An Information Theoretic Hierarchical Classifier for Machine Vision

by

Michael J. Andrews

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Electrical and Computer Engineering
by

May 1999

APPROVED:

Professor David Cyganski, Major Advisor

Professor Denise W. Nicoletti

Professor Michael A. Gennert

Abstract

A fundamental problem in machine vision is the classification of objects which may have unknown position, orientation, or a combination of these and other transformations. The massive amount of data required to accurately form an appearance-based model of an object under all values of shift and rotation transformations has discouraged the incorporation of the combination of both transformations into a single model representation.

This Master's Thesis documents the theory and implementation of a hierarchical classifier, named the Information Theoretic Decision Tree system, which has the demonstrated ability to form appearance-based models of objects which are shift and rotation invariant which can be searched with a great reduction in evaluations over a linear sequential search. Information theory is utilized to obtain a measure of information gain in a feature space recursive segmentation algorithm which positions hyperplanes to local information gain maxima. This is accomplished dynamically through a process of local optimization based on a conjugate gradient technique enveloped by a simulated annealing optimization loop. Several target model training strategies have been developed for shift and rotation invariance, notably the method of exemplar grouping, in which any combination of rotation and translation transformations of target object views can be simulated and folded into the appearance-based model. The decision tree structured target models produced as a result of this process efficiently represent the voluminous training data, affording rapid test-time classification of objects.

Acknowledgements

To my family: Mom, Dad, and Karen, who taught me to laugh.
Yes, I made the cover first, and no, Freddie would *not* have made it this far.

To my gurus: Professor David Cyganski and Professor Rich Falco, who showed me the way.
I have no idea how you put up with me for so long.

To my machine vision lab colleagues: Murti Amiji, Mike Driscoll, Deb Fraser, Brian Hazzard, Witek Jachimzcyk, Carleton Jillson, Jeremy Johnstone, Jim Kilian, Brent Modzelewski, Sean Price, Mike Roberts, Nandan Sinha, Spiderman, Pedro Soria-Rodríguez, and John Sullivan, who made me laugh.
I do hope you aren't scarred for life.

To my companion: Kim, who make fun of me whether I needed it or not.
Everyone liked the cookies.

To my predecessors: Jim Kilian, Sergey Perpelitsa, and Ryan Tomasetti, who allowed me to stand on their shoulders.
You did a lot of the hard stuff.

To my thesis committee: Professor David Cyganski, Professor Michael Gennert, and Professor Denise Nicoletti, who took on this burden in the final hour.
Your questions and comments added a completeness to this thesis which I couldn't have imagined. This is appreciated more than *you* could imagine.

To my sponsors, past and present: Textron Systems Division, Data Translation and the Army Research Office, who funded this work.

Sergey and Jim conducted the initial thrust of this work under ARO Grant # DAAH 04-95-1-0631. Ryan and I both worked under funding from Textron. Data Translation funded the current work. I can only hope you are happy with the results.

I can't believe I finished it.

Michael J. Andrews (`rocko`)
May, 1999

Contents

List of Figures

Chapter 1

Introduction and Background

> When one meets the concept of entropy in communication theory, he has a right to be rather excited – a right to suspect that one has hold of something that may turn out to be basic and important. [25, p.103]
>
> – Warren Weaver in *The Mathematical Theory of Communication*

In the fifty years which passed since Claude Elwood Shannon and Warren Weaver wrote their seminal masterpiece, information theory emerged as a unique discipline with far-reaching impact for all of engineering and science. Humbly presented as a framework for quantifying information and uncertainty in communication systems, Shannon's theory offered an intoxicating blend of mathematical rigor and conceptual naturalness found only in papers of fundamental importance. Among modern engineers, the measure of uncertainty, *entropy,* and the measure of information content, *mutual information,* rose to a nearly philosophical plane, influencing all communications systems designed in the second half of the twentieth century. The coming of the terms information theory and the now ubiquitous unit of binary information, the *bit*, signaled the start of the Information Age.

Pioneering philosophers on machines and thought suggested that machines exhibiting human-like intelligence would be forthcoming deliverables for the Information Age. Isaac Asimov's Three Laws of Robotics were first published in a short story[1] in 1942, from which robots emerged in the public eye for the first time as helpful machines, markedly orthogonal to the metal monsters found in the fiction of the day. And Alan Turing first proposed what has come to be known as the *Turing test* for machine intelligence in a 1950 paper[32]. In

[1]The Three Laws of Robotics were originally published in "Runaround" in *Astounding Science Fiction*, March, 1942. It can be argued that their widespread recognition was delayed until Asimov published a series of "Robots" novels in the 1950's

Turing's test, a machine and a human interact with a jury over teletype terminals; for a machine to pass, it must appear at least as human-like as the human it is pitted against. It was believed that intelligent machines would not only replace humans performing dangerous or repetitive tasks, but also aid us in solving the most difficult problems of the day. Smart machines would go where humans couldn't and do what humans didn't want to do. By all accounts, such machines have been slow to materialize. Late 20th century science fiction author and Information Age pundit Bruce Sterling telegraphs this sentiment:

> A faithful reader of SF from the 1940s and '50s might be surprised to learn that we're not hip-deep in robots by now. By this time, robots ought to be making our breakfasts, fetching our newspapers, and driving our atomic-powered personal helicopters. But this has not come to pass, and the reason is simple.
>
> We don't have any robot brains.[27]

Constructing artificial brains, be they destined for robots or not, has proven a troublesome task. When the study of artificial intelligence began in earnest nearly a half century ago, researchers couldn't have dreamed of the countless theories and systems which have sprouted like weeds in AI's once pristine landscape. Now, after fifty years, the surface of the machine intelligence problem has been scratched, but is far from cracked. In hopes to aid progress, artificial intelligence has been segmented down into a small number of distinct areas for study. Each of five human senses has attracted attention, although olfactory and taste sensory models have been slow to proliferate. But, perhaps due to the human propensity for the visual sense, it is arguably the area of machine vision which has amassed the greatest interest.

Machine vision systems, in general, respond to visual stimuli in a way which helps solve a certain problem. A typical machine vision system (figure 1.1) accepts input from a camera and outputs some form of distilled information or decision. Used in this manner, the term camera is a rather general description for what may be a stereo pair of cameras, a forward-looking infrared (FLIR) sensor, a black and white security camera, or any number of other visual transducers. The information output ranges from a simple binary pass/fail test to spatial positions of objects and motion vectors. Such machines, which can quickly recognize or understand visually observed environments are in great demand for numerous industrial and military applications:

- **Visual inspection and quality control for assembly** - high speed cameras, pass/fail decisions.

- **Guidance systems for smart munitions** - FLIR sensors, RADAR, target tracking and detection.
- **Vehicle navigation, automated control and driving** - stereo cameras, motion vector and object avoidance decisions.
- **Optical character recognition, handwriting recognition** - conventional cameras, touch sensitive surfaces, forgery detection and character identification.

Figure 1.1: A simple machine vision system flow model..

Historically, machine vision systems which purportedly offer good performance in one of the above categories have not done so by modeling the way in which the human brain might perform the same tasks. This is, in part, because the brain isn't well understood, and in part because the way which it is believed humans interpret visual scenes doesn't translate well to algorithmic implementations on digital computers. Demand for machine vision systems which produce reliable, timely results has created a design methodology differing from attempts at creating a general-purpose human-like machine brain. Machine vision systems which aim to be more than intellectual curiosities aren't based on cognitive models, relying instead on a foundation of mathematics and signal theory. Good performance is achieved by exploiting the strengths of the modern digital computer: storage with quick recall and fast mathematical operations. In particular, algorithms which translate images into geometric structures are readily handled by computers.

Any transformation of an input scene image can be used to form an abstract model of an object, against which a test scene can be compared to locate or determine the identity of particular objects. This technique, called model-based object recognition, folds-in important features of an object or objects to form a complete model, which is searched or compared in some way to a test image. Two fundamental classes comprise model-based object recognition algorithms: feature-based and appearance-based. Typically, feature-based recognition systems employ a detailed geometric representation of an object. Systems of this type find objects by matching key geometric structures in the model to similar structures found in the test image. The classic example of a feature-based recognition system

is a vehicle locator which searches for round wheels under a rectangular body, with some number of additional rectangles matching to the windows. Alternately, appearance-based recognition systems use a catalog of views or images of an object to form a complete description of the object. An appearance-based vehicle locator would search through a set of images of a car taken from viewing angles over 360° around the car.

When a machine vision system produces a decision regarding which entry in a target model best corresponds to part or all of the input test image, the system does so by means of a *classifier.* A machine vision classifier operates in conjunction with the model-based mechanism to determine a distinct target class which best matches the object under test. Specifically, the classifier is the component of a system which generates a decision - all preprocessing required to transform a test scene image to the same means of representation as used in the model are performed beforehand. Detection theory, which is applied to the design of communication systems used to extract information from signals which have been corrupted by noise, can also be applied to the machine vision problem. Using the language of *pattern recognition,* one can place hyperplanes in a decision space to isolate vector representations of structures in the model. Systems constructed in this manner tend to be extremely robust. The military application of machine vision, automatic target recognition (ATR), has an expansive body of literature documenting this signal theory approach to object recognition.

The communication system theory perspective on machine vision affords a certain mathematical rigor not present from other viewpoints. In the same way in which information theory is applied to characterize a communication channel, it can be applied to characterize the theoretical performance of a machine vision classifier. One can model a classifier as a communication channel with a capacity given by the maximum of the mutual information between the input and output data. In doing so, a natural measure of goodness-of-fit of the classifier to the target data is found.

This Master's thesis documents the design, implementation and testing of a model-based machine vision system which employs information theory in its classifier design. In this system, called the Information Theoretic Decision Tree (ITDT), a set of images of an object comprise an appearance-based model upon which an order is imposed by a hierarchical classifier constructed by maximizing the mutual information at each successive stage in the hierarchy. The system offers rapid test times by shifting the computational burden to an off-line information theoretic model decision tree construction process, a structure which

offers a great reduction in the number of target model exemplars which must be searched (figure 1.2).

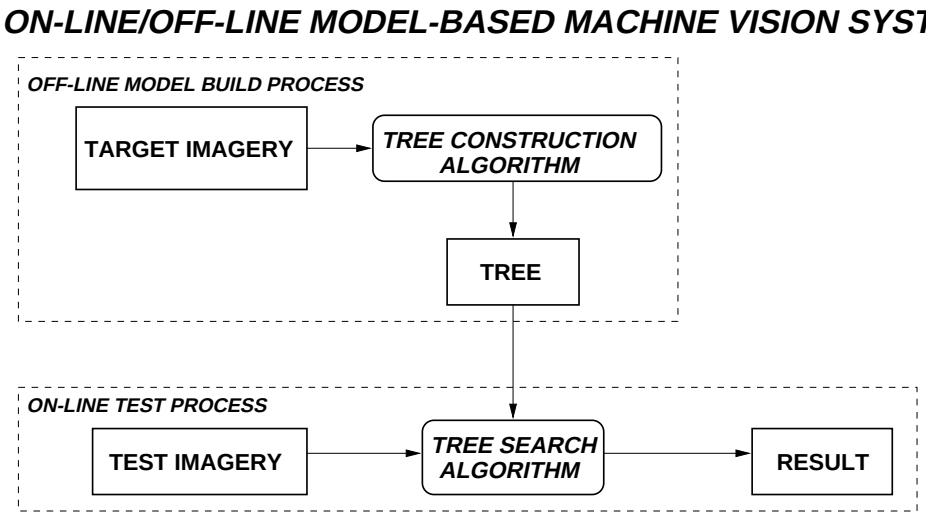

Figure 1.2: The Information Theoretic Decision Tree system.

The following sections in this chapter constitute the background material necessary for subsequent detailed exploration of the system: model-based machine vision, binary decision trees, pattern recognition and information theory as applied to pattern recognition. An exposition on the original contributions of the author to this problem and acknowledgement of past work done on the ITDT system are reserved until the end of this chapter. Chapter 2 is a detailed exposition on the theoretical foundation of the ITDT system. Chapter 3 documents the author's original contribution of exemplar grouping. Chapter 4 presents several training strategies for shift and rotation independence in the ITDT system, which make use of the method of exemplar grouping. Results for a variety of test cases are reported throughout the text.

1.1 Model-Based Machine Vision

Model-based machine vision operates under the deterministic ideal that given an exhaustive model of any particular object, it can be recognized. The eminent mathematician Pierre Simon de Laplace is one of the earliest progenitors of a related belief:

> An intellect which at any given moment knew all the forces that animate Nature and the mutual positions of the beings that comprise it, if this intellect were vast enough to submit its data to analysis, could condense into a single formula the movement of the greatest bodies of the universe and that of the lightest atom: for such an intellect nothing could be uncertain; and the future just like the past would be present before its eyes.[28, p.10] [2]

A model-based machine vision system is, in general, any system which uses a relatively complete representation of any given object to recognize it. Objects can be modeled in this way geometrically, by distinct features, or with textures, for example. The method which is the focus for the ITDT system development employs a series of views or images of the object to form an appearance-based model.

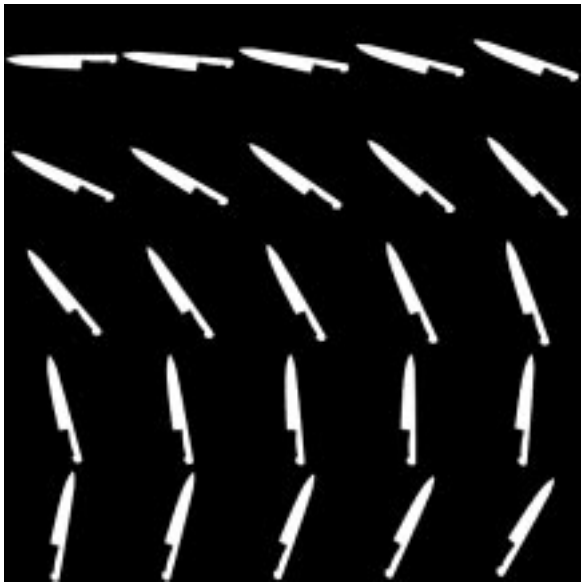

Figure 1.3: Rotation model of a knife silhouette.

[2] Originally stated in Laplace's *Philosophical Essays on Probabilities.* Quotation from Stewart[28].

Figure 1.3 shows 25 views of a knife silhouette at various rotations. In a simple testing scenario, a test image which has been acquired through a camera is compared to each image in the model, producing a *goodness of fit* metric. In the list of reported metrics, the maximum corresponds to the entry in the model database which most likely corresponds to the test image. However, in practice, testing a digital image against each element of a complete model is a *global search problem*. These tests can rarely be implemented, considering the storage limitations and test time requirements inherent to every machine vision problem. For this reason, nearly all research into model-based recognition has been in the development of systems which somehow reduce the amount of data to be searched, thereby increasing the feasibility of the model-based approach.

1.2 The Promise of Hierarchical Classifiers

Binary decision trees have long been a standard method of organizing data for efficient searching, and with good reason: the number of comparison operations required to locate a match is reduced to $\log_2$ of the number of comparisons required for a worst-case sequential search of the same data. A simple example best illustrates this.

Suppose a data retrieval system were to be designed to search through a list of names and associated personal data, locating the entry corresponding to a name provided by the user. A sequential search through the data (Figure 1.4) is the simplest method to implement. Each entry in the list is sequentially tested against the name for which the algorithm is searching, in this case, "Murti". If the name matches, the algorithm stops and returns the entry, if not, the search proceeds to the next entry.

Sequential searches perform an average of $N/2$ comparison operations to locate an entry in a list of N equally probable items. In this example, if the name "Brent" were sought (probability $1/N = 1/13$), a sequential search algorithm would return after only one comparison. Unfortunately, equally likely is the name "Witek" at the end of the list, which would be reached in thirteen comparisons. Therefore, on average, sequential searches through this list of 13 names will return after 6.5 comparisons. Moreover, although the list of names is in alphabetical order, a strictly sequential search doesn't make use of ordering information. The list doesn't have to be arranged in any particular order, and sequential searches will always require an average of $N/2$ comparison operations to locate a match.

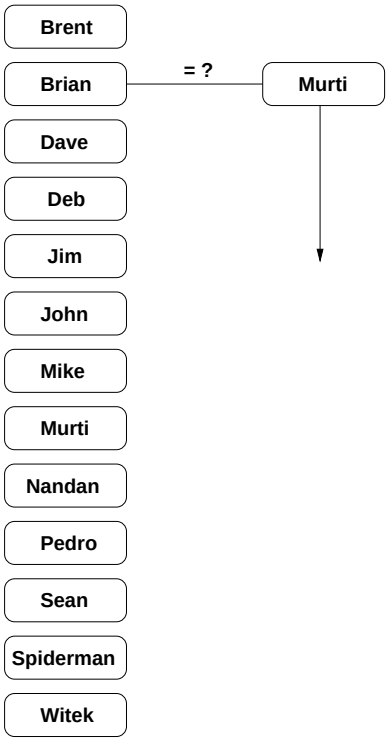

Figure 1.4: Sequential search through a list of names.

We can use binary decision trees if we can impose a hierarchy on any ordered list, reducing the number of comparison operations exponentially.

Figure 1.5 is a binary decision tree that can be used to find entries in the example list. The tree is comprised of a number of circular and rectangular nodes, connected by *branches* (Figure 1.6 is a guide to binary tree terminology.) A search algorithm *traverses* the tree by starting at the top and progressing either to the right or left branch of every circular node, eventually reaching a rectangular *leaf*, which is the closest match to the search name. Each circular node in the binary tree contains a name which is compared alphabetically to the name for which it is searching. When a search is invoked, the algorithm starts by *visiting* the topmost circular node of the tree, performing the following actions:

1. Alphabetically compare the search name to the name stored in the tree's node.
2. If the search name is found to precede the name stored in the node, continue down the left *branch.*
3. Otherwise, continue down the right branch.

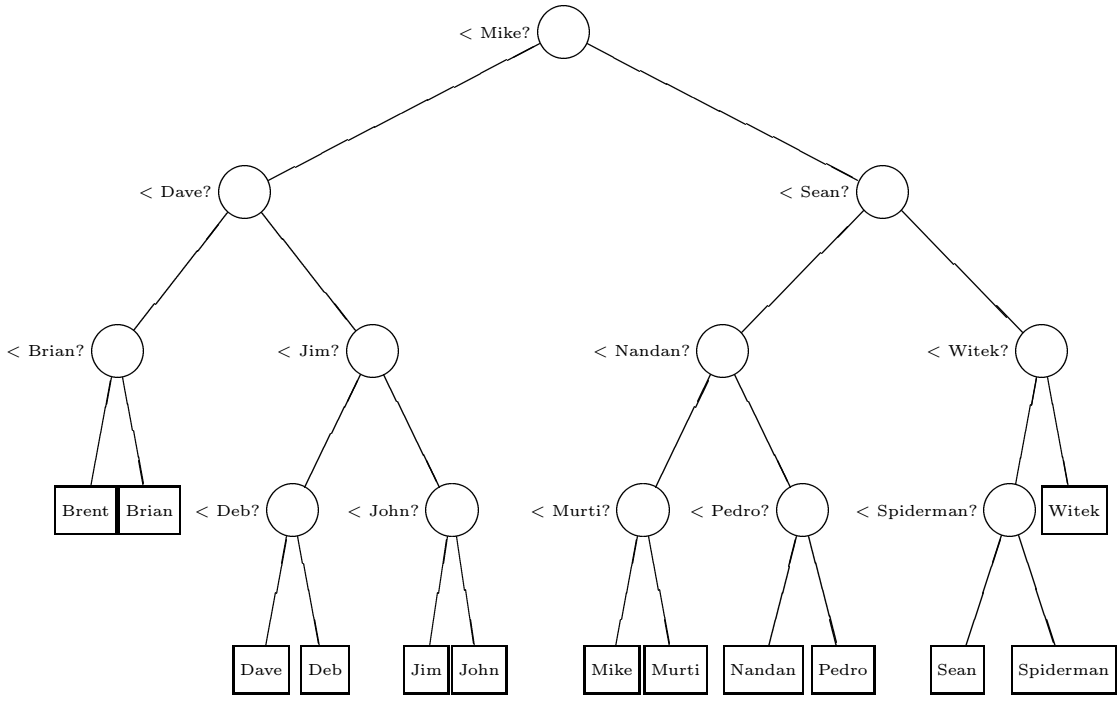

Figure 1.5: Binary decision tree reduces average number of comparisons for searches.

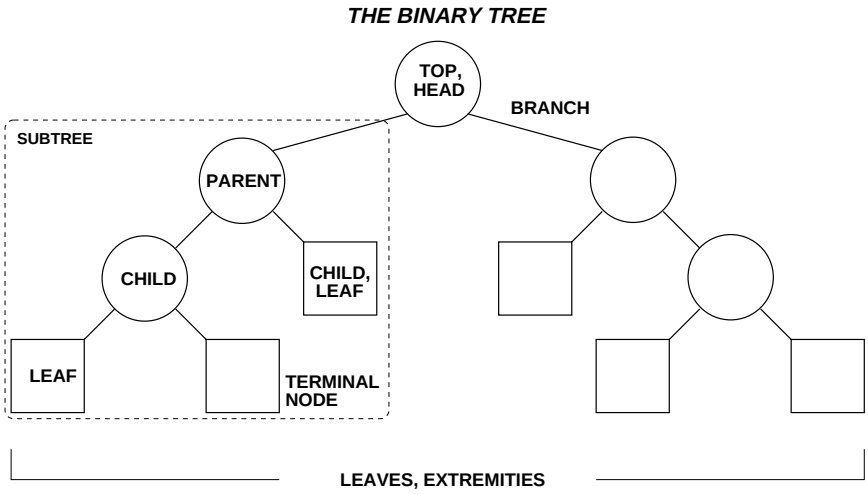

Figure 1.6: Common terms used to describe parts of a binary tree.

The tree is *traversed* in this manner, repeating the above process for each node visited. Returning to the example, a search for the name "Murti" involves the following four decision steps:

1. Q: Does "Murti" precede "Mike"? A: No. Visit the node to the right.
2. Q: Does "Murti" precede "Sean"? A: Yes. Visit the node to the left.
3. Q: Does "Murti" precede "Nandan"? A: Yes. Visit the node to the left.
4. Q: Does "Murti" precede "Murti"? A: No. Visit the node to the right.

Therefore, in the case of the name "Murti," this decision tree arrives at the result in four comparisons, whereas a sequential search would take eight comparisons. Three of the names require only three decisions, while the remaining ten names can be arrived at in four decisions, an average of $[(3)(3) + (10)(4)]/2 = 3.8$ decisions for any one name, and nearly half of the 6.5 decisions required in a sequential search. Optimally, binary trees used in this manner for searches approach search paths of length $\log_2 N$ - for this case, $\log_2 13 = 3.7$, indicating that this tree is nearly optimal.

In addition to the great computational speed benefit which can be gained by searching in a binary tree structure, the trees themselves often reveal underlying structure in the data they represent. Medical science, for years, has employed binary decision trees in patient diagnosis - finding a sequence of yes/no questions translates readily to their graphical structure, as seen in Figure 1.7, based upon a figure from *Classification and Regression Trees* [4, p.2]. And, applying the ITDT system to a variety of specific problems in machine vision has often resulted in a tree that points directly at a shortcoming in the training data or in the phrasing of the vision problem, as illustrated in the examples in chapters 2, 3, and 4.

In general, machine vision object models are not orderable. Although hierarchy is easily imposed on a set of names using alphabetical ordering, there exists no such clear set of rules for ordering sets of images or geometric representations. The information theoretic decision tree system formulates the ordering of a set of model views from an information theoretic standpoint, without any risk of errors. The details of this are discussed in chapter 2.

For an early test of the information theoretic decision tree system, a tree was built to discriminate among 53 poses of a Soviet T-72 tank. This particular test, called *pose identification*, doesn't identify which *vehicle* from a set of vehicles might be in a test image - instead, it asks, "If there is a T-72 at this location in this image, what direction is it facing?" Twenty five of the 53 poses are shown in Figure 1.8.

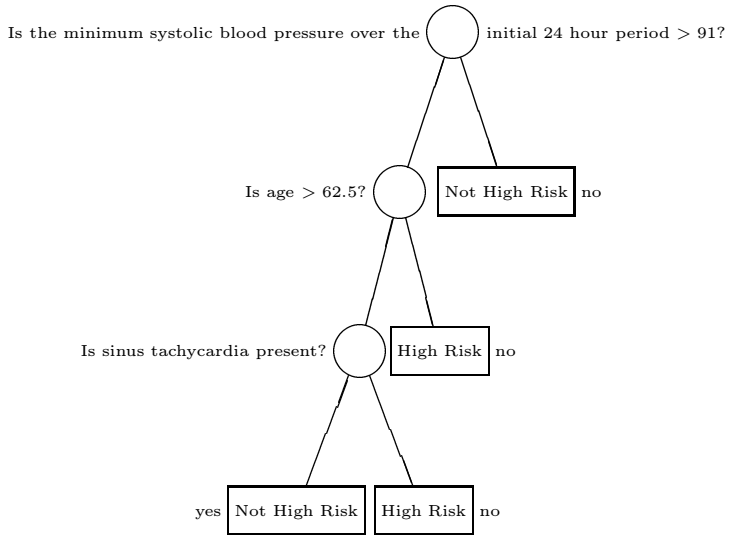

Figure 1.7: Decision tree for identification of high-risk heart attack conditions in patients.

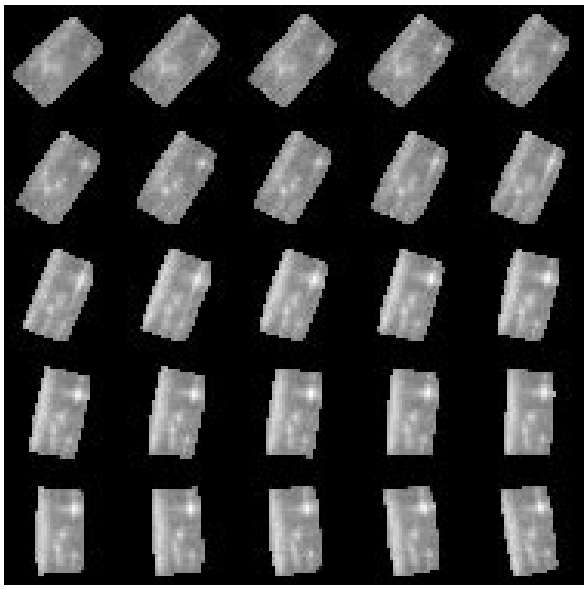

Figure 1.8: Twenty five poses of a T-72 tank.

The decision tree which was built by the ITDT system is shown in Figure 1.9. It requires an average of 5.8 decisions, with a maximum 6 decisions to correctly identify the pose. Compare this with the average of $53/2 = 26.5$ comparison operations needed to perform a sequential search over the same 53 poses. Notably, the binary tree is nearly optimal, as $\log_2 53 = 5.7$ decisions.

Pattern recognition provides a methodology for performing tests such as pose recognition, a language and method for modeling how the ITDT system orders images of target objects. A brief introduction to the science is presented in the following section, prior to a discussion of a rudimentary information theoretic machine vision system.

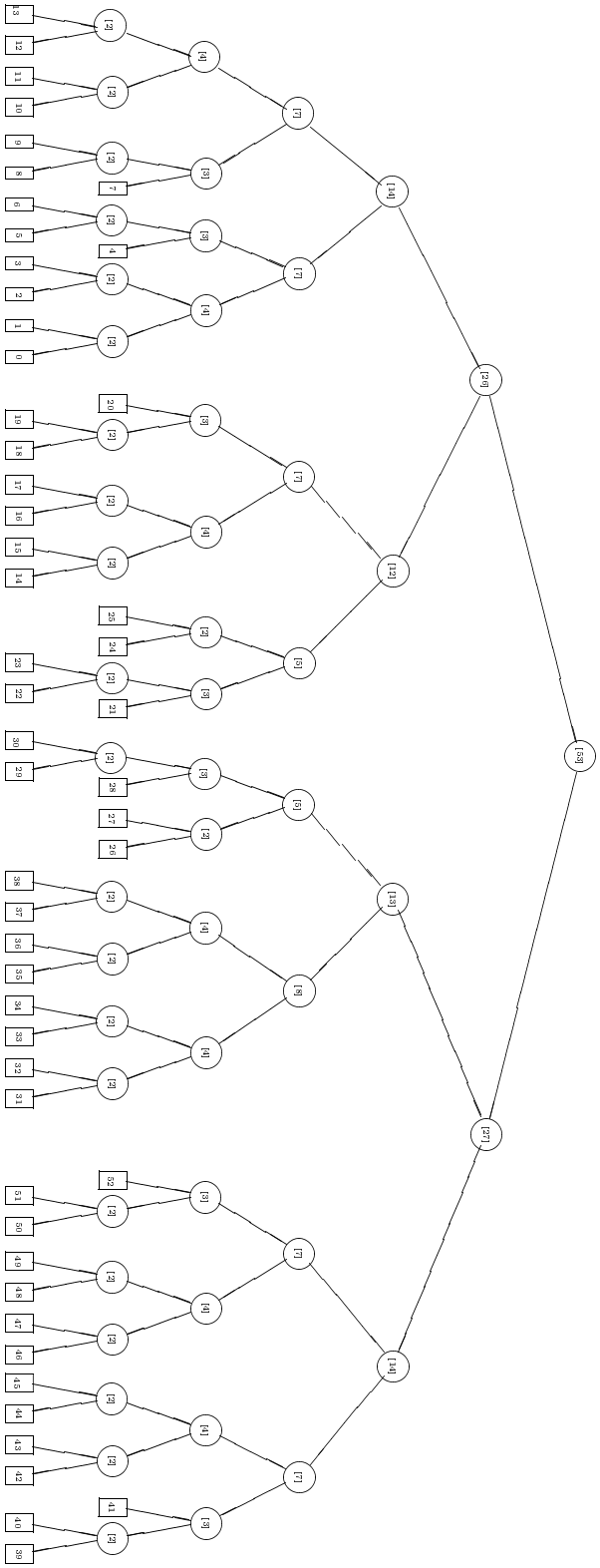

Figure 1.9: Binary decision tree built by the ITDT system to identify 53 poses of a Soviet T-72.

1.3 Pattern Recognition for Machine Vision

Pattern recognition has proven to be an effective tool in visualizing and solving problems in machine vision. To a measurable precision, any signal can be represented by an arbitrarily detailed pattern of data points organized into classes of patterns. Systems which perform pattern recognition simply seek to determine to which class, among a set of predetermined classes, a set of data belongs. By abstracting the problem of detection or recognition to the *pattern* level, these systems are able to solve problems in seemingly unrelated modalities - whether the patterns are the time series of temperature readings taken on a household thermometer or intensities of pixels in a digital image, they are dealt with in the same manner. The information theoretic decision tree (ITDT) is fundamentally a pattern recognition system. Although certain digital image specific enhancements are documented, there is no reason why an ITDT could not be developed to classify radar range data or population statistics, for example.

1.3.1 Geometry of Pattern Recognition

Pattern recognition techniques powerfully employ geometry to solve recognition problems and generate insight into their structure. Fundamentally, the terminology is centered around four geometric constructs: the feature space, the decision region, the decision boundary, and the feature vector (Figure 1.10.) Although a detailed treatment is beyond the scope of this report, Duda and Hart[6] is an excellent reference for fundamental concepts in feature vector-based classification.

From a set of N data values which comprise a particular pattern, one can form a N dimensional feature vector, $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ which represents the pattern as a single point in $N-$dimensional space. The vectors identified by "X" in Figure 1.10 are examples of two-dimensional vectors, each component representing some measure of a process.

Given a set of patterns, or feature vectors, a classifier can be designed which divides the *feature space* into regions which correspond to sets of feature vectors. In this way, patterns can be classified by determining geometrically in which region of the feature space they lie. Although the *decision boundary* created by a classifier can be a very complex high dimensional structure, it is frequently simplified and represented as an optimally-placed *hyperplane.*

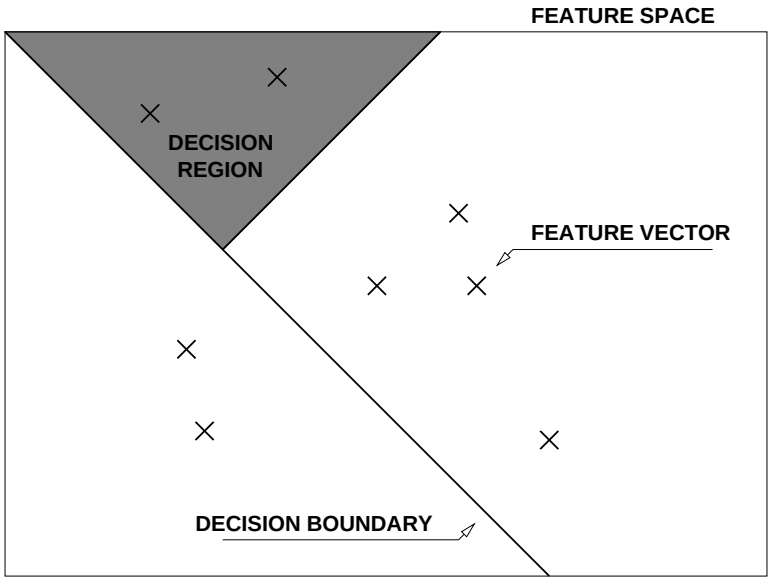

Figure 1.10: Terminology of Pattern Recognition Systems.

A hyperplane is the N-space equivalent of a 2-dimensional plane in 3-space. Hyperplanes always have one less dimension than the space in which they exist; in a 2D space, a hyperplane is merely a line, and in a 1D space, a hyperplane is reduced to a point. Perhaps more importantly, hyperplanes divide the space in which they lie into two regions just as a point divides a line into two regions, a line divides a plane and a plane divides three-dimensional space. In this way, a hyperplane acts as an unambiguous slice through a set of data points; in any space points either lie to one side or the other of the hyperplane.

By way of example, Figure 1.11 shows the feature space and decision boundary for a two hypothesis classifier. Each hypothesis is characterized as a symmetrical 2D Gaussian distribution: the first with a mean at (5,5), the second with a mean at (15,15). Dividing the 2D feature space is a decision boundary derived from a Bayes likelihood ratio test (LRT), a result from classical detection theory[30, pp.24-27] proven to minimize the cost of decisions made by the classifier. The boundary used in Figure 1.11 is derived in appendix A, along with the more general result for symmetric N-dimensional Gaussian distributions in appendix B.

In a purely analytic exercise, the representations of the two hypotheses do not require any kind of experimentally obtained data, as the mean and covariance matrix of the Gaus-

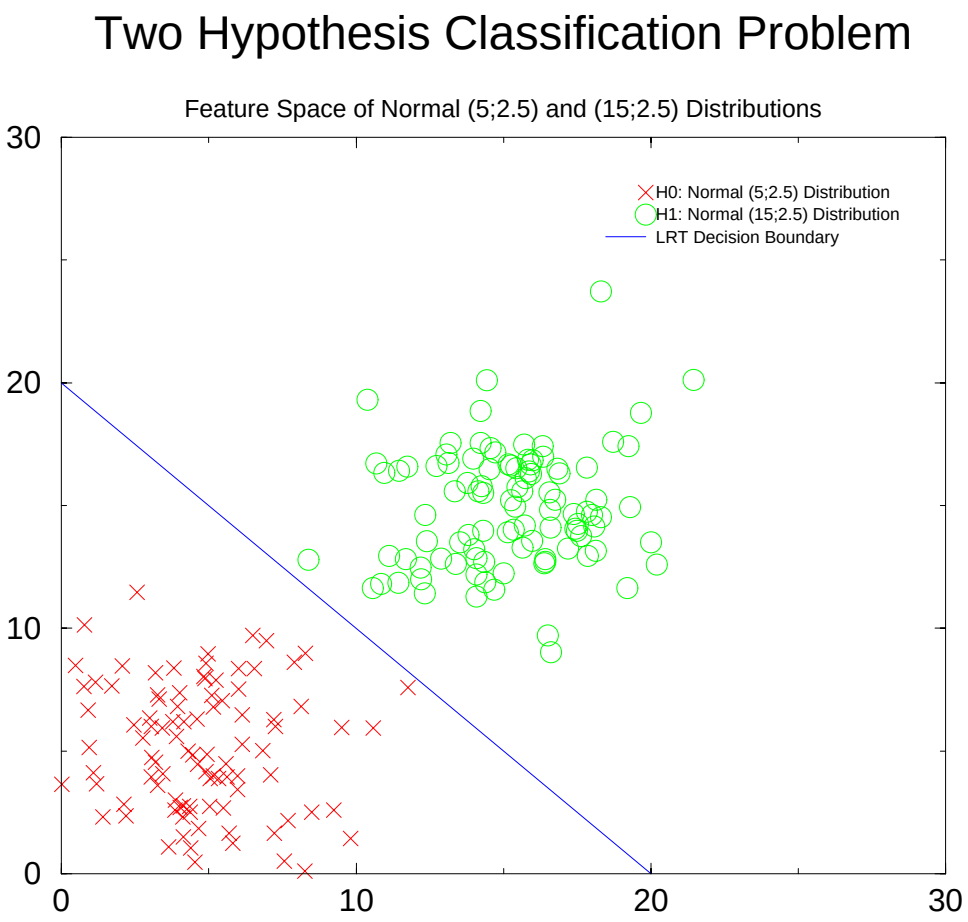

Figure 1.11: The feature space with decision boundary for a two hypothesis classifier.

sian distributions fully describe the problem and the unique solution. However, one can envision scenarios which, although contrived, might provide useful insight into the design and application of the data and solution shown in this plot. For instance, imagine that it was necessary to design a classifier to identify a test subject as one of two different people based on their commute times to and from work. For one hundred working days, the time it took for each person to drive to work and the time to drive home at night is recorded. If the X axis is assigned to the time to commute to work in the morning, and the Y axis is assigned to the evening return home trip time, then we can observe person A has about a five minute commute to work and person B has about a fifteen minute commute. Given only these data points, a classifier can be designed to compute a line which divides the feature space clearly into the area occupied by the feature vectors associated with person A and

the area occupied by feature vectors for person B. When supplied with a new data point, say $\{2, 4\}$, the data may be plotted as a feature vector and found to lie to the left of the decision boundary, within the person A region. In this case, the particular classifier used (to be explained in the following) proclaims the the data point was most likely generated by person A.

In the information theoretic decision tree system, a set of feature vectors form an appearance-based model of an object. Each feature vector is formed from a digital image, the pixels of which become the vector components in the feature vector, as shown in figure 1.12. Additionally, if one desires to remove the effect of varying light intensity on the overall image energy, one can normalize each feature vector to unit magnitude. When using this unit-magnitude constraint, the algorithm is modified to force hyperplanes to have zero offset so that they must pass through the origin, thus classifying feature vectors based on their orientation only, not their magnitude.

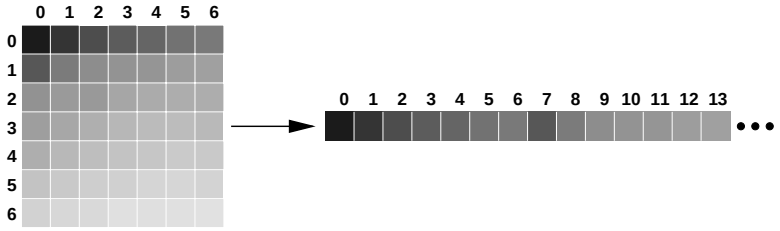

Figure 1.12: Feature vector components are the pixels in a digital image.

Fundamentally, the ITDT system is a pattern recognition system which has no notion of the origins of the feature vectors which it classifies. However, the feature vectors described throughout this report will always represent a digital image, unless otherwise stated. In Chapter 4, we consider a number of image-specific enhancements to the ITDT system.

In the section to follow, we consider a multiple-stage hierarchical classifier design. Based on a binary decision tree structure, hyperplanes are placed in each node so as to form binary classifiers which maximize mutual information through splitting a set of feature vectors into two classes. Each class formed in this manner is passed on to a node's child, where it is split again, stopping when a node contains only one feature vector.

1.4 Information Theoretic Approach to Hierarchical Classifier Design

Artfully employing decision trees in any system involves, above all, knowing what questions to ask at each internal node to split the data in a way which solves the problem in an efficient fashion. In Leo Breiman's seminal text *Classification and Regression Trees*, he cites three elements essential in building decision trees: (list from [4, p.22])

1. The selection of the splits
2. The decision when to declare a node terminal or to continue splitting it
3. The assignment of each terminal node to a class

Breiman goes on to state, "It turns out that the class assignment problem is simple. The whole story is in finding good splits and in knowing when to stop splitting."[4, p.23] As it happens, both the class assignments and the decision of when to stop splitting are trivial and linked together for the information theoretic decision tree. The ITDT system is designed to provide a decision framework for deciding among target classes by successively splitting the data. Each successive split reduces the data to be handled by the node's children, and the algorithm continues to divide the data until a node is found which only contains feature vectors belonging to one target class. Leaf nodes do not have an associated hyperplane, because there is no decision to be made for the node since it represents only one class. Conversely, all internal tree nodes have associated hyperplanes, since there is always a decision to be made at those nodes.

Each internal node of the decision tree contains a set of feature vectors, and a hyperplane which will be positioned to split the space appropriately. Both the hyperplane and targets (feature vectors) are represented explicitly as sets:

- $X = \{x_1, x_2, \ldots, x_N\}$ is the set of N target exemplars, represented as feature vectors. $P(X = x_1) = P(x_1)$ is the probability that any particular target is target x_1. In the following, targets are all assumed equally likely, so the random vector X is uniformly distributed and $P(x_i) = 1/N$ for all $i \in [1, \ldots, N]$.
- $Y = \{y_1, y_2\}$ are the sets of targets on the "left" side or the "right" side of the hyperplane, respectively. $P(y_1)$ is the probability that any target is on the left side.

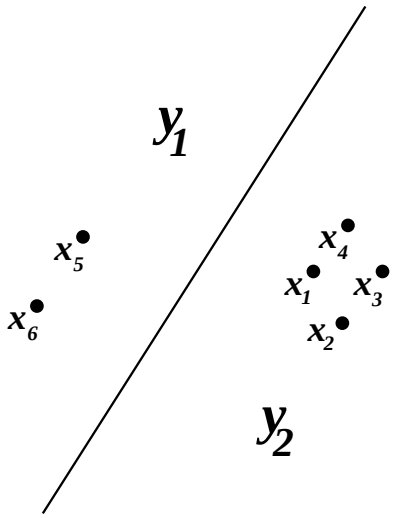

Figure 1.13: Representation of the decision space for a two-dimensional, six target problem.

Figure 1.13 shows the decision boundary in a particular 2D feature space. Six targets exist here, grouped such that $\{x_5, x_6\} \in y_1$ and $\{x_1, x_2, x_3, x_4\} \in y_2$. It is observed that $P(y_1) = 2/6$ and $P(y_2) = 4/6$. Interestingly, this problem is actually a fundamental topic of discrete probability - the "urn problem."[19, p.51] In Figure 1.14, there are two urns corresponding to the two decision regions. The first urn contains two balls (targets) and the second holds four, so the probability that any one ball is in urn y_2 is twice the probability for being in urn y_1.

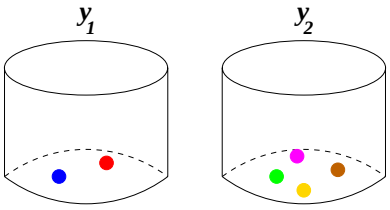

Figure 1.14: The *Urn Problem* is the same as the decision region problem.

The urn problem is a guessing game for two people. At the start of the game, a dealer and a player agree on a rule for placing any ball in the two urns based upon the color of the ball. It can be agreed, for instance, that the y_1 urn can contain only red balls and blue balls, and that the y_2 urn can contain green, violet, yellow or brown balls. The dealer then places a ball, unknown to the player, in an urn and hands the urn to the player and asks,

"what is the color of the ball in this urn?" Since the player knows the rules by which balls are placed in the two urns, if the dealer hands the player the y_1 urn, the player has one chance in two of guessing the right color; if the dealer handed the player the y_2 urn, the player would have one in four odds of guessing the right color ball.

The ball placement rule which was agreed upon by the dealer and the player governs how much *information* is revealed to the player about the ball in an urn. For example, the dealer may proclaim that all balls can be placed in the y_2 urn and no balls can be placed in the y_1 urn. If he places a ball in the y_2 urn and hands it to the player, the player has a one-in-six chance of guessing the color correctly, the same probability of guessing correctly as if there were no urns at all. Thus with this rule, no information is gained with the receipt of either urn.

Alternatively, the dealer might state that the y_1 urn may contain red balls, blue balls, and green balls and that the y_2 urn may contain yellow balls, violet balls and brown balls. The player could guess the color of any ball placed in either urn with probability 1/3. In the following sections, it will be shown quantitatively how this arrangement maximizes the average information gained by the player about the colors of balls in either urn.

1.4.1 Entropy

The entropy of a discrete random variable is a measure of its uncertainty or randomness. Throughout this document, all logarithms will be base two logarithms, and the resulting information measures are expressed in bits. The entropy of X is given as,

$$H(X) = -\sum_{\mathcal{X}} p(x) \log p(x). \tag{1.1}$$

Where $p(x)$ is the probability mass function for the random variable X, $p(x) = P_X(X = x)$, a convenient notation adopted from Cover and Thomas [5].

Entropy is maximized for a certain random variable if it is uniformly distributed. Cover and Thomas[5] state this in **Theorem 2.6.4**[5, pp.27],

> **Theorem 2.6.4**[5, pp.27]: *$H(X) \leq \log |\mathcal{X}|$, where $|\mathcal{X}|$ denotes the number of elements in the range of X, with equality if and only if X has a uniform distribution over $\mathcal{X}$.*

Put another way, $H(X)$ is maximum and equal to $\log N$ for a uniformly distributed discrete random variable with N outcomes. As an example, consider the random variable Y from the urn problem in Figure 1.14 with two states $Y = \{y_1, y_2\}$. We find the entropy:

$$H(Y) = -p(y_1)\log p(y_1) - p(y_2)\log p(y_2). \tag{1.2}$$

This entropy $H(Y)$ is bounded by theorem 2.6.4 [5, pp.27], so that $H(Y) \leq 1$. $H(Y)$ is maximized when Y is uniformly distributed, or when $P(y_1) = 1/2$ and $P(y_2) = 1/2$. Thus it becomes apparent that to do the *best job* of conveying information on average by means of an urn, each urn must contain an equal number of balls. Similarly, to do the best job of separating the set of target exemplars X between the two regions y_1 and y_2 formed by a hyperplane, the hyperplane must be placed so that an equal number of exemplars from X lie to each side.

Optimal Height of Binary Decision Trees

This entropy related result can be used to derive the theoretically optimal height of a binary decision tree. Two observations inspire this derivation. First, the entropy (in bits) of any uniform discrete random variable is the minimum number of bits necessary to uniquely identify any outcome or event of that variable. Consider a random variable A with 4 possible outcomes, the entropy: $H(A) = \log_2 4 = 2$ agrees with the number of bits in the base two representation of 4, $4_{10} = 11_2$.

Second, one bit can represent the outcome of a binary decision. In searching a binary tree, each node visited has a binary decision outcome, representable by one bit. The average height, $\overline{h}$ of a binary decision tree is the average number of decision nodes visited to reach a leaf node. The four hypothesis decision tree shown in Figure 1.15 has an average height of $\overline{h} = 2$.

Therefore, the average height of a binary decision tree is the average number of bits necessary to represent any outcome. In general, the optimal average height is the entropy of the set of N equally likely outcomes $\overline{h} = \log_2 N$, which matches the intuitive result presented in section 1.2.

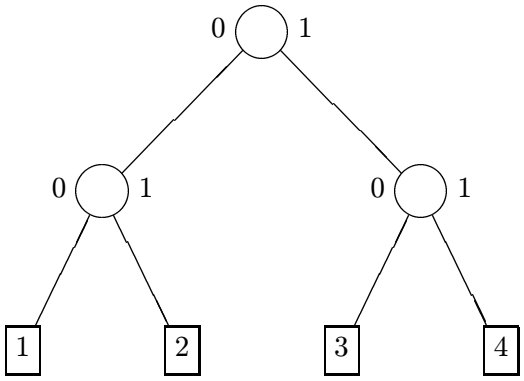

Figure 1.15: Four hypothesis binary decision tree.

Conditional and Joint Entropy

For systems of two or more random variables, both joint and conditional entropy measures may be computed. Given the joint probability mass function of two random variables, $p(x, y)$, the joint entropy can be formulated,

$$H(X, Y) = -\sum_{\mathcal{X}} \sum_{\mathcal{Y}} p(x, y) \log p(x, y). \tag{1.3}$$

As can the conditional entropy of Y given X,

$$H(Y|X) = -\sum_{\mathcal{X}} \sum_{\mathcal{Y}} p(x, y) \log p(y|x). \tag{1.4}$$

A useful tool when working with joint and conditional entropies is the chain rule for entropy:

$$H(Y|X) = H(X, Y) - H(X). \tag{1.5}$$

Here the conditional entropy of Y, given X is the joint entropy minus the entropy of the given variable. (A proof of this chain rule is given in Cover and Thomas as **theorem 2.2.1**[5, p.16].) When one places conditions on a random variable, one is applying a priori knowledge of that given variable. This chain rule states that conditional entropy is simply the total joint entropy with the uncertainty of the given variable removed. Given a priori knowledge

of a random variable, we may remove its uncertainty from the combined uncertainty of both variables.

1.4.2 Mutual Information

From an optimistic standpoint, one may wish to examine the information content, rather than the uncertainty in a two variable system. Building on the concept of entropy, mutual information is the reduction of uncertainty (entropy) of a random variable when knowledge of another is available, denoted $I(X;Y)$. It can be stated in terms of entropy, conditional entropy and joint entropy,

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) && (1.6)\\ &= H(Y) - H(Y|X) && (1.7)\\ &= H(X) + H(Y) - H(X,Y). && (1.8) \end{aligned}$$

Mutual information is conversely envisaged as the information gained about one variable by knowledge of another. It answers the question: by knowing something about X, how much information is revealed about Y? Intuitively, the quantity expressed by mutual information is symmetric, revealed by equations 1.6 and 1.7, such that $I(X;Y) = I(Y;X)$. The information uncovered about Y by knowing X is the same amount of information gained about X by knowing Y. Mutual information brings to light an interconnectedness of the random variables which it measures.

Mutual Information in the Urn Problem

In the urn problem, one can view $I(X;Y)$ as the information gained about the set of balls $X = \{x_1, x_2, \ldots, x_6\}$ by the way they are divided between the two urns $Y = \{y_1, y_2\}$. Explicitly, this mutual information is best represented in this case by:

$$I(X;Y) = H(Y) - H(Y|X), \tag{1.9}$$

forming the mutual information as the reduction in the uncertainty about Y by the uncertainty of Y when X is known. The conditional entropy,

$$H(Y|X) = -\sum_{\mathcal{X}} \sum_{\mathcal{Y}} p(x,y) \log p(y|x), \tag{1.10}$$

is formed from the joint probability mass function and the conditional probability of y given x. Logically, $p(y|x)$ reduces to either one or zero. A ball is always completely inside one urn or the other - it can't have been somehow placed partially in either urn. This indicates that the entropy of Y given X will *always* be zero - there is no uncertainty about which urn each of the balls came from, and the information function reduces to

$$I(X;Y) = H(Y). \tag{1.11}$$

Therefore, to maximize the information gained by the player of our game about the color of a ball from the urn in which it is placed, one must choose the rule that maximizes the uncertainty the dealer has about into which urn the still unchosen ball needs to be placed.

If, as in Figure 1.16, we place all the balls in the y_2 urn,

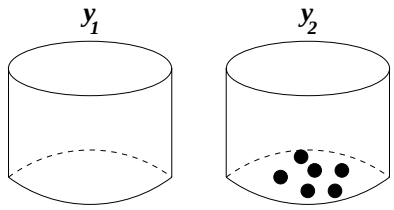

Figure 1.16: No information is gained about a ball if the rule is that all balls will be placed in one urn.

$$\begin{aligned} I(X;Y) &= H(Y) && (1.12)\\ &= -\sum_{\mathcal{Y}} p(y) \log p(y) && (1.13)\\ &= 0 \log 0 + 1 \log 1 && (1.14)\\ &= 0 \; bits. && (1.15) \end{aligned}$$

There is no uncertainty about which urn the balls are in, and hence, no information gained by dividing them in this manner. Conversely, we achieve one bit of information when each urn contains three balls:

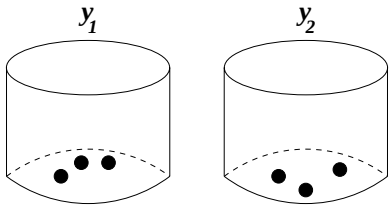

Figure 1.17: Information is maximized with complete separation.

$$
\begin{aligned}
I(X;Y) &= H(Y) && (1.16)\\
&= -\sum_{\mathcal{Y}} p(y)\log p(y) && (1.17)\\
&= -1/2\log 1/2 - 1/2\log 1/2 && (1.18)\\
&= 1\ bit. && (1.19)
\end{aligned}
$$

As a final example, we place five balls in the y_2 urn and one in the y_1:

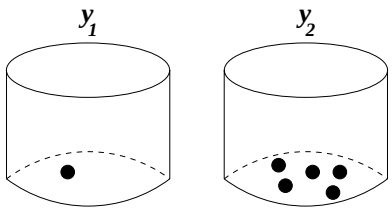

Figure 1.18: Some information can still be had with only partial separation.

$$
\begin{aligned}
I(X;Y) &= H(Y) && (1.20)\\
&= -\sum_{\mathcal{Y}} p(y)\log p(y) && (1.21)\\
&= 5/6\log 6/5 + 1/6\log 6 && (1.22)\\
&= 0.65\ bits. && (1.23)
\end{aligned}
$$

Indicating that even in a partial separation of the balls, some mutual information exists.

1.5 Original Contributions and Acknowledgment of Previous Work

Work on the ITDT system was initiated in 1996, under the direction of Professor Cyganski. Sergey Perepelitsa developed much of the original theoretical background and produced the first working prototype of the system, documented in a report to the Army Research Office[23]. In 1997, Ryan Tomasetti made a number of optimizations to the existing system which allowed it to successfully operate on larger data sets, documented in an unpublished report[29]. This document is the first large-scale work describing efforts past and present.

The author's primary original contributions are as follows:

1. Developed the concept of exemplar grouping to incorporate translation and rotation independence into the ITDT system.

2. Implemented a training system by which rotation and translation independence can be achieved with limited target imagery.

3. Enhanced the local optimization procedure for hyperplane placement, thereby reducing the size of the decision trees constructed. As a result, test speed is increased and the system is less susceptible incorrect decisions due to image corruption.

4. Optimized the tree construction process, yielding construction times nearly an order of magnitude faster, and enabling the system to operate on larger data sets.

In general, major contributions are documented starting in chapter 3. All examples used throughout the course of this report in addition to the general background in this introductory chapter are the author's. The derivation of mutual information for data segmentation given in section 2.2.1 is based on the derivation in Tomasetti[29], but has been reworked and expanded for clarity. Tomasetti's derivation of the information function derivatives is presented as a strict reproduction, in appendix C.

Chapter 2

Information Theoretic Decision Tree Theoretical Foundation

The Information Theoretic Decision Tree system affords a great reduction in comparison operations over a linear search by utilizing a binary decision tree structure to impose order on a set of feature vectors. This set of training feature vectors typically forms an appearance-based model of a particular object or classes of objects. The tree is constructed in a top-down fashion from these high-dimensional vectors by successively segmenting them, so that each leaf of the tree contains a single model class. Section 1.4 presented Leo Brieman's three essential elements for building decision trees. These elements can now be stated explicitly for the ITDT system:

1. **The selection of the splits** - A hyperplane is placed in the decision space at each node so as to maximize the information about the set of feature vectors gained by dividing the features into two subsets. The ITDT system assumes feature vectors are corrupted with additive i.i.d. Gaussian noise.

2. **The decision when to declare a node terminal or to continue splitting it** - Nodes are declared terminal when there is only one target class present in the node.

3. **The assignment of each terminal node to a class**[1] - Each terminal node, containing a feature vector corresponding to a single target class is simply assigned to that class.

[1]Breiman's list is found in [4], *Classification and Regression Trees*, pp.22.

Fundamentally, the ITDT system achieves its high performance goal through a single extension to the prior discussion of mutual information by hyperplane splits. By making a traditional communication theory assumption that the input data are corrupted by additive white Gaussian noise (AWGN), the ITDT system forms a model which is trained to reject subtle variations and noise in test imagery. The process of searching the resultant tree to classify a test vector is rapid; section 2.1 presents a top-down method. Section 2.2 presents the details of decision tree construction, including the maximization of mutual information at each successive node. Although the assumption of AWGN corrupted feature vectors increases the likelihood of successful object classification in non-ideal environments, it does complicate the derivation of mutual information for node splits. A full treatment, expanding the theory developed for the urn problem is presented in section 2.2.1.

2.1 Testing a Decision Tree with a Test Feature Vector

The process of locating a test feature vector in a decision region formed by successive hyperplane placements in a binary decision tree is independent of the means by which the hyperplane decision space splits were chosen. The hyperplane decision boundaries in the ITDT system were placed to maximize the information gained by splitting a set of training feature vectors. Once the hyperplanes have been placed, the test-time functionality of the system knows not of the means by which they were placed, only knowing their locations, orientations and the structure of the decision tree. Three primary structures are utilized in a search of a binary decision tree of this type:

1. a binary tree with a number of terminal nodes and a number of decision nodes,
2. a hyperplane decision boundary at each decision node of the tree, and
3. a feature vector to be classified by the system.

At test time, an acquired test feature vector is passed to the ITDT system, which is to search a pre-constructed decision tree and return a reference to the target class which best matches the test vector. Due to the order imposed on the feature vectors by the tree structure, the best match can be easily arrived upon with a top-down search of the decision tree, terminating in a leaf node which will contain a reference to a particular training

feature vector which is closest to the feature vector under test. The search process is given algorithmically in the following list:

1. Begin at the top of the tree.
2. If this node is a terminal leaf node, the search is ended. The test feature vector lies within the decision region of a single original training feature vector.
3. If the node is a decision node, compute the minimum signed distance d from the point identified by the test feature vector x to any point on the hyperplane z.
   - If $d > 0$, proceed down the left branch
   - otherwise, proceed down the right branch
4. Loop back to step 2.

The minimum distance from a point x to a hyperplane is the signed magnitude of a vector oriented normal to the hyperplane, originating on the hyperplane and terminating at x. Figure 2.1 shows the decision boundary hyperplane, feature vector and distance vector for a hypothetical two-dimensional problem. The sign of the distance identifies the side of the hyperplane to which the test feature vector falls.

The surface of the hyperplane, z, in $N-$space, is represented by a linear equation,

$$z_1 x_1 + z_2 x_2 + \cdots + z_N x_N = z_{N+1}. \tag{2.1}$$

The first N coefficients of the hyperplane's equation, $z_1, z_2, \ldots, z_N$, are the components of a vector oriented normal to the hyperplane. The final coefficient, z_{N+1}, controls the offset of the hyperplane from the origin; if $z_{N+1} = 0$, the hyperplane passes through the origin. A point lies on the surface of the hyperplane if equality holds in equation 2.1 when the point's coordinates are substituted for the parameters $x_1, x_2, \ldots, x_N$.

The minimum distance, d, is computed by substituting the components of the feature vector x for $x_1, x_2, \ldots, x_N$ in the left side of equation 2.1, subtracting the z_{N+1} component, and normalizing to the magnitude of the hyperplane:

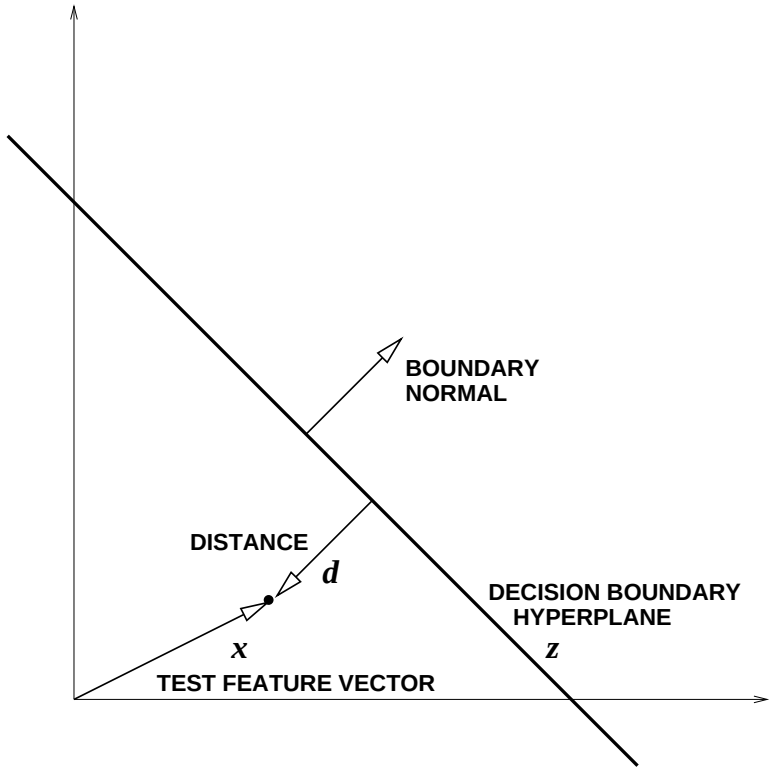

Figure 2.1: Calculating the minimum distance of a feature vector to a hyperplane.

$$d(x) = \frac{< \mathbf{x}, \mathbf{z} > -z_{N+1}}{||\mathbf{z}||}. \tag{2.2}$$

At the completion of a search, a reference is returned to a training feature vector which most resembles the feature vector under test. This reference is simply an index, and as such, does not quantify the extent to which the feature vector under test and the matching training feature vector resemble each other. A *goodness-of-fit* match metric can be computed as a distance measure from the test feature vector to the matching training feature vector, if the system can store the original training vectors.

2.2 Constructing an Information Theoretic Decision Tree

The information theoretic decision tree is constructed by repeatedly splitting the set of input training feature vectors into two sets, forming a binary tree in which each downward path defines a region in space occupied by a single training feature vector. At each new

node of the tree, a hyperplane is placed to form a decision boundary in feature space, splitting the set of feature vectors in that node into two distinct groups, one on the left of the hyperplane and one on the right. Each group is then used to create a corresponding left child node or right child node, which undergoes the same splitting process. Figure 2.2 depicts the structure of a binary decision tree constructed by successively segmenting a set of feature vectors.

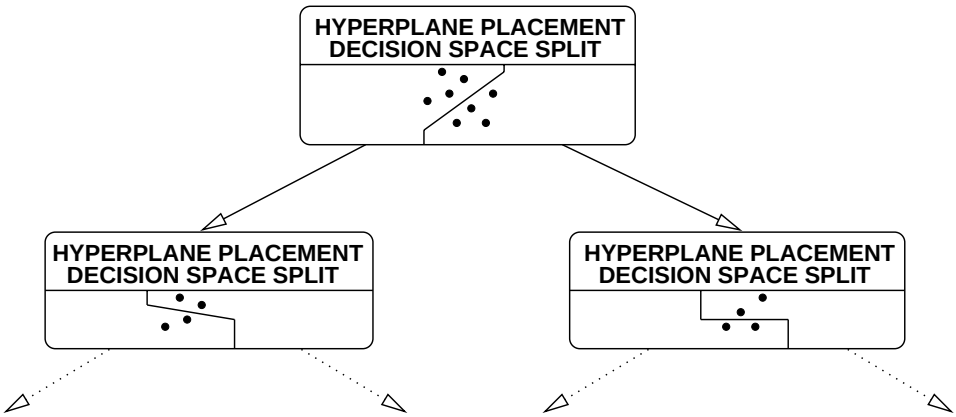

Figure 2.2: Construction of the binary decision tree.

The ITDT system maximizes mutual information for each node split in the same way as in the urn problem, save for one important difference. In the urn problem, the probability mass function describing the location of a particular ball was an impulse function - each ball was a single point. To account for limited variation in information represented as feature vectors, the ITDT system models each target exemplar as a feature vector corrupted by additive N-dimensional independent and identically distributed (i.i.d.) Gaussian noise. Under this system, with some non-zero probability, each exemplar can now occupy any point in an N-volume. When the space is cut with a hyperplane, it is possible that it may slice through one of the densities, indicating that an exemplar could lie to *both* sides of a hyperplane. In Figure 2.3, two i.i.d. Gaussian random vectors, x_1 and x_2 are represented with circles drawn at a radius of one standard deviation. The tails of both densities extend far off in any direction, creating a situation in which instances of the classes represented by the random vectors x_1 or x_2 can lie on either side of the hyperplane, with a non-zero probability.

In this example, the probability that an instance of the class represented by x_1 lies to the left of the hyperplane is greater than the probability that it lies to the right. The probability that an instance of the class represented by x_1 lies to the left of the hyperplane can be computed by evaluating the Gaussian cumulative distribution function, $Q(d/\sigma)$, from the length of the line segment shown in Figure 2.3 which is oriented perpendicular to the hyperplane, and connects it to the mean of density x_1.

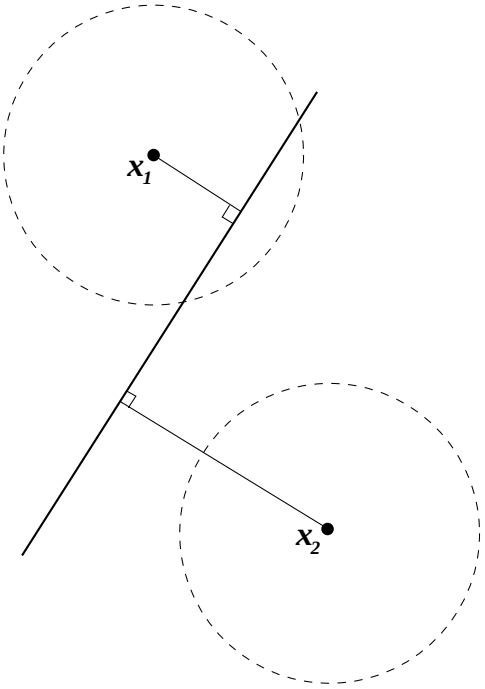

Figure 2.3: Decision boundary and two AWGN-corrupted feature vectors.

By definition, a symmetric Gaussian density has a hypersurface of equal probability at a given distance from the mean. When the space is split by a hyperplane, one can evaluate the conditional probability that a particular exemplar lies to one side or the other of the hyperplane if one knows the perpendicular distance of the exemplar's feature vector to the hyperplane, d_i. From these conditional probabilities, the total probabilities and entropies necessary to compute mutual information are derived.

2.2.1 Derivation of Mutual Information for Node Splits

The following information is given:

- $X = \{x_1, x_2, \ldots, x_N\}$ The set of N equally likely target exemplars which are taken to be the mean of a Gaussian vector with independent and identically distributed (i.i.d.) components.
- $Y = \{y_1, y_2\}$ Sets y_1 and y_2 formed by the placement of a hyperplane in feature space. The set of target exemplars, X, is distributed between these two sets.
- A hyperplane $z_1x_1 + z_2x_2 + \cdots + z_Nx_N = z_{N+1}$ which defines the two regions y_1 and y_2.
- σ: The standard deviation of all the i.i.d. Gaussian distributions associated with each target vector.

The mutual information is given as,

$$I(X;Y) = H(X) + H(Y) - H(X,Y). \tag{2.3}$$

Where,

$$\begin{aligned} H(X,Y) &= -\sum_{\mathcal{X}}\sum_{\mathcal{Y}} p(x,y)\log p(x,y), & (2.4)\\ H(X) &= -\sum_{\mathcal{X}} p(x)\log p(x), & (2.5)\\ H(Y) &= -\sum_{\mathcal{Y}} p(y)\log p(y). & (2.6) \end{aligned}$$

The entropy of the set of N equally likely target exemplars evaluates easily as:

$$H(X) = -\sum_{\mathcal{X}} p(x)\log p(x) = \log N \tag{2.7}$$

The entropy of Y is dependent on the total probability of any target exemplar lying in either the y_1 or y_2 regions.

$$H(Y) = -P(y_1)\log P(y_1) - P(y_2)\log P(y_2) \tag{2.8}$$

The total probability that any exemplar lies in region y_1 is,

$$P(y_1) = \sum_{i=1}^{N} P(y_1|x_i)P(x_i) = \frac{1}{N}\sum_{i=1}^{N} P(y_1|x_i). \tag{2.9}$$

Which is, in turn, dependent on the probability that a *particular* target exemplar x_i would lie in the y_1 region, determined by a N-dimensional additive Gaussian noise cumulative distribution function, given as the integral of the Gaussian probability density function $n(X, \sigma)$,

$$P(y_1|x_i) = \int_{\Omega} n(X, \sigma) d\Phi. \tag{2.10}$$

We establish an orthonormal coordinate system $\Phi = \{\phi_1, \phi_2, \ldots, \phi_N\}$ so that ϕ_i is parallel to the perpendicular distance d_i between the feature vector x_i and the hyperplane Z; its origin is centered at the point of intersection of the perpendicular with the hyperplane. In doing so, all components except the perpendicular one can be ignored and equation 2.10 reduces to,

$$P(y_1|x_i) = \int_{-d_i}^{\infty} n(X, \sigma) d\phi_i. \tag{2.11}$$

With the perpendicular distance from feature vector $\mathbf{x_i}$ to hyperplane $\mathbf{z}$ given as,

$$d_i = \frac{< \mathbf{x_i}, \mathbf{z} > -z_{N+1}}{||\mathbf{z}||}. \tag{2.12}$$

Where $||\mathbf{z}||$ is the vector norm of all components of the hyperplane $\mathbf{z}$,

$$\sqrt{z_1^2 + z_2^2 + \cdots + z_{N+1}^2}, \tag{2.13}$$

and $< \mathbf{x_i}, \mathbf{z} >$ is the inner product of the vectors $\mathbf{x_i}$ and $\mathbf{z}$. From this distance measure, the subset in Y in which a particular feature vector lies is defined by:

- $x_i \in y_1$ if $d_i \leq 0$
- $x_i \in y_2$ if $d_i > 0$

The cumulative distribution function for Gaussian densities can be expressed in terms of the Q function,

$$N(\eta, \sigma) = Q(\frac{x - \eta}{\sigma}) \tag{2.14}$$

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\lambda^2/2} d\lambda \tag{2.15}$$

Equation 2.11 can thus be rewritten,

$$\begin{aligned} P(y_1|x_i) &= \int_{-d_i}^{\infty} n(X, \sigma) d\phi_i & (2.16) \\ &= 1 - \int_{-\infty}^{d_i} n(X, \sigma) d\phi_i & (2.17) \\ &= 1 - Q(\frac{d_i}{\sigma}). & (2.18) \end{aligned}$$

The sets y_1 and y_2 are independent and represent a complete event space, therefore the conditional probability of region y_2 containing a given x_i is

$$P(y_2|x_i) = 1 - P(y_1|x_i) = Q(\frac{d_i}{\sigma}). \tag{2.19}$$

Returning to equation 2.9, the total probability that any exemplar is in y_1 may be simplified to:

$$P(y_1) = \frac{1}{N} \sum_{i=1}^{N} P(y_1|x_i) = \frac{1}{N} \sum_{i=1}^{N} [1 - Q(\frac{d_i}{\sigma})] = 1 - \frac{1}{N} \sum_{i=1}^{N} Q(\frac{d_i}{\sigma}). \tag{2.20}$$

And, the total probability of any exemplar in y_2 may be similarly simplified:

$$P(y_2) = \frac{1}{N} \sum_{i=1}^{N} P(y_2|x_i) = \frac{1}{N} \sum_{i=1}^{N} Q(\frac{d_i}{\sigma}). \tag{2.21}$$

The average entropy of the two regions y_1 and y_2 (from equation 2.6) follows as,

$$
\begin{aligned}
H(Y) &= -P(y_1)\log P(y_1) - P(y_2)\log P(y_2) && (2.22)\\
&= -\frac{1}{N}\left[\sum_{i=1}^{N}[1-Q(\frac{d_i}{\sigma})]\log(\ \frac{1}{N}\sum_{i=1}^{N}[1-Q(\frac{d_i}{\sigma})]\ )\right] + \\
&\quad -\frac{1}{N}\left[\sum_{i=1}^{N}Q(\frac{d_i}{\sigma})\log(\frac{1}{N}\sum_{i=1}^{N}Q(\frac{d_i}{\sigma}))\right] && (2.23)\\
&= \frac{\log N}{N}\sum_{i=1}^{N}\left(1-Q\left(\frac{d_i}{\sigma}\right)+Q\left(\frac{d_i}{\sigma}\right)\right) + \\
&\quad -\frac{1}{N}\sum_{i=1}^{N}\left(1-Q(\frac{d_i}{\sigma})\right)\log\left[\sum_{i=1}^{N}\left(1-Q(\frac{d_i}{\sigma})\right)\right] + \\
&\quad -\frac{1}{N}\sum_{i=1}^{N}Q(\frac{d_i}{\sigma})\log\left[\sum_{i=1}^{N}Q(\frac{d_i}{\sigma})\right] \\
&= \log N - \frac{1}{N}\sum_{i=1}^{N}\left(1-Q(\frac{d_i}{\sigma})\right)\log\left[\sum_{i=1}^{N}\left(1-Q(\frac{d_i}{\sigma})\right)\right] + \\
&\quad -\frac{1}{N}\sum_{i=1}^{N}Q(\frac{d_i}{\sigma})\log\left[\sum_{i=1}^{N}Q(\frac{d_i}{\sigma})\right]. && (2.24)
\end{aligned}
$$

And finally, the joint entropy is the average entropy of all of the combinations of X and Y, as given by:

$$
\begin{aligned}
H(X,Y) &= -\sum_{\mathcal{X}}\sum_{\mathcal{Y}}p(x,y)\log p(x,y) && (2.25)\\
&= -\sum_{\mathcal{X}}\sum_{\mathcal{Y}}p(y|x)p(x)\log(\ p(y|x)p(x)\ ) && (2.26)\\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[P(y_1|x_i)\log\frac{P(y_1|x_i)}{N}+P(y_2|x_i)\log\frac{P(y_2|x_i)}{N}\right] && (2.27)\\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[\left(1-Q(\frac{d_i}{\sigma})\right)\log\left(\frac{1-Q(\frac{d_i}{\sigma})}{N}\right)+Q(\frac{d_i}{\sigma})\log\left(\frac{Q(\frac{d_i}{\sigma})}{N}\right)\ \right] && (2.28)\\
&= -\frac{1}{N}\sum_{i=1}^{N}\left[\left(1-Q(\frac{d_i}{\sigma})\right)\log\frac{1}{N}+Q(\frac{d_i}{\sigma})\log\frac{1}{N}\right] + \\
&\quad -\frac{1}{N}\sum_{i=1}^{N}\left[\left(1-Q(\frac{d_i}{\sigma})\right)\log\left(1-Q(\frac{d_i}{\sigma})\right)+Q(\frac{d_i}{\sigma})\log Q(\frac{d_i}{\sigma})\right], && (2.29)
\end{aligned}
$$

which simplifies as given below,

$$H(X,Y) = \log N - \frac{1}{N}\sum_{i=1}^{N}\left[\left(1 - Q(\frac{d_i}{\sigma})\right)\log\left(1 - Q(\frac{d_i}{\sigma})\right) + Q(\frac{d_i}{\sigma})\log Q(\frac{d_i}{\sigma})\right]. \quad (2.30)$$

Combining equations 2.7, 2.24 and 2.30, an explicit expression for the mutual information of X and Y is arrived upon:

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X,Y) && (2.31)\\ &= \log N + \log N - \frac{1}{N}\sum_{i=1}^{N}\left(1 - Q(\frac{d_i}{\sigma})\right)\log\left[\sum_{i=1}^{N}\left(1 - Q(\frac{d_i}{\sigma})\right)\right] + \\ &\quad -\frac{1}{N}\sum_{i=1}^{N} Q(\frac{d_i}{\sigma})\log\left[\sum_{i=1}^{N} Q(\frac{d_i}{\sigma})\right] + \\ &\quad -\log N + \frac{1}{N}\sum_{i=1}^{N}\left[\left(1 - Q(\frac{d_i}{\sigma})\right)\log\left(1 - Q(\frac{d_i}{\sigma})\right) + Q(\frac{d_i}{\sigma})\log Q(\frac{d_i}{\sigma})\right] && (2.32)\\ &= \log N - \frac{1}{N}\sum_{i=1}^{N}\left(1 - Q(\frac{d_i}{\sigma})\right)\log\left[\sum_{i=1}^{N}\left(1 - Q(\frac{d_i}{\sigma})\right)\right] + \\ &\quad -\frac{1}{N}\sum_{i=1}^{N} Q(\frac{d_i}{\sigma})\log\left[\sum_{i=1}^{N} Q(\frac{d_i}{\sigma})\right] + \\ &\quad +\frac{1}{N}\sum_{i=1}^{N}\left[\left(1 - Q(\frac{d_i}{\sigma})\right)\log\left(1 - Q(\frac{d_i}{\sigma})\right) + Q(\frac{d_i}{\sigma})\log Q(\frac{d_i}{\sigma})\right]. && (2.33) \end{aligned}$$

2.2.2 Maximizing Mutual Information

Mutual information maximization in the decision tree is implemented as a local optimization procedure for the decision space at each node of the tree. Given a set of N target exemplars, the placement of a hyperplane splitting the node's decision space must be optimized dynamically to yield maximum mutual information by the measure derived above (2.33.) Central to the optimization procedure is the conjugate gradient minimization method, as found in Press et. al[18]. Overall, this optimization is a three stage process (Figure 2.4):

1. Form an initial placement guess and set an initial Gaussian standard deviation σ_0 which is large compared to the distance between the exemplars.

2. Find a local minimum for $-I(X;Y)$ (thereby finding a maximum for $I(X;Y)$) with the conjugate gradient minimization procedure

3. Choose a new standard deviation $\sigma_{k+1} < \sigma_k$ on the k^{th} execution of this step and and return to the local minimization (step 2), leaving the loop when the local minimum is unchanged over two iterations.

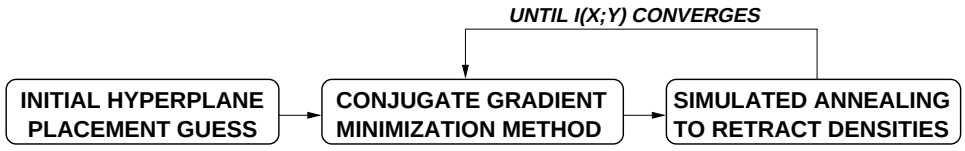

Figure 2.4: Three stage optimization procedure for node splits.

Initial Hyperplane Placement Guess

For the conjugate gradient minimization algorithm, it is important to place the hyperplane so that it cuts through the cloud of feature vectors. The closer it is placed to a local minimum, the quicker the conjugate gradient algorithm will converge.

After evaluation of many different methods of initial placement, the following method was arrived upon: place the hyperplane through the "center of gravity" of all feature vectors, oriented normal to the vector from the center of gravity to the feature vector which lies furthest away from the center of gravity (Figure 2.5). Placement in this manner ensures that the hyperplane cuts through the center of the target feature vectors and is oriented so as to separate at least one exemplar (the most different) from the majority. First the center of gravity or mean feature vector is calculated,

$$\overline{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x_i}. \tag{2.34}$$

Then a Euclidean distance measure D is applied to measure each feature vector's distance from the center of gravity,

$$D_i = \sqrt{< (\overline{\mathbf{x}} - \mathbf{x_i}), (\overline{\mathbf{x}} - \mathbf{x_i}) >} \tag{2.35}$$

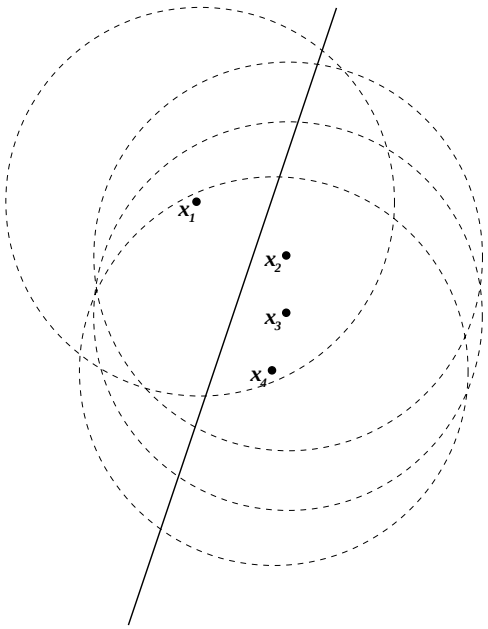

Figure 2.5: Initial guess for hyperplane placement.

retaining $\mathbf{x}_\Delta$, the feature vector furthest away from the center of gravity, identified by its argument,

$$\Delta = \arg\max_i D_i. \tag{2.36}$$

The coefficients of the hyperplane represented as $z_1x_1 + z_2x_2 + \cdots + z_Mx_M = z_{M+1}$ are the components of the vector normal to the surface of the hyperplane. By setting these components to those of the difference vector between the most distant feature vector and the center of gravity, the hyperplane is placed orthogonal to the difference vector's orientation.

$$\mathbf{z} = \mathbf{x}_\Delta - \overline{\mathbf{x}}. \tag{2.37}$$

The final coefficient is set so that the plane passes through the center of gravity,

$$z_{M+1} = < \overline{\mathbf{x}}, \mathbf{z} > . \tag{2.38}$$

Conjugate Gradient Optimization Method

The conjugate gradient local optimization algorithm used in the ITDT system is the Polak-Ribiere variant from Press et. al, *Numerical Recipes in C*[18, pp.317-324]. This particular method was chosen because it offers excellent performance and minimizes the number of evaluations of the mutual information function (equation 2.33). The fundamental trade-off with this algorithm, as with any gradient-based algorithm, is that the partial derivatives of the mutual information function must be evaluated. Appendix C contains the analytic derivation of these partial derivatives.

Figure 2.6 illustrates the local optimization behavior of the conjugate gradient algorithm. The algorithm iteratively adjusts the orientation and offset of the hyperplane, using the gradient of the information function to guide it to a local maximum of mutual information. The term *conjugate*, is applied to the description because the algorithm locates a local optimum by progressively minimizing over directions which are different, or conjugate, to the previous attempts.

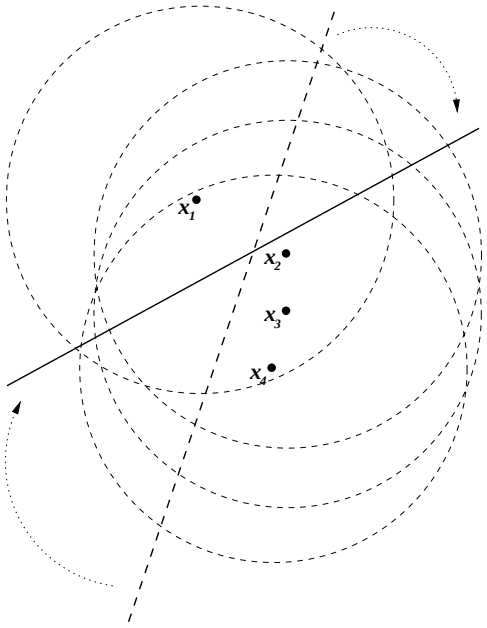

Figure 2.6: The conjugate gradient method finds the local minima of $-I(X;Y)$.

For a detailed treatment, the reader is urged to reference *Numerical Recipes.*[18]

Simulated Annealing

The optimization technique of *simulated annealing* gained widespread attention by often finding optimum or nearly optimum solutions of the *traveling salesman problem* – the problem of producing a minimum-length route on which a salesman travels to visit N cities and eventually return to his start point. As an optimization algorithm, it is a relatively slow one, but the extrema it finds tend to be nearly as good as the global optimum. Simulated annealing takes its name from the slow cooling process by which a blacksmith can create swords in the form of a single crystal of metal:

an·neal 1. To subject (glass or metal) to a process of heating and slow cooling to reduce brittleness.[1]

For the ITDT system, a simple simulated annealing algorithm acts as a control loop for the conjugate gradient optimization routine. Its philosophy of operation is that by beginning the optimization procedure with wide noise densities and slowly retracting them, optimizing for a local minimum along the way, the final minimum converged upon will be nearly as good as the global optimum. Three snapshots of this process are shown in Figure 2.7, demonstrating the retraction of the Gaussian densities. By beginning the optimization process with wide noise densities, the conjugate gradient is prevented from finding a low-quality local maximum in which it may only segment off a few of the feature vectors from a large group. Using wide noise densities provides a smooth surface for the algorithm to optimize on, yielding an initial local optimum which reflects much of the global configuration of the feature vectors.

At the start of the process of simulated annealing, the i.i.d. Gaussian noise densities corrupting the feature vectors are set to a standard deviation wider than the length of the maximum possible distance between any two feature vectors. In an earlier preprocessing stage, the maximum vector norm of the set of training feature vectors was computed and used to normalize the entire set of training feature vectors. As a result, the maximum vector norm of the set of feature vectors is 1.0. Choosing a standard deviation of 1.0 to begin the simulated annealing process ensures sufficiently wide noise densities to prevent immediate convergence to low-quality information maxima.

The simulated annealing process as used in the ITDT system is straightforward:

[1] from *The American Heritage College Dictionary*, 3rd ed., Houghton Mifflin Company, New York.

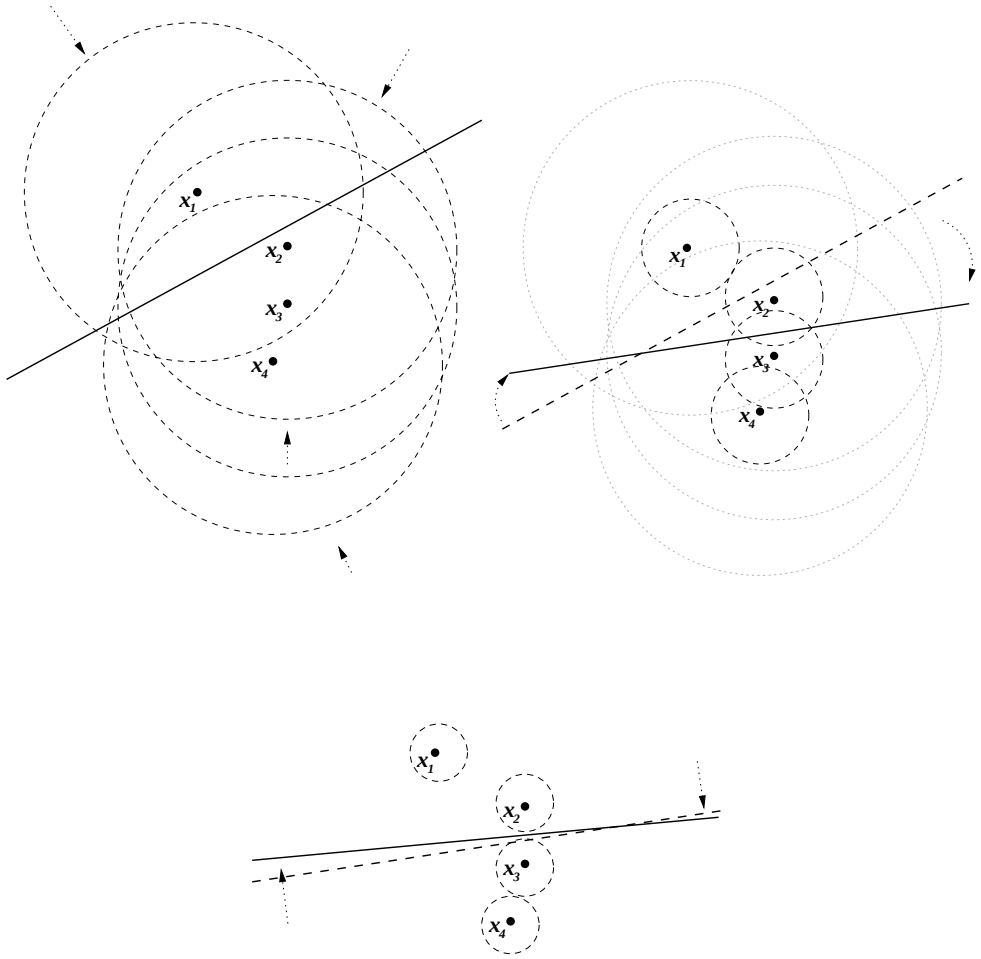

Figure 2.7: Simulated annealing is used to retract the noise densities.

1. Begin with noise densities at a standard deviation of $\sigma_1 = 1.0$.
2. Find a local minima with the conjugate gradient routine.
3. Halve the standard deviation of all the noise densities, $\sigma_{k+1} = \sigma_k/2$.
4. While $I(X;Y)$ is increasing, loop back to 2. Exit when $I(X;Y)$ remains unchanged from the previous loop.

The hyperplane placement reached by this algorithm is the final placement used in a particular node of the decision tree.

2.2.3 Considerations Regarding the Constructed Decision Tree

When constructing a decision tree, it is very likely that the mutual information maximization at some node splits will be much less than 1 bit. This non-optimality of a node split indicates that the feature vectors were difficult to segment, and results in a taller tree. Although a decision tree constructed to classify a set of target feature vectors must, by the very way in which it is constructed, correctly classify the imagery used to train it, non-optimal node splits increase the ITDT system's sensitivity to noise in test imagery. This will be demonstrated in the results for T72 pose classification in the next section.

Figure 2.8 shows an optimal size binary tree constructed to classify four targets. If, for example, vectors x_2 and x_3 were located near each other (Figure 2.9), a decision tree similar to the one in Figure 2.10 could result.

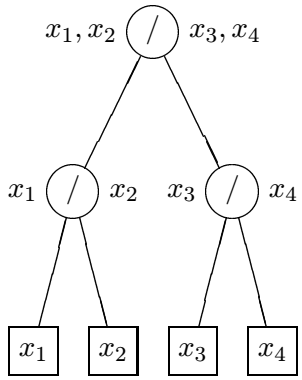

Figure 2.8: Progressive exemplar splitting in the decision tree.

In the tree in Figure 2.10, three decisions have to be made to classify vectors x_2 and x_3 correctly. Examining the arrangement of the decision boundaries in Figure 2.11, it is shown that since x_2 and x_3 were so similar, deciding between them was put off until there were no other feature vectors in the decision space.

However, a non-optimal height tree does indicate increased sensitivity to subtle variations in the test imagery from the imagery used to train the system. In this example, if a

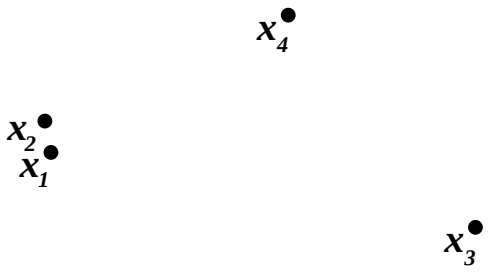

Figure 2.9: Difficult arrangement of feature vectors.

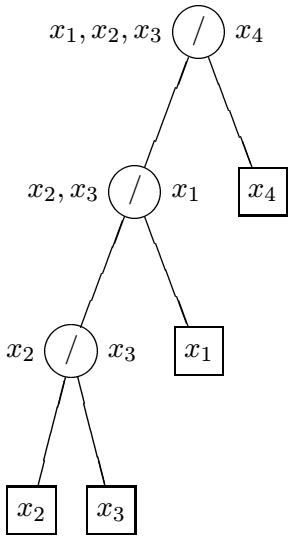

Figure 2.10: Greater than optimal-size tree doesn't increase classification errors.

test feature vector known to fall within the x_2 class were to differ only slightly from the training vector for x_2, it could easily be misclassified as an x_1 class feature vector. In practice, if feature vectors such as x_1 and x_2 represented digital images in an appearance-based model, they would appear as two nearly identical images to the human eye. Any machine vision classifier would be hard-pressed to discriminate between them.

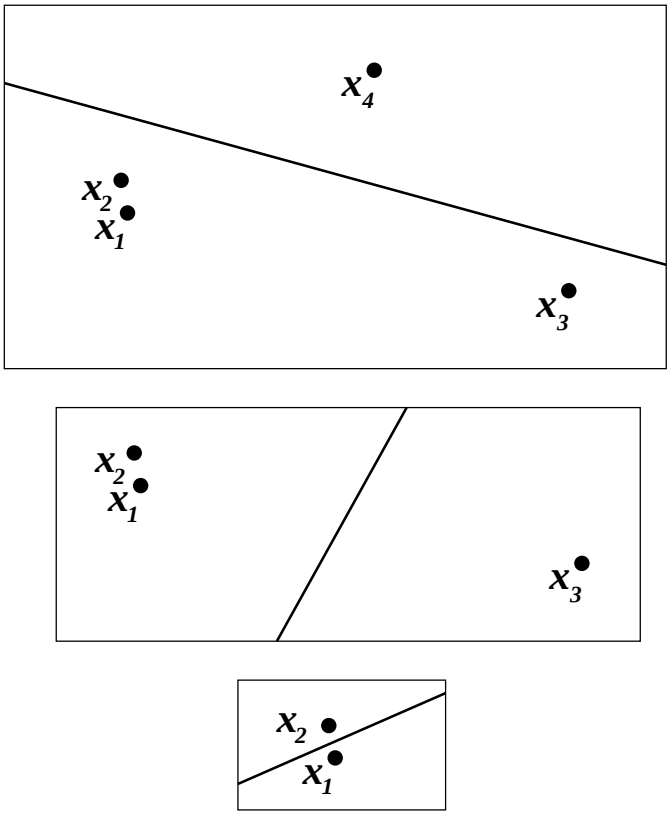

Figure 2.11: Decision spaces.

2.3 T72 SAR Data Pose Estimation

One of the earliest tests of the ITDT system involved construction of a pose estimation decision tree from synthetic aperture radar (SAR) imagery of a Soviet T72 tank over about 120° of azimuth angle. Figure 1.8 on page 11 shows twenty-five of the images used to construct the tree, and the tree generated for this particular test was reproduced earlier in Figure 1.9 on page 13. Table 2.1 contains the results of this tree construction. Feature vectors were formed directly from the SAR imagery of the tank, as described in section 1.3.1, and depicted in Figure 1.12 on page 17.

Time to build tree	1m 28s
Maximum decisions ($\max h$)	6
Average height ($\overline{h}$)	5.8
Optimal height	5.7
Dimensions	1296 (36x36)
Training feature vectors	53

Table 2.1: T72 Pose Estimation Tree

This test and all subsequent tests were conducted using an implementation of the ITDT system written in the C++ language. The system runs under the Linux operating system on an Intel Pentium II based PC operating at 350MHz.

This particular tree was quickly constructed, in one minute and twenty eight seconds. Referring to the tree drawn on page 13, the measures of height in the tree refer to the number of circular decision nodes visited to reach any leaf in the tree. For this tree, the maximum number of decisions was 6, and the average number of decisions was 5.8. An optimal size tree for 53 exemplars would have had, on average, 5.7 decisions to reach a leaf node.

Some insight can be gained into this optimality by returning the hyperplanes in the decision tree to an image space representation, and viewing the resulting hyperplane images aligned in a binary tree-like image representation. This conversion from hyperplane coefficients to image space is possible because the hyperplane coefficients form a normal vector to the hyperplane - essentially the difference vector between the feature vectors on the left and the feature vectors on the right. Figure 2.12 shows how the hyperplane is, in effect, the perpendicular bisector of a difference vector.

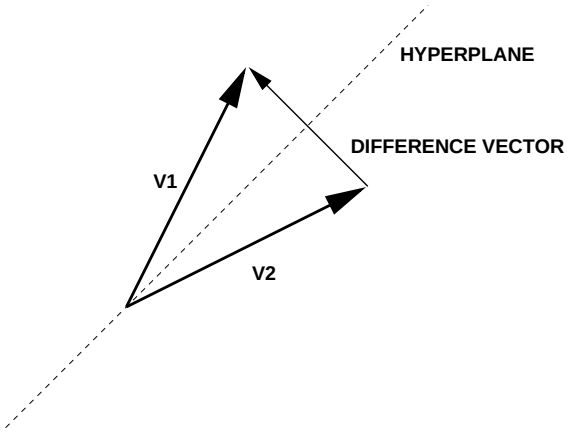

Figure 2.12: An optimal hyperplane separating two exemplars is perpendicular to their difference vector.

This difference vector is a representation of the major differences between the left and right decision regions formed by the hyperplane. It can reveal image features which were leveraged in the final hyperplane placement which maximized mutual information in a particular node split. Since the z_{N+1} coefficient of the hyperplane governs only its offset, it can be discarded, retaining the pure difference information. Hyperplanes of the first three levels of the tree are arranged pictorially in a binary tree fashion in Figure 2.13. The hyperplane images are located where the decision nodes would be in an equivalent binary tree skeleton image: for example, the centered image at the top is the hyperplane image for the node at the top of the tree and the images to its lower left and lower right are the hyperplanes in that node's left and right children, respectively.

The extreme black and white values present in the hyperplanes occur on the outline of the target. Intuitively, the hyperplanes were placed by the ITDT system so as to exploit edge transitions which clearly delineate the pose of the target.

In addition to the tree construction, a series of noise corruption tests were performed. For these tests, the original training data were subject to additive white Gaussian noise at a specified signal-to-noise ratio (SNR). Performance wasn't found to degrade until signal-to-noise ratios near 0dB (one) were reached, at which point the signal power of the target image is equal to the noise power. Figure 2.14 displays several levels of noise corruption, at nine signal-to-noise ratios ranging from 12dB to -12dB.

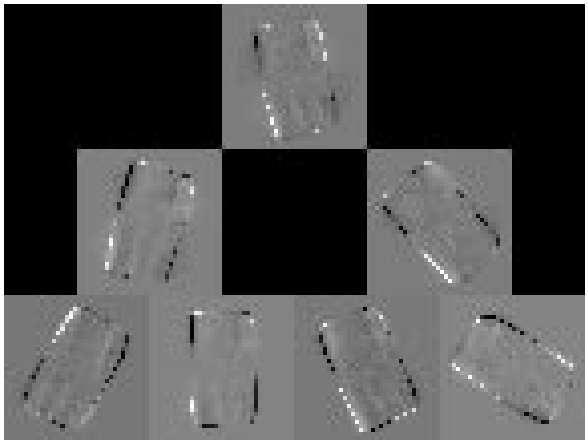

Figure 2.13: The first three levels of hyperplanes in the T72 pose estimation tree.

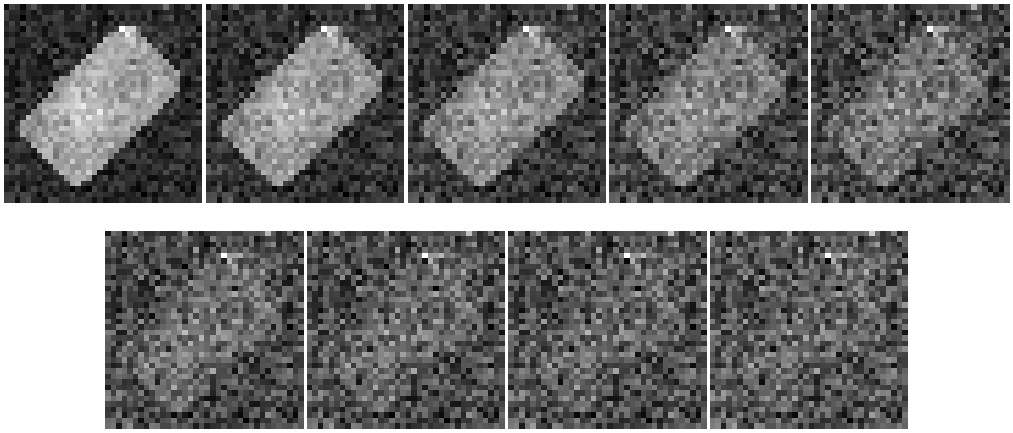

Figure 2.14: Signal to noise ratios: 12dB through -12dB.

This testing brought to light a condition improved by addition of the simulated annealing algorithm. Figure 2.15 shows the error rates versus SNR of this test in the original ITDT system, and error rates versus SNR for the ITDT system with simulated annealing. It is interesting to note that the tree constructed by the original system had an average node split mutual information of 9.75, while the tree constructed with the simulated annealing-based optimization strategy had an average mutual information of 9.85. It is inferred that by taking greater care to find local information maxima which are nearly of the quality of global maxima that the decision tree becomes less sensitive to noise. As such, the tree constructed with the aid of simulated annealing has no classification errors until the noise

power is 3dB higher than the target image power.

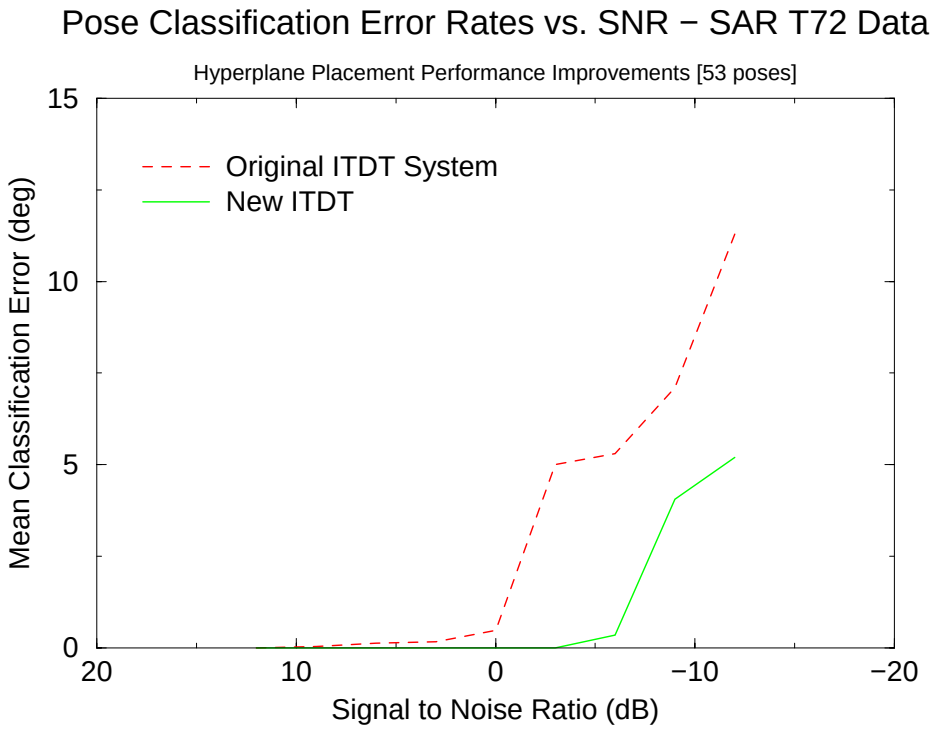

Figure 2.15: Simulated annealing improves hyperplane placement accuracy.

Chapter 3

Exemplar Grouping

In lieu of analytically obtained expressions for probability densities of target classes, most pattern recognition systems allow arbitrarily many feature vectors to contribute to the definition of a particular class of target. Ideally, each feature vector added to the training set widens the set of training data, increasing the accuracy of the classifier. As shown on the left of Figure 3.1, target classes in the ITDT system are entirely characterized by single feature vectors, in this example: x_1, x_2, x_3, x_4. Each terminal node's decision region contains reference to only one of the training feature vectors. On the right side of Figure 3.1, the decision regions identifying particular target classes have been characterized by four groups: $\{x_{11}, x_{12}\}, \{x_{21}\}, \{x_{31}, x_{32}\}$, and $\{x_{41}, x_{42}, x_{43}\}$. If many feature vectors can be incorporated into the definition of a single target class, the ITDT system would be applicable to a whole new range of machine vision problems.

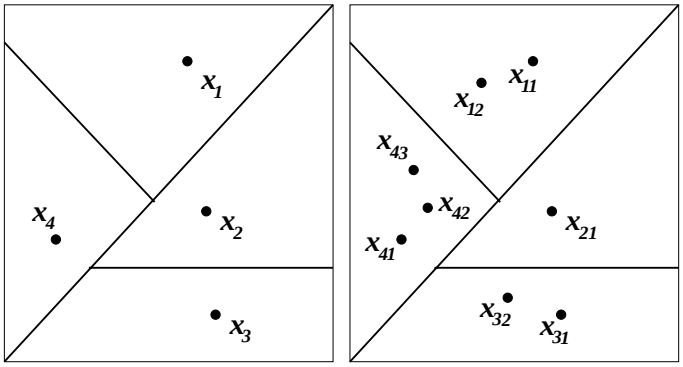

Figure 3.1: Extending the ITDT system to classes specified by multiple feature vectors.

In a practical machine vision problem, a classifier would be designed to discriminate among a number of hypotheses or target classes characterized by measures of many instances drawn from the target classes. If the analytic distributions of the data were known, the classifier would typically be constructed from these analytic representations. Figure 3.2 shows two 2D gamma densities, one with a mean at $(5, 5)$ and another with a mean at $(15, 15)$. As part of another ongoing project in the author's lab, a colleague[1] provided the decision region corresponding to a Bayes minimum error classifier for the two densities, as shown.

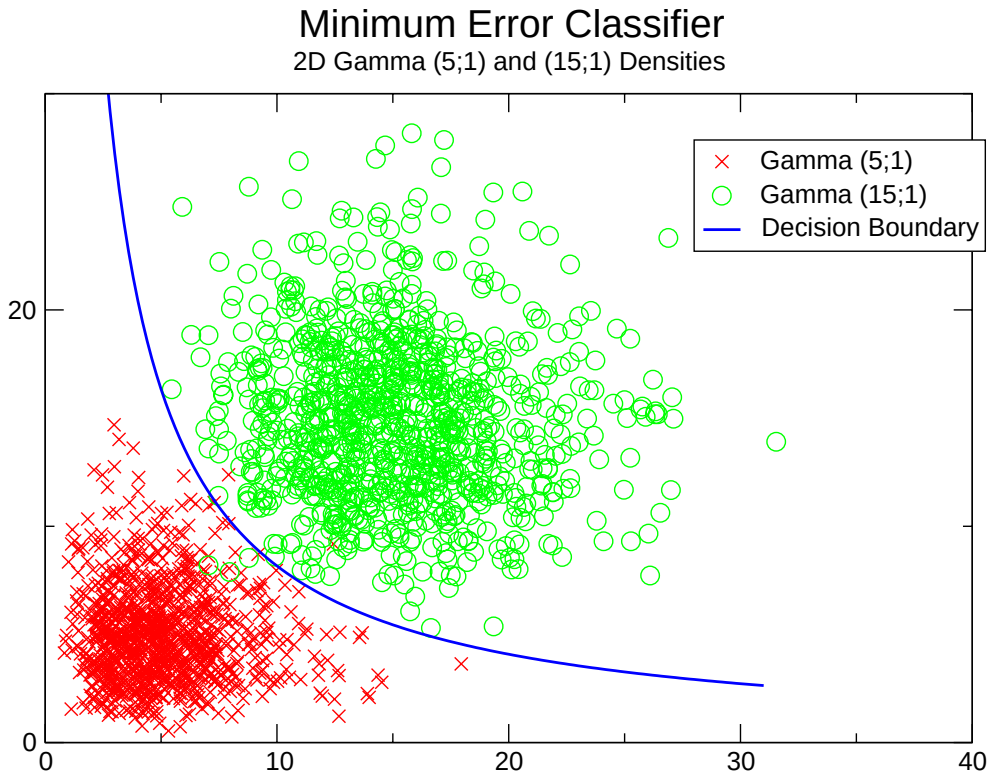

Figure 3.2: Minimum probability of error classifier for the two Gamma densities.

Each 2D gamma density is represented by 1000 samples in the plot. If only these data were available and the densities were unknown, the densities could be estimated from the data, densities could be assumed, or the classifier could incorporate the data using some non-parametric technique.

[1] Jim Kilian, thanks.

Three strategies were envisioned which could be employed to incorporate more than one feature vector to characterize a particular target class in the ITDT system. First, a single mean feature vector can be computed from any set of feature vectors and used in the system as-is. While this technique may work under certain limited conditions, it is often found to discard too much valuable information about the structure of the training data. Second, a decision tree can be constructed for all of the individual training feature vectors, regardless of the group to which they belong. While no modifications would be needed to the core of the ITDT system, the trees formed using this strategy would be so immense as to be completely impractical to form or to store. Finally, a multimodal density could be created for each target class as the weighted sum of the set of target feature vectors which characterize that class, and which are corrupted by i.i.d. Gaussian noise. This is the method which was chosen for further development.

This section describes the method of *exemplar grouping*, which extends the ITDT system to accept arbitrarily-sized sets of feature vectors as distinct classes (groups) of targets. It is not the aim of this extension to accurately model underlying probability distributions, although it may perform acceptably well for that function. Rather, exemplar grouping simply provides a means for grouping many feature vectors together to form groups characterized by multimodal Gaussian distributions. Groups of feature vectors can be formed from any input. In tests, to be discussed, excellent results have been realized in grouping together a series of translations or rotations of a particular target image.

3.1 Characterizing a Group of Feature Vectors

Previously, a set of N feature vectors, $X = \{x_1, x_2, \ldots, x_N\}$ was used to characterize N target classes in the ITDT system. This approach can be extended so that each target class x_i is characterized by a set of M_i feature vectors, $x_i = \{x_{i1}, x_{i2}, \ldots, x_{iM_i}\}$, which are independent and identically distributed Gaussian random processes associated with the each exemplar. Instead of supplying a set of feature vectors for the ITDT system to train on, one supplies a set comprised of N subsets, each of which contains an arbitrary number of feature vectors, M_i, which more completely characterize a particular target class. The set X, formed from N groups, is defined explicitly,

$$X = \{\{x_{11}, x_{12}, \ldots, x_{1M_1}\}, \{x_{21}, x_{22}, \ldots, x_{2M_2}\}, \ldots, \{x_{N1}, x_{N2}, \ldots, x_{NM_i}\}\}. \tag{3.1}$$

Grouping exemplars (which are again take to represent the means of additive i.i.d. Gaussian corrupted feature vectors) in this way only changes the previous lengthy derivation of $I(X;Y)$ by an additional step in the calculation of the conditional probabilities. Each feature vector x_{ij} is handled individually, with an evaluation of the Gaussian cumulative distribution function over an individual minimum distance d_{ij} to the hyperplane. Under exemplar grouping, the probability of an instance of a random process represented by a given exemplar lying in the y_1 region is the same as is given for a feature vector before in equation 2.18. Thus, the probability that such an instance lies in y_1 given the process associated with exemplar x_{ij}, is,

$$P(y_1|x_{ij}) = 1 - Q\left(\frac{d_{ij}}{\sigma}\right). \tag{3.2}$$

The evaluation of the conditional probability that a given group, the set of processes associated with a class of exemplars, lies in the y_1 region is required for the mutual information calculation in the ITDT system. Before, this quantity was given only as the conditional probability that a particular feature vector lay in y_1, equation 2.18. With exemplar grouping, one must now compute the total probability that any feature vector arising from the process which is associated with the group x_i lies in the region y_1, given by,

$$\begin{aligned} P(y_1|x_i) &= \sum_{j=1}^{M_i} P(y_1|x_{ij})P(x_{ij}) && (3.3)\\ &= \sum_{j=1}^{M_i} \left[1 - Q\left(\frac{d_{ij}}{\sigma}\right)\right] P(x_{ij}) && (3.4)\\ &= \frac{1}{M_i}\sum_{j=1}^{M_i} \left[1 - Q\left(\frac{d_{ij}}{\sigma}\right)\right], && (3.5) \end{aligned}$$

which is stated in terms of the conditional densities for each exemplar associated process.

This, in effect, states that the probability that a given group of exemplars, each represented by a random process, lies in region y_1 is the average of the probabilities that

each given exemplar within that group lies in region y_1. Expressions for the y_2 region are obtained similarly as a result of the independence of regions y_2 and y_1,

$$P(y_2|x_{ij}) = Q\left(\frac{d_{ij}}{\sigma}\right) \tag{3.6}$$

$$P(y_2|x_i) = \frac{1}{M_i}\sum_{j=1}^{M_i}\left[Q\left(\frac{d_{ij}}{\sigma}\right)\right] \tag{3.7}$$

Figure 3.3 shows a possible configuration of two groups and a hyperplane. In this scenario, group x_1 is formed from three exemplars, $\{x_{11}, x_{12}, x_{13}\}$ and group x_2 has two exemplars, $\{x_{21}, x_{22}\}$. In this example, the conditional probability that group x_1 lies in region y_1 is the average of the probabilities that x_{11} lies in y_1, x_{12} lies in y_1, and x_{13} lies in y_1.

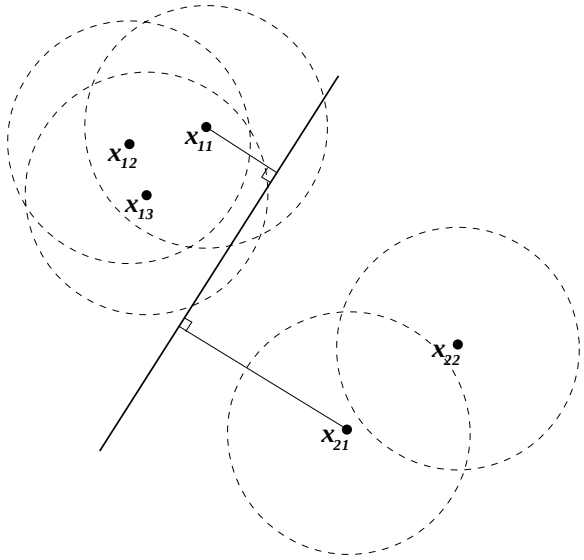

Figure 3.3: Exemplars each contribute to a group's total probability.

Once the conditional probabilities $P(y_1|x_i)$ and $P(y_2|x_i)$ have been computed, the derivation of mutual information of node splits under exemplar grouping is as given previously in section 2.2.1 for the mutual information of node splits using single feature vectors.

Substituting for $P(y_1|x_i)$ and $P(y_2|x_i)$ in equations 2.20 and 2.21, the total probability of any group lying in region y_1 or y_2 is as given below:

$$P(y_1) = \sum_{i=1}^{N} P(y_1|x_i)P(x_i) \tag{3.8}$$

$$= \frac{1}{M_i}\sum_{i=1}^{N}\sum_{j=1}^{M_i}[1 - Q(\frac{d_{ij}}{\sigma})]P(x_i) \tag{3.9}$$

$$= \frac{1}{NM_i}\sum_{i=1}^{N}\sum_{j=1}^{M_i}[1 - Q(\frac{d_{ij}}{\sigma})] \tag{3.10}$$

$$P(y_2) = \frac{1}{NM_i}\sum_{i=1}^{N}\sum_{j=1}^{M_i} Q(\frac{d_{ij}}{\sigma}) \tag{3.11}$$

The remainder of the derivation is as given previously (section 2.2.1), utilizing the expressions for $P(y_1|x_i)$ and $P(y_2|x_i)$ as derived in this section.

3.2 Derivation of Derivatives for the Conjugate Gradient

As the addition of exemplar grouping changed the final mutual information evaluation only by introducing the need for averaging over all vectors in each group, it also only introduces an averaging in the partial derivative evaluations required for the conjugate gradient optimization algorithm. Ryan Tomasetti's derivation in appendix C of the partial derivatives of the mutual information function with respect to distance resulted in the following expressions, restated here to maintain notation consistency with the body of this thesis (eqs. 3.12 - 3.18). The partial derivative expressions are as follows,

$$\frac{\partial H(X)}{\partial d_i} = 0 \tag{3.12}$$

$$\frac{\partial H(Y)}{\partial d_i} = \frac{1}{N}\log\left[\frac{P(Y_1)}{P(Y_2)}\right]\frac{\partial Q\left(\frac{d_i}{\sigma}\right)}{\partial d_i} \tag{3.13}$$

$$\frac{\partial H(X,Y)}{\partial d_i} = \frac{1}{N}\log\left[\frac{P(Y_1|X_i)}{P(Y_2|X_i)}\right]\frac{\partial Q\left(\frac{d_i}{\sigma}\right)}{\partial d_i} \tag{3.14}$$

$$\frac{\partial Q\left(\frac{d_i}{\sigma}\right)}{\partial d_i} = -\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{d_i}{\sigma}\right)^2} \tag{3.15}$$

$$\frac{\partial d_i}{\partial z_k} = \begin{cases} \frac{x_{ik}||Z|| - d_i z_k}{||Z||^2} & if\ k \neq D+1 \\ \frac{1}{||Z||} & if\ k = D+1 \end{cases} \tag{3.16}$$

$$\frac{\partial I(X;Y)}{\partial z_k} = \frac{\partial H(X)}{\partial d_i} + \frac{\partial H(Y)}{\partial d_i} - \frac{\partial H(X,Y)}{\partial d_i} \tag{3.17}$$

$$= \frac{\partial Q\left(\frac{d_i}{\sigma}\right)}{\partial d_i}\log\left[\frac{P(Y_1)P(Y_2|X_i)}{P(Y_2)P(Y_1|X_i)}\right]\frac{\partial d_i}{\partial z_k}. \tag{3.18}$$

Where,

- $X = \{x_1, x_2, \ldots, x_N\}$ is the set of training vectors,
- $Y = \{y_1, y_2\}$ is the set of regions created by the hyperplane,
- $Z = \{z_1, z_2, \ldots, z_{D+1}\}$ is the $D+1$ coefficient hyperplane,
- d_i is the perpendicular distance of x_i to the hyperplane Z,
- $i = 1, 2, \ldots, N$ is the index of training vectors,
- $k = 1, 2, \ldots, D$ is the index of vector components for D dimensional vectors.

The major change in the development with exemplar grouping occurs in the calculation of the conditional probabilities in equation 3.7 on page 54, in which the conditional probabilities $P(y_1|x_i)$ and $P(y_2|x_i)$ are arrived upon through an averaging process over all the vector members of each group. As stated previously in the development of exemplar grouping:

- $X = \{x_1, x_2, \ldots, x_N\}$ is now the set of training exemplar groups,

- $j = 1, 2, \ldots, M_i$ is the index of vectors in the group x_i, so that x_{ij} is the j^{th} vector in the i^{th} group, and
- d_{ij} is the perpendicular distance of the j^{th} vector of group i to the hyperplane Z.

The evaluation of the partial derivative of the $Q(x)$ function (with respect to distance) involves the only significant change to obtain partial derivatives of $H(Y)$ and $H(X,Y)$. The expression for the partial derivative of $Q(d_i/\sigma)$ with respect to distance d_i as given in equation 3.15 is altered under the exemplar grouping extension to be the partial derivative of the average of the evaluations of $Q(d_{ij}/\sigma)$ over all the members of group x_i, given as,

$$\frac{\partial Q\left(\frac{d_i}{\sigma}\right)}{\partial d_i} = \frac{\partial}{\partial d_i}\left[\frac{1}{M_i}\sum_{j=1}^{M_i} Q\left(\frac{d_{ij}}{\sigma}\right)\right] \tag{3.19}$$

$$= -\frac{1}{\sigma M_i\sqrt{2\pi}}\sum_{j=1}^{M_i} e^{-\frac{1}{2}\left(\frac{d_{ij}}{\sigma}\right)^2}. \tag{3.20}$$

Likewise, the partial derivative of the distance from each exemplar group to the hyperplane z, with respect to the hyperplane coefficients z_k, is calculated as the average of the contributions from the vector members of group,

$$\frac{\partial d_i}{\partial z_k} = \frac{\partial}{\partial z_k}\sum_{j=1}^{M_i} d_{ij} \tag{3.21}$$

$$= \begin{cases} \sum_{j=1}^{M_i} \frac{x_{ijk}||Z|| - d_{ij}z_k}{||Z||^2} & if\ k \neq D+1 \\ \frac{1}{||Z||} & if\ k = D+1 \end{cases}. \tag{3.22}$$

An explicit formulation for the partial derivative of mutual information with respect to the hyperplane coefficients, can be obtained by combining the above expressions, yielding, if $k \neq D+1$,

$$\frac{\partial I(X;Y)}{\partial z_k} = -\frac{1}{\sigma||Z||^2 M_i\sqrt{2\pi}}\log\left[\frac{P(Y_1)P(Y_2|X_i)}{P(Y_2)P(Y_1|X_i)}\right]\sum_{j=1}^{M_i}\left[\left(x_{ijk}||Z|| - d_{ij}z_k\right)e^{-\frac{1}{2}\left(\frac{d_{ij}}{\sigma}\right)^2}\right]. \tag{3.23}$$

If $k = D + 1$, we obtain:

$$\frac{\partial I(X;Y)}{\partial z_k} = -\frac{1}{\sigma||Z||M_i\sqrt{2\pi}} \log\left[\frac{P(Y_1)P(Y_2|X_i)}{P(Y_2)P(Y_1|X_i)}\right] \sum_{j=1}^{M_i} e^{-\frac{1}{2}\left(\frac{d_{ij}}{\sigma}\right)^2}. \tag{3.24}$$

3.3 Verification of Operation

As an initial test of the exemplar grouping extension, a decision tree was constructed with one group consisting of only a black image and a second group formed from six poses of an image of a die varying in azimuth angle from 0° to 15°, shown in figure 3.4.

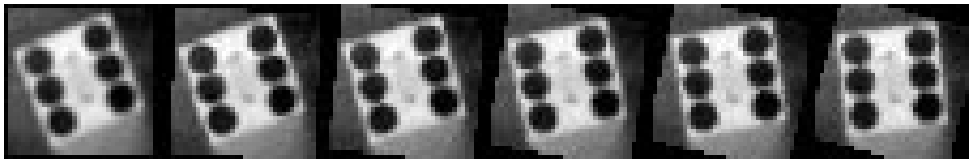

Figure 3.4: Image of a die rotated at 3° increments from 0° to 15°.

The ITDT system produced an optimal size tree with only one decision node, represented in the binary tree diagram in figure 3.5.

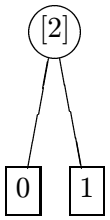

Figure 3.5: Optimal size tree for a two hypothesis classifier.

Figure 3.6 is the hyperplane created to decide between the six die poses and a black image. In it, the six poses of the die are visibly blurred together - confirming the notion of the averaging behavior of exemplar grouping. Notably, the bright white section found in the center of the image corresponds to the center section of the die, and is that part which is observed not to change under rotation of the die. Hence, as we have come to expect, the hyperplane placement exploits that portion of the image which best exemplifies the difference between the die and the background, the two classes under consideration.

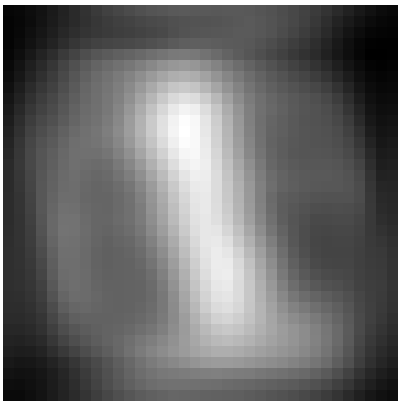

Figure 3.6: Hyperplane image representing the difference between 6 die poses and an entirely black image.

3.4 Two-Hypothesis Classifier for 2D Gamma Densities

A decision tree was constructed from the two 2D gamma distributed data depicted in the introduction to this section. Figure 3.2 shows the analytically obtained minimum Bayes error decision region for this set of data.

The statistics shown in Table 3.1 reveal that the ITDT system had difficulty in splitting off some of the training data, resulting in a tree that required a maximum of 9 decisions, and an average of 5.5 decisions. Two quantities are presented by which this height can be judged. First, the *worst-case optimal height* is the optimal height of a tree constructed to identify each datum with which it was trained. In this case, identifying each of the 2000 training exemplars would require a tree which was 11 decision nodes high. Second, the *best-case optimal height* is the optimal height of a tree constructed to classify the number of exemplar groups; here, the two exemplar groups would be characterized optimally by a decision tree which was 1 decision node high.

From figure 3.2 it is obvious that a single hyperplane will not suffice to represent the optimal decision surface constructed from the analytic class representation. Since in this 2D space a hyperplane is represented by a single straight line, one would expect a decision tree involving at least several decisions to obtain a similarly discriminatory result. As can be seen in Figure 3.7, the tree constructed by the ITDT system is rather strange looking. There are only two main branches, formed during the initial split of the data into its two primary groups. Experiencing difficulty in understanding the behavior of the system, the author developed slope-intercept equations for all sixteen hyperplanes and applied colored pencils

Time to build tree	5s
Maximum decisions ($\max h$)	9
Average height ($\overline{h}$)	5.5
Worst-Case Optimal height	11
Best-Case Optimal height	1
Decision nodes	16
Dimensions	2
Groups	2 (1000 members each)
Total exemplars	2000

Table 3.1: 2D Gamma Classification Tree

to a hard copy of the decision space, filling in regions corresponding to each distribution. Figure 3.9 on page 63 is a scanned reproduction of that work.

It is essential to realize that the ITDT system will always correctly classify its input training vectors. Thus, we can anticipate that the resulting decision system will not settle for even an approximation of the curved decision surface in Figure 3.2. Rather, a set of hyperplanes and associated decisions will be formed that successfully dissect the given feature space exemplar representations so that each one is properly classified. Figure 3.9 demonstrates that even under poor separation conditions, the decision tree construction process will go "out of its way" to segment the input data. Most notable in this figure is the small island of red centered at approximately $(0.3, 0.3)$, created simply to notch out a lone vector. Similar cases can be found elsewhere in this diagram.

The ITDT system was again tested with a pair of gamma densities, however, this time, only 100 trials in each group were utilized. The resultant datum are non-overlapping and are *linearly separable*. As would now be expected, the ITDT system chooses a single best-fit hyperplane, as shown in the plot in Figure 3.8.

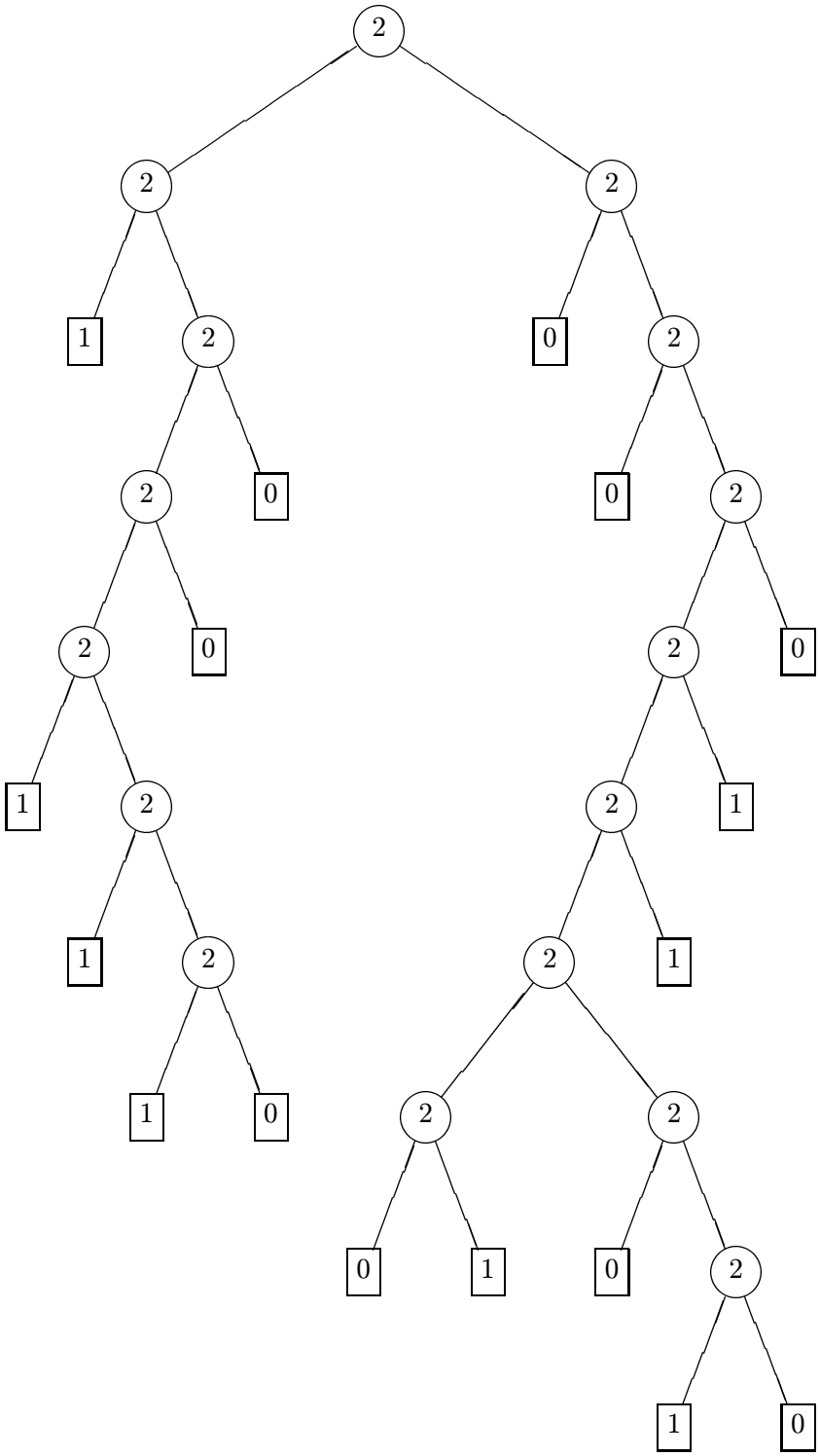

Figure 3.7: 2D Gamma Density Classification Tree.

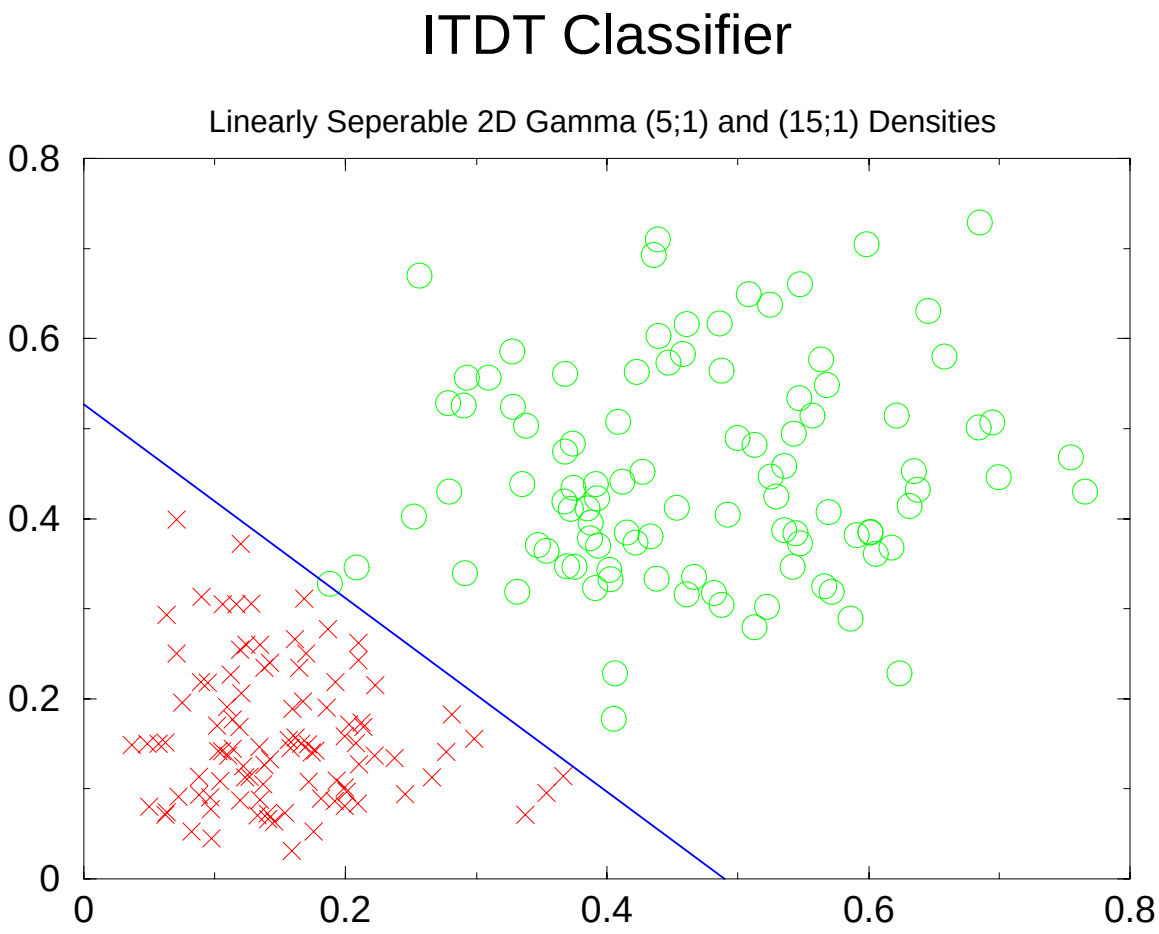

Figure 3.8: The hyperplane in the single ITDT decision node required to separate the two linearly separable gamma distributed sets of data.

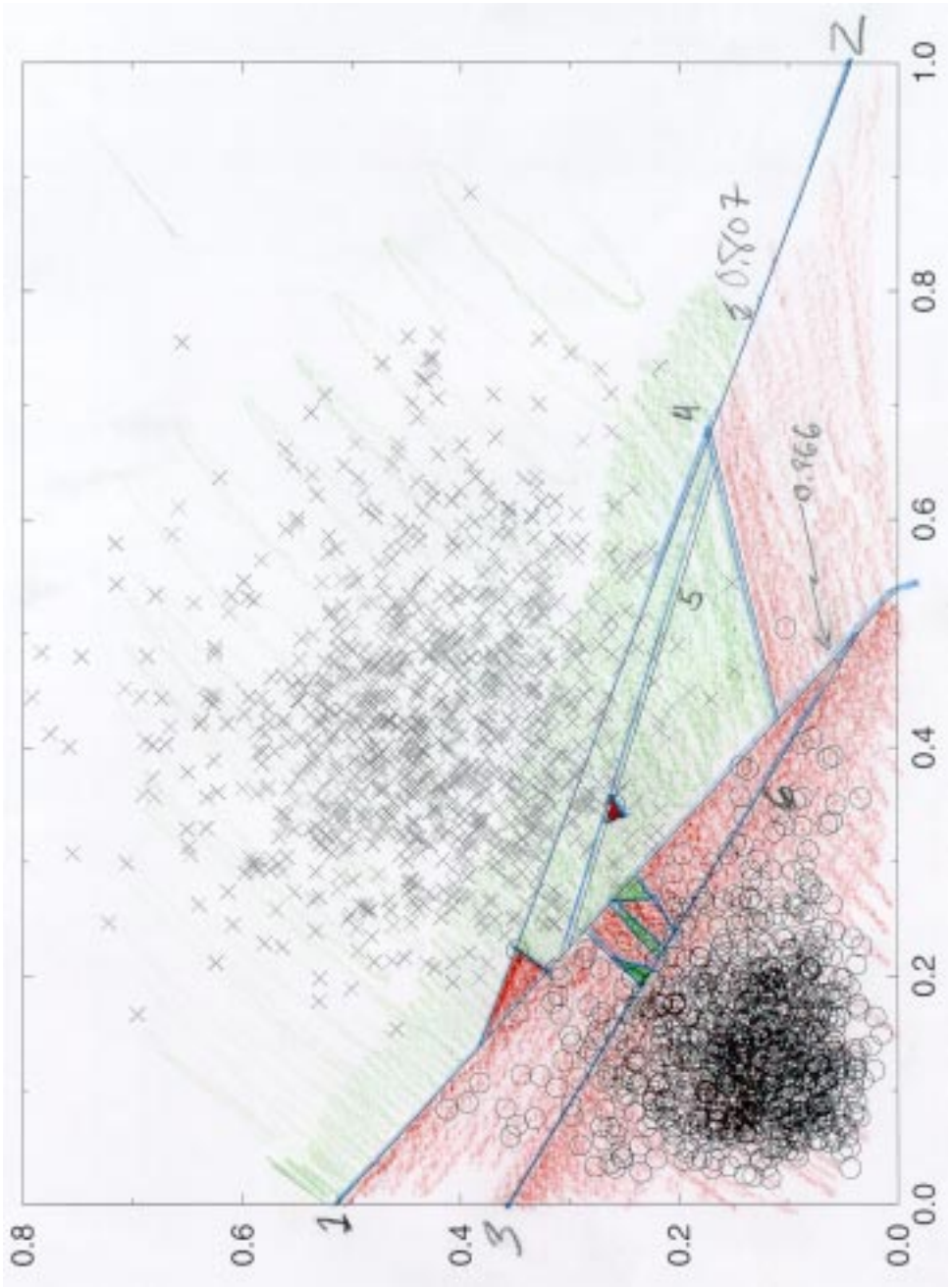

Figure 3.9: Decision regions for the 2D Gamma classifier carefully colored in by the author.

Chapter 4

Object Recognition Independent of Image Shift and Rotation

In general, machine vision systems require some degree of shift (translation) and rotation invariance. Very rarely will a situation arise in which an object's location in a scene image is known to sub-pixel accuracy. Even in controlled environments, objects shifted a few pixels or rotated slightly can wreak havoc with the discrimination capabilities of machine vision systems. The three objects shown in figure 4.1 differ only slightly, but enough to confuse a machine vision system which isn't designed to cope with translation and rotation.

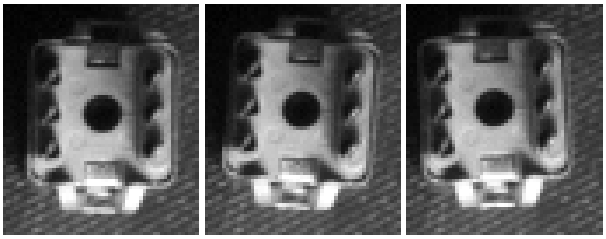

Figure 4.1: Even in controlled circumstances, slight variations in position and orientation arise.

Developing a means to obtain correct decisions regarding class membership despite target transformations was a primary goal for this thesis. As presented in the prior chapter, the ITDT system lacks an efficient means for handling arbitrary target shifts coupled with changes in orientation. This chapter chronicles the strategies employed in the ITDT system to handle unknown image shift and orientation: *search with distance metric, exemplar*

grouping and sliding window correlation.

4.1 Search with Distance Measure

When translation invariance cannot be built directly into the model of an object, a brute force search must be employed to locate the object in a test scene. Algorithms using this method, in general, scan a window across the test image, applying a test to the region of interest inside the window. A three stage implementation is typical:

1. Test for the object at each possible location in the test image, estimating which entry in the object model is the best match for that location in the test image. Figure 4.2 shows such a test, in which a window is scanned across the test scene image.

2. Compute a distance measure between the estimated object and the object within the current test window.

3. Report the location of the minimum distance of the above obtained over all such positionings of the test window.

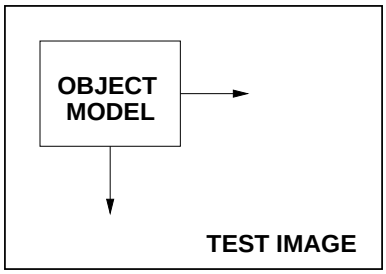

Figure 4.2: Simple search of a test image.

Excluding the possibility of objects which have partially slipped out of the test image, a $M \times M$ object model has $(N - M + 1)^2$ possible locations in a $N \times N$ test image.

Difficulty in using the search method with the ITDT system arises due to the need for a measure of goodness-of-fit of the object model at each test location. The ITDT system doesn't provide such a measure directly - searching the decision tree results only in a reference to the best matching target feature vector. However, if one is willing to preserve the original set of training feature vectors in addition to the decision tree, a Euclidean

distance measure can be computed between the best match training feature vector $\mathbf{x_{match}}$ and the vector formed by the region in the test image under test $\mathbf{y_i}$,

$$D_i = \sqrt{< (\mathbf{y_i} - \mathbf{x_{match}}), (\mathbf{y_i} - \mathbf{x_{match}}) >}. \tag{4.1}$$

The set of distances D_i form a distance metric image, in which the value at each pixel is a goodness-of-fit distance metric between the best match at that location determined by the ITDT system and the underlying region of interest. Searching the metric image for the minimum distance will yield the most probable location of a model object in the test image.

Computing the distance measures adds an additional M^2-dimension dot product operation for each of $N - M + 1$ test locations. For many applications, the conventional search with distance measure algorithm may be too computationally expensive.

4.2 Training for Rotation and Translation Independence

To achieve some measure shift or rotation invariance, it is possible to train the ITDT system with sets of images which have undergone a great many combinations of shifts and rotations. One can use the method of exemplar grouping to group these orientations or shifts of like images into classes, allowing the ITDT construction process to determine the minimum number of information maximizing decisions which need to be made to correctly classify the training imagery. However, it will be seen that even in the simplest cases, a massive amount of data may be required to train the system sufficiently.

Two strategies for handling shift and rotation invariance are implemented in the ITDT system as part of a training imagery generation stage. Given a single image of a target to classify, these training strategies can generate sets of training imagery in which the target is transformed over a user-specified ranges of shift and orientation. For recognition of objects which are smaller than the field of view of the camera, one can use the target mask-based algorithm presented in section 4.2.1; for images which are larger than the field of view, one can use the region-of-interest extraction algorithm described in section 4.2.2.

4.2.1 Translation and Rotation Training With Target Mask Images

In the first scenario handled by the ITDT training algorithms images are generated by a camera, which remains fixed on a relatively unchanging background while small objects

move into the field of view. The small objects may be at unknown orientation, shift, or both. Figure 4.3 shows three images typical of this application.

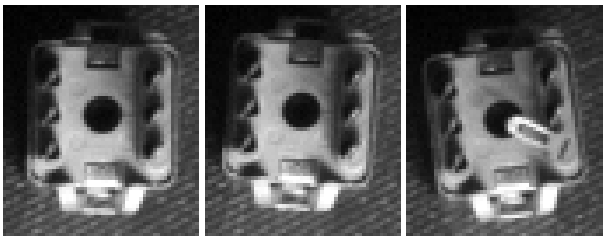

Figure 4.3: Small parts on a fixed background with small relative shifts and rotations.

In the example motivated by the images in Figure 4.3, the user desires to recognize three classes of targets: *no object present, object,* and *object with bent pin.* Three images, shown in Figure 4.4, are recorded and used to train the system to recognize each of the classes over several hundred translations and rotations. These images represent single exemplars of the empty background, a good object, and an object with a bent pin which would be rejected in a manufacturing process.

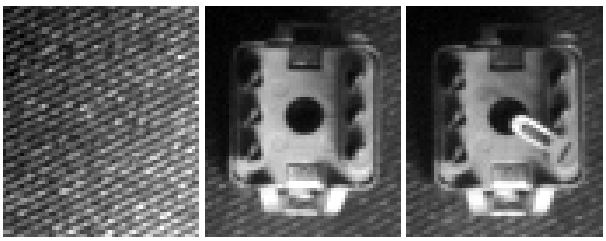

Figure 4.4: Training images for a three hypothesis classifier: no object, object, and an object with a bent bin.

Were the training system to rotate and translate this input imagery as stands to obtain the set of training vectors, two unwanted artifacts would be produced in the resultant images shown in Figure 4.5. First, the background would move with the object of interest, producing the same effect as if the camera were moved. This doesn't adequately model our physical setup, in which only the object of interest moves, while the background remains stationary. Second, the transformation algorithm is unable to create pixel data to fill in sections of the image which are rotated or shifted in from beyond the bounds of the original training data. These black, triangular-shaped artifacts can be observed in the transformed

images in Figure 4.5.

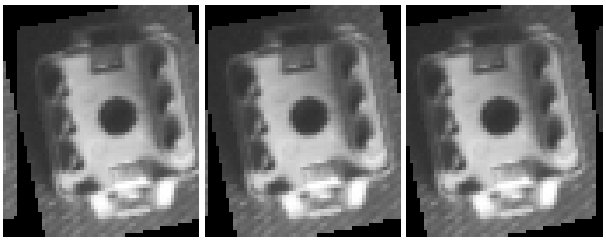

Figure 4.5: Simply translating and rotating training images introduces unwanted artifacts into the training imagery.

When the ITDT system was trained on the imagery in Figure 4.5, two of the resultant hyperplanes shown in Figure 4.6 indicate that the artifacts were included as discriminating characteristics of the training imagery, used to position the hyperplanes to maximum information gain. The learning nature of the ITDT system cannot compensate for inadequate training data.

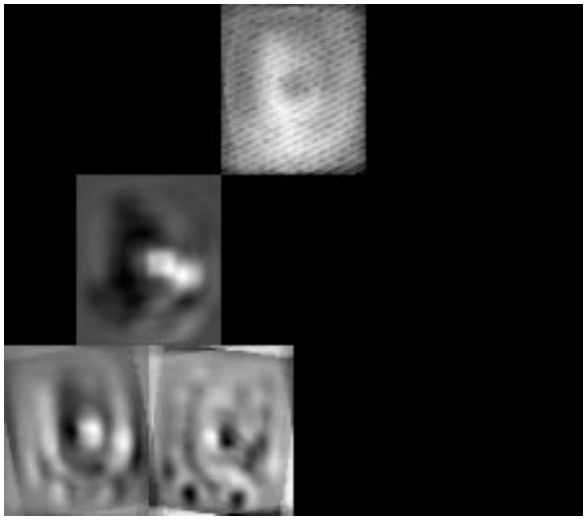

Figure 4.6: Artifacts in the training imagery are used as classification information in the decision tree.

To avoid introducing unwanted artifacts, the ITDT training system can employ a user-created mask of the training objects, allowing the object of interest to be projected onto the static background. Figure 4.7 shows the background, a training image and the binary mask image.

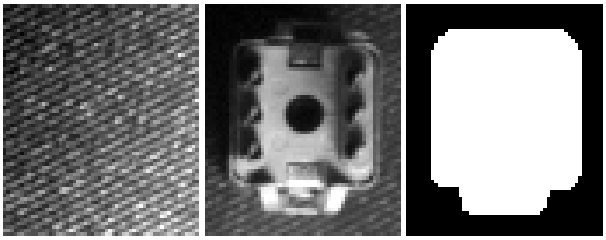

Figure 4.7: A training image mask is used to project the training object onto a static background.

The usage of masking eliminates the major artifacts observed earlier from the training imagery, as shown in Figure 4.8. Hyperplanes for these training images which are free of the artifacts introduced earlier are shown in the next section in Figure 4.11 on page 72.

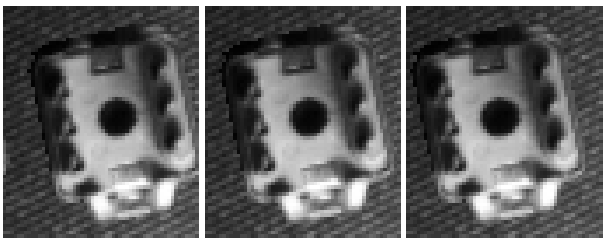

Figure 4.8: Input objects projected onto the supplied background using a mask image.

Assembly Line Part Verification Test

A three-hypothesis decision tree was constructed to detect the three simple objects shown in Figure 4.9: no object, object, and an object with a bent pin. This test simulates a machine vision system employed in a roughly constrained object placement system, in which the general location of an object is known, but the object may have been slightly shifted or rotated. In this example, the feature vectors formed from the digital training images have been normalized to unit magnitude and hyperplanes are constrained to pass through the origin.

The training data was generated using the method of object masking presented previously. For every 3° of rotation from $-12°$ to 9° and every one pixel shift from -10 pixels to $+9$ pixels in both X and Y image dimensions. Table 4.1 shows the statistics collected during the tree construction process. Using the method of exemplar grouping, the size of the

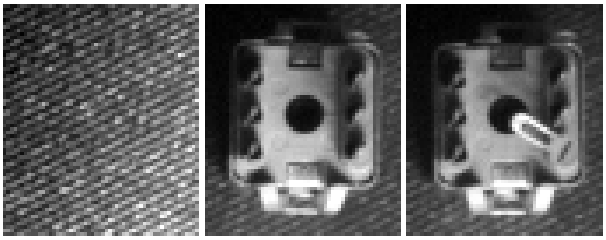

Figure 4.9: Training images.

search space was dramatically reduced. More than sixteen-hundred images were produced by the training algorithm, forming three groups. Although the optimal size tree for three groups would have only two decision nodes, considering the massive size of the training set which would require sixteen-hundred decision nodes to characterize, the fifteen decision nodes generated by the ITDT system demonstrate a remarkable reduction in model data.

Time to build tree	47m 30s
Maximum decisions ($\max h$)	6
Average height ($\overline{h}$)	3.7
Best-Case Optimal height	1.6
Worst-Case Optimal height	10.6
Nodes	15
Dimensions	3696 (56x66)
Groups	3 (members in groups: 1, 800, 800)

Table 4.1: Three hypothesis assembly line detection tree

The tree in Figure 4.10 shows that the *no object* hypothesis was immediately split off to the right, and that the rest of the tree below that point is well-balanced. Examining the hyperplanes in Figure 4.11 reveals how the ITDT system illuminated the bent pin as the major difference between the two target object hypotheses.

Searching this tree to produce correct classifications required between 1ms and 3ms on a 350MHz Intel Pentium II based PC, requiring approximately $500\mu s$ at each decision node. For comparison, if a sequential search were conducted over the same set of training data, an average of $1601/2 \approx 800$ comparisons would be required, with a search time of 400ms using a similar comparison operation.

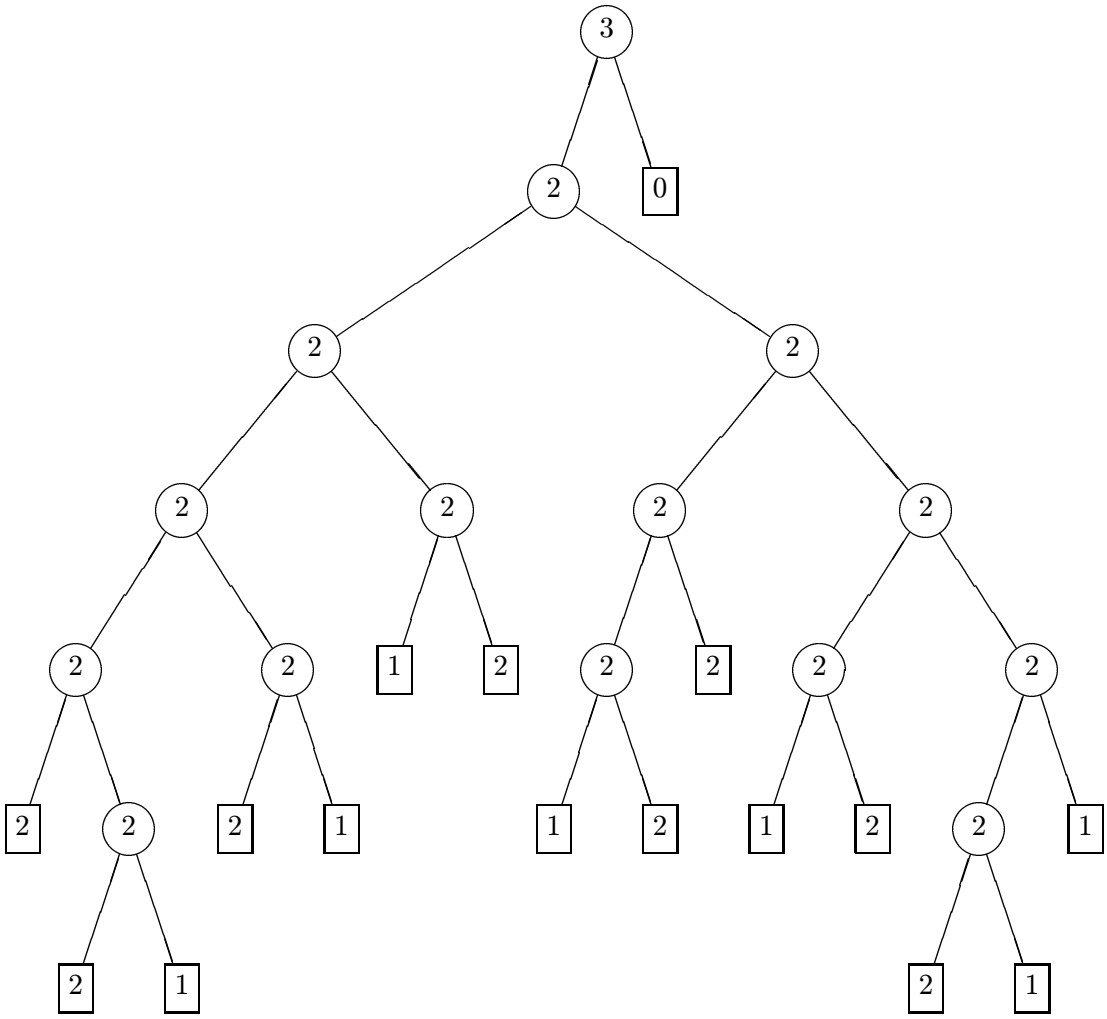

Figure 4.10: This decision tree skeleton represents more than 1600 training images in a three hypothesis assembly line part classifier.

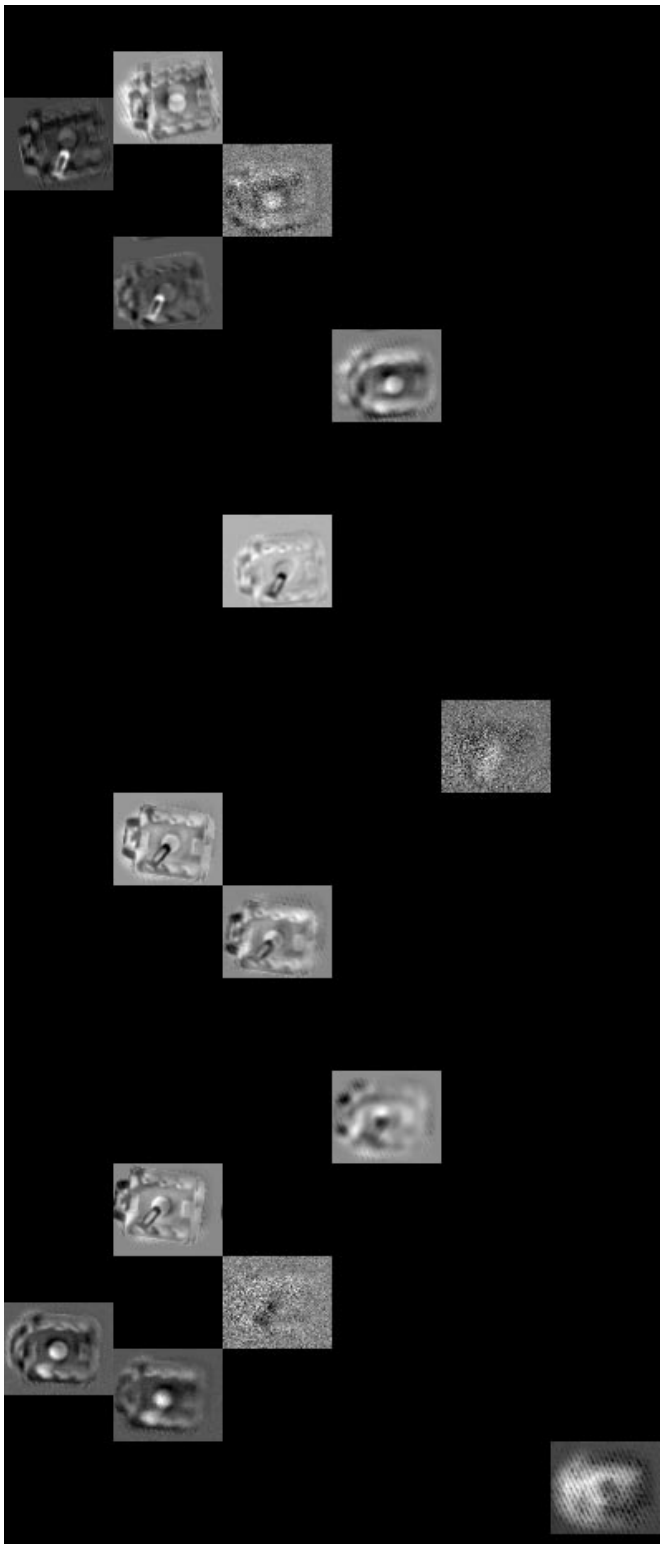

Figure 4.11: Hyperplanes in this hypothesis detection decision tree. Only the left side of the tree is shown. The image is properly viewed by rotating the paper 90° counter-clockwise.

4.2.2 Translation and Rotation Training With Region-of-Interest Extraction

A second configuration is possible wherein the feature to detect is part of a larger object which occupies more than the field of view of the camera. This is the case with large circuit boards, or in the following example, with UPC symbols on soda cans. In most cases, it is usually known to some accuracy where the object to be recognized will lie in the figure, and a region of interest (ROI) is extracted for test (Figure 4.12).[1]

Figure 4.12: A region of interest is extracted from the test scene.

This situation is best addressed by obtaining training images by means of performing the necessary transformations on an image of the entire scene, and subsequently extracting the region of interest, rather than by performing the transformations on the ROI explicitly. In doing so, information previously on the outside edge of the ROI is able to enter the ROI for appropriate shifts and rotations, as depicted in Figure 4.13.

Soda Can UPC Symbol Locator Test

Research conducted by a colleague[2] in the machine vision lab produced a correlation-based system for recognizing UPC symbols on the cans of several brands of soda. Performance was acceptable, although classifications typically required several hundred milliseconds on an Intel Pentium II 350MHz PC.

[1] Coke, Coca-Cola, and the Coke can are registered trademarks of the Coca-Cola company.
[2] John Sullivan, thanks.

Figure 4.13: Extraction of ROI's *after* translation of target scene images yields good training images from objects which are larger than the region of interest.

Figure 4.14: Three target UPC symbols are to be classified by the ITDT system.

A decision tree was constructed to detect the three UPC symbols shown in Figure 4.14 by extracting a region of interest, delineated by the box in Figure 4.15. The training data was constructed from 125 (25×5) shifts of each training image using the ROI extraction method described in the previous section. No rotation invariance was required.

Table 4.2 contains the statistics collected from the tree construction process. Remarkably, only a maximum of three decisions is required to identify any target, resulting in test times of $1 - 3$ms due to the large size (102×86) of each training image. With 375 total training images, it would have taken an average of nearly 200ms to search sequentially through this set. Eighteen test scenes were acquired and classified correctly with the

Figure 4.15: A region of interest is extracted for UPC symbol identification.

resultant decision tree.

Time to build tree	9m 9s
Maximum decisions ($\max h$)	3
Average height ($\overline{h}$)	2.3
Best-Case Optimal height	1.6
Worst-Case Optimal height	8.6
Decision Nodes	4
Dimensions	8772 (102x86)
Groups	3 (125 members each)

Table 4.2: Soda Can UPC Symbol Locator Tree Table

Figures 4.16 and 4.2.2 depict the decision tree constructed to classify the three UPC symbols. An interesting phenomenon can be observed in the hyperplane images in Figure 4.2.2. One observes a speckled nature to the second-level hyperplane, believed initially by the author to reflect a relatively low information gain measure for that hyperplane placement, reported during the tree construction process. In addition to the statistics reported in the table, the ITDT system reports the mutual information gain to which the dynamic hyperplane positioning algorithm finally converged. In this particular tree, a decision made at the top node gains 0.92 bits of information, decisions made at either of the two third-level nodes gain a full bit of information, but a decision at the second level node only offers a gain of 0.66 bits of information. Low information measures indicate that the

feature vectors could not be well separated with a hyperplane. It is possible that these low information gains are caused by a convergence to a low-quality local information maximum during dynamic hyperplane placement, although simulated annealing reduces the chances of converging to low-quality local maxima. More likely, low information gain measures indicate that a single hyperplane is insufficient to divide a particular arrangement of feature vectors in the high dimensional space.

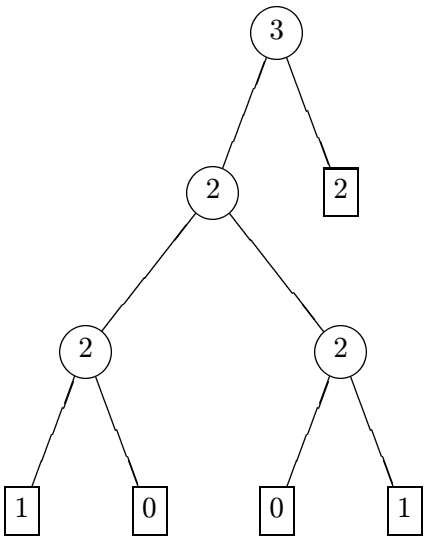

Figure 4.16: Soda can UPC symbol identification decision tree.

After some experimentation, it was determined that the speckled effect described above was caused by an assumption made early in the training process. During the training process, each image in this set of data was normalized to unit vector magnitude to minimize the classifier's sensitivity to lighting variations, and the hyperplanes were constrained to pass through the origin so as to discriminate only based on orientation, not image magnitude. When both the unit-magnitude constraint and the zero hyperplane offset constraint were removed from the system, the smoother appearing hyperplanes in Figure 4.2.2 were the result.

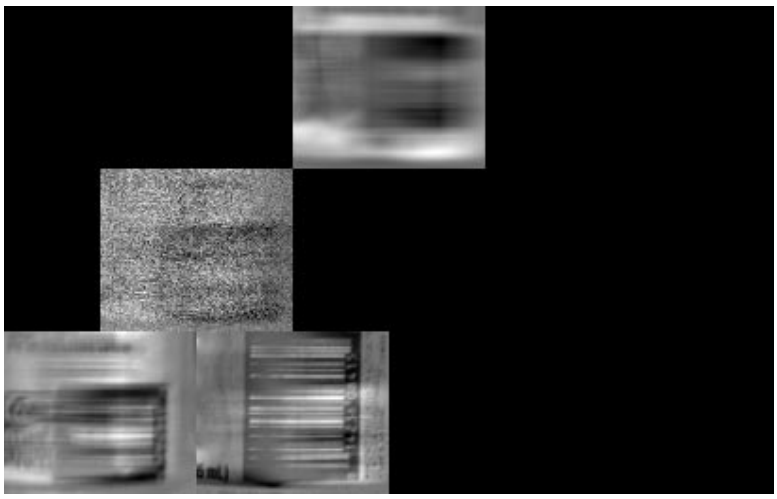

Figure 4.17: Hyperplanes from the soda can UPC symbol identification tree.

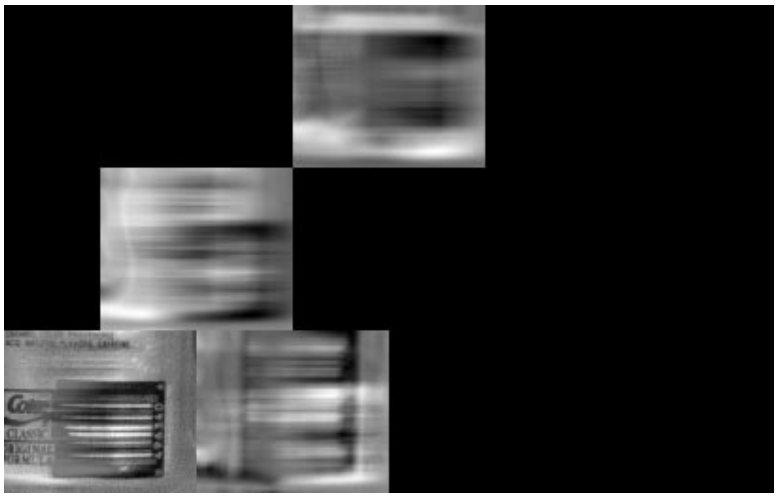

Figure 4.18: The speckle effect at the second level hyperplane was eliminated by removing both the unit-magnitude normalization and zero offset hyperplane constraints.

4.3 Two Stage Pose Classification With Match Metric

For all of the previous tests in which exemplar grouping was used, each target class was comprised of multiple images produced by a training algorithm which generates a set of transformed images from a small number of input images. The decision trees constructed by the ITDT system for these training data were minimally sized, as they were constructed to discriminate among a small number of groups, rather than among each image in the large set of individual training images. However, the fast search times afforded by this method only produce a decision regarding which among the target classes does the test image most resemble. More useful in a robust machine vision system implementation would be a decision accompanied by a match metric, a goodness-of-fit of the test image to the chosen target image from the model.

As it stands, there exists no way to extract a match metric from decision trees which used exemplar grouping. Any classification performed by such a tree results in a decision on an entire set of feature vectors, with no clear means of extracting a difference between the test image and the set of images in a particular target class. Two stage classification is intended to alleviate this problem. If one can apply two decision trees, one may, for instance, use one tree to determine the location of an object in an image, regardless of its orientation. Subsequently, a pose estimation decision tree can be applied at that location which does not use exemplar grouping. In this pose tree may be stored the original set of training images, upon which the pose estimate will act as an index to select a particular training image. A distance measure can then be generated to compare the training image of the object at a particular pose to the object at the identified location in the test image.

In the following discussion, we will consider the problem of estimating the pose and location of a die within a 33×33 pixel image, shown in Figure 4.19.

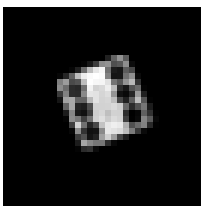

Figure 4.19: Die training image.

4.3.1 Stage 1: Pose Invariant Die Shift Estimation

In the ITDT two-stage classification process, the initial stage is comprised of a decision tree which classifies a particular location in a test image at which an object lies, independent of the pose of that image. This tree was constructed from 225 groups, one group for each image location over ± 7 pixel shifts in the X and Y image dimensions. Each group is formed from eighteen poses of the die, ranging from 0° to 180°.

The statistics of the constructed decision tree in Table 4.3, show that the tree offers a modest reduction in the amount of data to be searched, with an average height of 8.85, and with 779 decision nodes. Were a tree constructed from all 4050 exemplars individually, it would have taken far longer to construct, yielding a very large tree with 4049 decision nodes and 12 levels. Memory usage and storage size are considerations when trees become very large; representing each hyperplane as $N-$dimensional arrays of double-precision floating points requires 34MB of memory for this 4049 node tree, while only 6.5MB would be required for the 779 node first-stage tree.

Time to build tree	5h 0m
Maximum decisions ($\max h$)	12
Average height ($\overline{h}$)	8.85
Worst-Case Optimal height	12
Best-Case Optimal height	7.8
Nodes	779
Dimensions	(33x33)
Groups	225 (18 members each)

Table 4.3: Pose Invariant Die Shift Estimation Tree

The first four levels of hyperplanes in the tree are depicted in Figure 4.20. As we would now expect, the round shapes in each hyperplane are due to the averaging behavior of exemplar grouping over the set of eighteen poses in each target class.

Pose Estimation Stage

Once a location has been determined by the shift estimation tree, a die-sized region of interest is extracted from the test image at this location, forming a test image which may be classified for pose. This rapidly built tree was of optimal height (4.2) for the eighteen poses which it represents, based on the statistics shown in Table 4.4. The hyperplane images

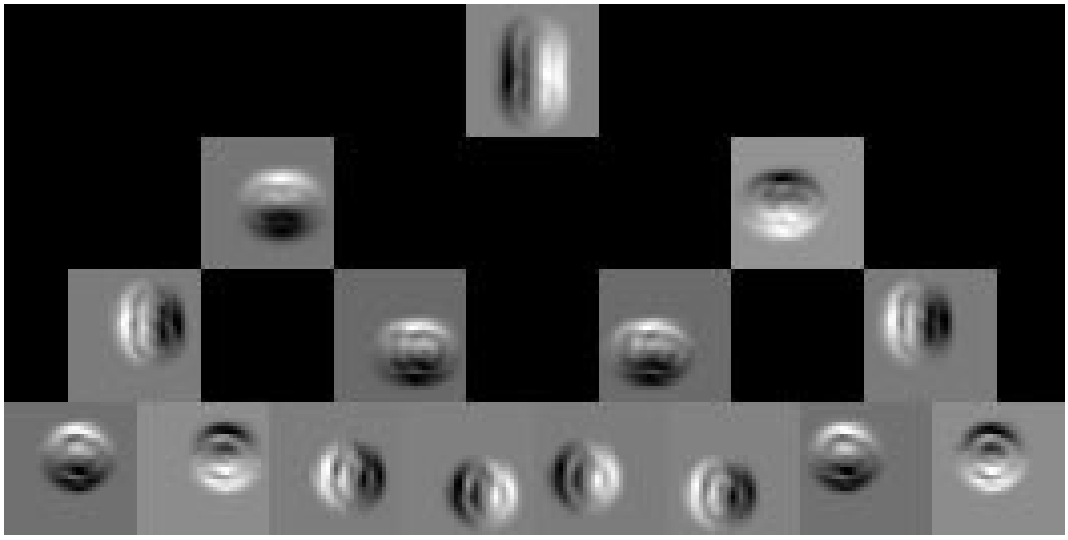

Figure 4.20: Hyperplanes in the Pose Invariant Die Shift Estimation Tree

represented in Figure 4.21 accentuate edge information of the die, including the black dots, much like the earlier T72 pose classification tree presented in section 2.3.

Time to build tree	0m 5s
Maximum decisions ($\max h$)	5
Average height ($\overline{h}$)	4.2
Optimal height	4.2
Dimensions	272 (17x16)
Groups	18

Table 4.4: Die Pose Estimation Tree

All test cases were properly classified by this two-stage system. However, there is a tradeoff in using such a two-stage classifier: storage size versus test time, shown in Table 4.5.

If a single tree is constructed, a maximum of twelve decisions will be required, but more than four thousand decision node hyperplanes must be stored by the system. If a 2-stage strategy is used, the maximum number of decisions raises to seventeen, but only approximately eight hundred decision nodes need to be stored. Each will perform with the same accuracy, and for each, a match metric can be computed. It is the author's belief that this tradeoff is best resolved on a case to case basis.

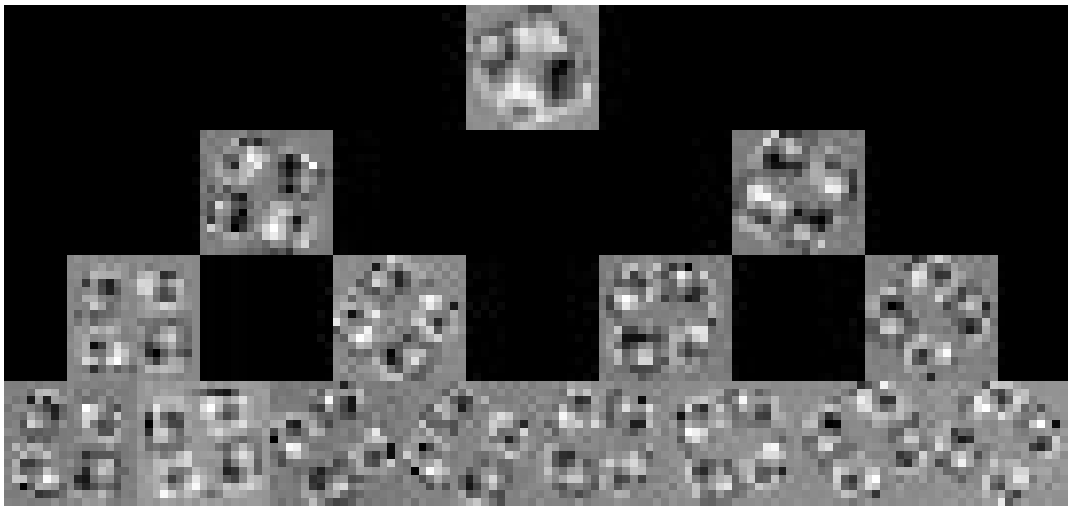

Figure 4.21: Hyperplanes in the Die Pose Estimation Tree

	Decision Nodes	Total Tree Height
2-stage system	797	17
single tree pose classifier	4049	12

Table 4.5: Tradeoffs in Decision Tree Pose Classification Strategies

Chapter 5

Conclusions

The Information Theoretic Decision Tree machine vision system described in this work has proven to offer extremely rapid classification times for complex appearance-based models. Fundamental to the system's operation, information theory has proven an effective mechanism for classifying machine vision feature vectors. The ITDT system constructs a binary decision tree by recursively splitting a set of training feature vectors with hyperplanes placed to maximize the information gained by a particular split - minimizing the number of binary questions which must be asked to correctly classify the training data. Using a local optimization algorithm wrapped with simulated annealing, hyperplanes are placed at nearly globally optimal information maxima, without knowledge of underlying probability densities.

The author contributed new capabilities and optimizations to the ITDT system, which began development three years ago under an Army Research Office contract. The author:

1. Developed the concept of exemplar grouping to incorporate translation and rotation independence into the ITDT system.

2. Implemented a training system by which rotation and translation independence can be achieved with limited target imagery.

3. Enhanced the local optimization procedure for hyperplane placement, thereby reducing the size of the decision trees constructed. As a result, test speed is increased and the system is less susceptible to noise.

4. Optimized the tree construction process, yielding construction times nearly an order

of magnitude faster, and enabling the system to operate on larger data sets.

This document is the first work to describe the ITDT system in a detailed fashion. As such, the detailed treatment of the theoretical background presented herein is solely the author's. Much was learned of the fusion between information theory and machine vision during the course of this thesis. It is the author's hope that this document preserves this knowledge, enabling future researchers to leverage against it to reach an even greater understanding of the problem of machine vision.

Appendix A

The Bayes Likelihood Ratio Test (LRT) for Gaussian Distributions

The binary Bayes likelihood ratio test has been proven[30, pp. 24-27] to minimize the average cost of a particular decision in a classifier. In general, this cost can be assigned arbitrarily to bias any of four outcomes:

- C_{00}: the cost of deciding hypothesis zero when hypothesis zero is correct.
- C_{01}: the cost of deciding hypothesis zero when hypothesis one is correct.
- C_{11}: the cost of deciding hypothesis one when hypothesis one is correct.
- C_{10}: the cost of deciding hypothesis one when hypothesis zero is correct.

In some problems, the cost of deciding hypothesis zero (H_0) when H_1 was the correct choice may actually be higher than the reverse, deciding H_1 when H_0 was correct. Automatic target recognition (ATR) in particular assigns these costs indirectly. In a typical ATR problem, a machine vision system is designed to decide, given a digital image, whether a target is present in the image or no target is present. All non-target features or clutter (trees, fields, some dwellings, etc.) invariably fall under H_0 and targets fall under H_1. In a military application, a high cost of identifying clutter as a target (a false alarm) is based on the cost of sending a vehicle to that location. On the other hand, the low cost of missing a target which was improperly classified as clutter is justified because, invariably, other nearby targets which weren't missed will warrant initiating a response anyway.

In the problem and solution regions presented in figure 1.11, the costs were set to their extremes. The cost of either correct decision C_{00} and C_{11} was set to zero, or no cost at all; false alarms (C_{10}) and misses (C_{01}) were both assigned a cost of one.

Apart from the costs of classification, some additional pieces of information are required to evaluate the Bayes likelihood ratio test: a priori probabilities and distributions for each hypothesis. In many problems the hypotheses can be assumed equally likely as it is in this binary detection problem, where $P(H_0) = P(H_1) = 1/2$. The distributions of the data comprising each hypothesis can be estimated from observations of real data. In our example, each is a Gaussian normal distribution: $p(H_0) = N(5; 2.5)$ and $p(H_1) = N(15; 2.5)$.

A detailed derivation of the Bayes criterion likelihood ratio test can be found in Van Trees's classic text *Detection, Estimation, and Modulation Theory* [30, pp. 24-27]. Observing a feature vector $\mathbf{R}$, equation A.1 defines the optimal Bayes decision in terms of the underlying probability densities and costs.

$$\frac{p(\mathbf{R}|H_1)}{p(\mathbf{R}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P(H_0)(C_{10} - C_{00})}{P(H_1)(C_{01} - C_{11})} \tag{A.1}$$

If the costs and a priori probabilities are assigned as discussed, we obtain,

$$\frac{p(\mathbf{R}|H_1)}{p(\mathbf{R}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{(1/2)(1-0)}{(1/2)(1-0)} \tag{A.2}$$

$$\frac{p(\mathbf{R}|H_1)}{p(\mathbf{R}|H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} 1. \tag{A.3}$$

The threshold of the test, η, on the right side of the inequality is now equal to 1 for our example. The left side of the inequality is the likelihood ratio, defined as the ratio of the conditional probabilities of feature vector $\mathbf{R}$ given hypothesis one in the numerator and given hypothesis zero in the denominator. In this problem, the densities are two dimensional, so the likelihood ratio is formed from the joint densities, each the product of two independent and identically distributed Gaussian probability density functions, as shown in equation A.4.

$$p(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-m)^2-(y-m)^2}{2\sigma^2}} \tag{A.4}$$

From equations A.3 and A.4, the Bayes likelihood ratio test for this configuration is derived:

$$\frac{\frac{1}{2\pi\sigma_1^2}\ \exp(-\frac{(x-m_1)^2-(y-m_1)^2}{2\sigma^2})}{\frac{1}{2\pi\sigma_0^2}\ \exp(-\frac{(x-m_0)^2-(y-m)^2}{2\sigma_0^2})} \underset{H_0}{\overset{H_1}{\gtrless}} 1 \tag{A.5}$$

$$\ln(\frac{\sigma_0^2}{\sigma_1^2}) - \frac{(x-m_1)^2+(y-m_1)^2}{2\sigma_1^2} + \frac{(x-m_0)^2+(y-m_0)^2}{2\sigma_0^2} \underset{H_0}{\overset{H_1}{\gtrless}} 0 \tag{A.6}$$

Equation A.6 in actuality expresses a log-likelihood ratio, which is valid because the natural logarithm is a monotonically increasing function. Additionally, since both densities have equal variance, the expression can be further simplified.

$$-(x-m_1)^2-(y-m_1)^2+(x-m_0)^2+(y-m_0)^2 \underset{H_0}{\overset{H_1}{\gtrless}} 0 \tag{A.7}$$

$$x(m_1-m_0)+y(m_1-m_0)-m_1^2+m_0^2 \underset{H_0}{\overset{H_1}{\gtrless}} 0 \tag{A.8}$$

$$x+y \underset{H_0}{\overset{H_1}{\gtrless}} \frac{m_1^2-m_0^2}{m_1-m_0} \tag{A.9}$$

The Bayes likelihood ratio test in equation A.9 minimizes the probability of error when used as a binary classifier with hypotheses characterized by symmetric 2D normal densities, each formed from two i.i.d. jointly normal distributions with equal variance throughout. For the specific case of two densities with means at 5 and 15, equation A.10 is the explicit likelihood ratio test, producing the boundary seen in figure 1.11.

$$x+y \underset{H_0}{\overset{H_1}{\gtrless}} 20 \tag{A.10}$$

Appendix B

Bayes LRT Generalized for $N-$Dimensional Symmetrical Gaussian Distributions

A useful result presaged by the above derivations is a more general binary hypothesis LRT for N-dimensional symmetric Gaussian distributions. In this generalization we will allow each component of the mean vector to take on unrelated values. Vector components will be expressed with x_i, denoting the ith component of $\mathbf{x}$. Thus, the mean vector components under hypotheses 0 and 1 will be given, respectively, by m_{0i} and m_{1i}.

$$\frac{\prod_{i=0}^{N-1} \frac{1}{\sigma\sqrt{2\pi}} e^{(x_i-m_{0i})^2/2\sigma^2}}{\prod_{i=0}^{N-1} \frac{1}{\sigma\sqrt{2\pi}} e^{(x_i-m_{1i})^2/2\sigma^2}} \underset{H_0}{\overset{H_1}{\gtrless}} 1 \tag{B.1}$$

$$\prod_{i=0}^{N-1} e^{\frac{(x_i-m_{1i})^2-(x_i-m_{0i})^2}{2\sigma^2}} \underset{H_0}{\overset{H_1}{\gtrless}} 1 \tag{B.2}$$

$$\sum_{i=0}^{N-1} [(x_i - m_{1i})^2 - (x_i - m_{0i})^2] \underset{H_0}{\overset{H_1}{\gtrless}} 0 \tag{B.3}$$

$$\sum_{i=0}^{N-1} [2(m_{0i} - m_{1i})x_i + m_{1i}^2 - m_{0i}^2] \underset{H_0}{\overset{H_1}{\gtrless}} 0 \tag{B.4}$$

Therefore, hyperplanes of the form $a_i x_i + b = 0$ are the type of boundary which minimizes the probability of error for Bayes likelihood ratio tests with binary hypotheses characterized by N-dimensional symmetrical normal distributions with equal variance. For more involved

problems, the boundaries become increasingly complex. However, one can approximate any high-dimensional boundary as a series of hyperplanes, reducing a complex problem to many easily solved problems.

Appendix C

Information Function Partial Derivatives (General Case)

Ryan Tomasetti[29, pp.7-10]

Let $h_1, \ldots, h_{m+1}$ be the $m+1$ coefficients of the hyperplane where m is the dimension of the image space. Let $d_1, \ldots, d_n$ be the perpendicular distances from the n image points to the hyperplane.

$$\begin{aligned} \frac{\partial}{\partial h_j} I(\mathcal{P};\mathcal{Y}) &= \sum_{k=1}^{n} \frac{\partial d_k}{\partial h_j} \frac{\partial}{\partial d_k} I(\mathcal{P};\mathcal{Y}) \\ &= \sum_{k=1}^{n} \frac{\partial d_k}{\partial h_j} \left[\frac{\partial}{\partial d_k} H(\mathcal{P}) + \frac{\partial}{\partial d_k} H(\mathcal{Y}) - \frac{\partial}{\partial d_k} H(\mathcal{P},\mathcal{Y}) \right] \end{aligned} \tag{C.1}$$

$$\begin{aligned} \frac{\partial}{\partial d_k} H(\mathcal{P}) &= \frac{\partial}{\partial d_k} \left[\log(n) \right] \\ &= 0 \end{aligned} \tag{C.2}$$

$$
\begin{aligned}
\frac{\partial}{\partial d_k} H(\mathcal{Y}) &= \frac{\partial}{\partial d_k}\left[-\Pr(y_1)\log\left[\Pr(y_1)\right] - \Pr(y_2)\log\left[\Pr(y_2)\right]\right] \\
&= \frac{\partial}{\partial d_k}\left[-\Pr(y_1)\log\left[\Pr(y_1)\right]\right] + \frac{\partial}{\partial d_k}\left[-\Pr(y_2)\log\left[\Pr(y_2)\right]\right] \\
&= \frac{\partial}{\partial d_k}\left[-\left[\sum_{i=1}^{n}\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right]\log\left[\sum_{i=1}^{n}\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right]\right] + \\
&\quad + \frac{\partial}{\partial d_k}\left[-\left[\sum_{i=1}^{n}\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right]\log\left[\sum_{i=1}^{n}\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right]\right] \\
&= -\left(\sum_{i=1}^{n}\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}\log\left(\sum_{i=1}^{n}\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right) - \\
&\quad - \log\left(\sum_{i=1}^{n}\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}\left(\sum_{i=1}^{n}\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right) - \\
&\quad - \left(\sum_{i=1}^{n}\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}\log\left(\sum_{i=1}^{n}\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right) - \\
&\quad - \log\left(\sum_{i=1}^{n}\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}\left(\sum_{i=1}^{n}\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right) - \\
&= \frac{1}{n\ln(2)}\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) + \frac{1}{n}\log\left(\sum_{i=1}^{n}\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) - \\
&\quad - \frac{1}{n\ln(2)}\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) - \frac{1}{n}\log\left(\sum_{i=1}^{n}\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) \\
&= \frac{1}{n}\left[\log\left(\frac{\sum_{i=1}^{n}1-Q\left(\frac{d_i}{\sigma}\right)}{\sum_{i=1}^{n}Q\left(\frac{d_i}{\sigma}\right)}\right)\right]\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) \qquad \text{(C.3)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial d_k} H(\mathcal{P},\mathcal{Y}) &= \frac{\partial}{\partial d_k}\left[-\sum_{i=1}^{n}\left[\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\log\left(\frac{1-Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\right]\right] + \\
&\quad + \frac{\partial}{\partial d_k}\left[-\sum_{i=1}^{n}\left[\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\log\left(\frac{Q\left(\frac{d_i}{\sigma}\right)}{n}\right)\right]\right] \\
&= -\frac{\partial}{\partial d_k}\left[\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{n}\log\left(\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{n}\right)\right] - \\
&\quad - \frac{\partial}{\partial d_k}\left[\frac{Q\left(\frac{d_k}{\sigma}\right)}{n}\log\left(\frac{Q\left(\frac{d_k}{\sigma}\right)}{n}\right)\right] \\
&= -\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{n}\frac{\partial}{\partial d_k}\log\left(\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{n}\right) - \\
&\quad - \log\left(\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}\left(\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{n}\right) - \\
&\quad - \frac{Q\left(\frac{d_k}{\sigma}\right)}{n}\frac{\partial}{\partial d_k}\log\left(\frac{Q\left(\frac{d_k}{\sigma}\right)}{n}\right) - \\
&\quad - \log\left(\frac{Q\left(\frac{d_k}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}\left(\frac{Q\left(\frac{d_k}{\sigma}\right)}{n}\right) \\
&= \frac{1}{n\ln(2)}\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) + \frac{1}{n}\log\left(\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) - \\
&\quad - \frac{1}{n\ln(2)}\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) - \frac{1}{n}\log\left(\frac{Q\left(\frac{d_k}{\sigma}\right)}{n}\right)\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) \\
&= \frac{1}{n}\left[\log\left(\frac{1-Q\left(\frac{d_k}{\sigma}\right)}{Q\left(\frac{d_k}{\sigma}\right)}\right)\right]\frac{\partial}{\partial d_k}Q\left(\frac{d_k}{\sigma}\right) \qquad \text{(C.4)}
\end{aligned}
$$

In all of these functions, the derivative of the Q function is necessary. While an approximation is acceptable for direct evaluation of Q, we should resort to the definition in order to compute the derivative.

$$\begin{aligned}
\frac{\partial}{\partial d_k} Q\left(\frac{d_k}{\sigma}\right) &= \frac{1}{\sqrt{2\pi}} \frac{\partial}{\partial d_k} \int_{\frac{d_k}{\sigma}}^{\infty} e^{-\frac{1}{2}\lambda^2} d\lambda \\
&= -\frac{1}{\sqrt{2\pi}} \int_{\infty}^{\frac{d_k}{\sigma}} e^{-\frac{1}{2}\lambda^2} d\lambda \\
&= -\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{d_k}{\sigma}\right)^2} \qquad \text{(C.5)}
\end{aligned}$$

Finally, we need the derivative of the distance function with respect to the hyperplane coefficients. Let $x_1, \ldots, x_n$ be the n image points in the image space.

$$\begin{aligned}
\frac{\partial d_k}{\partial h_j} &= \frac{\partial}{\partial h_j} \frac{h \cdot x_k + h_{m+1}}{\|h\|} \\
&= \begin{cases} \frac{x_{kj}\|h\| - d_k h_j}{\|h\|^2} & : \quad j \neq m+1 \\ \frac{1}{\|h\|} & : \quad j = m+1 \end{cases} \qquad \text{(C.6)}
\end{aligned}$$

Combining (C.2), (C.3), (C.4), and (C.5) into (C.1), we receive,

$$\begin{aligned}
\frac{\partial}{\partial h_j} I(\mathcal{P};\mathcal{Y}) &= \sum_{k=1}^{n} \frac{\partial d_k}{\partial h_j} \left[\frac{\partial}{\partial d_k} H(\mathcal{P}) + \frac{\partial}{\partial d_k} H(\mathcal{Y}) - \frac{\partial}{\partial d_k} H(\mathcal{P},\mathcal{Y})\right] \\
&= \sum_{k=1}^{n} \frac{\partial d_k}{\partial h_j} \left\{\frac{1}{n}\left[\log\left(\frac{\sum_{i=1}^{n} 1 - Q\left(\frac{d_i}{\sigma}\right)}{\sum_{i=1}^{n} Q\left(\frac{d_i}{\sigma}\right)}\right) - \log\left(\frac{1 - Q\left(\frac{d_k}{\sigma}\right)}{Q\left(\frac{d_k}{\sigma}\right)}\right)\right] \frac{\partial}{\partial d_k} Q\left(\frac{d_i}{\sigma}\right)\right\} \\
&= -\frac{1}{n\sigma\sqrt{2\pi}} \sum_{k=1}^{n} e^{-\frac{1}{2}\left(\frac{d_k}{\sigma}\right)^2} \left[\log\left(\frac{Q\left(\frac{d_k}{\sigma}\right)}{1 - Q\left(\frac{d_k}{\sigma}\right)} \frac{\sum_{i=1}^{n} 1 - Q\left(\frac{d_i}{\sigma}\right)}{\sum_{i=1}^{n} Q\left(\frac{d_i}{\sigma}\right)}\right)\right] \frac{\partial d_k}{\partial h_j} \qquad \text{(C.7)}
\end{aligned}$$

where $\frac{\partial d_k}{\partial h_j}$ is as defined in (C.6).

Bibliography

[1] Abbott, E. A. *Flatland*. Dover, New York 1952.

[2] Armstrong, M. A. *Basic Topology*. Springer-Verlag of New York, Inc., 1983.

[3] Blahut, R. E. Hypothesis testing and information theory. *IEEE Transactions on Information Theory IT-20*, 4 (July 1974), 405–417.

[4] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[5] Cover, T. M., and Thomas, J. A. *Elements of Information Theory*. John Wiley and Sons, Inc., New York, 1991.

[6] Duda, R. O., and Hart, P. E. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[7] Ernest G. Henrichon, J., and Fu, K.-S. A nonparametric partitioning procedure for pattern classification. *IEEE Transactions on Computers C-18*, 7 (July 1969), 614–624.

[8] Feldman, R. M., and Valdez-Flores, C. *Applied Probability and Stochastic Processes*. PWS Publishing Company, Boston, MA, 1996.

[9] Friedman, J. H. A recursive partitioning decision rule for nonparametric classification. *IEEE Transactions on Computers C-26* (April 1977), 404–408.

[10] Fukunaga, K. *Introduction to Statistical Pattern Recognition*, second ed. Academic Press, Inc., 1990.

[11] Gallager, R. G. *Information Theory and Reliable Communication*. John Wiley and Sons, 1968.

[12] Jain, A. K. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[13] Kahn, J., and Kim, J. H. Entropy and sorting. *Journal of Computer and System Sciences*, 51 (1995), 390–399.

[14] Millman, R. S., and Parker, G. D. *Elements of Differential Geometry*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1977.

[15] Minsky, M. *The Society of Mind*. Simon & Schuster, Inc., 1985.

[16] Papoulis, A. *Probability, Random Variables, and Stochastic Processes*, 3rd ed. McGraw-Hill, Inc., 1991.

[17] Payne, H. J., and Meisel, W. S. An algorithm for constructing optimal binary decision trees. *IEEE Transactions on Computers C-26*, 9 (September 1977), 905–916.

[18] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. *Numerical Recipes in C: The Art of Scientific Computing*, first ed. Cambridge University Press, 1988.

[19] Ross, S. *A First Course in Probability*, 4th ed. Macmillan College Publishing Co., 1994.

[20] Rounds, E. M. A combined nonparametric approach to feature selection and binary decision tree design. *Pattern Recognition 12* (January 1980), 313–317.

[21] Rucker, R. *Geometry, Relativity and the Fourth Dimension*. Dover, New York, 1977.

[22] Scott, D. W. *Multivariate Density Estimation: theory, practice, and visualization*. John Wiley and Sons, Inc., 1992.

[23] Sergey Perepelitsa, James Kilian, D. C. Information theoretic decision tree aro final report. Tech. rep., WPI Machine Vision Lab, 1997.

[24] Sethi, I. K., and Sarvarayudu, G. P. R. Hierarchical classifier design using mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-4*, 4 (July 1982), 441–445.

[25] Shannon, C. E., and Weaver, W. *The Mathematical Theory of Communication.* Urbana, University of Illinois Press, 1949.

[26] Shaw, R. *The Dripping Faucet as a Model Chaotic System.* Aerial Press, Inc., Santa Cruz, CA, 1984.

[27] Sterling, B. F&sf science column #9: Robotica '93. *The Magazine of Fantasy and Science Fiction* (October 1993). `http://www.eff.org/pub/Publications/Bruce_Sterling/FSF_columns/fsf.09`.

[28] Stewart, I. *Does God Play Dice? The Mathematics of Chaos.* Blackwell, Cambridge, MA, 1989.

[29] Tomasetti, R. Decision tree automatic target recognition system: Phase two. WPI Machine Vision Lab, 1997.

[30] Trees, H. L. V. *Detection, Estimation, and Modulation Theory.* John Wiley and Sons, Inc., 1968.

[31] Trucco, E., and Verri, A. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, New Jersey, 1998.

[32] Turing, A. Computing machinery and intelligence. *Mind 59*, 236 (1950), 433–460.