Real-Time Camera-Based OCR

W.J. Baron

MASTER'S THESIS

July 2002

University of Groningen
Artificial Intelligence Department

Supervisor:
Prof. Schomaker

968

Real-Time Camera-Based OCR

W.J. Baron

MASTER'S THESIS

July 2002

University of Groningen
Artificial Intelligence Department

Supervisor:
Prof. Schomaker

Abstract

This paper describes a system for performing real-time text detection and optical character recognition in fairly complex indoor scenery. It is initially intended to provide navigational cues to robot agents and it operates by coupling text detection utilizing localized measures (Mirmehdi) with neural network based optical character recognition techniques through a region growing process.

Text detection using statistical measures has been shown to be strongly invariant to the colour, orientation and size (including illegibly small) of text regions while convolutional neural networks display the same relative invariance with respect to the position and scale of individual characters. The two techniques complement each other and ease the real-time requirement; text detection provides information on where text may be, region growing on its orientation and size, and convolutional networks identify the characters that are actually present.

The camera-based reading process works as follows; various local properties of a (video) image such as the variance, edge count, density, and orientation are employed by a feed forward network to identify regions as text or non-text. Likely candidates are then selected using a fast constrained region growing technique and these are normalized and fed to a convolution feed forward neural network which performs optical character recognition. Depending on the application, the resulting characters may be processed further or simply be projected back onto the original location to restore the spatial orientation.

While the current project targets robot vision, a robust system would have many applications such as providing an aid to the visually impaired or searching digital media, as well as offering some fascinating possibilities in the fields of augmented reality and wearable computing.

Contents

Preface

This paper is the result of eight months of reading, writing, and programming, and forms the final requirement of the study Artificial Intelligence at the University of Groningen, The Netherlands. My interests lie in robotics and computer vision so when Professor Schomaker suggested robot reading to me it was an instant hit, and the topic has over these months been a pleasure to work on.

It is my current experience that the hardest part of research is documenting results. Ideas quickly become familiar and then seem logical. Soon this logic is taken for granted and the thirst for new ideas begins. The fields of robotics and computer vision are both huge and fascinating which is a particularly dangerous combination for any curious mind. It is for this reason that I would like to start by thanking my supervisor Professor Schomaker for encouraging new ideas while at the same time somehow helping me avoid the pitfalls and many side roads, thereby ensuring I reached this stage of completion.

I would also like to thank my current employers Mathieu van Echtelt and Wouter Gazendam for making it possible for me to work part time during this period, for allowing me to make use of office equipment, and for their patience and flexibility. Lastly I would like to thank my parents for a lifetime of support and my girlfriend Nelleke for her love, encouragement, and those occasional timely words of wisdom. Thank you!

Groningen, July 2002

Wiebe J. Baron.

1 Introduction

Our ability to read plays a big role in our sense of orientation. Those who have ever been lost in a foreign city will recognize the horror of not understanding signs and street names, or the destinations of busses and trains. The problem is exasperated by the tendency of our artificial world to look too familiar; a building, a street, or a city looks just like any other. The same applies to indoor environments. If all signs were removed from the average building complex many people might well have trouble finding their own offices. The typical research robot exists in such an environment, every time it powers up it is thrown into an unknown world. Its only hope for self-orientation and the topic of this paper is for the robot to be able to read. Robot labs can of course be modified to simplify the task of navigation. Floors and roofs may for example be coded, or lights and other electrical devices used to mark entrances and charging stations etc. But a robot capable of reading text would require no more modification to an indoor environment then a human would. Floor, hall and room numbers would suffice for basic navigation and these could be changed without requiring a software update freeing the robot from the confinements of its birthplace.

A system capable of recognizing text in real-world scenes could in many ways be considered simply a generalization of current optical character recognition (OCR) packages widely available on the market. But not being restricted to a piece of paper on a scanner introduces a great many possibilities that go far beyond what OCR has yet had to offer. While this project has developed from the field of robotics and several design choices have been made that reflect that point of departure, there are many other applications for camera-based OCR worth mentioning and some of these will briefly be discussed before moving on.

One use for camera OCR which closely resembles the office navigation task is identifying labelled objects. Several such systems are already in use today although they operate in rather specialized fields. An example is the automatic registration of license plates from video images, a process that is fairly advanced and is actually used on the roads today. Other possible applications include bar codes that can be understood by both humans and machines, and security systems for tracking employee movements.

Media archiving would also benefit enormously from real-world OCR. The extraction of captions and text content would help organize and search through what may be several billion hours of archived video broadcast collected since the invention of film. The same applies to searching the internet, where text is increasingly embedded in graphics and film.

An exciting area for camera-based OCR is the field of augmented reality. Black et al [1] of Xerox PARC provide an overview of a project called 'The Digital Office' in which camera systems interpret human gestures and scan paper documents etc. They describe a fascinating camera-based system called Zombieboard which allows an arbitrary whiteboard to be used as a computer interface; an action can be performed for example by drawing a button and selecting it by marking it with a cross or a check (Igarashi [12] is another recommended read on whiteboard interaction). These areas are so promising because they can redefine how we work; rather then humans being forced into the restrictive and unhealthy world of computers, keyboards and mice, computers are instead moved into our world and our natural forms of interaction. Camera OCR will play a critical role in making such as step possible.

Yet another augmented reality type application is in the field of wearable computing. A camera and display unit embedded in a pair of glasses could allow an advanced OCR system to augment the wearer's view of the world with an endless range of information. It could for example be used for the automatic translation and projection of signs, building names, menu cards or documents when on foreign business trips. Such an augmented system could also assist in performing tasks much like the Zombieboard does; an example application that assists with written calculations and is based on the system developed in this paper is described in Appendix B.

The aim of this paper is to work towards the goal of camera-based machine reading, details of which will be described more precisely in Chapter 2 along with some problems that may arise along the way and possibilities that should be considered. In Chapter 3 several publications of work done on camera-based OCR or similar fields will be reviewed and ideas evaluated for their applicability to our goals. Chapter 4 then describes the system that has been designed here to address those goals and some results are presented in Chapter 5. Lastly Chapter 6 offers some concluding remarks and a fairly lengthy list of suggestions for further work.

2 Formulation of the Problem

In this chapter we will formulate the problem of machine reading in an environment which for now we will simply describe as being more generalized then the typical document scanner. Exactly how general we should define our problem will become clear by first looking at some background on human and machine reading and analyzing what our goals are. We will then narrow down the problem space by introducing a set of assumptions and also look at some issues that will arise and need to be solved when pointing an optical character recognizer at the real world. Finally we will outline the specific problems we would like to address in this paper.

2.1 Background

From a biological perspective the human visual system seems to be divided into two general subsystems, one that is responsible for detecting *where* objects in a scene are and the other for answering *what* those objects are. These two subsystems work together and are closely coupled but not much is known about the actual nature of their interaction. Several general models have been proposed however, figure 1 (b) is a typical example of a '*what and where*" model developed by Rybak et al [17].

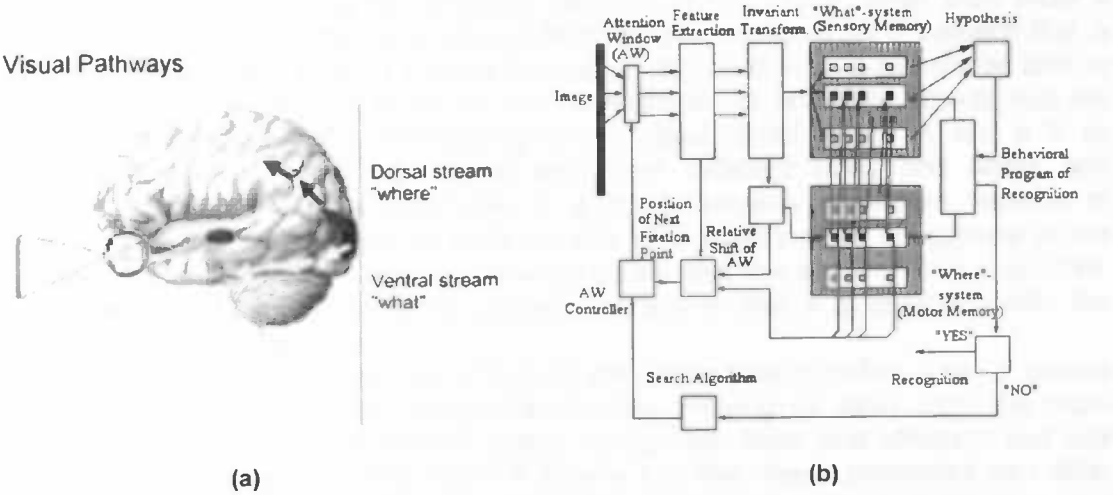


Figure 1: (a) What and where subsystems of the visual cortex. (b) A model for the implementation of a 'what and where' system. A low-level subsystem detects primary features while higher-level sensory and motor memory structures predict and control the attention window (source: Rybak et al [17]).

There are many interesting things about the human 'where and what' system. It seems to approach vision as a behavioural process for example, where the eye moves and then fixates on informative parts of an image, guided by a problem related selection mechanism. This is not just a matter of computational efficiency but also plays an actual role in image identification. Indeed the problem oriented nature of perception is apparent in the asymmetry of certain visual tasks. When reading for example, text recognition appears to take place in a region local to the fixation point extending about 4 characters to the left and 14 to the right [4]. Peripheral vision guides this focus by identifying word

boundaries and the general flow of text. Clearly this reading asymmetry is an acquired skill, there being languages where the direction is reversed, but this suggests that we as humans have a vast number of what we might call heuristics at our disposal which we can apply concurrently to solve specific tasks.
The process of optical character recognition is fairly well developed and effective when the precise position of the characters has been located (generally through a process known as segmentation, in effect answering the 'where' question) and skew, scaling variation and rotation of letters is minimal. When it comes to camera vision however, these conditions are rarely met and traditional OCR systems fail. One reason for this is that we humans use a great deal of our general knowledge of the world when we read. We know that paper is often covered in text and that surfaces such as walls may hold writing too, or that some materials are easier to write on then others and that for example water and rock and living organisms such as trees generally do not contain text. If a machine was to read the way we do, it would have to posses a great deal of our qualities. Designing a system that can recognize any object in general is beyond our current technological means but we can perhaps set a more reasonable goal by looking at what makes text different from the general class of objects and use this knowledge to introduce some assumptions that will limit the scope of the problem.

2.2 Assumptions and Considerations

We will start by limiting the camera input to the world as experienced by an indoor autonomous robot, this being the targeted platform of this study. What then must a system be able to do in order to read text from images captured by a camera that is capable of looking at and moving around the real world? Must such a system be able to recognize paper as a common holder of text for example, or chairs as objects that are unlikely to contain text? And is recognizing text all about identifying an 'A' and a '9' as individual objects that could be rotated and have different sides and styles and dimensions, much like say a telephone or a cup? Certainly a system capable of recognizing all such objects might be quite capable of recognizing text but general object recognition is not a minimum necessity because clearly text has some unique properties. Perhaps the most obvious distinctive property of text is that it is designed with the general intention that it be read!
By examining some of the properties of text we should be able to define a set of criteria that simplify the process of text recognition thereby moving us away from the more global and difficult problem of general object recognition. Note that although text has some very specific properties that make it appear text-like; these properties can differ from language to language. The images in figure 2 of English and Chinese text demonstrate that different languages, as patterns viewed from a distance, can look very different.

(a) (b)

Figure 2: A comparison between English and Chinese characters as viewed from a distance.

Note that the system should be able to recognize possible text areas even if the text itself is too small to be read because this is inevitably the case with low resolution images. The mobility of the system should compensate for this by moving towards regions of interest. The observation that languages look different from a distance means that for now the targeted text will be restricted to Roman characters and the English language in particular.
Some final points to consider are related to the target robot platform itself, in this case a Pioneer II from ActivMedia Robotics. Because of its small size the robot is always looking up at the world, and this means that text is almost always distorted in a manner that cannot be corrected by zoom or relocation. The processing power available to the OCR system is also quite limited while the goal is for it to at least approach real-time operation, so clearly the system must be as lean and efficient as possible. Lastly the system should work under the typical variety of indoor lighting conditions.

2.3 Problem Definition

The purpose of this project is to design a system for an autonomous robot which when offered a typical indoor scene can recognize textural regions and, if legible, identify the text present in the image. This involves solving the following three central problems:

1. **Recognize a pattern as being text or not**

If a pattern is recognized as text but is too small to be read the robot can zoom in with its camera or reposition itself.

2. **Identify the actual characters present in a text region**

If the text regions are legible the following step is to identify the characters. The system must be able to deal with rotation and some degree of distortion.

3. **Perform 1 and 2 quickly and efficiently**

The system must operate in real-time with modest CPU and memory requirements.

Of course many additional systems will be required to actually make use of the system such as OCR post processing to improve recognition rates, camera and robot positioning control, and some sort of learning or goal driven system to give the robot purpose. These will be looked at briefly, though an in depth discussion is beyond the scope of this project.

3 Survey of Results in Literature

In this chapter we will briefly discuss the current state of OCR and then look at some research into camera-based character recognition.

3.1 Optical Character Recognition

There is a great deal of literature available concerning optical character recognition. The first reference to machine reading was in a 1929 patient filed by G. Tauschek in Germany and with the event of digital computers in the 1950s OCR quickly became the primary research field of pattern recognition. Since then OCR has developed into what we might call an industry problem, that is to say that central hurdles have been overcome with character recognition rates over 99% for printed text, but that the issues are now one of data collection, performance, document layout analysis and indeed marketing. On the other hand these high recognition rates only apply under extremely constrained conditions, input for example is assumed to come from a high resolution document scanner and the documents themselves are assumed to be of high quality print, consisting of properly aligned single colour text. Handwritten or distorted text documents are generally very poorly recognized and are therefore still an active area of research today. And of course there are still many improvements to be made with document scanners such as the preservation of layout and style, and the recognition of special characters such as mathematical symbols.
The process of optical character recognition generally consists of four steps; the first is document analysis where various techniques are used to break down the layout of a page into paragraphs, lines and individual characters. Characteristic features are then extracted such as strokes, end points, holes etc thereby reducing the classification search space and, with correctly chosen features, introducing some invariance to scale, rotation and distortions. Sets of features are then matched according to some distance analysis or template matching scheme and the results are then subject to a contextual processing system which makes use of syntactic and semantic knowledge to select the most probably output.

3.2 Real-World OCR Literature

While much work has been done on general text recognition, camera-based text detection literature is considerably rarer. This is perhaps in part due to the magnitude and multifaceted nature of the problem; it is both a superset of standard document OCR and involves complicating issues from other fields such as computer vision, robotics and mechanical systems. Unlike traditional OCR, there are in addition currently no standardized training sets or procedures available, making comparisons difficult as well. In this section we will review some research on real-world OCR, particularly the work of Clark and Mirmehdi [2,3] and LeCun et al [5,6,7]. The remaining literature surveyed here is selected because it either contributes to a possible design of, or directly addresses, some aspects of camera-based reading.

3.2.1 Text Detection using Localized Measures

In multi-step OCR systems text detection is usually the first step undertaken. The existing approaches to text detection generally use techniques such as edge filtering, texture segmentation, colour quantization and neural networks and bootstrapping [16]. A novel approach to text detection is that taken by Clark and Mirmehdi [2] who demonstrate a technique based on the local statistical properties of an image. What is particularly interesting about this method is that if offers a considerable amount of invariance to text orientation, scale and colour, and is yet simple enough to offer real-time performance.
The technique makes use of characteristic textural properties and works by applying several different transformations to local regions of an image, thereby capturing what are hopefully complementary aspects of the text areas. Clark and Mirmehi propose the 5 measures detailed below:

1. The variance of the greyscale histogram over a local circular region N:

$$M_1 = \sum_{i=1}^{N} \left(H(i) - \overline{H}\right)^2$$

2. The edge density over a local region M, where E(i) is the edge magnitude at a pixel location found using a Sobel filter:

$$M_2 = \sum_{i=1}^{M} E(i)$$

3. The histogram variance over a local region:

$$M_3 = \sum_{i=1}^{8} \sum_{j=1}^{B} \left(H(j) - G_i(j)\right)^2$$

   This measure indicates the changes in variance over an area by summing the squared difference between the central histogram H and its eight neighbours G_i.

4. The asymmetry across edge angle histograms:

$$M_4 = \frac{1}{E} \sum_{\theta=0}^{\pi} \left(A(\theta) - A(\theta + \pi)\right)^2$$

   This measure looks at the distribution in the direction of edges over a local region. $A(\theta)$ is the total magnitude of edges in direction θ and E the total edge magnitude which normalizes the measure. There are various methods for computing these values such as using Compass Edge Detectors (see 4.3.2 for an example).

5. Edge magnitude distribution over a local region.

$$M_5 = \sum_{\theta=0}^{2\pi} \left(A(\theta) - \overline{A} \right)^2$$

This is similar to measure 3 but now with respect to edge magnitudes.

These different measures can be shown to respond differently to text and non-text regions but because no single method fully identifies text they are then combined using a neural network. Figure 3 for example shows an input image (a), its local variance plotted as a 3D histogram (b), and the output of a neural net that has apparently learned to recognize intermediate variances as text (c).

(a) (b) (c)

Figure 3: The text from input image (a) shows low responses in its local variance histogram (b) due to aliasing. The system must learn that a low variance is an indication of the possible presence of text (c).

Clark and Mirmehi [3] use the output of such a neural network to mark text regions in the original image and then threshold them by using partial sums and neighbouring regions (intensity correction) for threshold selection. The orientation of text plains are then estimated by calculating horizontal vanishing points using a projection profile searching technique. These vanishing points are then used to transform the text regions so they are fronto-parallel and suitable for traditional OCR processing.

3.2.2 Text Recognition using Convolutional Networks

LeCun et al [7] have extensively demonstrated the relative local invariance of convolutional neural networks to the scale, position and orientation of input characters. Convolutional networks are different then fully connected neural networks because they belong to the class of networks in which the structure itself is specially designed to play a role in classification. With convolutional networks a combination of weight sharing and local connections causes the induced local field of a given hidden neuron to yield a convolution sum, hence the name (see [9] for details).
Figure 4 is an example of a multilayer convolutional network designed by LeCun (LeNet-5) to perform hand written digit recognition (note that the input image contains an 'A' which is somewhat confusing as it suggests that we are dealing with a letter rather then a digit classifier). Every layer in the network (except for the input layer) has a number of feature maps and each feature map has one set of weights that extract certain features from the previous layer through the process of convolution. These feature maps are then sub-sampled and form the input for the next layer. This alternation between convoluting and sub-sampling layers is inspired by the simple and complex neural cell orderings described by Hubel and Wiesel (1962). It has the effect of slowly allowing the feature

space to grow while reducing the spatial resolution and thereby introducing invariance to position and size [7].

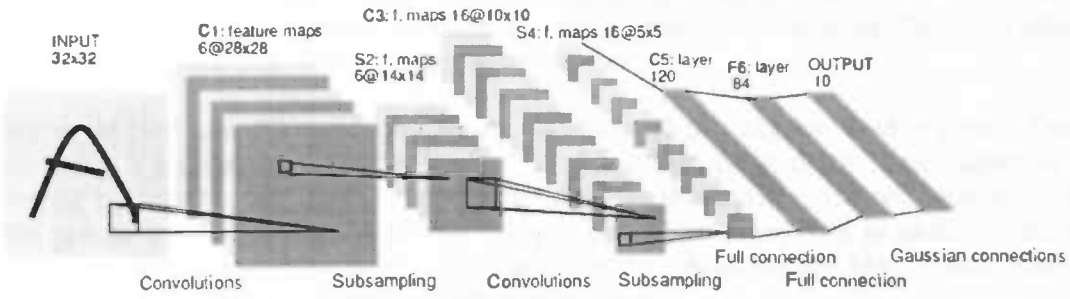


Figure 4: Architecture of a convolutional network (LeNet-5) showing the reduction of spatial resolution and the growth of separate feature maps (source: LeCun et al [7]).

The network above cannot be applied to an entire document because the loss of spatial position and conflicts between multiple characters would render it inoperable. The network could however scan across an image classifying the local input at each pixel location. This would result in a probability landscape which would then need to be analysed to extract individual characters. Through weight sharing and local connections the network is kept fairly small considering the number of layers and the task it performs, having approximately 340,000 connections but only 60,000 free parameters. An additional benefit of weight sharing is that it lends itself well to parallel operation on specialized hardware or multiprocessor systems.

3.2.3 Other Text Detection and Segmentation Techniques

Wu, Manmatha, and Riseman [20] describe a four step system for text detection and recognition. First a text segmentation module, based on the standard approach of applying a series of Gaussian second order derivatives, is used to classify textures. Strokes are then extracted from the segmented areas to build bounding boxes for the text-like regions. Text is then cleared up and binarized, rebounded for a tighter fit, and lastly fed to a commercial OCR system. The system is able to extract text with complex backgrounds though the orientation of text is restricted to an angle of less then 30 degrees.

Sato et al [18] address the extraction of overlaid text captions from video images. To compensate for the low resolution broadcast images, sub-pixel interpolation is used in combination with multi frame integration; the latter also supporting the removal of dynamic complex backgrounds. Edge filtering and cluster detection is used to find text boundaries which are extracted if they meet certain aspect ratio and size conditions. Character segmentation is performed with a simple technique using vertical projection profiles, made possible because the system is restricted to horizontally spaced and white coloured text. The segmented characters are then processed by a commercial OCR package.

J. Yang et al [21] developed a camera-based automatic sign recognition and translation system for Chinese words and characters. The system is based on adaptive edge detection and heuristic search for text location, and Gaussian mixture models for

segmentation. The system is for various reasons not able to deal with rotated text however. It uses an Example Based Machine Translation (EBMT) system for recognition and translation. Word recognition is very complicated because of the presence of over 3000 possible characters. Most Chinese/Japanese recognizers use a multi-step approach where a pre-classifier is uses to reject unlikely candidates thereby reducing the possible outputs to a more manageable number.

Myers et al [16] use 3D geometry to try and normalize detected text regions. Text is found using a technique similar to Wu's [20] after which optimal top- and baselines are extracted by rotating text regions and searching for the maximum horizontal projection profile gradient for all angles. A similar vertical projection procedure is used to calculate image shear and these parameters are then used to normalize the text. These methods appear to be very computationally expensive however.

Li et al [13] use a neural feed-forward network to locate text in video images by computing high-frequency wavelet coefficients of a frame which form the inputs to a network in a way similar to Mirmehdi [2]. A pyramid approach is taken to allow the recognition of a range of character sizes. An interesting detail is that if an input is recognized as text the whole window area is marked as being text, rather then just the central pixel location. Bounding boxes are then created by connected component analysis and lines of text are separated using horizontal projection profiles. Finally text regions are tracked using a module based on the Sum of Squared Differences.

Lienhart et al [14] developed a system for localizing and segmenting text in images and videos. The text detector is a feed-forward neural network and is trained for a very narrow set of positions and scales, but is able to achieve invariance by being applied to every position of multi-scaled images. Like others, projection profiles are used to detect separate lines and characters. The system is limited to horizontal text which supposedly accounts for 99% of text occurrences. The limitation is imposed because of the diminishing returns of a rotation independent approach. They argue that as long as actual recognition rates are significantly below the 99%, just achieving horizontal text recognition is a big enough challenge.

Each of the systems described here have their own set of limitations. For example only those of Myers [16] and Mirmehdi [2] are designed to deal with rotated text. Sato's system is specifically designed for captions and therefore has certain limitations, and makes use of certain features, not present in camera-based OCR. Lienhart, Li, Myers and Sato all make use of projection profiles for character segmentation which limit the possible character styles and may be computationally expensive. We hope to improve on these systems in two respects; a greater invariance to rotation and scale, and much more modest processing requirements across the entire system. Possibilities for integrating the camera itself as a behavioural component of the system will also be looked at as this could offer alternatives to certain computationally expensive procedures.

4 Text Detection and Character Recognition

This chapter introduces the approach taken in addressing camera-based real-world OCR. It starts by reviewing several design decisions that were made early on with respect to learning procedures, and then presents some initial exploratory experiments that demonstrate several ideas and possible difficulties. Addressing these then leads towards the final design model which is discussed in the remainder of this chapter.

4.1 Design Decisions

There are many forms of learning. Some examples are decision tree learning for approximating discrete valued target functions, Bayesian learning which is probabilistic and can cover a whole range of functions, genetic algorithms, instance-based, analytic, and reinforcement learning. Each has their own advantages and disadvantages with respect to specific problems. For real-world reading we have chosen neural networks as the tool for learning because they are robust and well suited to dealing with noisy high dimensional camera input. The correct choice of inputs, layers, hidden neurons and so forth, are critical to the success of such a system. There is a minimum network size for example, under which performance drops and a maximum size above which the network start to over-fit, run slower, and learn new examples less quickly. There is however no inherent intelligence in neural net training; it simply performs gradient descent on some error function typically describing the difference between a desired and actual output. Although a monolithic network might learn to read given sufficient data, for fast learning and response the real intelligence must lie in the design of the system as a whole. The emphasis here is then not on neural networks per se but rather on the generation of a robust architecture.

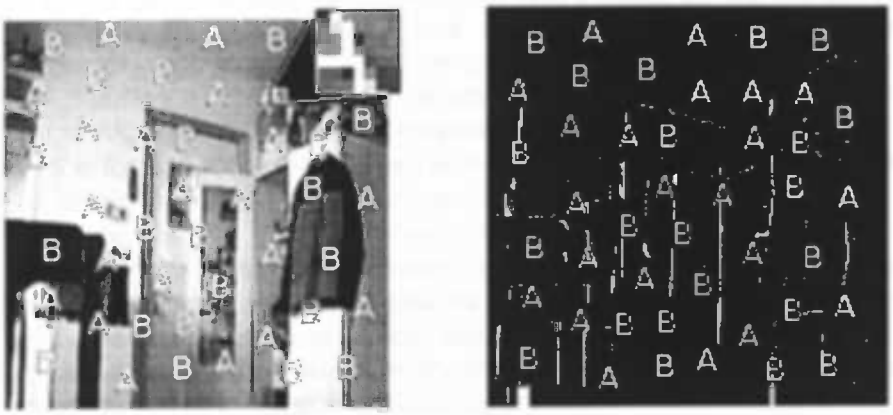
4.2 Initial Study

The text reading system is intended to run on actual robot hardware with rather modest processing power and this has therefore led to a careful consideration of approaches that might ease computational requirements. An example of this is the biologically inspired "*where and what*" concept mentioned in chapter 2 which is an approach often taken in commercial OCR systems, though it is typically a process that takes place in two distinct serial steps as this reduces overall complexity. However an integrated approach is desirable because it should offer training benefits and flexibility, so initially both multi-step and integrated approaches were example.

4.2.1 Simple Convolution Window

The standard OCR technique entails identifying where text is and then later what the text is. Figure 5 (a) and (b) shows the input and output of a text segmentation system that uses a three layer neural network trained to recognize uppercase characters. The network consists of 49 input neurons, 7 hidden neurons, and 2 output neurons to distinguish text from non-text. Training is with back propagation over approximately 300 epochs with momentum and a high learning rate (0.1). A window scans across the entire

input image and the raw contents of the scanning window form the input to the network, the output of which is used to construct a hypothesis map (b).



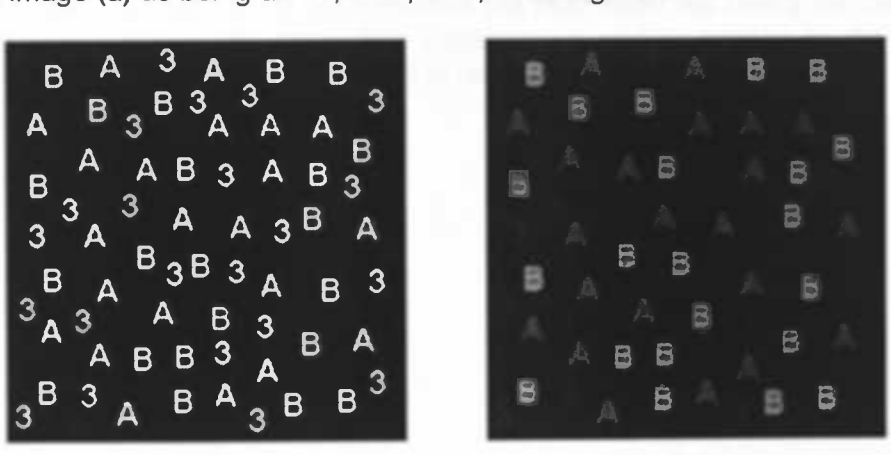
(a) (b)

Figure 5: A windows is run across an input image (a) and its contents are passed to a three layer neural network trained to recognize the overlaid uppercase characters as text, producing hypothesis map (b).

Clearly the example is somewhat artificial but it raises the important question of what the next step should be. Classical OCR makes use of this idea of "*where and what*" more out of necessity then choice because cleanly segmented text is essential for the application of the various hand crafted feature extractors and classifiers typically used to identify individual letters. A consequence of this multi-step and hard-wired approach is that the process is not continuous from beginning to end and therefore not directly suitable for global training.

4.2.2 Integrated Approach

An alterative might be to implicitly answer the "*where*" question by simply designing a system that answers the "*what*" question at every spatial location. An example of this is shown in figure 6 where a similar network to the one above attempts to classify every location in image (a) as being an 'A', a 'B', a '3', or background.



(a) (b)

Figure 6: A three layer network with four output neurons is trained to recognize the characters 'A', 'B', '3' or background. Tested on input (a), it produces plot (b) which is colour coded according to the most active output activation.

This system is again somewhat simplistic because of the cleanliness of the input data. Although the letters seem to have been identified and located almost perfectly, what has in fact happening is that the network has simply memorized nearly every correct mapping. To identify the letter 'A' for example, the network simply had to learn 62 patterns corresponding to the 62 pixels of which the letter consists. To address such over fitting, large networks like LeCun's (discussed in chapter 3) are specially designed to encourage the formation of feature extractors which are shared between letters and generalize to different scales and orientations. And indeed, applying such networks should results in results like that above even on real-world images with the difference that the output would be smudged because convolutional networks are not sensitive to position. However, for performance reasons this approach is not practical for our purposes; even the restricted experiments above border on the limits of our real-time requirements. A full scale analysis of every pixel location is simply computational too expensive both in terms of training and actual operation[1].

4.3 Model

The model introduced here is currently implemented as a four step system that resembles the traditional approach in that these steps are serial. The reasons for a serial model are pragmatic; it offers real-time operation and clear data flow but this may come at some cost in overall performance so we will discuss feedback possibilities at the end of this chapter and in chapter 6.

4.3.1 Description of Model

The four steps are displayed in figure 7 and consist of text detection, region growing, character classification and post processing modules; the first two loosely corresponding to segmentation in traditional OCR systems. Only the first three will be discussed in detail as they are the initial targets of this study. The dotted lines indicate points of interaction or feedback with other subsystems responsible for camera control, navigation and planning, and also included is a system labelled "*what and where*". None of these subsystems have currently been implemented but their roles will briefly be discussed in section 4.3.6. In addition, an alternative model more closely related to the biological "*what and where*" system will be proposed in chapter 6.

[1] While convolutional networks can be optimized by using window differences, the best case scenario for a 7x7 input window on a 200x200 image still requires over 800 full passes, which in combination with 300,000 connections puts us well beyond the limits of real-time operation.

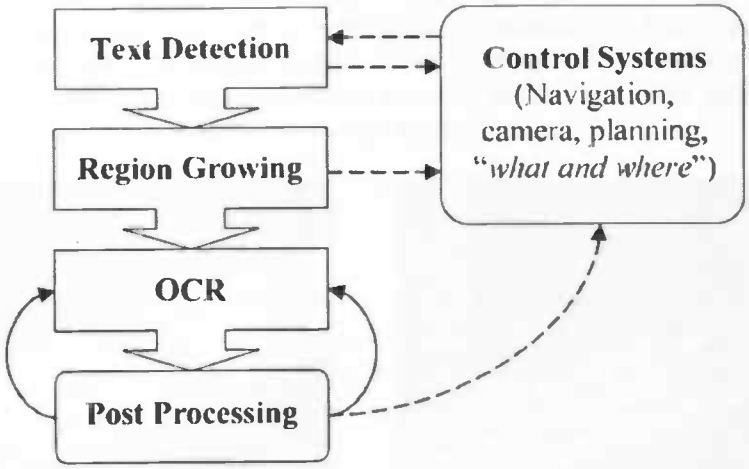


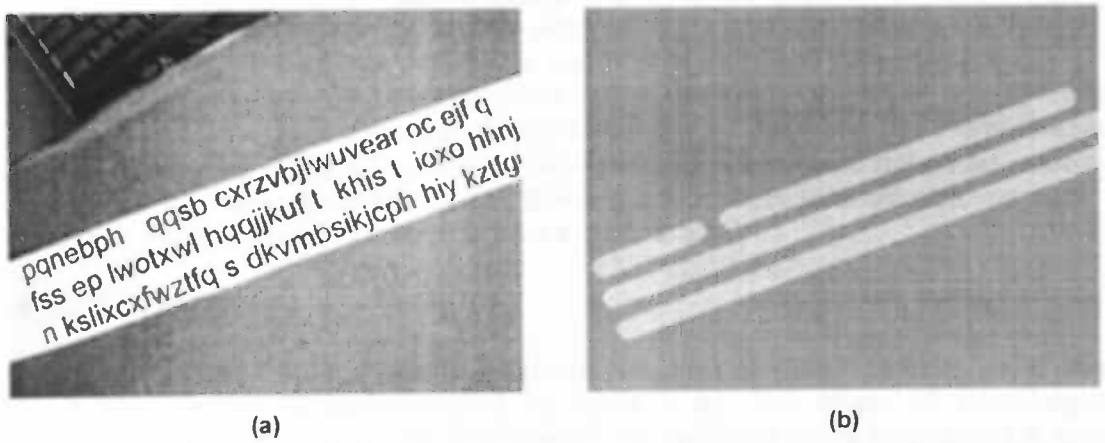
Figure 7: Process flow model of the camera-based OCR system. The dotted lines indicate unimplemented connections; the current system is purely serial.

The details of this multi-step approach will be described in the next few sections but first we will look at how they facilitate real-time operation. The text detection module uses a computationally simple and readily optimised process based on localised measures (Mirmehdi et al) to mark image areas containing text (or conversely reject areas without text). This greatly reduces the area over which a character classifier might have to operate. The next region growing module further simplifies the classification process by generating likely text boundaries, and by normalising the scale and rotation of identified text regions. The image classifier itself is currently a simple time delayed neural network which runs linearly across the bounded input strips generated by region growing. A full scale 2D convolutional network (LeCun et al) might still be preferable however, even within the boundaries of the strip, as it offers an approach that can deal with some degree of local variation in position, shape and size. This, in combination with text detection and region growing should provide a first step towards robust visual text recognition.

4.3.2 Text Detection

In chapter 2 we asked the question of what makes text appear as text. One answer is that text is a collection of certain visual patterns or properties. Some typical properties of printed characters are for example that they tend to contrast well with the background and that they are usually facing in the same directions and are roughly of the same size as their neighbours. Letters are also almost always found near to each other, and roman characters in particular tend to be lined up in parallel rows. It is the combination of such properties that contribute towards our definition of text, and it is this concept that we will use to implement text detection.
The text detection approach taken here largely follows the work of Mirmehdi et al [2]. Several local statistical properties of an image are calculated and fed to a multi layer feed-forward neural network which is trained to recognize certain combinations of measures as text or non-text. There are two slight differences with respect to Mirmehdi, one being that the focus here is on identifying individual lines of text rather then text regions, and secondly that the training process is specially designed to accommodate the following processing step which is region growing. The latter is demonstrated by the

images in figure 8. Figure 8 (b) is the target image for network training which for every pixel location indicates whether the corresponding location in figure 8 (a) should be recognized as text or not. The target image doesn't follow the contours of the actual text but rather fills in small gaps between and around letters with the aim of generating outputs that form strips much like the depicted target image.



(a) (b)

Figure 8: Source (a) and target (b) images for network training. The training image marks horizontal strips of text as this is the desired target response for the trained network.

We use a slight variation of the statistical measures proposed by Mirmehdi to identify several different general properties of text (see 3.2.1 for details):

- Variance of greyscale histogram in a local 6x6 pixel region
- Edge density in a local 6x6 pixel region
- Change in variance across a 4x4 pixel region[2]
- Degree of asymmetry in edges across a 10x10 pixel region[3]
- Edge direction distribution across a 10x10 pixel region

Note that the selection of suitable properties is critical when using a hard-wired approach such as localized measures. The properties must be as generally applicable as possible to the huge variety of possible characters yet also describe significantly different features of text; that is that they should be fairly orthogonal amongst themselves. One set of statistical measures is usually not sufficient to describe all possible character sets and therefore the following constraints on the input are introduced:

- Both the text colour and the background colour should be uniform and have high contrast grey values. This is necessary for meaningful measures of changing variance and for the generalized calculation of edge symmetry and distribution.
- Characters should be no more then 30 pixels in diameter (height or width). This is because local measures cannot capture characteristics of characters significantly larger then the filter dimensions.

[2] Note that this measure differs from Mirmehi's in that he uses a continuous spatial density assumption.

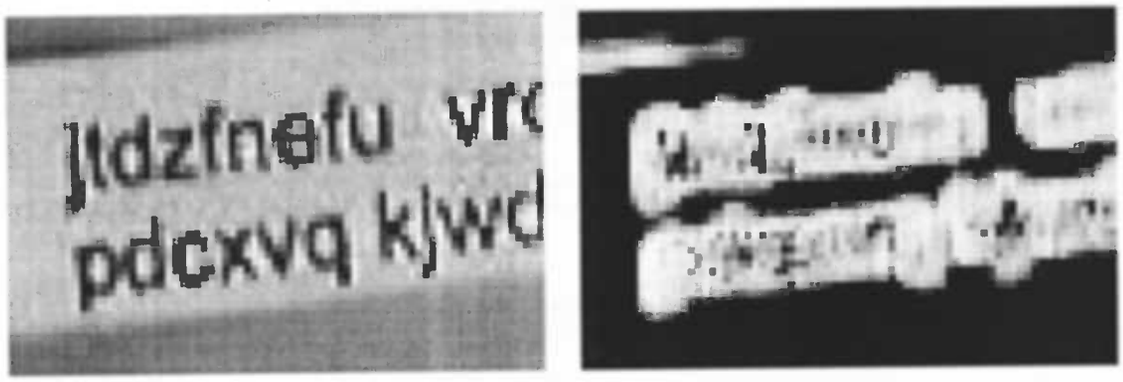
[3] Edge magnitude and orientation is approximated using Prewitt compass edge detecting kernels sensitive to directions spaced across eight 45° steps.

- Characters should be greater then 4 pixels in height. The characteristics of text much smaller then filter dimensions also cannot effectively be captured.

Note that Li et al [13] uses a different training procedure in that a window is only recognized as text if the entire input region contains text. This has the advantage of a much stricter classifier but the disadvantage that it rejects characters smaller then the window sizes. Allowing it to properly deal with rotated text would also require the use of circular filters which, though not prohibitive, are inconvenient and time consuming.
The statistical measures described here risk facing the same difficulties that all feature extractors do, notable poor generalization and flexibility. If the measures themselves are inherently general for the given problem space however, this should not be an issue and the approach will have performance and learning advantages over alternatives such as raw network methods (see chapter 5.2.1 for a comparison).

4.3.3 Region Growing

The training process from the previous section produces outputs that fill in whole words or even lines of text as demonstrated by figure 9 (b). The edges of these regions approximate the boundaries of the text regions. To calculate these boundaries a region growing process is used.



(a) (b)

Figure 9: Input (a) and output (b) images of the trained network. The intensity of (b) is a measure of the certainty that it is text. Notice how the output image resembles figure 8 (b) in form.

By applying certain constraints to character dimensions we can use the boundary region grown to compute the orientation of the text and normalize its scale thereby making it suitable for OCR. These restrictions will also help in accepting and rejecting text and not-text regions respectively.

The chosen constraints are as follows:

- Text should be at least 8 pixels in height; smaller regions are rejected as candidates for OCR. Clearly OCR will fail on smaller characters as they are generally illegible.
- Words or lines should be three times as wide as they are high. This follows from the assumption that characters are generally lined up next to each other in rows, and this restriction is used to accurately calculate the direction of a given word or line.

- The angle of text in an image should not be rotated more then 90 degrees from the horizon. This limitation is actually just a convenience that avoids the issue of having to flip text that is reversed or mirrored.
- Azimuth (the viewing angle from normal) should be no more then approximately 45 degrees. This limit is somewhat arbitrary but greater angles cause excessive character distortion thereby potentially limiting OCR performance.

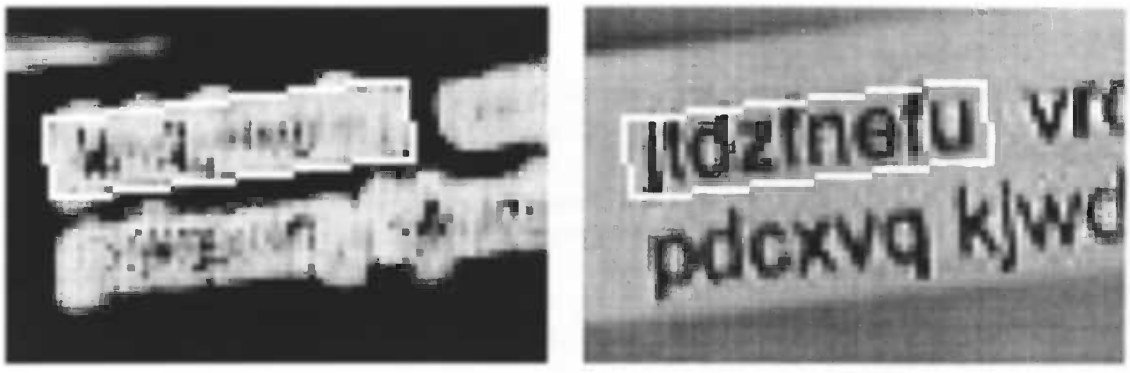
Note that only the width/height ratio constraint is seriously restrictive because it may reject isolated characters and short words. The remaining three constraints should simply be treated as cues for the navigational system to reposition itself.
To find the bounds of a region, a rectangular window is grown over the area until some threshold criteria is met. There are many ways of doing this; the computationally effective approach taken here is demonstrated by the pseudo-code in table 1.

Table 1: Pseudo-code that performs region growing.

```
        FOR each pixel position in the text detector output image
1)          IF the position is a text candidate AND it has not yet been processed
                Centre a small rectangle on the pixel location
                REPEAT
2)                  FOR each position in a local region
3)                      WHILE widening the rectangle increases the sum of text candidates in the rectangle
                            Increase the width of the rectangle
                        ENDWHILE
4)                      WHILE heightening the rectangle increases the sum of text candidates
                            Increase the height of the rectangle
                        ENDWHILE
5)                      FOR angles -μ/4 to μ/4
                            IF rotating the rectangle to this angle increases the sum of text candidates
                                Rotate the rectangle to this angle
                            ENDIF
                        ENDFOR
                    ENDFOR
6)              UNTIL the size AND orientation of the rectangle no longer increases
                Mark the rectangular area as having been processed
            ENDIF
        ENDFOR
```

The output image of the text detector is scanned for intensities above a certain threshold 1). At each candidate location the rectangle is shifted across a small window 2) to find an optimum central location. At each local position the rectangle attempts to grow first in width 3) and then in height 4), and then rotate clockwise and counter clockwise 5). Growth and/or rotation is permitted in a given direction if the sum of text candidates defined as Σ_{pixels} (Rectangle(pixels) – threshold), is greater after growth then that sum before growth (i.e. the sum of all thresholded pixel intensities in a rectangle must have increased). The whole process is repeated until no more growth takes place 6) after which the region is marked as processed and a new candidate is sought.

Once a set of boundaries has been found it is tested against the minimum height and width/height ratio constraints and if they are met the area is marked as containing text and excluded from further examination. Figure 10 demonstrates typical results of this region growing process.



(a) (b)

Figure 10: Outline of a grown rectangular region (a) superimposed on the original image (b).

Each rectangular region is then extracted, rotated, and resampled to a fixed height producing a normalized image suitable for OCR such as figure 11.

Figure 11: The result of rotating and re-sampling the region in figure 10 (b), the strip is now suitable for OCR processing.

Note that at this stage the image could also be examined to check for vertically mirrored text, multiple lines (resulting from poor separation between lines) and reverse video (light text on a dark background).

4.3.4 Optical Character Recognition

The results from the previous processing stage could simply be fed to a commercial OCR package for analysis but we have chosen instead to perform OCR ourselves for the sake of simplicity, integration, some flexibility, and controlled performance. OCR itself is not the focus of this study however and therefore rather severe constraints have been placed on the input images to simplify the architecture and the training process. These constraints are as follows:

- The input images should consist of dark text on a light background.
- All text should be of the Arial font.
- The characters must only consist of the lowercase letters 'a' to 'z'; the space between words is also recognized as a distinct character.
- In addition, the constraints from the previous two stages also apply.

OCR is performed by a multi layer feed forward network with a fixed 13x13 input window and trained using standard back-propagation. The window is shifted horizontally across the previously normalized input images, generating character candidates for each horizontal pixel location. These candidates are analysed during the following post processing stage. Although the OCR system implemented here is very restricted it is sufficient for experimentation in a controlled environment.

4.3.5 Post Processing

There are many ways of generating character sequences from the output of the OCR network, for example via linguistic constraining using Markov models [8] or Graph Transformer Networks [7], but as a proof-of-concept a process that simply selects the most active neurons across a local area is used (see section 5.3).
The post processing stage can consist of any number of additional steps, such as the spatial reconstruction of recognized words, or indeed using the results themselves for tasks such as navigation, learning and reading. An application demonstrating the use of this text recognition system in the field of augmented reality is outlined in appendix B.

4.3.6 Control Systems

The process described thus far involves a strictly serial flow of data. The model in figure 7 does however includes an additional unconnected module labelled "control systems" which covers camera control, navigation, planning, and a "*what and where*" subsystem. These control systems have yet to be implemented but form a natural extension to the work covered here. They can interact with each stage of the text reading system and exist both to make use of and to contribute to the OCR task. In terms of feedback the text detection module can directly inform the control systems about the possible presence of text. If there is interest from the planning or navigation systems then region growing can confirm the text presence and/or indicate to the navigation and camera system that repositioning is necessary, both in terms of distance and orientation. Lastly the OCR and post processing modules can provide the planning and navigation systems with the actually contents of detected text, such as room numbers. The "*what and where*" system involves an alternative design and this will be discussed in section 6.2.2.

5 Results

In this chapter experimental results for each of the three main processing stages will be presented. The focus is primarily on the text detection and region growing systems and their integration but overall performance including that of the character classifier will also be looked at and discussed.

5.1 Experimental Setup

Neural networks form the basic learning tool for the text recognition system previously described. They are all multi layer feed-forward networks trained using back-propagation; some details of the network design are given in appendix B. Neural networks must generally be trained with some care. Recognition rates bases purely on training data sets are not very meaningful for example, samples must be split into non-overlapping training sets and test sets. Empirical experimentation with neural networks shows that the error rates for test and training data differ according to the following approximation:

$$E_{test} - E_{train} = k\left(\frac{h}{P}\right)^{\alpha}$$

P is the number of training samples, h the complexity of the machine, α is some number between 0.5 and 1, and k is a constant. This approximation demonstrates that an increasingly complex system requires a larger number of training samples to maintain a low error difference, or conversely that a small error difference suggests a sufficient number of samples have been used for the given network complexity. It should in any case be kept in mind that in training a network, minimizing E_{train} is not sufficient; the gap must be minimized too[4].

To test and train the text detector 61 greyscale images were captured using a low cost Creative Video Blaster USB web camera, each at a resolution of 320 x 240 pixels. After training the text detector several of the resulting output images were then used to test and calibrate the region growing procedure. All 61 original images were then run through the text detecting and region growing modules, and the normalized outputs were then fed through a marking utility where characters were colour coded and saved to disk. This resulted in some 500 images containing 4000 characters which were then used to train and test the text classifier. Training times varied from about one minute for the text detector to half an hour for the character recognizer using standard desktop hardware.

5.2 System Results

Results from each of the three main modules are presented in the following sections.

[4] Figures 27 and 28 (appendix A) demonstrate the issues of training and testing error differences, and machine complexity.

5.2.1 Text Detection

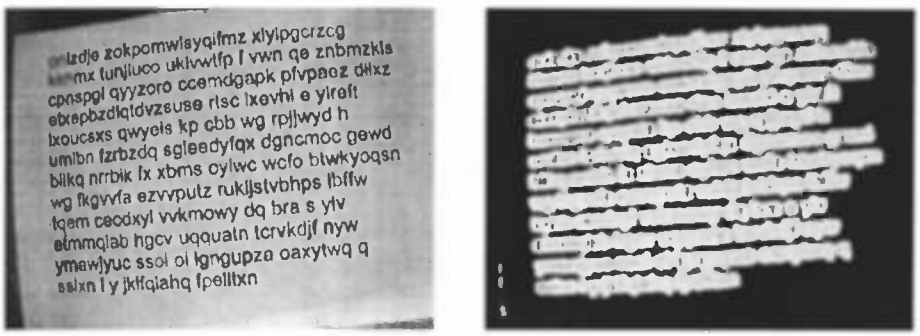
There are many different possible combinations of local measures that can be used for the purpose of text detection. To assess the impact of different measures 16 combinations were tested. The tests consisted of 31 training images containing a total of 105,000 background examples and 23,000 text examples, and 30 test images containing 90,000 background samples and 34,000 text samples. From the training set 50, 100, or 150 thousand background samples and an equal number of text samples were randomly selected and used to train a three layer feed-forward neural network using back-propagation. First the 5 measures presented in section 4.3.2 were tested followed by various combinations of these measures. For comparison, the same text detection task was also run using a fully connected feed forward network with a 7x7 input field on the raw image data. A neural net was also tested in combination with the 5 local measures. A final test involved shifting the input image so that its average intensity was 0.5, thereby easing network convergence. The results are displayed in table 2.

Table 2: A comparison of training and testing errors using various measure combinations.

Measure	Samples Trained	Training Error BG	Testing Error BG	Training Error Txt	Testing Error Txt
1 Local variance	100000	87.1	82.5	3.7	6.0
2 Edge density	100000	10.1	13.5	25.6	19.1
3 Δ Variance	50000	19.9	28.6	16.0	18.2
4 Angle symmetry	50000	29.0	40.1	10.8	11.2
5 Angle distribution	50000	11.7	19.7	9.0	6.6
1 & 2	50000	16.5	23.0	12.1	8.8
1 & 3	50000	18.7	27.5	13.3	14.2
1 & 4	50000	31.2	42.3	8.8	9.2
1 & 5	50000	10.5	17.3	13.3	14.2
1 & 2 & 5	50000	13.2	21.3	7.1	3.3
1 & 2 & 3 & 5	50000	12.4	19.9	8.0	4.0
1 & 2 & 3 & 4 & 5	50000	11.3	18.8	7.2	3.2
3 Layer NN (7x7 input)	150000	9.2	13.2	17.1	12.0
1 – 5 & NN (5x5 input)	50000	11.3	17.5	10.0	4.6
1 & 2 & 3 & 4 & 5 shift	50000	10.1	16.5	4.5	2.4

Localized measures have several advantages over fully connected neural networks, they run much faster because of their small size and easily optimized inputs, and they are able to train and adapt much more rapidly, at least within the dimensions covered by the chosen statistical measures. Determining the best combination of measures depends on the relative importance of background recognition rates as compared to text rates. Measures 1 and 5 for example produce fewer errors on background detection then the combination of measures 1 to 5, while the latter perform significantly better at correctly recognizing text. Conversely, measure 1 performs better at recognizing text then measures 1 to 5 combined but this is simple because it identifies everything as text. Clearly the overall goal is to minimize both text errors and background errors. A trade-off point will however have to be chosen between the acceptable number of false positives and false negatives and this point depends both on the application at hand and on the potential ability of the system to compensate for certain errors. Because the system can reject false positives (that is background that is identified as text) both at the region

growing phase and at the OCR phase, a lower text detection error rate is of more importance then low background error rates. The combination of measures 1 to 5 and a shift were therefore selected for the system because they offer the overall lowest error rates at an acceptable performance.
Figure 12 represents typical input and output patterns of the text detection system employing the chosen measures 1 to 5. The training method described in section 4.3.2 results in the filling of text regions forming almost rectangular stripes thereby facilitating the region growing process previously described.



(a) b)

Figure 12: The text detector produces strips marking lines of text.

Of course this image contains little other then text which makes those regions rather simple to identify. The image in figure 13 is rather more complex and demonstrates the degree to which false positives are generated; although figure 13 (a) contains no text regions, many areas are nonetheless identified as being text. A visual examination of the regions marked as containing text reveals that they are indeed text like however.

(a) (b)

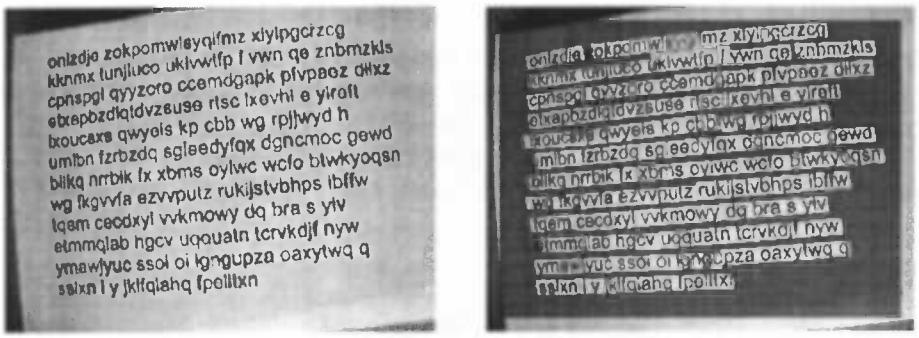
Figure 13: Most of image (a) has been marked as not containing text, there are however many small areas that appear to have text-like properties (b).

We will briefly return to these images in the next section to look at how the region growing process deals with false positives.

5.2.2 Region Growing

Region growing is performed as described in section 4.3.3. To analyze performance a subset of the available characters were hand counted and then processed by the text

detecting and region growing modules. The characters within the selected regions were then visually examined and rejected if illegible or incorrectly aligned. 1322 characters were counted in total of which 1267 were correctly detected and extracted by text detection and region growing. This suggests a bounding accuracy of approximately 96%. These results are of course made possible by the earlier constraints imposed on the colour, size and type of text, but the approximate 96% extraction rate is nevertheless very promising and should be sufficient for many tasks in a carefully prepared indoor environment. Figure 14 shows typical results of the region growing process.



(a) (b)

Figure 14: Input image (a) and the resulting regions that have been found (b).

The ability of region growing to reject false positives is demonstrated by processing the complex image from the last section, repeated below. Figure 15 (a) has been processed by the region growing module and the false text regions from figure 13 (b) have been rejected by the region growing constraints.

(a) (b)

Figure 15: None of the possible text regions from figure 13 (b) have met the requirements for the region growing procedure and so no text is detected.

5.2.3 Optical Character Recognition

The last of the three main processing stages is OCR. Text strips extracted in the last section are fed to a multilayer neural network trained using a stochastic gradient descent algorithm. The network consists of 169 input neurons, a single hidden layer with 48 neurons, and 28 output neurons. Of the output neurons 26 represent the letters 'a' to 'z',

one is for detecting spaces, and the other for detecting background (i.e. neither text nor spaces). Table 3 is an overview of the data used to train and test the OCR system.

Table 3: A summary of the data used to train and test the optical character recognition network.

	Images	Samples	Letter Samples	Average samples/letter
Total	508	31,534	3941	151
Training	308	20,605	2553	98
Testing	200	10,928	1388	53

Notice that the number of samples of each letter is actually quite small considering the enormous possible variety of text position, scale, contrast, and form[5]. Again, it is not the intention here to produces a state-of-the-art OCR system but rather to demonstrate an integrated proof of concept. During training the individual characters were manually centred in the horizontal OCR input field[6] producing the training and testing results plotted in figure 16.

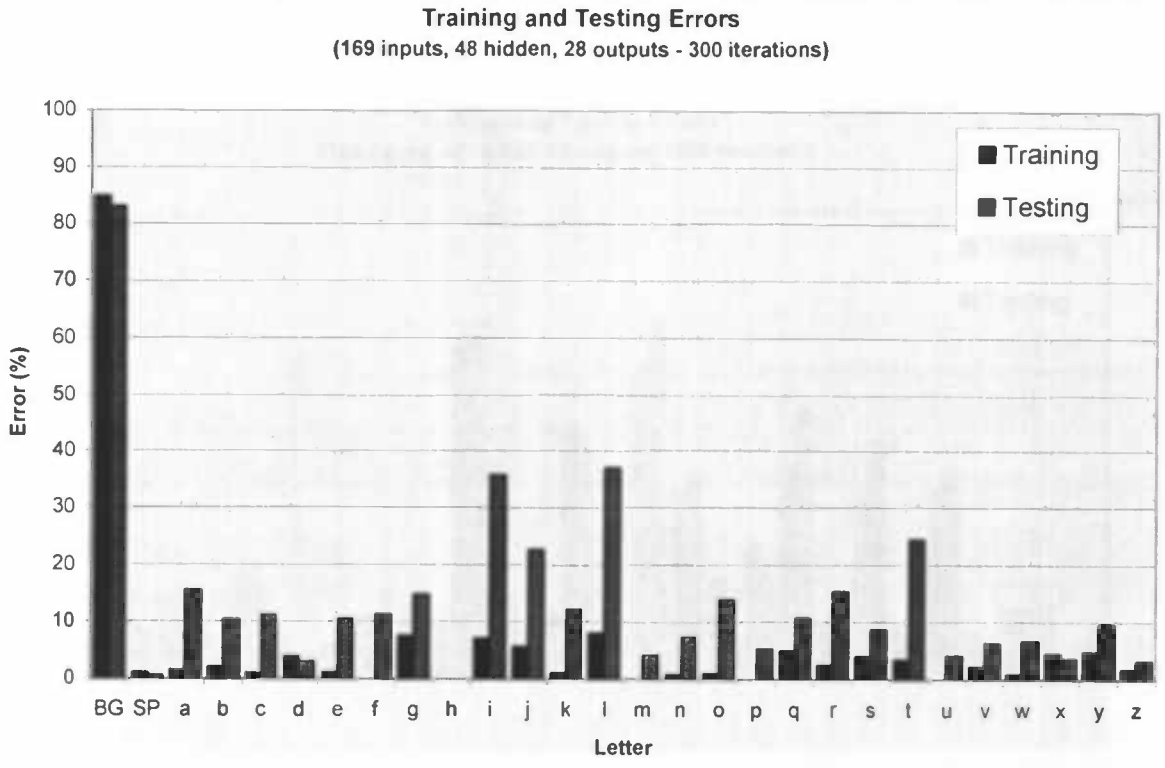


Figure 16: Error rates of different characters including spaces (SP), and background detection (BG) during training (blue) and testing (red).

[5] Figure 29 (appendix A) contains a collection of typical training patterns for the letter 'a'.
[6] Vertical centering is implicitly performed by region growing though more precise character based centering should be straightforward to implement and would reduce the OCR search space.

Excluding the background error, the average error rate is about 11% on the test data, ± 2% at a 99% confidence level and 2½ % on training data (± 1%). Most common errors result from the similar looking letters 'i' 'j' and 'l'. The difference between training and testing results indicates some degree of over fitting has taken place but a more significant problem is the poor performance of the background detector. This is hardly surprising considering that the background is defined as everything except horizontally centred text[7]. These false positives are generally eliminated by first segmenting individual characters and using these segments for OCR. There is then no need for a background detector. The goal here however is to make use of a convolutional network so that character segmentation would not be required, to do this the network was retrained without first horizontally aligning the training characters. The target for each training sample is then defined as the character nearest to the centre of the input field. With an average inter-character spacing of about 8 pixels this produced around eight times as many training samples for each letter, but eliminated the need for a background detector[8]. The testing and training errors of the different characters are plotted in figure 17. These are obviously significantly higher then in the previous case and this is largely due to the small size of the network and its inability to learn invariance with respect to input image position.

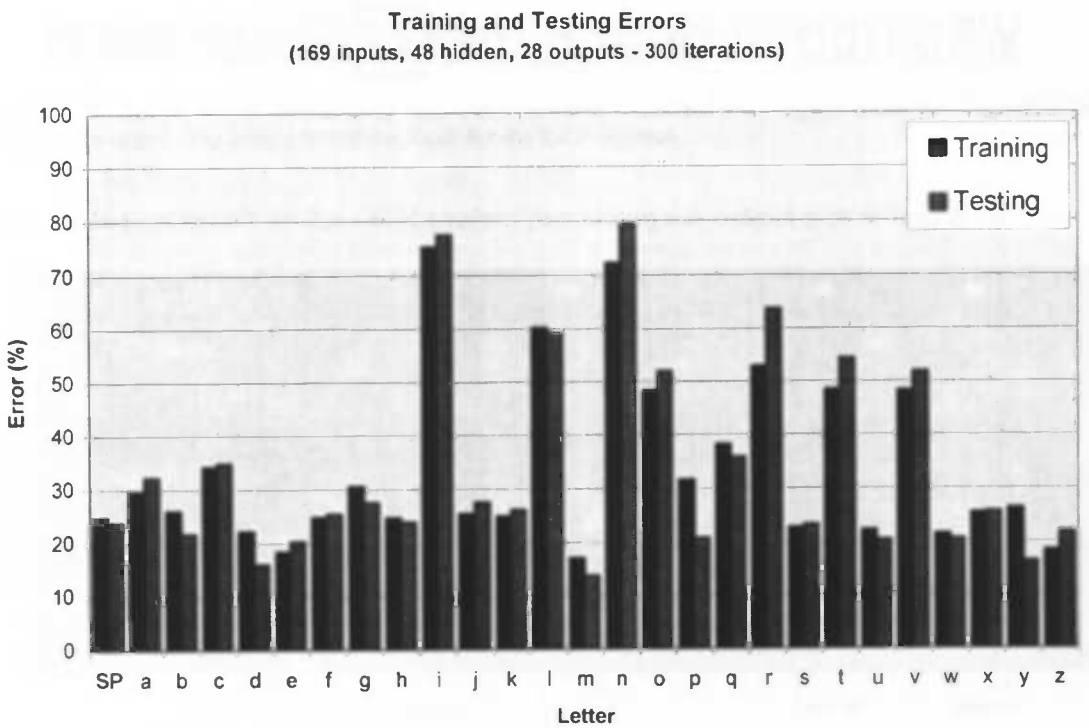


Figure 17: Error rates of different characters including spaces (SP) when no character segmentation is used.

[7] Figure 25 (appendix A) shows that the background detector is unable to generalize.
[8] This assumes that the input samples always contain text which while true for the training and test samples would generally not be the case.

The error rate is on average about 34% for both training and testing (+/- 3% at a 99% confidence level based on 1400 samples) indicating that the network has generalized well as a result of the much larger sample base but simply isn't powerful enough to learn the large and varied number of samples[9].

5.3 Overall Performance

One problem with using a scanning network to recognize text is interpreting the resulting output. Because likely text candidates are generated for each pixel location some form of post processing must be used to determine the most likely character. A very simple demonstration procedure is used here for that purpose and works as follows:
For every letter candidate the network responses of the previous and following two pixel locations are summed and the letter with the highest total response is selected. If the same character is selected for the next pixel locations it is ignored so as to prevent continues repetition. This has the side effect of occasionally rejecting legitimate letter repetitions but there are many other ways of dealing with this.
Figure 18 shows the output of the region growing process and the example will be used to demonstrate the character selection procedure.

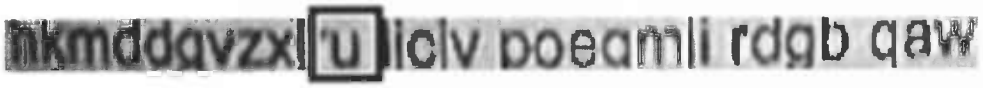


Figure 18: A line of text is selected by text detection and extracted with region growing. A window that scans linearly across the image forms the input for the OCR system.

Figure 18 is processed by the OCR system producing the output plot of figure 19.

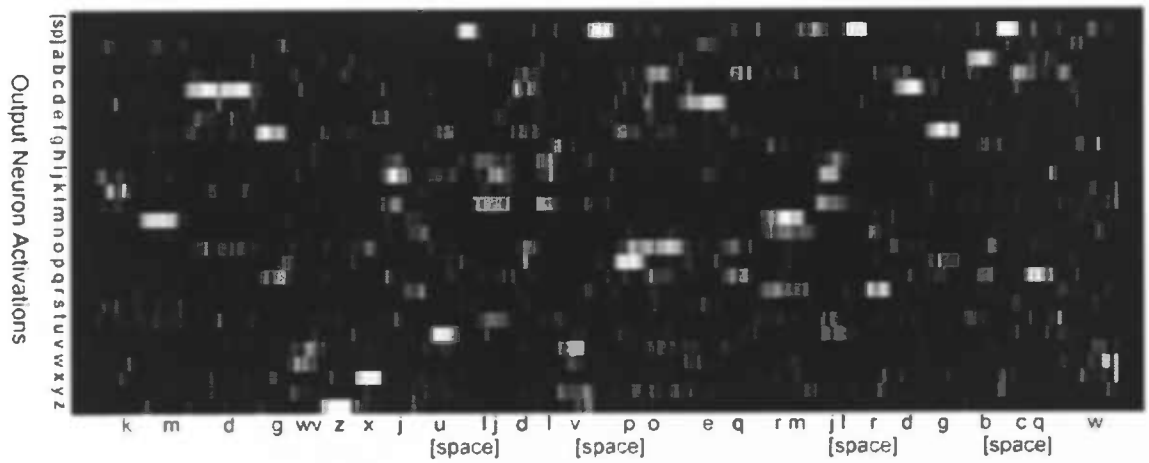


Figure 19: Plot of network response to figure 18. The image displays the activation of each output neuron (y-axis) to the input window of the corresponding position in figure 18 (x-axis). The most active neuron ID is displayed below the plot.

[9] Figure 26 (appendix A) shows the training progress of each character. See figure 30 for examples of typical training input patterns.

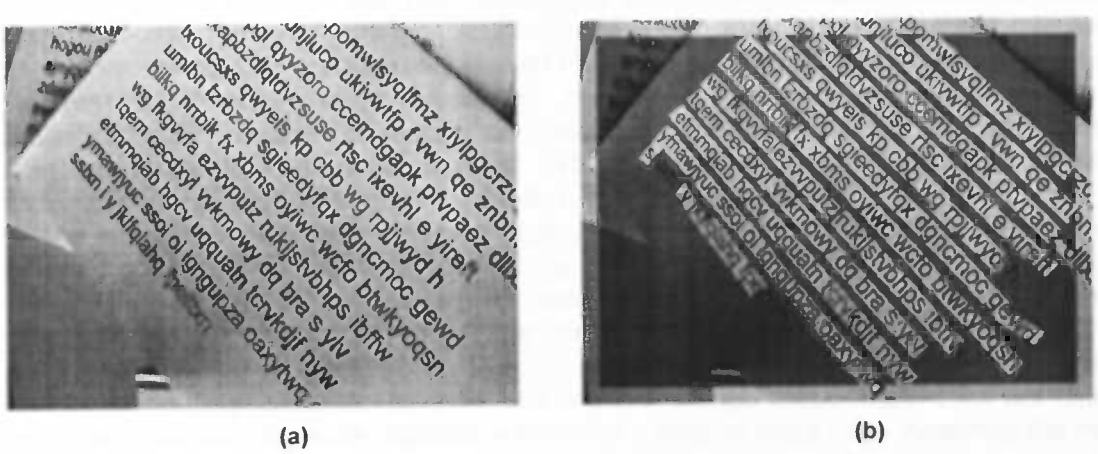
The letters beneath the plot are those selected by the post processor producing the following final output: **kmdgwvzxju ljdlv poeqrmjl rdgb cqw**. The first character is too close to the edge of the input image and is therefore skipped. The most common forms of error include the omission of repeating letters (double 'd') and the merging and splitting of letters (cl -> d, m -> rm) partly as a result of the simple output analysis procedure used. But it is the simple scanning network that clearly hampers recognition rates. The overall performance is fairly poor; approximately 60% of the characters from the original camera images are correctly identified. Given that 96% of the characters were correctly extracted the blame falls squarely on the OCR system. We will discuss a character segmentation technique in chapter 6 which should improve OCR performance significantly and bring overall recognition rates up to about 85%.

6 Conclusions and Recommendations

In this final chapter we will review the results of the previous sections and make some concluding remarks and recommendations. There is a great deal of follow-up work to be done and many possible improvements made so these too will be examined in some detail in this chapter.

6.1 Conclusions

The total recognition rate of the sample set is approximately 60% which is fairly low, though perhaps not unusually so for a camera-based reading system. This is however almost totally due to the poor performance of the simple OCR classifier which alone accounts for 97% of the errors. The text detection and bounding subsystems perform extremely well together, even on misaligned text as figure 20 demonstrates. Of course the system is subject to and trained to meet the constraints set in chapter 4, and performance using different letter styles and colours is therefore expected to be considerably poorer. On the other hand this system is able to deal with arbitrarily positioned and orientated characters with considerable scale variance. This is something most other systems can not do which seriously limits their applicability to the task of robot reading. Substituting the OCR module with a commercial OCR package would of course improve recognition rates dramatically but there is always a price to pay for such black-box solutions, indeed it is worth asking whether reducing 3D text to its 2D counterpart is really the way to go at all. In any case, most closed OCR packages assume the camera position is fixed which makes the promising possibilities of feedback systems harder to implement (see 6.2.2).



(a) (b)

Figure 20: The system is able to extract lines of text from any orientation.

Perhaps the most promising aspect of the system presented here is the demonstrated potential for real-time operation; the combination of a coerced text detector with a simple and efficient bounding procedure make practical implementation possible even on fairly modest hardware. Indeed a small mathematic system was implemented (described in appendix B) to demonstrate the systems applicability towards augmented reality, unoptimized it is able to process approximately 5 images per second on a desktop PC.

Some thoughts and details for further optimizations and improvements will be discussed in the next section but even in its current state the developed system should prove useful for robot navigation.

6.2 Suggestions for Further Work

This section covers a whole range of possibilities for further research. First several ideas are proposed for optimizing the existing system for real-time operation. Alternative text detection approaches are then briefly discussed and lastly a possible model for a "*what and where*" system is presented.

6.2.1 System Enhancements

We will first discuss possible enhancements for each of the individual steps presented in this paper followed by some general enhancements and extensions.
The text detector from chapter 4 was trained to recognize characters ranging from 4 to 30 pixels in height. A problem resulting from this large range is that recognition and rejection is compromised because different character sizes have fundamentally different local properties. One possibility that would allow for the detection of a larger range of text sizes would be to make use of pyramid input re-sampling. Input images would be scaled to various sizes and processed by the text detector. The text detector would be trained to respond to a very narrow range of scales and the normalized outputs of the various images would then be combined producing the final output. Addressing the detection of illegibly small text would require a different approach however, as super sampling cannot restore inter-pixel details. An interesting solution to this would be to use a second text detector, or indeed a whole series of text detectors, trained specifically for the purpose of detecting certain character sizes. This could come at very little additional processing cost because the computationally expensive localized measures could be reused. Some caution would be required however as the size of the areas over which the measures act can considerably impact the scale of features that they can detect.
Another enhancement is to adjust the measures used during processing. Lenient measures could be used to extract and normalize the orientation of text after which a new set of stricter measures, that for example assume text is horizontal, would be applied to more carefully segment individual lines and characters.
There are also a few fairly obvious improvements that could be made to the region growing module. The evaluation procedure itself is highly optimized using Bresenham's line drawing algorithm for high speed rectangular candidate counting, but the region growing procedure described in 4.3.3 is inefficient and could easily be improved. And there are other ways of extracting text boundaries too, such as extracting text strokes [20] or calculating centre of gravities and inertias; though these techniques are likely to be computationally far more expensive then the methods used here. Adapting the region growing algorithm to support non-parallel top- and baselines as demonstrated by figure 21 should be quite straightforward and would both improve the normalized scale of characters for OCR and give the system an indication of the orientation of the text plane in relation to the viewer which would prove useful for repositioning the camera.

Figure 21: Weakening the restriction of the region growing module from a rectangle to a quadrangle could allow the system to make use of top- and baselines to estimate the orientation of text planes.

During the third classification step the individual characters are recognized by a neural network scanning across the words or lines. This scanning process avoids the need for character segmentation but results in a recognition rate of only 63%. This method was chosen as a precursor to a full scale LeNet [7] type network which can operate without segmentation, but such a network is both complex and monolithic. The simple OCR network trained on aligned characters has a recognition rate of about 89% which indicates that effective character segmentation would dramatically improve classification results while at the same time increasing performance by not having to run the network at every pixel location. Character segmentation techniques are usually based on vertical projection profiles. They are subject to frequent errors because of the difficulty in distinguishing spaces from local in-character projection minimums. This can usually be resolved by offering the OCR systems multiple hypothesises to choose from. The problem is significantly more difficult however when characters touch or by design cannot be vertically split as may be the case with *italic* text. Figure 22 show a typical text section extracted by region growing and its projection profile. The characters are extremely difficult to segment because they touch which makes their projections difficult to interpret.

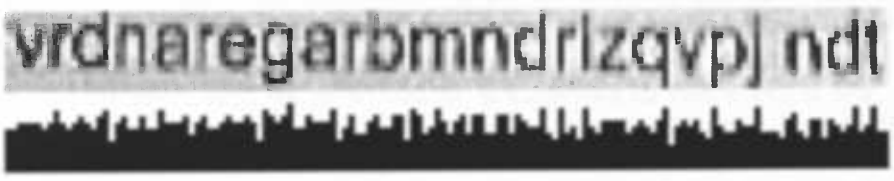


Figure 22: Low resolution characters and their vertical projection profile. Note that many characters (such as the first three) are impossible to segment using projection profiles alone.

To address this problem a neural network is trained to distinguish between spaces and characters. The network input window scans across the text image marking each horizontal position as text or space. The segmentation results of the example in figure 22 are displayed in figure 23. The network is trained on a very limited sample set but demonstrates its potential ability and usefulness in segmenting characters.

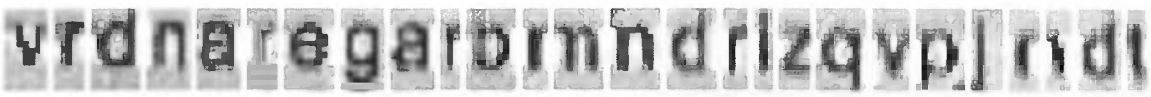


Figure 23: The same low resolution characters as from figure 22 are segmented by a neural network trained to recognize spaces. Several errors are still made such as 'm' being segmented into an 'r' and an 'n' but the overall alternative character hypothesis space is very small.

Post processing is another area open to a great deal of improvements. These range from checking outputs against a simple dictionary, to full scale syntax and semantic analysis. Markov Models and Graph Transformer Networks can in addition take advantage of probability analysis and language redundancy to reconstruct likely outputs. An in-depth discussion of such systems is beyond the scope of this paper.
One basic technique that would benefit overall system performance would be to make use of multiple frames through the mobility of the camera and robot. The camera is able to reposition itself or zoom in on uncertain characters and this behaviour should be considered as part of the reading system. This would improve recognition rates and reduce the necessary processing costs by introducing the concept of selective attention. This also raises some new questions concerning the general interaction of the text recognizer with other systems which will need answered. How for example should camera zooming and robot repositioning be coupled or integrated into the reading system? A possible model for this will be discussed in the next section.
We will briefly return to the discussion of convolutional networks. The current system is serial which has the advantage of breaking down and constraining the flow of data but there are several reasons why an integrated approach would be preferred. The hand crafted nature of first detecting text regions, and then the text itself, is prone to possibly generating errors as a result of poorly chosen features. Integrating the locator and the recognizer into one globally trainable system might allow it to find more optimal solutions and be more robust. But what might an integrated approach look like? The suggestion of running a powerful and fully trainable system such as LeNet-5 [7] across the entire input image would be an example of such an approach, though it is likely to be prohibitively slow for real-time operation. One might attempt to resolve this by combining the network with a much simpler early rejection system thereby reducing the computation to likely candidates. Another alternative might be a hand crafted neural network initialized to perform the three steps as presented in chapter 4 but that is then free to adjust its parameters through training.
One final pragmatic suggestion for improving performance is to make use of graphics hardware acceleration. Modern graphic processors are capable of high speed filtering, convolutions, general image manipulation tasks such as re-sampling, rotation, subtraction and addition, and dynamic colour adjustment to name but a few. Their use has been demonstrated for the accelerated calculation of 3D convolutions [10] and for performing wavelet transformations [11].

6.2.2 "What and Where" Model

In this final section we return to the biologically inspired "*what and where*" system as discussed at the beginning of chapter 2. The feedback module of figure 7 refers to such a system grouped under the collection of control systems but its form or function has not been discussed. The "*what and where*" model of chapter 2 is different from the camera-based OCR systems discussed thus far, and indeed from the model of figure 7, because the questions "*what*" and "*where*" are in part answered by the interaction of dynamic visual system. Designing such a system simply because that is the way human vision

works is in practice not such a good idea, what is efficient and effective for a massively parallel biological system may not be so for a low powered serial computer. Some concepts from the human vision system are however applicable in that they could possibly ease the necessary processing requirements and at the same time facilitate a more active form of vision. The model depicted in figure 24 is a possible outline for the design of such a system.

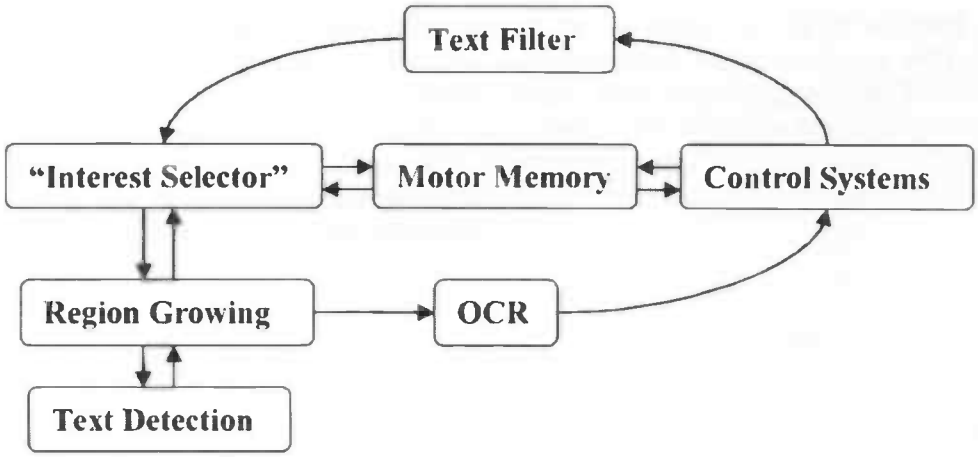


Figure 24: Model of an active vision system.

The system requires an active camera and is designed to fixate and move between areas of interest which in this case are text regions. The process starts at the Test Filter module which is a high speed but very basic variant of the text detector of section 4.3.2, perhaps using only one local measure. Its purpose is simply to indicate regions that may contain text. A number of these regions are selected by an "Interest Selector" (perhaps by selecting most active regions of a sub-sampled image) and their relative vector positions are fed to the Motor Memory module. The area of greatest interest is passed through the Motor Memory on to the Control Systems which update the camera position in an attempt to centre the candidate text area in the visual field. This in turn updates the Motor Memory, and feedback through the Text Filter is used to calibrate and correct these saccade motions. Region Growing then starts in the central area of the visual field. The quality of the output from the Text Filter module is not sufficient however to correctly indicate the boundaries and orientation of lines of text, so the Region Growing module performs text detection on the area within its rectangle while it is growing. This is done in the same manner as in section 4.3.2 but because this only needs to take place locally overall system performance is dramatically improved. If text regions are found then OCR is performed and the results are passed to the Control Systems. If no regions are found then control returns to the "Interest Selector" and an alternative region is selected.
The role of the Motor Memory is twofold. First it map distances in the visual field to physical motions allowing the system to fixate as described above. Secondly it keeps track of the approximate location of interesting areas that may have fallen out of the visual field due to rotation or motion. This is important because in practice the system must constantly move towards or zoom in to regions of interest due to the low resolution of the camera system, and so a general awareness of the location of other candidates is critical to achieving real-time performance. Note that the vectors in the Motor Memory module are stored in terms of motor movements and that their design must facilitate learning and forgetting to deal with continually changing environments and uncertainties.

A slight variation of this design which more closely resembles the human visual system would involve a ring of increasingly powerful filters when moving towards the centre of the visual field. The outer ring would be similar to the Text Filter module of figure 24 and the central ring being functionally equivalent to the Text Detection module. Centring a region of interest in the visual field of the camera would then automatically reveal the orientation and scale of the text.

Clearly there is still a great deal of research to be done on camera-based optical character recognition. The system that has currently been implemented is very general in that it can be applied to any form of digital media, from web images to MPEG film. The feedback model of figure 24 on the other hand makes use of active camera systems and this introduces some new problems but also some very exciting possibilities worth investigating. For now we hope that the work done here has demonstrated the feasibility of real-time camera-based reading systems.

References

[1] M. Black, F. Brard, A. Jepson, W. Newman, E. Saund, G. Socher, and M. Taylor, *The Digital Office: Overview*, In Proceedings of the AAAI Spring Symposium on Intelligent Environments, 1998.

[2] P. Clark and M. Mirmehdi, *Finding text regions using localised measures*, In Proceedings of the 11th British Machine Vision Conference, pages 675--684, 2000.

[3] P. Clark and M. Mirmehdi, *Estimating the Orientation and Recovery of Text Planes in a Single Image,* In Proceedings of the 12th British Machine Vision Conference, pages 421--430, 2001.

[4] G. McConkie and K. Rayner, *Asymmetry of the perceptual span in reading,* Bulletin of the Psychonomic Society, 8, 365-368, 1976.

[5] Y. LeCun and Y. Bengio, *Convolutional Networks for Images, Speech, and Time-Series,* In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, MIT Press, 1995

[6] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. D. Jackel and H. Baird, *Constrained neural network for unconstrained handwritten digit recognition,* First International Workshop on Frontiers in Handwriting Recognition, April 1990.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, Proceedings of the IEEE, 86(11), pp.2278 - 2324.

[8] I. Guyon and F. Pereira, *Design of a Linguistic Postprocessor using Variable Memory Length Markov Models,* IEEE International Conference on Document Analysis and Recognition, pp. 454--457, 1995

[9] S. Haykin, *Neural Networks – a comprehensive foundation*, 1999 Prentice-Hall Inc.

[10] M. Hopf and T. Ertl, *Accelerating 3D convolution using graphics hardware,* IEEE Visualization '99, pages 471-474, San Francisco, 1999.

[11] M. Hopf and T. Ertl, *Hardware Based Wavelet Transformations,* Erlangen '99 Workshop on Vision, Modeling and Visualization, November 1999.

[12] T. Igarashi, K. Edwards, A. LaMarca, E. Mynatt, *An Architecture for Pen-based Interaction on Electronic Whiteboards*, AVI 2000, ACM Press, Palermo (Italy), pp.68 – 75, May. 2000.

[13] H. Li, D. Doermann, and O. Kia, *Automatic text detection and tracking in digital video*, IEEE Transactions on Image Processing, vol. 9, pp. 147–156, Jan. 2000.

[14] R. Lienhart, and A. Wernicke, *Localizing and Segmenting Text in Images and Videos*, IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 4, April 2002.

[15] T. Mitchell, *Machine Learning*, 1997 McGraw-Hill Companies Inc.

[16] G. K. Myers, R. C. Bolles, Q. Luong, J. A. Herson, *Recognition of Text in 3-D Scenes*, Fourth Symposium on Document Image Understanding Technology, Columbia, Maryland 2001.

[17] I. A. Rybak, V. I. Gusakova, A. V. Golovan, L. N. Podladchikova, and N. A. Shevtsova, *A model of attention-guided visual perception and recognition,* Vision Research 38 (2), pp. 387-400, 1998.

[18] T. Sato, T. Kanade, E. K. Hughes, M. A. Smith, and Shin-ichi Satoh, *Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Caption,* Submitted to ACM Multimedia Systems Special Issue on Video Libraries, 1998.

[19] R. Vaillant, C. Monrocq, and Y. Le Cun. *Original approach for the localisation of objects in images*, IEEE Proceedings on Vision, Image, and Signal Processing, 141(4), August 1994.

[20] V. Wu, R. Manmatha, and E. M. Riseman, *Automatic Text Detection and Recognition*, in Proceedings of the Image Understanding Workshop, pp. 707-712, 1997.

[21] J. Yang, J. Gao, Y. Zang, X. Chen, and A. Waibel, *An Automatic Sign Recognition and Translation System*, Proceedings of the Workshop on Perceptive User Interfaces (PUI'01), November, 2001.

Appendix A - Images

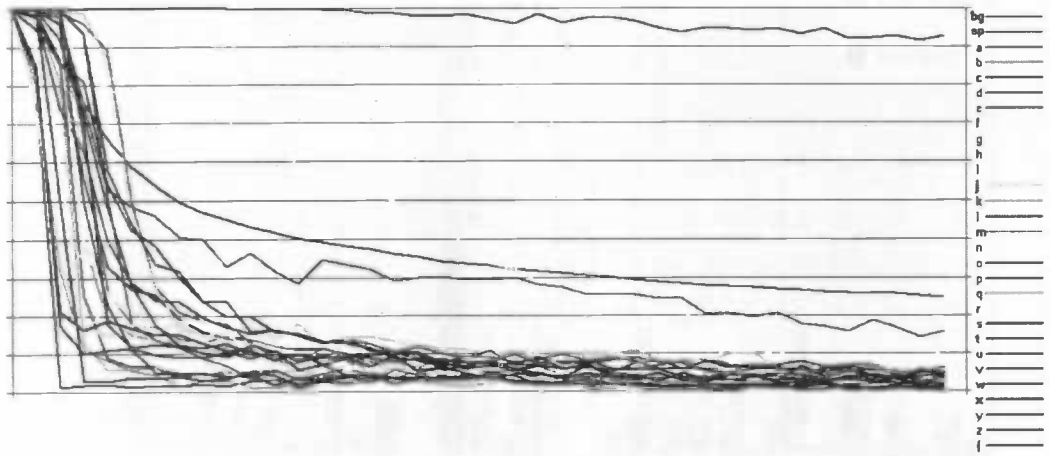


Figure 25: OCR network training over approximately 300 epochs. The y-axis indicates the percentage of incorrect network responses to a given character, where a response is represented by the most active output neuron. Characters are horizontally centred on the input window of the neural network. The smooth line represents the average error rate. Notice that the background detector fails to generalize. The two characters that perform poorly (the white and grey lines) are the letters 'l' and 'i' which look very similar.

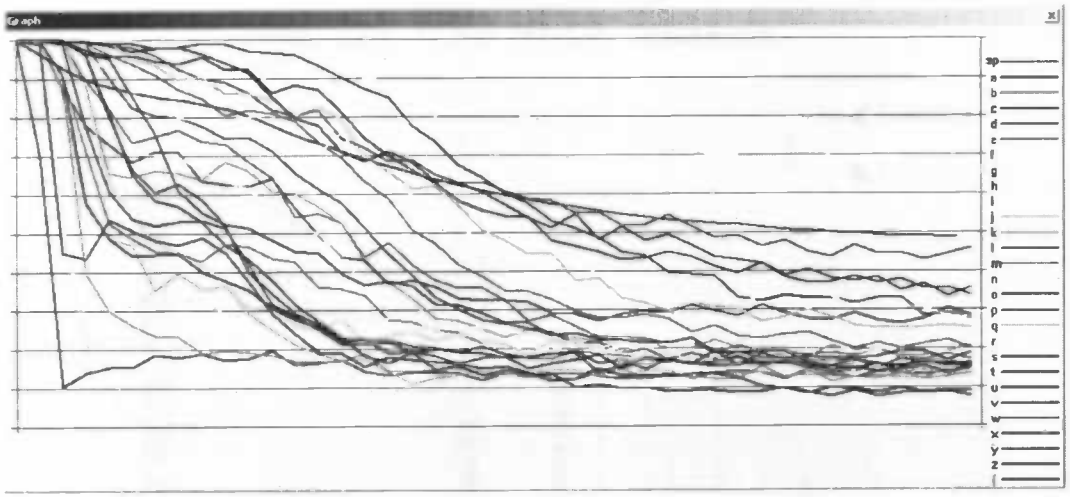


Figure 26: OCR network training over approximately 300 epochs. The characters here are not horizontally aligned so the network must become position invariant to generalize effectively.

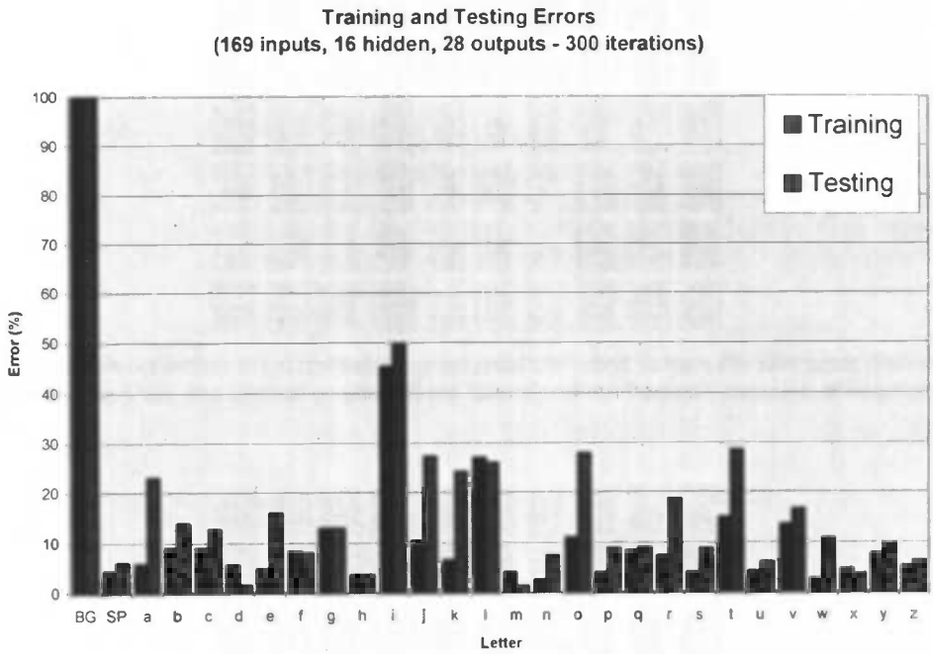


Figure 27: Error levels when trained with a network containing only 16 hidden layer. Training and testing error rates are similar indicating that the network has generalized fairly well. The relatively poor performance suggests that the network is too small.

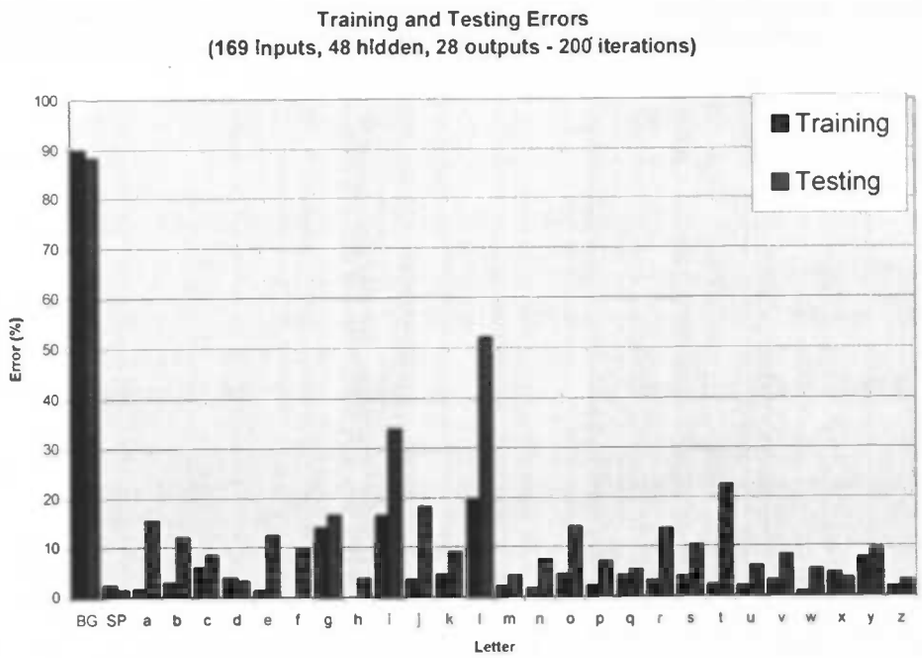


Figure 28: Error levels with a network containing 48 hidden neurons. The fairly large difference between training and testing errors indicates that more training data and/or additional iterations are required.

Figure 29: A sample collection of horizontally aligned letters 'a' used to train the character classifier. Notice the large variation in scale and vertical position of the letters and the frequent inclusion of fragments of other characters.

Figure 30: A sample collection of unaligned letters 'a' used to train the character classifier. In addition to the variation in scale and vertical position there is now a large variation in horizontal position.

Appendix B – Implementation Details

Implementation details

The system has been designed from the start to be reused. Completely written in C, it consists of a general purpose neural network implementation, basic XML handling for data transfer and storage, various image processing and I/O functions as well as Windows platform specific visualizing tools and TWAIN video capture functions.
The neural network code performs its own process housekeeping thereby supporting any feed-forward network structure without the limitation of explicit layers. Network size is only limited by available memory. The network uses a sigmoid activation function by default but others are dynamically supported. Training rates and momentum can also be freely selected and adjusted. The XML parser is used to load and save network structures with their weights and biases and to organized text and training data and output. Image processing functions include amongst other the following:

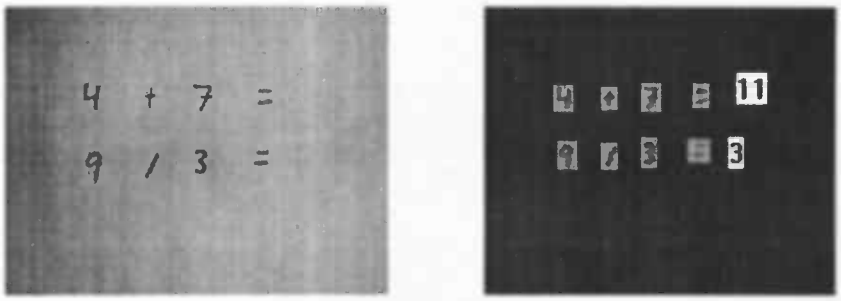
- Image memory management
- resampling
- cropping
- rotation
- various filters (smoothing, equalizing etc)

The I/O functions support the reading and writing of ASCII and binary PGM and PPM formats and the reading of uncompressed and run length encoded TGA image formats.

Example Application

To demonstrate the basic concepts of the design and its real-time potential a demonstration system was set up that recognizes handwritten numbers and mathematical operators from camera images, and solves (simple) written equations. The system would typically be integrated in an augmented reality setup where a users wears a pair of glasses with a projection system and miniature camera. The system is a striped version of the one described in this paper; it uses only three localized measures, assumes all characters are well spaced, and doesn't attempt to correct rotations (a consequence of the spacing as the 'width/height' constraint is then not met). Examples of handwritten numbers 0 – 9, operators '+',' –', '/', '*', and the '=' are trained for 300 iterations with a three layer neural network. The network is able to generalize well because the individual characters are well bounded due to the spacing.
The system scans from left to right through the input video frames storing recognized numbers and operators that are approximately aligned, and performing the written computations where possible. When a '=' character is recognized the systems completes the computation and projects the answer to the right of the equals sign. An example of a typical input and output frame is given in figure 31. The system runs at approximately 5 frames per second on mid-range PC hardware at a video resolution of 176x144 pixels. The application has a lot of potential but there are several hurdles that would have to be overcome for practical use to be possible. The resolution for example is too low for text

to be identified from a typical reading distance and active camera systems such as proposed in section 6.2.2 are not a viable option.



(a) (b)

Figure 31: Original input image (a) and the bounded output with the results of the equation overlaid onto the image (b).