of such an unlikely transform does not affect the probability density estimate in this classifier, many characters are confused with "1"s.

For the LT classifier, two additional parameters were optimized before evaluating the

[20] Due to a programming error, the experiments reported used a kernel with a slightly different form:

$$K(T;T_i) = \exp\left(-\frac{1}{2\sigma^2}\frac{\|T_i^{-1}T - I\|_F^2 + \|TT_i^{-1} - I\|_F^2}{2}\right). \tag{3.48}$$

We believe that for practical purposes, the difference between this form and Equation 3.47 is minimal.

[21] The reason that elements of the *training set* from the NIST database were used as *test* data is as follows. The test portion of the NIST database contains writing from a different class of writers than the training data, and we did not wish to model this type of variability here. While the MNIST database of AT&T addresses this issue by mixing the training and testing portions of the NIST database to create a new database, we wanted the higher resolution available with the original NIST database, and so did not use the MNIST data.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	77	2	4	1	1	7	5	2	1	0
1	11	82	0	0	0	0	3	1	3	0
2	4	3	73	5	2	9	2	1	1	0
3	7	1	5	74	0	5	4	1	2	1
4	0	1	3	1	60	3	2	4	7	19
5	13	3	1	16	1	59	3	2	2	0
6	8	2	4	2	6	21	49	0	8	0
7	3	14	15	14	3	1	1	37	9	3
8	7	3	5	12	0	13	2	0	53	5
9	1	6	0	16	3	0	0	8	33	33

Table 3.1: Confusion matrix for the observed image classifier using the independent pixel model. Accuracy was 59.7%.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	76	15	7	0	0	0	2	0	0	0
1	1	87	0	3	2	5	0	0	0	2
2	18	13	54	1	1	0	0	7	4	2
3	9	16	3	61	0	4	0	1	1	5
4	11	26	0	0	53	3	1	0	3	3
5	21	17	0	2	0	56	3	0	1	0
6	30	17	2	1	1	4	34	3	7	1
7	14	50	5	3	5	4	7	11	1	0
8	14	33	3	10	0	1	2	0	37	0
9	23	22	5	8	8	0	6	5	8	15

Table 3.2: Confusion matrix for the latent image classifier using the independent pixel model. Accuracy was 48.4%.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	77	2	4	1	1	7	5	2	1	0
1	11	82	0	0	0	0	3	1	3	0
2	4	3	73	5	2	9	2	1	1	0
3	7	1	5	74	0	5	4	1	2	1
4	0	1	3	1	60	3	2	4	7	19
5	13	3	1	16	1	59	3	2	2	0
6	8	2	4	2	6	21	49	0	8	0
7	3	14	15	14	3	1	1	37	9	3
8	7	3	5	12	0	13	2	0	53	5
9	1	6	0	16	3	0	0	8	33	33

Table 3.3: Results for the independent pixel density and the LT classifier. This classifier outperformed the other two classifiers, as predicted. Accuracy was 64.0%.

test examples. The first parameter σ^2 is the variance of the kernel used in the transform density estimate (Equation 3.47). In addition, the logarithm of the transform density and the logarithm of the image density were combined using a weight λ according to

$$\log(p_{final}(I_L, T)) = \log(p(I_L)) + \lambda * \log(p(T)). \tag{3.50}$$

The weight λ compensates for two factors: a) that $p(T)$ is not a true probability density since it is unnormalized, and b) that the independent pixel model typically causes $p(I_L)$ to be a gross underestimate of the true probability of the latent image, since many pixels effectively represent the same information. Thus the λ balances the transform density to better match the image density. These parameters were adjusted to maximize classification accuracy on the training data. Because there were so few parameters relative to the size of the training set, overfitting was not a significant issue. Ultimately, then, Equation 3.50 was used to evaluated log likelihoods in the "maximum likelihood" LT classifier.

The results for the LT classifier were better, as predicted, than for either of the other classifiers. The accuracy rate for this classifier was 64.0%. The results for this classifier are reported in Table 3.3.

3.7 Improving recognition with the Hausdorff distance

While the principles of factorized classifiers were upheld by the results in the previous section, the results were not spectacular to say the least. This is probably because the independent pixel probability model for both observed images and latent images is a very poor model.

There are many schemes in machine vision that implicitly model the dependencies between neighboring pixels in the same image. One such technique is the Hausdorff distance measure for computing the similarity of two binary images. In this section, we briefly explain how the Hausdorff measure, when used in a nearest neighbor scheme, can be thought of as putting an implicit probability distribution over a set of images. We then use this scheme to improve all three classifiers from the previous section. We shall see that while the classifiers maintain the same relative performance, they all improve in accuracy.

3.7.1 Nearest neighbor as an ML classifier

The simple Euclidean nearest neighbor classifier can be thought of as defining an implicit probability density over classes in the following sense. The nearest-neighbor rule (Duda and Hart, 1973), using a Euclidean distance between images,

$$D(I^1, I^2) = \sqrt{\sum_{i=1}^{P} (I_i^1 - I_i^2)^2}, \tag{3.51}$$

is equivalent to a limiting case of an ML classifier in which the densities have been estimated by a non-parametric kernel estimator (and the class prior is uniform). As the (spherical and unimodal) kernel shrinks to a delta function, the likelihood of a test sample being in a particular class is dominated, and in the limit solely determined, by the nearest element of that class. Under this type of probability density, the likelihood ratio between the nearest class and all other classes goes to infinity.[22] Hence, the class with maximum likelihood is the class with a sample nearest to the test sample.

3.7.2 The Hausdorff distance: a robust distance measure for binary images

However, the Euclidean distance function for describing the distance between binary images is highly sensitive to many types of changes that people (which we cite as examples of good classifiers) consider to be nearly irrelevant. Among these are small affine transformations of one of the images. These will be addressed by our generative model, which explicitly models affine deformations. But there are other changes, such as the thickness of the strokes in the character, that can dramatically affect distance measurements between images and are not modeled by our generative process. To mitigate the effects of this variability on distance estimates, and hence on the implicit probability densities, we adopt a distance measure, the *Hausdorff distance*(Huttenlocher et al., 1993), which is designed to reduce the effect of local dilations and erosions (as caused, for example, by line thickness) in binary images. This is a simple change from the standard Euclidean distance that substantially improves the performance of all three classifiers relative to a Euclidean nearest neighbor scheme and to the independent pixel model.

The Hausdorff distance gives a measure of the difference between two binary images. Let A be the set of two-dimensional points defined by the pixel centers of the foreground pixels of image I^1. Let B be the set of points defined by the foreground pixels of image I^2. Then the (asymmetric) Hausdorff distance is given by

$$D(I^1, I^2) = \max_{a \in A} \min_{b \in B} \|a - b\|. \tag{3.52}$$

In other words, the Hausdorff distance between two point sets is the maximum distance from any point in the first set to its closest neighbor in the second set.

This can also be thought of as the number of single pixel dilations necessary for image I^1 to completely "cover" image I^2. That is, considering the 1-valued pixels as foreground and the 0-valued pixels as background, the foreground of image I^1 is dilated one pixel at

[22] This assumes the nearest neighbor is unique, i.e. that images from two different classes are not the *exact* same distance from the test image.

a time until the foreground in image I^2 is a subset of the foreground in image I^1. This can be thought of as a modified template match, where the dilation is an effort to discount slight misalignments due to stroke thickness and local variations in curvature, etc. It is substantially more robust for handwritten character recognition than a simple template matching approach.

There are many common variations of the Hausdorff distance. One such variation, sometimes referred to as the Hausdorff fraction, is the fraction of foreground pixels in image I^1 that are within a distance δ of any foreground pixel in I^2. By increasing the tolerance δ, the measure becomes more robust to spurious pixels and distortions of the two images. Of course, the "distance" is also reduced for non-matching images as δ is increased. We denote the Hausdorff fraction with tolerance δ by $H_\delta(I^1, I^2)$. In this work we use a symmetric version of the Hausdorff fraction with $\delta = 1$, namely

$$D_S(I^1, I^2) = D_1(I^1, I^2) + D_1(I^2, I^1). \tag{3.53}$$

To classify a test character J using the Hausdorff nearest neighbor method, one finds the element of the training set that maximizes this measure and assigns the corresponding label:

$$c^* = \arg\max_{c \in \mathcal{C}} \left[\max_{I \in \mathcal{I}_c} D_S(I, J) \right]. \tag{3.54}$$

Here $\mathcal{I}_c$ represents the training set for class c. This is just a nearest neighbor classifier using the symmetric Hausdorff fraction instead of the traditional Euclidean distance. Note that a nearest neighbor classifier based upon the Hausdorff distance is also equivalent to an ML classifier for some probability mass functions on each class.[23] In particular, interpreting the maximum Hausdorff fraction for a set of images as a negative log likelihood will enable us to combine such a measure with a transform probability estimate in a principled fashion.

3.7.3 Hausdorff experiments

While the Hausdorff distance is robust to small perturbations of a character, it does not handle larger distortions well. In particular, translations, rotations, shears, and scales of more than a pixel in any direction substantially affect the distance from one character to another using this distance measure. Hence, such a classifier would not be expected to exhibit very good performance unless there were enough characters in the training data for each class to represent the spatial variability due to random distortion in addition to the variability in latent images.

To test the Hausdorff nearest neighbor classifier in the observed image classification framework, we again used 1000 training examples of each class, selected at random but without replacement, from the NIST handwritten digit database. As reported in Table 3.4, this classifier correctly classified 94.6% from a (disjoint) test set of 1000 examples of the same database. We shall see that even with such large training sets, it is easy to improve upon the performance of this classifier by using the previously described factorized model to more efficiently model the latent images and transforms.

[23] For any deterministic classification rule acting on a finite sized input space, there exists some set of probability mass functions (one for each class) that reproduce the same result as an ML classifier using those probability functions. Such probability mass functions can be easily constructed (in theory) by collecting the N_j elements of the input space that are classified as class j and assigning them probability $\frac{1}{N_j}$ in class j and probability 0 in the other classes.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	98	0	0	0	0	0	1	0	1	0
1	0	99	0	0	0	0	0	0	1	0
2	6	0	89	1	0	0	1	2	1	0
3	1	0	0	92	0	1	0	0	5	1
4	0	1	0	0	96	0	0	0	0	3
5	2	0	0	5	0	91	1	0	1	0
6	1	0	0	0	0	0	99	0	0	0
7	0	2	0	0	2	0	0	95	0	1
8	1	2	0	4	0	0	0	2	89	2
9	1	0	0	0	0	0	0	1	0	98

Table 3.4: Results of using the Hausdorff nearest neighbor method on observed images. The accuracy was 94.6%. The results are dramatically better than for the independent pixel model, but still inferior to the factorized classifier using the Hausdorff distance.

We next used the Hausdorff nearest neighbor scheme within the latent image classification framework. That is, we applied the Hausdorff nearest neighbor scheme to a set of estimated latent images from congealing and each estimated latent image for a test character. Here the accuracy again decreased relative to the observed image classifier, this time to 89.4%. The confusion matrix is shown in Table 3.5. This can again be explained by inappropriate alignment to the "1" model and similar effects.

With a probability density for affine transformations in hand, the Hausdorff fraction can be combined with information from the transform density to improve performance.[24] By combining a scaled estimate of a transform's log-probability density ($p(\hat{T}_J)$) with the Hausdorff fraction obtained from the latent image,

$$c^* = \arg\min_{c \in \mathcal{C}} \left[\min_{I \in \mathcal{I}_c} \left(1 - D_S(\hat{I}_L, \hat{J}_L) \right) + \lambda \log p(\hat{T}_J) \right]. \tag{3.55}$$

performance was increased to 97.4%. Again, the parameter λ was estimated by maximizing the classification accuracy on the training data. The confusion matrix for this experiment is shown in Table 3.6.

The fact that these experiments exhibited the same relative performance as the independent pixel model classifiers strengthens are qualitative analysis of the benefits of a factorized model. While the observed image classifier would be expected to asymptotically overtake the LT classifier in both the independent pixel case and the Hausdorff nearest neighbor case, for data sets of moderate size (1000 training examples), the latent-image transform decomposition appears to have an advantage.

[24] It is assumed that the $1 - D_S$, one less the Hausdorff fraction, is a quantity that behaves like a negative log probability density. Thus, it makes sense to add this quantity to the negative log probability density of the transform to obtain a final score for the match between a test image and a particular training image. Because the Hausdorff fraction varies with image resolution, the Hausdorff fraction was modified by a scalar adjustment factor γ before being added to the transform probability density. This γ was optimized manually for each image resolution on a small hold out set.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	86	14	0	0	0	0	0	0	0	0
1	0	100	0	0	0	0	0	0	0	0
2	1	13	82	0	1	0	0	2	1	0
3	0	8	0	92	0	0	0	0	0	0
4	0	9	0	0	90	0	0	0	0	1
5	0	4	0	1	0	92	3	0	0	0
6	0	2	0	0	0	1	97	0	0	0
7	0	4	0	0	0	0	0	95	0	1
8	0	34	0	0	0	0	0	0	66	0
9	0	6	0	0	0	0	0	0	0	94

Table 3.5: Results for the latent image classifier using the Hausdorff nearest neighbor method. The performance (89.4%) is again worse than for either the observed image classifier or the LT classifier using the Hausdorff distance, as predicted. Notice again the large number of characters misclassified as "1"s. This is due to the fact that most characters, under arbitrary affine transformation, can be made to look like a one. Without a penalty on such severe squashing functions, confusion is likely to occur.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	97	1	0	0	0	0	2	0	0	0
1	0	100	0	0	0	0	0	0	0	0
2	2	1	93	1	1	0	0	1	1	0
3	1	0	0	97	0	1	0	0	1	0
4	0	0	0	0	99	0	0	0	0	1
5	0	1	0	1	0	95	3	0	0	0
6	0	0	0	0	0	0	100	0	0	0
7	0	0	0	0	1	0	0	97	1	1
8	1	1	0	1	0	0	0	0	97	0
9	0	0	0	0	0	0	0	0	1	99

Table 3.6: Classification results for the LT classifier using the Hausdorff distance on images. The accuracy was 97.4%.

Image density	Observed Img.	Latent Img.	Latent Img.-Transform
Independent pixels	59.7%	48.4%	64.0%
Hausdorff	94.6%	89.4%	97.4%

Table 3.7: Summary of classification results for 1000 training example experiments. All experiments in the table used 1000 training examples of each digit class. The first row of the table shows experiments using the independent pixel image models. The second row shows experiments using the Hausdorff distance measure and a nearest neighbor scheme. Notice that the relative performance of the three classifiers in each row is the same for the independent pixel model and for the Hausdorff classifiers.

3.8 Summary

This chapter has presented a generative image process which models the image as a canonical "latent image" and a transform away from that latent image. By taking advantage of the approximate independence of these parts, classifiers can be improved, whether they are based on simple probabilistic models of images (independent pixel models) or implicit probability densities (Hausdorff nearest neighbor).

Table 3.7 summarizes the results of this chapter. The point of these results is not to compete with the best classifiers available for handwritten digits. Rather, it is to establish a technique that can be adapted to cases in which we have only a single training example per class. In the next chapter, we will be comparing similar classifiers, but this time using only a single training example per digit class. Other information about the classifier will be drawn from support sets consisting of handwritten letters.

Chapter 4

A One Example Classifier

4.1 Introduction

In the previous chapter, several handwritten digit classifiers based on 1000 training examples of each class were explored. We saw that by treating the latent images as independent from the affine variability in a set of digits, we could form a factorization that allowed efficient density estimation of each class. The question posed in this thesis is, "Can knowledge learned in developing such factorized densities be used in new tasks to improve the efficiency of learning?"

The key insight of this chapter, and of the first half of the thesis, is that because distributions over spatial transformations in handwritten characters are similar from class to class, a transform distribution estimated for one class makes a good surrogate for the true distribution from another class.

This chapter uses this insight to develop handwritten digit classifiers from a single training example of each digit. It achieves this by using support sets, in the form of sets of handwritten letters, to model densities over affine transformations. These densities are then used in place of the actual handwritten digit transformation densities. These techniques are used to produce the best single example classifiers reported to date. We start the chapter by revisiting the generative image model introduced in Chapter 3.

4.2 A change to the image model

Suppose we consider a slight change to the generative model proposed in Section 3.2. In particular, we consider the modified model shown in Figure 4-1, in which the transformation variable no longer depends upon the class variable.

In this scenario, the density over transformations is the same for every class. If the model is interpreted to be valid for all handwritten characters, then it is clear that the densities over transforms for any handwritten *digit* can be learned by looking at transform densities for handwritten *letters*. This is the idea used in this section. (The validity of this assumption is discussed below.)

4.3 Similarity of distributions

Until now, we have claimed without evidence that the densities over affine transforms for characters were "close enough" to warrant borrowing the densities from letters as estimates

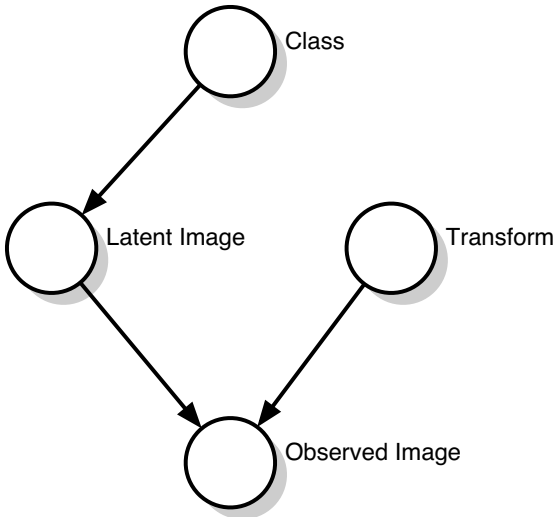


Figure 4-1: [A revised generative image model in which the transformation density does not depend upon the character class. This allows us to estimate this density with one class and share it with another class.

of densities on transforms for digits. In this section, we present experimental evidence in support of this claim. We compare the similarity of estimated distributions over transforms across different types of characters. The goal is not to show that distributions over transformations are exactly the same across all characters. Most likely, they are not. Instead, we show that distributions over transforms of letters and digits are closer to each other than they are to various "straw man" distributions that might well be chosen as ad hoc densities over transformations, in the absence of other prior knowledge. We conclude that borrowing digit transform densities from letters is superior to several other densities that one might craft by hand.

We defined the following straw man, or ad hoc, distributions:

- A 4-dimensional distribution[1] with each coordinate uniformly distributed between -1 and 1.
- A 4-dimensional Gaussian distribution centered at the origin, with diagonal covariance equal to 1.
- A 4-dimensional Gaussian distribution centered at the identity matrix (that is, at the vector $v^T = [1\,0\,0\,1]$) with diagonal unit covariance.

From a learning-to-learn point of view, the specification of ad hoc distributions already represents "cheating," since we are incorporating our own prior knowledge into the transform density, rather than using a more uninformative prior, such as an (improper) uniform distribution. Nevertheless, we will show that in the KL-divergence sense, letter transforms and digit transforms are closer to each other than they are to any of these ad hoc distributions.

[1]To simplify these experiments the translational component of the affine transform was ignored, and densities over the set of non-singular 2x2 matrices, representing the set of two-dimensional linear transforms, were considered. Hence, the estimated densities had four coordinates.

A sufficient condition for justifying the lending of transform densities from letters to digits is that *the densities over transforms for all characters be "similar enough."* Here, we take similar enough to mean, more similar than to our straw-man transform distributions. Of course, in a true application scenario, if we had densities over all of the digits in order to verify this claim, then there would be no need to borrow densities in the first place. Thus, for the purposes of simulating a true learning-to-learn scenario, we assume that we have no knowledge of digit transform densities. Instead, we must take the similarity of letter transform densities as an approximation to the similarity between various letter and digit transform densities. If the letter transform densities are more similar to each other than to the straw-man distributions, then we shall make the *assumption* that the letter distributions are also similar to the digit distributions, since letters and digits are both subsets of the general class of handwritten characters, and share a common generative source.

Ten sets of 100 letters (a-j) from the NIST database were congealed. The derived transforms were used to compute pseudo-probabilities according to Equation 3.49 (but without the translational component of the transforms). A discrete approximation to the distribution was then computed by computing the unnormalized density at a large number of grid points. This also enabled approximate normalization of the distributions.

This process produced a discrete approximation to a transform distribution for each letter (a-j). The straw-man distributions were discretized at the same resolution. We then computed the KL-divergences, defined as

$$D(p\|q) \equiv \sum_{x\in\mathcal{X}} p(x) \log \frac{p(x)}{q(x)}, \tag{4.1}$$

for all density pairs p and q, and in both orderings.

The matrices in Figure 4-2 show KL-divergences between various transformation densities. The upper left matrix in the figure shows similarities between the letter transform densities with each other (rows 1-10 and columns 1-10), the third straw-man density with each of the letters (bottom row and rightmost column), and the straw-man distribution with itself (bottom right element of the matrix).

The matrix has been normalized so that the lowest value of the KL-divergence is black and the highest value is white. It is easy to see that the letter transform densities are much more similar to each other than they are to the straw-man density. The difference was even more pronounced for the other straw-man densities. That is, the third straw-man density provided the best approximation to the letter transform densities. However, even the best straw-man distribution was a poor approximation to the letter distributions. In fact, the *minimum* KL-divergence between a letter distribution and the straw-man distribution was more than twice as large as the *maximum* KL-divergence between any two letter distributions. From this we conclude that sharing transform distributions among letters is superior, in the KL-divergence sense, to hand crafting transform distributions.

Using the assumption that similarity in letter transform distributions implies similarity in all character transform distributions, we adopt approximations to digit distributions by borrowing from the letters. Since in reality, we do have access to digit transform distributions, we can check the validity of this assumption.

The matrix at the bottom of Figure 4-2 shows KL-divergences among the digits (upper 10 rows and leftmost 10 columns), letters(rows 10-20 and columns 10-20), and the straw-man distribution (single bottom row and rightmost column). Once again, we see that all of the digit and letter distributions are significantly closer to each other than they are to the

straw man distribution. The matrix in the upper right of the figure shows the digit-to-digit and digit-to-straw-man KL-divergences.

Thus, while we had to make an assumption that the similarity of the letter distributions was representative of the similarity of character transform distributions in general, we did not have to blindly share information between distributions. If the transform distributions over letters were very different across classes, it would suggest that such a density-borrowing scheme, as discussed below, would not be appropriate. Hence, like Bollacker and Ghosh's supraclassifier architecture described in Section 1.2.3, we have some protection in our method against having performance degrade in a new task by making inappropriate assumptions about the relationship between old tasks and new tasks. Next, armed with evidence that densities are sharable across characters, we describe the construction of a one-example classifier.

4.4 Building the one example classifier

The basic sequence of steps for building a one example classifier using support sets from other classes is as follows:

1. Obtain a set of training examples for each support class.

2. Run the congealing algorithm for each of the support sets, producing a set of latent images and transforms for each support class.

3. Estimate a density for the latent images and for the transforms, as was done in Chapter 3.

4. Build an ML classifier for the support classes using these densities.

5. Optimize the transform density kernel parameter σ^2 and latent image-transform weighting parameter λ by maximizing the accuracy of the classifier on the training support sets.

6. Borrow a set of transforms for the one example classifier. This can be done by combining transform densities from the support classes in some way, or just using the transform density from a single support class.

7. Obtain one example of each target class.

8. Estimate a latent image density for each class from the single example. For the independent pixel model, this is done according to the method described in Section 3.5.1. For the Hausdorff nearest neighbor scheme, the example itself is the entire model.

9. Using the latent image density from the single example and the borrowed transform density, we have all the components of a classifier.

The specifics of the implementation were as follows. In the first experiment, we implemented a one sample classifier using handwritten *letters* from the NIST database as support sets. The independent pixel model (Section 3.5.1) was used to estimate probability densities for latent images. Ten letter data sets (a-j), each consisting of 100 examples of a particular letter, were congealed. Ten transforms were taken randomly from each congealed letter data

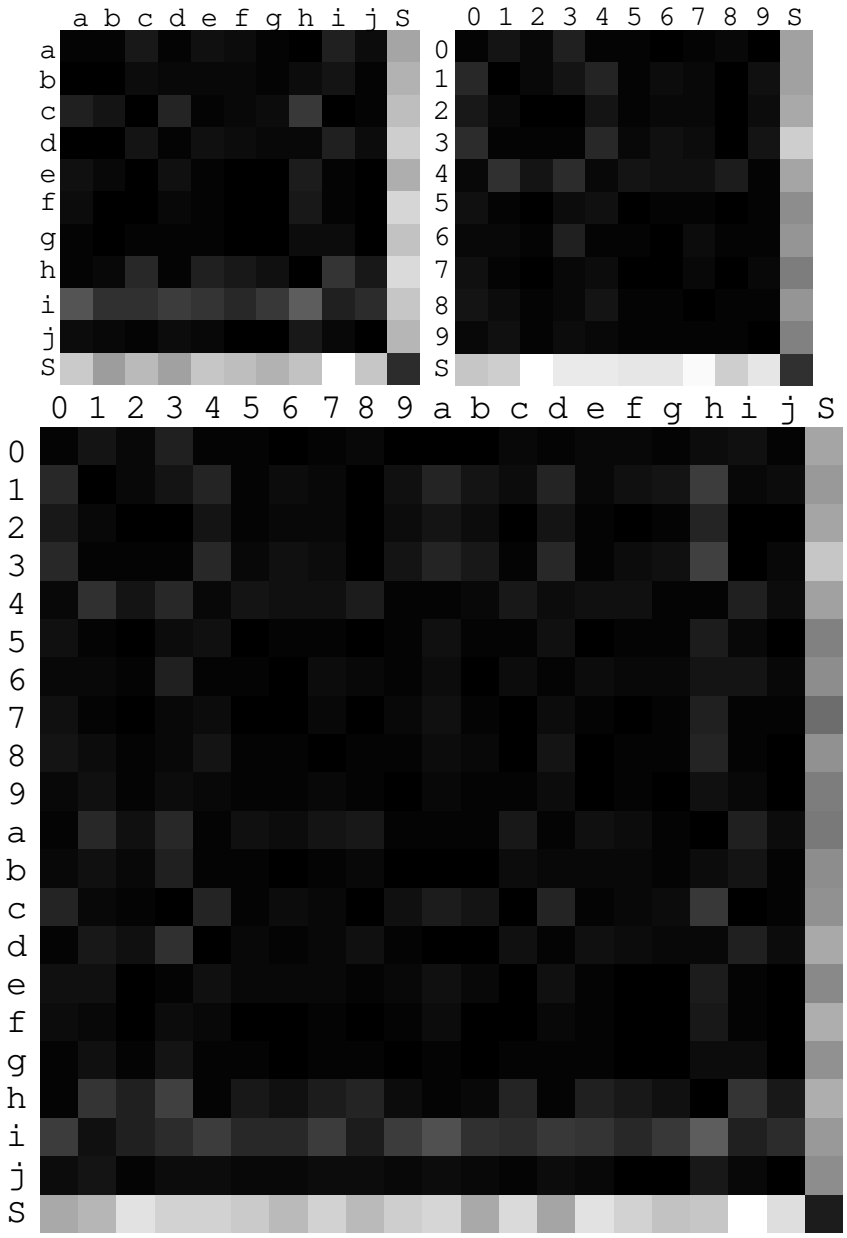


Figure 4-2: Measures of variation for probability densities over spatial transformations. The figure in the upper left shows graphically the KL-divergences for probability densities over transforms for the letters a-j and for an alternative "strawman" distribution. All KL-divergences are relatively low (black) except for those between the strawman and the digit distributions (light gray at right and bottom). This implies that the letter distributions are closer (in the KL sense) to each other than they are to the straw man. The upper right figure shows the relationship of the distribution on transformations of digits to the strawman. Finally, the large figure at the bottom shows all pairwise KL divergences between the digits 0-9 (top rows and left-side columns), the letters a-j (rows 11-20 and columns 11-20), and the strawman (bottom row and rightmost column). The diagonals of all of the plots show the KL-divergence between two densities estimated from different samples of the same distribution.

set and combined into a set of 100 transforms which constituted the "borrowed" transform set $\mathcal{T}_B$.

A letter classifier based on the factorized probability model described above was built. Test letters were classified according to Equation 3.50. Several iterations of the classifier were run in order to optimize two parameters of the classifier: the λ of Equation 3.50 and the σ^2 parameter of Equation 3.47. If one wants principled estimates of these parameters, then the only option is to optimize them for the letter classification problem and then borrow their values for the digit classifier, since with only a single example of each digit, it will be impossible to create a hold-out set for parameter optimization in our digit classifier. Because the parameters λ and σ^2 control generic quantities that are not directly related to the form of each latent image distribution, we hope that their optimal values for the letter classifier will provide effective values for the digit classifier.

Next, a single example of each digit class was drawn randomly from the NIST database as a training example. Figure 4-3 shows the example of each character used in the first experiment. This single example was used to define the latent image density for the class. The transform density was built from the borrowed transform set $\mathcal{T}_B$ according to the density estimator described above. This completes the definition of our handwritten digit models from a single example.

4.4.1 Aligning a test sample with a single-image model

But another major challenge that has not been discussed is bringing a test example of unknown class into correspondence with the single training example of each digit so that the factorized probability can be evaluated, i.e. congealing a test example with a one-example model. Previously, when test congealing was implemented, it was assumed that there were enough training examples of each class to benefit from the natural blurring of the training examples. When the training set for each class consists of only a single example, this blurring does not occur, and problems of local minima in image alignment can arise. The problem is circumvented in the following way:

1. A synthetic image set is created for a digit class j from the single training example by operating on the example with each of the transforms $T_i \in \mathcal{T}_B$.

2. Test congealing is performed with the synthetic image set exactly as if there were a real training set for each digit.

A subset of such a synthetic data set is shown in Figure 3-1. Notice that there is only a single latent image. That is, all of the images in the figure are equivalent up to an affine transformation.

Although convergence rates for test examples are not quite as good when congealing with such artificial data sets as when congealing with real training sets, this method is still good enough for alignment in most cases, especially if the one example of each digit set is typical, which we can interpret to mean "not too different" from other members of the class. With this final problem solved, the one-example classifier is ready to be applied.

Since these experiments were based upon only a single example of each class, the results would be expected to vary greatly. If the training set contains a typical example of each class, the results would tend to be much better. If one or more of the single examples of a class is atypical, then performance for that class is likely to be very poor.[2] Thus, the

[2]This corresponds nicely to learning in humans as well.

Image density	Observed Img.	Latent Img.	Latent Img.-Transform
Independent pixels	35.2/3.6	15.2/4.1	41.9/5.1
Hausdorff	32.6/4.6	63.6/3.9	74.7/3.6

Table 4.1: Summary of results for experiments with one randomly selected training example per class. The first row of the table shows experiments using the independent pixel image models. The second row shows experiments using the Hausdorff distance measure and a nearest neighbor scheme. The left number in each cell is the mean percent correct across 10 experiments and the right number is the standard deviation of these values.

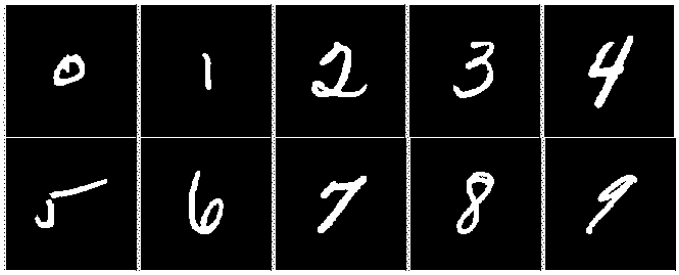


Figure 4-3: A randomly chosen training set for one of the one-example classifier experiments. Note the poor quality of the digits five and nine. This led to poor performance on these digits, as shown in Table 4.2.

experiments were repeated for 10 different (random) training sets to get a measure of the variability in performance.

4.4.2 Results

The experiments were run using both the independent pixel models and the Hausdorff nearest neighbor implicit density. For comparison, the observed image and latent image classifiers (in addition to the LT classifier described above) were also run. The results are summarized in Table 4.1.

We make just a few comments. Once, again the relative performance of the three types of classifiers is shown. Of course, in this case, the LT classifier has an extra advantage in that it is using a density estimated from a large number of letters.

The main point of interest however, is that the mean classification accuracy for the Hausdorff-based LT classifier was 74.7%, with a standard deviation of approximately 3.6%. This is better than any previously reported result for such a task.

It is instructive to look at the confusion matrix for one of the Hausdorff-based LT classifier experiments (there were 10 in all), shown in Table 4.2. The set of training examples for this classifier is shown in Figure 4-3.

In the figure, note the rather unusual sample of the digit five that was used for training. As seen in the corresponding table entry (the sixth row), the classifier was essentially unable to learn anything from this example, correctly identifying only 12 of the 100 test fives correctly, which is approximately equal to chance behavior. The flatness of the nine also gave the system trouble as can be seen from the confusion matrix.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	100	0	0	0	0	0	0	0	0	0
1	3	96	0	0	0	0	0	0	0	1
2	26	0	55	0	2	0	0	8	7	2
3	25	1	1	52	1	0	0	0	12	8
4	1	0	0	0	95	0	0	1	0	3
5	55	0	1	8	5	12	1	2	14	2
6	27	0	0	0	0	0	73	0	0	0
7	1	2	0	0	0	0	0	95	1	1
8	12	1	0	1	2	0	0	2	75	7
9	5	0	0	0	47	0	0	0	0	48

Table 4.2: A confusion matrix for the one-example classifier with randomly chosen training digits. Note the very poor performance on the digit five, due to the low quality of the training sample. The accuracy for this individual experiment was 70.1%. The mean accuracy for the Hausdorff LT classifier was 74.7%.

Image density	Observed Img.	Latent Img.	Latent Img.-Transform
Independent pixels	41.1%	16.2%	64.7%
Hausdorff	48.2%	79.2%	88.6%

Table 4.3: Results of experiments with one *hand-picked* training example per class. The first row of the table shows experiments using the independent pixel image models. The second row shows experiments using the Hausdorff distance measure and a nearest neighbor scheme.

4.5 Hand-picking the training example

Another set of experiments was run to simulate an active teaching scenario in which a teacher cooperates with the learner by providing an example of each character to be as "typical" as possible. In these experiments, the single example of each digit was selected manually to be generally representative of the NIST database characters for that class. Results from these experiments are summarized in Table 4.3.

For this experiment, the performance increased dramatically to 88.6%. In comparison, the simple Hausdorff nearest neighbor classifier, applied to the single training example case gave a recognition rate of only 48.2%. The affine adjusted Hausdorff nearest neighbor achieved a recognition accuracy of 79.2%. Adding the previously learned density on transforms thus made a dramatic difference in this sparse data setting.

The achievement of almost 90% accuracy for a single example classifier is among the significant achievements of this thesis. While this number does not compete with top classification results for large training sample classifiers (accuracy $> 99\%$), it represents a significant first step in a new domain, trying to get the most out of a single example. By further factorizing distributions for character support sets, it is likely that these numbers can be improved and extended to other data types.

In addition it strongly suggests that most of the information about a character is contained in a single image of that character. Along with prior knowledge, a single example gives us most of the information we need to make a classifier with non-trivial performance.

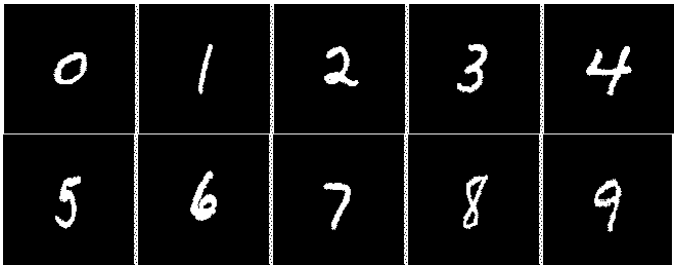


Figure 4-4: The training set for the hand-picked one-example classifier experiment. Each digit was selected from the first 100 examples in the training set, chosen subjectively as a "canonical" example of each digit. The confusion matrix for the classifier is given in Table 4.4.

	Assigned label									
True Label	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	97	0	0	0	0	0	3	0	0	0
1	0	100	0	0	0	0	0	0	0	0
2	3	0	81	4	0	1	3	3	3	2
3	1	0	1	73	0	17	0	0	6	2
4	0	0	0	0	88	0	3	0	5	4
5	0	0	0	0	0	75	15	0	10	0
6	2	0	0	0	0	0	98	0	0	0
7	1	1	0	0	0	0	0	88	1	9
8	2	0	0	3	0	2	1	0	92	0
9	3	0	0	0	0	0	0	1	2	94

Table 4.4: The confusion matrix for the hand-picked one-example classifier. Unlike the randomly chosen digits, the worst classification rate for any digit is 73% (for the digit 3). The overall accuracy for this test was 88.6%

4.6 New questions

Since the success of these methods is heavily dependent upon the effectiveness of the congealing alignment algorithm, this type of approach raises a host of questions. How robust is congealing? How computationally expensive is it? How does it compare to other methods of preprocessing images? How can it be generalized? These questions and other quantitative details about congealing are addressed in the next chapter.

Chapter 5

A Detailed Look at Congealing

In the last chapter, we presented a framework for developing densities on a set of transforms and then using these densities in a new learning problem to achieve a highly efficient classifier. *Congealing*, a process of jointly aligning a set of images, was at the center of this method.

In this chapter, we take a more detailed look at congealing. Our aim is to address, and give at least partial answers to the following questions.

- Under what conditions is the congealing objective function the optimal one? What specific model is associated with this objective function?
- Under this model, how does congealing compare to traditional forms of preprocessing such as centroiding and deshearing for the purposes of alignment or correspondence?
- How sensitive is congealing to evenly distributed noise? What about highly structured noise?
- How does congealing relate to traditional preprocessing methods when the goal is developing a model or probability density estimate of a class?
- How does congealing relate to other methods, such as the "tangent distance" method, for addressing spatial variability in image data sets?
- How often does congealing get stuck in local minima of the objective function?
- What is the computational complexity of congealing?
- How can congealing be generalized? Under what circumstances is it applicable?

A variety of experimental results are presented to support comparisons to other methods. In addition, generalizations of several aspects of congealing are presented, and preliminary results for several data types are presented.

5.1 Congealing and maximum likelihood

In this section, we present several properties of entropy and probability in order to better understand congealing. Most of these analyses require strong assumptions such as pixel independence. They nevertheless provide insight into the workings of the algorithm.

5.1.1 Random transformations increase entropy

Consider again the generative model of Figure 3-3, but with a new restriction on the set of transformations. For the following argument, we consider only *permuting transformations*, i.e. transformations that only shuffle pixels around in the image rather than combining, mixing, or eliminating them. Note that if a *single* such transformation T is applied to every sample of the latent image random variable $\mathbf{I}_L$, the entropy of the new image random variable $T(\mathbf{I}_L) \equiv \mathbf{I}$ will be the same, since no information is gained or lost. That is,

$$p_{\mathbf{I}_L}(I_L) = p_{T(\mathbf{I}_L)}(T(I_L)) \equiv p_{\mathbf{I}}(I), \tag{5.1}$$

so

$$H(\mathbf{I}_L) = H(T(\mathbf{I}_L)) \equiv H(\mathbf{I}), \tag{5.2}$$

when T is fixed.

However, if each latent image is transformed randomly and independently, then the distribution over transformed images $\mathbf{I}$ can be written

$$\begin{aligned} p_{\mathbf{I}}(I) &= \sum_{T \in \mathcal{T}} p_{\mathbf{T}}(T) * p_{\mathbf{I}}(I) && (5.3)\\ &= \sum_{T \in \mathcal{T}} p_{\mathbf{T}}(T) * p_{T(T^{-1}(\mathbf{I}))}(I) && (5.4)\\ &= \sum_{T \in \mathcal{T}} p_{\mathbf{T}}(T) * p_{T(\mathbf{I}_L)}(I) && (5.5)\\ &= \sum_{T \in \mathcal{T}} p_{\mathbf{T}}(T) * p_{\mathbf{I}_L(\mathbf{I},T)}(I_L(I,T)). && (5.6) \end{aligned}$$

$p_{T(\mathbf{I}_L)}(I)$ assigns a probability to I based upon its unique latent image associated with a transform T. Thus, $p_{\mathbf{I}}(\cdot)$ is a probability distribution that is a convex combination of probability distributions $p_{T(\mathbf{I}_L)}(\cdot)$, each of which has the same entropy as the original latent image distribution (by Equation 5.2).

By the concavity of entropy as a function of probability distributions,

$$H(\mathbf{I}) \geq H(\mathbf{I}_L). \tag{5.7}$$

See (Cover and Thomas, 1991), page 30, for a proof of the concavity of entropy as a function of the distribution.

Undoing the entropy increase

Since the observed image distribution has higher entropy than the latent image distribution for all possible *independent* permuting transform distributions, we attempt to recover the set of latent images from the observed image by minimizing the entropy of the observed images.

If a completely arbitrary set of possible permuting transforms were allowed, it would be trivial to reduce the entropy of the observed image set using permuting transforms: one could simply transfer all of the black pixels in each image to the left side of the image and transfer the remaining white pixels to the right side. For images with equal numbers of black and white pixels, this would produce a zero entropy configuration. However, this procedure is unlikely to reproduce the original latent image set.

However, for a small set of transforms, say translations, the probability is low that any of the transforms, acting upon a latent image in its original position, will increase the probability of this image.

We conclude that for most latent images, the maximum probability position is in the original position rather than some transformed position. The greater the number of transformations allowed, the less likely this is to be true, since there is almost always *some transformation* that will make a set of pixels more likely under a generative model. The set of affine transformations is a small enough set so that the probability of being able to increase the probability of an image from the original latent position is small, although it can occur (especially in the case of zeroes, whose density is approximately invariant under rotation). From here forward, we assume that the maximum probability transformation of an image is, with high probability, the original latent image.[1]

If only one latent image were perturbed from its original position, then recovering the one missing latent image would be straightforward. We could estimate the latent image density from the set of unperturbed latent images and maximize the probability of the perturbed image under the set of transformations. However, since typically all of the images have been perturbed, we can only recover the latent images up to a global transformation of all of the images. But under the assumption of permuting transformations, all of these latent image configurations assign the same probability to each image, and so serve essentially the same purpose for classification.

5.1.2 Upper bound on image entropy

If we accept that minimizing $H(\mathbf{I})$ is a good way to recover the latent images from a data set, then we need a procedure for doing this. Note that the summed pixelwise entropies are an upper bound on the entropy of an image process:

$$\begin{aligned} H(\mathbf{I}) &= H(x_1, x_2, ..., x_P) && (5.8)\\ &= H(x_1) + H(x_2|x_1) + ... + H(x_P|x_1, x_2, ..., x_{P-1}) && (5.9)\\ &\leq H(x_1) + H(x_2) + ... + H(x_P). && (5.10) \end{aligned}$$

The last step follows since conditioning always reduces entropy or leaves it unchanged (Cover and Thomas, 1991). By reducing this upper bound on the image entropy (through transformations), we hope to reduce the entropy of the distribution itself. Next it is shown that for an image process with independent pixels, illustrated by the graphical model in Figure 5-1, there is equality in the final step, and the pixelwise entropy minimization becomes equivalent to a maximum likelihood method.

5.1.3 An independent pixel model

We now turn to a brief analysis of congealing under the assumption that the pixels in the latent image are generated independently given their class, as shown in Figure 5-1. We continue to assume that transformations only permute pixels, rather than mixing pixels or discarding them off the edge of the image.

[1] Such an analysis suggests a method for setting the number of transformations in a factorized model. If we wish to be able to recover the latent image with probability $1 - \epsilon$, then we cannot allow more than some number of transformations as determined by ϵ.

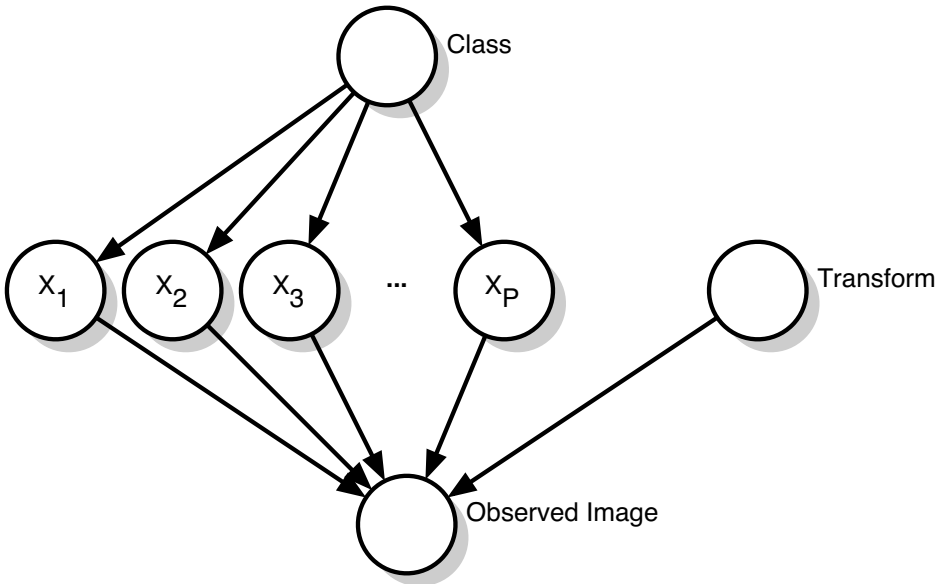


Figure 5-1: An independent pixel model.

Recall that the quantity to be minimized by the congealing algorithm, ignoring the penalty on transform parameters, is

$$\sum_{i=1}^{P} H(x_i), \tag{5.11}$$

the sum of the pixel stack entropies. We wish to show that this corresponds to maximizing the joint probability of a set of images under a set of permuting transformations. From the assumed class conditional independence of the pixels, the probability of image I^j is the product of the pixel probabilities:

$$p(I^j) = \prod_{i=1}^{P} p_i(x_i^j). \tag{5.12}$$

Then the probability of a set $\mathcal{I}$ of N images is just

$$p(\mathcal{I}) = \prod_{j=1}^{N} \prod_{i=1}^{P} p_i(x_i^j). \tag{5.13}$$

Let N_i^0 be the number of 0-valued pixels in pixel stack i. Let N_i^1 be the number of

1-valued pixels in pixel stack i. Starting with the logarithm of 5.13, we have

$$\begin{aligned}
\log_2 p(\mathcal{I}) &= \sum_{j=1}^{N}\sum_{i=1}^{P} \log_2 p_i(x_i^j) && (5.14)\\
&= \sum_{i=1}^{P}\sum_{j=1}^{N} \log_2 p_i(x_i^j) && (5.15)\\
&= \sum_{i=1}^{P}\left(\sum_{j|x_i^j=0} \log_2 p_i(x_i^j) + \sum_{j|x_i^j=1} \log_2 p_i(x_i^j)\right) && (5.16)\\
&\approx \sum_{i=1}^{P}\left(N_i^0 \log_2 \frac{N_i^0}{N} + N_i^1 \log_2 \frac{N_i^1}{N}\right) && (5.17)\\
&= N\sum_{i=1}^{P}\left(\frac{N_i^0}{N} \log_2 \frac{N_i^0}{N} + \frac{N_i^1}{N} \log_2 \frac{N_i^1}{N}\right) && (5.18)\\
&= -N\sum_{i=1}^{P} H(x_i). && (5.19)
\end{aligned}$$

Hence, the logarithm of the joint probability of a set of images as defined above is approximately equal to a positive constant times the negative of the sum of the pixelwise entropies. Hence, minimizing the summed pixelwise entropies is approximately equivalent to maximizing the image probabilities under this model.

5.1.4 An important detail

Note that the probabilities $p_i(\cdot)$ are estimated in the fourth step (5.17) by the pixel stacks themselves, and hence change with the permutations of the image pixels. By the equivalence established above, we are maximizing the estimated probabilities of the pixels in each pixel stack. These probabilities are computed as

$$p_i(x_i^k) = \frac{\sum_j \delta(x_i^j - x_i^k)}{N}, \tag{5.20}$$

where $\delta(\cdot)$ is an indicator function. That is, the probability of a particular value occurring is estimated as its empirical frequency within the pixel stack.

According to this formula, the value of a pixel is being used in the computation of its own probability. This may seem inappropriate in an optimization setting, where such a method may lead to bias of the probabilities. In particular, one might choose to remove a particular pixel from its own probability calculation by considering a computation such as

$$Prob(x_i^k) = \frac{\sum_{j\neq k} \delta(x_i^j - x_i^k)}{N-1}, \tag{5.21}$$

where the sum is taken over all pixels except the one being evaluated. This is akin to a cross validation procedure for evaluating the probability of a pixel in its own pixel stack.

However, adopting such an estimate would substantially change the behavior of the congealing algorithm. In particular, using such an estimate, no image could ever "move

through" a position such that it had a unique pixel value in a particular location. That is, every foreground pixel in an image would be forced to stay within the limits defined by the support of the foreground pixels in the other images. Any image with a foreground pixel straying outside this support region would be assigned zero probability, which could not possibly be a step up in the probability function, and hence would not be chosen by the algorithm. The same restriction would apply to background pixels.

Alternatively, this "boxing in" of images in the congealing process could be resolved by introducing a noise process into the generative model to explain pixels that would have zero probability under the noise free model. We did not investigate this path, but other authors have used this type of approach in similar problems (Frey and Jojic, 1999a).

5.1.5 Smooth transformations

Until now in these analyses, the assumption has been made that the generative transformations were simple permutations of the pixels in an image. But congealing was initially defined in terms of smooth affine transformations that can shrink, enlarge, or mix pixels with each other. In fact, if we seek the maximum probability solution when allowing affine transforms, all of the images are shrunk to zero size, a state that gives all of the pixel stacks zero entropy. So, while the maximum likelihood interpretation gives us some intuition for what congealing does, it is not a perfect explanation.

However, the penalized minimum entropy solution of Equation 3.16 gives us a non-degenerate solution that is intuitively more appealing. Is there some principle which justifies the penalty on the affine parameter magnitudes introduced in Equation 3.16? We suggest a geometric interpretation of congealing which answers this question in the affirmative. It fits particularly well with the algorithm and suggests a new interpretation of what the algorithm is doing. It also suggests that the congealing algorithm is actually consistent with a maximum likelihood interpretation in a certain sense that will be described below.

Congealing as a measure of central tendency

The penalty on the affine transform parameter magnitudes can be justified in the following sense. Suppose that we had a set of images that shared a single latent image. Let the manifold of images associated with a particular latent image be $\mathcal{M}$. One interpretation of congealing is to find a measure of the central tendency or "pseudo-mean" of this set of images on the manifold, i.e. with respect to the affine family of transforms. But the usual definition of a mean,

$$\bar{I} = \frac{1}{N} \sum_{j=1}^{N} I^j, \tag{5.22}$$

requires that addition and scalar multiplication be defined on the space. The set of images $\mathcal{M}$ is not closed with respect to addition and scalar multiplication, so it is not immediately obvious how to define the notion of the mean on such a space. Is there a potential substitute for the common definition of mean? Many generalizations of the mean to Riemannian manifolds[2] have been developed (see for example (Grenander, 1963; Pennec, 1996)).

[2]Some terminology from differential geometry has been used in this section. A brief introduction to differential geometry can be found in (Schutz, 1980). A more extensive treatment can be found in (Spivak, 1999b).

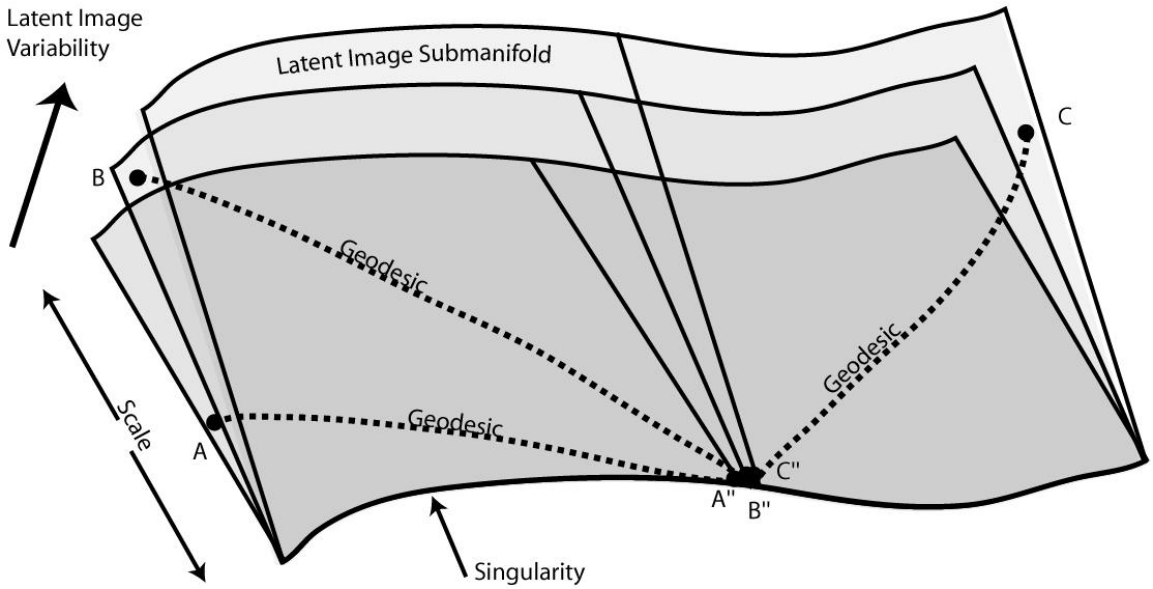


Figure 5-2: The result of congealing without the parameter magnitude penalty. Note the longer length (relative to Figure 5-3) of the geodesic paths from the original images (A, B, C) to the recovered latent image estimates (A'', B'', C'').

We suggest that a reasonable way to generalize the mean in this problem is to define it as

$$\bar{I} = \arg\min_{I \in \mathcal{M}} \sum_{j=1}^{N} D(I, I^j)^2, \tag{5.23}$$

that is, the point $\bar{I}$ in the latent image manifold $\mathcal{M}$ such that the sum of squared (minimum geodesic) distances to that point, from each of the points in the data sample, is minimized. That is, the pseudo-mean is defined to be the point on the manifold closest to all of the other points in terms of the manifold's metric.[3] Notice that this definition recovers the traditional mean for a Euclidean space.

Thus in this restricted case of a set of images with a fixed latent image, we can view congealing as an attempt to solve the above quadratic minimization problem, where the entropy minimization term would cause the images to move to the same point (image), and the coordinate penalty causes this point (image) to be approximately in the "middle" of the original set of points.

To solve this problem precisely as stated in Equation 5.23, one would need to define a metric for the Riemannian manifold, and to solve a variational problem to compute each distance in the sum. Iterating, one could then compute the true minimum point $\bar{I}$. Rather than attempting to solve this difficult optimization problem, we allow our minimization of the penalty on the transform coordinates to act as an approximation.

In general, random images from a set of characters do not share the same latent image. Figures 5-2 and 5-3 illustrate a *foliated* space of images, in which each submanifold (a leaf

[3]It is not always the case that such a point will be unique, or even that it will exist. For example, the mean of a set of points symmetrically distributed on the equator of a sphere could be viewed as either pole, since each pole achieves the minimum of Equation 5.23.

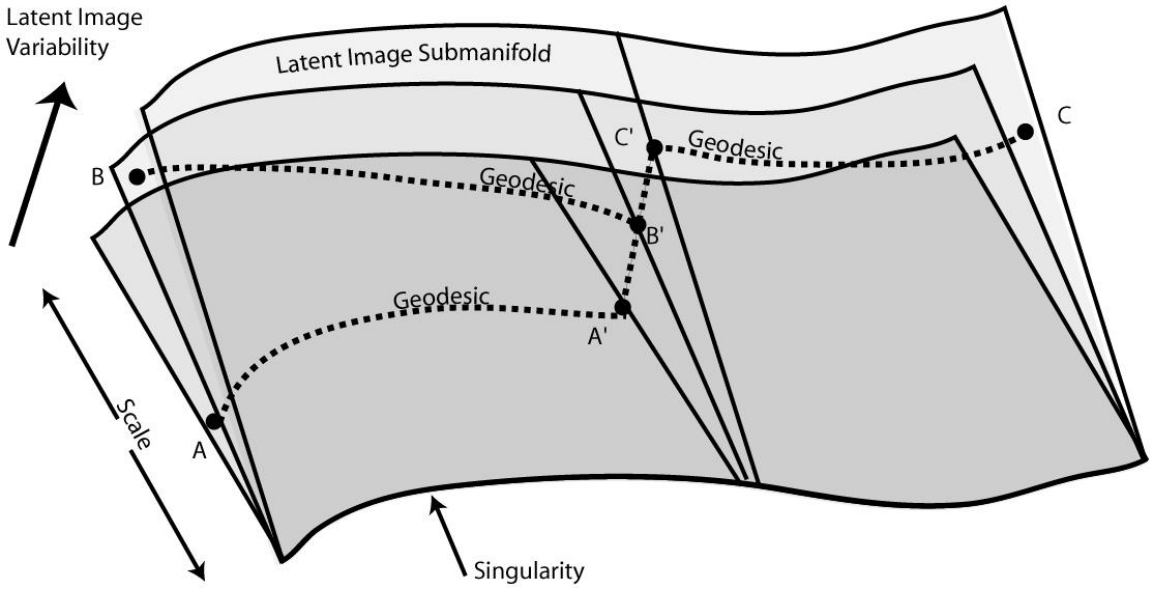


Figure 5-3: The result of congealing after having added the parameter magnitude penalty. Note the shorter length (relative to Figure 5-2) of the geodesic paths from the original images (A, B, C) to the recovered latent image estimates (A', B', C').

of the foliation) represents a manifold of images with a common latent image. When images are shrunk to have zero size, they converge to the same point irrespective of their latent image. This is shown as a singular region (which would be more properly shown as a single singular point) at the bottom of the figures.

Figure 5-2 illustrates the congealing of three images or points (A, B, and C), one on each latent image submanifold. In this illustration, no parameter magnitude penalty was included, and the images all shrank to size zero, converging at the singularity. In Figure 5-3, a parameter magnitude penalty is included, causing the images to converge at the pseudo-mean discussed above. Notice that the geodesic arc lengths are shorter in the constrained case than in the unconstrained case.

The pixel stack entropy term of the minimization can be seen as a constraint that the final congealed images are all "close to each other" in the sense that the perturbation of any image, while the others are held constant, will only increase their entropy. It is in this sense that congealing can be still be seen as a maximum likelihood method. Under this constraint, the parameter magnitude penalty causes the images to end up in the "middle" of the original set of images rather than at a singularity or other non-central point.

5.2 Comparison to traditional preprocessing methods

Congealing has been presented as one method for dealing with the affine variability of images. It is both a method for creating probabilistic models from data, and for aligning test examples to those models. It differs from traditional preprocessing techniques such as centroiding and deshearing, which are also designed to remove affine variability, in several important respects. One of the major differences between congealing and other methods of eliminating affine variability is that congealing is a function of a class model. That is, congealing (of a test sample) is done differently depending upon whether we are comparing

Preprocessing Method	Sum of Pixelwise Entropies									
	0	1	2	3	4	5	6	7	8	9
None	1160	687	1293	1316	1326	1410	1518	1215	1422	1373
Centroiding	1153	569	1216	1267	1265	1393	1444	1101	1387	1358
Scale norm.	894	542	963	810	829	861	763	732	926	791
Deshearing	1184	452	1212	1234	1240	1273	1364	953	1436	1182
Congealing	**558**	**354**	**834**	**733**	**721**	**873**	**780**	**700**	**923**	**842**

Table 5.1: Comparison via pixel entropies of preprocessing alignment algorithms. Congealing produces lower entropy image alignments for all ten digit sets relative to every other algorithm tested.

to a model of "2"s, or "3"s, or to something completely different, like faces. To a certain extent, then, congealing is not directly comparable to traditional preprocessing methods, since the former produces many possible outputs for a given input, while the latter algorithms produce a single output.

Nevertheless, there are certain comparisons which are informative. This section compares preprocessing methods and congealing using a variety of measures which provide insight into the differences between these approaches. First, congealing is analyzed relative to other algorithms as a method of joint alignment. Various properties of this alignment are discussed, including accuracy, sensitivity to noise in the images, and computational complexity. Classification experiments using a variety of preprocessing algorithms are then reported.

5.2.1 Preprocessing algorithms evaluated

The following preprocessing and alignment algorithms are compared to congealing.

Centroiding

In centroiding, the image centroid is first computed. The image is then shifted so that its centroid matches the center of the image. Centroiding requires that the image be segmented, and that a weight be given to each pixel value. It does not tend to work well for images other than binary images.

Scale normalization

The image is first centroided. Then it is scaled uniformly in x and y so that $max(width, height)$ has some fixed value.

Deshearing

There are many deshearing algorithms. The deshearing algorithm implemented in this work also starts with centroiding. Then the axes of least and greatest second moments are found. For whichever of these is closer to vertical, the image is sheared so that the moment is vertical.

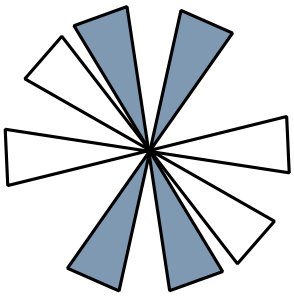


Figure 5-4: An example of two images that will not congeal to the global minimum of the correspondence fitness function. This problem can usually be alleviated in practice by congealing a large number of samples simultaneously.

Exhaustive search over alignments to a model

This technique is very similar to congealing, but assumes a fixed set of allowable transforms as in (Frey and Jojic, 1999a). While it is not implemented in these experiments, we make some qualitative comments about its advantages and disadvantages relative to congealing.

5.2.2 Precision of alignment

To measure the quality of the preprocessing method, we simply compute the mean entropy per pixel in the aligned image set. (The images are rescaled so that the average logarithm of the transform determinant is 1, insuring that the mean pixel entropy is not artificially increased or decreased.) Since congealing explicitly minimizes this quantity (plus the penalty term), it might be expected to have the lowest value. This does not necessarily imply that congealing will produce a superior classifier, but it does give insight into how close other algorithms come to minimizing the pixelwise entropies, or reducing the complexity of a model.

Ten samples of each digit were randomly chosen and run through each preprocessing algorithm. The results of these experiments are reported in Table 5.1 for each digit. *Not only does congealing produce lower entropy models in every case, but it avoids throwing out information by keeping track of the transformation that created the better image.*

5.2.3 Local minima

One problem with iterative methods such as congealing is that an image may fail to achieve the global minimum of the objective function. This can be caused by the so-called "zero-gradient" problem or the existence of local minima in the objective function. An example of the zero-gradient problem is shown in Figure 5-4. Note that the gray "X" and the white "X" do not overlap at all, despite the fact that their centroids are aligned. Thus a differential change in relative rotation of the two characters will not improve alignment according to the minimum entropy cost function. The figure illustrates a local minimum problem as well. It arises when one leg of an "X" overlaps a leg of the other "X", while the other legs do not overlap. In such a scenario, any perturbation of the rotation parameter would only increase the entropy. This scenario thus represents a local minimum of the entropy function.

The congealing process has a serendipitous advantage in that it often circumvents these two types of optimization problems. Because the alignment process is done over an ensemble

Number of Images	Percent trapped in local minimum									
	0	1	2	3	4	5	6	7	8	9
2	50	0	0	0	0	0	50	0	0	0
10	0	0	10	0	0	0	0	0	10	0
100	0	0	0	1	0	4	1	1	4	1
1000	0.6	0	0.9	0.7	0.3	1.1	0.3	0.0	0.0	0.6

Table 5.2: Percentages of images that do *not* reach global minimum of the probability function.

of images which has a data-dependent smoothing effect, these two issues arise infrequently. This can be understood by re-examining the average observed images of Figure 3-7, which show the relatively smooth "landscape" for hill-climbing in the congealing setting.

We note that for aligning a pair of images, a strategy of blurring one of the images is commonly used (Simard et al., 1993; Vasconcelos and Lippman, 1998). This can be thought of as a type of implicit congealing over horizontal and vertical translations, since convolving an image with a circular Gaussian distribution is equivalent to averaging a set of equivalent latent images that have been shifted horizontally and vertically according to a Gaussian distribution. Congealing improves upon this method by using the true distribution over transforms as a "convolution kernel". A limitation, however, is that enough images must be present to form a good approximation of the distribution.

Alignment experiment 1

To study the problems of local minima and zero gradients, the following experiments were performed. Training sets of four different sizes were congealed for each digit. The number of characters which failed to converge to the best alignment was evaluated. This judgment was made subjectively, based upon whether a human observer (the author) could find a better alignment of the character. The results are reported in Table 5.2. When a small number of examples are used in congealing, the lack of sufficient smoothing causes a greater number of local minima problems, as shown in the first two rows of the table.

Alignment experiment 2

Another phenomenon may occur when the observed data points are spread widely apart. In this case, congealing may produce multiple convergence centers rather than a single center. The following experiment was done to examine this issue more systematically. Starting with a single image of a "4" from the NIST database, we generated a sequence of 100 images rotated at uniform intervals from $-\frac{\theta}{2}$ to $\frac{\theta}{2}$. For $\theta < 68$ degrees, the images congealed to a unique position, but when $\theta > 68$ degrees, two "centers" emerged. This is due to a local minimum in the congealing process, as illustrated in Figure 5-5. Although this lack of convergence to a single global "center" is not ideal, it does not preclude us from using the resulting density model, which has relatively low entropy. That is, even in the presence of multiple convergent "centers" we are performing an important dimensionality reduction in the data by congealing. The key property is that a test character will be congealed to a predictable location for comparison with the model, without losing information about the character. Such multiple convergence centers were seen in the actual training data in the case of the class of eights. An example of each latent image is shown in Figures 5-5(c) and

a **b** **c** **d**

Figure 5-5: **a,b**. Two distinct centers of convergence for a set of rotated "4" images. The algorithm aligned the horizontal part of some fours with the vertical part of others and got stuck in this local minimum. However, since any test character which happens to be a four should rotate to one of these two positions, this can still make a good model for classification. **c,d**. Two centers for different "8" images.

(d).

5.2.4 Sensitivity to noise

Another important property of an alignment algorithm is its sensitivity to various types of noise in the input data. Congealing proved to be fairly robust to independent pixelwise noise, or *shot noise*. Experiments were done in which a fixed percentage of pixels, chosen randomly in the image, were replaced with either black or white pixels with equal probability. Figure 5-6 shows a "0" with 40% noise added and a "2" with 20% noise.

While the congealing algorithm worked without modification in many of the experiments, there were some changes in the algorithm's performance. For low levels of noise (0-10%), there was no discernible difference in the algorithm. For higher levels of noise and a small congealing set, the algorithm gets trapped in local minima. For example, with 40% noise and only 10 training examples, the zeroes were unable to congeal. That is, the algorithm converged to a local minimum after only very minor changes in the images. However, by raising the number of training examples to 20, the data set congealed with 100% convergence.

Even when most of the images converged to their correct positions, high levels of noise affected the number of examples converging properly. While at a 10% noise rate the twos converged as with no noise, at a 20% noise rate, 4% failed to converge, and at a 30% noise rate, only 74% of the twos in a set of 50 converged.

Except for the Exhaustive Search technique, the other methods for alignment depend upon segmenting the image automatically. This is a major drawback for such methods, since segmentation is a difficult and ill-posed problem itself. The Exhaustive Search technique will not suffer from the local minimum problems of congealing, but will not achieve as accurate an alignment as congealing when congealing converges.

Shot noise is just one of many types of variations that can be added to images. Rather than attempting a systematic study of a large number of other noise types, we offer examples of unusual characters in the training set that either successfully converged, or did not, in order to point out various properties of the congealing method and how it works in practice when spurious features have been added to images, or when certain features of characters have been occluded or omitted.

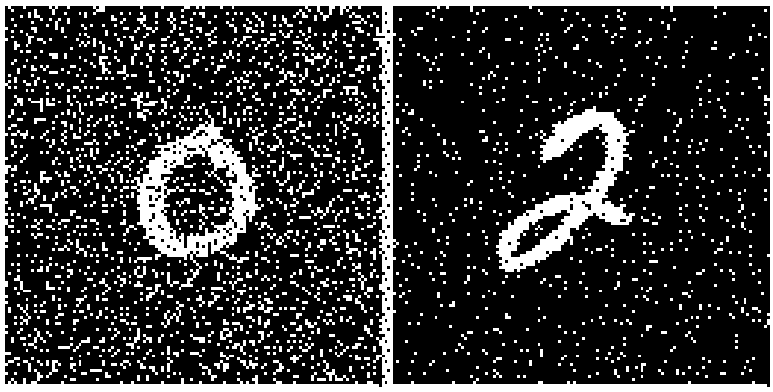


Figure 5-6: Congealing with noisy images. On the left is the image of a zero in which 40% of the pixels have been randomly assigned values of 0 or 1. On the right, a two in which 20% of the values have been reassigned.

Figure 5-7 shows a variety of examples of unusual characters that successfully congealed with their models. The figure shows the congealed versions, rather than the original observed images. The algorithm exhibits good robustness to spurious strokes (first, second, third and sixth frames) and to occlusions and omissions (fourth, fifth, and last frames). There were few cases in which congealing failed due to spurious strokes. Most failures seemed to be due to extreme transformations, however. Examples of such failures are shown in Figure 5-8.

5.2.5 Computational complexity

Centroiding, scale normalization, and deshearing can all be performed for one image in time $\mathcal{O}(P)$, where P is the number of pixels in an image. Each cycle of the algorithm for congealing a test character takes time $\mathcal{O}(PK)$, where K is the number of parameters of the transformation set. There is no analytic form for the number of steps to convergence, but in our digit experiments, the number averaged about 10 steps. If we take this number of steps as a constant, then the order of congealing with a one-parameter set of transforms is equivalent to the other algorithms, i.e. it is linear in the number of pixels. This complexity grows linearly with the number of parameters, while the number of steps to convergence, based upon informal observations, appears to remain approximately constant.

Similar methods (Frey and Jojic, 1999a) which use a fixed set of transformations are linear in the number of allowed transformations. This particular algorithm, Transform-Invariant Clustering, iterates using an Expectation-Maximization (EM) approach, so it also contains an unpredictable iterative time element. Each step of this algorithm is of order $\mathcal{O}(LP)$, where L is the number of transformations allowed. However, adding another *dimension* to the set of the transformations, e.g. rotation, increases the computation geometrically. Thus doubling the number of parameters in the set of transformations squares the complexity, whereas in congealing it only doubles the complexity. In this respect, then, congealing has a major advantage for representing complex deformations that are parameterized by more than a few parameters. Additional differences between the two algorithms, and additional details of Transform-Invariant Clustering, are discussed below in Section 5.3.3.

As a side note, the reader may have noticed that the number of affine parameters given in Algorithm 1 (7) is actually greater than the number of degrees of freedom in a two-dimensional affine transform (6). That is, the set of parameters overspecifies the transform.

Figure 5-7: Difficult cases that congealing handled well. These are the images after congealing has aligned them to the model as well as possible. The first two cases are "1"s with unusual or spurious strokes. The third frame shows a "2" with an unusual tail. The algorithm essentially ignored the tail and aligned the rest of the character. The fourth frame shows a "2" that has been clipped, but the algorithm puts the remainder of the character in correspondence with the model. The last three cases are all "3"s, with missing pixels, spurious stroke, and clipped bottom respectively.

The extra parameters have been used since there is no simple orthogonal coordinate system for affine transformations. Because of this fact, any parameterization tends to have regions in which there is no coordinate pointing close to the direction of the gradient. This can cause slow convergence of coordinate descent procedures. To alleviate this problem, we simply add an additional coordinate to the coordinate descent procedure by overparameterizing the manifold of transformations. This allows the algorithm to move faster in the appropriate direction (near the gradient) at certain points on the manifold, with only a small incremental cost to the complexity of each step of the algorithm.[4]

5.3 Related work in modeling spatial transformations

There have been numerous other efforts to address shape variability in images. These include (Amit et al., 1991; Frey and Jojic, 1999a; Vetter et al., 1997). This section discusses prior work in which the latent image-transform factorization was used to model shape variability in sets of images.

5.3.1 Active appearance models

In (Cootes et al., 2001), the authors describe a system which models observed images as a combination of "shape" variations and "appearance" variations. These terms correspond to the transform and latent image components of the model we use. While these models have a number of interesting potential applications in face recognition, tracking, and other vision tasks, the models are built upon manually identified correspondences. We shall focus our attention on models that can be learned automatically.

5.3.2 Vectorization

Jones and Poggio have presented models for images that consist of separate "texture" and "shape" components (Jones and Poggio, 1995). These parts correspond to the latent image and transform components used in this thesis. The shape components in these models are linear models. That is, deformation vector fields are combined linearly to produce variations

[4]For an illustrated discussion of coordinate system degeneracies and singularities, the reader may wish to consult (Misner et al., 1973).

in the shape of latent images. Separating an observed image into its shape and texture parts is termed vectorization by the authors.

These models were initially developed for images of faces by manually identifying correspondences among all of the prototype faces. The "flows" that put these faces in correspondence then provide the model of deformation with which one can do synthesis, analysis, and other tasks.

In (Vetter et al., 1997), the authors introduce a technique for "bootstrapping" these correspondences by iteratively estimating more and more refined models of texture and shape with the aid of an optical flow algorithm. The joint nature of this optimization makes it quite similar to congealing. There are several important differences as well. Unlike in congealing, the deformation model is unconstrained. That is, virtually any flow field can arise from the algorithm. This makes the procedure both more flexible, and more difficult to get working correctly. Another difference is that the authors do not explicitly define an optimization criterion other than the subjective one of good visual alignment. Nevertheless, the algorithm is ultimately very similar to one which minimizes the entropies of the final latent images, like congealing.

5.3.3 Transform invariant clustering

The work of Jojic and Frey (Frey and Jojic, 1999a; Frey and Jojic, 1999b) has the greatest similarity to congealing. In these papers and in more recent work, the authors use the generative latent image-transform model of image production. They produce models of latent images by simultaneously maximizing the posterior likelihood of a set of latent images under a fixed set of transformations.

The authors use the EM algorithm to maximize the likelihood of the latent images simultaneously under a set of models. One key difference with our own work is that the authors entertain a finite set of transformations rather than a continuous set. This allows the authors to perform a full Bayesian analysis at each step, calculating the likelihood of an observed image under a particular model by integrating over all possible latent images and transforms. Hence, they do not need to resort to using the "peakiness assumption" of Equation 3.11.

However, with a fixed set of transforms, the number of modes of spatial deformations that can be modeled is limited. The complexity of their algorithm is linear in the number of possible transformations, whereas congealing is linear in the number of *types of parameters*. This means congealing can be used with much larger sets of transforms, and with unbounded resolution within each parameter. This means that congealing can achieve potentially more accurate alignments.

Another nice feature of these papers is that the authors incorporate an explicit noise model, which has not been done for congealing. This effectively allows the authors to weigh certain parts of the image more heavily than others in the alignment procedure. Incorporating such a noise model into the congealing procedure would be a straightforward improvement for future work.

5.4 Generalizations

Congealing has been defined so far as a fairly specific algorithm: the alignment through affine transformation of binary images by minimization of an entropy-like objective function. Congealing can easily be extended to non-binary images, to non-affine transformations, and

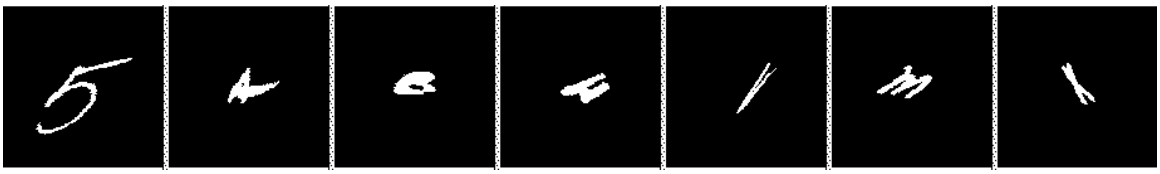


Figure 5-8: Cases that congealing got wrong. The figure shows the latent image estimates, not the original characters. The actual characters are, from left to right, "5", "A", "B", "H", "V", "m", and "x". With the possible exception of the "A", none of these are in the maximum probability alignment with the model.

even to non-images. This section presents a variety of extensions to congealing, gives some experimental results, and discusses the essential features that make congealing different than a generic joint optimization procedure.

5.4.1 Multi-valued data

Perhaps the most obvious extension of congealing is to non-binary images.

n-ary images

We start by considering a simple extension to images with n-ary valued pixels, which we shall call n-ary images for short. Such images arise whenever all of the pixels in an image are assigned a discrete label from a finite set of n labels. For example, medical images like magnetic resonance (MR) images are often segmented by tissue type. An example of such a segmented image is shown in Figure 5-9**a**.

Since the binary entropy function is just a special case of the general discrete entropy function

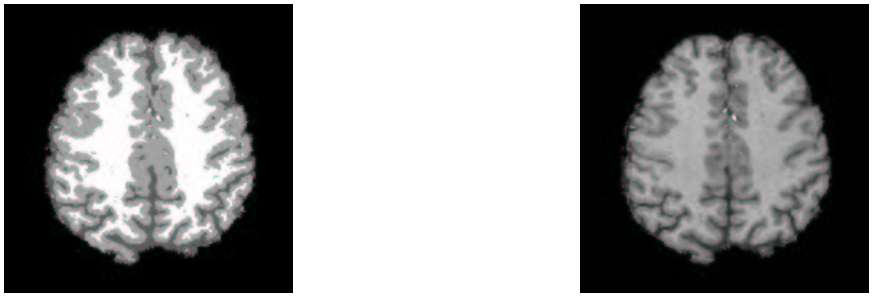
$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x), \tag{5.24}$$

no change in the objective function is required. However, one implementation detail must be addressed. The distribution of values in each pixel stack must be maintained by the algorithm so that it may be determined whether an adjustment of an image raises or lowers the objective function. Hence, a histogram of values must be maintained for each pixel stack, rather than just a mean value, as with binary images.[5] As n gets large, this histogram maintenance can represent a substantial storage requirement, especially when congealing 3-D volumes, as discussed below.

Gray valued images

Congealing can also be extended to gray-valued images. By gray-valued images, we mean images that represent an approximation to a continuous value at each pixel. In particular, the values in each pixel are not merely labels, but have a quantitative meaning. This is

[5] Actually, one bin in the histogram need not be maintained since it can be recovered from the other bins using the sum constraint. However, other than for the binary case, simplicity of coding and diminishing returns usually dictates that we ignore this possible savings and simply store the entire histogram at each pixel location.



a **b**

Figure 5-9: **a**. A segmented brain image. The four distinct image values represent background (black), cerebrospinal fluid (dark gray), gray matter (light gray), and white matter (white). **b**. The original magnetic resonance brain image from which the segmented image was derived. The pixels in this image represent the scalar strength of the magnetic resonance signal at each pixel.

a significantly different case for generalization than the n-ary image case, since it has an impact on how we compute the entropy of a pixel stack.

Take again the case of MR images, but this time raw (rather than segmented) images, as shown in Figure 5-9b. Here, the value of a pixel is a function of some physical parameter, such as the proton density of the brain tissue. It would be nice to generalize congealing to use the differential entropy function in this case:

$$h(X) = -\int_{-\infty}^{\infty} p(x) \log p(x) dx. \tag{5.25}$$

However, since the density in general may have no parametric form, we must compute the entropy using a discrete approximation. We may do this by using a histogram representation of $p(X)$. Unfortunately, the bin size in such a histogram representation has a significant impact on the entropy estimates. To mitigate this effect, we perform a kernel estimate of each pixel stack distribution at each step in the algorithm and then use a fine-grained histogram entropy estimate of the resulting approximate distribution's entropy.

A simple test was performed to evaluate the basic plausibility of gray-scale congealing. A single gray-scale image was perturbed using a set of random affine transforms, as shown in Figure 5-10. The images were then congealed. The goal was to recover the original latent image.

Unlike the case of binary digits, the background is not constant, and more to the point, is non-zero. This means that the image set has a natural tendency to want to move away from the center, since the replacement of image pixels by constant off-image pixels tends to reduce the entropy of the data set. To combat this problem, a scalar Gaussian weight mask was introduced to emphasize the central pixels in the image more than the border pixels. The Gaussian mask was adjusted to have a standard deviation equal to about one third of the total image width. This gave the border pixels approximately zero weight while allowing a large group of pixels near the center of the image to have a significant impact on the congealing. With such a modification, the gray-scale images congealed as shown in Figure 5-10b.

Figure 5-10: Gray scale congealing. On the left are images which have been randomly perturbed in an affine manner. On the right are the congealed versions of the images. The fact that all latent images are the same makes this congealing task artificially easy.

Limitations of congealing

This problem is artificially simple in that the latent images are all identical. One would like to apply congealing methods to distributions with variable latent images. Here we see one of the major limitations of congealing. Congealing works by assuming that, on average, pixels of a particular value should be put in correspondence as often as possible. However, in a set of images of faces, if one face in the distribution is lit from the left, and another is lit from the right, an algorithm (like congealing) that matches pixel brightnesses will not put the images in correspondence correctly.

To alleviate this problem, the idea of forming edge images before congealing, and then congealing these edge images, was attempted. We computed edge images for each image in a standard face database. We then ran the congealing algorithm on the set of edge images. While the entropy was substantially reduced, the resulting model was not of sufficient quality to use in a classifier. Congealing such a set of images thus remains a goal for future research. Using similarity template representations of images (Stauffer and Grimson, 2001), and congealing these (Stauffer et al., 2002), appears to be a promising new direction for solving this problem.

5.4.2 Other transformations

Another way to generalize congealing is to consider sets of transformations other than the affine transformations considered previously. Other sets of transformations, such as vector fields that are linear combinations of sinusoidally varying basis fields have been used in other applications, such as medical image registration (Christensen, 1999).

The congealing software was modified to accept as a transform basis any set of transformations that could be represented as vector fields over pixel space. The transforms were

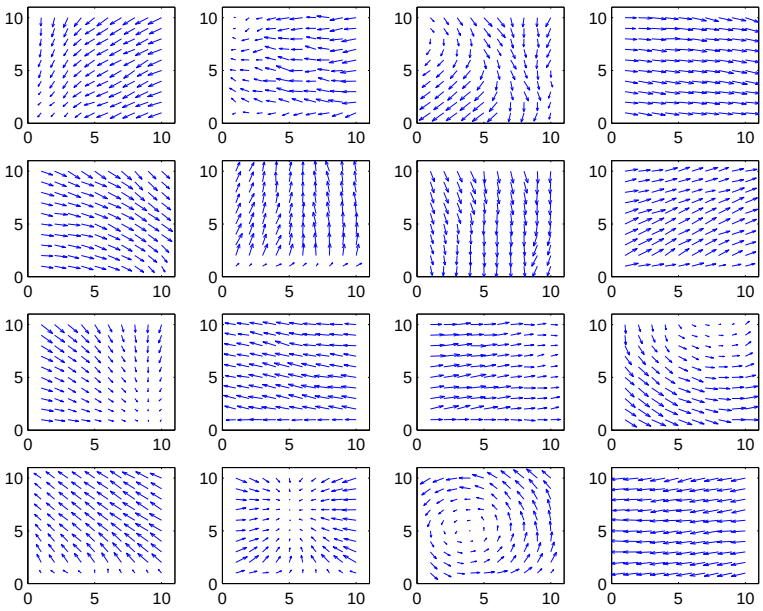


Figure 5-11: Means of vector field clusters derived from video sequences. These vector fields were used as a basis for congealing in place of affine transforms.

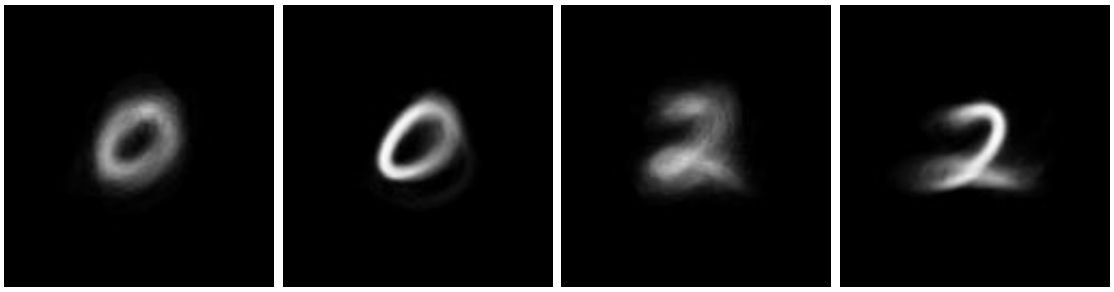
then applied by iterative application to the observed image in order to congeal to latent images. This process is substantially more computationally expensive than congealing with affine transforms, since a sequence of arbitrary transformations does not form a group (as in the affine case). The lack of group structure means that every change in a transform parameter required the reapplication of a sequence of transformations in order to achieve the final result. This introduced another slowdown, linear in the number of coordinate steps per basis vector field, into the algorithm. Nevertheless, several experiments were completed.

Perhaps the most interesting experiment involved using a set of learned transformation bases. As described in (Black et al., 1997b), a set of statistically common motion flow fields can be learned from video data. The set of learned vector fields shown in Figure 5-11, which we learned simply by clustering a large set of optical flow fields taken from video data, was adopted as a transform basis for congealing.

We congealed a set of zeroes and a set of twos using this basis. The starting and ending final images for this process are shown in Figure 5-12. Comparing this figure to the final mean images obtained from congealing with affine transforms (Figure 3-7), we can see that the entropy was not reduced as much. However, the algorithm still reduced the entropy further than did the common centroiding algorithm. The zeroes were reduced to a total pixel entropy of 998 (vs. 1153 for centroiding), and the twos were reduced to a total pixel entropy of 1115 (vs. 1216 for centroiding). Thus, with a set of transforms *learned from data*, we were able to outperform, according to this measure, a standard alignment technique.

5.4.3 Non-images

Another way to extend congealing is to apply it to data types other than pixels in images. There is a large number of potential application areas, a few of which we explore here. We



a **b** **c** **d**

Figure 5-12: Mean images during the congealing process using a learned vector field basis. The reader may wish to compare these images to those in Figure 3-7. **a.** The initial mean image for the set of zeroes. **b.** The final mean image for the set of zeroes. **c.** The initial mean image for the set of twos. **d.** The final mean image for the set of twos.

can characterize data sources as discrete valued, scalar valued, or vector valued functions over R^d, where d could take on values of 1, 2, 3, or more. Here we shall consider congealing the following data types:

- Scalar valued functions in one dimension (electroencephalograms).
- Vector valued functions of one dimension (on-line handwriting).
- Binary, n-ary, and continuous valued functions of two dimensions (as already discussed).
- Discrete valued functions of three dimensions (segmented medical imaging volumes).
- Scalar valued functions of four dimensions (image similarity templates).

One-dimensional signals

There are a variety of one-dimensional signals that are appropriate for congealing. Whenever we have a collection of signals that have undergone a one-dimensional transformation in addition to additive noise, we can apply the technique. Time sequences of a process that has variable duration and starting times are an example of such a process.

Event Related Potentials (ERPs) are a type of electroencephalogram signal that is well-suited to congealing. Methods similar to congealing, but that use correlation rather than an entropy measure, and that do not scale the data in time, have been developed (Woody, 1967). We propose application of congealing to these signals as a direction for future research.

Another type of one-dimensional data that is amenable to congealing is on-line handwriting.[6] By on-line handwriting, we mean handwriting data that is collected as an ordered sequence of two-dimensional points from a pointing device such as a mouse or electronic pen, as illustrated in Figure 5-13. That is, as a particular letter is drawn, the position of the pointer is sampled as regular time intervals. Hence the signal is a sequence of two-dimensional points, parameterized by time.

[6]The experiments with on-line handwriting described here were done in collaboration with Nick Matsakis in the MIT Artificial Intelligence Laboratory.

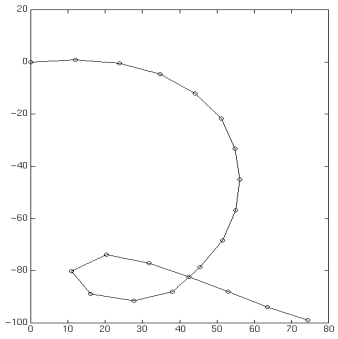


Figure 5-13: A handwritten two drawn with a pointing device like a mouse.

A common approach to modeling such on-line handwriting characters is to form linear Gaussian models of corresponding coordinate positions for each class of characters (see, e.g. (Matsakis, 2000)). The upper left image in Figure 5-14 shows the mean position of the n-th sample point in a set of handwritten "3"s. It is the character obtained by simply averaging the set of coordinates of the first points from each "3", the set of second points, and so on, and then connecting the points in order. The principal components of deformation away from this mean (the modes explaining the greatest variation in the training set) are shown in the other eight components by adding these deformations to the mean "3".

Such models are useful for recognition and synthesis of on-line handwriting data, and have been used for recognizers such as the handheld device character recognition systems. However, these models lose some fidelity due to the misalignment of features in different versions of the same character. For example, if the central vertex of a "3" is the tenth sampled point in one character but the eighth sampled point in another character, then a linear model of character coordinates will be corrupted, since these points are out of correspondence. This is similar to the difficulty in producing accurate image models from non-registered images, as in (Turk and Pentland, 1991), when image pixel brightness values from non-corresponding points are assumed to be in correspondence.

Congealing can reduce the complexity and improve the accuracy of these on-line handwriting models by putting sampled points into correspondence without changing their coordinates. This is done by applying shifts and scalings to the *time parameter* with which these curves are parameterized. For example, the vertex point of one "3" may not align with the vertex point of another "3" simply because they occurred at different latencies after the character was started. By reparameterizing the character with a "warped time" parameter to get better alignment *without changing the shape of the character at all*, we can reduce the number of significant principal components in the model. This is done by smoothly shifting and scaling the time variable for each character curve individually, a type of one-dimensional congealing. This can be thought of as reparameterizing the curve without changing the points that make it up. If we can put corresponding features of a digit class, like the vertex in the middle of the "3" character, into better correspondence with each other, then we can produce a lower entropy model of the characters. We refer to this process as congealing because it is a process of joint alignment achieved through continuous, smooth, and independent warping of continuous signals.

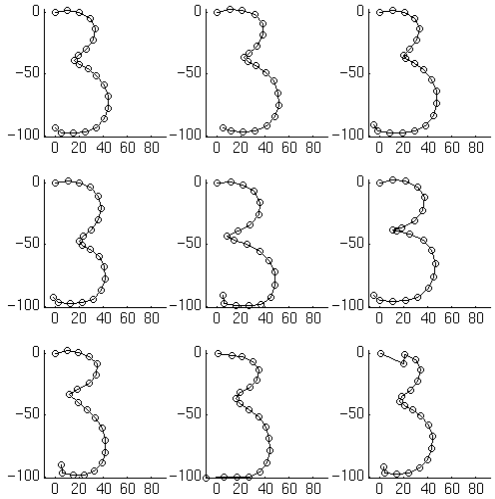


Figure 5-14: Principal modes of variation of a set of on-line handwritten "3"s. Each image shows the mean "3" plus a distortion caused by the nth eigenvector.

To illustrate the complexity reduction achieved by this time-based congealing, we plot the eigenvalues of the singular value decomposition for a set of "3" characters, both before and after congealing. The plot is shown in Figure 5-15. The top curve shows the eigenvalues before congealing, and the bottom curve after congealing. On average, the eigenvalues were reduced by approximately 50%, simply by reparameterizing the on-line character curves. The reduced eigenvalues indicate that the model has less variance in coordinate space than it did before the congealing. We emphasize again that this is done *without changing the shape or appearance of the character.* This has many potential applications in improving on-line character recognition.

Three-dimensional signals

We have applied congealing to the problem of aligning segmented three-dimensional medical image volumes as well. This has great appeal for the formation of atlases, since one may find a notion of the pseudo-mean, as defined in Section 5.1.5, of a set of brains under a set of transformations. Finding such a natural notion of the mean would allow construction of a brain atlas that was not biased toward any particular individual's brain.

A set of 30 such neonatal brains were aligned by congealing. The neonatal brain volume data were generously provided by Dr. Petra Hüppi. The acquisition protocol for these MR studies is detailed in (Hüppi et al., 1998). In order to reduce the number of steps needed by the algorithm in the full resolution scenario, the brains were successively aligned at four different resolutions, so that only the last part of the alignment needed to take place at the full resolution.

Figure 5-16 shows two unaligned neonatal brains (segmented as binary volumes) as series of slices. Figure 5-17 shows two brains after alignment using congealing. Each subfigure contains nine coronal (face-parallel) slices through a brain volume, ordered sequentially

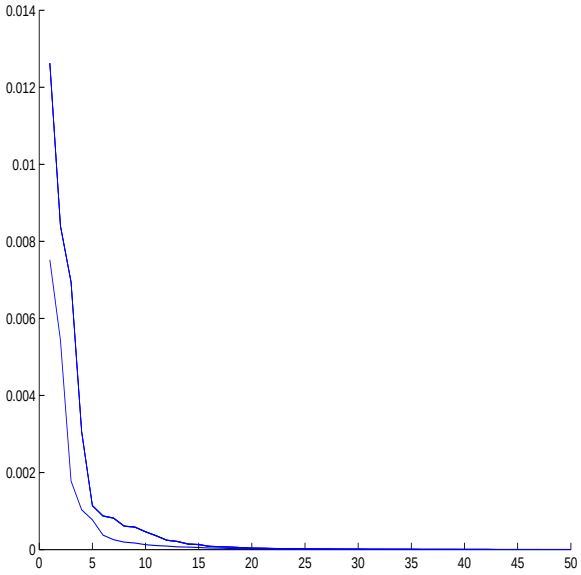


Figure 5-15: Reduction in variability of on-line handwriting data.

from left to right and top to bottom (as text on a page). Consider the unaligned brains in Figure 5-16. Notice that while there is substantial tissue in the second image (top middle) of the top figure, no tissue appears in the bottom figure until the third image. Also, in the top figure, the descending temporal lobes are clearly visible in the middle image, while no such structures are visible at this slice in the lower figure until two slices later. These and a variety of other differences indicate that the volumes are not well aligned.

On the other hand, the features of the two brains shown in Figure 5-17 match well. The brains have been transformed to match as well as possible. While there are still minor differences between corresponding images, these are not differences that can be eliminated by the simple set of transformations used.

In addition to being an exciting new possibility for defining brain atlases, the minimum entropy criterion has been proposed as a general tool for evaluating other alignment algorithms and the quality of brain atlases, as reported in (Warfield et al., 2001).

Four-dimensional signals

Finally, recent models of images based on pixel co-occurrences have been found to be effective in representing image structure for recognition, segmentation, and other common vision tasks (Stauffer and Grimson, 2001). A *similarity template* as described in this work is a representation of similarities between every pair of pixels in an image. Thus a single entry in a similarity template is indexed by a quadruple (x_1, y_1, x_2, y_2), the coordinates of the two image pixels whose similarity is described. Such models can be made more robust with respect to spatial deformations in the original images via congealing, as described in (Stauffer et al., 2002). This represents an interesting case in which each example is a four-dimensional array computed from an image, but in which the spatial transformations to align these four-dimensional objects are performed in the original (two-dimensional) image space. This suggests the use of congealing in a wide range of scenarios in which the goal is

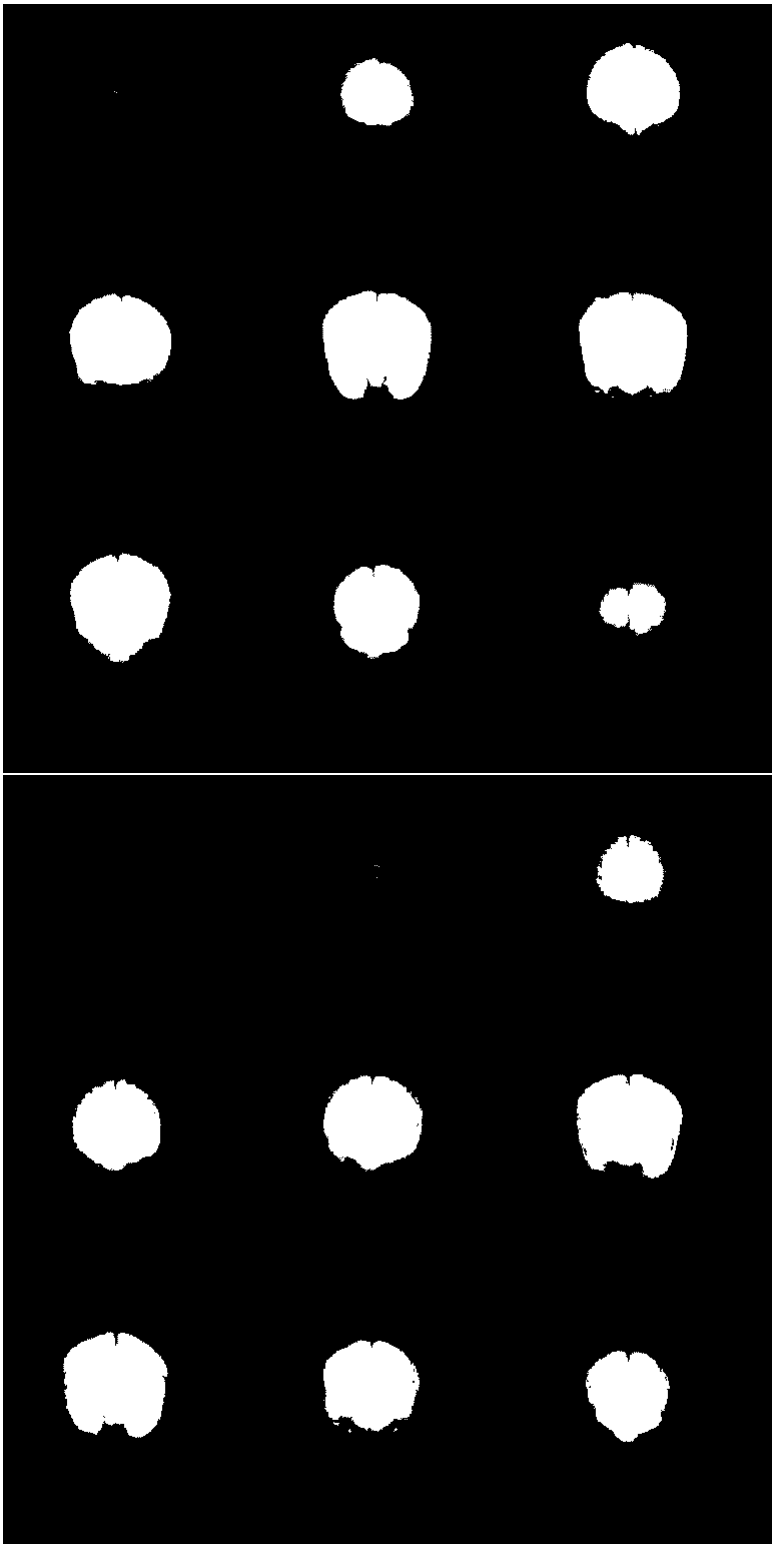


Figure 5-16: Two of the thirty unaligned brains. Notice the differences between corresponding slices in the upper and lower figures.

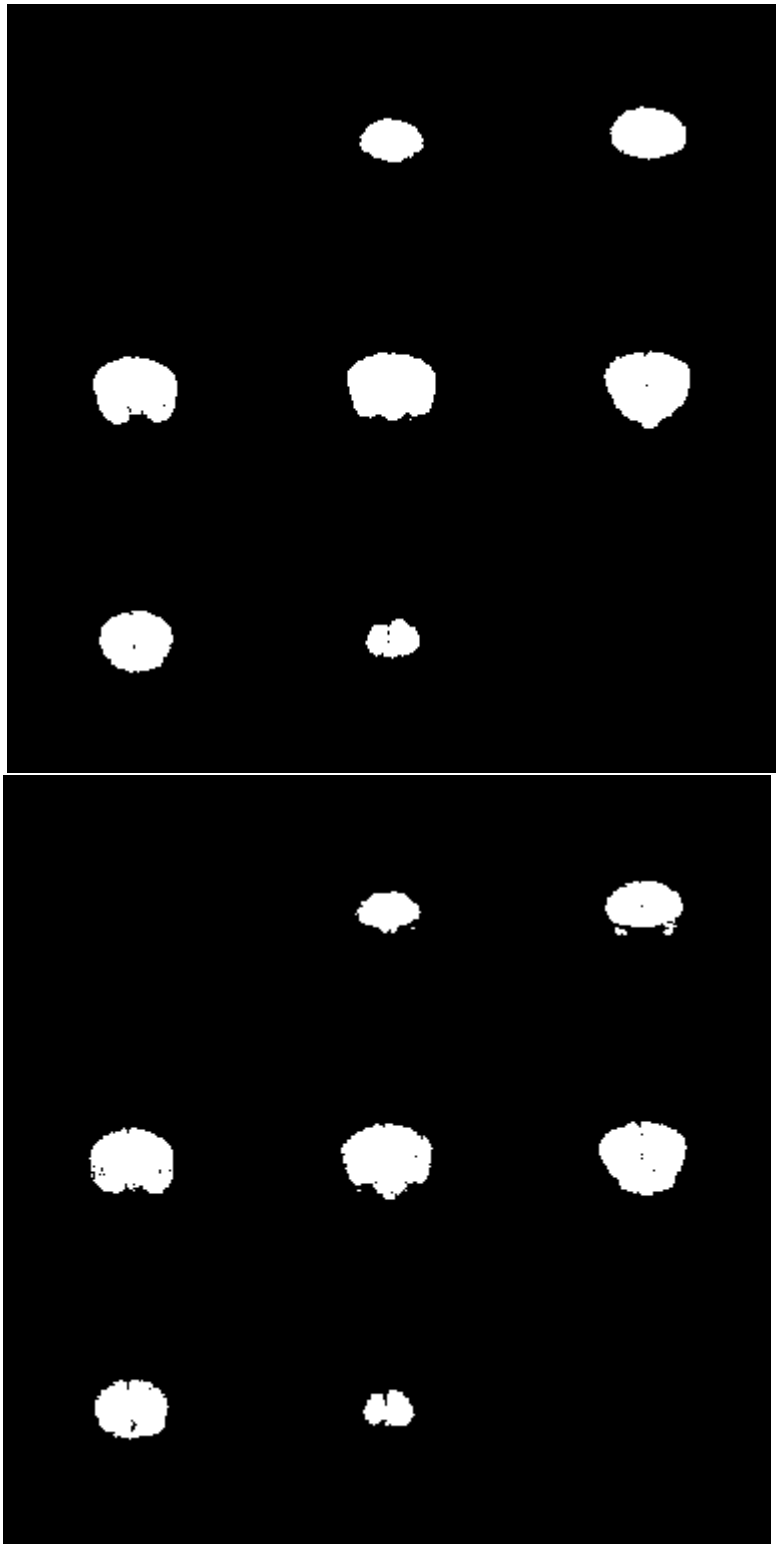


Figure 5-17: Two of the thirty aligned brains aligned with 3-D congealing. Notice the similar structures in corresponding slices between the upper and lower figures.

the reduction of the entropy of feature measurements, but the transformations are done on the original image data.

5.4.4 Conclusions

Congealing is differentiated from other joint optimization procedures such as that described in (Frey and Jojic, 1999a) in that it performs a joint coordinate descent, rather than performing an exhaustive search over possible transforms for each image or other signal. It can be applied to any signal over a field where there is a notion of local movement or deformation. William Wells and John Fisher (personal communication) have suggested that its applicability could be characterized by an *autoentropy* function of a typical latent image. The autoentropy function can be defined as the mutual information of an image with itself under varying amounts of translation, rotation, etc., very much like an autocorrelation function. A wider peak in the the autoentropy function would imply a greater capture range for the algorithm with a small number of examples. Such an analysis is left for future work.

This concludes the part of the thesis on modeling spatial changes in images. Next, we consider the modeling of image color changes.

Chapter 6

Color Flow Fields

6.1 Introduction

This chapter introduces the second major topic area in this thesis, modeling of image variations due to lighting change and certain non-linear camera effects.[1] While the topics of spatial variability and variability due to lighting are typically treated very differently in machine vision, it turns out that there is a great deal they have in common. Many of the ideas presented in the first chapters will be relevant for this seemingly different topic area.

After exploring a representation and model of color change in this chapter, we provide a unifying framework for color changes and spatial changes in Chapter 7. The material in this chapter has appeared in similar forms in the (Miller and Tieu, 2001a) and (Miller and Tieu, 2001b). (Since the topics of this chapter and Chapter 7 are so different from previous chapters, and to maintain consistency with previous publications, some slight notation changes are made relative to previous chapters. For example, image indices appear from here forward as subscripts rather than superscripts. However, these are not so signficant that they should cause trouble for the reader.)

The number of possible images of an object or scene, even when taken from a single viewpoint with a fixed camera, is very large. Light sources, shadows, camera aperture, exposure time, transducer non-linearities, and camera processing (such as auto-gain-control and color balancing) can all affect the final image of a scene (Horn, 1986). Humans seem to have no trouble at all compensating for these effects when they occur in small or moderate amounts. However, these effects have a significant impact on the digital images obtained with cameras and hence on image processing algorithms, often hampering or eliminating our ability to produce reliable recognition algorithms.

Addressing the variability of images due to these *photic parameters* has been an important problem in machine vision. We distinguish photic parameters from *geometric parameters*, such as camera orientation or blurring, that affect which parts of the scene a particular pixel represents. We also note that photic parameters are more general than "lighting parameters" that would typically only refer to light sources and shadowing. We include in photic parameters anything which affects the final RGB values in an image given that the geometric parameters and the objects in the scene have been fixed.

In this chapter, we develop a statistical linear model of *color change space*, by observing

[1] Much of the work on color flow fields was done in collaboration with Kinh Tieu in the Artificial Intelligence Laboratory at MIT.

how the colors in static images change under naturally occurring lighting changes. This model describes how colors change *jointly* under typical (statistically common) photic parameter changes. Such a model can be used for a number of tasks, including synthesis of images of new objects under different lighting conditions, image matching, and shadow detection. Results for each of these tasks will be reported.

Several aspects of our model merit discussion. First, it is obtained from video data in a completely unsupervised fashion. The model uses no prior knowledge of lighting conditions, surface reflectances, or other parameters during data collection and modeling. It also has no built-in knowledge of the physics of image acquisition or "typical" image color changes, such as brightness changes. It is completely data driven. Second, it is a single global model. That is, it does not need to be re-estimated for new objects or scenes. While it may not apply to all scenes equally well, it is a model of frequently occurring joint color changes, which is meant to apply to all scenes. Third, while our model is linear in *color change space*, each joint color change that we model (a 3-D vector field) is completely arbitrary, and is not itself restricted to being linear. That is, we define a linear space whose basis elements are *vector fields* that represent nonlinear color changes. This gives us great modeling power, while capacity is controlled through the number of basis fields allowed.

After discussing previous work in Section 4.2, we describe the form of the statistical model and how it is obtained from observations in Section 4.3. In Section 4.4, we show how our color change model and a single observed image can be used to generate a large family of related images. We also give an efficient procedure for finding the best fit of the model to the difference between two images, allowing us to determine how much of the difference between the images can be explained by typical joint color changes. In Section 4.5 we give preliminary results for image matching (object recognition) and shadow detection.

6.2 Previous work

The color constancy literature contains a large body of work for estimating surface reflectances and various photic parameters from images. A common approach is to use linear models of reflectance and illuminant spectra (often, the illuminant matrix absorbs an assumed linear camera transfer function) (Marimont and Wandell, 1992). A surface reflectance (as a function of wavelength λ) can be written as $S(\lambda) \approx \sum_{i=1}^{n} \alpha_i S_i(\lambda)$. Similarly, illuminants can be represented with a fixed basis as $E(\lambda) \approx \sum_{i=1}^{m} \beta_i E_i(\lambda)$. The basis functions for these models can be estimated, for example, by performing PCA on color measurements with known surface reflectances or illuminants. Given a large enough set of camera responses or RGB values, the surface reflectance coefficients can be recovered by solving a set of linear equations if the illuminant is known, again assuming no other non-linearities in the image formation.

A variety of algorithms have been developed to estimate the illuminant from a single image. This can be done if some part of the image has a known surface reflectance. Making strong assumptions about the distribution of reflectances in a typical image leads to two simple methods. Gray world algorithms (Buchsbaum, 1980) assume that the average reflectance of all the surfaces in a scene is gray. White world algorithms (McCann et al., 1977) assume that the brightest pixel corresponds to a scene point with maximal reflectance. Brainard and Freeman attacked this problem probabilistically (Brainard and Freeman, 1997) by defining prior distributions on particular illuminants and surfaces types. They then used Bayes rule and a new estimator (the *maximum local mass* estimator) to

choose a single best estimate of the illuminant and surface.

Some researchers have redefined the problem to one of finding the relative illuminant (a mapping of colors under an unknown illuminant to a canonical one). Color gamut mapping (Forsyth, 1990) models an illuminant using a "canonical gamut" or convex hull of all achievable image RGB values under the illuminant. Each pixel in an image under an unknown illuminant may require a separate mapping to move it within the "canonical gamut". Since each such mapping defines a convex hull, the intersection of all such hulls may provide enough constraints to specify a "best" mapping. (Cardei et al., 1997) trained a multi-layer neural network using back-propagation to estimate the parameters of a linear color mapping. The method was shown to outperform simpler methods such as gray/white world algorithms when trained and tested on artificially generated scenes from a database of surface reflectances and illuminants. A third approach by (Lenz and Meer, 1997) works in the log color spectra space. In this space, the effect of a relative illuminant is a set of constant shifts in the scalar coefficients of linear models for the image colors and illuminant. The shifts are computed as differences between the modes of the distribution of coefficients of randomly selected pixels of some set of representative colors.

Note that in these approaches, illumination is assumed to be constant across the image plane. The mapping of RGB values from an unknown illuminant to a canonical one is assumed to be linear in color space. A diagonal linear operator is commonly used to adjust each of the R, G, and B channels independently. Not surprisingly, the gray world and white world assumptions are often violated. Moreover, a purely linear mapping will not adequately model non-linear variations such as camera auto-gain-control.

(Belhumeur and Kriegman, 1998) bypasses the need to predict specific scene properties by proving statements about the sets of all images of a particular object as certain conditions change. They show that the set of images of a gray Lambertian convex object under all lighting conditions form a convex cone.[2] Only three non-degenerate samples from this cone are required to generate the set of images from this space. Nowhere in this process do they need to explicitly calculate surface angles or reflectances.

One aspect of this approach that we hoped to improve upon was the need to use several examples (in this case, three) to apply the geometry of the analysis to a particular scene. That is, we wanted a model which, based upon a single image, could make useful predictions about other images of the same scene. This work is in the same spirit, although we use a statistical method rather than a geometric one.

6.3 Color flows

In the following, let $\mathcal{C} = \{(r, g, b)^T \in \mathbb{R}^3 : 0 \leq r \leq 255, 0 \leq g \leq 255, 0 \leq b \leq 255\}$ be the set of all possible observable image color 3-vectors. Let the vector-valued color of an image pixel p be denoted by $\mathbf{c}(p) \in \mathcal{C}$.

Suppose we are given two P-pixel RGB color images I_1 and I_2 of the same scene taken under two different sets of photic parameters θ_1 and θ_2 (the images are registered). Each pair of corresponding image pixels p_1^k and $p_2^k, 1 \leq k \leq P$, in the two images represents a mapping $\mathbf{c}(p_1^k) \mapsto \mathbf{c}(p_2^k)$. That is, it tells us how a particular pixel's color changed from

[2]This result depends upon the important assumption that the camera, including the transducers, the aperture, and the lens introduce no non-linearities into the system. The authors' results on color images also do not address the issue of metamers, and assume that light is composed of only the wavelengths red, green, and blue.

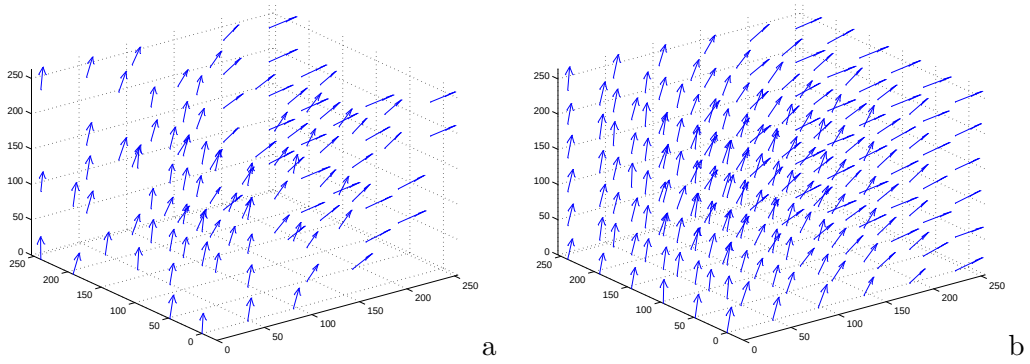


Figure 6-1: Color flows as vector fields in color space. **a.** A (synthetic) partially observed color flow obtained from a pair of images under different lighting conditions. An arrow appears at each point in color space where a color was observed. **b.** The interpolated completion of this color flow.

image I_1 to image I_2. This single-color mapping is conveniently represented simply by the vector difference between the two pixel colors:

$$\mathbf{d}(p_1^k, p_2^k) = \mathbf{c}(p_2^k) - \mathbf{c}(p_1^k). \tag{6.1}$$

By computing P of these vector differences (one for each pair of pixels) and placing each vector difference at the point $\mathbf{c}(p_1^k)$ in the color space $\mathcal{C}$, we have created a vector field that is defined at all points in $\mathcal{C}$ for which there are colors in image I_1.

That is, we are defining a vector field Φ' over $\mathcal{C}$ via

$$\Phi'(\mathbf{c}(p_1^k)) = \mathbf{d}(p_1^k, p_2^k), \quad 1 \leq k \leq P. \tag{6.2}$$

This can be visualized as a collection of P arrows in color space, each arrow going from a source color to a destination color based on the photic parameter change $\theta_1 \mapsto \theta_2$. We call this vector field Φ' a *partially observed color flow* (see Figure 6-1**a**). The "partially observed" indicates that the vector field is only defined at the particular color points that happen to be in image I_1.

To obtain a *full color flow*(see Figure 6-1**b**), i.e. a vector field Φ defined at all points in $\mathcal{C}$, from a partially observed color flow Φ', we must address two issues. First, there will be many points in $\mathcal{C}$ at which no vector difference is defined. Second, there may be multiple pixels of a particular color in image I_1 that are mapped to different colors in image I_2. We propose the following interpolation scheme,[3] which defines the flow at a color point $(r, g, b)^T$ by computing a weighted proximity-based average of nearby observed "flow vectors":

$$\Phi(r, g, b) = \frac{\sum_{k=1}^{P} e^{-\|\mathbf{c}(p_1^k) - (r,g,b)^T\|^2 / 2\sigma^2} \Phi'(\mathbf{c}(p_1^k))}{\sum_{k=1}^{P} e^{-\|\mathbf{c}(p_1^k) - (r,g,b)^T\|^2 / 2\sigma^2}}. \tag{6.3}$$

This defines a color flow vector at every point in $\mathcal{C}$. Note that the Euclidean distance function used is defined in *color space*, not in the space defined by the [x,y] coordinates

[3]This scheme is analogous to a Parzen-Rosenblatt non-parametric kernel estimator for densities, using a 3-D Gaussian kernel. To be a good estimate, the true flow should therefore be locally smooth.

of the image. σ^2 is a variance term which controls the mixing of observed flow vectors to form the interpolated flow vector. As $\sigma^2 \to 0$, the interpolation scheme degenerates to a nearest-neighbor scheme, and as $\sigma^2 \to \infty$, all flow vectors get set to the average observed flow vector. In our experiments, we found empirically that a value of $\sigma^2 = 16$ (with colors on a scale from $0 - 255$) worked well in selecting a neighborhood over which vectors would be combined. Also note that color flows are defined so that a color point with only a single nearby neighbor will inherit a flow vector that is nearly *parallel* to its neighbor. The idea is that if a particular color, under a photic parameter change $\theta_1 \mapsto \theta_2$, is observed to get a little bit darker and a little bit bluer, for example, then its neighbors in color space are also defined to exhibit this behavior.

We have thus outlined a procedure for using a pair of corresponding images $\mathcal{I} = (I_1, I_2)$ to generate a full color flow. We will write for brevity $\Phi = \Phi(\mathcal{I})$ to designate the flow generated from the image pair $\mathcal{I}$.

6.3.1 Structure in the space of color flows

Certainly an image feature appearing as one color, say blue, in one image could appear as almost any other color in another image. Thus the *marginal distribution* of mappings for a particular color, when integrated over all possible photic parameter changes, is very broadly distributed. However, when color mappings are considered jointly, i.e. as color flows, we hypothesize that the space of possible mappings is much more compact. We test this hypothesis by statistically modeling the space of joint color maps, i.e. the space of color flows.

Consider for a moment a flat Lambertian surface that may have different reflectances as a function of the wavelength. While in principle it is possible for a change in lighting to map any color from such a surface to any other color *independently of all other colors*,[4] we know from experience that many such joint maps are unlikely. This suggests that there is significant structure in the space of color flows. (We will address below the significant issue of non-flat surfaces and shadows, which can cause highly "incoherent" maps.)

In learning color flows from real data, many common color flows can be anticipated. To name a few examples, flows which make most colors a little darker, lighter, or redder would certainly be expected. These types of flows can be well modeled with simple global linear operators acting on each color vector. That is, we can define a 3x3 matrix $\mathbf{A}$ that maps a color $\mathbf{c_1}$ in the image I_1 to a color $\mathbf{c_2}$ in the image I_2 via

$$\mathbf{c_2} = \mathbf{A}\mathbf{c_1}. \tag{6.4}$$

Such linear maps work well for many types of common photic parameter changes. However, there are many effects which these simple maps cannot model. Perhaps the most significant is the combination of a large brightness change coupled with a non-linear gain-control adjustment or brightness re-normalization by the camera. Such photic changes will tend to leave the bright and dim parts of the image alone, while spreading the central colors of color space toward the margins.

[4] By carefully choosing surface properties such as the reflectance of a point as a function of wavelength, $S(p, \lambda)$, and lighting conditions $E(\lambda)$, any mapping $\tilde{\Phi}$ can, in principle be observed even on a flat Lambertian surface. However, as noted in (Stiles et al., 1977; Maloney, 1986), the metamerism which would cause such effects is uncommon in practice.

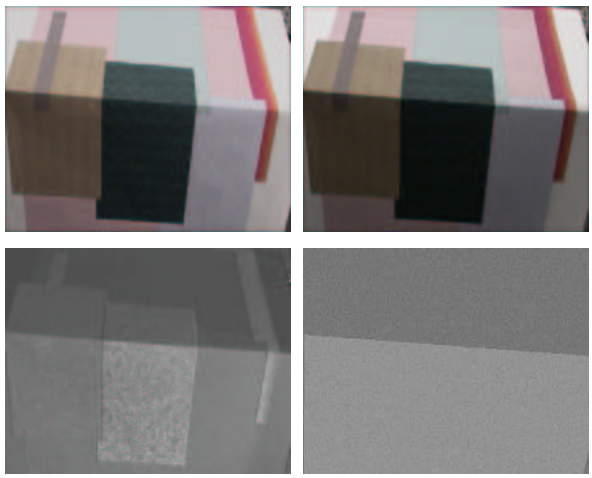


Figure 6-2: Evidence of non-linear color changes. The ratio of the top two images is shown in the lower left corner. The ideal ratio image, corresponding to a linear lighting model, is shown in the lower right.

An example of non-linearities in the imaging process is demonstrated in Figure 6-2. The top two images in Figure 6-2 are two photographs of a box covered with multicolored paper. The photos show the top and one side of the box. The lower left image is the ratio of the brightness of these two images, sometimes called the *quotient image* (Shashua and Riklin-Raviv, 2001). Since this is the ratio of images of two distinct smooth surfaces, it should have only two regions of smoothly varying pixels. However, it is clear that the ratios seen are variable, even within the individual regions. Examining the original images, it is clear that the ratio image is a function not only of surface normal, but also of albedo. The darker regions in the original images show different values in the ratio image than the lighter regions. The fact that the ratio image is still a function of the albedo is direct evidence of a non-linearity in the imaging process. These types of changes cannot be captured well by the simple linear operator described above, but can be captured by modeling the space of color flows.

Another pair of images exhibiting a non-linear color flow is shown in Figures 6-3**a** and **b**. Figure 6-3**a** shows the original image and **b** shows an image with contrast increased using a quadratic transformation of the brightness value. Notice that the brighter areas of the original image get brighter and the darker portions get darker. This effect cannot be modeled using a scheme such as that given in Equation 6.4. The non-linear color flow allows us to recognize that images **a** and **b** may be of the same object, i.e. to "match" the images.

6.3.2 Color flow PCA

Our aim was to capture the structure in color flow space by observing real-world data in an unsupervised fashion. To do this, we gathered data as follows. A large color palette (approximately 1 square meter) was printed on standard non-glossy plotter paper using every color that could be produced by our Hewlett Packard DesignJet 650C pen plotter (see Figure 6-4). The poster was mounted on a wall in our office so that it was in the direct

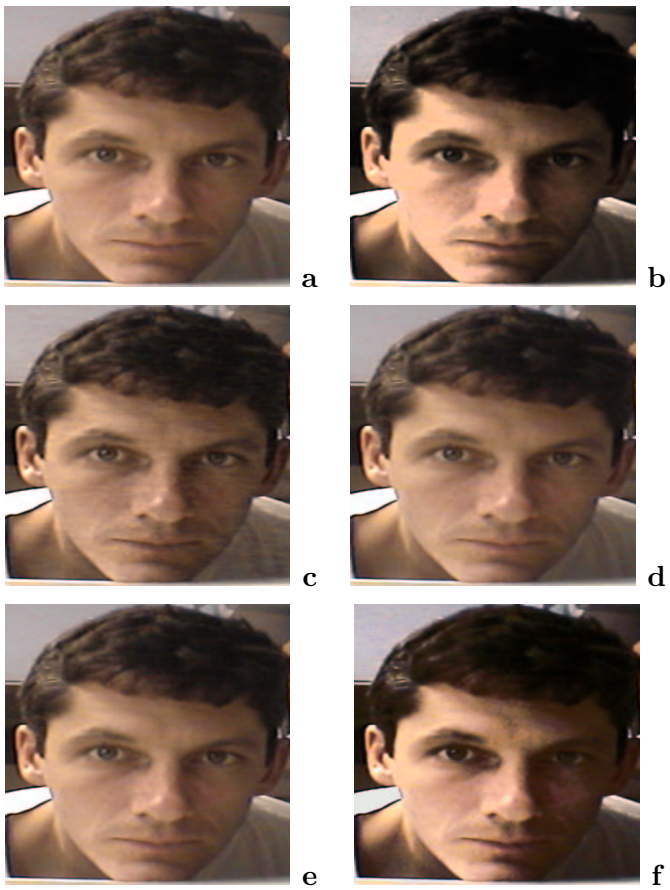


Figure 6-3: Matching non-linear color changes with color flows. Image **b** is the result of applying a non-linear operator to the colors in image **a**. **c-f** are attempts to match **b** using **a** and four different algorithms. Our algorithm (image **f**) was the only one to capture the non-linearity.

line of overhead lights and computer monitors, but not in the direct light from the single office window. An inexpensive video camera (the PC-75WR, Supercircuits, Inc.) with auto-gain-control was aimed at the poster so that the poster occupied about 95% of the field of view.

Images of the poster were captured using the video camera under a wide variety of lighting conditions, including various intervals during sunrise, sunset, at midday, and with various combinations of office lights and outdoor lighting (controlled by adjusting blinds). People used the office during the acquisition process as well, thus affecting the ambient lighting conditions. It is important to note that a variety of non-linear normalization mechanisms built into the camera were operating during this process.

Our goal was to capture as many common lighting conditions as possible. We did not use unusual lighting conditions such as specially colored lights. Although a few images that were captured probably contained strong shadows, most of the captured images were shadow-free. Smooth lighting gradients across the poster were not explicitly avoided or created in our acquisition process.

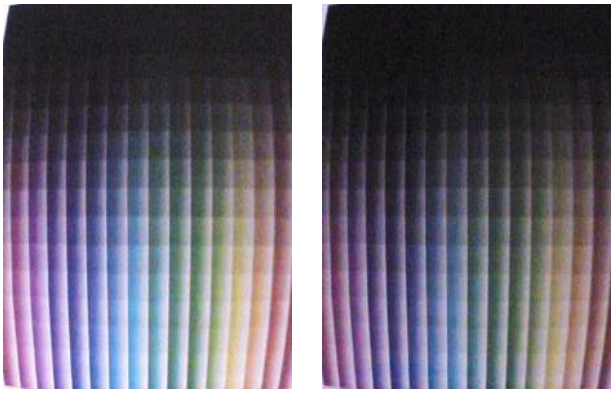


Figure 6-4: Images of the poster used for observing color flows, under two different "natural" office lighting conditions. Note that the variation in a single image is due to reflectance rather than a lighting gradient.

A total of 1646 raw images of the poster were obtained in this manner. We then chose a set of 800 image pairs $\mathcal{I}^j = (I_1^j, I_2^j), 1 \leq j \leq 800$, by randomly and independently selecting individual images from the set of raw images. Each image pair was then used to estimate a full color flow $\Phi(\mathcal{I}^j)$ as described in Equation 6.3.

Note that since a color flow Φ can be represented as a collection of $3Q$ coordinates, it can be thought of as a point in $\mathbb{R}^{3Q}$. Here Q is the number of distinct RGB colors at which we compute a flow vector, and each flow vector requires 3 coordinates: dr, dg, and db, to represent the change in each color component. In our experiments we used $Q = 16^3 = 4096$ distinct RGB colors (equally spaced in RGB space), so a full color flow was represented by a vector of $3 * 4096 = 12288$ components.

Given a large number of color flows (or points in $\mathbb{R}^{3Q}$), there are many possible choices for modeling their distribution. We chose to use Principal Components Analysis since 1) the flows are well represented (in the mean-squared-error sense) by a small number of principal components (see Figure 6-5), and 2) finding the optimal description of a difference image in terms of color flows was computationally efficient using this representation (see Section 6.4).

The principal components of the color flows were computed (in MATLAB), using the "economy size" singular value decomposition. This takes advantage of the fact that the data matrix has a small number of columns (samples) relative to the number of components in a single sample.

We call the principal components of the color flow data "color eigenflows", or just eigenflows,[5] for short. We emphasize that these principal components of color flows have *nothing to do with the distribution of colors in images*, but only model the distribution of *changes in color*. This is a key and potentially confusing point. In particular, we point out that our work is very different from approaches that compute principal components in the intensity or color space itself, such as (Turk and Pentland, 1991) and (Soriano et al., 1999). Perhaps the most important difference is that our model is a global model for all images,

[5]PCA has been applied to *motion* vector fields as in (Lien, 1998), and these have also been termed "eigenflows".

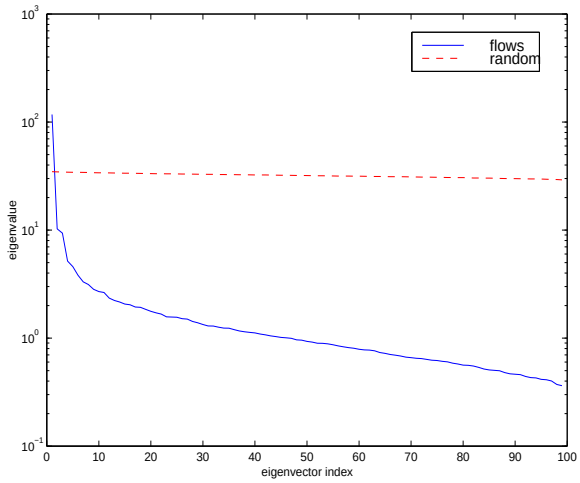


Figure 6-5: Eigenvalues of the color flow covariance matrix. The rapid drop off in magnitude indicates that a small number of eigenflows can be used to represent most of the variance in the distribution of flows.

while the above methods are models only for a particular set of images, such as faces.

An important question in applying PCA is whether the data can be well represented with a "small" number of principal components. In Figure 6-5, we plot the eigenvalues associated with the first 100 eigenflows. This rapidly descending curve indicates that most of the magnitude of an average sample flow is contained in the first ten components. This can be contrasted with the eigenvalue curve for a set of random flows, which is also shown in the plot.

6.4 Using color flows to synthesize novel images

How do we generate a new image from a source image and a color flow or group of color flows? Let $\mathbf{c}(p)$ be the color of a pixel p in the source image, and let Φ be a color flow that we have computed at a discrete set of Q points according to Equation 6.3. For each pixel in the new image, its color $\mathbf{c}'$ can be computed as

$$\mathbf{c}'(p) = \mathbf{c}(p) + \alpha\Phi(\hat{\mathbf{c}}(p)), \tag{6.5}$$

where α is a scalar multiplier that represents the "quantity of flow". $\hat{\mathbf{c}}(p)$ is interpreted to be the color vector closest to $\mathbf{c}(p)$ (in color space) at which Φ has been computed. If the $\mathbf{c}'(p)$ has components greater than the allowed range of 0–255, then these components must be truncated.

Figure 6-6 shows the effect of each of the eigenflows on an image of a face. Each vertical sequence of images represents an original image (in the middle of the column), and the images above and below it represent the addition or subtraction of each eigenflow, with α varying between ± 8 standard deviations for each eigenflow.

We stress that the eigenflows were only computed once (on the color palette data),and that they were applied to the face image without any knowledge of the parameters under

which the face image was taken.

The first eigenflow (on the left of Figure 6-6) represents a generic brightness change that could probably be represented well with a linear model. Notice, however, the third column in Figure 6-6. Moving downward from the middle image, the contrast grows. The shadowed side of the face grows darker while the lighted part of the face grows lighter. This effect cannot be achieved with a simple matrix multiplication as given in Equation 6.4. It is precisely these types of non-linear flows we wish to model.

6.4.1 From flow bases to image bases

Let $\mathcal{S}$ be the set of all images that can be created from a novel image and a set of eigenflows. Assuming no color truncation, we show how we can efficiently find the image in $\mathcal{S}$ which is closest (in an L_2 sense) to a target image.

Let $p_{x,y}$ be a pixel whose location in an image is at coordinates [x,y]. Let $I[x, y]$ be the vector at the location [x,y] in an image or in a difference image. Suppose we view an image I as a function that takes as an argument a color flow and that generates a difference image D by placing at each (x,y) pixel in D the color change vector $\Phi(\mathbf{c}(p_{x,y}))$. We denote this simply as

$$D = I(\Phi). \tag{6.6}$$

Then this "image operator" $I(\cdot)$ is linear in its argument since for each pixel (x,y)

$$\begin{aligned} (I(\Phi + \Psi))[x, y] &= (\Phi + \Psi)(\mathbf{c}(p_{x,y})) && (6.7)\\ &= \Phi(\mathbf{c}(p_{x,y})) + \Psi(\mathbf{c}(p_{x,y}))). && (6.8) \end{aligned}$$

The + signs in the first line represent vector field addition. The + in the second line refers to vector addition. The second line assumes that we can perform a meaningful component-wise addition of the color flows.

Hence, the difference pixels in a total difference image can be obtained by adding the difference pixels in the difference images due to each eigenflow (the *difference image basis*). This allows us to compute *any* of the possible image flows for a particular image and set of eigenflows from a (non-orthogonal) difference image basis. In particular let the difference image basis for a particular source image I and set of E eigenflows $\Psi_i, 1 \leq i \leq E$, be represented as

$$D_i = I(\Psi_i). \tag{6.9}$$

Then the set of images $\mathcal{S}$ that can be formed using a source image and a set of eigenflows is

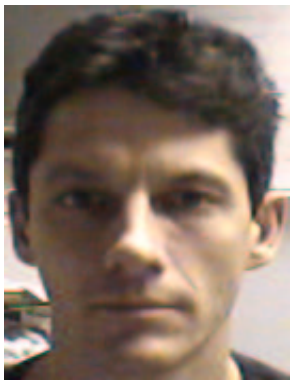
$$\mathcal{S} = \{S : S = I + \sum_{i=1}^{E} \gamma_i D_i\}, \tag{6.10}$$

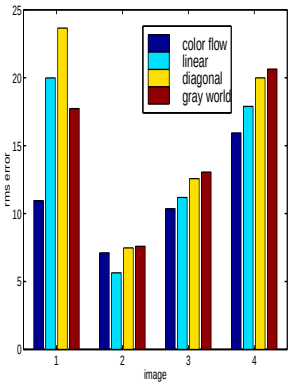
where the γ_i's are scalar multipliers, and here I is just an image and not a function. In our experiments, we used $E = 30$ of the top eigenvectors to define the space $\mathcal{S}$.

6.4.2 Flowing one image to another

Suppose we have two images and we pose the question of whether they are images of the same object or scene. We suggest that if we can "flow" one image to another then the images are likely to be of the same scene.

Figure 6-6: Effects of the first three eigenflows. See text.


a



b

Figure 6-7: **a.** Original image used for image matching experiments. **b.** Errors per pixel component in the reconstruction of the target image for each method.

We can only flow image I_1 to another image I_2 if it is possible to represent the difference image as a linear combination of the D_i's, i.e. if $I_2 \in \mathcal{S}$. However, we may be able to get "close" to I_2 even if I_2 is not an element of $\mathcal{S}$.

Fortunately, we can directly solve for the optimal (in the least-squares sense) γ_i's by just solving the system

$$D = \sum_{i=1}^{E} \gamma_i D_i, \tag{6.11}$$

using the standard pseudo-inverse, where $D = I_2 - I_1$. This minimizes the error between the two images using the eigenflows.[6] Thus, once we have a basis for difference images of a source image, we can quickly compute the best flow to any target image. We point out again that this analysis ignores truncation effects. While truncation can only reduce the error between a synthetic image and a target image, it may change which solution is optimal in some cases.

6.5 Experiments

One use of the color change model is for image matching applications. The goal of such a system would be to flow one image to another as well as possible when the images are actually of the same scene, but not to endow the system with enough capacity to be able to

[6]The generative model for difference images can be extended to include a noise source as in

$$D = \sum_{i=1}^{E} \gamma_i D_i + n,$$

where n is pixelwise and channelwise independent Gaussian noise with diagonal covariance Λ_n. Estimation of the MAP coefficients $\hat{\gamma}_i$ then becomes

$$\hat{\gamma}_{MAP} = (\mathbf{D}^T \Lambda_N^{-1} \mathbf{D} + \Lambda_\gamma^{-1})^{-1} \mathbf{D} \Lambda_N^{-1} D \tag{6.12}$$

$$= (\mathbf{D}^T \Lambda_N^{-1} \mathbf{D} + \mathbf{I})^{-1} \mathbf{D} \Lambda_N^{-1} D. \tag{6.13}$$

Figure 6-8: Target images for image matching experiments. The images in the top row were taken with a digital camera. The images in the bottom row are the best approximations of those images using the eigenflows and the source image from Figure 6-7.

flow between images that do not in fact match. An ideal system would thus flow one image to a matching image with zero error, and have large errors for non-matching images. Then setting a threshold on such an error would determine whether two images were of the same scene.

We first examined our ability to flow a source image to a matching target image under different photic parameters. We compared our system to 3 other methods commonly used for brightness and color normalization. We shall refer to the other methods as **linear**, **diagonal**, and **gray world**. The **linear** method finds the matrix **A** according to Equation 6.4 that minimizes the L_2 fit between the synthetic image and the target image. **diagonal** does the same except that it restricts the matrix **A** to be diagonal. **gray world** adjusts each color channel in the synthetic image linearly so that the mean red, green, and blue values match the mean channel values in the target image.

While our goal was to reduce the numerical difference between two images using flows, it is instructive to examine one example which was particularly visually compelling, shown in Figure 6-3. Part **a** of the figure shows an image taken with a digital camera. Part **b** shows the image adjusted by squaring the brightness component (in an HSV representation) and re-normalizing it to 255. The goal was to adjust image **a** to match **b** as closely as possible (in a least squares sense). Images **c-f** represent the **linear, diagonal, gray world,** and **eigenflow** methods respectively. While visual results are somewhat subjective, it is clear that our method was the only method that was able to significantly darken the darker side of the face while brightening the lighter side of the face. The other methods which all implement linear operations in color space (ours allows non-linear flows) are unable to perform this type of operation.

In another experiment, five images of a face were taken while changing various camera parameters, but lighting was held constant. One image was used as the source image (Figure 6-7**a**) in each of the four algorithms to approximate each of the other four images (see Figure 6-8).

Figure 6-7**b** shows the component-wise RMS errors between the synthesized images and the target image for each method. Our method outperforms the other methods in all but one task, on which it was second.

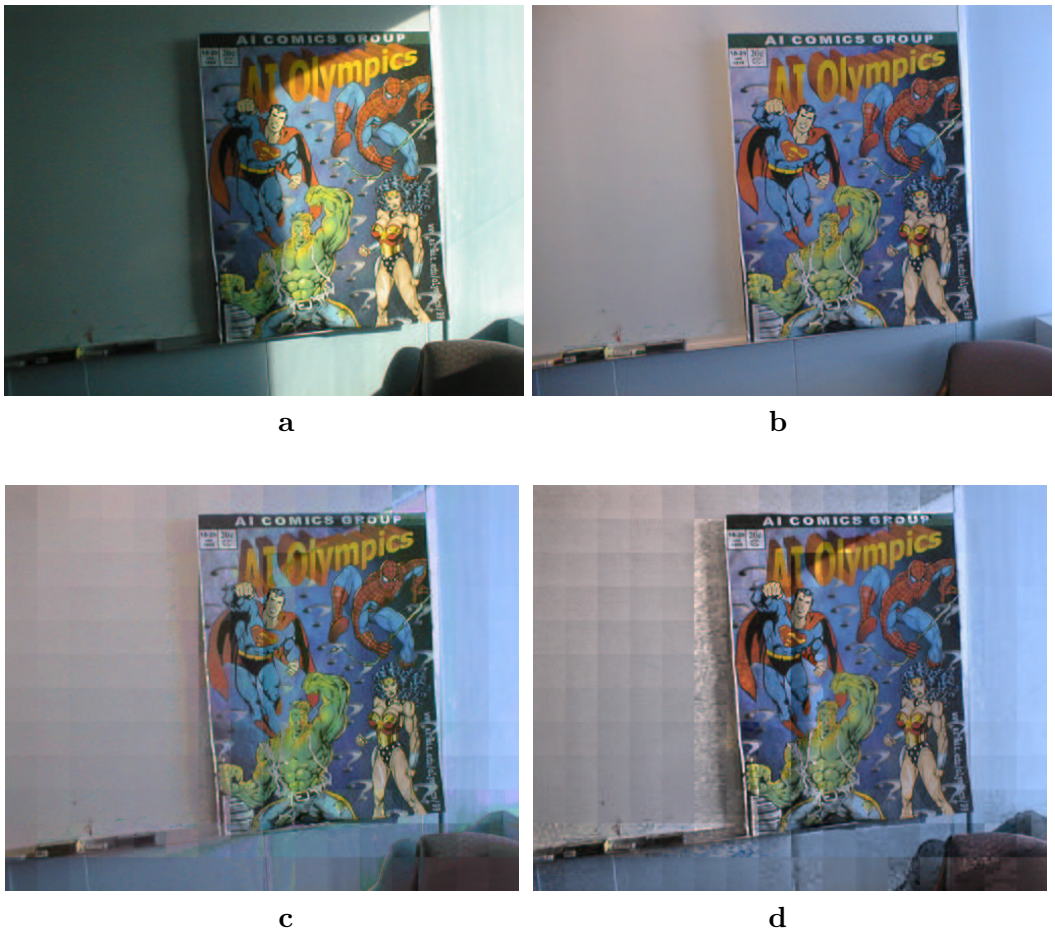


Figure 6-9: Modeling lighting changes with color flows. **a.** Image with strong shadow. **b.** The same image under more uniform lighting conditions. **c.** Flow from **a** to **b** using eigenflows. **d.** Flow from **a** to **b** using **linear**.

In another test, the source and target images were taken under very different lighting conditions (Figures 6-9**a** and **b**). Furthermore, shadowing effects and lighting direction changed between the two images. None of the methods could handle these effects when applied globally. To handle these effects, we used each method on small patches of the image. Our method again performed the best, with an RMS error of 13.8 per pixel component, compared with errors of 17.3, 20.1, and 20.6 for the other methods. Figures 6-9**c** and **d** show the reconstruction of image **b** using our method and the best alternative method (**linear**). There are obvious visual artifacts in the linear method, while our method seems to have produced a much better synthetic image, especially in the shadow region at the edge of the poster.

One danger of allowing too many parameters in mapping one image to another is that images that do not actually match will be matched with low error. By performing synthesis on patches of images, we greatly increase the capacity of the model, running the risk of over-parameterizing or over-fitting our model. We performed one experiment to measure the over-fitting of our method versus the others. We horizontally flipped the image in Figure 6-9**b** and used this as a target image. On the left of Figure 6-10 is the reversed image. On the right side of Figure 6-10 is the result of trying to flow the original image

a **b**

Figure 6-10: Evaluating the capacity of the color flow model. **a.** Mirror image of superman image shown in Figure 6-9**b**. **b.** The attempt to flow the superman image to its own mirror image. The failure to do this implies that the color change model is not overparameterized.

to the reversed version. In this case, we wanted the error to be large, indicating that we were unable to synthesize a similar image using our model. The RMS error per pixel component was 33.2 for our method versus 41.5, 47.3, and 48.7 for the other methods. Note that while our method had lower error (which is undesirable), there was still a significant spread between matching images and non-matching images.

We believe we can improve differentiation between matching and non-matching image pairs by assigning a cost to the *change in coefficients* γ_i across each image patch. For images which do not match, we would expect the γ_i's to change rapidly to accommodate the changing image. For images which do match, sharp changes would only be necessary at shadow boundaries or sharp changes in the surface orientation relative to directional light sources. We believe this can significantly enhance the method, by adding a strong source of information about how the capacity of the model is actually being used to match a particular image pair.

6.5.1 Shadows

Shadows pose a number of interesting problems in computer vision. Shadows confuse tracking algorithms (Toyama et al., 1999), backgrounding schemes and object recognition algorithms. For example, shadows can have a dramatic effect on the magnitude of difference images, despite the fact that no "new objects" have entered a scene. Shadows can also move across an image and appear as moving objects. Many of these problems could be eliminated if we could recognize that a particular region of an image is equivalent to a previously seen version of the scene, but under a different lighting. For example, suppose that the lighting impinging upon a flat surface has changed due to a nearby lamp being turned on. The changing angle of incidence will make it difficult to model the image transformation as a single mapping of color space from one image to the other.

In Figure 6-11**a**, we show a simple background image. In Figure 6-11**b**, a person and his shadow have appeared in the image. We consider the problem of distinguishing between the person (a new object) and the shadow (a lighting change). We did the following experiment. With simple image differencing, we segmented the image into two approximately connected

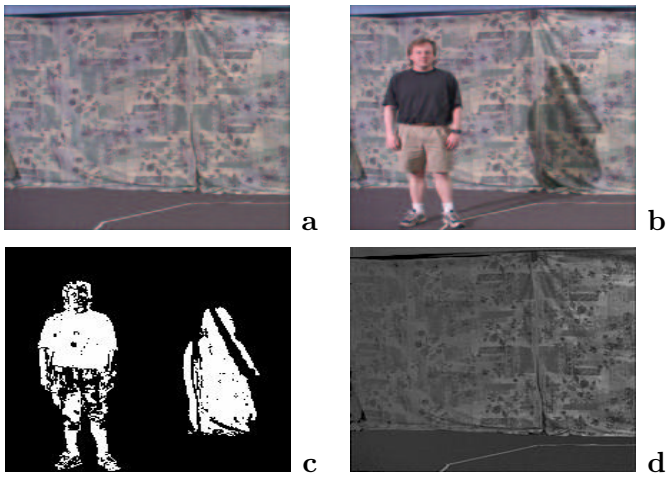


Figure 6-11: Backgrounding with color flows. **a** A background image. **b** A new object and shadow have appeared. **c** For each of the two regions, a "flow" was done between the original image and the new image based on the pixels in each region. **d** The color flow of the original image using the eigenflow coefficients recovered from the shadow region. The color flow using the coefficients from the non-shadow region are unable to give a reasonable reconstruction of the new image.

regions that did not match the previous background (Figure 6-11**c**). For each component, we then flowed (chose eigenflow coefficients) the region from image **a** to image **b** according to Equation 6.11. Figure 6-11**d** shows the full image based on the shadow flow.

To distinguish between shadows and non-shadows, we want the average residual error for non-shadows to be high while the average residual error for shadows to be low. Since these are real images, however, a constant color flow across an entire region may not model the image change well.

However, we can easily extend our basic model to allow linearly or quadratically (or other low order polynomially) varying fields of eigenflow coefficients. That is, we can find the best least squares fit of the difference image allowing our γ estimates to vary linearly or quadratically over the image. We implemented this technique by computing flows $\gamma_{x,y}$ between corresponding image patches (indexed by x and y), and then minimizing the following form:

$$\arg\min_{M} \sum_{x,y} (\gamma_{x,y} - M c_{x,y})^T \Sigma_{x,y}^{-1} (\gamma_{x,y} - M c_{x,y}). \tag{6.14}$$

Here, each $c_{x,y}$ is a vector polynomial of the form $[x\ y\ 1]^T$ for the linear case and $[x^2\ xy\ y^2\ x\ y\ 1]^T$ for the quadratic case. M is an Ex3 matrix in the linear case and an Ex6 matrix in the quadratic case. It defines each of the E planes or quadrics respectively. The $\Sigma_{x,y}^{-1}$'s are the error covariances in the estimate of the $\gamma_{x,y}$'s for each patch.

Allowing the γ's to vary over the image greatly increases the capacity of a matcher, but by limiting this variation to linear or quadratic variation, the capacity is still not able to qualitatively match "non-matching" images. Note that this smooth variation in eigenflow coefficients can model either a nearby light source *or* a smoothly curving surface, since either of these conditions will result in a smoothly varying lighting change.

	constant	linear	quadratic
shadow	36.5	12.5	12.0
non-shadow	110.6	64.8	59.8

Table 6.1: Error residuals for shadow and non-shadow regions after color flows. Constant, linearly varying, and quadratically varying flows were used.

We consider three versions of the experiment. In the first, we assign a single vector of flow coefficients to all of the pixels in the region. In the second experiment, we allowed the γ values to vary linearly across the image. This should lead to a reduction in the error of both regions' residuals. In the final experiment, we fitted quadratically varying γ values to estimate the image difference. The results of these experiments appear in Table 6.1.

In each case, the residual error for the shadow region is much lower than for the non-shadow region. Of course, we have not specified where to select the threshold so that this procedure works in general. Furthermore, there are other methods available, such as normalized correlation and methods such as (Gordon et al., 1999; Horprasert et al., 2000), which could also distinguish between these two regions. However, this demonstrates another potential application of our model. We believe because it can handle non-linear camera effects and can be adjusted across the image that it can successfully model a great deal of the variability in true shadows, whereas it still does not have so much capacity as to match images which are not in fact of the same scene. However, it still has limitations, such as when a shadow is so dark that it cannot be distinguished from a black object.

6.5.2 Conclusions

Except for the synthesis experiments, most of the experiments in this chapter are preliminary and only a proof of concept. Much larger experiments need to be performed to establish the utility of the color change model for particular applications. However, since the color change model represents a compact description of lighting changes, including non-linearities, we are optimistic about these applications. To use this method as part of an object recognition system, we have to deal with geometric variation in addition to photic parameters, which suggests an approach that combines congealing with color flow methods.

In the next chapter, we bring together the notion of optical flow fields (or spatial transformations) and color flow fields into a unified framework.

Chapter 7

Feature Flow Fields

7.1 Introduction

In Chapters 4 and 6, we discussed methods for using models of spatial change and color change to develop object models from only a single example of an object. In this last chapter, we show how these modes of change can be considered two examples of the same phenomenon.[1] We present a unifying framework in which object-independent modes of variation are learned from continuous-time data such as video sequences. These modes of change can then be used directly as a model of change for a particular feature type, or as a starting point in defining a bias for creating more refined, environment specific models.

As shown by Black et al. (Black et al., 1997b; Fleet et al., 2000), it is possible to learn, from generic video data, close approximations to the standard affine deformations that are expected from visual geometry,[2] and to do so in a completely unsupervised fashion. This can be achieved simply by performing clustering or principal components analysis on a set of optical flow fields derived from a simple video sequence from a moving camera. Such a set of statistically gathered flows is shown in Figure 5-11.

This process parallels the process used in Chapter 6 to define common modes of color change. Thus, it is not surprising that with a little work, a formalism can be developed in which these types of learning are viewed similarly. In addition to the conceptual simplification, the benefits of this unification will be manifold, immediately suggesting a wide variety of algorithms that may be borrowed from one domain and used in another.

The structure defined in this chapter is termed a *feature flow field*. The feature flow field is simply a representation of a feature change (a flow vector) at each point in a feature space (the field). Two very different feature flow fields, like color flows and optical flows, really differ only in the type of feature they are modeling.

For optical flow fields, the feature at each point in the feature flow field is a coordinate feature. Each element of the flow field describes how the position of a particular piece of the world (like the corner of a box) moves from one image to another. The field in this case is the two-dimensional plane of the image itself, and the optical flow is defined on a regular array of grid points in this field.

For color flow fields, the feature at each point is an RGB color triple. Each element of the flow field describes how the color of an image element changes from one image to the

[1] Most of the contents of this chapter have appeared previously as (Miller et al., 2001).

[2] Although optical projections produce more complex deformations than affine (perspective, that is), affine represent a close approximation to most visual projections.

next. The field in a color flow field is the three-dimensional space of colors. Color changes are defined on a regular array of points in this field. We now define feature flow fields more formally.

7.2 The feature flow field

The feature flow field can be easily understood as a generalization of the color flow field of Chapter 6. To the extent possible, notation was borrowed from Chapter 6. In the following, let z be a *world point* that can be seen in an image. By world point, we mean a feature of the physical world, like the corner of a box or the tip of someone's index finger. Let $\mathbf{f}(z) \in \mathbb{R}^D$ be a vector-valued feature of the world point z. That is, it is a property, in the image, of the world point named z. Further assume that each component of $\mathbf{f}(z)$ takes values from a finite set $\mathcal{S}$. Let $\mathcal{F} = \{s \in \mathcal{S}^D\}$ be the set of all possible feature vector values.

We will consider two types of features. In the first application, we will use *coordinate features*, in which $\mathbf{f}(z)$ will represent the *image* coordinates of some world point z that appears in the image. In this case, the space $\mathcal{F}$ would be all possible coordinate values of a pixel in the image. In the second application, the features will be *color features*, in which $\mathbf{f}(z)$ will represent the integer RGB color values in the image of the world point z. In this case, the space $\mathcal{F}$ would be all possible RGB color triples for a pixel.

The notions of world points and coordinate features are potentially confusing. To understand coordinate features, consider an identifiable world point in some scene. This could be the tip of a person's index finger for example. Let the world point at the person's finger tip be denoted z. If the tip of the person's finger in image I_1 is at pixel coordinates $[x_1\ y_1]^T$ and in image I_2 is at pixel coordinates $[x_2\ y_2]^T$, then the coordinate feature flow (or optical flow) for the world point z, $\mathbf{f}(z)$, will be the 2-vector defined by $\mathbf{f}(z) = [x_2\ y_2]^T - [x_1\ y_1]^T$.

Suppose we are given two P-pixel images I_1 and I_2 of the same scene taken under two different parameter settings, represented by θ_1 and θ_2. These could represent different lighting conditions, different camera positions, different camera gain settings, etc. We assume the images are consecutive images from a video sequence, and that the parameter values vary smoothly through time. We assume that we have a method of putting the images in correspondence. For optical flow fields, this will entail some sort of optical flow algorithm (Horn, 1986). For color flow fields, the scene is static and so images are automatically in correspondence. In optical flow, the pixel corresponding to a world point in image I_2 has often moved out of the scene. For simplicity, we will ignore that issue here, but it can be remedied using the interpolation scheme discussed below.

Let each pixel $p_1^k,\ 1 \leq k \leq P$, in image I_1 define a world point z^k. That is, z^k is the world point imaged by pixel p_1^k of image I_1. Let pixel p_2^k be the pixel that is imaging world point z^k in image I_2. From here forward, we shall use the term "pixel" to mean the world point imaged by that pixel.

Each pair of corresponding image pixels p_1^k and p_2^k, in the two images can be interpreted as a mapping $\mathbf{f}(p_1^k) \mapsto \mathbf{f}(p_2^k)$.[3] That is, it tells us how a particular pixel's feature changes from image I_1 to image I_2. This single feature mapping is conveniently represented by the vector difference between the two pixel features:

$$\mathbf{d}(p_1^k, p_2^k) = \mathbf{f}(p_2^k) - \mathbf{f}(p_1^k). \tag{7.1}$$

[3]Here $\mathbf{f}(p_1^k)$ is meant as a shorthand for $\mathbf{f}(z(p_1^k))$, i.e. the feature of the world point imaged by pixel p_1^k.

By computing P of these vector differences (one for each pair of pixels) and placing each vector difference at the point $\mathbf{f}(p_1^k)$ in the feature space $\mathcal{F}$, we have created a vector field that is defined at all points in $\mathcal{F}$ for which there are feature values in image I_1.

That is, we are defining a vector field Φ' over $\mathcal{F}$ via

$$\Phi'(\mathbf{f}(p_1^k)) = \mathbf{d}(p_1^k, p_2^k), \quad 1 \leq k \leq P. \tag{7.2}$$

This can be visualized as a collection of P arrows in feature space, each arrow going from a source feature to a destination feature based on the parameter change $\theta_1 \mapsto \theta_2$. We call this vector field Φ' a *partially observed feature flow*. The "partially observed" indicates that the vector field is only defined at the particular feature points that happen to be observed in image I_1. Figure 3-2**b** shows an (optical) flow field and Figure 6-1 shows both partial and complete (color) flow fields.

As in the chapter on color flow fields, we need to specify a method for interpolating the values in the feature flow field that have not been observed. We adopt the interpolation technique from Chapter 6 as a generic method for all flow fields.

To obtain a *full feature flow*, i.e. a vector field Φ defined at all points in $\mathcal{F}$, from a partially observed feature flow Φ', we must address two issues. First, there may be points in $\mathcal{F}$ at which no vector difference is defined. Second, there may be multiple pixels of a particular feature value in image I_1 that correspond to different feature values in image I_2. This latter problem is more of an issue with color flow fields, since it is quite common to have a single color in one image map to multiple colors in another image. We again use the radial basis function interpolation scheme, which defines the flow at a feature point $\mathbf{f}^*$ by computing a weighted proximity-based average of observed "flow vectors":

$$\Phi(\mathbf{f}^*) = \frac{\sum_{k=1}^{P} e^{-\|\mathbf{f}(p_1^k) - \mathbf{f}^*\|^2 / 2\sigma^2} \Phi'(\mathbf{f}(p_1^k))}{\sum_{k=1}^{P} e^{-\|\mathbf{f}(p_1^k) - \mathbf{f}^*\|^2 / 2\sigma^2}}. \tag{7.3}$$

This defines a feature flow vector at every point in $\mathcal{F}$. Note that the Euclidean distance function used is defined in *feature space*, not necessarily in the space defined by the [x,y] coordinates of the image. σ^2 is a variance term which controls the mixing of observed flow vectors to form the interpolated flow vector. Feature flows are defined so that a feature point with only a single nearby neighbor will inherit a flow vector that is nearly *parallel* to its neighbor.

7.3 Benefits of the generalization

We have already seen feature flow fields applied to the task of modeling color changes in Chapter 6. But one might legitimately ask, "What role do feature flow fields play in the domain of spatial changes, since we already have good models of many spatial changes, i.e. the affine model and the projective model?" There are at least two answers to this question.

First, from a machine learning viewpoint, the question of how a learning machine can discover certain modes of variability on its own, rather than being programmed with them, is very important. This is closely related to the issue of learning a *bias* for learning, as described by Baxter and others (Baxter, 2000). Understanding how a simple type of variability, such as affine variability, can be learned without supervision should enable the development of algorithms that learn other types of variability which are less easy to specify directly. The

congealing of digits parameterized by *learned* modes of spatial variation rather than by affine variation, as described in Section 5.4.2, represents a new level of autonomy in handwriting recognition algorithms. No information about spatial deformations was provided to the program. It was simply provided with a video and a method of learning about very general spatial changes. These learned modes of spatial change were then applied to the problem of aligning handwritten digits.[4]

Second, while affine models suffice for many problems, more accurate statistical models based upon real video data may be superior in certain domains. For example, a vision system for an automobile would see spatial deformations heavily biased toward expansion, due to the forward movement of the vehicle, with occasional flow fields due to turning of the vehicle. Vertical translation, rotation about the viewing axis, and other modes of spatial variability would be rare. Thus, one would expect such a system to benefit from a custom statistical model of such deformations.

There are many other benefits derived from putting optical flow fields and color flow fields in the same theoretical framework. Many ideas that have been applied in one domain are transferrable to the other. For example, many of the ideas from the optical flow literature, which is large and well-developed, can be applied to color flows. Several authors have experimented with layered optical flows (Darrell and Pentland, 1991) and with mixtures of optical flows (Jepson and Black, 1993). As discussed in these papers, differences in a scene through time, due to motion in the scene, often cannot be explained by a single motion, but must be explained with multiple independent motion models. In (Jepson and Black, 1993), the authors provide an optimization procedure based on the EM algorithm to estimate the various motion fields, and the pixel membership in these motion fields.

An equivalent idea can be applied with color flows. That is, a scene with multiple lighting changes could be explained using such an EM approach. For example, it is immediately obvious to a person that a scene such as that shown in Figure 6-9 has undergone two distinct lighting changes from one frame to the next. While we introduced in Chapter 6 the notion of piecewise modeling of color change on a regular grid to explain complex lighting changes between two images, a better explanation of the image change in Figure 6-9 would describe the change in the scene by describing the two separate lighting changes and the exact regions over which they apply. An approach equivalent to the EM approach taken above is suggested, and represents a direction for future work with color flows.

Another opportunity for transference is the idea of simplifying color flow models in a way similar to how motion fields have been simplified. In particular, the construction of a tight fitting non-parametric model over spatial transformations was possible, as described in Chapter 3, because a very general and high-dimensional spatial transformation model was simplified to a lower-dimensional affine model. This suggests putting a density over affine color flows, rather than the very general vector field color flows, to more accurately model the statistics of color change. Specifically, we propose developing a non-parametric density over affine color changes analagous to the density estimator described in Section 3.5.2.

If the goal is to develop learning algorithms that are assisted as little as possible by

[4] A common objection to this approach is that there is no clear connection between the modes of spatial variability in a videotape and the modes of spatial variability in handwritten digits. In other words, why should modes of spatial variation learned from a video help us to model digits? We point out first that the human visual system must deal with affine variability, since it arises naturally out of the imaging of the three-dimensional world. We conjecture that, since handwriting was developed by humans, affine variability in handwriting is tolerated precisely because this is a mode of variation to which we are relatively insensitive. This is one way of justifying such a technique.

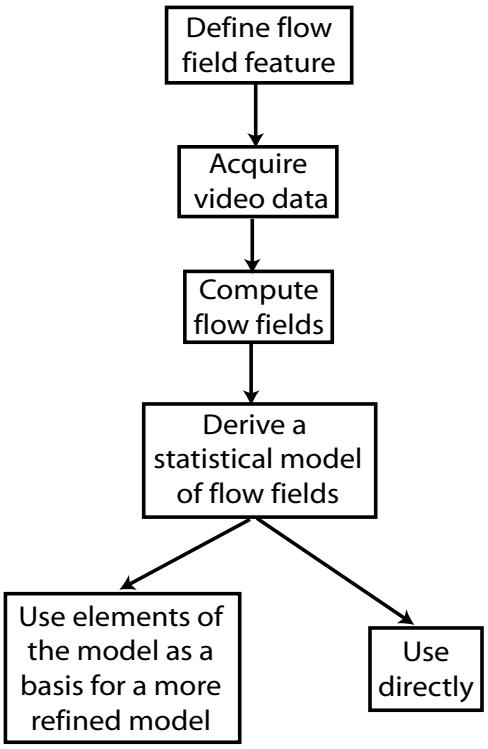


Figure 7-1:

preprogrammed biases about how images vary, then the feature flow field analysis suggests at least a partial strategy: observe continuous data, differentiate feature fields, and do statistical analysis on the results. This is a very general paradigm.

This type of approach is diagrammed in Figure 7-1. To model a type of change, we first define a feature, like color or position. Next, continuous video data is acquired. Each pair of frames in the video provide us with a sample of a feature flow field. Next, a statistical analysis of the flows can provide us with the most common modes of change. These flow components can be used directly as a model, or as a set of basis flows over which more refined models can be developed.

7.4 Other type of flow fields

In future work, we hope to find other examples of feature flow fields, including, for example, auditory flow fields. We conjecture that there is a great deal of statistical structure in joint frequency-power change fields. By this we mean that if an audio sequence is observed through time, and a flow field is created between the power of each frequency at one time and the power of the same frequency at the next time, we believe these fields should show a great deal of statistical structure. For example, auditory harmonics tend to increase and decrease in power together, and this structure should be captured in such an analysis.

Alternatively, one could create a "power-frequency" flow field, in which features of the frequency distribution, such as local power maxima, are tracked through time. The hope is that such a procedure would automatically detect certain joint frequency shifting effects such as the Doppler effect due to a passing sound source.

The frequency-power flows are in correspondence automatically, since the position of a particular frequency in a spectrogram or power spectrum never changes. In this sense, they are the auditory analog of color flow fields. The power-frequency flows, where feature tracking is required to establish correspondence, is more reminiscent of optical flow fields.

By using frequency flow fields as a basis, congealing of word spectrograms, for example, may be possible. That is, suppose we have the same word spoken by 100 different speakers. We propose forming a factorized model of the word by congealing the words to a canonical form (a "latent word") and forming a density on common deformations away from this word ("accents").

7.5 Conclusions

This thesis has focused on the sharing of knowledge between tasks in machine vision by reusing densities on image change. We have leveraged the ideas of our predecessors on learning-to-learn, but we have also made significant new contributions. By taking advantage of the continuity properties of images, building probabilistic factorized models, and focusing on the modeling of change, we have accomplished a number of goals that might not have been possible with more general models.

Among the achievements were the development of a classifier from a single training example of each class, the congealing method for joint image alignment, and a new model of joint color change. We hope that the framework suggested in this last chapter provides motivation and suggestions for future work.

Bibliography

Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276.

Amit, Y., Grenander, U., and Piccioni, M. (1991). Structural image restoration through deformable templates. *Journal of the American Statitistical Assocation*, 86:376–387.

Baddeley, R. J. and Hancock, P. J. B. (1992). Principal components of natural images. *Network: computation in neural systems*, 3:61–70.

Baxter, J. (1995). Learning internal representations. In *Proceedings of the 8th International ACM Workshop on Computational Learning Theory*, pages 311–320.

Baxter, J. (1998). Theoretical models of learning to learn. In Thrun, S. and Pratt, L., editors, *Learning to learn*. Kluwer Academic Publishers.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198.

Belhumeur, P. N. and Kriegman, D. (1998). What is the set of images of an object under all possible illumination conditions? *International Journal of Computer Vision*, 28(3):1–16.

Bell, A. J. and Sejnowski, T. J. (1997). Edges are the "independent components" of natural scenes. In *Advanced in Neural Information Processing Systems 9*.

Belongie, S., Malik, J., and Puzicha, J. (2001). Matching shapes. In *International Conference on Computer Vision*.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis, 2nd edition*. Springer.

Beymer, D. and Poggio, T. (1995). Face recognition from one example view. Technical Report CBCL Paper 121/AI Memo 1536, Massachusetts Institute of Technology.

Black, M. J., Fleet, D. J., and Yacoob, Y. (1998). A framework for modeling appearance change in image sequences. In *International Conference on Computer Vision*, pages 660–667.

Black, M. J., Fleet, D. J., and Yacoob, Y. (2000). Robustly estimating changes in image appearance. *Computer Vision and Image Understanding*, 78(1):8–31.

Black, M. J., Yacoob, Y., and Fleet, D. J. (1997a). Modeling appearance change in image sequences. In *3rd International Workshop on Visual Form*.

Black, M. J., Yacoob, Y., Jepson, A. D., and Fleet, D. J. (1997b). Learning parameterized models of image motion. In *IEEE Computer Vision and Pattern Recognition*, pages 561–567.

Bollacker, K. D. and Ghosh, J. (1998). A supra-classifier architecture for scalable knowledge reuse. In *International Conference on Machine Learning*, pages 64–72.

Brainard, D. H. and Freeman, W. T. (1997). Bayesian color constancy. *Journal of the Optical Society of America, A*, 14(7):1393–1411.

Buchsbaum, G. (1980). A spatial processor model for object color perception. *Journal of the Franklin Institute*, 310.

Cardei, V. C., Funt, B. V., and Barnard, K. (1997). Modeling color constancy with neural networks. In *Proceedings of the International Conference on Vision, Recognition, and Action: Neural Models of Mind and Machine*.

Christensen, G. E. (1999). Consistent linear-elastic transformations for image matching. In Kuba, A., Samal, M., and Todd-Pokropek, A., editors, *Information Processing in Medical Imaging, Lecture Notes in Computer Science 1613, 16th International Conference*, pages 224–237. Springer.

Cootes, T. F., Edwards, G. J., and Taylor, C. J. (2001). Active appearance models. *IEEE Pattern Analysis and Machine Intelligence*, 23(6):681–685.

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

Darrell, T. and Pentland, A. (1991). Robust estimation of a multi-layer motion representation. In *Proceedings of the IEEE Workshop on Visual Motion*.

Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer Verlag.

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

Edelman, S. and O'Toole, A. J. (2001). Viewpoint generalization in face recognition: The role of category-specific processes. In Wenger, M. J. and Townsend, J. T., editors, *Computational, geometrical, and process perspectives on face cognition*, pages 397–428. Lawrence Erlbaum Assoc., Inc.

Fleet, D. J., Black, M. J., Yacoob, Y., and Jepson, A. D. (2000). Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):171–193.

Forsyth, D. A. (1990). A novel algorithm for color constancy. *International Journal of Computer Vision*, 5(1).

Frey, B. and Jojic, N. (1999a). Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In *IEEE Computer Vision and Pattern Recognition*, pages 416–422.

Frey, B. and Jojic, N. (1999b). Transformed component analysis: Joint estimation of spatial transformations and image components. In *International Conference on Computer Vision*.

Gordon, G., Darrell, T., Harville, M., and Woodfill, J. (1999). Background estimation and removal based on range and color. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Grenander, U. (1963). *Probabilities on Algebraic Structures*. John Wiley & Sons.

Grother, P. (1995). NIST special database 19 handprinted forms and characters database. Technical report, National Institute of Standards.

Horn, B. K. P. (1986). *Robot Vision*. MIT Press.

Horprasert, T., Harwood, D., and Davis, L. S. (2000). A robust background subtraction and shadow detection. In *Proceedings of the Asian Conference on Computer Vision, 2000*.

Hüppi, P., Warfield, S., Kikinis, R., Barnes, P. D., Zientara, G. P., Jolesz, F. A., Tsuji, M. K., and Volpe, J. J. (1998). Quantitative magnetic resonance imaging of brain development in premature and mature newborns. *Annals of Neurology*, 43:224–235.

Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the hausdorff distance. *IEEE Pattern Analysis and Machine Intelligence*, 15(9):850–863.

Jensen, F. (1996). *An Introduction to Bayesian Networks*. Springer.

Jepson, A. and Black, M. J. (1993). Mixture models for optical flow. In *IEEE Computer Vision and Pattern Recognition*, pages 760–761.

Jones, M. and Poggio, T. (1995). Model-based matching by linear combinations of prototypes. Technical Report AI Memo 1583, Massachusetts Institute of Technology.

Lando, M. and Edelman, S. (1995). Generalization from a single view in face recognition. Technical Report CS-TR 95-02, Weizmann Institute of Science.

LeCun, Y., Bottou, L., and Bengio, Y. (1997). Reading checks with graph transformer networks. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 151–154, Munich. IEEE.

LeCun, Y., Jackel, L. D., Bottou, L., Brunot, A., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., and Vapnik, V. (1995). Comparison of learning algorithms for handwritten digit recognition. In Fogelman, F. and Gallinari, P., editors, *International Conference on Artificial Neural Networks*, pages 53–60.

Lenz, R. and Meer, P. (1997). Illumination independent color image representation using log-eigenspectra. Technical Report LiTH-ISY-R-1947, Linköping University.

Lien, J. J. (1998). *Automatic Recognition of Facial Expressions Using Hidden Markov Models and Estimation of Expression Intensity*. PhD thesis, Carnegie Mellon University.

Maloney, L. T. (1986). Evaluation of linear models of surface spectral reflectance with small numbers of parameters. *Journal of the Optical Society of America*, A1.

Marimont, D. H. and Wandell, B. A. (1992). Linear models of surface and illuminant spectra. *Journal of the Optical Society of America*, 11.

Matsakis, N. (2000). A system for the recognition of handwritten mathematics. Master's thesis, Massachusetts Institute of Technology.

McCann, J. J., Hall, J. A., and Land, E. H. (1977). Color mondrian experiments: The study of average spectral distributions. *Journal of the Optical Society of America*, A(67).

Miller, E., Matsakis, N., and Viola, P. (2000). Learning from one example through shared densities on transforms. In *IEEE Computer Vision and Pattern Recognition*.

Miller, E. and Tieu, K. (2001a). Color eigenflows: Statistical modeling of joint color changes. In *International Conference on Computer Vision*, volume 1, pages 607–614.

Miller, E. and Tieu, K. (2001b). Lighting invariance through joint color change models. In *Proceedings of Workshop on Identifying Objects Across Variations in Lighting: Psychophysics and Computation at CVPR 2001*.

Miller, E., Tieu, K., and Stauffer, C. P. (2001). Learning object-independent modes of variation with feature flow fields. Technical Report AIM-2001-021, MIT Artificial Intelligence Laboratory Memorandum.

Misner, C. W., Thorne, K. S., and Wheeler, J. A. (1973). *Gravitation*. W. H. Freeman and Company.

Moses, Y., Ullman, S., and Edelman, S. (1996). Generalization to novel images in upright and inverted faces. *Perception*, 25:443–462.

Ng, A. and Jordan, M. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advanced in Neural Information Processing Systems 14*.

Parzen, E. (1962). On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076.

Pennec, X. (1996). *L'Incertitude dans les Problèmes de Reconnaissance et de Recalage: Application en Imagerie Médicale et Biologie Moléculaire*. PhD thesis, École Polytechnique, Paris, France.

Pratt, L. and Jennings, B. (1998). A survey of connectionist network reuse through transfer. In Thrun, S. and Pratt, L., editors, *Learning to learn*. Kluwer Academic Publishers.

Schutz, B. (1980). *Geometrical Methods of Mathematical Physics*. Cambridge University Press.

Sharkey, N. E. and Sharkey, A. J. C. (1993). Adaptive generalization. *Artificial Intelligence Review*, 7:313–328.

Shashua, A. and Riklin-Raviv, R. (2001). The quotient image: Class-based re-rendering and recognition with varying illuminations. *IEEE Pattern Analysis and Machine Intelligence*, 3(2):129–130.

Simard, P., LeCun, Y., and Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In *Advanced in Neural Information Processing Systems 5*, pages 51–58.

Simard, P., LeCun, Y., Denker, J., and Victorri, B. (1998). Transformation invariance in pattern recognition - tangent distance and tangent propagation. In Orr, G. and Muller, K.-R., editors, *Neural Networks: Tricks of the Trade*. Springer.

Soriano, M., Marszalec, E., and Pietikainen, M. (1999). Color correction of face images under different illuminants by rgb eigenfaces. In *Proceedings of the Second International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 148–153.

Spivak, M. (1999a). *A Comprehensive Introduction to Differential Geometry, Volume I, 3rd Edition*. Publish or Perish Press.

Spivak, M. (1999b). *A Comprehensive Introduction to Differential Geometry, Volumes I-V, 3rd Edition*. Publish or Perish Press.

Stauffer, C. and Grimson, E. (2001). Similarity templates for detection and recognition. In *IEEE Computer Vision and Pattern Recognition*.

Stauffer, C., Miller, E., and Tieu, K. (2002). Transform-invariant image decomposition with similarity templates. In *Advanced in Neural Information Processing Systems 14*.

Stiles, W. S., Wyszecki, G., and Ohta, N. (1977). Counting metameric object-color stimuli using frequency limited spectral reflectance functions. *Journal of the Optical Society of America*, 67(6).

Thrun, S. (1996). *Explanation-based neural network learning: A lifelong learning approach*. Kluwer.

Thrun, S. and Pratt, L. (1998). Learning to learn: introduction and overview. In Thrun, S. and Pratt, L., editors, *Learning to learn*. Kluwer Academic Publishers.

Toyama, K., Krumm, J., Brumitt, B., and Meyers, B. (1999). Wallflower: Principles and practice of background maintenance. In *IEEE Computer Vision and Pattern Recognition*, pages 255–261.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cogntive Neuroscience*, 3(1):71–86.

Vapnik, V. (1998). *Statistical Learning Theory*. John Wiley & Sons.

Vasconcelos, N. and Lippman, A. (1998). Multiresolution tangent distance for affine-invariant classification. In *Advanced in Neural Information Processing Systems 10*.

Vetter, T., Jones, M., and Poggio, T. (1997). A bootstrapping algorithm for learning linear models of object classes. In *IEEE Computer Vision and Pattern Recognition*, pages 40–46.

Viola, P. and Wells III., W. M. (1997). Mutual information: An approach for the registration of object models and images. *International Journal of Computer Vision*.

Warfield, S., Rexilius, J., Hüppi, P., Inder, T., Miller, E., Wells, W., Zientara, G., Jolesz, F., and Kikinis, R. (2001). A binary entropy measure to assess nonrigid registration algorithms. In *Medical Image Computing and Computer-Assisted Intervention*.

Woody, C. D. (1967). Characterization of an adaptive filter for the analysis of variable latency neuroelectric signals. *Medical and Biological Engineering*, 5:539–553.