

BROWN UNIVERSITY
Department of Computer Science
Master's Project
CS-95-M4

“Reading Signs:
Robot Vision for Optical Character
Recognition and Motion Planning”
by
Bobby Mander

***Reading Signs:
Robot Vision for Optical Character
Recognition and Motion Planning***

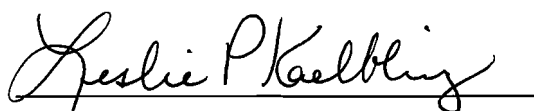
by Bobby Mander

*Department of Computer Science
Brown University
Masters Project
Professor Leslie Pack Kaelbling
April 30, 1995*

***Reading Signs:
Robot Vision for Optical Character
Recognition and Motion Planning***

by Bobby Mander

Submitted in partial fulfillment of the requirements for the Masters of Science
Degree in the Department of Computer Science at Brown University

A handwritten signature in cursive script, reading "Leslie P Kaelbling", written over a horizontal line.

Professor Leslie Pack Kaelbling
Advisor

*Department of Computer Science
Brown University
Masters Project
Professor Leslie Pack Kaelbling
April 30, 1995*

Table Of Contents

0 Abstract.....	page 2
1 Introduction.....	page 2
2 Problem Domain.....	page 3
3 History.....	page 5
4 Theory.....	page 7
5 Implementation.....	page 19
6 Results.....	page 29
7 Conclusion.....	page 31
Acknowledgments.....	page 32
References.....	page 33
Appendix A User's Guide.....	page 36

0 Abstract

Described herein is a project that uses robot vision techniques in order to conduct optical character recognition and motion planning. Specifically, the project employs image processing, image analysis, character classifiers, word analysis, and motion planning. All of these techniques combine into a single application that controls a robot named Ramona. The application's purpose is to have Ramona travel down the Center for Information Technology's hallways and read the signs on the doors.

1 Introduction

The goal of this paper is to describe a robot vision application that is able to navigate a robot down a hallway and have the robot read office signs on the wall.

For a problem that can be stated so simply, the design and implementation is quite the opposite. The design of algorithms to perform tasks we perform routinely and without difficulty is prone to hidden complications. Robot vision is one such branch of artificial intelligence which at first glance appears to be rather straightforward. But when the problem is studied, the hidden complexities of implementing algorithms arise in the form of camera limitations, lack of stereo vision, difficult feature extraction, etc. These complexities can be mostly overcome, however.

The project developed for Ramona, a mobile robot residing at Brown University, dealt with and solved many robot vision problems. The system is composed of five distinct components:

- A front-end image processing subsystem that handles the correction and enhancement of the raw camera image.
- An image analysis subsystem that attempts to find signs or hallways in the picture through relevant feature extraction.
- A classifier subsystem which is actually composed of three separate classifiers that attempts to determine each of the characters found in the image.
- A word analysis subsystem which attempts to match a word found in the image with a dictionary of words in order to eliminate character errors in the classifier subsystem.
- And finally a navigation subsystem which uses the feature extraction of the last image processed in order to determine the next movement action. All of these systems cooperate in order to produce the final robot vision optical character recognition and motion planning system (OCR).

The remainder of this paper will show the actual problem domain handled by this project, detail the history of the techniques used to solve similar problems, describe the theory behind some of the techniques used in this project, discuss the design of the OCR system including several of the crucial design decisions made and difficult problems encountered, submit the preliminary results

of the OCR system, and finally present concluding remarks including possible future growth.

2 Problem Domain

The OCR system described in this paper is meant to solve a very specific problem. The problem is the analysis of images generated by a frame grabber on board a mobile robot and the use of these images to discover and decipher signs and text or else to discover and navigate through hallways on the Center for Information Technology's fourth floor.

The OCR system relies completely on visual images. Visual images are used to drive the image processing. Processed visual images are used to handle the relevant feature extraction. Feature extraction results are then used to drive the various classifiers. And finally feature extraction is used to drive the navigation system and the planning algorithm.

The visual image used by the OCR system is in the form of an 8-bit grey-scale bitmap produced by device drivers on Ramona, our mobile robot. Typical examples of the images gathered by the OCR system are shown below. Figure 1 is an image of a hallway that the OCR system should recognize. Figure 2 is an image of a sign the OCR system should be able to parse and decipher. Figure 3 is an image of part of a wall and part of a hallway. The OCR system should be able to recognize no relevant features in this image.

Figure 1 CIT Hallway Image



Figure 2 CIT Sign Image

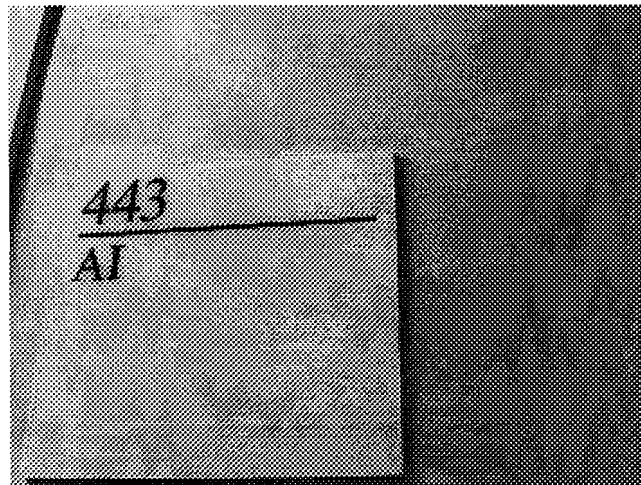


Figure 3 CIT Wall Image



As is clearly seen, the problem domain that the OCR system operates in appears somewhat constrained. Nevertheless, the problem domain is variable enough to make this problem difficult to solve. From these images, it is clear that luminosity, noise, and skew present major problems. What the OCR system provides are robust techniques with the ability to handle all of the images in this simplified, yet still complex problem domain.

3 History

There is a long list of relevant work dealing with optical character recognition, motion planning, and robot vision. Only a small subset is addressed here. For further information the reader is directed to the following publications: *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *Advances in Neural Information Processing Systems*. The related work can be grouped into two distinct areas: optical character feature extraction and recognition methods and motion planning methods. We will deal with these two areas separately.

There are a number of optical character recognition methods under investigation. The demand for optical character recognition is very high with post offices relying more and more on computer technology to sort mail by zipcode, computer fax systems requiring that bitmap-transmitted faxes be received as ASCII text instead of pixel data, and banks using computer signature verifiers to validate personal checks. With this heavy demand for optical character recognition, much work has been done in the field to improve the speed and accuracy of character classifiers.

The most widely used method may be the back-propagation neural network learning approach. Using some arbitrary and complex network topology, many computer scientists have successfully trained these neural networks to discriminate between the different characters of not only English but other languages, such as complex Japanese. Mori and Yokosawa[11,12] have developed such a network system to classify Kanji characters with a relatively simple three layer network. Mighell, Wilkinson, and Goodman[10] have used back-propagation to successfully verify signatures obtained from a digitizing camera. Martin and Pittman[8] also report on a back-propagation neural network that is able to recognize hand printed letters and digits. And finally LeCun and others[3] at AT&T Bell Laboratories have applied the same neural networks to the problem of recognizing hand-written digits. Amazingly all of these approaches use either the standard back-propagation learning algorithm, which is over a decade old, or else a close mutation in order to achieve their comparatively good results. The OCR system attempts to use a back-propagation network in a similar way and hopefully to achieve similar results.

There are newer, more advanced techniques for dealing with optical character recognition. Husain and Kabuka[7] introduce a feature recognition network that partitions the character into its underlying segments and classifies based on these segments. Chen[2] describes a hidden markov model stochastic network approach to word recognition. Chen's approach also uses an on-line dictionary, quite like the OCR system, as we shall see. Wakahara[20] presents an image processing type approach using iterative local affine transformations in order to produce a mask that best matches an input binary character image. Rocha and Pavlidis[15] show a classification system which relies on structural descriptions of characters using prototypes and a template matching facility. The OCR system uses a similar template matching facility, although it is based on bitmap image data and not structural descriptions. Wang and Pavlidis[19] describe a system which uses greyscale bitmap data in order to construct a character's topological surface and from there

extract relevant features that aid in classifying the character. And finally Ohya, Shio, and Akamatsu[13] elaborate on an algorithm which isolates characters in scene images using an image segmentation method based upon adaptive thresholding. They then use these isolated characters in a template matching fashion in order to deduce character classifications.

Probably the most representative paper in the field of optical feedback motion planning is the ALVINN project. Pomerleau[14] has developed a back-propagation neural network that uses road images in order to decide how a vehicle can best follow the road. The system is very sensitive to training data, as can be expected, but seems to perform well in practice under normal conditions. Overall, Pomerleau must conclude, however, that much more work needs to be done in the field of autonomous navigation using robot vision. The OCR system, and systems like it, will continue to be implemented as the field becomes more developed. Another vision based control system known as MURPHY has been developed by Mel[9]. MURPHY is a kinematic controller and path planner that learns using a neural network under the sigma-pi learning scheme. The neural network that drives MURPHY is large and complex, as is the workings of a robotic controller, yet initial results show that the network is able to successfully learn how to handle the robotic arm.

Clearly there have been many approaches to the problem of optical character recognition and vision based motion planning. Those projects mentioned above are only representative of the growing field.

4 Theory

There is a rich theory behind many of the techniques employed in this project. The techniques can be separated along the same lines as the components in the OCR system. Namely, there are techniques describing front-end image processing, image analysis and feature extraction, and pattern recognition and classification. Appropriately, these different techniques will be described separately next.

4.1 Image Processing

Image processing is a mature field with a wealth of techniques to choose from. From thresholding, to eroding, to dilating, to smoothing, to warping, one can combine these methods and change their parameters to produce resulting images far different from the original. The real difficulty in image processing lies in choosing the techniques that are best for a given problem domain. And the real limitation of image processing is the fact that the techniques are only applicable to a subset of all the images you may encounter, i.e. image processing is tailored to the problem you want to solve.

The image processing techniques used in this project can be separated into those techniques that correct imaging defects, those techniques that enhance the image, those techniques that reduce the image information, and those techniques that operate on binary images. Image correction removes problems caused by the image acquisition hardware and software. Image enhancement removes irrelevant information and magnifies meaningful information. Image reduction reduces the excess information in an image. And binary image processing further enhances a possibly noisy or error filled image. The techniques for each of these areas is presented next. The image processing techniques described apply to greyscale and binary images only. Color image processing is far more complex and is not covered since the OCR system has only greyscale images available to it.

4.1.1 Image Correction

The inherent limitations of image acquisition devices require the usage of image correction techniques. These limitations include camera response curves, blooming, pincushion and barrel distortion, corner distortion, analog errors, and digitization approximations. All of these limitations introduce noise and errors into an image. These errors may be corrected using image correction techniques.

The first defect an image usually has is random noise scattered throughout the image. The many ways to correct for random noise all involve neighborhood operations. Neighborhood averaging takes the average pixel value for small regions of the image and replaces every pixel value in that region with the average. While this reduces the random noise, it also reduces the resolution of the image as all pixels within a region are identical. A better way to remove random noise is by

smoothing through the use of a kernel. A kernel is a matrix of values that are used as coefficients to multiply the pixel values in an image. The kernel is usually a square matrix. This square matrix is successively applied to the image with the target pixel at the center. The pixel values are multiplied by the kernel coefficients and the sum of these terms is divided by the sum of the kernel coefficients. This resulting value replaces the pixel value of the center pixel. In this way, each pixel is replaced by the kernel product average. Since this average tends to smooth away the random noise that may be present in an image, this process is called smoothing. But smoothing also blurs sharp edges in an image, a side effect which may or may not be beneficial. A sample 3x3 smoothing kernel is shown below.

Sample Smoothing Kernel

1	2	1
2	4	2
1	2	1

This kernel takes a weighted average that favors the central pixel and those pixels directly above and below the central pixel. Other more complex and involved kernels such as Gaussian kernels have been investigated but as the size of the kernel increases, so does the computing time.

A major correction that often must be performed on an image is rotation. If an image is skewed in any way, the features within it and the data that is meant to be analyzed will be corrupted. In order to correct for this, a rotation must be performed. Rotations can be performed in two dimensions with a straightforward linear transformation or in three dimensions using some sort of perspective rotation. Linear rotations are the simplest and require the least amount of information to perform. Unfortunately, most real world images are skewed in three dimensions and therefore require more complex rotation methods to be performed. In the OCR system, a skewed image is corrected using a two dimensional linear rotation approximation that results in a better final image, although not a perfect one.

But before the rotation can be performed, the proper angle of rotation must be determined from the skewed image. This can be difficult in itself. This determination is based solely on some form of feature extraction that is application dependent. Assuming that the proper angle has been determined, the rotation is accomplished using a trigonometric transformation to map the original image's pixels to the rotated image's pixels using simple sine and cosine functions of the following form:

$$\hat{x} = x + y \sin \phi$$

$$\hat{y} = y - x \cos \phi$$

Figure 4 shows a skewed image that has been rotated linearly to form Figure 5. Notice how the line separating the top and bottom of the sign is now horizontal and the characters are more

upright. But also notice that the bottom of the sign is slanted somewhat downward due to errors caused by using the two dimensional approximation.

Figure 4 Skewed Image

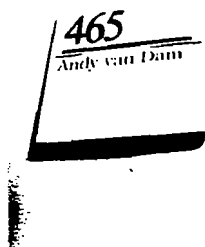
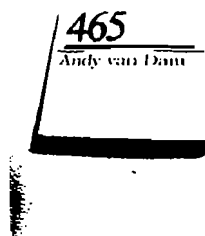


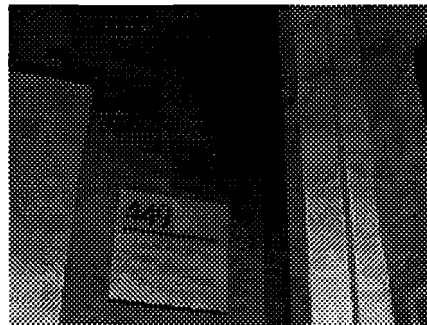
Figure 5 Rotationally "Corrected" Image



4.1.2 Image Enhancement

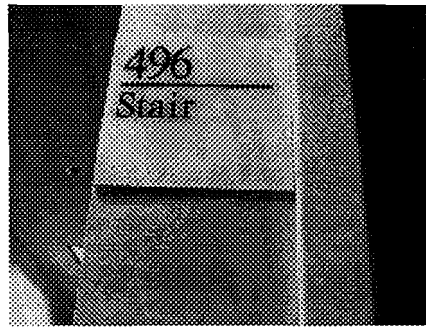
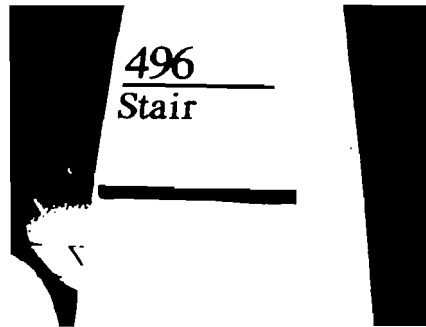
After image correction has taken place, image enhancement can proceed. Again, there are many techniques to choose from. Only those employed by the OCR system will be described here.

In many cases the original image's pixel values will not cover the entire range of valid pixel values. This would occur in images which are very dark or very light. In these cases the image must be enhanced so that the pixel values are made to cover the entire range. This will help in later feature extraction because it increases the contrast of differing pixels. This process is called contrast expansion or histogram equalization. It simply consists of computing the minimum and maximum values of the original image and using a linear scaling function to map the pixels in this range to the maximum range of greyscales, usually from 0 to 255 for 8-bit images. The scaling function may be nonlinear, resulting in different contrast expansions depending on some properties of a specialized problem domain. This technique works well in practice as images that originally were seen as almost completely dark were transformed into distinct, easily recognizable images. Figure 6 shows a sample noisy, dark image that has been corrected and enhanced by first smoothing and then contrast expanding to form Figure 7.

Figure 6 Image Before Correction and Enhancement**Figure 7 Image After Correction and Enhancement**

4.1.3 Image Reduction

In order to reduce the information in the image while at the same time enhancing it, thresholding is often done. Thresholding involves transforming a greyscale image into a binary one by simply clipping all pixel values below a certain level to zero, while clipping all other pixel values to 1. This enhances the contrast between very different pixels while eliminating the contrast between similar pixels. Thresholding is also very important in reducing the information in the image to a manageable size for later processing. The main difficulty in thresholding is determining the proper middle pixel value with which to clip. This value should depend on the frequency of the pixel values above and below the clipping value, otherwise the thresholding will produce much more of white than black, for example. Therefore some sort of median finding or averaging of the pixel values is performed in order to deduce the threshold clipping value. Figure 8 shows a complex image that has been reduced to a binary image in Figure 9.

Figure 8 Image Before Thresholding**Figure 9 Image After Thresholding**

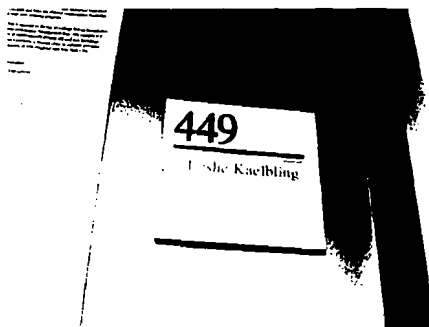
Another reduction method is image scaling. Scaling is a straightforward mapping from a large pixel space to a smaller pixel space. This is used, for instance, to map irregular or variable-sized features into well-defined, constant-sized, smaller features. Scaling is done by simply averaging local pixel areas for each pixel in the target area. Obviously, scaling loses a significant amount of information.

4.1.4 Binary Image Processing

Once thresholding has converted the greyscale image to a binary one, binary image processing techniques can take over. The binary image and its manipulation has been thoroughly examined by the image processing community. This has resulted in very effective and efficient algorithms. The most important algorithms for the purpose of the OCR system are those for erosion and dilation.

Erosion and dilation are neighborhood operations that effectively remove or add pixels to an image. The classical erosion method is to remove a black binary pixel if it is touching a white binary pixel. This test, done for all pixels in the image, results in a new image with some of the pixels removed. The classical dilation is just the opposite, a black binary pixel is added if it is touching another black pixel. The resulting image is one which has several pixels added to the original image. The operation of these neighborhood techniques is simple yet very powerful. Figure 10 shows a binary image for which repeated dilations have been applied to produce Figure

11.

Figure 12 Image Before Dilations**Figure 13 Image After Dilations**

These two techniques can be combined by computing an erosion followed by a dilation. This effectively opens spaces between features. Features are defined to be distinct entities in a binary image. The determination of the geometry of features will be presented in the next section. If the dilation is computed first followed by the erosion, the result is the effective closing up of spaces between broken parts of the same feature. Because of these effects erosion followed by dilation is commonly called an opening, and dilation followed by erosion is commonly called a closing. For the OCR system, a closing will be particularly important to ensure that single characters found in the image are completely connected. Repeated applications of openings or closings increase the impact of the effect. In fact, in most applications an arbitrary number of these is performed in order to obtain the desired opening or closing level.

A more general form of the erosion and dilation operators count neighbors. For example, a general form of erosion would count the number of white pixels surrounding a black pixel. Only if a defined count threshold is exceeded will the pixel be replaced by a white pixel. This causes the erosion to be more selective if the threshold is high, and less selective if it is low. This effectively changes the amount of erosion that occurs. Similarly, a general dilation would count the number of black pixels surrounding a white pixel, and if this count exceeds some other threshold, the white pixel would be switched to black.

The difficulty in using general erosion and dilation is, of course, the setting of the thresholds. This parameter tuning is a recurrent theme in image processing for which there are no mathematical solutions, only heuristic ones. Nevertheless, general erosion and dilation will be shown useful, after proper parameter tuning, in the OCR system in its attempt to isolate vertical line features.

4.2 Image Analysis

Image analysis is concerned with extracting appropriate information from images. Image analysis is strongly connected with the problem domain, hence image analysis is very difficult to generalize. One must determine the information that should be present in the image and how to isolate it. This isolation of information is also known as feature extraction. Feature extraction can take many forms when dealing with greyscale or binary images. For greyscale images, examples of methods that attempt to extract features include contour operations, gradient operators, and split-and-merge. For binary images, the feature extraction methods are a little less complex.

In binary images, the features are usually continuous areas of black pixels on a white background or white pixels on a black background. The purpose of feature extraction is to identify and correctly represent all of the features relevant to the given problem. Representations of general regions can be accomplished with a variety of different data structures and encoding methods such as run-length encoding, boundary polygonal encoding, or chain encoding. These data structures are general enough to handle most problem domains. Along with these data structures, a generic binary bitmap feature extracting algorithm is available. This algorithm processes pixels and places touching pixels into the same group, while placing other touching pixels into separate groups.

The only problem with using the method is the computational expense of using general methods to solve specific problems. As we shall see later, the well defined feature extraction processes that occur in the OCR system call for far simpler data structures and more efficient image analysis. In particular, when looking for an office sign, the features we are looking for are isolated horizontal lines, text words, and characters within those words. And in the case when we are looking for a hallway, we are looking specifically for slanted vertical lines. These are the only features of interest in the OCR system. Because we can describe exactly what we are looking for in the OCR system's feature extraction, we can improve performance by an order of magnitude, at least.

There are feature extraction methods that deal with greyscale images as well. One that is particularly important for the OCR system is known as the Laplacian. The Laplacian is concerned with enhancing any edges present in an image. The Laplacian is so named for its mathematical equivalent second derivative functionality. In image processing we are determining the second derivative of brightness in an image. The Laplacian is easily implemented in image processing using a

kernel operator with the following kernel:

Sample Laplacian Kernel

-1	-1	-1
-1	8	-1
-1	-1	-1

This effectively subtracts the brightness of neighborhood pixels from the central pixel and replaces the central pixel with that value mapped to the proper 0-255 range for 8-bit images. This mapping is done simply by adding the middle 128 grey value and clipping. The result of the Laplacian is an image in which middle grey areas indicate those areas devoid of edges while white and black areas indicate those areas with edges. The resulting image can be converted to binary image with edges demarcated by applying a modified threshold operator in which values within a certain threshold of the middle grey pixel value are replaced by white and all others replaced by black. The trick is, of course, to find the appropriate threshold to use. Many techniques can be used, but most rely on examining the histogram of the Laplacian image. By choosing an appropriate level of black pixel acceptance, the histogram can be used to determine the threshold. Other operators, besides thresholding, can be applied to the Laplacian image in order to sharpen edges, if necessary. Figure 14 shows an example of an image for which the Laplacian operator has been applied to produce Figure 15 and furthermore the special thresholding has been applied to produce Figure 16.

Figure 14 Image Before Laplacian



Figure 15 Image After Laplacian**Figure 16 Image After Laplacian Threshold**

4.3 Pattern Recognition and Classification

Pattern recognition and classification is a growing field with a rich set of literature devoted to it. The *IEEE Transactions on Pattern Analysis and Machine Intelligence* journal is a testament to that. There is a cornucopia of methods to use to tackle the task of pattern recognition, some specific to certain problem domains such as speech recognition, while others are more general and can be adapted to most any problem domain from visual text recognition to weather forecasting. The cutting edge techniques in this field are often in the preliminary stages of development. But their stable predecessors have entrenched themselves and shown that they are able to successfully solve many problems. The OCR system relies on the somewhat 'classical' methods that are available in the pattern recognition field.

The OCR system uses three types of classifier systems. A back-propagation neural network, a competitive neural network, and a template matching system. Each of these separate classifiers will be discussed next.

4.3.1 Back-Propagation Neural Network Classifiers

Back-propagation is a supervised learning method applied to feed-forward neural networks. Back-propagation operates by passing error terms from the output of the neural network back through the hidden levels of the network. It relies on the concept of gradient descent in order to modify node weights and thus produce learning in the network.

Back-propagation first passes a given input through a multiple-level feed-forward neural network in the normal way. It then computes an error amount on the output side based on the actual desired response. This error is used to modify the weights of the output nodes using a gradient descent update rule. This error is then propagated backward through the network to the node layer preceding the output layer. An error value is then computed for the hidden layer nodes, the hidden layer nodes' weights are updated, and the error is propagated backward again. This process continues all the way back to the first layer in the neural network at which point a new input can be given and the process repeated. The complete learning algorithm is omitted for space.

Several additions can be made to the learning algorithm in order to learn more quickly. Momentum is often used to quicken the rather slow gradient descent weight update rule of the back-propagation method. The idea is to update a node's weights based on the gradient descent update rule with an additional momentum term which adds a value proportional to the last change the node's weights experienced. This effectively averages the changes in weights to prevent frequent troublesome oscillations in the node weights. The momentum term's factor is known as the momentum parameter and may be a little tricky to set properly.

Another addition is the variation of learning rate. The learning rate determines the rate of change of a node's weights. As learning progresses and more and more samples are presented it makes sense to reduce the learning rate so that the network becomes more stable. This decayed learning will eventually end as the learning rate goes to zero. Other ways of varying the learning rate, such as increasing the learning rate if the overall network cost function has decreased in the last iteration, while decreasing the learning rate if it has not, have been proposed and seem to work well in practice.

The main difficulty in using back-propagation neural networks is determining the network layout for a given problem domain. The network layout is the number of layers in the neural network and the number of nodes at each level. For a given network configuration some problem domains may not be learnable and therefore not solvable. For other network configurations the learning process may be terribly slow. In fact, the whole problem of determining if a given neural network can solve a given problem domain is NP-hard. The network configuration is not the only problem. There is the problem of setting the tunable parameters such as learning rate. Of course, there is no mathematical way to do this, so heuristics and tests are heavily relied upon. And finally there is the problem of presenting representative data in random fashion. Supervised learning can be difficult and the results can be poor if such real data is not presented to the network.

4.3.2 Competitive Neural Network Classifiers

Competitive neural network classifiers rely on the use of a winner-take-all neural network. The network topology is simple, with a single layer of outputs units, one for each possible classifica-

tion of inputs. Each of the output units is fully connected to the inputs. The learning method is unsupervised and efficient in its quest to correctly classify inputs.

When an input is presented to the system, the node that computes the largest output value is considered the winner, while the rest are all losers for this iteration. The winning node's weights are then updated in order to move it directly towards the current input unit. This increases the chances that the winning node will be the winner the next time the same input is presented to the system. This is often called the standard competitive learning rule. In the standard competitive learning methodology only the winner is updated, while the losers remain stagnant for the given input. The iteration is repeated on many training inputs until the network has been deemed to have learned the classifications.

Simple competitive learning does not work well in practice. Therefore, there are several standard additions that can be used to augment the standard learning rule. The first is called leaky learning and it involves not only updating the winning node's weights but also the losing node's weights according to the standard competitive learning rule. There would be a different learning rate factor used for the winning node and a much smaller learning rate factor for losing nodes to ensure the winner learns more than the loser. This effectively moves all the output units in the direction of the current input. This helps to bring all the output units into the problem domain. If an output unit's weights are selected randomly to begin with, it may be that some node is never given a chance to win for a certain problem domain. Leaky learning ensures that all nodes get a chance to win.

The other major addition to competitive learning is often called a conscience. The idea is to make winners feel guilty about frequent wins and to give losers a better chance to win. This is done by subtracting a threshold or bias term from the output of a given node. The bias is updated during each learning step. The bias is incremented by some amount if a node has won and decremented by some amount if a node has lost. This will ensure that not only do losers win more often and winners win less often, but on average each of the output units will win the same number of times, if the inputs are chosen according to a uniform random distribution. This is a very powerful and effective technique because of this property.

Again there are difficulties involved with the use of competitive networks which are similar to the problems in using back-propagation networks. For a given problem domain, it may not be known how many classifications of the inputs there are. This can make determining the number of output nodes in the network difficult. Also the choice of various parameters such as learning rate, winning node threshold, losing node threshold, and leaky learning rate require extensive experimentation in order to arrive at effective values.

4.3.3 Template Matching Classifiers

The theory behind distance computing classifiers is so simple, it barely deserves attention. But in order to be complete, it is briefly described here. These classifiers use a set of templates stored in a database. For a given input, a distance metric is computed to determine the distance from the input to each of the set of templates. The template which gives the lowest distance metric is considered the classification.

Amazingly simple, yet extremely powerful, template matching classifiers join a number of other scientific and mathematical techniques which defy our ability to complicate everything unnecessarily. Yet there are problems even with such a simple technique. Namely for some problem domains, such a metric may be difficult or impossible to compute. Consider, for example, complex VLSI chips. There are millions of components to consider when computing the distance metric and the distance metric is not immediately obvious. Couple this difficulty with the fact that a distance metric must be computed for every template in the database (which could be a large set), this computation becomes unwieldy and much too expensive. Therefore template matching classifiers are only applicable to a subset of problem domains for which distance metrics are easy to compute and the number of templates is small. Luckily, this is the case for character recognition in the OCR system.

5 Implementation

The OCR system was implemented in the C++ programming language. Initial design, development, and testing took place on UNIX SUN SPARC 10 stations. The system was then ported to operate on Ramona, a B21 robot from RWI. Ramona is driven by a 90 Megahertz 486DX4 computer system running the Linux operating system. In the end, the OCR system was made to be cross-platform compatible.

Ramona, standing approximately three feet tall, is equipped with two forward panning and tilting cameras, various sensors including sonars, motor drivers that allow it to rotate and translate, a speech output mechanism, three onboard computer systems, and, most importantly, a vision system involving a digitizing frame grabber. The vision system encodes 8-bit greyscale images with pixel dimensions of 640 horizontal and 480 vertical. Several other smaller bitmap dimensions are possible, but these smaller bitmaps were not considered appropriate for the OCR system which requires the highest pixel resolution possible. These images are what drive the OCR system.

Following the discussion of the implementation details, two sequences of images are presented to show the various processing steps involved in the image processing and front-end feature extraction. Figures 18-20 show the sequence of images created when a sign has been found. Figures 21 and 22 show the sequence of images created when a hallway has been found.

With the implementation setting described, we can move on to the software design.

5.1 General Design

The implementation is function oriented, rather than object oriented, in order to easily be used as a library of functions for later developers. The OCR system is summarized in Figure 17, which shows a high-level view of the OCR system's control and data flow.

The functions involved in the OCR system can be separated into several different areas. The first is the hardware interface segment, which controls the image acquisition, speech synthesis, camera movement, and navigation control hardware. The second is the image processing segment which performs the front-end processing to convert the raw image we acquire into a usable fine-tuned bitmap. The third is the sign-finding feature-extraction segment, which looks for signs in the image processed image. The fourth is the word-location feature-extraction segment, which isolates the words and characters in a found sign. The fifth is the classification segment, which attempts to classify the characters found in a sign. The classification segment is the most important and complex of all the segments. The seventh and final segment is the one which attempts to find hallways in the image. Each of these segments will be generally discussed in the next sections.

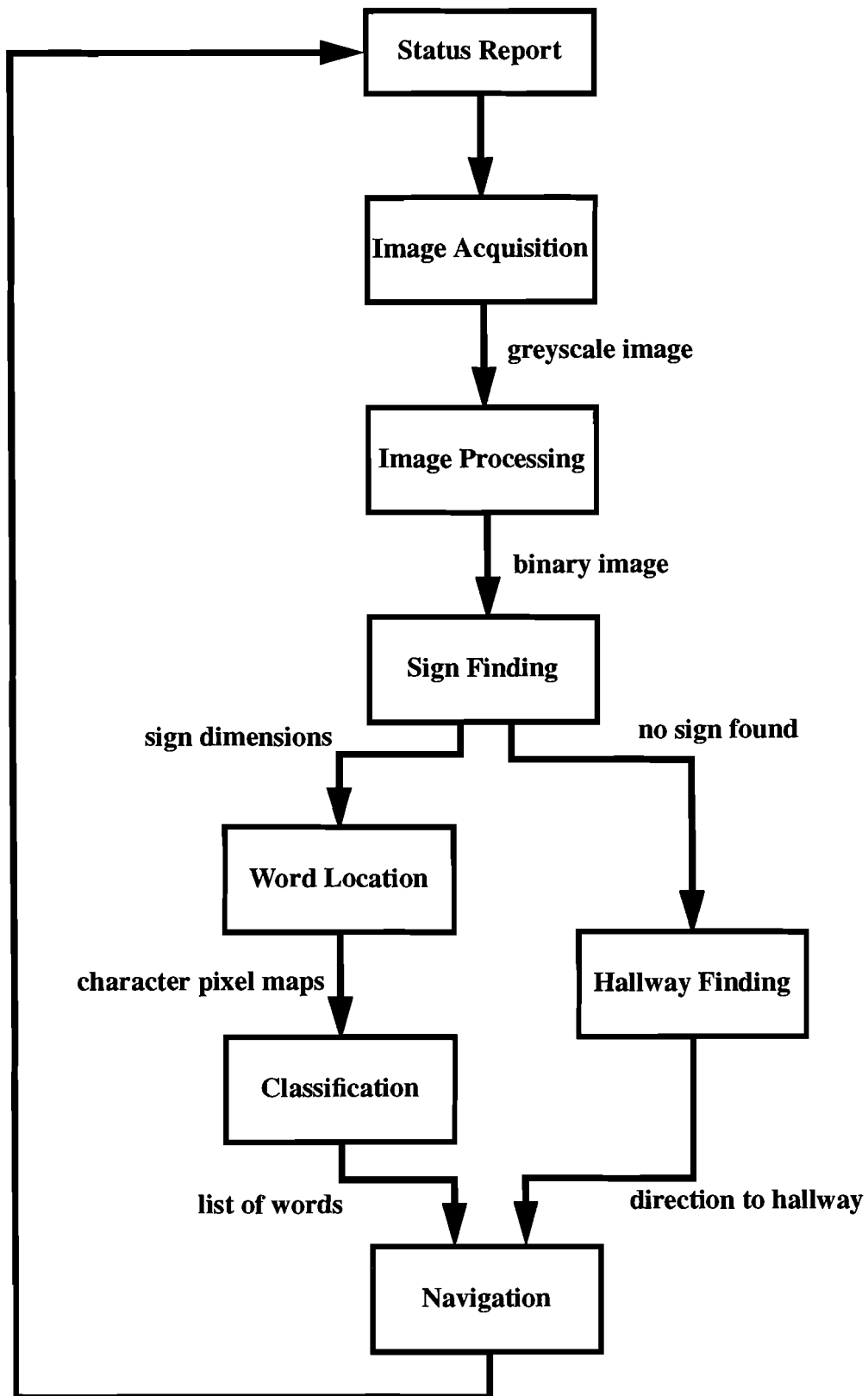


Figure 17 OCR System Control and Data Flow

5.2 Hardware Interface Segment

The OCR system uses device drivers provided with Ramona. These drivers include a vision driver for acquiring PGM format image files, a speech driver for producing speech from text input, a camera movement driver that allows panning and tilting of the two forward cameras, and a navigation system driver which allows primitive movement commands to be processed. Each of these interfaces require a server to be running on the appropriate computers on Ramona. The servers and clients use a message passing system to process and send requests based on the TCX protocol.

The operating scheme used by the OCR system is simple. The first thing done is an initialization tilt of the camera upward by 45 degrees so that signs on the wall will become visible to the short robot. The camera stays at this tilt throughout execution. Then images are acquired and processed in a loop. Based upon the classification of each image, navigation proceeds according to simple rules. The first rule is that if we cannot classify the last image, we rotate an arbitrary distance clockwise in order to look for a sign or hallway somewhere else. The second rule is that if we have found a hallway, translate down the hallway a fixed amount, then rotate counterclockwise 90 degrees in order to view the wall. The third and final rule says that if a sign has been found, rotate clockwise 90 degrees and begin the search for the hallway again. As is very clear, the navigation scheme is tied to the vision system. Without proper vision, movement becomes impossible.

There are several limiting factors with the hardware for the OCR system. The first is the camera interface. The camera does not have automatic gain control or focusing control, therefore the OCR system relies on a fixed focusing depth and aperture for all pictures. This plays serious havoc with the lighting and noise of most images acquired. The image processing segment must correct for this. The second limiting factor is the speed of the image acquisition system. What the OCR system requires is real-time visual feedback in order to be able to effectively perform serious motion planning. Unfortunately Ramona does not provide for this.

5.3 Image Processing Segment

The image processing segment is a computationally intensive series of operations that are performed to every image acquired from Ramona's hardware. The image processing segment also includes other operations that are performed only in certain situations, depending on the features that have been extracted from the image. The operations in the image processing segment are based upon the standard image processing techniques described in the theory section. The efficiency of the image processing techniques has been steadily improved during the implementation to take advantage of better bitmap data structures and algorithm design.

The invariant sequence of image processing steps applied to each acquired image is as follows:

- A smoothing operator is applied based upon a neighborhood of 8 pixels and the smoothing kernel shown in the theory section.
- The image contrast is expanded in order to encompass the entire 0-255 range. This is particularly effective because most of the acquired images are relatively dark
- A thresholding is performed based upon an adjusted median pixel value. Using the adjusted median is very important because the thresholding causes a great loss of information. Optimally, we want to lose only irrelevant information. The median makes our thresholded image close to optimal, in this regard.
- A series of closings is performed by invoking a dilation followed by an erosion. This is used to close possibly fragmented features such as lines and characters. Unfortunately, this also causes features that did not touch before to sometimes join.

Other image processing operations that are performed on demand include the following:

- Image rotations are performed using a trigonometric mapping based upon some angle of rotation either clockwise or counterclockwise. This rotation is crucial for sign finding, as we shall see.
- Special image erosions and dilations can be performed based upon specific neighborhood information such as number of black or white pixels. This feature is used specifically for hallway finding.
- A Laplacian operator can be applied to an image using the Laplacian kernel presented in the theory section. This operator is applied to a greyscale image in order to find edges. This operator is used for hallway finding.
- The resulting image produced by the Laplacian operator can be thresholded using a special threshold function. This function thresholds by setting pixel values close to the middle grey pixel value to white, while setting those farther away to black. This demarcates lines with black and background in white. This thresholding is used in conjunction with the Laplacian in order to deduce the presence or absence of hallways.

The image processing segment is organized as a library of generic calls to perform the various functions. These calls have been somewhat optimized based on the manipulation of an optimal representation of a bitmap structure, which has evolved over time.

5.4 Sign Finding Segment

Specific feature extraction is called for in order to find a sign in an image. The OCR system uses specific features of the sign in order to optimize and ease the feature extraction operations necessary in order to find signs. But there are several complications that arise and must be dealt with.

The biggest and most frequent complication is the presence of skewed images acquired and pro-

cessed by the system. Skewed images cause signs to appear crooked. This not only makes the signs difficult to detect, but also the characters within the sign nearly impossible to classify. What is performed to alleviate for skewed images is a feature search. This feature search looks for thin slanted lines surrounded by white space in the image. These thin slanted lines are candidates for the black line that separates the top and bottom of each sign. Using a slanted-line feature-extraction image-scan and a slanted-line validation technique, the best candidate slanted line in an image is found. From the best candidate slanted line a slope is calculated and a correcting angle of rotation determined. This angle is used to rotate the image clockwise or counterclockwise.

The resulting image is then subjected to a second, quicker feature extraction pass. This second pass now looks for horizontal lines of minimum and maximum specified width. From the candidate lines found (if any) the best is taken as the basis for a sign. If none are found, then there is no sign in the image. Based on the geometry of the central sign line, the complete dimensions of the sign can be determined based on the characteristic white space that appears above and below all signs. With the many portions of the feature extraction complete, the complete specifications and dimensions of the sign are saved and passed on to the next segment.

5.5 Word and Character Finding Segment

If a sign was found in an acquired image, the geometry of the sign is used to fuel the feature extraction in the word and character finding segment. This segment relies on the known geometry of the sign. This assumed geometry is that words appear above the sign and words appear below the sign. Therefore a feature extraction search is made above and below to find lines of text which can be read. This feature extraction basically finds the dimensions of each line of text in the sign using horizontal and vertical passes through the thresholded image's bitmap. These horizontal and vertical passes have associated with them parameters such as the number of vertical white lines that separate each line and the minimum line height. These parameters have been tested and trained using many example sign images. The result of this is a list of rectangles enclosing each of the lines in the sign.

From this list of line-enclosing rectangles, the word-finding segment continues its search for even smaller features. These smaller features are the words in each text line of the sign. Using similar techniques of horizontal and vertical sweeping through the thresholded image's bitmap, we can determine the rectangles enclosing each word in a text line. But, of course, the parameters associated with the horizontal and vertical sweeping used for finding words are different than those for finding lines.

And from here we must continue still further down to the character level. Each of the characters within a word must be completely isolated so that its pixels can be passed on to the classifiers. And once again, we must use the techniques of horizontal and vertical sweeping in order to determine each character's dimensions. It must be stressed that although the same style technique is

used for each of the line, word, and character feature extraction steps, the parameters are of primary importance in making the techniques work for all images. It is the determination of these parameters that is the most difficult part of implementing the sweeping algorithm. And to complicate matters, other considerations such as random noise, discontinuous features, and edge effects must be taken into account. A multitude of problems was encountered when trying to implement the sweeping algorithm that would be able to isolate all characters, but, in the end, the sweeping algorithm was able to perform reasonably well.

Once the rectangles enclosing each character have been determined, a scaling must occur. This scaling must translate an arbitrarily sized character into a character that will fit into a 7 pixels vertical and 5 pixels horizontal pixel map. This is necessary so that the classifiers are fast in classification as well as training. This scaling is rather straightforward and exactly as described in the theory section.

5.6 Classification Segment

At this point we have isolated each word, isolated each character within each word, and identified the pixels corresponding to each character. These character pixel maps are presented to the three classifiers one by one. Each of the classifiers, back-propagation, competitive, and template matching, return the ASCII character ID of the computed character classification.

But the classifiers also return one more piece of useful information, a reliability estimate. This estimate is a percentage estimate of the correctness of the deduced classification produced by each classifier. In other words, this reliability is how sure each classifier is of its classification. This reliability is used in determining which classifier output to use in the end. To combine the classifications of the three classifiers, a majority vote is taken first. If there are two classifiers who have produced the same ASCII code classification, that classification is taken. If there is a split vote among the classifiers, the classification with the highest reliability estimate is chosen.

Once each of the characters in all of the words has been processed, we can move on to the word classification phase. Word classification takes place using a dictionary lookup feature. Given a database of words stored in a dictionary, we attempt to find the closest match to the word proposed by the earlier classifiers. If we find a close enough match, we replace the classified word with the dictionary word.

The above discussion explains how the classifiers are used. What is also needed is a summary of the way the classifiers operate and are trained. This is presented for each below.

5.6.1 Back-Propagation Classifier

The back-propagation classifier was trained using synthetic pixel maps that are continually modified by the addition of a small amount of random noise. The synthetic pixel maps were created at

first using idealized character pixel data, but later revised to better conform to the actual test data found in various signs throughout the CIT. Using the standard back-propagation learning method, millions and millions of synthetic characters were presented to the neural network. Various learning parameters were tried until the seeming optimal ones were stumbled across and used. Learning was considered complete when the network was able to classify over 95% of the synthetic letters correctly. Once this occurred, the network characteristics were written to file so that training would not have to take place again.

The original network topology consisted of 21 nodes spread across three levels. There are six outputs from the neural network. These outputs correspond to individual bits in a character code. When these bits are combined into a single decimal value, this code is then mapped to a standard ASCII character code. The 35 inputs to the neural network correspond to each pixel in the character pixel map. The topology of the network is somewhat arbitrary. Knowing that a hidden level is required to solve difficult problems such as character recognition, it was determined that the network must be multi-level. What was chosen was a relatively simple topology that would not take too long to learn. From various papers and experimental results, it has been shown that larger networks do not entirely mean better networks, so the network chosen was mid-sized.

Several other network topologies were tested and performed slightly better than the original. One of the networks, with 50 nodes spread across one level, was able to function the best in the presence of noise in the images. Since most images did contain noise, this was the most practical topology. Because of this practicality, this topology was used.

The reliability of the back-propagation network classifier is based upon how far the actual floating point output is from the desired integer output. This value is scaled to give a percentage reliability figure. Several other reliability estimates were considered until this relatively simple one was chosen.

5.6.2 Competitive Classifier

The competitive classifier was trained in the same way as the back-propagation classifier, only the learning rule used was a modified competitive learning rule. The modifications included a small learning rate applied to losing nodes (leaky learning) and a threshold maintained for each node (conscience). Many, many different learning parameters were tried before the optimal ones were found and used. Again, millions and millions of synthetic characters were fed into the system until the classification results were satisfactory. The resulting characteristics of the fully trained competitive network were saved to file to avoid the repetitive and time consuming learning phase.

The actual network topology consists of 50 nodes arranged in one output level. Each of the nodes correspond to one ASCII character. In addition to the neural network, we must maintain a history of which training letters were classified to which node. In this way, based upon the node which wins for a given test character, we can determine which character it is by finding in the database

that letter for which this node won the most.

The reliability of the classification is based upon the number of times the node in question has won for the classified letter and the total number of times the node has won. This was the most straightforward method of computing the reliability. Other estimating methods were tried, but were not successful.

5.6.3 Template Matching Classifier

The template matching classifier requires no training and is very memory and time efficient. It simply maintains a list of synthetic character pixel maps in memory. These are the templates. When a character pixel map arrives to be classified, a distance between the character and each of the templates is computed. This distance is used to compute the reliability of the classification by dividing it by the maximum possible distance between character and template. Despite its simplicity, the template matching classifier is a superior classification performer.

5.6.4 Dictionary Lookup

Once the characters in all of the words have been classified, one last classification method is invoked. This classification method attempts to find a word in a dictionary that is closest to the word produced by the concatenation of the characters from our classifiers. In other words, we have already performed character classification, the dictionary lookup feature attempts to perform word classification.

The dictionary lookup feature is exactly that. It maintains a list of dictionary words. It performs a template matching between the classifier determined word and each of the template words in the dictionary. It does this by computing a string distance for each word pair. This string distance function is heuristic in nature, but seems to work well. If a word is found in the dictionary that is close enough to the classified word, the classified word is replaced by the word in the dictionary. The distance threshold between words in order to perform the replacement is dependent on the word length, but is small enough so that wildly different words are not issued as replacements for classified words.

An additional wrinkle was added to the dictionary lookup so that words comparing equally would not compute distance equally. This was done to favor high frequency words like 'the' and 'to'. Therefore a classified word such as 'tle' would be replaced by 'the' instead of arbitrarily similar words like 'tee'. The frequency of English words was obtained using code written for another project, a trigram statistical language processor which kept a database of word probabilities, among other things, within it. These probabilities were used to sort the dictionary with highest frequency words appearing first and lower frequency words last. The dictionary lookup then takes the first word in the dictionary having the minimum distance to the classifier word, thus considering higher frequency words first.

5.7 Hallway Finding Segment

If there is no sign in the current image, the OCR system resorts to searching for hallways. Finding a hallway requires finding edges in images. This is best served using Laplacian operators. Therefore, in order to find a hall, the Laplacian operator is applied to the original image-processed grey-scale image. The resulting image is then specially thresholded using the computed histogram and a black pixel acceptance percentage of 30% (as described in the theory section) to isolate edges in black and background in white. But this is not enough, because noise is normally present throughout the image. Therefore a special closing must occur that enhances the edges while removing the noise. This special closing uses a dilation and erosion that counts the number of neighboring white or black pixels in order to flip pixel values. This has proven effective in accomplishing the edge enhancement and noise rejection task.

Once the image processing is complete, the feature extraction proceeds by searching for vertically slanted lines. Vertically slanted lines are important because a hallway can be defined by two vertically slanted lines which are convergent towards the middle of the image. If you look again at Figure 1, you will see that the ceiling of this image has such convergent lines that define the hallway. This is an example of extracting 3D features from planar images. If two convergent vertically slanted lines are found, we know a hallway exists in an image.

But we can do better. Based upon these slanted lines, we can deduce the angle of rotation required in order to look directly down the hallway. Based upon simple geometry and trigonometric functions, this angle can be computed. The OCR system does this computation in order to guide the robot directly down the hallway.

5.8 Sample Sequence of Images Created

Figure 18 Raw Sign Image

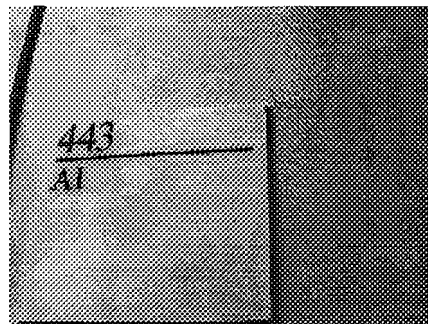


Figure 19 Image Processed Sign Image

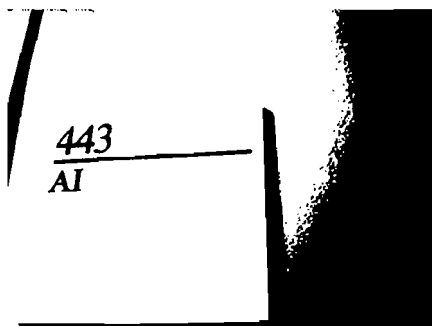


Figure 20 "Corrected" Sign Image

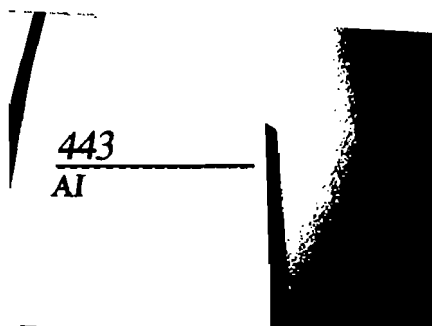


Figure 21 Raw Hallway Image

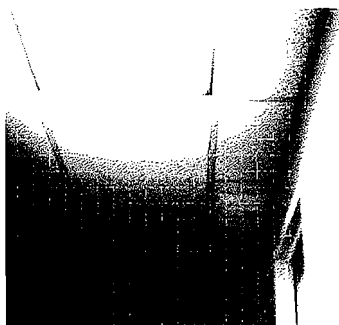


Figure 22 Image Processed Hallway Image



6 Results

The results obtained using the OCR system are preliminary in nature, yet very promising. The results of the OCR system can be subdivided into three sections: sign location, word and character classification, and hallway location. Each of these is treated separately next.

6.1 Sign Location

The OCR system was able to find signs in most images containing them. There were some false alarms, however, meaning the OCR system classified an image as having a sign when it in fact did not. When a sign was discovered in an image, the OCR system was able to correctly find its dimensions almost every time. The OCR system was also able to correctly isolate words in the sign correctly. But when the OCR system explored down to the pixel level, problems arose in trying to isolate certain characters within words.

As the rectangles enclosing the words grew smaller, the OCR system had a tougher and tougher time dealing with errors in the image. This is due to uncorrectable errors in the acquired image. Some words were too small, with the characters within the words having discontinuous features because the smooth lines of the character were too small to be digitized correctly. As the word size decreased and the character size decreased to match, these pixel errors tended to dominate the character. With such small characters, the OCR system was unpredictable at best.

6.2 Word and Character Classification

But with larger words, the OCR system was able to classify characters and words quite well. The office numbers appearing above signs were correctly classified over 90% of the time. And if an error did occur, it involved only a single digit in the number. Signs with large words below were also classified favorably, although the success rate was lower.

The various classifiers did not perform equally. It seemed the competitive network was the best of all classifiers in the presence of noise in all the test images. The template matching classifier was not far behind, though, in its reliability. The back-propagation network, however, was severely lacking in its ability to predict characters. Its reliability was less than half that of the other classifiers. This is due to the training the network underwent. Because synthetic data perturbed by a small amount of noise was used, the network did not have a truly representative sampling to learn with. Luckily, the reliability reported by each of the classifiers was indicative of the “true” reliability of their classifications so that the classifications could be combined effectively.

Yet, the classifiers were not completely accurate. So the dictionary lookup feature was invoked and produced favorable results. Classified words such as ‘okly’ were correctly changed to ‘only’ and ‘exif’ to ‘exit’.

In sum, the classifiers performed relatively well as a group in producing intelligible output words.

6.3 Hallway Location

There was also a good, although lower, probability that the OCR system could detect the presence of a hallway in an image. Once again, major errors were present in most of the images and in some of these images the errors could not be overcome. The most difficult error to deal with was the irregular lighting of the hallways. If an image is not uniformly lit, image processing runs into problems because it mostly considers the image as a whole when thresholding, contrast expanding, or median calculating, etc. Because some hallways had very light and very dark spots, the contrast expansion was not effective in spreading the pixel values properly. This difficulty was propagated throughout the hallway finding algorithms, because edges could not be found and, therefore, features appeared absent. In future versions of the OCR system, this problem can be dealt with by performing an illumination adjustment so that luminosity is constant throughout the image.

Another problem when detecting hallways was the detection of faint or incomplete features. Edges found were only small subsets of the actual edges in the image because of random noise causing lines to appear fragmented. Even after dilations and closings the problem was apparent. Unfortunately, the answer to this problem could not be found.

Even with all these problems, hallway location was able to perform successfully for a large subset of actual hallway images. In the future, it is hoped that this subset will be expanded.

7 Conclusion

The OCR system has shown itself effective in the analysis and use of images gathered by an onboard Robot Vision system. The OCR system has been able to find and read signs in many different kinds of images with varying levels of errors. The system has also been able to successfully find hallways in many images. In short, the OCR system has been able to meet most of its design criteria.

But the OCR system is far from perfect. It does not find all signs and hallways and it does not correctly read all signs. It also performs quite slowly. Because of this lack of speed, the OCR system's use on Ramona was very limited and most of the results were obtained in the off-line analysis of images. The OCR system also makes many mistakes and wrong assumptions. Because the OCR system relies on heuristic techniques and specific image processing algorithms, the OCR system is not easily generalizable. What is needed is a system that can adapt to most any visual environment and find features without knowing a priori what environment the system is in. This idea's fruition is certainly not very far, yet it is not too close either. But this idea is the goal for which the robot vision branch of artificial intelligence continues to strive.

Acknowledgments

Special thanks to Professor Leslie Kaelbling for her support, my wife Trupti for her patience, and my boss Paul for his understanding.

References

- [1] Avrim Blum and Ronald L. Rivest. "Training a 3-Node Neural Network is NP-Complete" *Advances in Neural Information Processing Systems, Volume 1*. Morgan Kaufmann Publishers, 1989.
- [2] Mou-Yen Chen, Amlan Kundu, and Jian Zhou. "Off-Line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network" *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 5, May 1994.
- [3] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. "Handwritten Digit Recognition with a Back-Propagation Network" *Advances in Neural Information Processing Systems, Volume 2*. Morgan Kaufmann Publishers, 1990.
- [4] J.S. Denker, W.R. Gardner, H.P. Graf, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, H.S. Baird, and I. Guyon. "Neural Network Recognizer for Hand-Written Zip Code Digits" *Advances in Neural Information Processing Systems, Volume 1*. Morgan Kaufmann Publishers, 1989.
- [5] Sheri L. Gish and W.E. Blanz. "Comparing the Performance of Connectionist and Statistical Classifiers on an Image Segmentation Problem" *Advances in Neural Information Processing Systems, Volume 2*. Morgan Kaufmann Publishers, 1990.
- [6] John Hertz, Anders Krogh, and Richard Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, 1991.
- [7] Basit Hussain and M.R. Kabuka. "A Novel Feature Recognition Neural Network and its Application to Character Recognition" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 1, January 1994.
- [8] Gale L. Martin and James A. Pittman. "Recognizing Hand-Printed Letters and Digits" *Advances in Neural Information Processing Systems, Volume 2*. Morgan Kaufmann Publishers, 1990.
- [9] Bartlett W. Mel. "Further Explorations in Visually-Guided Reaching: Making MURPHY Smarter" *Advances in Neural Information Processing Systems, Volume 1*. Morgan Kaufmann Publishers, 1989.

- [10] Dorothy A. Mighell, Timothy S. Wilkinson, and Joseph W. Goodman. "Back-propagation and its Application to Handwritten Signature Verification" *Advances in Neural Information Processing Systems, Volume 1*. Morgan Kaufmann Publishers, 1989.
- [11] Yoshihiro Mori and Kazuhiko Yokosawa. "Neural Networks that Learn to Discriminate Similar Kanji Characters" *Advances in Neural Information Processing Systems, Volume 1*. Morgan Kaufmann Publishers, 1989.
- [12] Yoshihiro Mori and Kazuki Joe. "A Large-Scale Neural Network Which Recognizes Handwritten Kanji Characters" *Advances in Neural Information Processing Systems, Volume 2*. Morgan Kaufmann Publishers, 1990.
- [13] Jun Ohya, Akio Shio, and Shigeru Akamatsu. "Recognizing Characters in Scene Images" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 2. February, 1994.
- [14] Dean A. Pomerleau. "ALVINN: An Autonomous Land Vehicle in a Neural Network" *Advances in Neural Information Processing Systems, Volume 1*. Morgan Kaufmann Publishers, 1989.
- [15] Jairo Rocha and Theo Pavlidis. "A Shape Analysis Model with Applications to a Character Recognition System" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 4, April 1994.
- [16] D.E. Rumelhart and D. Zipser. "Feature Discovery by Competitive Learning" *Readings in Machine Learning*. Morgan Kaufmann Publishers, Inc., 1990.
- [17] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. "Learning Internal Representation by Error Propagation" *Readings in Machine Learning*. Morgan Kaufmann Publishers, Inc., 1990.
- [18] John C. Russ. *The Image Processing Handbook*. CRC Press, 1992.
- [19] Li Wang and Theo Pavlidis. "Direct Gray-Scale Extraction of Features for Character Recognition" *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 15, No. 10, October 1993.
- [20] Toru Wakahara. "Shape Matching Using LAT and its Application to Handwritten Numerical Recognition" *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 16, No. 6, June 1994.

- [21] Patrick Henry Winston. *Artificial Intelligence: Third Edition*. Addison-Wesley Publishing Company, 1992.

Appendix A User's Guide

The OCR system is very easy to use because it is a hands off system. You simply invoke it with a few command line arguments and sit back and watch. The system is designed to run on Ramona, the mobile robot, but can also run on a SUN SPARC 10 machine on simulated data. The next two sections describe the invocation arguments and the typical system output respectively.

A.1 OCR System Invocation

The OCR system is invoked with the following command line:

```
<bum@vramona>% ocr [-rwns]
```

All of the command line parameters are optional. Here is a description of each of them:

-r [competitive backprop]	This option with no following arguments tries to read the competitive and back-propagation neural network setup files named "competitive_network" and "backprop_network" respectively. If either the argument competitive or backprop follows this option then only that neural network setup file is read, the other neural network is therefore meant to be trained. If this option does not appear on the command line then it is assumed the neural networks are both meant to be trained.
-w [competitive backprop]	This option with no following arguments tries to write the competitive and backpropagation neural network setup files named "competitive_network" and "backprop_network" respectively. If either the argument competitive or backprop follows this option then only that neural network setup file will be written. If this option does not appear on the command line, neither of the neural networks will be written.
-n <number of iterations>	This option defines the number of iterations through the OCR system should be run before stopping. The default is continuous running of the OCR system until the user hits <Ctrl-C>. The number of iterations is equal to the number of different images acquired and processed by the system.
-s <simulation file>	This option directs the OCR system to use the simulation file argument in order to read a list of image files from memory and process them in sequence. This option is most useful when debugging from a SUN workstation.

A.2 OCR System Execution

The OCR system provides feedback to the terminal as well as through the robot's speech system and navigation system. A typical example of execution output is the following:

```
<bum@vramona>% ocr -r -s simulate.sign
```

```
Welcome to the OCR System  
Version 1.0  
by Bobby Mander
```

```
Reading the synthetic training letters from file.  
Reading dictionary.  
Reading the competitive neural network from file.  
The competitive neural network got 0.98 correct.  
Reading the backprop neural network from file.  
The backprop neural network got 0.50 correct.
```

```
Reading data/s1.pgm  
We found a sign!  
The sign had 2 words.  
Word 1 was 443  
Word 2 was AI
```

```
Reading data/s2.pgm  
We found a sign!  
The sign had 2 words.  
Word 1 was 443  
Word 2 was AI
```

```
Reading data/s3.pgm  
We found a sign!  
The sign had 1 words.  
Word 1 was 445  
Reading data/s4.pgm  
We found a sign!  
The sign had 1 words.  
Word 1 was 445
```