A THEORY
OF DOCUMENT OBJECT LOCATOR COMBINATION

by

Jung Soh

A dissertation
submitted to the Faculty of the Graduate School
of State University of New York at Buffalo
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

June 1998

Dissertation Committee

Dr. Sargur N. Srihari (Chairman)

SUNY Distinguished Professor, Department of Computer Science

Director, Center of Excellence for Document Analysis and Recognition (CEDAR)

Dr. Venugopal Govindaraju

Research Associate Professor, Department of Computer Science

Associate Director, CEDAR

Dr. Zhongfei Zhang

Research Assistant Professor, Department of Computer Science

Research Scientist, CEDAR

Outside Reader

Dr. David Doermann

Co-Director, Language and Media Processing Laboratory

Center for Automation Research

University of Maryland at College Park

This dissertation was defended on March 25, 1998.

To my parents,

Kyung-Boon Min and Doo-Young Soh,

who made this possible

Abstract

Traditional approaches to document object location use a single locator that is expected to locate as many instances of the object class of interest as possible. However, if the class includes subclasses with diverse visual characteristics or is not characterized by easily computable visual features, it is difficult for the single locator to account for wide variation in object characteristics within the class. As a result, increasingly complex models of objects to be located are used.

An alternative approach is to combine the decisions of multiple locators, each of which is suitable for certain image conditions. This approach utilizes a collection of simple locators that complement one another, rather than relying on one complex locator. An effective method for combining the location results is vital to the success of this approach. This thesis presents a theory of combining the results of multiple document object locators tuned to different object characteristics. The purpose of the combination is to achieve integrated location performance which is better than that of any individual locator.

Location results are represented by objective confidence values, regardless of their types. This allows for combining locators at any output information level. Since different locators use different segmentation algorithms, multiple decisions on a single object are not readily available. Object equivalence and correspondence are used to determine what decisions to combine. The highest confidence, confidence summation, weighted confidence summation, probabilistic confidence estimation, and fuzzy integral methods are proposed for combining confidence values. A method for dynamic selection of locators based on training image set partitioning is proposed. Differences in locator performance levels are represented by locator relevance values, which are used in a combination method and dynamic selection. Measures for evaluating combination performance are proposed and compared.

The theory is applied to the address block location problem. A multiple locator algorithm employing a machine-printed address block locator, a handwritten address block locator, and a clustering-based locator is created and tested on a database of 1,100 mail piece images, in two separate experiments. The experimental results show that combining location results using the theory provides location performance that excels any individual locator performance, and therefore is a viable approach to document object location problems.

Acknowledgements

I would like to express my gratitude to the numerous people who in one way or another helped me complete this dissertation.

Dr. Sargur N. Srihari, my advisor, gave me the challenge of this research and the freedom of my own thinking. His thoughtful and effective advice enabled me to complete this dissertation successfully. I am grateful for his guidance and support.

Dr. Venugopal Govindaraju and Dr. Zhongfei Zhang were members of my dissertation committee. I benefited from their expertise and the valuable comments they offered me.

Dr. David Doermann of the University of Maryland was my outside reader. He carefully reviewed this dissertation and gave me useful feedback.

Dr. Jonathan J. Hull and Dr. David B. Sher were former members of my dissertation committee. Their points of view helped me develop several ideas in this research. Dr. Stuart C. Shapiro and Dr. William J. Rapaport provided me with timely advice on my Ph.D. program as former Directors of Graduate Studies.

Paul W. Palumbo co-directed the real-time address block location (RT-ABL) project of CEDAR, from which this research was initiated. He provided an open and friendly research environment and I enjoyed working with him. Dr. Yong-Chul Shin made several suggestions for improving this dissertation. Other members of the RT-ABL team also contributed to this dissertation through the many ideas and discussions we had.

Eloise Benzel, graduate secretary of the Computer Science Department, kindly took care of many student affairs for me while I was off campus. Gloria Koontz and Sally Elder, department secretaries, were always friendly and helpful. Eugenia H. Smith, secretary of CEDAR, was a great help with various things around my dissertation defense time.

The staff of Seoul Mail Center generously let me use a portion of their mail piece collection in my experiments. Many researchers of the Image Processing Division of Systems Engineering Research Institute showed interest and encouraged me while I worked on this dissertation. Min Wang, Suck-Joo Kang, and Jae-Woo Kim were especially friendly and cheering.

My parents gave me the opportunity of this higher education and stood by me all the time. My brother and sister, Yeong and Yoon-Kyung, always encouraged me during my graduate study. Other family members, especially my mother-in-law and father-in-law, warmly supported me in many ways. My son and daughter, Joon-Won and Hye-Yun, brought me the deep joy I have never felt before. My wife, Sung-Hye, deserves special thanks for being my classmate, colleague, critic, friend, and lover.

Contents

List of Tables **xii**

List of Figures **xiv**

1 Introduction **1**

1.1 Definitions 2

1.2 Motivation 3

1.3 Overview of Main Results 5

1.3.1 Theory 6

1.3.2 Experiments 8

1.4 Dissertation Outline 9

2 Literature Review **11**

2.1 Combination Approaches in Related Areas 12

2.1.1 Combining Document Object Locators 12

2.1.2 Combining Segmentation Methods 13

2.1.3 Classifier Combination 17

2.1.4 Integrating Visual Modules 26

2.2 Approaches to Address Block Location 28

2.2.1 Knowledge-Based Systems 28

2.2.2 Pattern Recognition 31

2.2.3 Geometric Feature-Based Approaches 31

2.2.4 Texture Analysis 32
2.2.5 Address Syntax Analysis 33

3 A Theory of Document Object Locator Combination **35**
3.1 Problem Statement 35
3.2 Formulation 37
3.3 Objective Confidence 44
3.3.1 Measurement Level 45
3.3.2 Rank Level 52
3.3.3 Abstract Level 54
3.4 Object Equivalence and Correspondence 55
3.4.1 Deciding Equivalence 57
3.4.2 Measuring Correspondence 58
3.4.3 Deciding Correspondence 61
3.5 Locator Relevance 63
3.5.1 Static Relevance 65
3.5.2 Dynamic Relevance 66
3.5.3 Dynamic Selection of Locators 69
3.6 Confidence Value Combination 70
3.6.1 Highest Confidence 71
3.6.2 Confidence Summation 72
3.6.3 Weighted Confidence Summation 73
3.6.4 Probabilistic Confidence Estimation 74
3.6.5 Fuzzy Integral 77
3.6.6 Summary of Combination Methods 79
3.7 Performance Evaluation 80

4 A Multiple Locator Algorithm for Address Block Location **84**
4.1 Address Block Location Problem 84

4.2 Multiple Locator Algorithm 87
4.2.1 Algorithm Description 87
4.2.2 Comparison with Previous Approaches 89
4.3 Machine-Printed Address Block Locator 90
4.3.1 Line Segmentation 92
4.3.2 Block Segmentation 95
4.3.3 Block Evaluation 96
4.4 Handwritten Address Block Locator 98
4.4.1 Preliminary Line Segmentation 101
4.4.2 Line Segmentation 105
4.4.3 Block Segmentation 109
4.5 Clustering-Based Locator 112
4.5.1 Block Segmentation 113
4.5.2 Block Evaluation 120

5 Address Block Location Experiments **122**

5.1 Mail Piece Image Preprocessing 122
5.2 Mail Piece Image Database 123
5.3 Training Results 126
5.3.1 Objective Confidence 126
5.3.2 Correspondence Thresholds 127
5.3.3 Dynamic Locator Relevance 129
5.3.4 Functions for Probabilistic Confidence Estimation 129
5.4 Test Results and Analysis 132

6 Conclusions **138**

6.1 Research Conclusions 138
6.2 Future Research 140

A Mathematical Details **142**

A.1 Estimating Regression Coefficients 142
A.1.1 Least Squares 142
A.1.2 Weighted Least Squares 144
A.2 Fuzzy Measures and Fuzzy Integrals 145
A.3 Minimizing Within-Group Variance 148

B Examples of Address Block Locator Combination 151

References 158

List of Tables

3.1 Example of data for computing objective confidence from measurements by linear regression 49

3.2 Example data for computing objective confidence from measurements by logistic regression 51

3.3 Example of computing objective confidence from ranks 54

3.4 Example of correspondence decisions 64

3.5 Possible agreement classes for three locators 68

3.6 Example of data for probabilistic confidence estimation 76

3.7 Example of confidence and relevance values 79

3.8 Requirements of confidence combination methods 80

4.1 Degrees of membership for the blocks in Figure 4.10 98

4.2 Block hypothesis generation 111

5.1 Composition of the mail piece image database 125

5.2 Image sets in two experiments 125

5.3 Objective confidence values determined by training 126

5.4 Training data for computing objective confidence for ML by logistic regression 127

5.5 Correspondence threshold values determined by training 128

5.6 Dynamic relevance values determined by training 130

5.7 Functions for probabilistic confidence estimation determined by training 131

5.8 Test results in terms of location performance 133

5.9 Test results in terms of combination performance 134

5.10 Agreement class decision results 136

5.11 Combination performance of top choices separated by agreement class 136

A.1 Example of measurements and fuzzy densities 148

List of Figures

1.1 Role of document object locator combination 2

1.2 Address block location input and output 9

3.1 Components of proposed theory 39

3.2 Examples of equivalent objects and corresponding objects 42

3.3 Plot of data in Table 3.1 and the fitted linear regression function . . . 50

3.4 Example of logistic response functions 50

3.5 Plot of data in Table 3.2 and the fitted logistic regression function . . 52

3.6 Difference between classifier combination and locator combination . . 56

3.7 Examples of overlapping objects 60

3.8 Example of computing the correspondence degree 61

3.9 Computing and using dynamic relevance 67

3.10 Evaluating combination performance 83

4.1 Types of address block location outputs 85

4.2 Multiple locator algorithm for address block location 88

4.3 Machine-printed address block locator 91

4.4 Input connected components for the machine-printed address block locator 92

4.5 Merging a connected component to a line 93

4.6 Initial line segmentation result 94

4.7 Postprocessed line segmentation result 94

4.8 Merging a line to a block 96
4.9 Functions to compute the degree of membership in the destination address class 99
4.10 Output of the machine-printed address block locator 100
4.11 Handwritten address block locator 101
4.12 Input connected components for the handwritten address block locator 102
4.13 Area-filled connected components 103
4.14 Horizontal projection profile 103
4.15 Preliminary line hypothesis generation and testing algorithm 106
4.16 Searching for the cut point 107
4.17 Preliminary line segmentation result 107
4.18 Vertical projection profiles of preliminary lines 108
4.19 Line segmentation result 109
4.20 Typical incorrect block hypotheses 112
4.21 Output of the handwritten address block locator 112
4.22 Clustering-based locator 113
4.23 Input connected components for the clustering-based locator 115
4.24 Connected components represented as points 115
4.25 Six clusters during block segmentation 118
4.26 Five clusters during block segmentation 118
4.27 Four clusters during block segmentation 119
4.28 Three clusters during block segmentation and also the output of the clustering-based locator 119
4.29 Two clusters during block segmentation 119
4.30 Measuring the city-block distance to the reference point 121

5.1 Mail piece image binarization 124
5.2 Connected component detection result 124
5.3 Plot of training data and objective confidence functions for ML . . . 128

B.1 Combination example 1 152
B.2 Combination example 2 153
B.3 Combination example 3 154
B.4 Combination example 4 155
B.5 Combination example 5 156
B.6 Combination example 6 157

Chapter 1

Introduction

Document object location, visually finding the object in the class of interest in a document image, is an important first step in many document image analysis tasks. Traditional approaches to this problem use a single location method that is expected to locate all or as many as possible instances of the object class of interest. However, if the class includes subclasses with diverse visual characteristics or is not characterized by easily computable visual features, it is difficult for the single locator to account for wide variation in object characteristics within the class. As a result, increasingly complex models of objects to be located are used.

An alternative approach is to combine decisions of multiple locators, each of which is suitable for certain image conditions. This approach utilizes a collection of simple locators that can complement one another, rather than relying on one complex locator. An effective method for combining the location results is vital for this approach to be successful. In problems like character recognition, word recognition, and image segmentation, approaches based on the same ideas as this one have been used in various forms. Nevertheless, integration of document object location results has not been studied systematically.

This thesis is concerned with a theory of combining the results of multiple document object locators, where each is tuned to different object characteristics. Fig-

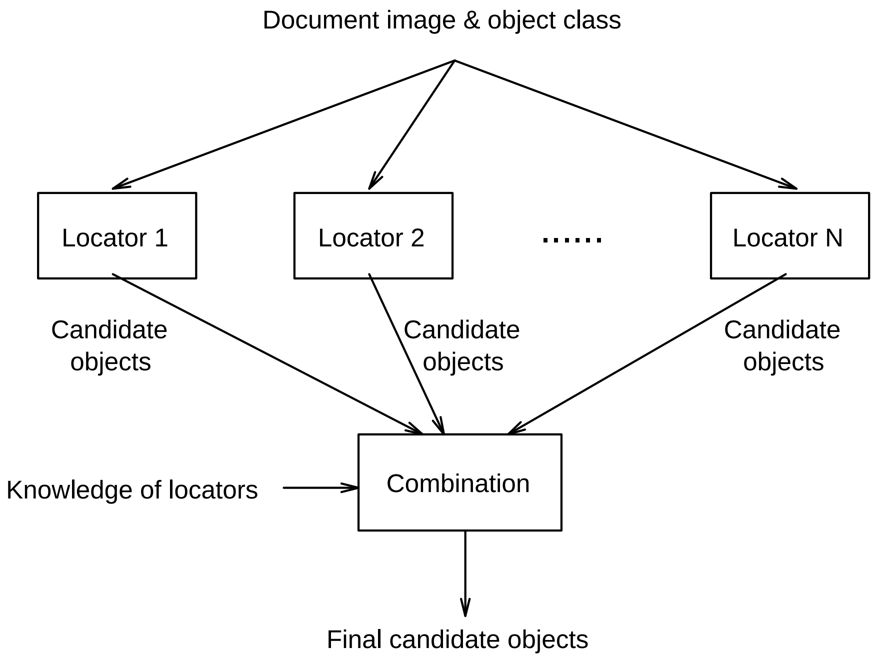


Figure 1.1: Role of document object locator combination

ure 1.1 illustrates the role of document object locator combination. The purpose of the combination is to provide integrated location performance which is better than that of any individual locator.

This chapter is organized as follows. Section 1.1 defines a few terms used throughout the thesis. Section 1.2 discusses the motivation for solving the problem and Section 1.3 presents an overview of the main results of this research. Finally, the dissertation is outlined in Section 1.4.

1.1 Definitions

We first define a few terms to be used frequently in this thesis.

A *true object* is an object that belongs to the class of interest. In other words, it is the object we want to locate. Depending on the object class and the domain of images, multiple true objects can exist in an image. For example, if we are finding text objects in a journal page image, there will be many true (text) objects. In this research, we consider only those cases where only one true object exists in an input image.

Object location is the task of visually finding the true object in an image.

Document object location is the task of visually finding the true object in a document image.

An *object locator* or a *locator* is a function that maps an image into a set of subimages (regions) where each subimage is a candidate for the true object. If the subimage set has more than one element, either the subimages are ranked based on their likelihoods of being the true object or each subimage is accompanied by a measurement value related to the confidence of the locator in it being the true object. By this definition, an image segmenter that simply outputs a set of subimages without any ordering information is *not* regarded as an object locator.

1.2 Motivation

There are three primary factors that affect the difficulty of an object location problem: (i) the distinctness of the visual features of objects in the class of interest; (ii) the variation of visual characteristics of objects in the class of interest; and (iii) the visual complexity of the environment. Accordingly, the difficulty level varies widely with the application.

For document object location, we are faced with all of those difficulties. For example, in the address block location problem, most objects are text objects such that address blocks are not visually distinct from other types of objects, address blocks show a wide variation in visual characteristics, and mail piece images may contain extraneous graphic and text objects, making the environment visually complex.

Most approaches to document object location consist of segmentation and classification stages. The segmentation stage divides an input image into several regions (or subimages) such that each region corresponds to a unique object. The classification stage classifies each region into one of predetermined object classes, one of which is the class of interest.

There are several types of errors associated with segmentation as identified by Ranganath [80].

1. Missing objects: One or more objects may not be visible in the segmented image. This error is possible when the object is not clearly visible due to glare or shadows.

2. False objects: Extraneous marks or shadows may be segmented as regions which do not correspond to any scene objects.

3. Fragmented objects or oversegmentation: A single object may be segmented into more than one region.

4. Merged objects or undersegmentation: Two or more objects may be combined into a single region.

All of the above types of errors can occur during document image segmentation.

An incorrectly segmented object leads to mismeasured region attributes which may not exactly represent the attributes of the corresponding object in the scene. This may result in incorrect classification of the object. Besides this dependency of classification correctness on segmentation correctness, inherent difficulties of classification itself remains with the two-stage approaches to document object location.

For difficult pattern recognition problems, using multiple recognition algorithms and combining their results has shown to be a promising direction. A similar concept can be used for document object location problems involving object classes that are not visually distinct from one another and have subclasses with widely varying visual characteristics.

That is, it might be more effective to use a collection of simple algorithms each of which is good for certain instances of the object class and combine their results than attempting to use one complex algorithm designed to locate all possible instances of an object class. Such an approach would apply several object locators tuned to different object subclasses or characteristics and combine their results to achieve combined location performance that excels the performance of any single locator.

For document object location, several alternative methods exist both for segmentation and classification. Given a document image, locators based on different methodologies output their own sets of objects with possibly different output information levels ranging from a unique candidate to a set of candidates with measurement values. Combining these locators of different types is a challenging task. Also, locators using different segmentation algorithms generate object sets that often have no intersection; no two objects from different locators are identical. Decisions of two locators cannot be directly combined in this case. Thus a method of determining object groups is needed such that decisions on objects in the same group can be combined. In addition, locator performance levels vary and using the knowledge of this difference during combination will be useful.

It will be interesting to investigate these issues for combining multiple solutions to produce a robust solution to document object location problems. This thesis attempts to provide a general theory for doing that.

Though our research focus will be on document objet location, we expect that our theory can also be applied, with little modification, to other problem domains that have the same difficulties of object location we discussed. Some examples include face location in complex backgrounds, vehicle license plate location in road scenes, and hand location in a gesture-based human-computer interface.

1.3 Overview of Main Results

The primary goal of this research is to develop a theory of combining results of multiple document object locators such that the combined location performance is better than any individual locator performance. In this section, we present an overview of the main results of this research. In short, we developed such a combination theory and demonstrated its effectiveness in an experimental domain.

1.3.1 Theory

There are several issues to be investigated in a theory of multiple document object locator combination. For each such issue, we explain its significance and briefly describe our solution to it in the theory we develop.

Representing Locator Outputs

The information contained in the output of a locator can be at any of the following levels, in order of increasing amount of information: (i) a single object only; (ii) multiple objects that are ranked; and (iii) multiple objects where a measurement value is associated with each one. Consequently, unless we restrict locator types so that their outputs are all at one level, we will need a mechanism to represent location results at different levels in a consistent form. Such a mechanism will facilitate fruitful integration of multiple location results at arbitrary levels.

We do not restrict locator output levels, and develop a scheme capable of representing locator outputs at any level. Locator outputs are represented uniformly using the concept of *objective confidence* values, which are probabilistic values acquired by observing the behavior of the locators to be combined on a set of training images. We show that this representation scheme allows for easy incorporation of knowledge of locator performance as well as consistent representation of location results.

Determining What Decisions to Combine

In any decision combination problem, it is of great importance to determine what decisions to combine. This determination is natural for some decision combination problems like multiple classifier combination, where multiple decisions on a single candidate class exist.

On the contrary, in multiple locator combination, unless the locators adopt the same segmentation algorithm, multiple decisions on a single object are not readily available. Rather, decisions on similar objects are available, because different locators usually use different segmentation algorithms. Therefore, we need to develop a method of generating groups of objects such that decisions on the objects in a group can be combined.

We introduce the concepts of *object equivalence* and *object correspondence*, and show that they are useful for handling this task. Basically, decisions on objects that are equivalent or corresponding can be directly combined as if the decisions were made on a single object. Methods for determining whether two objects from different locators are equivalent or corresponding (to each other) are developed.

Using Knowledge of Locators

When we have multiple locators, their performance levels are not necessarily uniform. This implies that it will be useful to incorporate the knowledge of this difference into a locator combination process. Objective confidence values contain this knowledge to some extent, since they are obtained from statistics on locator behavior. This knowledge can also be extracted explicitly in terms of the *locator relevance* values, which represent how relevant the locators are to a set of training images. Some combination methods proposed in our theory use the locator relevance values, while others do not.

In addition, because some locator is usually more relevant than others to images satisfying certain conditions, it will be worthwhile to compute locator relevance values dynamically, depending on input image conditions. We present a method for doing that based on training image set partitioning and test image classification into one of the partitions. Dynamic locator relevance values are used for realizing dynamic selection of locators.

Combining Confidence Values

The goal of locator combination methods is to combine objective confidence values such that the true object is assigned as high a combined confidence value as possible. An ideal combination method would always assign the highest confidence to the true object as long as it is output by some locator. In practice, a locator combination method will be considered as successful if it performs better on a set of test images than any individual locator.

We propose five combination methods: the highest confidence, confidence summation, weighted confidence summation, probabilistic confidence estimation, and fuzzy integral methods. We also propose a method of dynamic locator selection.

Evaluating Combination Performance

Proper evaluation of combination performance is an important aspect of a locator combination theory. One commonly used measure is the extent to which location performance is improved over individual locator performance, measured on a test image set. Though this measure can be important practically, we note that it varies too much according to the overall quality of location results that are input to a combination method, rather than the combination method itself.

We show that we need to distinguish between the location performance improvement due to combination and the combination performance. We propose a measure of combination performance and demonstrate its usefulness.

1.3.2 Experiments

We experimented with the address block location problem, which is visually finding the destination address block given an image of a mail piece. The destination address block thus found would be processed by an appropriate address interpretation module for automatic mail sorting. In the problem, objects are blocks formed by segmenting a mail piece image and the true object is the destination address block.

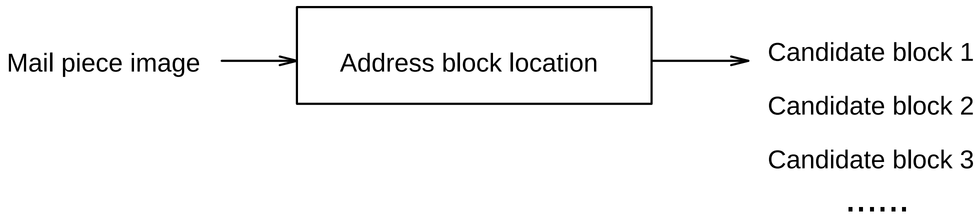


Figure 1.2: Address block location input and output

A locator outputs candidate blocks for the destination address block as Figure 1.2 shows. The output can be a single block, a set of ranked blocks, or a set of blocks with measurement values.

This problem is challenging because it possesses all of the three types of difficulties of object location problems discussed in Section 1.2. The problem has been studied for many years. Various solutions have been proposed and tested on different sets of mail piece images. However, a multiple locator approach has not been studied seriously yet.

We apply our theory to the problem to demonstrate its use in a difficult document object location problem. The experimental address block location algorithm employs three locators: a machine-printed address block locator, a handwritten address block locator, and a clustering-based locator. Each of them provides output at a distinct information level. The experimental results show that our theory, when applied to a document object location problem known to be difficult, provides combined location performance that is much better than any individual locator performance.

1.4 Dissertation Outline

The outline of this dissertation is as follows. Chapter 2 is a review of multiple expert approaches to pattern recognition and computer vision problems, and a survey of approaches to address block location. Chapter 3 presents our theory of multiple document object locator combination. Chapter 4 describes an address block location algorithm that uses multiple locators and combines their results based on the theory.

Chapter 5 presents experimental results of testing the algorithm on a set of mail piece images and analyzes them. Lastly, Chapter 6 gives research conclusions and discusses future research directions.

Chapter 2

Literature Review

In their paper on combining two techniques for segmentation, Pavlidis and Liow [74, page 225] point out: "The general principle of integration is well accepted in vision. But carrying out the general principle is quite challenging and this may explain the sparsity of the literature on integrated techniques." Pavlidis [73, page 224] expresses another view: "The main obstacle to integration of methodologies is that various approaches are offered as 'panaceas' rather than as members of a toolset." These quotations help explain why integration of approaches to vision problems has not been pursued actively in the past.

Research literature on combining multiple document object location methods is very rare. However, using multiple methods and integrating their results is a well-known approach to computer vision problems, and substantial research on using multiple methods has been done in areas related to this research, which we review in Section 2.1.

The address block location problem has been studied for many years and various approaches, ranging from simple pattern classification approaches to complex rule-based systems, have been proposed. Major approaches to the problem proposed so far are reviewed in Section 2.2.

2.1 Combination Approaches in Related Areas

In this section, we review combination approaches in research areas related to document object location. Section 2.1.1 reviews a few attempts at utilizing multiple document object locators. Research on image segmentation using multiple techniques is reviewed in Section 2.1.2. Considerable work on utilizing multiple classifiers has been done recently, which we review in Section 2.1.3. A review of some frameworks for integrating visual modules is given in Section 2.1.4.

2.1.1 Combining Document Object Locators

A few address block location systems [43, 70, 56] attempt to combine multiple document object location results. Jelinek et al. [43] present an address block location system with three separate channels of processing corresponding to red, green, and blue images. Each channel performs an identical sequence of processing steps. The 'multiple channel reconciliation' module merges and sorts three lists of top ten candidates, one from each channel, into the final list of top ten candidates.

Palumbo et al. [70] discuss an address block location system that uses four segmentation methods designed for different text types and/or based on different segmentation techniques. This system processes the four segmentation results independently using the same ranking method, to derive four different sets of ranked candidate blocks. Only the top choices from each ranking are compared using the knowledge of destination addresses to determine the candidate most likely to be the destination address block. If two top choices from different rankings occupy almost identical regions, they are treated as one candidate.

Lii et al. [56] describe an extended version of the system presented in [70] that uses five segmentation methods. The system generates five independent ranking results. From each ranking, the first block with the city, state, and Zip Code information is collected. If multiple blocks are collected, they are grouped based on spatial overlap, the group close to the lower-right corner of the mail piece image is selected, and the best-looking block of that group becomes the final choice. If only one block is collected and its location is reasonable it will be taken as the destination address. Otherwise, the combination results as generated in [70] will be output. In short, this system collects from the initial rankings those candidates showing the address syntax and reranks them.

Remarks

The approaches described are of limited combination capability in that only the top choices are combined and the same ranking method is used on all segmentation results. Moreover, no clear guidelines as to reranking of candidates has been suggested.

2.1.2 Combining Segmentation Methods

Combination of segmentation methods is not the subject of this research. However, we can obtain helpful insights by reviewing combination approaches to image segmentation problems, as some type of segmentation is necessary for any object location task. Many research results published on this topic is based on combining region and edge information, because they are often complementary in many image domains.

Using Region and Edge Information

Nazif and Levine [64] propose an expert system which integrates region and edge information by a set of heuristic rules. The system has both region analysis and boundary analysis rules. Segmentation results on a natural scene and a blocks world image are shown and compared with histogram splitting and split-and-merge results.

Sugiyama and Abe [89] propose an image segmentation method that uses both edge and region information to partition an image into 'certain' and 'uncertain' regions. The interior of a certain region has uniform features and no edges while its boundary is an edge and not uniform. Pixels not belonging to any certain region are classified as uncertain.

Using Edge Information to Improve Region-Based Segmentation

The studies reported in [1, 24, 74, 97] use the strategy of using edge information to improve region-based segmentation or to control a region growing process. In effect, the segmentation result of one method is modified or improved using the result of another method. Therefore, these can be classified as approaches that uses multiple techniques, rather than as approaches that integrates multiple segmentation results.

Anderson et al. [1] use the number of regions that is assumed to be known to estimate parameters of an edge detection algorithm. The edge detection result is then used to initialize and assist a region growing algorithm where a local similarity threshold is gradually increased until the expected number of regions are produced.

Fua and Hanson [24] apply high-level domain knowledge and an edge-based technique to select the best segmentation from a series of images segmented based on regions. They apply the method to extracting instances of generic shapes such as buildings, roads, and vegetation clumps.

Pavlidis and Liow [74] present a method that combines region growing and edge detection for image segmentation. The method starts with a split-and-merge algorithm in which parameters are set up so that an oversegmented image results. Possible errors resulting from region growing processes are classified into three types, and region boundaries are eliminated or modified based on the edge detection result. Results are shown using tool images and aerial photograph images. They conclude that the combination of two methods yields far better results than those of a single method.

Xiaohan et al. [97] use local gradient information to control a region growing process. Edge information influences the region growing process via a parameter that reflects the sensitivity to edge pixels. The results are demonstrated on magnetic resonance images of a human head.

Selecting Region Growing or Edge Detection

Salotti and Garbay [83] propose a segmentation method which allows interactions between various modules used for low-level segmentation. The method first calculates local properties of pixels and decides region types and edge types. Based on this information, the method dynamically activates region growing or edge detection modules. However, the method was not fully experimented with and only region growing part was implemented.

Pixel-Level Thresholding Result Selection

Mehtre and Chatterjee [62] propose a composite method for segmentation of fingerprint images. They introduce the directional method and the variance method which are combined at pixel level. The directional method fails on images with uniform regions, as the expression for the direction computation becomes undefined in such regions. To overcome this limitation, the composite method employs the variance method where the directional method fails, which is detected by the low gray variance property of the uniform regions. Depending on the variance property, the method selects the result from one of the methods, pixel by pixel. This method integrates two thresholding results rather than segmentation results, by pixel-level selection.

Segmentation Map Integration

Chu and Aggarwal [14] investigate parallel integration of multiple segmentation results. They describe an algorithm that integrates multiple region segmentation maps and edge maps. The integration algorithm attempts to generate a consensus of the true underlying segmentation from multiple observations. The algorithm allows user-specified weights on segmentation maps to emphasize a specific segmentation map. Results are demonstrated on four source images of a single scene, representing range, intensity, velocity, and thermal information, respectively. Four different methods are applied to segment or detect edges on the images and their results are integrated using various weights, emphasizing different input components. However, the authors do not discuss how to determine the weights.

Selectively Using the Second Segmentation Method

Having a secondary segmentation method and using it only when required can be efficient. Wang and Srihari [94] use two segmentation methods for address block location, one for machine-printed text (MSEG) and the other for handwritten text (HSEG). MSEG is always applied first because it is faster and more mail pieces have a machine-printed destination address than a handwritten one. If investigating MSEG results provide clues to suggest the existence of a handwritten address, then HSEG is used. The approach uses the method with the higher utility first and uses the method with the lower utility only when it is deemed necessary to do so.

Region Ranking Modification by Segmentation Results

Palumbo et al. [69] discuss an address block location system that uses three segmentation methods, one for machine-printed addresses and two for handwritten addresses. The segmentation results, three different sets of blocks, are processed together to produce a ranking of candidates for the destination address block. After the ranking is generated, a candidate block that (i) is generated by a segmentation method different from the one that produced the top-ranked block, (ii) is assigned high confidence, and (iii) overlaps a lot with the top-ranked block is identified. If such a block exists, it is compared with the top-ranked block and the block with more complete line structure is favored. This system basically uses multiple segmentation results to possibly change the top-ranked block.

Aggregating Fuzzy Pixel Labelings

Krishnapuram and Lee [49] as well as Keller and Krishnapuram [46] describe fusion of color information for segmentation and labeling of an outdoor scene. Three feature images, representing intensity values, blue-red differences $(b-r)$, and excess green values $(2g-r-b)$, respectively, are used. The pixel position is used as the fourth feature. These features are used to segment a scene and label the regions as sky, tree, roof, walls, grass, and road. Based on each feature, the degrees of membership of each pixel in the labels are computed. A two-layer neural network combines the membership values using the γ-model [101] as the aggregation operator, which allows varying degrees of compensation between the union and intersection.

Local Decision Integration

Etemad et al. [20, 21] present a document page segmentation algorithm which classifies regions into text, image, and graphics. The algorithm uses feature vectors based on a multi-scale wavelet packet representation for local classification. Segmentation is done by propagating soft local decisions made on small windows across neighboring blocks and integrating them. Local votes are combined by weighted majority voting. Good performance on complex document layouts using simple feature sets and classifiers is reported,

Remarks

Using both region and edge information is a prevalent research trend in combining segmentation results. However, parallel integration of multiple segmentation methods has not been explored much.

2.1.3 Classifier Combination

Combining multiple document object locators is not a classification problem, but reviewing approaches to combining multiple classifiers must be useful. Both problems are concerned with the effective use of multiple sets of candidates generated by multiple methods. In classifier combination, the candidates are classes, while in object locator combination, they are objects. Though classes and objects have quite different natures, they are both entities for which decisions from multiple methods are to be combined.

A review of the research literature on multiple classifier combination is given by Ho [30]. We mainly discuss more recent research results in this area, where character recognition and word recognition are most intensively studied problems. We also review multiple classifier approaches to other classification problems such as face recognition, cytology screening, person identification, and oceanic signal classification.

Dynamic Classifier Selection

The principle of dynamic classifier selection is to apply the classifier that is determined to be most appropriate for a given input pattern. Some methods of realizing this capability have been proposed.

Xu et al. [98] suggest a method of classifier combination called the 'associative switch.' Classifiers output a single class or reject the input. Attached to the end of each classifier is a switch controlling whether the classifier output will pass through to the final result. The switch is controlled by a neural network that inputs an unknown pattern and generates a code that represents the states of the switches. Experiments on handwritten numerals show improvements over individual classifier performance.

Ho et al. [30, 32] suggest a method for dynamic classifier selection based on image set partitioning. It divides the training set into several partitions by mutually exclusive conditions easily computable from an image. For each partition, classifier performance is measured and the best classifier is determined. A test image is first classified into a partition and the best classifier for that partition is used. They describe an example of applying this idea to choosing a different parameter set for the same regression model for combining classifiers, based on the agreement among top choice classes from classifiers. For each partition, different classifier weights are determined instead of determining the best classifier.

Lee and Srihari [52] emphasize exploiting the dependency of classifier performance on pattern characteristics in classifier combination. They propose two methods of dynamically selecting the most appropriate classifier for a given pattern: postconditioned selection and preconditioned selection. Postconditioned selection is based on the conditions of classifier outputs, whereas preconditioned selection is based on the conditions of the input image. They argue that preconditioned selection has lower error bounds than postconditioned selection. A neural network solution to preconditioned selection is given. Handprinted digit recognition is used as the example problem illustrating the benefits of the solution.

Serial Combination

Duerr et al. [17] present a hierarchical multi-stage organization of a handwritten numeral recognition system. The system consists of a statistical classifier, a structural classifier, a stage for reducing the number of hypotheses, and a combined stage for making the final classification. They claim that the performance of the combined system is definitely higher than that of pure statistical or structural systems.

Kimura and Shridhar [47] use a statistical classifier and a structural classifier in two parallel combinations and four sequential combinations. Separate combination rules are used for each combination of classifiers. Experiments with recognizing handwritten numerals show that the parallel combinations usually perform better than the sequential ones exploitation of multiple memberships is effective. They also show that for sequential combinations applying the algorithm with the lower error rate first is better than applying them in reverse order.

Madhvanath and Govindaraju [60] investigate serial combination of handwritten word classifiers. A fast classifier is used as the lexicon reducer in a preprocessing step and a better and generally slower classifier is used as the recognizer. They conclude that the performance of a serial combination depends not only on the individual recognition performance of classifiers but also on the relative orthogonality of their features. They also conjecture that higher recognition rates will be achieved when serially combining two classifiers using less correlated features.

Voting

Hansen and Salamon [28] study collective performance of multiple neural networks. The collective decision is reached by the plurality voting rule, where the final decision is the classification made by more networks than any other, and the majority voting rule, where the classification made by more than half the networks is the final classification. They also investigate models that incorporate independent errors, input difficulty, and network proficiency. Experiments are performed on the generalized XOR problem and a random pattern classification problem.

Network Averaging

Perrone and Cooper [75] propose network averaging as a method of combining multiple neural networks. They propose the 'generalized ensemble method' which averages a set of regression estimates obtained from a neural network ensemble. The method takes a set of neural networks which have converged to local minima and apply averaging to construct an improved estimate. Experimental results on character recognition using the NIST OCR database are reported. They conjecture that the reason for the improvement is that networks which have fallen into different local minima will perform poorly in different regions of the feature space, and the errors will be weakly correlated.

Dempster-Shafer Evidence Theory

Shlien [86] proposes the use of multiple binary decision tree classifiers where each tree is designed using a different feature selection criterion. A single feature vector is thus processed by several decision trees. Classification results are combined by the Dempster-Shafer model [5, 84] which combines associated confidence measures. Experimental results on recognizing machine-printed characters show performance improvements.

Rogova [81] presents a method for combining neural network classifiers based on the Dempster-Shafer theory of evidence. Two experiments are performed: one on a private database and the other on the NIST database [92]. In each experiment, three classifiers are employed and several combinations are tested. One observation they make is that a better result is not necessarily achieved by combining classifiers with better individual performance, and in some cases combining more independent classifiers produces a better result.

Behavior-Knowledge Space Method

Huang and Suen [37, 35] propose a method called the 'behavior-knowledge space method,' which can aggregate decisions of individual classifiers. The method assumes that classifiers either output a unique class label or reject the input. The method consists of: (i) the knowledge-modeling stage, which extracts knowledge from the former behavior of classifiers and constructs a K-dimensional behavior-knowledge space, where K is the number of classifiers; and (ii) the operation stage, which combines decisions of individual classifiers by entering a unit of the space and making a final decision. Handwritten numerals are recognized using the method and the results are compared with voting, the Bayesian method, and the Dempster-Shafer method. The experiments show that all four methods perform better than any individual classifier and that the proposed method generally performs best.

Neural Network

Huang and Suen [34] use a backpropagation neural network to combine the results of multiple classifiers that output measurement values. Measurement values output by a classifier are transformed into a new form such that their range is between 0 and 1, their sum for all classes is 1, and the rank order is preserved. The transformed values are treated as features and input to a neural network functioning as another classifier.

Lee and Srihari [51] propose a multilayer perceptron as an architecture to combine multiple classifier outputs. They identify desirable properties of combination including omnitype, mixed type, correlated classifier combination, redundant classifier elimination, model complexity control, and dynamic selection. A multilayer perceptron takes outputs of several classifiers, some of them properly transformed beforehand, and outputs a decision vector indicating confidence values for all classes. Experiments are performed with handwritten digit recognition.

Fuzzy Integral

Tahani and Keller [90] give theoretical developments of an evidence fusion technique in computer vision based on the fuzzy integral [88]. Experiments in recognizing an armored personnel carrier and two types of tanks from forward looking infrared images are performed. For each object detected by preprocessing, a Bayes, a fuzzy k-mean, and a feature-level fuzzy integral classifiers are used and their results are combined using the fuzzy integral. They show that the technique is able to nonlinearly combine objective evidence with subjective evaluation of the worth of the sources toward the final decision.

Arbuckle et al. [3, 4], in a face recognition problem, treat a linear discriminant classifier used at five different image scalings as five different classifiers and combine their results using the fuzzy integral. To use the fuzzy integral for combination, distance measures in the range $[0,\infty)$ are converted into probabilities using the method suggested by Xu et al. [99], and error rates are used as the basis of classifier reliabilities. The combined classifier shows a significant improvement in the error rate.

Cao et al. [12] use the fuzzy integral to combine results of multiple character classifiers. They propose a method to assign classifier relevance using classifier performance matrices. Four neural network classifiers are combined and experiments are performed on handwritten numerals.

Cho and Kim [13] use the fuzzy integral to combine results of three neural network classifiers. They use a multiple network architecture for online handwriting recognition. The recognition results using the fuzzy integral are compared with results of other consensus methods such as the majority vote and the Borda count [8].

Gader and Mohamed [25, 26] use the fuzzy integral to combine three handwritten word classifiers based on segmentation, hidden Markov models [79], and fuzzy hidden Markov models, respectively. They compare combination results using the methods of the Borda count, weighted Borda count, Sugeno integral [88], and Choquet integral [63]. For calculating fuzzy measure densities, which also play the role of weights for the weighted Borda count method, they experiment with fixed and data-dependent densities and conclude that the latter lead to better performance.

Multiple Combination Methods

Studies of classifier combination described so far employ a particular method for combination. Many other studies propose and experiment with multiple combination methods. In general, these works attempt to compare effects of different combination methods for a pattern recognition problem, rather than advocate one method.

Xu et al. [99] investigate the classifier combination problem systematically and describe various approaches to the problem. Three levels in classifier output information are defined: the abstract level, the rank level, and the measurement level. One approach is proposed for combining measurement-level classifiers such as Bayesian, k-nearest neighbor, and various distance classifiers. Three approaches based on majority voting, Bayesian formalism, Dempster-Shafer formalism, respectively, are suggested for combining abstract-level classifiers. Various experiments on handwritten numerals show that combining multiple classifiers is a promising direction.

Franke and Mandler [22] compare two approaches to combining the votes of cooperating classifiers. One is based on the Dempster-Shafer theory of evidence and the other is a statistical approach with some assumptions on input data. Both approaches are tested in user-dependent recognition of on-line handwritten characters. They conclude that the Dempster-Shafer approach is better if there is not much training data, whereas the statistical approach is better when more training data is available.

Huang and Suen [36] study a method for combination of classifiers which output class labels along with corresponding measurement values. A measurement value is converted into a confidence value which indicates the probability of the class with the measurement value being the true class, based on training samples. Such confidence values for each class are simply summed up to determine the accumulated confidence value for the class. The final decision is made by selecting the class with the highest accumulated confidence value. Comparison with the voting method and the Bayesian method shows that the confidence accumulation method performs best. The authors thus conclude that measurement values can be used effectively if they are available.

Lee et al. [53] propose integration of neural network and decision tree classifiers for classifying objects appearing in pap smear slides as one of 'normal,' 'abnormal,' and 'artifact.' Class confidence values are derived using a method similar to that used by Shlien [86], which are combined by a heuristic method and a Dempster-Shafer theory based method. Combination results show considerable improvements in classification correctness.

Ghosh et al. [27] present and compare four methods of evidence combination for classification of underwater acoustic signals. The four methods are entropy-based weighting of individual classifier outputs, combination of confidence factors in a manner similar to that used in MYCIN [10], majority voting, and averaging. All the methods are reported to have shown better results than those obtained by the best individual classifier.

Ho et al. [30, 31, 32, 33] discuss a multiple classifier system, where classifier decisions are represented as rankings of classes. The rankings are combined by methods that either reduce or rerank a given set of classes. They propose an intersection method and a union method for class set reduction. For class set reranking, three methods based on the highest rank, the Borda count, and logistic regression, respectively, are proposed. The methods are tested in degraded machine-printed character recognition and word recognition, and substantial improvements over individual classifier performance are reported.

Matsui et al. [61] propose three combination rules for combining handwritten numeral recognition results which include candidate names, their order, and dissimilarity values. The proposed rules are voting on the first candidate, minimal sum of candidate orders, and minimal sum of dissimilarities. To incorporate classifier characteristics, they also propose the 'candidate appearance likelihood method,' which is based on the conditional probability values relating the combined classifier output vector and its correctness for each class. They conclude that the candidate appearance likelihood method is more promising than the simple combination rules.

Plessis et al. [77] describe strategies for combining results of handwritten cursive word recognition methods. The methods are holistic word recognition, segment/recognize with statistical classifier, and segment/recognize with neural classifier. They experiment with three different combination configurations for combining the three methods, each one possessing a different degree of parallelism.

Tumer and Ghosh [91] employ two neural network classifiers whose outputs are interpreted as Bayesian a posteriori probabilities and combine their results. Two separate feature sets are used, resulting in four classifiers in effect. Each of the six possible pairs of classifiers are tested in experiments on classifying underwater passive sonar data set. They perform combination using the maximum and averaging operators and compare their performance with that of the best classifier and the theoretically best performance. Either of the two combination methods achieved improvements over the best classifier among each pair of classifiers.

Brunelli and Falavigna [9] present a person identification system that integrates speaker recognition and face recognition modules. Two classifiers using acoustic features and three using visual ones are combined using two different approaches. In the first approach, classification scores are normalized and combined using a weighted geometric average. In the second one, a mapping from the scores and ranks of classifiers into the interval (0,1) is defined. Using the mapping, they propose a method of combining measurement-level and rank-level classifiers. The performance of the integrated system is shown to be superior to that of the acoustic and visual subsystems.

Powalka et al. [78] discuss combination of multiple recognizers at the word level. They propose two approaches: word list merging and linear combination. Word list merging considers only the results of recognizers and does not take into account recognizer performance. Linear combination considers recognizer characteristics. Weights for the linear combination are derived from a set of handwriting characteristics. Recognition experiments are done with handwritten words.

Remarks

There are three possible types of classifier outputs. A classifier can output a unique class decision, ranked candidate classes, or candidate classes with associated measurement values. Most combination approaches combine decisions of a single type. Combination methods capable of combining classifier decisions of different types are given in [9, 51].

We also observe that most approaches organize classifiers in a parallel fashion rather than a serial fashion. Serial combination of classifiers is investigated in [17, 47, 60]. Computational models for dynamically selecting the best classifier for an input pattern are proposed in [52, 98].

2.1.4 Integrating Visual Modules

Integration of vision modules is recognized as a promising approach towards robust solutions to many vision tasks, especially in low-level vision and 3-D reconstruction. We review a few studies of this sort.

Sher [85] addresses the problem of combining outputs of several boundary detectors that return likelihoods. The likelihoods are combined in a Bayesian framework such that the combined detector works well when the assumptions of at least one of the individual algorithms are met. Boundary detection experiments using the proposed combiner is demonstrated using artificial, toy, and aerial images.

Pinz and Bartl [76] give examples which show that information fusion in image understanding can be carried out at several levels. In classification of remote sensing images, several image classification results are combined at the image level, by looking at a single pixel at a certain location in all classification images. Multiple ocular fundus images of different geometry can be registered by combining multiple vessel descriptions at the image description level. In interpreting billiard scenes, three images i_1, i_2, and i_3, where i_1 and i_2 are taken at the same time from different viewing positions and i_2 and i_3 are taken at different times but from the same viewing position, are analyzed and the results are combined at the scene description level. They conclude that fusion at certain level has typical properties which favor certain fusion methods.

Wong and Frei [96] suggest a scheme to combine three methods for object recognition. The first method uses geometric features, the second is appropriate for jagged structures in objects, and the third is for objects described well with a skeleton. They in essence use different methods for recognizing different types of objects such that a method usually recognizes only a portion of the scene. A weight is associated with each object recognized and also with a method used for combining recognition values of multiple methods. They claim that if the weights are chosen right, the actual combination approach does not matter. Experiments with line drawing object images show that using multiple simple methods enable more objects to be recognized.

Pankanti and Jain [72] discuss an implementation for integrating perceptual organization, stereo, shape from shading, and line labeling modules. The modules explicitly exchange information with an objective of obtaining accurate 3-D reconstruction. Experiments with synthetic and real data indicate that integrated systems are likely to produce better reconstruction of the input scene than the individual modules.

Remarks

Studies in integrating visual modules mostly deal with combining object recognition modules at various levels. Integrating object location modules has hardly been studied.

2.2 Approaches to Address Block Location

Research on address block location has been active during the past decade. Various approaches ranging from simple pattern classification to complex knowledge-based systems have been proposed in the literature. A survey of five early address block location systems is given by Koljonen and Glickman [48]. Brief reviews of approaches to address block location can also be found in [38, 54].

We classify approaches proposed so far into five major categories based on their most prominent features: knowledge-based systems, pattern recognition approach, geometric feature-based approach, texture analysis, and address syntax analysis. Our review is organized by these categories.

2.2.1 Knowledge-Based Systems

Knowledge-based or rule-based systems were commonly used for early address block location systems designed to handle several different types of mail streams such as letters, manual letters, flats, and irregular parcels and pieces. Because mail piece images from different streams have very different characteristics, a complex set of rules was required to process mail piece images that might come from any stream appropriately.

Srihari et al. [87] describe an address block location system (ABLS) based on the blackboard model [19, 66] of problem solving. ABLS is composed of six parts:

mail statistics database, rule-based inference engine, control data, control mechanism, blackboard, and toolbox. It is a framework for coordinating a collection of specialized image and symbolic processing tools needed for mail piece image analysis. Experiments are performed on mail piece images acquired from live mail. Arising from ABLS, a computational framework for object recognition in a complex visual environment is described in [93, 94].

Palumbo et al. [69, 70] present a real-time address block location system developed on the basis of ABLS. The system uses less image processing operations than ABLS and replaces the complex rule-based inference engine of ABLS with a more deterministic evidence combination method. In particular, the system uses multiple 'pools of processors' to achieve real-time performance.

Antognini and Turnbaugh [2] use an expert system shell to implement a rule-based address block location system. Using color analysis, the system first looks for colored address labels. If it finds one, the processing skips to the very end. Otherwise, gray-scale processing determines whether the image is 'cluttered.' If so, the mail piece is assumed to have a machine-printed address and machine-printing segmentation rules are used. Otherwise, segmentation rules for handwriting are applied. When the image has been deemed cluttered and confidence values of top candidate blocks are low, the cluttered assumption is retracted and the image is reprocessed using handwriting segmentation rules.

Bartneck [6] presents hybrid knowledge representation schemes for expressing knowledge used in various modules for address block location. Frame-based knowledge representation is used for image acquisition, iconic image processing, block segmentation, and block evaluation modules. A rule-based scheme is used for decision expert, conflict expert, and control expert modules. The termination module uses procedural knowledge.

De Gaetano [15] uses a rule-based expert system only in the block ranking stage. Color and gray-level processing to extract candidate blocks produce Prolog facts about the incoming mail piece. Knowledge of mail characteristics is represented as Prolog predicates that are partitioned into five levels. The local level analyzes local block features, the context level determines the mail class (letter, flat, magazine, and paper), and the orientation level determines if the image is right side up. The weak global and strong global levels analyze relationships between blocks. The system uses three types of reasoning models. Probabilistic theory is used to deal with statistical data, fuzzy set theory to handle memberships (such as in the context level), and evidence combination is performed using confidence factors as in MYCIN [10].

Jelinek et al. [43] propose an address block location architecture with three channels of processing corresponding to red, green, and blue color images, where identical processing is done in each channel. Low-level processing consists of the 'similarity pattern method' and 'edge linking and classification.' Segmentation partitions the edge image into meaningful regions of the image called windows. The window tree is pruned and features are assigned to remaining windows. Based on the features, each window is interpreted using both local and global knowledge, resulting in an interpretation vector. A global expert compiles a list of the top ten candidates and pass it to the multichannel reconciliation module that combines rankings from all color channels and outputs a list of candidates.

Lee and Kim [54] describe a knowledge-based system for address block location on handwritten Korean envelopes. Correct segmentation of a handwritten Korean envelope image is very difficult, since the return address and the destination address are usually close to each other. Accordingly, they use an iterative approach that merges and splits connected components and verifies the candidate block without segmentation and selection. At each iteration, hypothesized destination address blocks are verified by statistical knowledge. If verification succeeds, the system terminates. Otherwise, the block is merged or split using merging and splitting conditions, and the process repeats.

2.2.2 Pattern Recognition

A pure pattern recognition approach to address block location considers each candidate block as a pattern and classifies it into one of several object classes encountered in mail pieces. Destination address is found by selecting the block with the highest confidence measure of belonging to the destination address class. This approach has not been pursued much, mainly because many researchers considered it too straightforward to adapt to widely varying visual properties of destination addresses.

Bergman et al. [7] propose an approach of this type. The approach detects horizontal and vertical edges separately and form candidate blocks by finding edge pixels showing a signature of text and merging them properly. It then extracts eight features for each candidate block and use the Mahalanobis distance metric [18] to classify the block as one of destination address, return address, stamp, clutter, large clutter, and small address fragment. For each candidate block, its distance to the destination address class and the weight of the class assigned to it are combined to compute the confidence measure for it. The block with the highest confidence measure is identified as the destination address block.

2.2.3 Geometric Feature-Based Approaches

Approaches based on geometric features analyze local features of segmented blocks to extract candidate blocks and use simple geometric relations to choose the destination address block. These approaches neither perform rule-based inference nor use classifiers.

Yeh et al. [100] present an approach of this kind. The Laplacian operator is applied and connected components are detected and merged to eliminate spurious ones. For each remaining component, several local features are calculated. A 'proximity tree' representing the similarity of these features among components is constructed, which is cut at some distance to generate several clusters of components. Each cluster is assigned scores for belonging to categories of destination address, return address, and stamp, based on its position. The cluster with the highest score for destination address is selected as the destination address.

Downton and Leedham [16] propose a simple approach based on geometric features. Horizontal smearing is done on the binary image, and its resolution is reduced to remove small regions of high black pixel density. Simple local features are calculated to identify regions that can be part of the destination address, which are then smeared vertically to form candidate blocks. The block nearest to the nominal center point, the point at half the horizontal length from the left end and one third the vertical height from the bottom, is chosen as the destination address.

2.2.4 Texture Analysis

Texture analysis approaches use a certain homogeneous texture property of text regions as a cue for isolating text from mail piece images. Transition analysis techniques treat the stability of specific distances relating to characters as texture. Multichannel Gabor filtering [41] is another technique proposed for text segmentation from mail piece images using textures.

Oriot et al. [67] use transition analysis to identify text regions for address block location. Possible text zones are isolated by analyzing two distances based on transitions: the horizontal distance between two successive '01' transitions and the vertical distance between a '01' transition and the following '10' transition. Word merging and line merging on the result of the transition analysis generate 'consistent' text blocks. For each block, local block features are calculated and weights assigned for feature values are combined into a confidence value of being the destination address.

Jain and Bhattacharjee [38] view the address block location problem as a texture segmentation problem. Their algorithm uses Gabor filters [41] to obtain a three-cluster segmentation, corresponding to text, non-text, and boundaries between text and non-text regions. It then identifies connected components that belong to the text regions, and use projection profiles to identify groups of words. For each group of words, the enclosing rectangle is constructed, which is a candidate address block. No heuristic to identify the destination address block was implemented.

Jain and Bhattacharjee [39] also propose a two-class (text and non-text) supervised pixel classification scheme. It applies four Gabor filters to extract texture features and uses a one-layer feedforward neural network with four input units and one output unit for pixel classification. Connected components of the text pixels are grouped to form candidate address blocks among which the one closest to the image center is chosen as the destination address.

Jain and Chen [40] use color and texture analysis to find address blocks on magazine covers. Their approach first performs a color thresholding to find white regions in the input color image. It then identifies text regions using Gabor filter-based texture segmentation. Integrating color and texture information is done by finding text inside white regions and merging adjacent text regions in the same white region. Some heuristics are used to remove candidate blocks which cannot be an address block.

2.2.5 Address Syntax Analysis

All approaches reviewed until now do not use any kind of character recognition, since character recognition on full mail piece images is too computationally expensive to be practical. However, address syntax can be used to verify candidate address blocks with the aid of a character recognition module.

Lii et al. [56] describe an extended version of the system presented in [70] that uses character recognition and address syntax. The system locates word candidates and assign a syntactic tag to each word by character recognition and word matching. Finding address blocks is formulated as a consistent labeling problem [58] using the tags and address syntax models. Address syntax analysis is also used in various aspects of the system, such as recovering lost critical information (city, state, and Zip Code), block ranking, and improving address block segmentation. Improved performance in comparison with the system without character recognition is reported.

Lii and Srihari [57] present a method to locate the recipient name and address on fax cover pages. The method performs connected component analysis, word segmentation, and text recognition to locate keywords such as 'To' and 'Company'. The keyword location is used as the reference position for block segmentation. Address block segmentation is carried out by applying the knowledge of fax cover page structures in reference to the keyword location. The method shows better performance on machine-printed fax cover pages than on handwritten ones.

Chapter 3

A Theory of Document Object Locator Combination

This chapter presents our theory of document object locator combination. Section 3.1 states the proposed problem in detail. In Section 3.2, we formulate the problem and describe the major components of our theory using the formulation. Section 3.3 describes issues regarding representation of location results and our solution to handling different types of results. Section 3.4 introduces the object correspondence problem, discusses its significance in our theory, and presents our solution to it. Section 3.5 discusses the concept of locator relevance and describes its importance in locator combination. In Section 3.6, we propose several methods for combining confidence values. Finally, Section 3.7 discusses criteria for evaluating the performance of locator combination.

3.1 Problem Statement

We are attempting to locate the object of interest, defined as the *true object* in Section 1.1, in a document image. Let us suppose that we have multiple locators for the task. Given a document image, each locator outputs a set of document objects, in the form of subimages or regions, which are the candidates for the true object. The object set can have any number of objects. As defined in Section 1.1, if the object set has more than one object, either they are ranked or each of them has a measurement value related to the confidence of the locator in it being the true object. In this environment, we state the problem we propose to solve.

The Problem

Let us suppose that we have multiple sets of document objects where each set contains the candidates for the true object output by one of the locators. Then the problem is to generate a new set of objects that are the candidates for the true object, subject to the conditions:

1. Each object in the new object set is an object output by *at least one* of the locators.

2. Each object in the new object set is assigned a *confidence* value indicating its likelihood of being the true object, such that the confidence value assigned to the true object is as close to the highest (of all confidence values that are output) as possible.

We refer to this problem as the *document object locator combination problem.*

The first condition means that no new object is generated by a solution. It also implies that the solution is not to be another locator that performs a kind of segmentation of the input image. Consequently, if the true object were not output by any locator, the new object set would *not* contain it. The second condition means that the final decision is to be represented by confidence values, not by a unique object or ranked objects. Given these conditions, a perfect solution would assign the highest confidence value to the true object, if it were output as a candidate by at least one locator. The number of objects in the new object set is insignificant in the solution.

Assumptions

We make two assumptions to develop a proper solution to the problem.

1. *Only one* true object exists in any input image.

2. *Knowledge* of the previous performance of the locators on a reasonably large set of images is *available*.

The first assumption is mainly for ease of developing a solution to the problem. We show in Section 6.2 that the cases where multiple true objects exist can be handled by minor modifications to the solution we develop. The second assumption implies that a good solution should use the knowledge to make informed decisions regarding the likelihoods of the candidates from the locators being the true object, rather than presume that all the locators perform at the same level.

Evaluation

In general, we consider a locator combination as successful if its location performance is superior to that of any locator used. Equivalently, a successful locator combination performs better than the best locator. Otherwise, combining locators is not justified because additional computation is involved without any gain in location performance. In Section 3.7, we discuss details of the performance evaluation criteria that dictate how to measure performance improvements owing to locator combination and the performance of the combination itself.

3.2 Formulation

In this section, we formulate our theory by means of transformations to clarify input-output relationships. We first note that there are three levels (or types) of results that can be generated by object locators. Xu et al. [99] identifies three levels in classifier output information. Analogous output levels can be defined regarding locator output information as follows:

1. *Abstract level*: A locator outputs a unique object, without any more information.

2. *Rank level*: A locator outputs a set of objects, ranked by the (internal) confidence of the locator in the objects.

3. *Measurement level*: A locator outputs a set of objects, where each object is accompanied by a measurement value related to the confidence of the locator in the object.

The amount of output information is smallest at the abstract level, intermediate at the rank level, and largest at the measurement level.

In general, it is not practical to assume that the available object locators whose results are to be combined all have the same output level. A locator combination theory relying on the assumption will have limited applicability. Therefore, we do not place any restriction on the levels of locator output information. This means that we are attempting to develop a theory which offers methods for combining location results that can be at any of the three levels defined above.

In addition, for measurement-level output, no assumption is made regarding the scale, range, distribution, and interpretation of the measurement adopted by the locator. In effect, we presume that different measurement-level locators use different and totally unknown ways to compute measurement values.

The components of our theory are shown in Figure 3.1. Let x be the input image and $m_1, \ldots, m_N$ be the object locators, where N is the number of locators available. Since the locator output levels are not restricted, the output of the locator m_i given an input image x can be represented by a vector which can take any one of the three

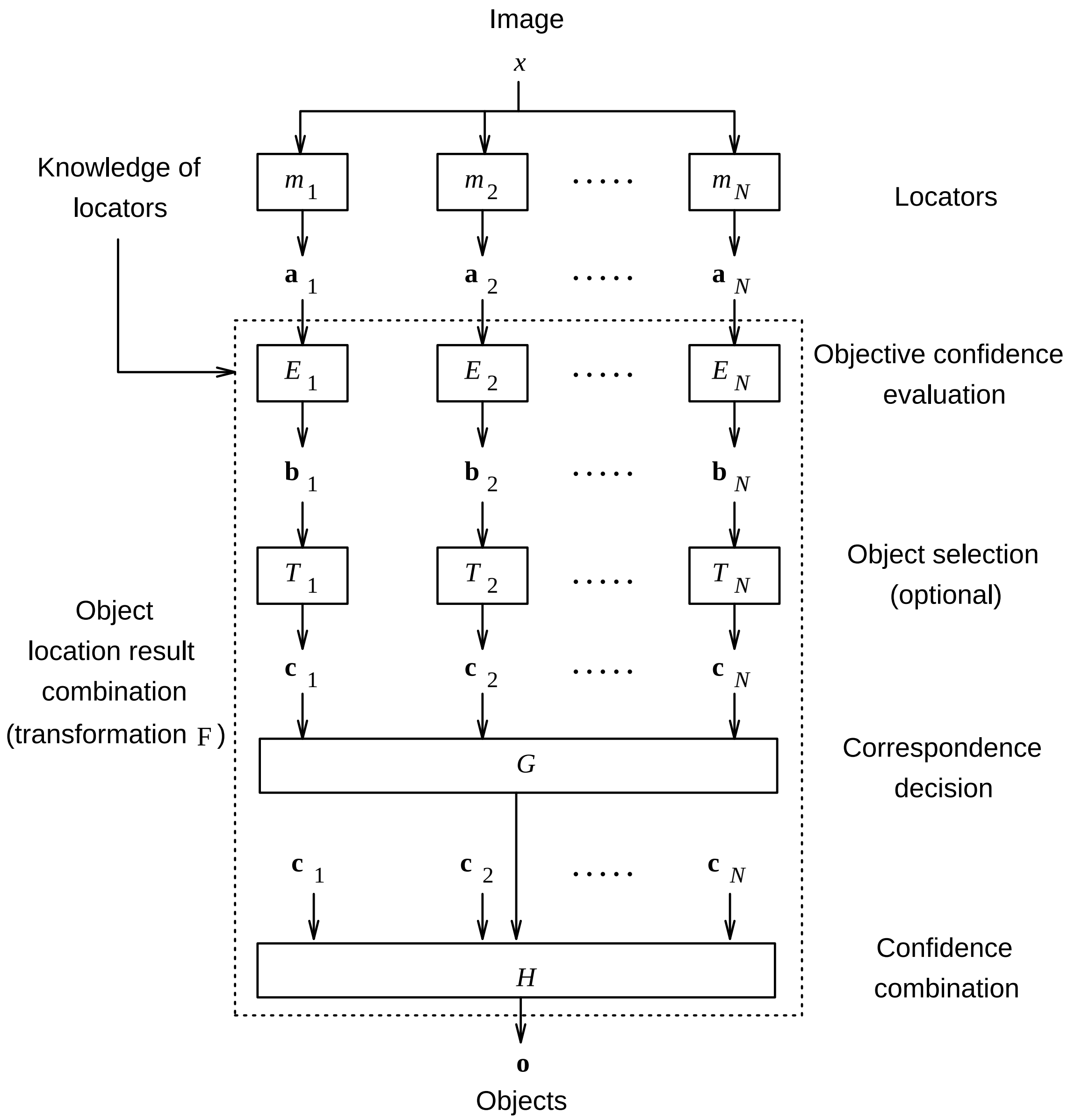


Figure 3.1: Components of proposed theory

forms associated with the three output levels:

$$m_i(x) = \mathbf{a}_i = \begin{cases} (o_{i1}) & \text{if abstract level} \\ (o_{i1}, \ldots, o_{iJ_i}) & \text{if rank level} \\ ((o_{i1}, u_{i1}), \ldots, (o_{iJ_i}, u_{iJ_i})) & \text{if measurement level} \end{cases} \tag{3.1}$$

where o_{ij} is the j-th candidate object generated by m_i, u_{ij} is m_i's measurement value associated with o_{ij}, and J_i is the number of objects generated by m_i. Thus, regardless of the output level, $\mathbf{a}_i$ is the raw output of m_i represented in a vector form.

Given this representation of the location results, we can represent the input and output of the locator combination problem succinctly by a transformation from raw output vectors to another vector:

$$\mathbf{o} = \mathcal{F}(\mathbf{a}_1, \ldots, \mathbf{a}_N) = ((o_1, v_1), \ldots, (o_K, v_K)) \tag{3.2}$$

where K is the number of objects in the final combination output, o_k is the k-th object in order of descending confidence assigned by $\mathcal{F}$, and v_k is $\mathcal{F}$'s confidence in o_k. Accordingly, it holds that $v_1 \geq \ldots \geq v_K$. No new object is generated by $\mathcal{F}$ but some objects can be eliminated by $\mathcal{F}$. Thus, $K \leq \sum_{i=1}^{N} |\mathbf{a}_i|$. Also, for any object $o_k, 1 \leq k \leq K$, there exists at least one object $o_{ij}, 1 \leq i \leq N, 1 \leq j \leq J_i$ such that $o_k = o_{ij}$.

A priori knowledge of locators is used by $\mathcal{F}$ as Figure 3.1 indicates. Such knowledge is extracted in various forms from the previous behavior of locators on a set of training images and is used in several places. For brevity, the transformations defined in this section do not represent the incorporation of the knowledge of locators explicitly by their input variables.

The first task of $\mathcal{F}$ is to represent the results of different locators in a consistent form. This is absolutely necessary for meaningful integration of the location results which might be at different information levels. We choose to convert them into the most informative form, the one used to represent locator output at the measurement level. This task of converting a locator output vector to another is accomplished by the following transformation:

$$\mathbf{b}_i = \mathcal{E}_i(\mathbf{a}_i) = ((o_{i1}, v_{i1}), \ldots, (o_{iJ_i}, v_{iJ_i})) \tag{3.3}$$

where v_{ij} is the *objective confidence* value for the object o_{ij}. Objective confidence for each locator output level is defined in Section 3.3. From now on, a confidence value means either an objective confidence value or a confidence value derived by a combination of objective confidence values, depending on the context. In particular, it is to be distinguished from a *measurement value* which is a value associated with an object output by a measurement-level locator and whose interpretation is unknown.

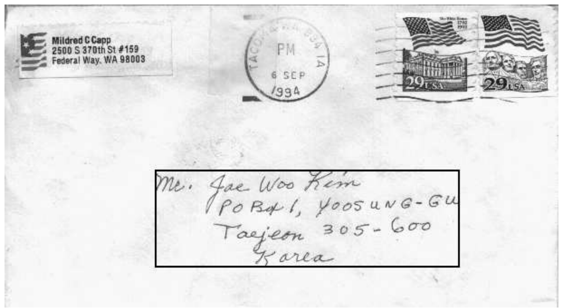
$\mathcal{E}_i$ derives objective confidence values for candidate objects (when $\mathbf{a}_i$ is at the abstract or rank level) or converts subjective measurement values to objective confidence values (when $\mathbf{a}_i$ is at the measurement level). $\mathcal{E}_i$ performs the appropriate kind of mapping according to the output level of m_i.

Since it helps to reduce the number of objects to consider as long as the true object is not missed, we can optionally select a subset of objects from the set of all objects output by m_i. An operation to select an appropriate number of top elements of the vector $\mathbf{b}_i$ does it:

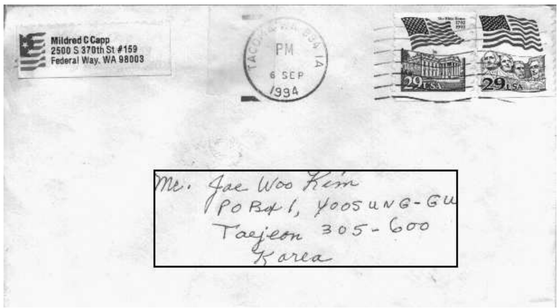
$$\mathbf{c}_i = \mathcal{T}_i(\mathbf{b}_i) = ((o_{i1}, v_{i1}), \ldots, (o_{iK_i}, v_{iK_i})) \tag{3.4}$$

where K_i is the number of selected objects generated by m_i. How many objects to select (the value K_i) can be determined by an appropriate thresholding operation on the confidence values $v_{i1}, \ldots, v_{iJ_i}$. If the number of objects produced by a locator tends to be small, we can choose to skip this operation entirely for the locator, resulting in $K_i = J_i$.

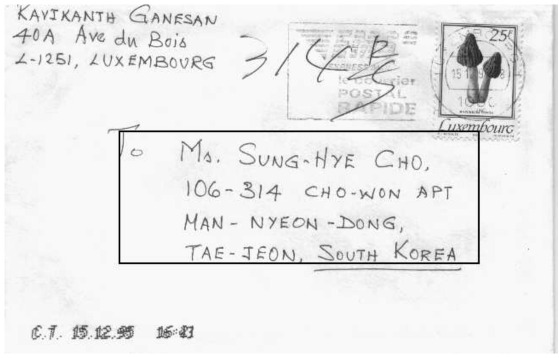
Before proceeding further, we define *object equivalence* and *object correspondence.*



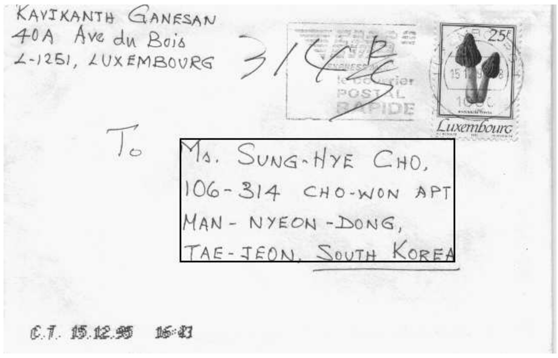
(a)



(b)



(c)



(d)

Figure 3.2: Examples of equivalent objects and corresponding objects: objects of (a) and (c) are output by one locator, and objects of (b) and (d) are output by another; objects of (a) and (b) are equivalent (and also corresponding), and objects of (c) and (d) are corresponding.

Definition 3.1 *Two objects are equivalent if they are output by different locators and occupy an identical spatial position in terms of pixel coordinates. An object is equivalent to itself.*

Definition 3.2 *Two objects correspond (to each other) if they are equivalent or if they are output by different locators and occupy different spatial positions, but in reality are two different segmentations of a single object. An object corresponds to itself.*

Figure 3.2 shows examples of equivalent objects and corresponding objects.

After we produce the vectors $\mathbf{c}_1, \ldots, \mathbf{c}_N$, we compare all the objects in them to derive, for each object, two sets which form the basis on which confidence value combination is done. This comparison operation is defined as

$$\begin{aligned} \mathcal{G}(\mathbf{c}_1, \ldots, \mathbf{c}_N) &= (\mathbf{E}_{11}, \ldots, \mathbf{E}_{1K_1}, \ldots, \mathbf{E}_{N1}, \ldots, \mathbf{E}_{NK_N}, \\ &\quad \mathbf{C}_{11}, \ldots, \mathbf{C}_{1K_1}, \ldots, \mathbf{C}_{N1}, \ldots, \mathbf{C}_{NK_N}) \end{aligned} \tag{3.5}$$

where $\mathbf{E}_{ij}$ is the *equivalence set* of the object o_{ij} and $\mathbf{C}_{ij}$ is the *correspondence set* of the object o_{ij}. They are both sets of object indices defined as

$$\mathbf{E}_{ij} = \{(p, q) \mid o_{ij} \text{ and } o_{pq} \text{ are equivalent}\}, \tag{3.6}$$

$$\mathbf{C}_{ij} = \{(p, q) \mid o_{ij} \text{ and } o_{pq} \text{ correspond}\}. \tag{3.7}$$

The equivalence set $\mathbf{E}_{ij}$ contains the indices of all objects that are equivalent to o_{ij}, that is, objects that are exact duplicates of o_{ij}. If two objects are equivalent their equivalence sets are identical. Since we use multiple locators, a single object can be output by different locators, perhaps with different ranks or measurement values. Many groups of equivalent objects will exist if the input image is easy to segment correctly. Otherwise, there will be few such groups. As confidence values for equivalent objects can be combined to derive a consensus decision on the object, identifying equivalent objects is important.

The correspondence set $\mathbf{C}_{ij}$ is the set of the indices of all objects that correspond to o_{ij}. Two equivalent objects correspond to each other, implying that for any object its equivalence set is a subset of its correspondence set, namely, for all (i, j), $\mathbf{E}_{ij} \subseteq \mathbf{C}_{ij}$. Since there are usually very few or no groups of equivalent objects for input images that are difficult to segment, we need to determine correspondence of objects. Proper determination as to whether two different objects output by different locators are actually an identical object is critical, because decisions on such objects can be combined directly. We refer to this problem as the *object correspondence problem.* It differs from the *visual correspondence problem*, which consists of matching all pairs of corresponding points from two images of the same scene, according to the definition in [29]. Hereafter, correspondence means object correspondence.

The last task of the transformation $\mathcal{F}$ of (3.2) is the computation of combined confidence values for the objects in $\mathbf{c}_1, \ldots, \mathbf{c}_N$ as follows:

$$\mathbf{o} = \mathcal{H}(\mathbf{c}_1, \ldots, \mathbf{c}_N, \mathcal{G}(\mathbf{c}_1, \ldots, \mathbf{c}_N)) = ((o_1, v_1), \ldots, (o_K, v_K)) \tag{3.8}$$

where K is the number of objects in the final combination output, o_k is the k-th object in order of descending confidence, and v_k is the newly computed confidence value for the object o_k. The above transformation uses the objective confidence values v_{ij} for $1 \leq i \leq N, 1 \leq j \leq K_i$, along with the equivalence sets and correspondence sets to compute new integrated confidence values. From a group of equivalent objects, only one object is output by $\mathcal{H}$, and thus $K \leq \sum_{i=1}^{N} |\mathbf{c}_i|$.

3.3 Objective Confidence

Adequate representation of decisions is of great importance in any decision combination problem. The requirements of a good representation scheme for location result combination are fairly obvious. Firstly, the scheme must be capable of representing the locator decisions without distortion and as accurately as possible. Secondly, it must allow for meaningful and systematic combination of location results.

In this section, we discuss how we represent location results at different levels in a uniform fashion using the concept of objective confidence. Specifically, we show, for each locator output information level, how the transformation $\mathcal{E}_i$ of (3.3) can be realized by probabilistic interpretation of location results.

3.3.1 Measurement Level

A measurement-level object locator assigns a measurement value to each object it generates as a candidate for the true object. Measurement values can indicate anything including distances, similarity measures, belief values, and subjective confidence assignments. In fact, they are values of any measurement the locator adopts to differentiate the likelihoods of objects being the true object. Measures of different attributes cannot be combined as is to produce a meaningful measure in locator combination.

Even if available locators output measurements of the same attribute, different locators often use different measurement ranges, scales, and distributions. Therefore, combining them as is is not useful either. Furthermore, even when two locators use the same measurement ranges and scales, it is still not clear whether two identical measurement values, one from each locator, signify the same amount of confidence indeed.

For these reasons, measurement values from different locators need to be normalized before we attempt to combine them. There are methods for doing it so that measurement values all have a standard range such as from 0 to 1 [9, 34]. However, those methods cannot be used for abstract or rank-level output, where no measurement values are available to begin with. Therefore, we do not normalize measurement values, but evaluate them regarding the probabilistic confidence they indicate, based on the previous locator behavior.

We use the notion of objective confidence, which we briefly mentioned in Section 3.2. This was used in [36] for combination of measurement values produced by multiple classifiers at the measurement level. We extend the concept so that it is applicable to object locators. The transformation $\mathcal{E}_i$ of (3.3) evaluates measurement values by converting u_{ij} values into v_{ij} values, using a function of measurement values defined as

$$e_i(u) = P(\text{object from } m_i \text{ with measurement value } u \text{ is the true object}). \tag{3.9}$$

This can be regarded as a probabilistic interpretation of measurement values from a particular locator.

Let $\mathbf{a}_i = ((o_{i1}, u_{i1}), \ldots, (o_{iJ_i}, u_{iJ_i}))$ be the measurement-level output of the locator m_i, as in (3.1). Actual evaluation of the probability value is based on a basic property of conditional probability and experimental knowledge of m_i:

$$\begin{aligned} e_i(u) &= P(o_{ij} \text{ is the true object} \mid u_{ij} = u) \\ &= \frac{P(o_{ij} \text{ is the true object, } u_{ij} = u)}{P(u_{ij} = u)}. \end{aligned} \tag{3.10}$$

This is the objective confidence value for the measurement value u of the locator m_i. Note that $u_1 \geq u_2$ does not necessarily imply $e_i(u_1) \geq e_i(u_2)$. Whether this monotonicity holds depends on the prediction accuracy of the measurement values from the locator m_i. However, we expect that measurement values output by a properly designed measurement-level locator will exhibit this property. We discuss three methods for estimating (3.10) and explain why we prefer logistic regression over other methods.

Direct Estimation

For direct estimation of (3.10), we count, in a training run, the frequency of the measurement value u being assigned to an object and the frequency of an object with the measurement value u being the true object. Calling the former count as N_u and the latter as $N_{u \wedge true}$, we can estimate the objective confidence value for u as

$$e_i(u) = \frac{N_{u \wedge true}}{N_u}. \tag{3.11}$$

An immediate problem with this simple method is that objective confidence values are defined only for those measurement values that were output at least once by m_i in the training run. That is, if N_u is zero, $e_i(u)$ is undefined. In addition, if N_u is very small, the value of $e_i(u)$ will be based on too small a number of statistical samples to be meaningful. These cases would happen if there were no or too few cases where the measurement value u was output while processing a set of training images.

Even if we use a very large number of training images, it is always possible that some measurement value is never generated in the training run. Therefore, this approach is feasible only if sufficient training data is used so that N_u is reasonably large for *all possible* u values. However, this stringent condition is nearly impossible to satisfy, because measurement values are usually on a continuous scale.

Linear Regression

One way to compute the objective confidence value for any possible measurement value is to use the training data to model by a function the relation between the measurement values and the probabilities that those values are assigned to a true object. We can then use the function to predict the objective confidence given a measurement value, because such a function exactly corresponds to e_i defined in (3.10).

The simplest relation that can exist between u and $N_{u \wedge true}/N_u$ is a linear relation. Assuming a linear relation, we can use the simple linear regression model

$$E(Y) = \beta_0 + \beta_1 X \tag{3.12}$$

where Y is the random variable whose expected value we want to estimate given a value of X. The regression coefficients β_0 and β_1 are determined from a set of observed data.

For our problem, let Y be 1 if a measurement value is given to a true object and 0 if it is given to a non-true object, and $P(Y = 1) = \pi$. Then the expected value of Y is

$$E(Y) = 1 \cdot P(Y = 1) + 0 \cdot P(Y = 0) = 1 \cdot \pi + 0 \cdot (1 - \pi) = \pi.$$

Therefore, we have from (3.12),

$$\pi = \beta_0 + \beta_1 X. \tag{3.13}$$

If we estimate π by the ratio used in direct estimation ($N_{u \wedge true}/N_u$), we can use this model to generate a function for computing objective confidence values. The regression coefficients can be determined by the *least squares* method, which is described in detail in Section A.1. Once we obtain the β_0 and β_1 values, we compute the objective confidence value for u as

$$e_i(u) = \beta_0 + \beta_1 u. \tag{3.14}$$

Unfortunately, the linearity assumption is not justified in many cases. In particular, when the value of the dependent variable (Y) is a binary response (0 or 1), the relation between the independent variable and the probability of it generating a response of 1 is frequently only *approximately* linear. When this is the case, we should use other regression functions which can better fit the given data. The following real-world example illustrates this point.

Example Table 3.1 shows an example of real data, where the object locator m_i outputs measurement values in the range $[0, 1]$. Since the measurement values are continuous, we need to group them in some way such that the estimates of π's are meaningful. We divide the range of measurements into ten equal-sized bins. All measurements output in a training run on 550 images are grouped into those bins, and for each bin, the mean value of measurements in it is taken as the value of X. For example, the fourth row of Table 3.1 says: 126 measurement values are between 0.3 and 0.4, their mean value is 0.3389, and four of them are given to the true object. Using the least squares method to determine the simple linear regression function, we obtain

$$e_i(u) = -0.2058 + 1.9187u$$

Table 3.1: Example of data for computing objective confidence from measurements by linear regression

$X = u$	N_u	$N_{u \wedge true}$	$\pi = N_{u \wedge true}/N_u$
0.0147	1851	0	0.0000
0.1410	406	0	0.0000
0.2420	214	1	0.0047
0.3389	126	4	0.0317
0.4472	53	7	0.1321
0.5400	51	21	0.4118
0.6434	90	61	0.6778
0.7443	81	65	0.8025
0.8444	56	50	0.8929
0.9200	6	5	0.8333

as the function to calculate objective confidence values for this particular locator. Figure 3.3 shows the result of fitting this regression function. Without resorting to goodness of fit tests, we see that the fitting is not satisfactory.

Logistic Regression

As linear regression is probably too simple to properly model the relation we are interested in, we investigate another regression model. When the dependent variable is binary, the shape of the response function is frequently *sigmoidal* [65]. Figure 3.4 shows such functions of the form

$$E(Y) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)} \tag{3.15}$$

called *logistic response* functions. If we let $E(Y) = \pi$ and make the transformation

$$\pi' = \ln\left(\frac{\pi}{1-\pi}\right) \tag{3.16}$$

we obtain from (3.15)

$$\pi' = \beta_0 + \beta_1 X. \tag{3.17}$$

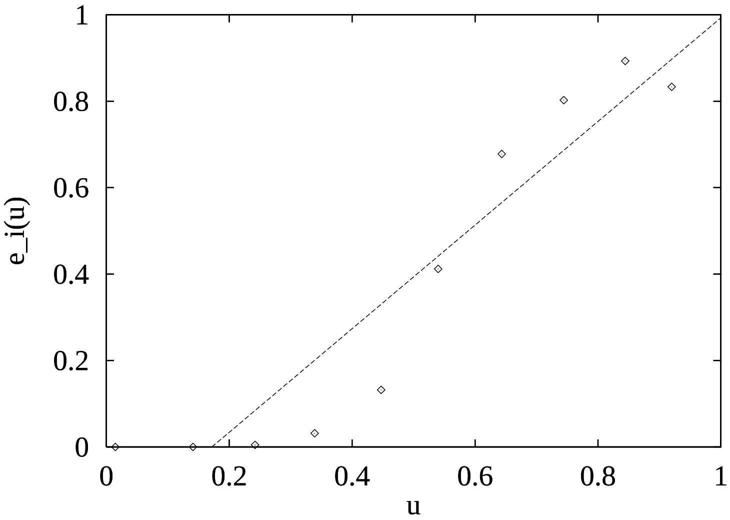


Figure 3.3: Plot of data in Table 3.1 and the fitted linear regression function

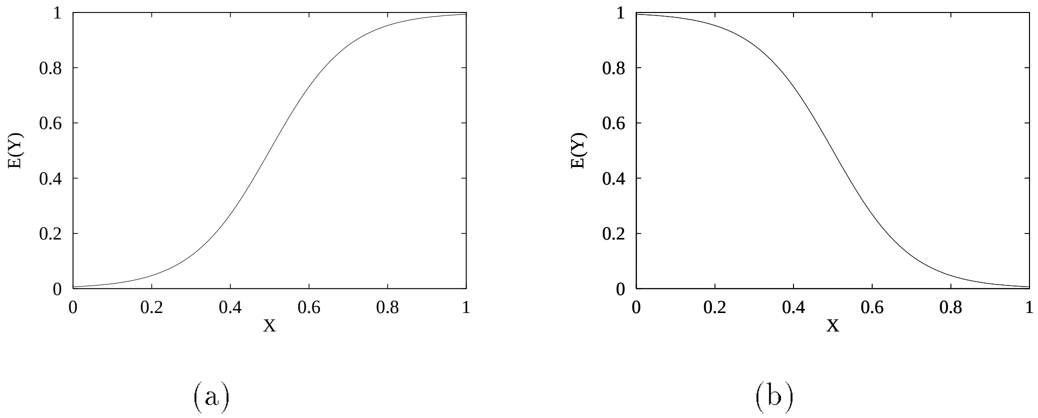


Figure 3.4: Example of logistic response functions: (a) monotonic increasing, $E(Y) = \frac{\exp(-5+10X)}{1+\exp(-5+10X)}$; (b) monotonic decreasing, $E(Y) = \frac{\exp(5-10X)}{1+\exp(5-10X)}$.

Table 3.2: Example data for computing objective confidence from measurements by logistic regression

$X = u$	N_u	$N_{u \wedge true}$	$\pi = \frac{N_{u \wedge true}}{N_u}$	$\pi' = \ln\left(\frac{\pi}{1-\pi}\right)$
0.0147	1851	0	0.0003	−8.2164
0.1410	406	0	0.0012	−6.6983
0.2420	214	1	0.0047	−5.3613
0.3389	126	4	0.0317	−3.4177
0.4472	53	7	0.1321	−1.8827
0.5400	51	21	0.4118	−0.3567
0.6434	90	61	0.6778	0.7436
0.7443	81	65	0.8025	1.4018
0.8444	56	50	0.8929	2.1203
0.9200	6	5	0.8333	1.6094

The transformation (3.16) is called the *logit transformation* of π and linearizes (3.15).

For our problem, let Y be the same as in linear regression. Then we can use the logistic regression model (3.17) to estimate the objective confidence given a measurement value, by estimating π in the same way as in linear regression and making the logit transformation. The regression coefficients can be estimated using the *weighted least squares* method, described in detail in Section A.1. Once the coefficients are estimated, we can compute the objective confidence value for u as

$$e_i(u) = \frac{\exp(\beta_0 + \beta_1 u)}{1 + \exp(\beta_0 + \beta_1 u)}. \tag{3.18}$$

Both theoretical prediction and experimental trials suggest that logistic regression is a proper model for explaining the relation we are interested in: measurement values and the probabilities that they are assigned to a true object.

Example Table 3.2 lists the same data as in Table 3.1, except the addition of the logit transformation results (π'). Also, the π's were modified when $\pi = 0$, such that $\pi = 1/(2N_u)$, because the logit transformation is undefined in those cases [65,

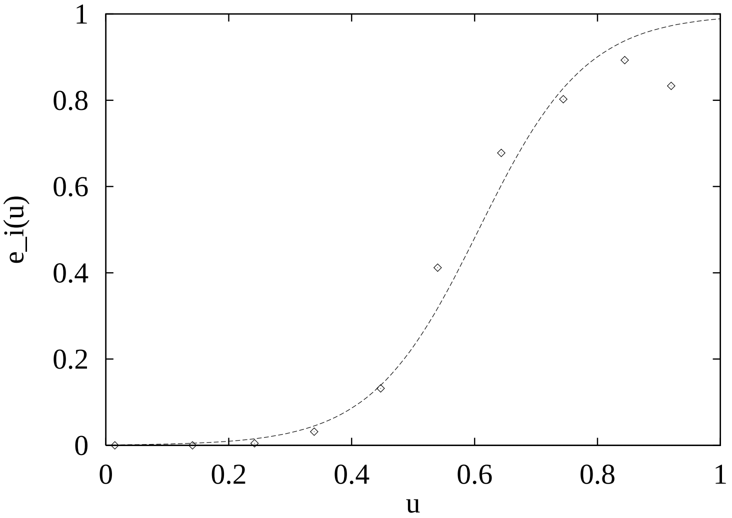


Figure 3.5: Plot of data in Table 3.2 and the fitted logistic regression function

page 586]. Using the weighted least squares method, we have

$$e_i(u) = \frac{\exp(-6.9256 + 11.4137u)}{1 + \exp(-6.9256 + 11.4137u)}$$

as the function to compute object confidence values for the locator. Figure 3.5 shows the result of fitting the regression function. We see that the fitting is satisfactory except for the last data point (0.9200, 0.8333) which is based only on six samples.

3.3.2 Rank Level

A rank-level object locator outputs an ordered list of objects, but does not provide measurement values. Equivalently, it assigns a rank to each object it generates as a candidate for the true object. Unlike the case of measurement-level output, we can combine ranks directly because they are always on the same integer scale. However, two identical ranks from two different locators do not necessarily signify the same level of confidence.

One of the advantages of using objective confidence is that it can be naturally extended to handle the case of rank-level output as well as measurement-level output.

For rank-level output, objective confidence values are derived from ranks of objects. The transformation $\mathcal{E}_i$ of (3.3) evaluates rank values by converting them to objective confidence values, using a function of ranks defined as

$$e_i(r) = P(\text{object from } m_i \text{ at rank } r \text{ is the true object}). \tag{3.19}$$

This function derives objective confidence values from ranks based on a priori knowledge of the performance of m_i.

Let $\mathbf{a}_i = (o_{i1}, \ldots, o_{iJ_i})$ be the rank-level output of the locator m_i. Then, $e_i(r)$ can be evaluated using a method similar to the one used in (3.10) for measurement values:

$$\begin{aligned} e_i(r) &= P(o_{ij} \text{ is the true object} \mid j = r) \\ &= \frac{P(o_{ij} \text{ is the true object, } j = r)}{P(j = r)}. \end{aligned} \tag{3.20}$$

This is the objective confidence value for the rank r of the locator m_i.

The evaluation can be done by counting, from experimental results on training images, the frequency of the rank r being assigned to an object and the frequency of an object at rank r being the true object. Calling the former frequency as N_r and the latter as $N_{r \wedge true}$, we can compute the objective confidence value of r as

$$e_i(r) = \frac{N_{r \wedge true}}{N_r}. \tag{3.21}$$

We see that rank values which only had relative interpretation are converted to objective confidence values with the same probabilistic interpretation as the case of measurement-level locators.

Rank values are discrete integers, whereas measurement values are usually continuous. In general, there are much fewer different values that ranks can take on than measurements can. Accordingly, the possibility of the frequency of a rank value being

Table 3.3: Example of computing objective confidence from ranks

r	N_r	$N_{r\wedge true}$	$e_i(r) = N_{r\wedge true}/N_r$
1	550	285	0.5182
2	550	6	0.0109
3	483	1	0.0021
4	290	1	0.0034
5	172	0	0.0000

too small, such that direct estimation is infeasible, is much smaller than that of a measurement value being so. Therefore, we can directly estimate objective confidence values for rank-level locators.

Example Table 3.3 shows a real example of computing objective confidence values from rank values. In the example, $e_i(r)$ values are very close to zero for ranks greater than 1. In general, we expect that $e_i(r)$ values will drop drastically for $r > 1$, if m_i is a properly designed rank-level locator.

3.3.3 Abstract Level

An abstract-level object locator outputs a unique object, without any more information. Such a locator has full confidence in the unique object. This can be considered as equivalent to assigning the highest measurement value to the object in case of a measurement-level locator, and to assigning the first rank to the object in case of a rank-level locator, with the number of objects being one in both cases.

Therefore, an objective confidence value can be defined naturally for a locator of this type based on its performance:

$$e_i = P(\text{object from } m_i \text{ is the true object}). \tag{3.22}$$

Note that e_i is a constant that is associated with *any* object m_i outputs.

Let $\mathbf{a}_i = (o_{i1})$ be the abstract-level output of the locator m_i. To obtain e_i, we only need to rule out the reject cases, leading to the following evaluation of e_i:

$$\begin{aligned} e_i &= P(o_{i1} \text{ is the true object} \mid \text{not reject}) \\ &= \frac{P(o_{i1} \text{ is the true object, not reject})}{P(\text{not reject})}. \end{aligned} \tag{3.23}$$

This is the objective confidence value for the locator m_i.

Obviously, e_i is the same as the location success rate of m_i when reject cases are excluded. That is, if we call the number of times an object is output by m_i as N_a and the number of times such an object is the true object as $N_{a \wedge true}$, the objective confidence value for m_i is

$$e_i = \frac{N_{a \wedge true}}{N_a}. \tag{3.24}$$

The e_i value has the same probabilistic interpretation as the cases of a measurement-level locator and a rank-level locator. The accuracy of e_i directly depends on the appropriateness of the training image set used.

3.4 Object Equivalence and Correspondence

Object equivalence and object correspondence were defined in Section 3.2. In this section we describe why the concepts are needed and methods of finding equivalent objects and corresponding objects.

Consensus decision in its ordinary sense refers to decision making based on multiple decisions on a single set of entities, such that multiple decisions on a single entity are available for direct combination to generate a consensus decision on that entity. A good example in this category is classification result combination, where decisions are made on a single set of classes and multiple decisions on a single class can be combined to derive a consensus decision on that class.

There is one fundamental difference between these types of decision combination problems and the object locator combination problem. In the latter, multiple deci-

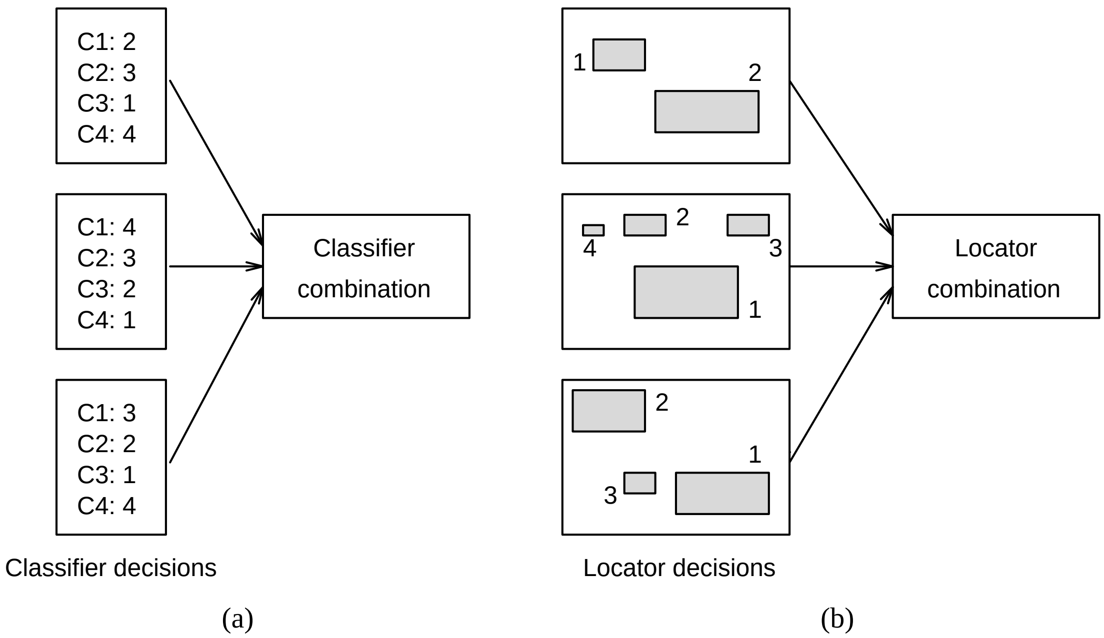


Figure 3.6: Difference between classifier combination and locator combination: (a) three rank-level classifiers are combined for a four-class problem, where all decisions are on the class set; (b) three rank-level locators are combined, where each decision is on a distinct object set.

sions on a single entity (object in this case) are *not* immediately available in most cases. Instead, multiple decisions on *similar* objects are available for use. This is because each locator makes decisions on its own set of objects generated by its own segmentation algorithm. As a result, often different object boundaries are generated for the same object. Figure 3.6 illustrates this difference by comparing examples of classifier combination and locator combination.

Therefore, in order to fully exploit the advantages of combining multiple location results, we need an appropriate method of deciding whether two objects from different locators are actually a single object. This was referred to as the object correspondence problem in Section 3.2. Object equivalence and object correspondence are essential concepts for developing a solution to the problem.

Consider a collection of objects represented by $\mathbf{c}_i$'s defined in (3.4). Given such a collection, we should solve the correspondence problem for each pair of objects (o_{ij}, o_{pq}), where $1 \leq i, p \leq N, i \neq p, 1 \leq j \leq K_i$, and $1 \leq q \leq K_p$.

In this section, we discuss a method of deciding object equivalence, how to measure object correspondence, and two ways to use correspondence measures. In effect, we describe practical ways to realize the transformation $\mathcal{G}$ of (3.5).

3.4.1 Deciding Equivalence

The simplest approach to deciding whether two objects are equivalent is to compare their spatial positions. The equivalence decision rule is: decide that two objects from different locators are actually a single object, if and only if they occupy exactly the same spatial regions in terms of pixel coordinates. The rule says that two equivalent objects spatially overlap completely.

More concretely, let o_{ij} and o_{pq} be two objects output by different locators, and $\mathbf{P}_{ij}$ and $\mathbf{P}_{pq}$ be the sets of pixels contained in o_{ij} and o_{pq}, respectively. Then, o_{ij} and o_{pq} are considered to be equivalent objects, if and only if $\mathbf{P}_{ij} = \mathbf{P}_{pq}$.

The main advantage of this approach is that we can be sure that decisions on equivalence based on the above rule are always correct. However, different locators usually produce different spatial regions for one actual object because of different segmentation algorithms used. As a result, the cases in which two objects satisfy the stringent condition of identical spatial position are rare, except when the input image is easy to segment correctly.

Accordingly, if we consider only object equivalence, there will be only a small number of cases where multiple decisions on a single object can be combined; most objects will not have an equivalent object. For this reason, although conceptually simple, object equivalence alone has little use in practice, unless locators make decisions on a single set of objects generated by one segmentation algorithm. Therefore, we should also consider the cases where two objects do not occupy the same spatial region yet correspond to a single object. This leads us to consider object correspondence as well as object equivalence.

The results of equivalence decisions are represented in equivalence sets defined in (3.6). If two objects are equivalent, their equivalence sets will be the same. For example, if o_{ij} and o_{pq} are equivalent $\mathbf{E}_{ij} = \mathbf{E}_{pq}$.

3.4.2 Measuring Correspondence

Given two objects produced by different locators, their correspondence cannot be determined as easily as their equivalence can be. This is because there is no simple rule to determine object correspondence. Therefore, we first measure the *degree of correspondence* between two objects. Such measures can be used effectively to guide the decision on their correspondence. We propose two approaches to measuring object correspondence: one based on the contents of objects and the other based on the spatial positions of objects. We discuss each approach and explain why we prefer the latter.

Content-Based Approach

The first approach is to compare the contents of objects. The basic idea of this approach is that we can determine whether two objects are indeed the same object by analyzing their contents and checking how similar they are. To do this, we extract a set of visual conditions or features from each object to be compared. Then we compare those sets of features to measure their similarity. This similarity value is converted to a degree value ranging from 0 to 1, which becomes the degree of correspondence.

Specifically, let o_{ij} and o_{pq} be two objects output by different locators, and $\mathbf{x}_{ij}$ and $\mathbf{x}_{pq}$ be the two feature vectors extracted from o_{ij} and o_{pq}, respectively. Then the degree of correspondence between o_{ij} and o_{pq} can be obtained by computing the similarity of $\mathbf{x}_{ij}$ and $\mathbf{x}_{pq}$ and converting it to a degree. For example, denoting the degree of correspondence as $cor_{ij,pq}$, we can compute it as follows:

$$cor_{ij,pq} = f(d(\mathbf{x}_{ij}, \mathbf{x}_{pq})) \tag{3.25}$$

In the equation, d is a distance function and f is a function which converts distances to correspondence degrees.

This approach converts the problem of comparing image objects to one of comparing features and as a result greatly reduces the problem complexity. However, there are two main difficulties with it. The first one is that it is difficult to convert similarity measures to degrees so that degree values have a clear interpretation. This is a normalization problem. The second one is that it is nearly impossible for our theory to specify what features to use for comparison. The feature selection task should be left up to the application, and in fact it will be done better in the context of a particular application domain than in a general theory.

Position-Based Approach

The content-based approach focuses on what objects contain without regard to where objects are. An alternative approach is to rely on spatial positions of objects to compute degrees of correspondence. This approach can be realized by relaxing the rule used to decide object equivalence, to allow for some mismatch of positions of two objects instead of requiring complete overlap. This extension is natural since object correspondence is a broader concept than object equivalence.

This idea leads us to consider the degree of overlap between two objects as the degree of their correspondence. Given two objects from different locators, we first measure the area of overlapping region, the region shared by the two objects. This area needs to be normalized by a reasonable method so as to be usable as a degree of overlap.

Let o_{ij} and o_{pq} be the two generic objects under consideration, $area_{ij}$ and $area_{pq}$ be their areas, respectively, and $area_{ij \cap pq}$ be the area of the overlapping region. Then, there are two simple ways to normalize $area_{ij \cap pq}$:

1. $area_{ij \cap pq}/min(area_{ij}, area_{pq})$

2. $area_{ij \cap pq}/max(area_{ij}, area_{pq})$

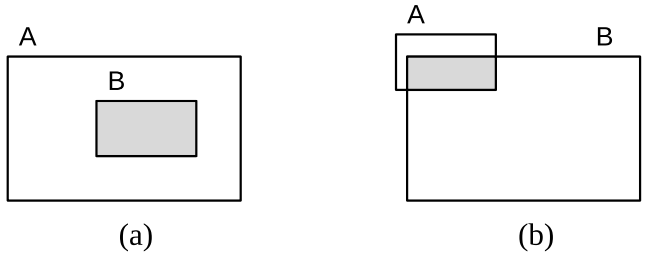


Figure 3.7: Examples of overlapping objects: in both (a) and (b), normalizing the overlap area with respect to the area of the smaller object will result in too high a degree of overlap, while normalizing with respect to the area of the larger object will result in too low a degree.

These formulas do produce a degree of overlap between 0 and 1, but are not adequate measures unless the sizes of the two overlapping objects are quite similar.

Figure 3.7 shows two cases of overlapping objects. In Figure 3.7(a), we would obtain the overlap degree of 1 if we use the first formula above, and a small degree if we use the second one, both of which are clearly inadequate. In Figure 3.7(b), the first formula would result in too big a degree and the second one would result in too small a degree. In fact, both formulas generate a degree that results from ignoring the part of one object outside the overlapping region. Put differently, the two formulas ignore some part of one object and are biased.

We could also normalize $area_{ij \cap pq}$ with respect to the average of the areas of the two objects to alleviate this problem. That is, we can use $area_{ij \cap pq}/((area_{ij} + area_{pq})/2)$. However, this formula can still generate an unexpected degree of overlap depending on relative sizes of the two objects.

We now propose our method of measuring the degree of correspondence between two objects. Let $\mathbf{P}_{ij}$ and $\mathbf{P}_{pq}$ be the sets of pixels contained in o_{ij} and o_{pq}, respectively. In addition, let $area_{ij \cup pq}$ be the area occupied by $\mathbf{P}_{ij} \cup \mathbf{P}_{pq}$ and $area_{ij \cap pq}$ be the area occupied by $\mathbf{P}_{ij} \cap \mathbf{P}_{pq}$. Consider the following measure:

$$cor_{ij,pq} = \frac{area_{ij \cap pq}}{area_{ij \cup pq}} = \frac{area_{ij \cap pq}}{area_{ij} + area_{pq} - area_{ij \cap pq}}. \tag{3.26}$$

This measure is simply the ratio of the overlapping region to the total area occupied

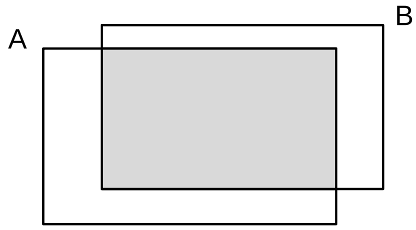


Figure 3.8: Example of computing the correspondence degree: let the areas of A, B, and $A \cap B$ be 20, 19, and 16, respectively; the correspondence degree is $16/(20+19-16) = 16/23 = 0.6957$.

by the two objects. It can be considered as a case of the unified measure of uncertainty introduced by Wang [95]:

$$\mu_C(x) = \frac{w^+}{w} \tag{3.27}$$

where C is the fuzzy concept to be measured, w is the weight of all relevant (either positive or negative) evidence, and w^+ is the weight of all positive evidence. In the measure (3.26), we can regard the correspondence of the two objects as the fuzzy concept and the pixels in the objects as pieces of evidence, where all pixels are equally weighted. Figure 3.8 shows an example of computing the ratio. If the value of $cor_{ij,pq}$ is 1, o_{ij} and o_{pq} not only correspond to each other but also are equivalent.

The measure yields a high value for two objects that occupy slightly different regions but actually correspond to one object, which can result from different segmentation algorithms. This happens often for an object which is slightly difficult to segment correctly. On the contrary, it generates a low value for two objects that share a region but are indeed different objects, which might result from a fairly difficult image to segment. In either case, unlike the previous measures, it is not biased towards one object and can indicate exactly how much of the total area is shared by the two objects.

3.4.3 Deciding Correspondence

We can use the degree of correspondence between two objects to decide whether they correspond or not. The decision method we propose is to apply thresholding on correspondence degrees. Given two objects o_{ij} and o_{pq} with the correspondence degree $cor_{ij,pq}$, the decision rule is:

$$o_{ij} \text{ and } o_{pq} \begin{cases} \text{correspond} & \text{if } cor_{ij,pq} \geq th_{ip} \\ \text{do not correspond} & \text{otherwise} \end{cases} \tag{3.28}$$

where th_{ip} is the threshold value for the pair of locators m_i and m_p.

To determine the threshold value th_{ip}, we collect all pairs of objects, one from m_i and the other from m_p, that have a non-zero correspondence degree. For all such pairs of objects that arise from an experimental run on a set of training images, we determine *manually* whether the two objects constituting a pair are actually a single object or not. Then we have the data consisting of pairs where each pair consists of a correspondence degree and the decision on whether the two objects that had the correspondence degree are the same or different. This training data can be represented separately by the two sets defined as

$$\mathbf{D}_{ip} = \{(cor = cor_{ij,pq}, D_{cor}) \mid o_{ij} \text{ and } o_{pq} \text{ are different objects}\}, \tag{3.29}$$

$$\mathbf{S}_{ip} = \{(cor = cor_{ij,pq}, S_{cor}) \mid o_{ij} \text{ and } o_{pq} \text{ are the same objects}\}, \tag{3.30}$$

where D_{cor} is the number of times two objects that are different have the correspondence degree cor, and S_{cor} is the number of times two objects that are the same have the correspondence degree cor.

Once we have the above two sets of data, we find the threshold value as follows. The thresholding operation should minimize the possibility of wrong correspondence decisions. Consider the two sets

$$\mathbf{D}'_{ip} = \{(cor, D_{cor}) \mid (cor, D_{cor}) \in \mathbf{D}_{ip} \wedge cor \geq th_{ip}\} \tag{3.31}$$

$$\mathbf{S}'_{ip} = \{(cor, S_{cor}) \mid (cor, S_{cor}) \in \mathbf{S}_{ip} \wedge cor < th_{ip}\}. \tag{3.32}$$

The first set represents the wrong decisions which would be made if th_{ip} were used as the threshold, by classifying non-corresponding objects as corresponding. On the contrary, the second one represents wrong decisions made by classifying corresponding objects as not corresponding.

The actual number of cases for which wrong decisions would be made is

$$W_{ip} = \sum_{(cor, D_{cor}) \in \mathbf{D}'_{ip}} D_{cor} + \sum_{(cor, S_{cor}) \in \mathbf{S}'_{ip}} S_{cor}. \tag{3.33}$$

Therefore, we select the value of cor which minimizes W_{ip} as the threshold value th_{ip}. That value will lead to the *optimal* correspondence decision results on the training data.

The results of correspondence decisions are represented in correspondence sets defined in (3.7). During confidence value combination, these sets are used to select groups of objects where confidence values of objects in a group are combined.

Example Table 3.4 illustrates the correspondence decision procedure by an example. The table contains the correspondence degree data obtained by applying the locators m_1 and m_2 on a set of images. Of all the pairs of locators with a correspondence degree greater than 0, one from m_1 and the other from m_2, $\sum_{(cor, D_{cor}) \in \mathbf{D}_{12}} D_{cor} = 282$ pairs consist of different objects and $\sum_{(cor, S_{cor}) \in \mathbf{S}_{12}} S_{cor} = 122$ pairs consist of the same object segmented slightly differently. From the data, we obtain th_{12} value of 0.8295. Deciding correspondence using this threshold value results in 9 cases of wrong decisions, 3 from $\mathbf{D}_{12}$ and another 6 from $\mathbf{S}_{12}$. Thus the correspondence decision error rate using the threshold value th_{12} on this training data is $9/404 = 2.23\%$.

3.5 Locator Relevance

In this section, we introduce *locator relevance* and describe its use in our theory. The relevance of a locator to a given image is a measure of the relevance of the locator

Table 3.4: Example of correspondence decisions

$\mathbf{D}_{12}$ (degree, count)	Decision by $th_{12} = 0.8295$	$\mathbf{S}_{12}$ (degree, count)	Decision by $th_{12} = 0.8295$
(0.0001, 1)	degree $< th_{12}$:	(0.6178, 1)	degree $< th_{12}$:
(0.0004, 1)	Not corresponding	(0.7586, 1)	Not corresponding
(0.0008, 1)	(correct, 279 cases)	(0.7637, 2)	(wrong, 6 cases)
(0.0010, 1)		(0.7775, 1)	
(0.0018, 1)		(0.8102, 1)	
(0.0029, 1)		(**0.8295**, 1)	degree $\geq th_{12}$:
(0.0055, 1)		(0.8313, 1)	Corresponding
(0.0181, 1)		(0.8487, 1)	(correct, 116 cases)
(0.0237, 1)		(0.8497, 1)	
(0.0242, 1)		(0.8497, 1)	
(0.0277, 1)		(0.8650, 1)	
(0.0282, 1)		(0.8791, 1)	
(0.0300, 1)		⋮	
(0.0307, 1)		(0.9915, 1)	
(0.0361, 1)		(0.9917, 3)	
(0.0390, 1)		(0.9922, 2)	
(0.0508, 1)		(0.9924, 3)	
(0.0519, 1)		(0.9927, 1)	
(0.0520, 1)		(0.9928, 1)	
(0.0541, 1)		(0.9929, 1)	
(0.0598, 1)		(0.9931, 1)	
⋮		(0.9932, 1)	
(0.7452, 1)		(0.9934, 1)	
(0.7500, 1)		(0.9935, 1)	
(0.7511, 1)		(0.9936, 1)	
(0.7605, 1)		(0.9937, 2)	
(0.7612, 1)		(0.9940, 2)	
(0.7954, 1)		(0.9943, 1)	
(0.7989, 1)		(0.9949, 1)	
(0.8246, 1)		(0.9965, 1)	
(0.8293, 1)		(0.9974, 2)	
(0.8672, 1)	degree $\geq th_{12}$:	(0.9976, 1)	
(0.8998, 1)	Corresponding	(0.9977, 1)	
(0.9020, 1)	(wrong, 3 cases)	(1.0000, 9)	

to location of the true object in the image. The exact attribute that the relevance measure of a locator indicates varies depending on what measure is used. However, it can be generally thought of as the weight, expected worth, or reliability of the locator towards the final decision made by a combination process. We can also regard it as the importance of objective confidence values of objects generated from the locator.

Since we make no assumption as to the performance levels of locators involved, we expect that different locators will exhibit different performance levels. Therefore, treating locators differently during combination based on their expected relevance is better than treating all of them equally.

We show below that locator relevance can be assigned statically or computed dynamically. We discuss two contrasting methods of image set partitioning to compute locator relevance dynamically. Then we describe dynamic locator selection as it relates to dynamic locator relevance.

3.5.1 Static Relevance

The simplest approach to computing the relevance of a locator in a combination process is to use its performance figures. Suppose we have the location results of the locator m_i on a set of training images. The static relevance of the locator is defined as

$$rel_i = P(m_i\text{'s top choice is the true object}). \tag{3.34}$$

We can calculate the probability using the rates of top-choice location success, error, and reject. If we let these figures be r_s, r_e, and r_r, respectively, the relevance of the locator m_i can be computed by

$$rel_i = \frac{r_s}{1 - r_r}. \tag{3.35}$$

This computation is repeated for each locator to obtain the relevance values of all locators.

The above formula simply measures the overall reliability of the decisions made by m_i for a training image set. We call the relevance value computed in this way as static because a fixed relevance value is used for a locator. Some combination methods require this type of relevance values for all locators to be combined, as described in Section 3.6.

3.5.2 Dynamic Relevance

We can let the relevance of a locator vary dynamically according to the input image. This means that the relevance of a locator is determined based on the characteristics of the input image as well as on its previous performance on a set of training images.

Our scheme for computing relevance dynamically uses a partitioning of training images. We partition the training images using a set of mutually exclusive conditions computable from the images. The relevance of a locator is computed for each partition separately using (3.35). Given a new input image, we first determine which partition the image belongs to using the same conditions, then use the relevance of that partition as the relevance of the locator to the image. This process is illustrated in Figure 3.9.

Partition sizes affect how dynamic the relevance computation process will be. At one extreme, if the whole set of training images is one partition, relevance computation will be static. At the other extreme, if each training image constitutes its own partition, the partitioning is meaningless. Thus the method of partitioning images is of great importance. Image partitioning can be based on characteristics of input images or location results.

Image-Based Approach

We can use characteristics of images to partition a training image set. We define a set of image characteristics that divide the training images into several partitions. The actual characteristics can be based on the observed behavior of locators.

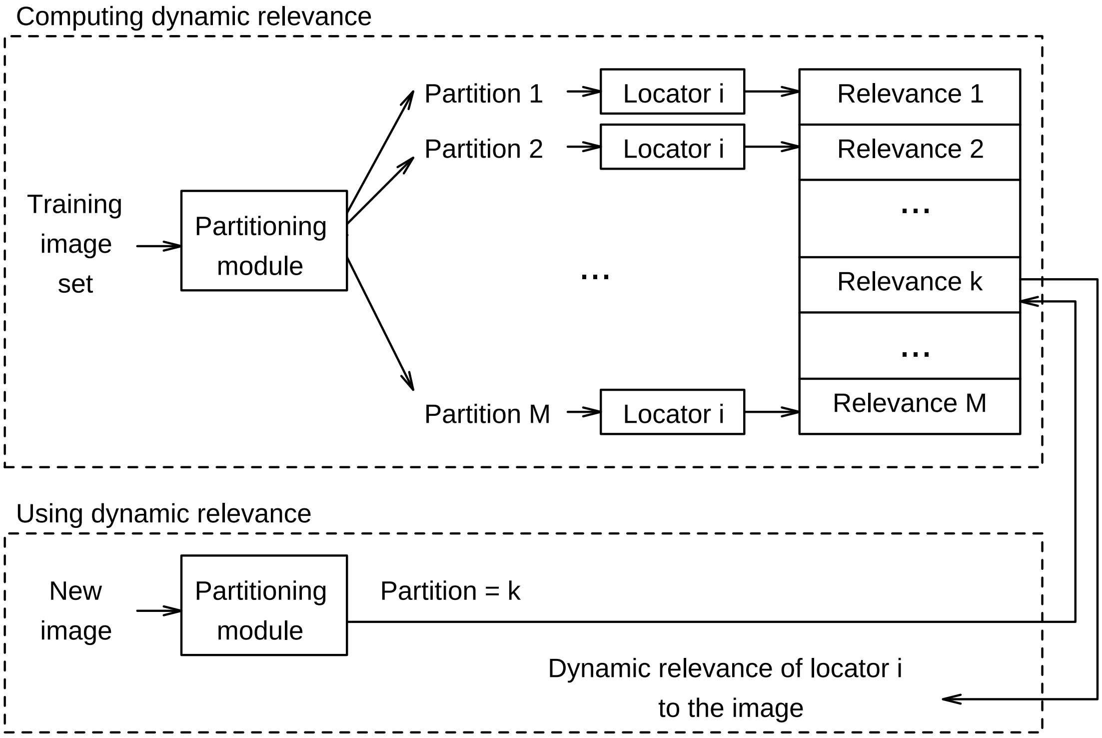


Figure 3.9: Computing and using dynamic relevance

We use multiple locators under the general premise that different locators work well for different types of images. Therefore, for a locator that works well for certain types of images, we attempt to extract computable characteristics of those types of images, and use the characteristics to define a *characteristic class*. We expect that the relevance of the locator will be generally high for that characteristic class and low for other classes. The number of such classes is not necessarily the same as the number of locators. There is no simple rule to define characteristic classes suitably.

For an input image, we determine which characteristic class it belongs to by computing various characteristics which define the classes. The relevance values for the characteristic class is used as the relevance values for the image. That is, we estimate the dynamic relevance of the locator m_i to the image x as

$$\begin{aligned} rel_i(x) &= P(m_i\text{'s top choice is the true object} \mid x) \\ &= P(m_i\text{'s top choice is the true object} \mid c(x)) \qquad (3.36) \end{aligned}$$

Table 3.5: Possible agreement classes for three locators

Agreement class	Locators		
	m_1	m_2	m_3
1			
2	○	○	
3		○	○
4	○		○
5	○	○	○

where $c(x)$ is the characteristic class the image x belongs to. Designing $c(x)$ properly is usually a complicated task that depends heavily on the application context. Therefore, we prefer a more general approach to this one.

Result-Based Approach

We can use certain conditions on the location results to partition training images. A useful criterion is the agreement among top-ranked objects. Locators tend to agree on top objects for simple input images, but tend to disagree for complex input images. Consequently, the extent of top object agreement can be used as a rough indicator of the input image complexity.

Given a collection of location results from multiple locators, we compare the top-choice objects to decide whether some of them agree. To decide whether two objects agree, we use the method of deciding correspondence described in Section 3.4.3. Namely, two objects that correspond to each other are considered to agree.

Suppose that we have N locators whose results are available. Possible types of agreements are: none, two, ..., and all of them agree(s). Thus we can identify $1 + \sum_{k=2}^{N-1} {}_N C_k + 1 = 2 + \sum_{k=2}^{N-1} {}_N C_k$ different types of top object agreement. Calling these types as *agreement classes*, we determine locator performance separately for each agreement class. Table 3.5 shows the agreement classes for three locators, where locators whose top objects agree are marked by ○'s. There are $2 + {}_3C_2 = 5$ agreement classes and $3 \times 5 = 15$ different dynamic relevance values.

For a new input image, we determine which agreement class the location results belong to, that is, which locators generated top objects that agree. This classification is *not* related to whether a top object is the true object or not. The relevance values of locators for the agreement class are used as the relevance values for the image. That is, the dynamic relevance of the locator m_i to the image x is

$$\begin{aligned} rel_i(x) &= P(m_i\text{'s top choice is the true object} \mid x) \\ &= P(m_i\text{'s top choice is the true object} \mid a(m_1(x), \ldots, m_N(x))) \end{aligned} \tag{3.37}$$

where $a(m_1(x), \ldots, m_N(x))$ is the agreement class of the image x. Depending on the location problem, we only need to define exactly what an agreement of location results is. This approach is more general than the image-based approach and we adopt it for computing dynamic relevance of locators.

3.5.3 Dynamic Selection of Locators

It is useful to be able to predict ahead of time which locator would be the most effective one for processing an input image. If such prediction could always be made reliably, there would be no need to use all available locators and combine their results; we can simply apply the most effective locator for the input image and use its results. This task is known as *dynamic selection*, and doing it literally and accurately is extremely hard.

Using locator relevance, we can solve the dynamic locator selection problem to some extent. If we determine static relevance based on top-choice performance, we can simply sort and output the top choices of locators using their relevance values as the key. In fact, this method amounts to ranking the locators and always applying the ranking to combine the locator outputs.

If we are using dynamic relevance, we can first decide which image partition the given image belongs to. Then we can use the dynamic relevance values of locators for that partition to order the top choices of the locators. This way, different locator rankings are selected dynamically according to the input image. In case there are more than one locator with the highest relevance, a tie-breaking scheme is needed, such as using the objective confidence of the top-choice objects as the second key.

Therefore, the concept of dynamic locator relevance to an image naturally allows for dynamic selection of locators. In fact, if we compute locator relevance dynamically, no additional mechanism except for a tie-breaking scheme is required for dynamic selection.

3.6 Confidence Value Combination

This section proposes methods for combining objective confidence values to generate a consensus decision. Specifically, we describe how the transformation $\mathcal{H}$ of (3.8) can be implemented.

Correspondence sets play a vital role here since they indicate which objects can be treated as if they were one object for the purpose of decision combination. Confidence values of corresponding objects can be combined to generate an integrated confidence value. On the other hand, equivalence sets are used after the final confidence values are generated and objects are sorted, to remove all but one duplicates of an object. There is no need to output multiple equivalent objects.

In (3.8), $\mathcal{H}$ is defined as a transformation of a group of vectors into another vector. However, the essence of the combination denoted by $\mathcal{H}$ is the computation of a new confidence value for each object in the locator output. In other words, all the combination methods to be discussed can be characterized by a function

$$\hat{v}_{ij} = f(v_{i_1 j_1}, \ldots, v_{i_L j_L}) \tag{3.38}$$

where $\hat{v}_{ij}$ is the new confidence value for the object o_{ij} and $L = |\mathbf{C}_{ij}|$ is the number of objects corresponding to o_{ij}. This computation is repeated for every candidate object and the objects are sorted in descending order of $\hat{v}_{ij}$ values to generate the final combination output. If there is a tie, the original objective confidence values are used to break it. From a group of equivalent objects, only the object with the highest objective confidence is output.

3.6.1 Highest Confidence

When locators are highly specialized for a particular type of images, it is worthwhile to focus on the best-case behavior of locators. For most images, if we know that there is at least one locator that performs well, we can attempt to take the decision of that locator, rather than combine confidence values literally. The highest confidence method achieves this goal by selecting the highest value among the confidence values given to a group of corresponding objects as the final confidence value for the objects.

For the object o_{ij} with a correspondence set $\mathbf{C}_{ij}$, the new confidence value for it, denoted as $\hat{v}_{ij}$, is obtained by a maximum operation:

$$\hat{v}_{ij} = \max_{(p,q)\in\mathbf{C}_{ij}} (v_{pq}). \tag{3.39}$$

Candidate objects are sorted by the $\hat{v}_{ij}$ values to derive an ordered list of objects $((o_1, v_1), \ldots, (o_K, v_K))$.

Besides the training necessary to compute objective confidence values, this method does not need additional training. It does not use locator relevance explicitly, either.

Example Suppose that we have three locators, m_1, m_2, and m_3, which are measurement-level, rank-level, and abstract-level locators, respectively. Let their outputs for an input image be

$$\begin{aligned}
\mathbf{a}_1 &= ((o_{11}, 0.8290), (o_{12}, 0.7691), (o_{13}, 0.5840), (o_{14}, 0.4934)),\\
\mathbf{a}_2 &= ((o_{21}, 0.7346), (o_{22}, 0.3289), (o_{23}, 0.2005)),
\end{aligned}$$

$$\mathbf{a}_3 = ((o_{31}, 0.6248)),$$

and $\mathbf{C}_{13} = \{(1,3),(2,2),(3,1)\}$. Then the combined confidence for o_{13} is

$$\hat{v}_{13} = \max_{(p,q)\in\mathbf{C}_{13}}(v_{pq}) = \max(v_{13}, v_{22}, v_{31}) = \max(0.5840, 0.3289, 0.6248) = 0.6248.$$

3.6.2 Confidence Summation

A widely used method of information integration is linear combination which is the sum of certain values multiplied by weights. In our theory, it amounts to multiplying the confidence values to be combined by appropriate weights and summing them to compute the combined confidence value, which takes the form

$$\hat{v}_{ij} = \sum_{(p,q)\in\mathbf{C}_{ij}} w_{pq} v_{pq} \tag{3.40}$$

where $\hat{v}_{ij}$ is the new confidence value for o_{ij}, $\mathbf{C}_{ij}$ is the correspondence set of o_{ij}, and w_{pq} is the weight of the confidence value v_{pq}.

When there is no a priori knowledge of locator performance available, we let all the weights be equal. If we know that some locators are likely to perform better than others for a given input image, we use the weights to reflect this knowledge. However, the v_{pq} values in the above formula are objective confidence values, meaning that they already have differences in locator performance incorporated in them. Thus, using locator relevance values as weights by letting $w_{pq} = rel_p$ would be redundant. Therefore, the confidence summation method uses the formula

$$\hat{v}_{ij} = \sum_{(p,q)\in\mathbf{C}_{ij}} v_{pq}. \tag{3.41}$$

This produces, as the new confidence value of o_{ij}, the plain sum of objective confidence values for the objects that correspond to o_{ij}.

Like the highest confidence method, this method does not require training besides that necessary to compute objective confidence values. Locator relevance values are not used either. Intuitively, it gives the highest confidence to the object that collectively acquired the largest amount of confidence.

Example For the same locator outputs as in the example of Section 3.6.1, the final confidence for o_{13} is

$$\hat{v}_{13} = \sum_{(p,q) \in \mathbf{C}_{13}} v_{pq} = v_{13} + v_{22} + v_{31} = 0.5840 + 0.3289 + 0.6248 = 1.5377.$$

3.6.3 Weighted Confidence Summation

The correspondence degree between two objects is a measure of the similarity between the two objects. Therefore, we can use it as the degree to which one object influences the other during combination. A simple method of using this observation is to use correspondence degrees themselves in the combination process.

We can extend the confidence summation method in order to incorporate correspondence degrees as weights of confidence values to be combined. That is, the weighted confidence summation method computes the new confidence value for the object o_{ij} by

$$\hat{v}_{ij} = \sum_{(p,q) \in \mathbf{C}_{ij}} cor_{ij,pq} v_{pq}. \tag{3.42}$$

The above equation computes the new confidence for o_{ij} by summing up the confidence values for corresponding objects scaled by the individual correspondence degrees.

Like the confidence summation method, no training besides what is necessary to compute objective confidence values is required. Locator relevance values are not explicitly used either.

Example We use the same data as in the example of Section 3.6.1. In addition, assume that the correspondence degrees involving o_{13} are $cor_{13,13} = 1, cor_{13,22} = 0.8$, and $cor_{13,31} = 0.7$. Then the final confidence for o_{13} is

$$\begin{aligned}\hat{v}_{13} &= \sum_{(p,q)\in\mathbf{C}_{13}} cor_{ij,pq} v_{pq} = cor_{13,13} v_{13} + cor_{13,22} v_{22} + cor_{13,31} v_{31} \\ &= 1 \times 0.5840 + 0.8 \times 0.3289 + 0.7 \times 0.6248 \\ &= 1.2845.\end{aligned}$$

3.6.4 Probabilistic Confidence Estimation

The methods presented so far use predetermined formulas to compute a new confidence value. An alternative is to estimate the new confidence value by observing the behavior of locators on a set of training images. That is, we statistically estimate the values of the function f of (3.38) using training data.

Specifically, given a list of confidence values of corresponding objects, we estimate the probability that the object in question is the true object. Suppose that $(v_{i_1 j_1}, \ldots, v_{i_L j_L}), L = |\mathbf{C}_{ij}|$ is the list of confidence values for the objects corresponding to o_{ij}. From the training data, we count the number of times the locators $m_{i_1}, \ldots, m_{i_L}$ output L objects with confidence values $v_{i_1 j_1}, \ldots, v_{i_L j_L}$, respectively, and at the same time the objects correspond to one object. We also count how many times o_{ij} is the true object. Calling the former quantity as $N_{(v_{i_1 j_1}, \ldots, v_{i_L j_L})}$ and the latter as $N_{(v_{i_1 j_1}, \ldots, v_{i_L j_L}) \wedge true}$, we can use the following ratio to estimate the probability of o_{ij} being the true object:

$$P(o_{ij} \text{ is the true object} \mid (v_{i_1 j_1}, \ldots, v_{i_L j_L})) = \frac{N_{(v_{i_1 j_1}, \ldots, v_{i_L j_L}) \wedge true}}{N_{(v_{i_1 j_1}, \ldots, v_{i_L j_L})}}. \tag{3.43}$$

To compute the above ratio, we should count the frequency of a combination of events where an event is the occurrence of a particular confidence value. Therefore, unless we have a very large amount of training data, the frequencies will usually be too small for the ratio to be an adequate probability estimates. It is better to use an estimation function in such cases than to use the ratio directly.

In Section 3.3.1, we used logistic regression for computing objective confidence values from measurements, as we wanted to estimate the expected value of a binary dependent variable. The probability (3.43) can be estimated similarly, the only difference being that we have multiple independent variables for the current problem. We assume that the relation between the probability that an object is a true object ($Y = 1$) and the confidence values for its corresponding objects is sigmoidal. Then we can use a multiple logistic regression model:

$$E(Y) = \pi = \frac{\exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_L X_L)}{1 + \exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_L X_L)} \tag{3.44}$$

to estimate the probability. To find the regression coefficients, we use the observed data consisting of tuples $(X_1, \ldots, X_L, N_{(X_1,\ldots,X_L)}, \pi)$, where $x_1, \ldots, x_L$ are the confidence values for L objects output by L locators that correspond to an object and $\pi = \frac{N_{(X_1,\ldots,X_L)\wedge true}}{N_{(X_1,\ldots,X_L)}}$ is the the directly estimated probability that the combination of confidence values $(X_1, \ldots, X_L)$ indicates a true object. By performing logit transformation and using the weighted least squares method, we can estimate the values of $\beta_0, \beta_1, \ldots, \beta_L$. Then we can compute the new confidence for o_{ij} as

$$\hat{v}_{ij} = \frac{\exp(\beta_0 + \beta_1 v_{i_1 j_1} + \cdots + \beta_L v_{i_L j_L})}{1 + \exp(\beta_0 + \beta_1 v_{i_1 j_1} + \cdots + \beta_L v_{i_L j_L})}. \tag{3.45}$$

The regression coefficients need to be computed separately for each possible combination consisting of more than one locator, because the influence of a locator on the final confidence value varies depending on which combination it is a member of. For instance, if we have three locators m_1, m_2, and m_3, we should compute the coefficients for each of the locator combinations $\{m_1, m_2\}, \{m_1, m_3\}, \{m_2, m_3\}$, and $\{m_1, m_2, m_3\}$. In general, when we use N locators the number of coefficient sets required is $\sum_{k=2}^{N} {}_N C_k$. The appropriate coefficient set is selected by which locators produced objects corresponding to the object for which we want to compute the new confidence value.

Table 3.6: Example of data for probabilistic confidence estimation

X_1	X_2	$N_{(X_1,X_2)}$	$N_{(X_1,X_2)\wedge true}$	$\pi = \frac{N_{(X_1,X_2)\wedge true}}{N_{(X_1,X_2)}}$	$\pi' = \ln\left(\frac{\pi}{1-\pi}\right)$
0.0236	0.0109	60	0	0.0083	−4.7791
0.1567	0.0109	6	0	0.0833	−2.3979
0.2532	0.0109	3	0	0.1667	−1.6094
0.4442	0.0109	3	0	0.1667	−1.6094
0.5428	0.0109	2	0	0.2500	−1.0986
0.6135	0.0109	1	0	0.5000	0.0000
0.7314	0.0109	2	1	0.5000	0.0000
0.8591	0.0109	1	1	0.5000	0.0000
0.9358	0.0109	3	3	0.8333	1.6094
0.0292	0.5182	27	9	0.0185	−3.9703
0.1564	0.5182	5	1	0.2000	−1.3863
0.2282	0.5182	2	1	0.5000	0.0000
0.3221	0.5182	1	1	0.5000	0.0000
0.4580	0.5182	7	4	0.5714	0.2877
0.5518	0.5182	4	4	0.8750	1.9459
0.6707	0.5182	8	6	0.7500	1.0986
0.7449	0.5182	2	2	0.7500	1.0986
0.8373	0.5182	16	15	0.9375	2.7081
0.9441	0.5182	27	26	0.9630	3.2581

Unlike the maximum confidence method and the confidence summation method, this method needs extensive training to estimate the regression coefficients for all possible locator combinations. It does not use locator relevance values explicitly, but they are incorporated into the regression coefficients which play the role of weights for confidence values.

Example Table 3.6 lists the real data from two locators, one at the measurement level and the other at the rank level. The data is obtained by gathering confidence values for two objects, one from the first locator and the other from the second, that correspond to one object. The pairs of objective confidence values are grouped into ten equal-sized bins, based on the first confidence value from the measurement-level locator. The X_1 value is the mean of the first confidence values in each group. For each group of confidence value pairs, we count how many pairs are in it ($N_{(X_1,X_2)}$) and how many times the pairs indicate a true object ($N_{(X_1,X_2)\wedge true}$). Using the weighted least squares method, we obtain

$$\hat{v}_{ij} = \frac{\exp(-4.1384 + 5.9586 v_{i_1 j_1} + 3.1817 v_{i_2 j_2})}{1 + \exp(-4.1384 + 5.9586 v_{i_1 j_1} + 3.1817 v_{i_2 j_2})}$$

as the function to compute the new confidence value for an object to which two objects from the two locators correspond.

3.6.5 Fuzzy Integral

The class of nonlinear functionals called fuzzy integrals offers a means to combine information by taking into consideration the reliability of the information sources. It was first developed by Sugeno [88] and has been used in several information fusion problems [3, 4, 12, 90]. Fuzzy integrals are described in detail in Section A.2.

To use fuzzy integrals in our theory, we first need to determine what the fuzzy densities will be. For location result combination, locators are information sources and the locator relevance values can be used as the fuzzy density values for individual sources. We proposed a practical method for computing dynamic locator relevance based on agreement classes in 3.37. We expect that it will be better to use dynamic relevance values as fuzzy densities rather than static relevance values. The confidence values provided by locators are considered as measurements provided by information sources.

For an object o_{ij} for which we want to compute a combined confidence value, let $\mathbf{M}$ be the set of locators that generated objects corresponding to o_{ij}. That is,

$$\mathbf{M} = \{m_p \mid (p, q) \in \mathbf{C}_{ij}, \text{ for some } q\}. \tag{3.46}$$

We rearrange the locators in $\mathbf{M}$ so that the confidence values they generated for their corresponding objects are in descending order. In other words, we sort the elements of $\mathbf{M}$ as

$$\mathbf{M} = \{m_{p_1}, \ldots, m_{p_L}\}, \quad L = |\mathbf{C}_{ij}| \tag{3.47}$$

such that $v_{p_1 q_1} \geq \cdots \geq v_{p_L q_L}$. We also define a subset consisting of a given number of first elements of this set:

$$\mathbf{A}_k = \{m_{p_1}, \ldots, m_k\} \subseteq \mathbf{M}. \tag{3.48}$$

Using the above definitions, we apply the fuzzy integral to compute the final confidence for the object o_{ij} as

$$\hat{v}_{ij} = \max_{1 \leq k \leq |\mathbf{C}_{ij}|} [\min(v_{p_k q_k}, g(\mathbf{A}_k)] \tag{3.49}$$

where $g(\mathbf{A}_k)$ is the dynamic relevance of the subset of locators $\mathbf{A}_k$.

This method looks for the best agreement between the reliability of a set of sources ($g(\mathbf{A}_k)$) and the best security decision the set provides ($v_{p_k q_k}$). The minimum operation amounts to looking for the agreement, in that it takes the smaller of the safest decision from and the reliability of a set of sources. The maximum operation chooses the best of these agreement values. Intuitively, if the best confidence is known to be from a highly reliable source, it is likely that the confidence will be accepted. However, if it is from a marginally reliable source, it is possible for a lower confidence from a more reliable source to be accepted.

Example We use the same data as in the example of Section 3.6.1:

$$\begin{aligned}
\mathbf{a}_1 &= ((o_{11}, 0.8290), (o_{12}, 0.7691), (o_{13}, 0.5840), (o_{14}, 0.4934)), \\
\mathbf{a}_2 &= ((o_{21}, 0.7346), (o_{22}, 0.3289), (o_{23}, 0.2005)), \\
\mathbf{a}_3 &= ((o_{31}, 0.6248)),
\end{aligned}$$

Table 3.7: Example of confidence and relevance values

Locator set	Confidence ($v_{p_k q_k}$)	Relevance (g)
ϕ		0
$\{m_3\} = \mathbf{A}_1$	$0.6248 = v_{p_1 q_1} = v_{31}$	0.6248
$\{m_1\}$	$0.5840 = v_{p_2 q_2} = v_{13}$	0.4205
$\{m_2\}$	$0.3289 = v_{p_3 q_3} = v_{22}$	0.7346
$\{m_3, m_1\} = \mathbf{A}_2$		0.8055
$\{m_3, m_2\}$		0.9405
$\{m_1, m_2\}$		0.8731
$\{m_3, m_1, m_2\} = \mathbf{A}_3 = \mathbf{M}$		1

and $\mathbf{C}_{13} = \{(1,3),(2,2),(3,1)\}$. To compute the final confidence for o_{13}, we rearrange this data as shown in Table 3.7, because $v_{31} \geq v_{13} \geq v_{22}$. We assume that $g^1 = 0.4205, g^2 = 0.7346$, and $g^3 = 0.6248$, which results in $0.1930\lambda^2 + 1.0306\lambda + 0.7799 = 0$ and $\lambda = -0.9128$. The additional relevance values for the sets of multiple locators listed in the table are calculated using g^1, g^2, g^3 and λ. Thus the new confidence for the object o_{13} is

$$
\begin{aligned}
\hat{v}_{13} &= \max_{1 \leq k \leq 3} [\min(v_{p_k q_k}, g(\mathbf{A}_k)] \\
&= \max[\min(v_{31}, g(\mathbf{A}_1)), \min(v_{13}, g(\mathbf{A}_2)), \min(v_{22}, g(\mathbf{A}_3))] \\
&= \max[\min(0.6248, 0.6248), \min(0.5840, 0.8055), \min(0.3289, 1))] \\
&= 0.6248.
\end{aligned}
$$

3.6.6 Summary of Combination Methods

We presented five methods for combination of confidence values. All of them require some initial training to compute objective confidence values. In addition to the initial training, they have different requirements as summarized in Table 3.8. Besides the initial training, the probabilistic confidence estimation method requires more training to determine the estimation function for the combined confidence value. Locator relevance values are not used explicitly except in the fuzzy integral method. All of

Table 3.8: Requirements of confidence combination methods

Method	Additional training required	Locator relevance used explicitly	Correspondence sets required
Highest confidence	no	no	yes
Confidence summation	no	no	yes
Weighted confidence summation	no	no	yes
Probabilistic confidence estimation	yes	no	yes
Fuzzy integral	no	yes	yes

the combination methods need correspondence sets as the bases of deciding what confidence values to combine.

3.7 Performance Evaluation

In Section 3.1, we mentioned that we regard a locator combination as successful if its location performance is better than that of any locator used. In this section, we discuss actual methods of evaluating locator combination performance and compare them.

Assume that we have N locators, m_i for $1 \leq i \leq N$, and M test images. Let S_i be the number of images on which m_i located the true object as the top choice. Likewise, let S be the number of images on which the combination of m_i's located the true object as the top choice. Let p_i be the performance of m_i and p be that of the combination of m_i's. The performance of a locator can be represented by its top-choice success rate on a set of test images, such that

$$p_i = \frac{S_i}{M} \text{ for } 1 \leq i \leq N \text{ and } p = \frac{S}{M}. \tag{3.50}$$

We consider a combination of m_i's as successful if

$$p > p_i \text{ for all } 1 \leq i \leq N \tag{3.51}$$

or equivalently

$$p > \max_{1 \leq i \leq N}(p_i). \tag{3.52}$$

Otherwise, using the m_i for which p_i is the highest is better than combining m_i's.

If a combination is deemed to be successful, we will be interested in evaluating how successful it is. We note that the *location performance* of the combination, p, cannot be a good measure of the *combination performance*. This is because p depends on three factors:

1. How effective the combination method is. This is what we are actually interested in.

2. How good the individual locators perform. These are the qualities of individual locators.

3. How complementary the locators are. This is the quality of the collection of locator outputs. The *complementarity* of two locators, say m_1 and m_2, can be defined as

$$P(\text{only one of } m_1 \text{ and } m_2 \text{ outputs the true object as a candidate}) \tag{3.53}$$

We would want to minimize the effect of the second and third factors above in evaluating the combination performance, because those factors collectively represent the goodness of the input to the combination method. The location performance improvements over individual locators, namely, p/p_i for $1 \leq i \leq N$, are sometimes used as the indicator of the combination performance. However, this also relates to all of the three factors listed above.

In Section 3.1, we stated that a combination method is *not* to generate a new object that is not output by any of the locators to be combined. A direct consequence of this condition is that it is *impossible* for the combination to output the true object, unless it is output by at least one locator. Therefore, these cases should be ruled out when evaluating the combination performance.

Given N locators, let us call the number of test images for which *at least one* of the N locators outputs the true object as $M' \leq M$. Then, M' is the maximum number of images for which a combination of the N locators can output the true object. Therefore we propose the ratio

$$p' = \frac{S}{M'}. \tag{3.54}$$

as an appropriate measure of the combination performance. The number M' varies depending on the performance of individual locators and the complementarity of their outputs. Therefore, using the ratio minimizes the undesirable effect of inputs to the combination on the measure of the combination performance.

Figure 3.10 illustrates the difference between p' and other performance measures. The figure tells us that p' is a stable measure that has less to do with the individual locator performance and the complementarity of locator outputs, but has more to do with the combination method itself. On the other hand, to use p or p/p_i, it is probably necessary to supplement the performance figures with a measure of the complementarity of the locators for a given set of test images. The probability (3.53) and its extension to cases of more than two locators can be used for this purpose.

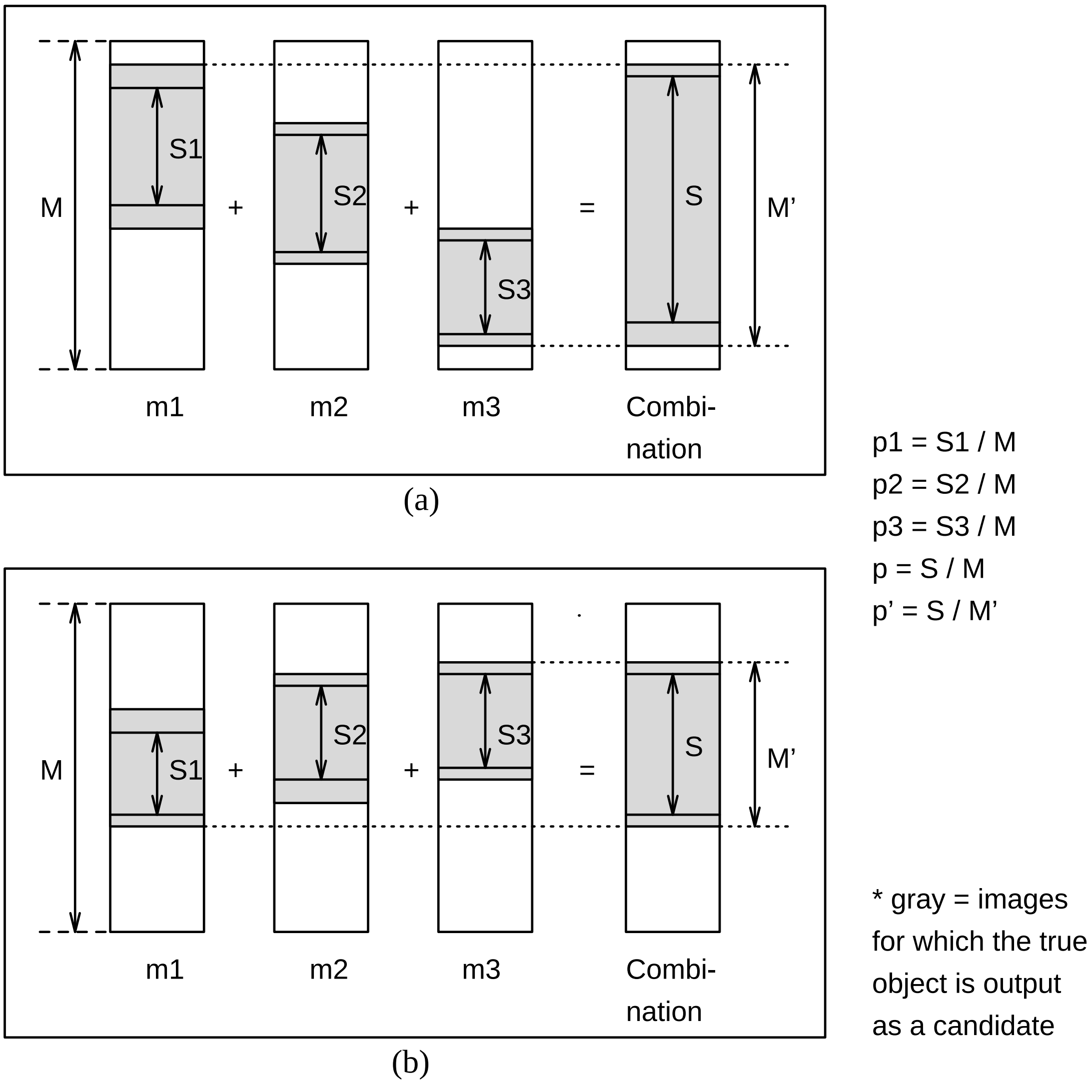


Figure 3.10: Evaluating combination performance: in (a), the complementarity of locators is relatively high, and $p, p/p_1, p/p_2, p/p_3$ will be high; in (b), the complementarity is relatively low, and $p, p/p_1, p/p_2, p/p_3$ will be low; p' is much less influenced by the quality of the combination input.

Chapter 4

A Multiple Locator Algorithm for Address Block Location

In this chapter, we describe an algorithm for address block location that uses multiple object locators and combines their results. Section 4.1 defines the address block location problem, describes its significance, and discusses the constraints we impose on the problem. Section 4.2 presents the flow of the algorithm and gives a comparison of it with previous approaches to the problem. Sections 4.3–4.5 describe the three address block locators employed in the algorithm in detail, respectively.

4.1 Address Block Location Problem

The Problem

The address block location problem is defined as follows. Given an input image of a mail piece, determine the position of the destination address block in terms of pixel coordinates. The output can be a unique block, a set of ranked candidate blocks, or a set of candidate blocks with measurement values, depending on the type of the locator used. Regardless of the locator type, a perfect solution would always output the address block as the top choice. Figure 4.1 shows the three possible types of

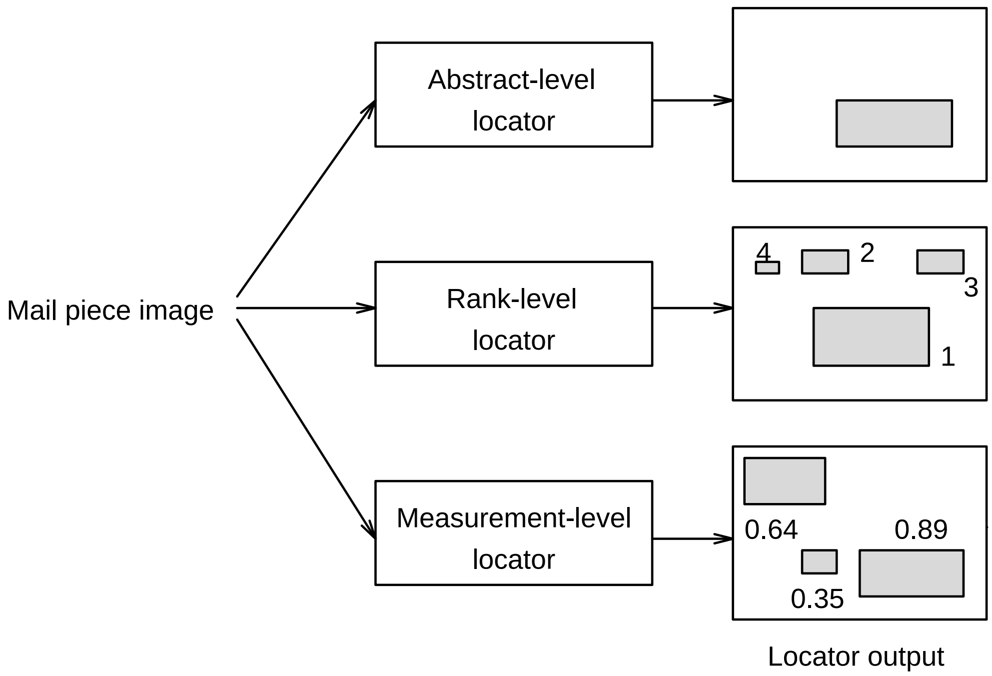


Figure 4.1: Types of address block location outputs: gray regions are candidate blocks.

address block location outputs.

Significance

In automatic mail sorting, a significant portion of optical character recognizer (OCR) failures is attributed to its inability to correctly locate the portion of the mail piece image that contains the destination address. To increase the rate of successful automatic sorting, those mail pieces that are rejected by the OCR should be processed by an address recognition unit (ARU).

The first subsystem of the ARU is an address block location subsystem, which locates the destination address and sends it to an address interpretation subsystem. Accordingly, a good solution to the address block location problem is significant to postal automation. More detail on the role of the ARU and the address block location subsystem in postal automation is given in [70]. The problem is also a challenging document object location problem for which a number of solutions have been proposed, as we reviewed in Section 2.2.

Constraints

We place a few constraints on the problem. These are necessary for us to focus our research on the core of the address block locator combination problem rather than try to handle all types of mail piece images that can be encountered.

1. The class of mail pieces we deal with is restricted to *letter mail*.
2. The mail piece image is *correctly oriented*.
3. *Only one* destination address exists on a mail piece.

Without the first constraint, we should consider flats and irregular parcels and pieces (IPPs) as well. They have relatively irregular mail piece face structures, such that many different locators or a complex locator need to be designed to handle them. This contradicts our goal of investigating how to combine simple locators effectively. Furthermore, focusing on letter mail is more useful than attempting to deal with all mail classes, because the letter mail class is far larger in quantity than any other mail classes.

The second constraint implies that the mail piece is correctly oriented at the time of image acquisition. Thus we exclude the cases where the image is upside-down, rotated or severely skewed. This eliminates the need for checking the possibility of wrongly placed mail pieces, which is not what we are interested in. A technique to deal with incorrectly oriented mail pieces in address block location is described in [93, 94].

The third constraint is realistic because a correctly addressed mail piece should have only one destination address. By this constraint, we also exclude mail piece images without a destination address. This constraint agrees with the first assumption on the document object location problem we stated in Section 3.1: there is only one true object in any input image.

4.2 Multiple Locator Algorithm

4.2.1 Algorithm Description

Figure 4.2 illustrates the flow of the multiple locator algorithm for address block location. It shows the major components of the algorithm. A mail piece image is input to the algorithm and preprocessed. The details of the preprocessing are described in Section 5.1. The preprocessing result in the form of connected components is input to multiple locators.

Among various image features that influence the process of address block location, one distinct feature is the printing method of the destination address. There can be locators designed for machine-printed addresses and handwritten addresses, respectively, and ones not aimed at a particular printing method. We use three locators, a machine-printed address block locator, a handwritten address block locator, and a clustering-based locator, and combine their results.

The locator outputs are at the measurement level, abstract level, rank level, respectively, as the difference in locator output forms in Figure 4.2 indicates. Given a set of locator outputs, the algorithm evaluates the objective confidence values for all candidates in the set, so that we have uniform representations of location results from multiple locators. An appropriate evaluation method is used depending on the output information level.

Since two sets of candidate blocks output by two different locators usually have very few cases of equivalent blocks, we need to determine if a block in one set corresponds to any block in the other set. This task is done by the correspondence decision module.

Using the uniformly represented locator outputs and the correspondence decision results, the algorithm combines the confidence values of corresponding blocks to derive a final set of candidate blocks with combined confidence values. Hence, the algorithm is a direct implementation of the theory of document object locator combination

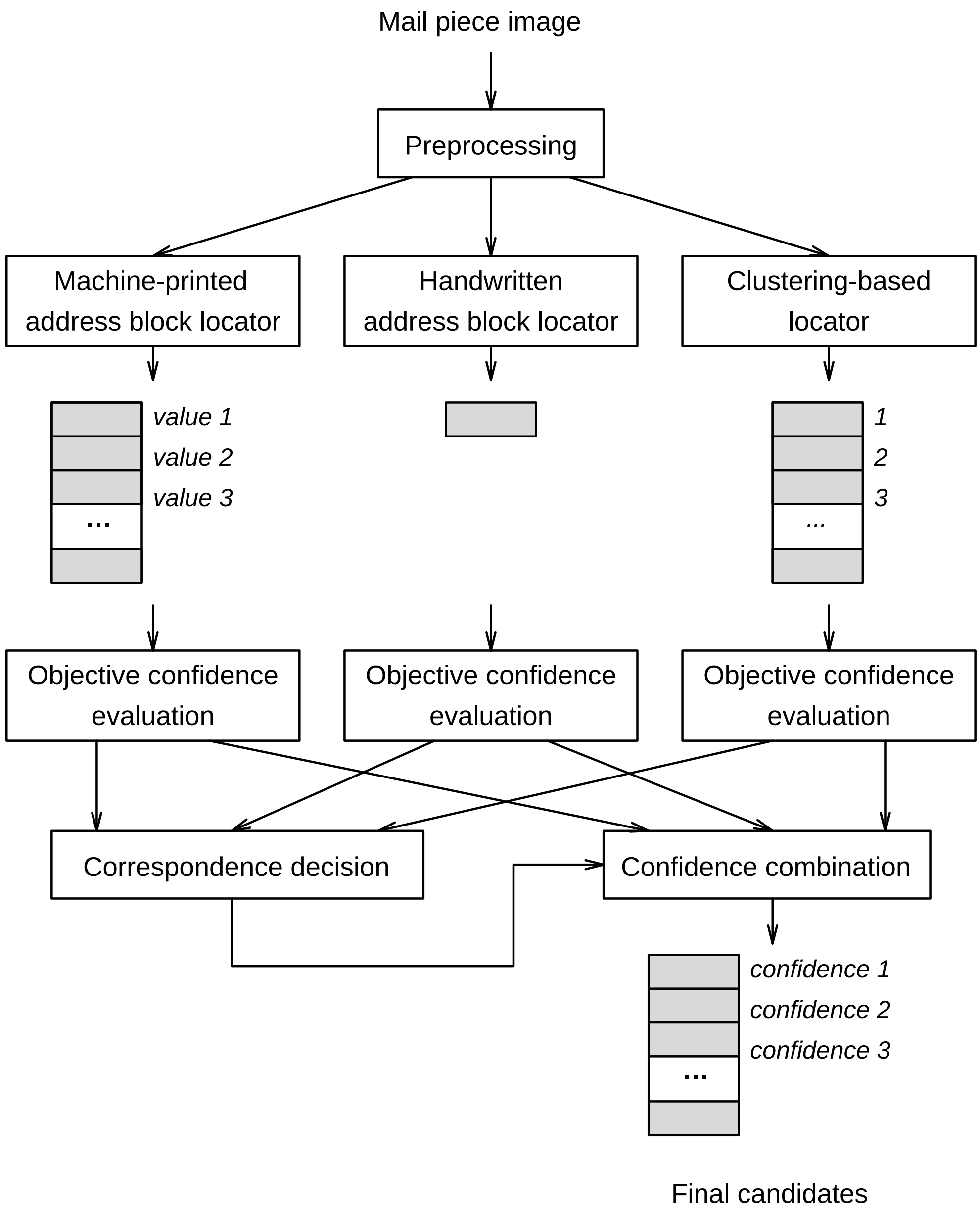


Figure 4.2: Multiple locator algorithm for address block location

proposed in Chapter 3.

4.2.2 Comparison with Previous Approaches

In Section 2.2, we presented a review of various approaches to address block location proposed so far. Most of those approaches employ relatively complex image processing modules and a set of heuristic rules to locate the destination address. Our approach differs from them in that it uses a collection of simplistic locators and attempts to make a judicious use of their results rather than rely on a particular address block locator.

The locators used by the algorithm are designed based on assumptions that are not always true when dealing with an uncontrolled stream of mail piece images. This aspect of the locators will be discussed more in Sections 4.3–4.5. Generally speaking, we hope that the assumptions of at least one of the locators are satisfied and the locator performs well in as many cases as possible. That will enable us to exploit the results of the successful locator by applying the proposed theory of document object locator combination. In other words, the multiple locator algorithm puts more emphasis on combining results of simple locators effectively than on developing a powerful locator.

The most common computational model for address block location consists of two stages: segmentation and classification. When we adopt this model, we are faced with difficulties inherent in each stage. The multiple locator algorithm copes with them by having multiple results in each stage:

1. Segmentation stage: It is hard to assess the correctness of a segmentation of a mail piece image reliably, no matter what segmentation algorithm is used. A direct but computationally expensive way is to recognize all words in a block and determine if the recognition results satisfy an address syntax and semantics. The multiple locator algorithm is less likely to be affected by errors in segmentation than a single locator, because it has multiple segmentation results produced by

multiple locators.

2. Classification stage: Incorrect segmentation results in misleading block feature measurements, which leads to erroneous classification of candidate blocks. Even if segmentation is done correctly, sources of ambiguity abound while classifying the segmented blocks. The algorithm combines decisions on similar blocks by different locators to help resolve this ambiguity.

4.3 Machine-Printed Address Block Locator

In this section, we describe an address block locator designed specifically for machine-printed addresses. The locator assumes that the input mail piece has a machine-printed destination address.

There are several differences between machine-printed and handwritten addresses. Some characteristics of machine-printed addresses relevant to their location are described relative to those of handwritten addresses as follows:

1. Character sizes are small.

2. Variation in character sizes is small.

3. Slopes of text lines with respect to the horizontal are small.

4. Spaces between words in a text line are regular.

5. The structure of a mail piece image with a machine-printed destination address is generally more complex than that of a mail piece image with a hand-written destination address.

Because a machine-printed address has high regularity in its constituents and their composition, we adopt a segmentation approach that exploits this regularity. However, there can be several address-like text blocks resulting from the segmentation.

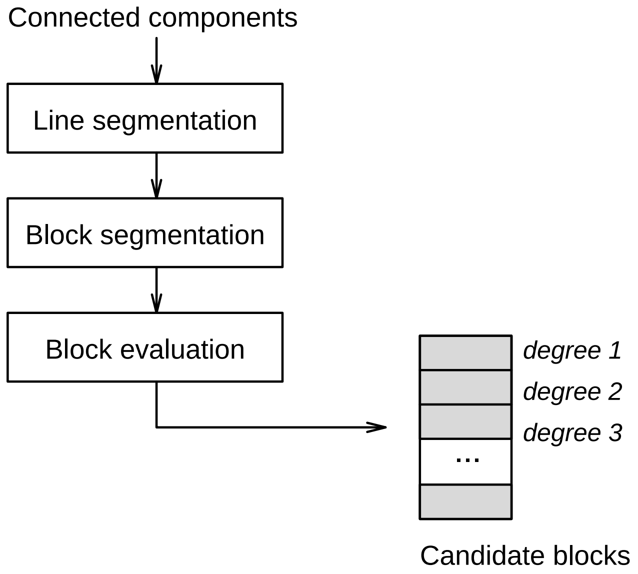


Figure 4.3: Machine-printed address block locator

Therefore, we need to evaluate candidate blocks using block features relevant to addresses. With these considerations, an address block locator that performs bottom-up segmentation of a mail piece and outputs candidate address blocks at the measurement level has been developed.

Figure 4.3 shows the algorithmic flow of the locator. The locator uses connected components of a mail piece image as input data. The connected components are grouped horizontally into lines and then grouped vertically into blocks. This two-level block segmentation method is inspired by the characteristics of machine printed addresses, especially the observation that characters are aligned horizontally to form text lines, which are aligned vertically to form text blocks. Consequently, if these assumptions are violated, the segmentation result will be unsatisfactory.

For each segmented block, block features such as the number of lines, number of connected components, location, and aspect ratio are measured and the degree of the block being the destination address is calculated. The candidate blocks are output with the degree values. Thus the machine-printed address block locator is a measurement-level locator.

Figure 4.4: Input connected components for the machine-printed address block locator

4.3.1 Line Segmentation

Preprocessing

We first examine connected components to remove the ones that are not likely to be a character or word. The width, height, aspect ratio, and area of each component are checked to see if the features are in the ranges that are typical of machine-printed characters or words. If not, it is not considered in the line segmentation process. Figure 4.4 shows an example of connected components input to the locator after this filtering.

The remaining connected components are sorted horizontally using the leftmost coordinate of each component as the key. This allows for one-pass, left-to-right merging of connected components to lines, starting with the leftmost one.

Component-to-Line Merging

The core of line segmentation is component-to-line merging. At any time during this stage, multiple partial lines exist. We examine each new connected component that is not yet part of a line to decide if it can be merged to an existing line. If there are multiple lines to which it can be merged, it is merged to the line that is horizontally closest to it. The selected line is expanded to include the new connected component. If the connected component cannot be merged to any line, it constitutes a new line

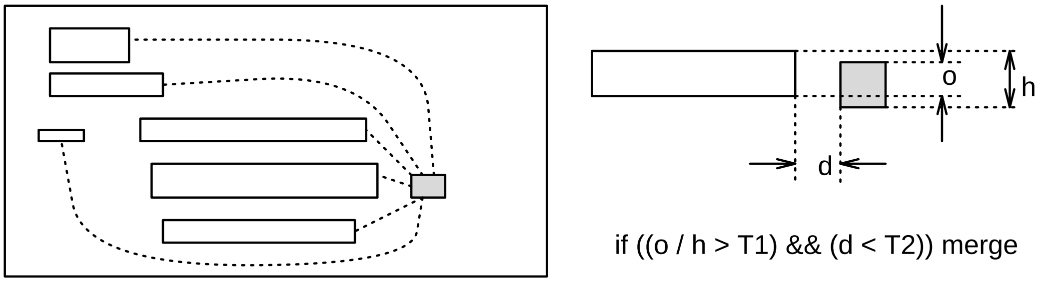


Figure 4.5: Merging a connected component to a line: the connected component is matched with all partial lines to see if it can be merged to one or more of them; the vertical overlap and horizontal distance are used as the merging criteria.

by itself.

The most critical step of this stage as well as of the whole line segmentation algorithm is deciding if a component can be merged to a line. For a connected component, we examine all existing partial lines to find ones that meet the following conditions:

1. The component and the line are aligned horizontally. We check this condition using the degree of vertical overlap between the component and the line (o/h in Figure 4.5).

2. The horizontal distance between the component and the line is short. We check this condition using the distance between the rightmost position of the line and the leftmost position of the component (d in Figure 4.5).

Figure 4.5 illustrates the process of merging a component to a line and how to check the two conditions above. The result of component-to-line merging on the connected components of Figure 4.4 is shown in Figure 4.6

Postprocessing

When component-to-line merging is completed, several lines are generated. These lines not necessarily correspond to the actual text lines, so that we need to postprocess them. Some of the lines may overlap significantly, which is an indication that a single

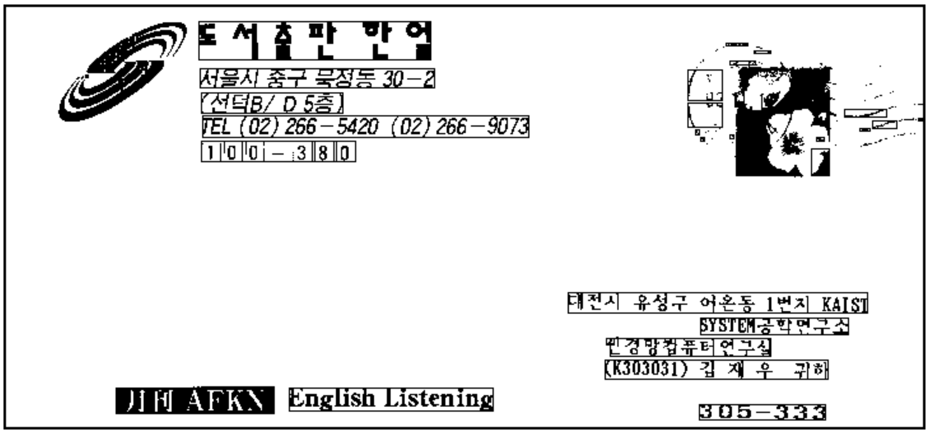


Figure 4.6: Initial line segmentation result

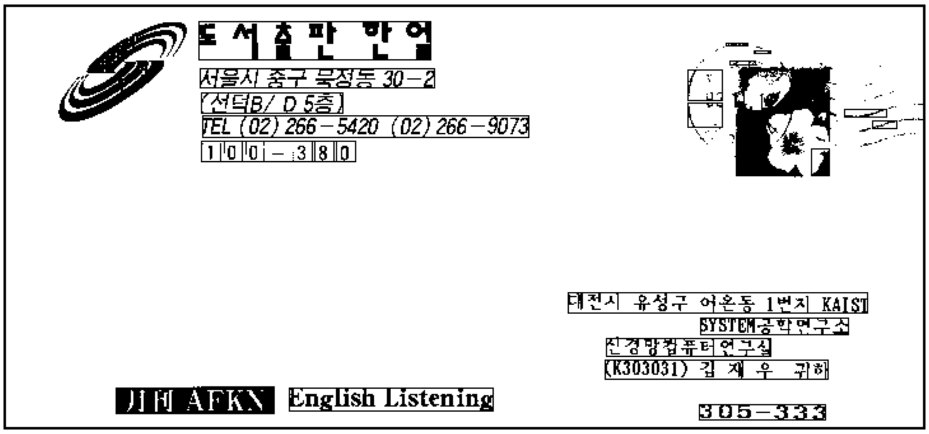


Figure 4.7: Postprocessed line segmentation result

line is split vertically into more than one line. This sometimes happens because the merging is done in the horizontal direction only. Those lines are merged into one line.

In addition, some lines can be very short and isolated. These lines can be part of a meter mark or graphics. They can also be non-address text that usually marks the beginning of the return address or the destination address. Since such lines are not needed and in fact are undesirable during block segmentation, they are removed. Figure 4.7 shows the result of line segmentation after postprocessing the lines in Figure 4.6.

4.3.2 Block Segmentation

Preprocessing

Block segmentation depends on the fact that lines constituting a block are vertically close and aligned. Just as connected components are sorted horizontally for line segmentation, the lines produced by line segmentation are sorted vertically based on their top positions. This vertical sorting is necessary for top-to-bottom merging of lines into blocks in one pass, beginning with the topmost line.

Line-to-Block Merging

Line-to-block merging is based on the same idea as that used in component-to-line merging, but the direction of merging is different. In this stage, we maintain multiple partial blocks at any time. Each new line that does not yet belong to a block is analyzed to decide if merging it to any existing block is feasible. If it can be merged to more than one block, it is merged to the line vertically nearest to it. The selected block is expanded to include the new line. If the line cannot be merged to any block, it becomes a new block by itself.

As in line segmentation, the most important step of line-to-block merging is to decide if a line can be merged to a block. For a new line, we look for a block with the following properties:

1. The line and the block are vertically aligned. We check this condition using the amount of horizontal overlap between the line and the block (o in Figure 4.8).

2. The vertical distance between the line and the block is short. We check this condition using the distance between the bottom position of the block and the top position of the line (d in Figure 4.8).

3. The height of the line (h in Figure 4.8) and the mean line height of the block ($mean\_h$ in Figure 4.8) are similar. We check this condition using the ratio of the smaller of the two heights to the bigger.

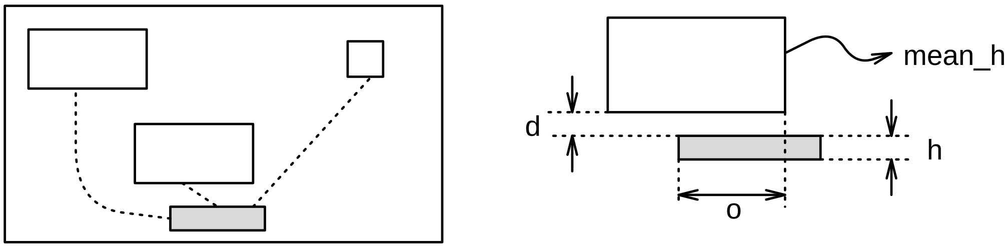


Figure 4.8: Merging a line to a block: the line is matched with all partial blocks to see if it can be merged to one or more of them; the horizontal overlap, vertical distance, and ratio of line heights are used as the merging criteria.

Figure 4.8 shows the process of merging a line to a block and the method of analyzing a pair of a line and a block. Using the notation of Figure 4.8, we decide that the line can be merged to the block if one of the following is true:

1. $(o \geq 0)$ and $(d < T_1)$ and $(\min(h/mean\_h, mean\_h/h) > T_2)$
2. $(T_5 < o < 0)$ and $(d < T_3)$ and $(\min(h/mean\_h, mean\_h/h) > T_4)$

When the horizontal overlap is smaller (the second item), the vertical distance should be smaller and the ratio of line heights should be bigger for the line to be merged to the block. Thus, $T_3 < T_1$ and $T_4 > T_2$. Figure 4.10 shows the block segmentation result obtained by merging the lines in Figure 4.7.

4.3.3 Block Evaluation

We evaluate the candidate address blocks generated by block segmentation as to their likelihoods of being the destination address. The results of the evaluation are represented by the measurement values assigned to the blocks.

We choose several block features to use for the evaluation. To compute the measurement value for a block, we compute degrees to which the block belongs to the destination address class using the features. A degree value is calculated from a single feature and the degrees are combined to calculate an overall degree. A degree in this context can be understood as a *degree of membership*, as used in fuzzy set theory, in the destination address class. The overall degrees are output with the candidate blocks.

Let us represent the features of a block by $x_1, x_2, \ldots, x_n$. We convert the feature values into degrees as follows:

$$d_i = f_i(x_i) \text{ for } 1 \leq i \leq n,\ 0 \leq d_i \leq 1 \tag{4.1}$$

where f_i is the function to convert the i-th feature (x_i) into the i-th degree (d_i). The d_i values are then combined by performing fuzzy intersection, which is usually done by taking the minimum of the degrees:

$$d = \min_{1 \leq i \leq n} (d_i). \tag{4.2}$$

Choosing the minimum of all the degrees as the combined degree has the effect of providing the safest decision for a given block. We cannot guarantee that this is the best way to combine the degrees. As a matter of fact, there is no known optimal way to do so. We will be content that the minimum operation is simple and results in reasonable measurement values for the purpose of this locator, because designing a powerful and complicated locator is not our concern.

The selection of block features is based on the characteristics of the machine-printed addresses. We use the following block features for the evaluation:

1. The number of lines in the block.

2. The number of connected components in the block.

3. The area of the block.

4. The aspect ratio of the block.

5. The location of the block.

Table 4.1: Degrees of membership for the blocks in Figure 4.10

Block id	Degree computed from feature				
	# of lines	# of components	Area	Aspect ratio	Location
1	1.000	1.000	1.000	1.000	0.738
2	1.000	1.000	1.000	1.000	0.336
3	0.667	0.120	0.266	0.304	0.168
4	0.333	0.020	0.068	0.341	0.353
5	0.333	0.360	0.633	0.000	0.800
6	1.000	0.100	0.134	1.000	0.000
7	0.667	0.040	0.139	1.000	0.000

The conversion of the features to appropriate degrees is illustrated in Figure 4.9. Knowledge of features of typical machine-printed destination addresses plays an important role in designing the conversion functions.

Figure 4.10 shows the final result of block segmentation and block evaluation results given the connected components of Figure 4.4. The degrees of membership in the destination address class computed from the features of the blocks in the image are listed in Table 4.1.

4.4 Handwritten Address Block Locator

We describe in this section an address block locator designed specifically for handwritten addresses. The locator assumes that the input mail piece has a handwritten destination address.

Handwritten addresses have the following characteristics, compared to machine-printed ones:

1. Character sizes are big.

2. Variation in character sizes is large.

3. Slopes of text lines with respect to the horizontal can sometimes be big.

4. Spaces between words in a text line are irregular.

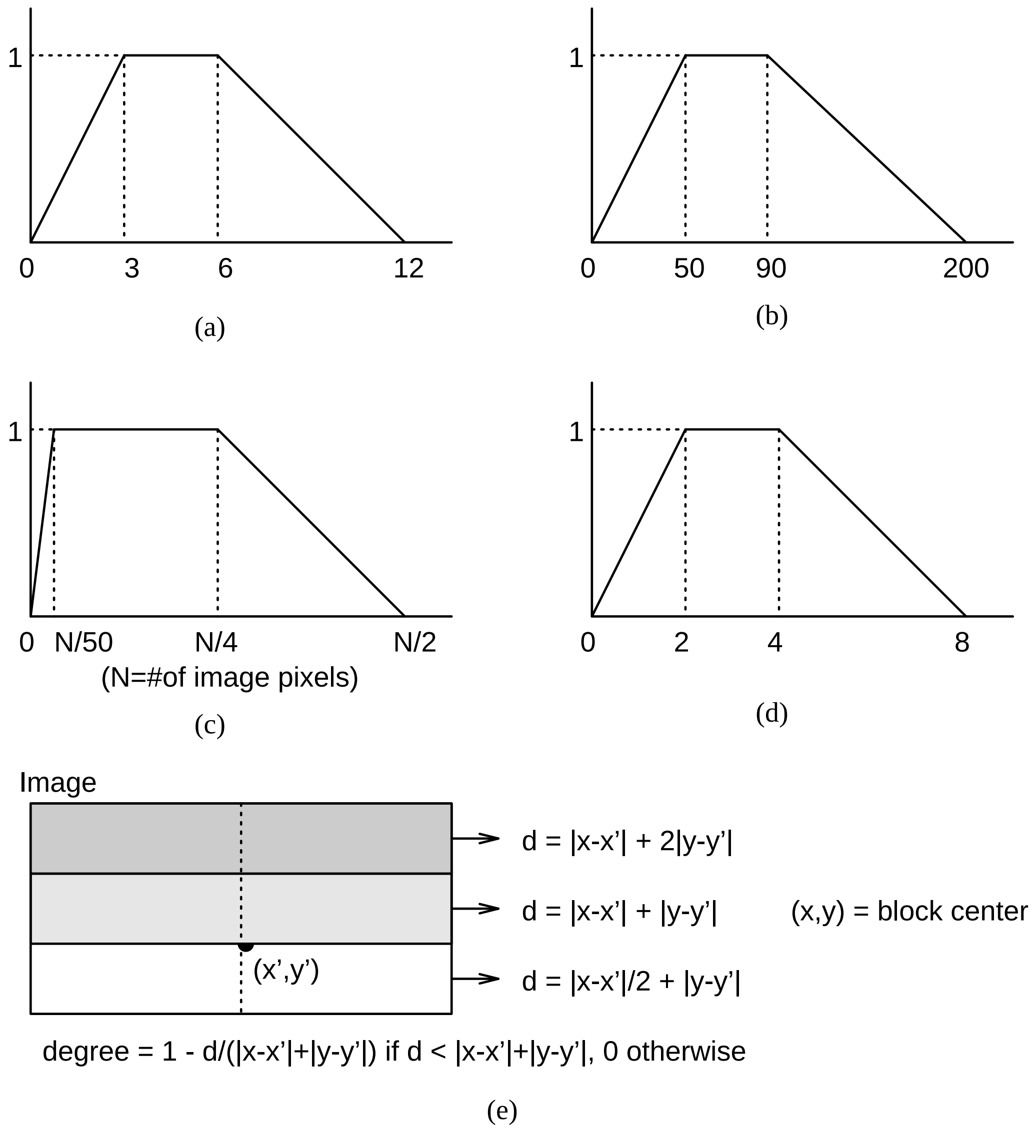


Figure 4.9: Functions to compute the degree of membership in the destination address class using a single block feature: (a) number of lines; (b) number of connected components; (c) area (width $\times$ height); (d) aspect ratio (width / height); (e) location; each feature is converted to a value between 0 and 1.

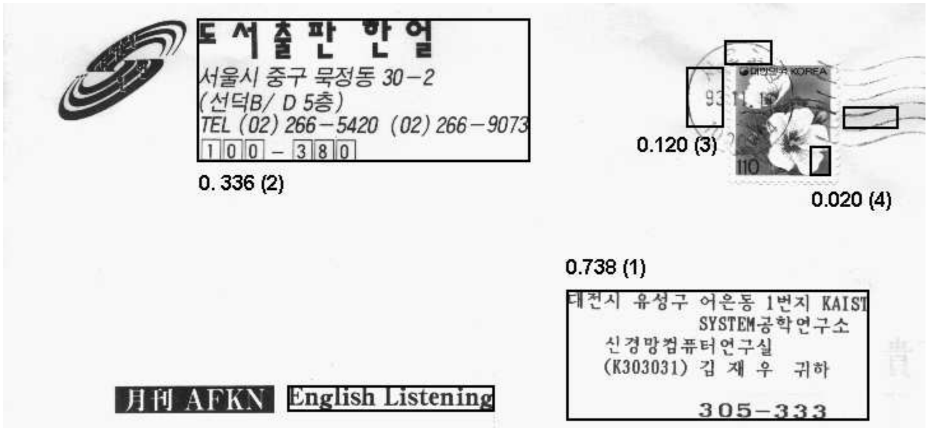


Figure 4.10: Output of the machine-printed address block locator: seven blocks are output, where only four of them have positive measurement values shown.

5. The structure of a mail piece image with a handwritten destination address is generally simpler than that of a mail piece image with a machine-printed destination address.

Mail piece images with a handwritten destination address typically have a simple structure: only the return address and the destination address. However, the address blocks themselves have high irregularity. Thus correct segmentation of the destination address is more important to a handwritten address block locator than correct classification of segmented blocks.

The direction of a handwritten text line sometimes deviates from the horizontal significantly. This means that a bottom-up segmentation approach using the horizontal alignment of connected components and vertical alignment of text lines, such as the one used for the machine-printed address block locator, would perform poorly for mail piece images with such text lines. Therefore, we adopt a top-down segmentation approach.

Figure 4.11 shows the structure of the locator. The locator partitions the whole image into several *preliminary* text lines. If this partitioning fails, the input image is rejected. Otherwise, the preliminary lines are processed further to detect the actual text lines. The collection of text lines is split into two groups, corresponding to the

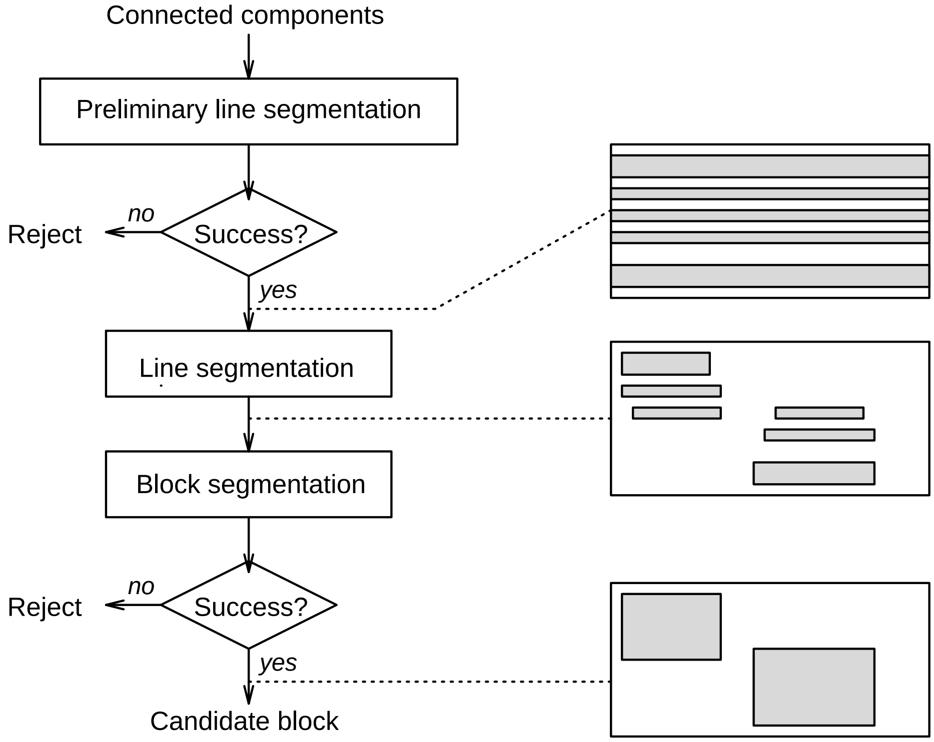


Figure 4.11: Handwritten address block locator

destination address and the return address, respectively. The text lines corresponding to the destination address are merged and output as the unique candidate block. Thus this locator assumes that only the return and destination addresses exist on the mail piece, which is commonly true of handwritten mail pieces. Since the locator outputs a unique candidate block without any other information, it is an abstract-level locator.

4.4.1 Preliminary Line Segmentation

A preliminary line is a text line whose location is defined only vertically. Thus it spans the whole width of a mail piece image and bounded at the top and bottom by two vertical coordinates. The goal of preliminary line segmentation is to partition the image into multiple preliminary lines, where a preliminary line contains one or more real text lines. We analyze a horizontal projection profile to estimate the positions of the preliminary lines.

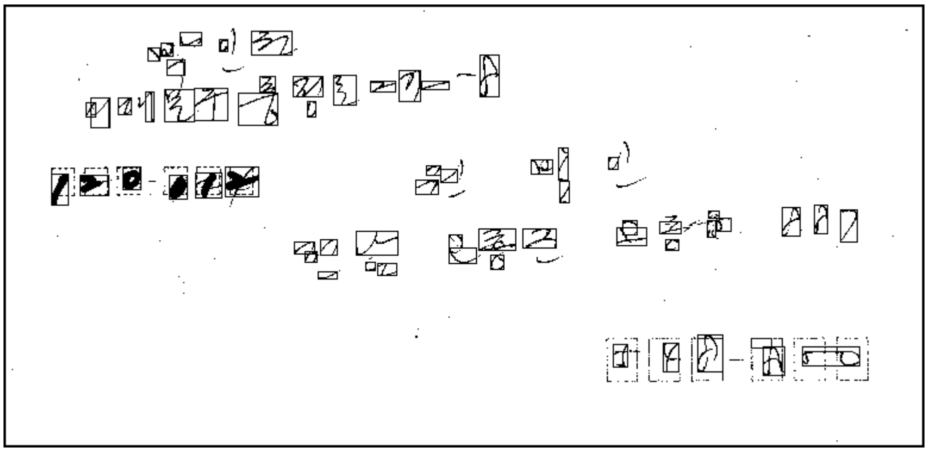


Figure 4.12: Input connected components for the handwritten address block locator

Preprocessing

Since we intend to use a horizontal projection profile as a basis for finding preliminary lines, we need to remove those connected components which are not likely to correspond to text. These components stem from input noise, line segments in image border areas resulting from binarization, boxes marking the stamp position, and other extraneous markings. For each connected component represented by its minimum bounding rectangle, we exclude it from further processing if one of the following conditions holds,

1. The width is too small or too big.
2. The height is too small or too big.
3. The area is too big.
4. The position is too close to the upper-left corner of the image.

An example of performing this filtering on raw connected components is shown in Figure 4.12.

Because a handwritten mail piece usually has sparse text, projecting the binary image as is will result in low peaks, so that finding preliminary lines is difficult. Therefore, we fill the minimum bounding rectangles enclosing the connected components

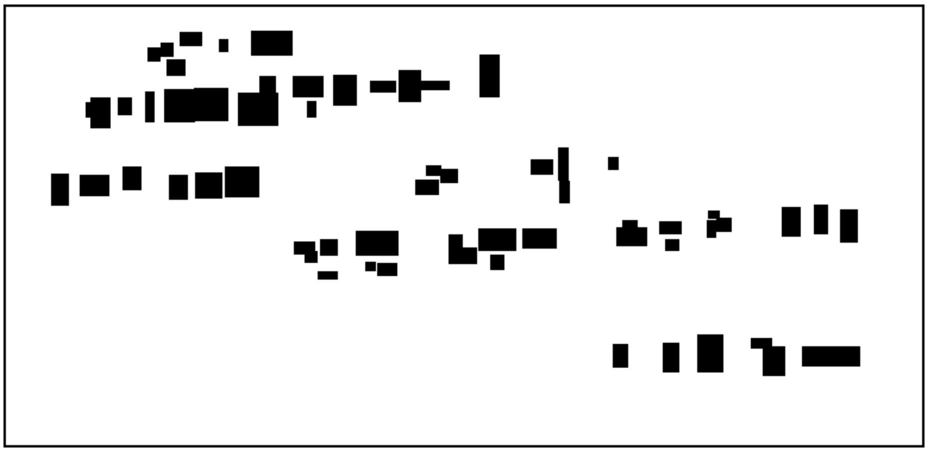


Figure 4.13: Area-filled connected components

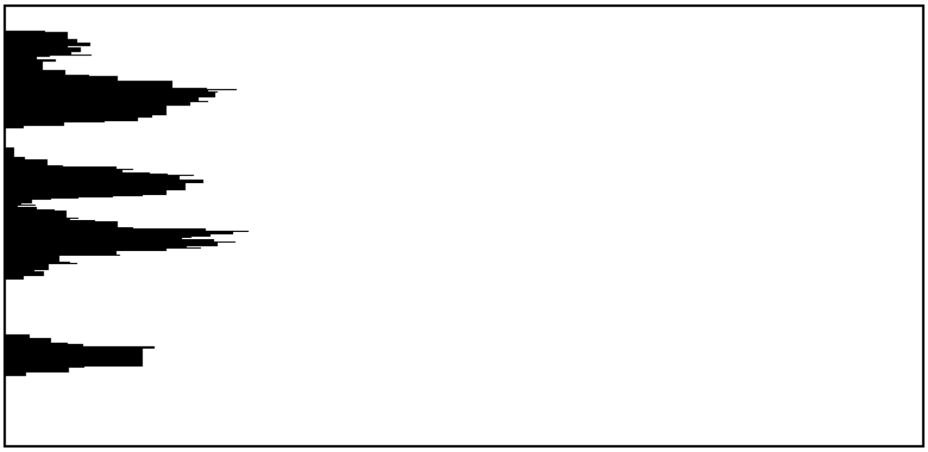


Figure 4.14: Horizontal projection profile

with black pixels and use the resulting image for horizontal projection. Figure 4.13 shows the result of this filling operation on the connected components of Figure 4.12.

Horizontal Projection Profile Generation

We project the area-filled connected components onto the vertical axis to generate a horizontal projection profile. This profile forms the basis of preliminary line segmentation. The horizontal projection profile of components in Figure 4.13 is shown in Figure 4.14.

The raw horizontal projection profile usually contains many sudden peaks, as in

Figure 4.14. Since these peaks interfere with finding the structure of preliminary lines, we try to remove them by smoothing the profile. The smoothing operation replaces each row of the profile with the mean of the five consecutive rows centered on it. That is, calling the profile as $h_i, 1 \leq i \leq n$, where n is the number of rows in the image, we do the operation

$$h_i = \frac{h_{i-2} + h_{i-1} + h_i + h_{i+1} + h_{i+2}}{5}. \tag{4.3}$$

If we increase the number of rows involved in the smoothing, we obtain more smoothing effect. The profile smoothed from the one in Figure 4.14 is shown in Figure 4.16, where we can see that most spurious peaks are removed.

Preliminary Line Hypothesis Generation and Testing

Ideally, a horizontal projection profile consists of prominent peaks and valleys. The peaks and valleys represent text lines and spaces between lines, respectively. Preliminary lines can be located by finding the peaks and valleys. In practice, with handwriting this ideal situation seldom exists; there are many local peaks and valleys which make detecting global peaks and valleys difficult. This observation leads us to investigate other methods of analyzing the profile.

We adopt a method that does not rely on peaks and valleys. We try to cut the profile vertically to find an appropriate *cut point*. A cut point superimposed on a horizontal projection profile defines a set of preliminary lines. For example, Figure 4.16 shows four cut points on a profile, which also define four different sets of preliminary lines.

The preliminary line segmentation algorithm searches for an appropriate cut point using the hypothesis generation and testing paradigm. Every cut point tried generates a hypothesis of preliminary lines. The hypothesis is tested by assessing the feasibility that the preliminary lines represent the actual text line positions. This process repeats until a satisfactory cut point is found. If the process fails to find one, the algorithm rejects the horizontal projection profile, and thus the input image. The algorithm is presented in more detail in Figure 4.15.

Generation of successive cut points is based on a binary search, for fast convergence to a satisfactory cut point. At first, the start and end pointers of the binary search are initialized to 0 and max, respectively (line 1 of Figure 4.15), where max is the maximum value of the profile. The algorithm then generates a new cut point (line 5) and the preliminary line hypothesis (line 6), and tests its goodness by examining the number of lines (line 7), line heights (line 8), and the degree to which the lines cover the connected components (line 10), until a good cut point is found (line 10). If the three features of the line set are in proper ranges, the current cut point is confirmed as the final one, and the resulting preliminary line hypothesis is verified as preliminary line positions.

If any of the three features is out of the proper range, one of the binary search pointers is modified according to the direction in which the feature value is out of the range. That is, if the number of lines is too small, the line heights are too small, or the degree of covering connected components is too small, the cut point is regarded as being too close to max, and the end pointer is set to the current cut point (lines 11, 13, and 15), restricting the search to the left half of the current search region. In the opposite case, the start pointer is set to the current cut point (lines 12, 14, and 16), limiting the search to the right half.

For example, in Figure 4.16, the cut points labeled 1, 2, 3, and 4 are successively tried and the fourth cut point is determined to be an appropriate one. The preliminary lines resulting from the final cut point in this case is shown in Figure 4.17.

4.4.2 Line Segmentation

A preliminary line is not a complete text line yet; its horizontal bounds are not determined. Line segmentation here refers to partitioning a set of preliminary lines

```
 0. input horizontal projection profile;
 1. p = 0; q = max;                  /* initialize binary search */
 2. cut_ok = FALSE;
 3. while (!cut_ok) {
 4.    if (q-p < 2)
           break;                    /* end binary search */
 5.    cut_point = (p+q)/2;
 6.    generate preliminary line hypothesis at current cut_point;
 7.    if (number of lines is OK) {
 8.        if (line heights are OK) {
 9             cover connected component image with lines;
10.            if (cover rate is OK)
                   cut_ok = TRUE;      /* test OK */
11.            else if (cover rate is too small)
                   q = cut_point;
12.            else
                   p = cut_point;
           }
13.        else if (line heights are too small)
               q = cut_point;
14.        else
               p = cut_point;
       }
15.    else if (number of lines is too small)
           q = cut_point;
16.    else
           p = cut_point;
   }
17. if (cut_ok)
       output preliminary line hypothesis;
18. else
       reject the profile;
```

Figure 4.15: Preliminary line hypothesis generation and testing algorithm

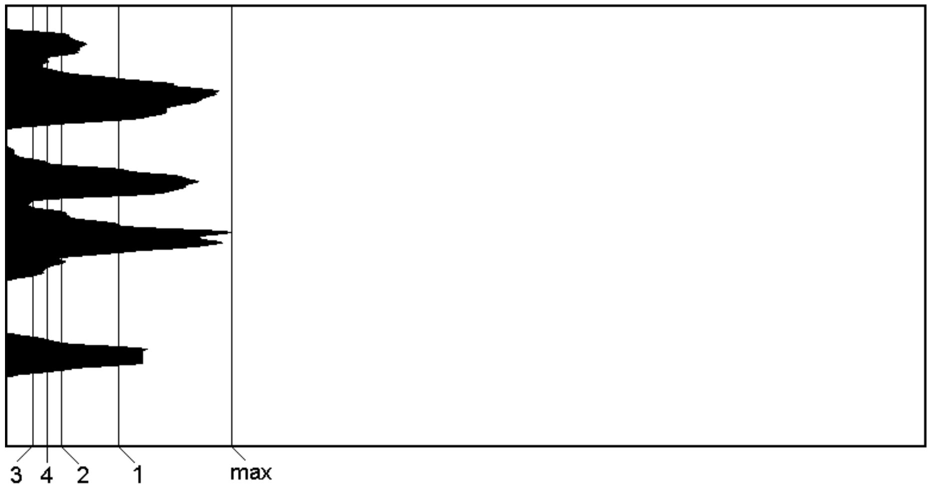


Figure 4.16: Searching for the cut point: the vertical lines indicating the cut points tried are drawn on the smoothed horizontal projection profile; the cut point is found at the fourth iteration.

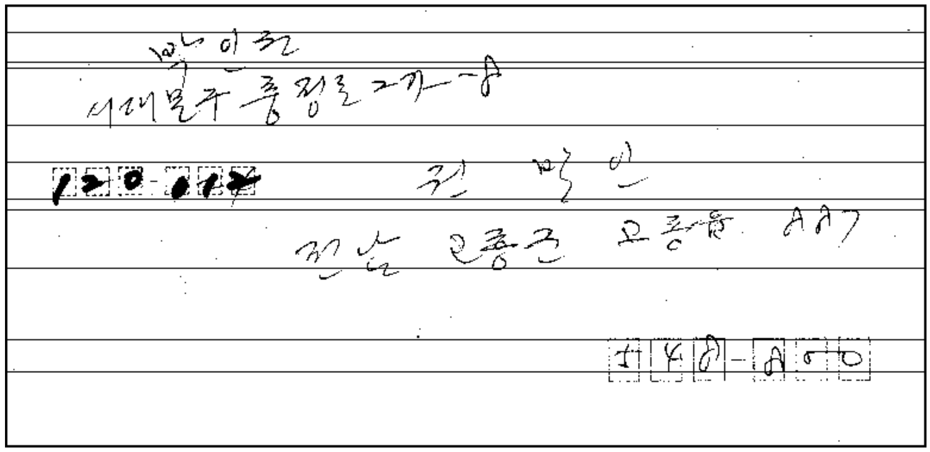


Figure 4.17: Preliminary line segmentation result: a vertical run of black pixels at the cut point 4 of Figure 4.16 defines a preliminary line.

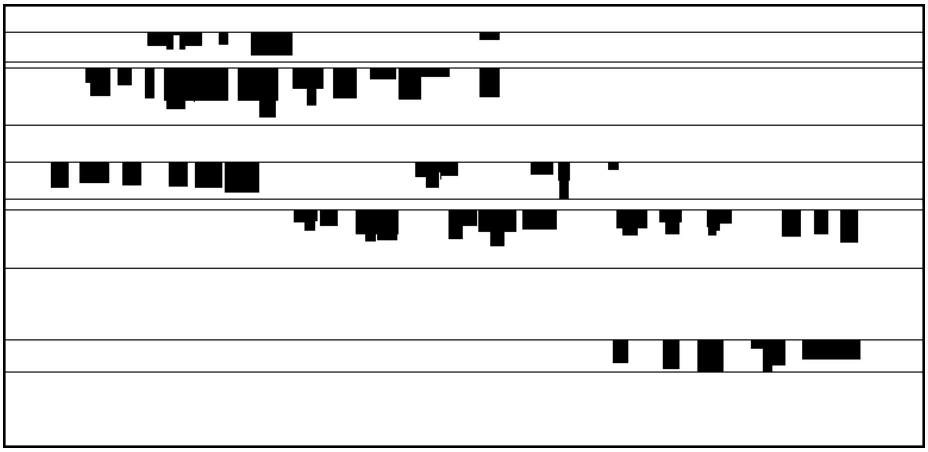


Figure 4.18: Vertical projection profiles of preliminary lines

into a set of text lines.

Vertical Projection Profile Generation

A few text lines whose vertical positions are similar can be included in a single preliminary line. To decide if multiple text lines exist in a preliminary line, we generate a vertical projection profile for each preliminary line. Only the portions of connected components included in a preliminary line are used in the projection. Figure 4.18 is the result of performing vertical projection using the preliminary lines of Figure 4.17 and area-filled connected components of Figure 4.13.

Text Line Generation

To generate text lines, we analyze each preliminary line to see if it can be split into multiple lines. For this analysis, the gaps between peaks of its vertical projection profile provides more important clue than the width or height of the peaks. The basic principle we use is to split a preliminary line into two text lines where the horizontal gap is unusually wide.

The method for detecting a wide gap is based on the assumption that the gaps in a text line has an approximately normal distribution. Calling the horizontal gaps existing in a vertical projection profile as g_i for $1 \leq i \leq n$, where n is the number of

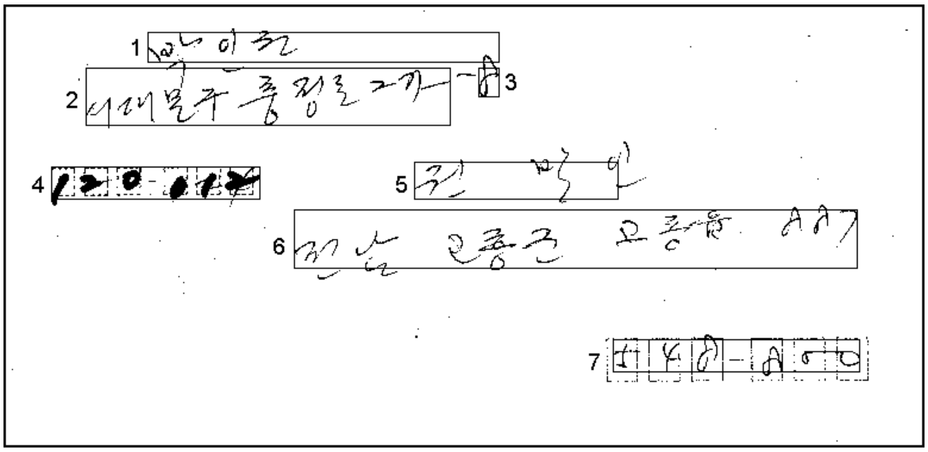


Figure 4.19: Line segmentation result

such gaps, we see if any gap satisfies the condition

$$(g_i > T_g) \text{ or } (g_i > m_g + c \times \sigma_g) \tag{4.4}$$

where T_g is a large gap value, m_g and σ_g are the mean and the standard deviation of gap widths, respectively, and c is a constant. In other words, we look for a gap whose absolute or relative width is very large. If there is no gap that satisfies the condition, the preliminary line becomes a text line by itself by finding the leftmost and the rightmost coordinates from the profile.

For example, Figure 4.19 shows the text lines generated from the profiles in Figure 4.18. We see that two preliminary lines are split into two text lines each, while the others are not.

4.4.3 Block Segmentation

For block segmentation, we use the following knowledge of the structure of a typical handwritten mail piece:

1. Only the destination address and the return address exist.

2. The destination address is to the lower-right of the return address.

Assuming these two conditions hold, block segmentation can be done by splitting the collection of text lines into two groups: the lower-right one corresponding to the destination address and the upper-left one corresponding to the return address.

Block Hypothesis Generation and Testing

Block segmentation also uses a hypothesis generation and testing approach. According to the number of text lines, we generate multiple hypotheses each of which amounts to a splitting of the collection of text lines. The text lines output by the line segmentation algorithm are numbered from top to bottom. Multiple lines produced from one preliminary line are numbered from left to right. The way the lines are numbered in Figure 4.19 is an example.

Given this line numbering scheme, we define a splitting simply by a separation point in the list of line numbers. This is possible since we are trying to partition lines into an upper-left group and a lower-right group. For each feasible number of lines, Table 4.2 lists the hypotheses generated. If the number of lines is n, there exist $n-1$ possible hypotheses. However, not all possible hypotheses are generated; only highly feasible ones are generated. If there are too many lines, the line segmentation itself is highly likely to be incorrect, and the block segmentation algorithm rejects the mail piece.

Consider a block hypothesis which includes the first line of the destination address as the last line of the return address block or the last line of the return address as the first line of the destination address block. This erroneous grouping of text lines generally results in a high aspect ratio and a lot of white space for one of the blocks, as Figure 4.20 indicates.

Block hypothesis verification is based on the above observation. For each hypothesis, we compute a score that is high if the aspect ratios are close to two and the amount of white space in the blocks is small. We choose the hypothesis with the highest score and take the upper-left block as the return address and the lower-right

Table 4.2: Block hypothesis generation: a hypothesis is represented by a set of two blocks; each block is represented by the list of line numbers it includes.

# of lines	Hypotheses {the upper-left block, the lower-right block}
2	{(1), (2)}
3	{(1, 2), (3)} {(1), (2, 3)}
4	{(1), (2, 3, 4)} {(1, 2), (3, 4)} {(1, 2, 3), (4)}
5	{(1, 2), (3, 4, 5)} {(1, 2, 3), (4, 5)}
6	{(1, 2), (3, 4, 5, 6)} {(1, 2, 3), (4, 5, 6)} {(1, 2, 3, 4), (5, 6)}
7	{(1, 2, 3), (4, 5, 6, 7)} {(1, 2, 3, 4), (5, 6, 7)}
8	{(1, 2, 3), (4, 5, 6, 7, 8)} {(1, 2, 3, 4), (5, 6, 7, 8)} {(1, 2, 3, 4, 5), (6, 7, 8)}
9	{(1, 2, 3, 4), (5, 6, 7, 8, 9)} {(1, 2, 3, 4, 5), (6, 7, 8, 9)}
10	{(1, 2, 3, 4), (5, 6, 7, 8, 9, 10)} {(1, 2, 3, 4, 5), (6, 7, 8, 9, 10)} {(1, 2, 3, 4, 5, 6), (7, 8, 9, 10)}
11	{(1, 2, 3, 4, 5), (6, 7, 8, 9, 10, 11)} {(1, 2, 3, 4, 5, 6), (7, 8, 9, 10, 11)}
12	{(1, 2, 3, 4, 5), (6, 7, 8, 9, 10, 11, 12)} {(1, 2, 3, 4, 5, 6), (7, 8, 9, 10, 11, 12)} {(1, 2, 3, 4, 5, 6, 7), (8, 9, 10, 11, 12)}

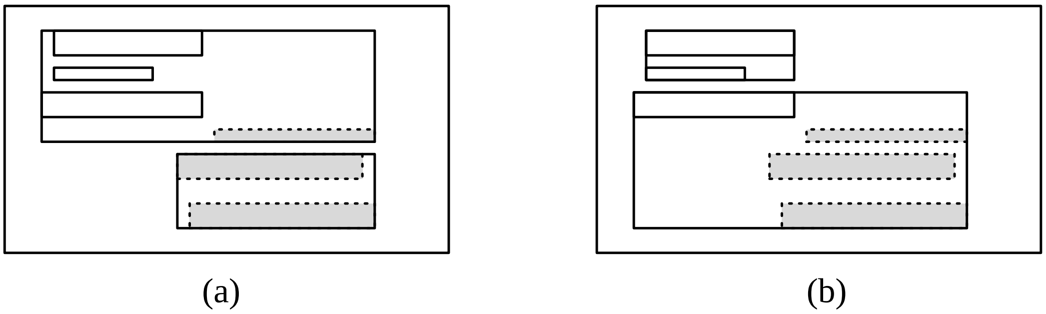


Figure 4.20: Typical incorrect block hypotheses: one of the blocks, the upper-left one in (a) and the lower-right one in (b), has a high aspect ratio and a lot of white space (gray: text lines of the true destination address).

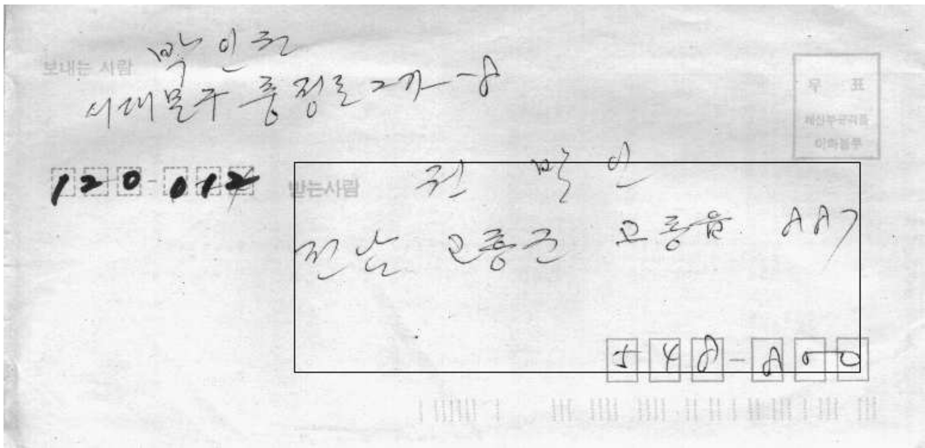


Figure 4.21: Output of the handwritten address block locator: the lower-right block from the block segmentation algorithm is output as a unique candidate.

one as the destination addresses. For example, the final output of the locator from the lines of Figure 4.19 is a single block as shown in Figure 4.21.

4.5 Clustering-Based Locator

Both the machine-printed address block locator and the handwritten address block locator described above are based on the structure formed by horizontal text lines, although they are based on different ideas. Accordingly, if the direction of text lines deviates severely from the horizontal or there are irregular markings that are not typical of horizontal text lines, neither locator would work well. A locator that does not rely on any text line structures would be able to cope with this difficulty.

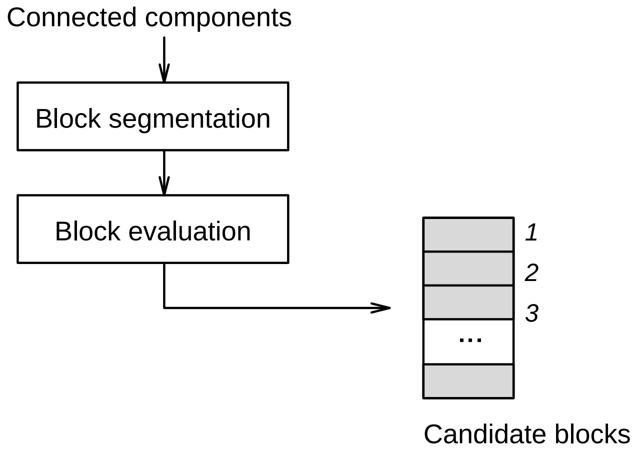


Figure 4.22: Clustering-based locator

We describe a locator based on direct clustering of connected components into blocks, without line segmentation. The locator assumes nothing on the address printing method or line structures. The clustering is based only on distances among connected components and does not use any other features. The assumption this locator relies on is, generally speaking, that distances among components of the same block is shorter than the distances between different blocks. If there is an unusually large space between characters or lines in a text block, the locator would perform poorly. Similarly, two or more blocks that are too close to one another will usually cause an unsatisfactory result.

Figure 4.22 shows the flow of the locator. Connected components input to the locator are treated as objects and they are clustered into blocks. The blocks are then evaluated using their locations and an ordered list of candidate blocks are output. Thus the locator is a rank-level locator.

4.5.1 Block Segmentation

According to Zupan [102, page 4], "Clusters are groups of objects linked together according to some rules. The goal of clustering is to find groups containing objects most homogeneous within these groups, while at the same time the groups are heterogeneous between themselves as much as possible."

The block segmentation algorithm adopts a standard clustering procedure called *agglomerative clustering*. It means that the clustering is done from the single objects to the final cluster by merging smaller clusters. The output of agglomerative clustering is a hierarchical tree (or dendrogram) representing the hierarchy of objects. Block segmentation is done by cutting the tree (or stopping generating the tree) at an appropriate point based on the desired number of clusters or certain other conditions.

Depending on how to measure the distance between two clusters, many different clustering methods are possible [102]. We discuss two simple examples. In the *single linkage method* or the nearest neighbor technique [44], the distance between two clusters I and J is the shortest distance between any two points, one in each cluster:

$$D_{IJ} = \min_{i \in I, j \in J} (d_{ij}) \tag{4.5}$$

where i and j are elements of clusters, and d_{ij} is the distance between them. Cluster sizes do not affect the distances in this method.

On the contrary, the *complete linkage method* or the furthest neighbor technique [44] defines the distance between two clusters as the longest distance between any two points, one in each cluster:

$$D_{IJ} = \max_{i \in I, j \in J} (d_{ij}) \tag{4.6}$$

In this method, if a cluster size is big, the distance between it and another cluster tends to be long. This is undesirable for block segmentation. Therefore, we use the single linkage method for block segmentation.

Preprocessing

First we remove those connected components which are not likely to represent a character or word, as in other locators. Remaining connected components are input to the locator. An example of input connected components to the locator is shown

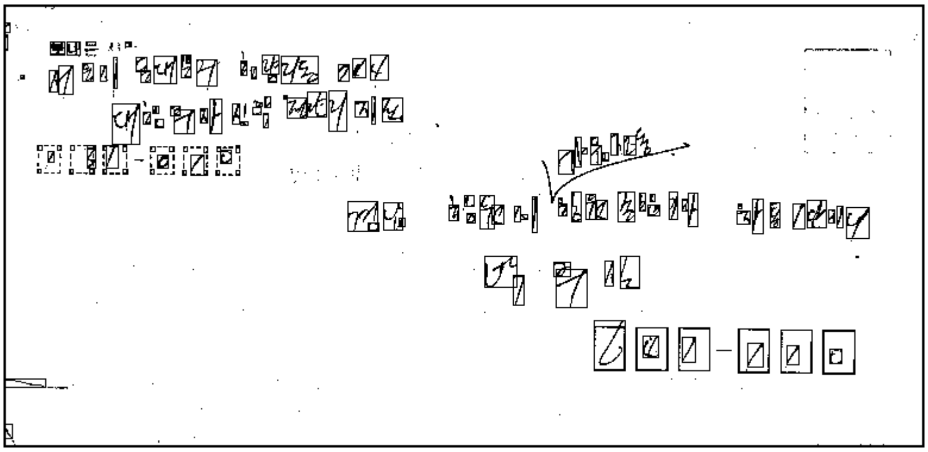


Figure 4.23: Input connected components for the clustering-based locator

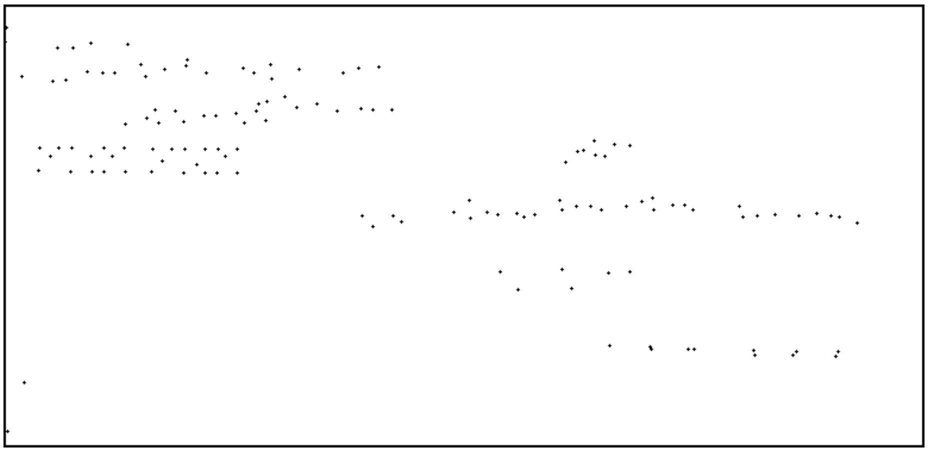


Figure 4.24: Connected components represented as points

in Figure 4.23.

In clustering of large data sets, an object is usually represented as a point in space. We use this representation, because it is simple and convenient. To treat a connected component as an object and represent it as a point, we use the center point of the minimum bounding rectangle of the connected component. Figure 4.24 is the point representation of the connected components in Figure 4.23. Using this representation implies that we will ignore all information but the connected component positions during block segmentation.

Distance Matrix Calculation

To calculate the distance between two clusters, we need to know the distance between any pair of points, one from each cluster. It is convenient to calculate the distance between any pair of input points beforehand, because they are used frequently. The calculation result is stored in the form of a distance matrix

$$D = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix} \tag{4.7}$$

where n is the number of points and d_{ij} is the Euclidean distance between two points i and j ($d_{ij} = d_{ji}$).

Cluster Initialization

In this stage, we initialize clusters. Each point constitutes its own cluster and thus the number of initial clusters is the same as the number of points.

Agglomerative Clustering

Block segmentation is accomplished by the clustering procedure that repeats the following steps:

1. Find the two closest clusters among all possible pairs of clusters. The distance between two clusters is evaluated by the single linkage method of (4.5). That is, find two clusters P and Q such that

   $$D_{PQ} = \min_{1 \leq I,J \leq C,\ I \neq J} (D_{IJ}) \tag{4.8}$$

   where C is the number of current clusters.

2. Merge the two clusters and set $C = C - 1$.

3. If C is in the appropriate range, evaluate the clusters:

   (a) If any of them is determined to be an incorrectly segmented address block, split the cluster formed last back to its constituent clusters and output the previous clusters as blocks.

   (b) Else go back to step 1.

   Else go back to step 1.

Thus block segmentation is done by repeated clustering of two closest clusters at each iteration, until further clustering results in unsatisfactory segmentation.

At each iteration, only one new cluster is formed and the number of clusters decreases by one. The most critical step in the clustering procedure is step 3(a), where we decide whether the last cluster merging produced an undesirable block. This step is a kind of backtracking where we go back to the previous result if the new result is determined to be worse.

The test of clusters is based on what happens when the clustering procedure proceeds beyond the optimal point. Steps 1 and 2 above finds two closest clusters and merges them, no matter how long the distance between them is. Consequently, a very wide or high cluster with sparse text content can be generated at the first iteration after the optimal point. If such a cluster is detected, we decide that the last merging of clusters was unnecessary and output the clusters at the previous iteration.

For testing a set of clusters to determine if a large sparse cluster exists, we measure three features for each cluster. For the i-th cluster, let the ratio of the cluster width to the image width, the ratio of the cluster height to the image height, and the ratio of the sum of connected component areas to the cluster area be w_i, h_i, and a_i, respectively. If the following condition holds, we decide that the clustering is past the

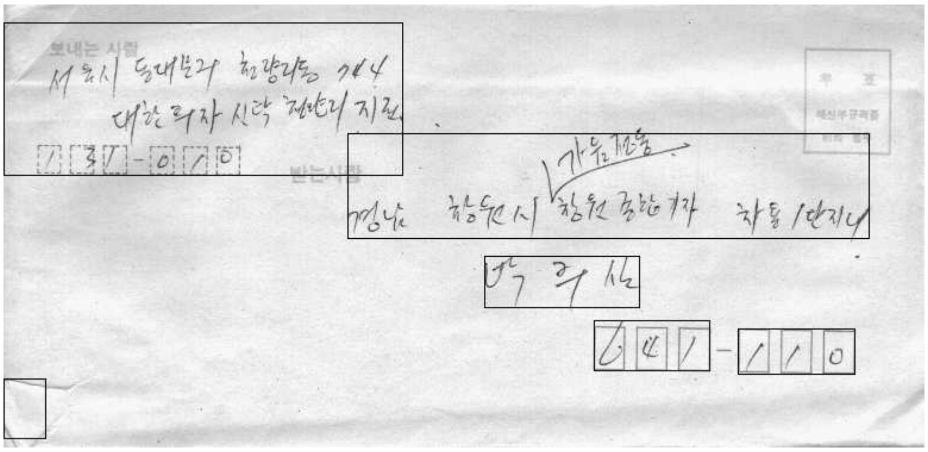


Figure 4.25: Six clusters during block segmentation

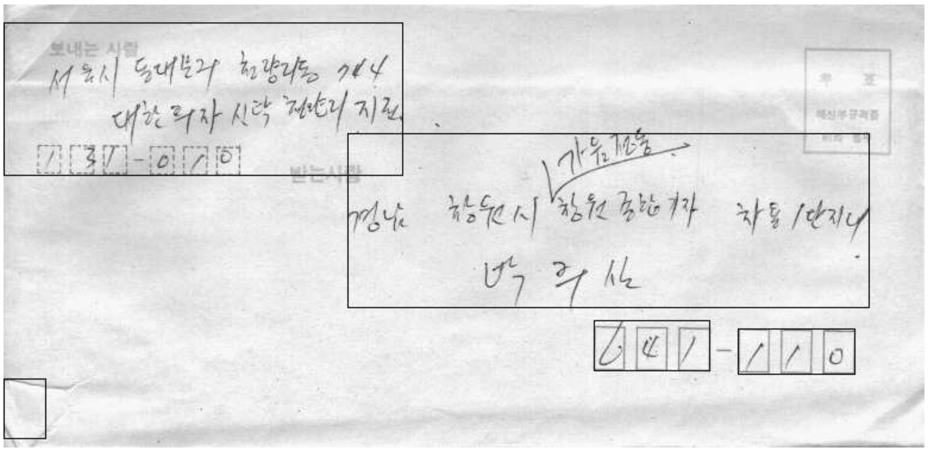


Figure 4.26: Five clusters during block segmentation

optimal point:

$$
\begin{aligned}
&(\max_{1 \le i \le C}(w_i) > T_1) \text{ or } (\max_{1 \le i \le C}(h_i) > T_2) \text{ or} \\
&((\max_{1 \le i \le C}(h_i) > T_3) \text{ and } (\arg\max_{1 \le i \le C}(h_i) = \arg\min_{1 \le i \le C}(a_i))) \qquad (4.9)
\end{aligned}
$$

where C is the number of clusters, T_1, T_2, and T_3 are predetermined threshold values. The test determines if a very wide block, a very high block, or a somewhat high block with the lowest density of connected components exists. Accordingly, $T_3 < T_2$.

Figures 4.25 to 4.29 show the clustering results using the points of Figure 4.24 when the number of clusters is within the desired range, From the figures, we see

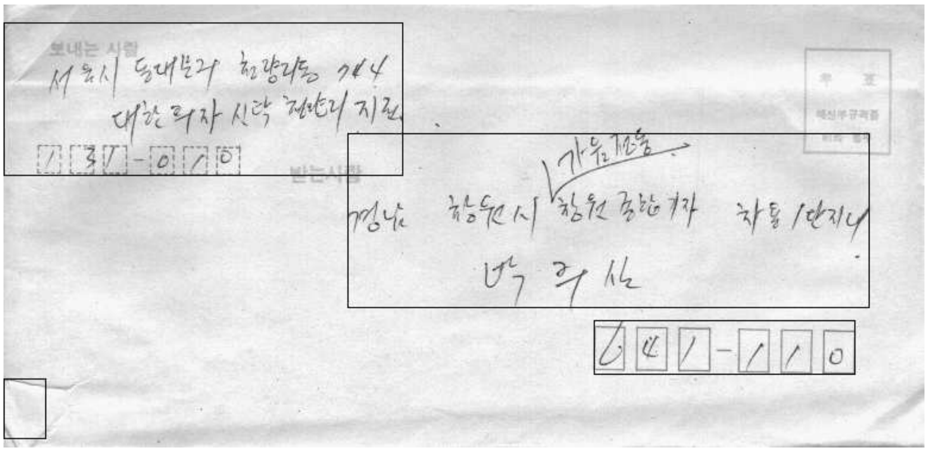


Figure 4.27: Four clusters during block segmentation

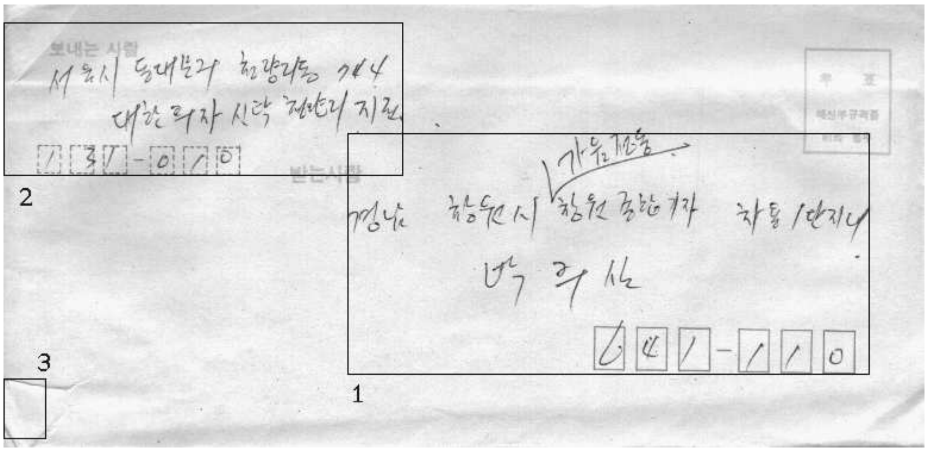


Figure 4.28: Three clusters during block segmentation and also the output of the clustering-based locator: three blocks are output with an ordering.

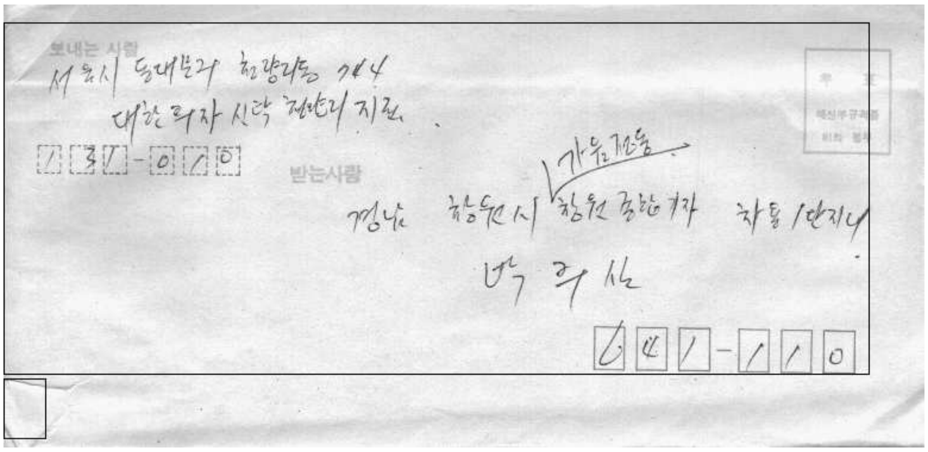


Figure 4.29: Two clusters during block segmentation: a block containing both the destination and return addresses is formed.

that the segmentation result is best when the number of clusters is three. When the number of clusters becomes two, a huge block is formed, which is detected by the test. Therefore, the three-cluster state becomes the block segmentation result.

Since the clustering procedure uses only the distances between connected components, represented by the distance matrix, we can state the condition under which the destination address will be correctly formed as a cluster *at some point* in the clustering procedure:

$$\max_{i,j}(d_{ij}) < \min_{p,q}(d_{pq}) \tag{4.10}$$

where i and j are both connected components of the destination address, and only one of p and q is a connected component of the destination address.

Stopping the clustering at the point the destination address is correctly clustered depends on the terminating condition test. If the test fails to detect the point just after the best point, the cluster of the destination address can be merged with other text and block segmentation can fail.

4.5.2 Block Evaluation

The clustering-based locator does not assume a particular printing method. Accordingly, for evaluating a set of blocks from the locator, using features that vary according to the printing method is not suitable. We consider only the block locations to generate a ranking of the blocks.

Irrespective of whether the destination address is machine-printed or handwritten, its location tends to be at the bottom of the mail piece, towards the right. Therefore, we use a measure indicating how close a block is to the expected location. We determine a reference point which is at two-thirds of the image width from the left end and two-thirds of the image height from the top end. For each block, its distance to the reference point is measured. The blocks are ranked in descending order of the distances.

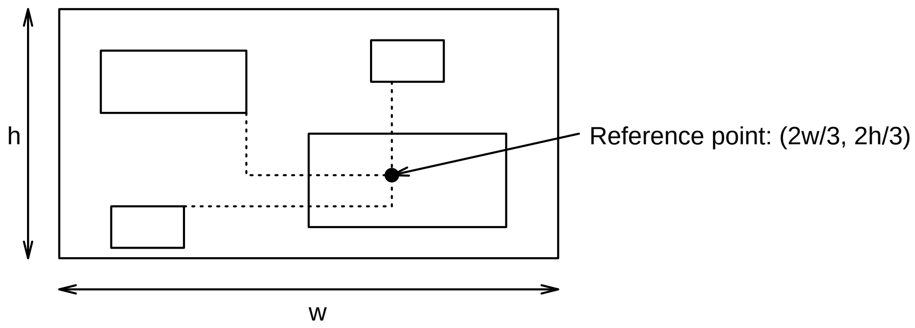


Figure 4.30: Measuring the city-block distance to the reference point: the dotted lines indicate the distances; the block including the reference point has the distance 0.

We use the city-block distance as the distance measure. The distance between a block B and the reference point r is calculated as

$$d(B,r) = d(b,r) = \begin{cases} \mid b_x - r_x \mid + \mid b_y - r_y \mid & \text{if } r \text{ is outside } B \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

where $r = (r_x, r_y)$ is the reference point and $b = (b_x, b_y)$ is the point on the border of the block B that is closest to r. Measuring the distances is illustrated in Figure 4.30.

Using the distances of the blocks to the reference point, we order the blocks. An example output of the locator is shown in Figure 4.28, where three ranked blocks are output.

Chapter 5

Address Block Location Experiments

This chapter describes the experiments in address block location using the algorithm we presented in Chapter 4. In Section 5.1, we describe details of mail piece image preprocessing. Section 5.2 describes the mail piece image database used in the experiments. Section 5.3 presents the results of training in terms of several parameters and functions required for testing. Finally, the test results from the experiments and analysis is given in Section 5.4.

5.1 Mail Piece Image Preprocessing

Preprocessing consists of two stages: mail piece image binarization and connected component detection. These two algorithms are common to all locators. Since all locators use connected components of a mail piece image as their input, we binarize the input gray-level image and detect connected components from the binary image.

Image Binarization

In our implementation, a mail piece images is scanned as a gray-level image. The purpose of binarizing this image is to separate the pixels belonging to text from the background.

A number of methods for document image binarization have been proposed so far. Evaluation of some of the methods is given in [45, 59, 71]. We use Otsu's thresholding method [68], which is one of the most commonly used document image binarization methods. Given a gray-level image, the method finds the global threshold value that maximizes 'class separability' using the gray-level histogram of the image. The method is faster than local thresholding methods, and preliminary experiments on our mail piece image database showed that it produced satisfactory thresholding results for most images, for the purpose of address block location for most images. Details of the thresholding method is presented in Section A.3. An example of mail piece image binarization using the method is shown in Figure 5.1.

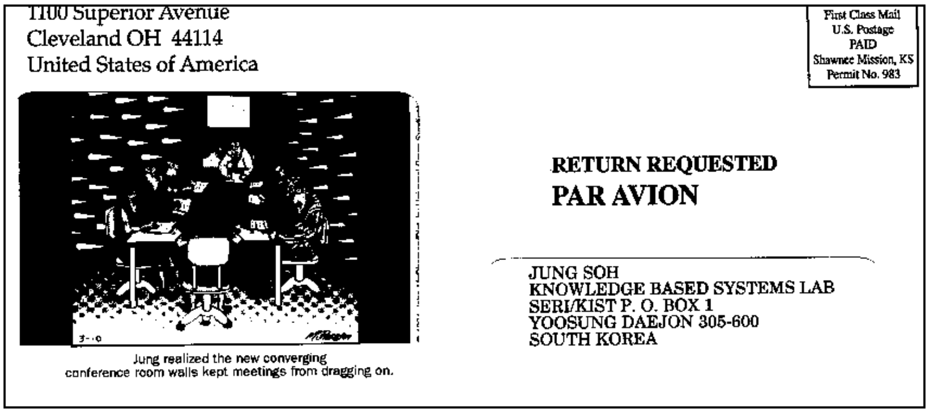
Connected Component Detection

Connected components of a mail piece image usually correspond to character strokes, characters, or words. After the binarization, connected components in the binary image are detected. There also exist many methods for connected component detection. Some of them are described in detail in [29, 82]. We use a two-pass algorithm that scans each row of the image, labels black pixels, and uses an equivalence table to resolve conflicts in labels. Figure 5.2 shows the result of detecting connected components on the image of Figure 5.1(b).

5.2 Mail Piece Image Database

Most mail pieces used in our experiments were provided by Seoul Mail Center in Seoul, Korea, where all mail pieces coming into and going out of Seoul are processed

(a)



(b)

Figure 5.1: Mail piece image binarization: (a) gray-level image; (b) binarized image.

Figure 5.2: Connected component detection result

Table 5.1: Composition of the mail piece image database

Image set	Seoul Mail Center	Hand Korean	Machine Korean	Hand English	Machine English	Total
Set 1	450	28	44	3	25	550
Set 2	450	28	44	3	25	550
Total	900	56	88	6	50	1100

Table 5.2: Image sets in two experiments

Experiment	Training set	Test Set
1	Set 1	Set 2
2	Set 2	Set 1

by automatic sorting facilities. The envelopes were gathered in an effort to tune a set of OCR machines in the Center to the characteristics of Korean postal codes. Most of the envelopes have a destination address handwritten by a postal employee, and do not have a stamp pasted.

We gathered another group of envelopes locally. This collection includes several kinds of envelopes: ones with a destination address that is handwritten in Korean, machine-printed in Korean, handwritten in English, and machine-printed in English. It adds variety to our image database.

All the envelopes that we collected were scanned as gray-level images at 100 pixels per inch, using a Microtek ScanMaker 600ZS scanner. Table 5.1 describes the composition of the mail piece image database used in the experiments. We have a total of 1,100 images in the database, divided randomly into two equal-sized sets, except that each one has the same number of images of each image type.

We conduct two separate experiments, Experiment 1 and Experiment 2, as shown in Table 5.2: one using Set 1 for training and Set 2 for testing, the other using Set 2 for training and Set 1 for testing. This approach will facilitate a more reliable test of our combination theory than doing only one experiment.

Table 5.3: Objective confidence values determined by training

Experiment	Locator	Objective confidence
1	ML (1)	$e_1(u) = \dfrac{\exp(-6.9256 + 11.4137u)}{1 + \exp(-6.9256 + 11.4137u)}$
	HL (2)	$e_2 = 0.7167$ (334/466, # of rejects= 84)
	CL (3)	$e_3(1) = 0.5182$ (285/550), $e_3(2) = 0.0109$ (6/550) $e_3(3) = 0.0021$ (1/483), $e_3(4) = 0.0034$ (1/290) $e_3(r) = 0$ for $r > 4$
2	ML (1)	$e_1(u) = \dfrac{\exp(-6.5515 + 11.1512u)}{1 + \exp(-6.5515 + 11.1512u)}$
	HL (2)	$e_2 = 0.7320$ (325/444, # of rejects= 106)
	CL (3)	$e_3(1) = 0.5200$ (286/550), $e_3(2) = 0.0164$ (9/550) $e_3(r) = 0$ for $r > 2$

5.3 Training Results

The multiple locator algorithm requires training to acquire knowledge of the behavior of the locators. The purpose of training is to extract this knowledge in the form of a number of parameters, which are used for testing the algorithm.

For brevity, we will call the machine-printed address block locator, the handwritten address block locator, and the clustering-based locator described in Chapter 4 as ML, HL, and CL, respectively. We will also use the locator numbers 1, 2, and 3, for ML, HL, and CL, respectively, if necessary. All training results that we present below are on the basis of the 550 images in a training set.

5.3.1 Objective Confidence

We described in Section 3.3 three different methods of obtaining objective confidence values for locators at different output information levels. The results of applying these methods to the locators ML, HL, and CL, which are at the measurement level, abstract level, and rank level, respectively, are shown in Table 5.3.

Logistic regression as described in Section 3.3.1 is used for deriving the function to compute objective confidence values for ML, a measurement-level locator. Table

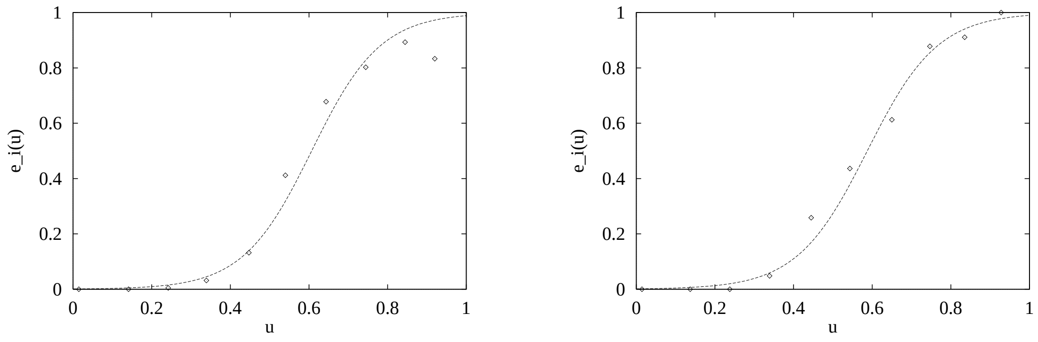
Table 5.4: Training data for computing objective confidence for ML by logistic regression

Experiment	$X = u$	N_u	$N_{u \wedge true}$	$\pi = \frac{N_{u \wedge true}}{N_u}$	$\pi' = \ln\left(\frac{\pi}{1-\pi}\right)$
1	0.0147	1851	0	0.0003	−8.2164
	0.1410	406	0	0.0012	−6.6983
	0.2420	214	1	0.0047	−5.3613
	0.3389	126	4	0.0317	−3.4177
	0.4472	53	7	0.1321	−1.8827
	0.5400	51	21	0.4118	−0.3567
	0.6434	90	61	0.6778	0.7436
	0.7443	81	65	0.8025	1.4018
	0.8444	56	50	0.8929	2.1203
	0.9200	6	5	0.8333	1.6094
2	0.0145	1924	0	0.0003	−8.2550
	0.1371	405	0	0.0012	−6.6958
	0.2378	183	0	0.0027	−5.8999
	0.3389	124	6	0.0484	−2.9789
	0.4448	58	15	0.2586	−1.0531
	0.5432	55	24	0.4364	−0.2559
	0.6500	80	49	0.6125	0.4578
	0.7468	82	72	0.8780	1.9741
	0.8351	56	51	0.9107	2.3224
	0.9279	11	11	0.9545	3.0445

5.4 shows the training data used for estimating the regression coefficients and Figure 5.3 shows the fitting of the regression functions to the data. Note that the π values are modified when $\pi = 0$ or 1 as explained in Section A.1. The objective confidence value for HL is computed by the ratio (3.24). Similarly, the ratio (3.21) generates the objective confidence values for CL.

5.3.2 Correspondence Thresholds

The next set of parameters to determine from training is the correspondence threshold values for all possible pairs of locators. A correspondence threshold value for a pair of locators is used to decide whether two spatially overlapping objects, one from each locator in the pair, correspond to each other. A detailed description of the method



(a) (b)

Figure 5.3: Plot of training data and objective confidence functions for ML: (a) Experiment 1; (b) Experiment 2.

Table 5.5: Correspondence threshold values determined by training

Experiment	Locator pair	Number of corresponding object pairs	Number of non-corresponding object pairs	Threshold	Decision error rate on training data
1	ML-HL	122	282	0.8295	2.23% (9/404)
	ML-CL	282	370	0.7958	6.29% (41/652)
	HL-CL	192	517	0.7727	4.65% (33/709)
2	ML-HL	125	253	0.7929	4.76% (18/378)
	ML-CL	267	335	0.7018	7.64% (46/602)
	HL-CL	180	478	0.7156	7.75% (51/658)

for finding such values was given in Section 3.4.3. For a locator pair, the method basically finds the threshold value that minimizes the number of wrong correspondence decisions on the set of training data for the two locators. Table 3.4 presents an example of the data necessary for determining a correspondence threshold, which is for the locator pair ML-HL in Experiment 1.

In each experiment, we need to determine three correspondence threshold values, one for each pair of ML-HL, ML-CL, and HL-CL. The results of determining the correspondence threshold values from training are given in Table 5.5. The determined threshold values result in the lowest possible correspondence decision error rates on the training data.

5.3.3 Dynamic Locator Relevance

Dynamic locator relevance values are used as fuzzy measure densities in confidence value combination using fuzzy integrals, described in Section 3.6.5. They are also used in experimenting with dynamic selection of locators. We use the result-based approach of (3.37) to compute dynamic locator relevance, where the agreement among the top choices from the locators is the criterion for image set partitioning.

The fuzzy integral method also requires fuzzy densities for sets of locators. These are calculated from the dynamic relevance values of the individual locators in the locator set using the properties of the g_λ-fuzzy measure described in Section A.2. Since we have three locators, $2 + {}_3C_2 = 5$ agreement classes exist. For each agreement class, we first measure the dynamic relevance values of the locators. These are the top-choice performance figures of the locators for the training images in the agreement class. Dynamic relevance values for the sets of locators are calculated using those values. Table 5.6 shows the dynamic locator relevance values obtained from training, for all agreement classes.

Dynamic selection is done by deciding the agreement class of the locator outputs and using the dynamic locator relevance values for that class to order the top choices of the locators. In many cases, there are multiple locators with the highest dynamic relevance. For example, locators ML and CL have the same highest dynamic relevance to the ML-CL agreement class, in both experiments. In these cases, we use the objective confidence of top-choice objects to break the tie and generate an ordering.

5.3.4 Functions for Probabilistic Confidence Estimation

For the probabilistic confidence estimation method of confidence value combination, described in Section 3.6.4, we need to estimate several coefficients for the regression functions of the form (3.45). Table 3.6 gives an example of data needed for estimating

Table 5.6: Dynamic relevance values determined by training

Experiment	Locator / Locator set	Agreement class (agreeing locators)				
		None	ML,HL	ML,CL	HL,CL	All
1	ML	0.1231 (32/260)	0.9500 (19/20)	0.7407 (60/81)	0.0227 (2/88)	0.9901 (100/101)
	HL	0.6010 (122/203)	0.9500 (19/20)	0.2037 (11/54)	0.9318 (82/88)	0.9901 (100/101)
	CL	0.1615 (42/260)	0.0500 (1/20)	0.7407 (60/81)	0.9318 (82/88)	0.9901 (100/101)
	{ML,HL}	0.7669	0.9999	0.8062	0.9335	0.9999
	{ML,CL}	0.2961	0.9526	0.9789	0.9335	0.9999
	{HL,CL}	0.8186	0.9526	0.8062	0.9999	0.9999
	{ML,HL,CL}	1.0000	1.0000	1.0000	1.0000	1.0000
2	ML	0.1389 (35/252)	0.9655 (28/29)	0.7609 (70/92)	0.0119 (1/84)	1.0000 (93/93)
	HL	0.5847 (107/183)	0.9655 (28/29)	0.3091 (17/55)	0.9524 (80/84)	1.0000 (93/93)
	CL	0.1667 (42/252)	0.0345 (1/29)	0.7609 (70/92)	0.9524 (80/84)	1.0000 (93/93)
	{ML,HL}	0.7662	1.0000	0.8480	0.9530	1.0000
	{ML,CL}	0.3178	0.9667	0.9754	0.9530	1.0000
	{HL,CL}	0.8026	0.9667	0.8480	1.0000	1.0000
	{ML,HL,CL}	1.0000	1.0000	1.0000	1.0000	1.0000

Table 5.7: Functions for probabilistic confidence estimation determined by training

Experiment	Locator tuple	Function
1	(ML,HL)	$\frac{e}{1+e}$, $e = \exp(0.5261 + 3.0060v_m - 2.0000v_h)$
	(ML,CL)	$\frac{e}{1+e}$, $e = \exp(-4.1384 + 5.9586v_m + 3.1817v_c)$
	(HL,CL)	$\begin{cases} 0.9318 & \text{if } v_c = 0.5182 \\ \text{undefined} & \text{otherwise} \end{cases}$
	(ML,HL,CL)	$\frac{e}{1+e}$, $e = \exp(0.8074 + 2.1467v_m - 2.1467v_h + 4.4091v_c)$
2	(ML,HL)	$\frac{e}{1+e}$, $e = \exp(0.5216 + 3.0630v_m - 1.0630v_h)$
	(ML,CL)	$\frac{e}{1+e}$, $e = \exp(-3.2739 + 4.3689v_m + 2.8081v_c)$
	(HL,CL)	$\begin{cases} 0.9524 & \text{if } v_c = 0.5200 \\ \text{undefined} & \text{otherwise} \end{cases}$
	(ML,HL,CL)	$\frac{e}{1+e}$, $e = \exp(0.8289 + 2.0897v_m - 1.4587v_h + 6.7380v_c)$
v_m, v_h, and v_c are the objective confidence values for the objects from ML, HL, and CL, respectively.		

the regression coefficients, which is for the locator tuple (ML,CL) in Experiment 1.

Since we use three locators, three pairs and one triple of locators exist. For each such tuple of locators, we determine a different regression function which will be used to estimate the combined confidence when objects from the locators constituting the tuple correspond to one object.

The estimated functions from the training data are shown in Table 5.7. For example, suppose that we have a pair of corresponding objects, one from ML and the other from CL, in Experiment 1. Then we use the function

$$\frac{\exp(-4.1384 + 5.9586v_m + 3.1817v_c)}{1 + \exp(-4.1384 + 5.9586v_m + 3.1817v_c)}$$

to calculate the combined confidence for both objects.

For the locator pair (HL,CL) in Experiment 1, there are only two possible pieces of observed data of v_h and v_c: (0.5182, 0.7167) and (0.0109, 0.7167) (refer to Table 5.3). These are too few data to find a regression function. Therefore, we use the direct estimation method of (3.43). For the first case, the correct rate is 82/88=0.9318. Since the second case occurs only once, we leave the combined value undefined and use the objective confidence values of the objects as the combined confidence values. The same situation happens for the locator pair (HL, CL) in Experiment 2.

5.4 Test Results and Analysis

The test of the proposed multiple locator algorithm for address block location is done in two separate experiments, each on the 550 images excluding the training images. All the five combination methods proposed in Section 3.6 are tested. For comparison, dynamic selection of locators based on agreement classes is also tested.

Test results in terms of location performance are summarized in Table 5.8. The table shows the location performance figures of individual locators, the five combination methods, and dynamic selection. The combination is successful for all five combination methods and dynamic selection, because the location performance of every combination method and dynamic selection is better than any individual locator performance. The fourth row of each experiment shows the *theoretically highest* location performance we expect from a locator combination, which happens to be 85.1% in both experiments. We see that this level of performance is obtained by considering up to four top choices at most, except in dynamic selection which uses only the top choices of locators.

There are considerable improvements in location performance due to combination as we can see by vertically comparing the location performance figures in Table 5.8. This is mainly because the individual locator performance are relatively poor, yet the locators are highly complementary.

It is important to distinguish the *combination performance* from the location performance improvement due to combination. This point was discussed in Section 3.7, where the two measures are contrasted. The combination performance figures as

Table 5.8: Test results in terms of location performance

Experiment	Locator / Combination method	% correct (of all test images) in top choice	top 2 choices	top 3 choices	top 4 choices
	ML	41.3	41.5		
	HL	59.1			
	CL	52.0	53.6		
	At least one of ML, HL, and CL	84.9	85.1		
	Highest confidence	76.5	82.7	85.1	
	Confidence summation	76.0	82.2	85.1	
1	Weighted confidence summation	76.0	82.4	85.1	
	Probabilistic confidence estimation	76.7	83.1	85.1	
	Fuzzy integral	72.2	80.5	84.9	85.1
	Dynamic selection	72.2	78.7	84.9	
	ML	38.7	38.9		
	HL	60.7			
	CL	51.8	52.9	53.1	53.3
	At least one of ML, HL, and CL	84.9	85.1		
	Highest confidence	74.9	82.7	84.9	85.1
	Confidence summation	76.7	82.7	84.9	85.1
2	Weighted confidence summation	76.7	83.1	84.9	85.1
	Probabilistic confidence estimation	75.1	83.1	84.9	85.1
	Fuzzy integral	72.2	80.5	84.5	85.1
	Dynamic selection	71.8	79.6	84.9	

Table 5.9: Test results in terms of combination performance

Experiment	Combination method	% correct (of k images) in			
		top choice	top 2 choices	top 3 choices	top 4 choices
1	Highest confidence	90.0	97.2	100.0	
	Confidence summation	89.3	96.6	100.0	
	Weighted confidence summation	89.3	96.8	100.0	
	Probabilistic confidence estimation	90.2	97.6	100.0	
	Fuzzy integral	84.8	94.7	99.8	100.0
	Dynamic selection	84.8	92.5	99.8	
2	Highest confidence	88.0	97.2	99.8	100.0
	Confidence summation	90.2	97.2	99.8	100.0
	Weighted confidence summation	90.2	97.6	99.8	100.0
	Probabilistic confidence estimation	88.2	97.6	99.8	100.0
	Fuzzy integral	84.8	94.7	99.4	100.0
	Dynamic selection	84.4	93.6	99.8	
k = number of test images for which *at least one* of ML, HL, and CL outputs the true object = 468 in both experiments					

defined in (3.54) are listed in Table 5.9. For measuring the combination performance, the basis is the number images for which *at least one* candidate, among all candidates from ML, HL, and CL, is the true object, which is 468 in both experiments. Those are the images for which we can expect to improve the location performance by combination.

For all the combination methods, the combination performance at top two choices is significantly higher than that at the top choice. This shows that in most cases a combination method fails to locate the true object as the top choice, it locates the true object as the second choice.

Focusing on the top-choice combination performance in Table 5.9, we make the following inferences:

1. The highest confidence method, confidence summation, weighted confidence summation, and probabilistic confidence estimation have similar levels of performance.

2. The fuzzy integral method and dynamic selection behave very similarly and their performance is somewhat lower than that of other methods.

3. The use of correspondence degrees as weights in weighted confidence summation does not improve the performance over (unweighted) confidence summation.

4. The simplest method, the highest confidence method, has the performance similar to that of other methods. This shows that objective confidence values reflect locator behavior and are useful indeed.

5. Probabilistic confidence estimation, despite additional training to determine the estimation functions, is not clearly better than other methods.

To see why the second observation is so, we recall that both methods depend heavily on dynamic locator relevance values based on agreement classes. To see how accurately the dynamic relevance values are used, we should first analyze the accuracy of agreement class decisions.

Table 5.10 shows the results of agreement class decisions in the form of a confusion matrix. The decisions are made by deciding correspondence among the top-choice objects of the locators and are not perfect. Given a test image, if we make a wrong agreement class decision, then the fuzzy integral method and dynamic selection will use a wrong set of dynamic relevance values, which might lead to an incorrect top choice.

To see the distribution of top-choice combination successes among agreement classes, we can separate the test images by agreement classes and evaluate the combination performance for each class. The combination performance of top choices according to the true agreement class is shown in Table 5.11. An obvious inference from the table is that the performance for the 'all agree' class is perfect and that for 'none agrees' class is the lowest. As the extent of the top-choice agreement indicates the difficulty with locating the true object, this observation is well-explained. The

Table 5.10: Agreement class decision results

Experiment	True class (agreeing locators)	Number of images in computed class					Total	% correct
		1	2	3	4	5		
1	1: None	229	2	13	8	0	252	90.9
	2: ML,HL	6	22	0	0	1	29	75.9
	3: ML,CL	11	0	72	1	8	92	78.3
	4: HL,CL	14	0	0	68	2	84	81.0
	5: All	1	19	4	1	68	93	73.1
	Total	261	43	89	78	79	550	
2	1: None	226	1	12	20	1	260	86.9
	2: ML,HL	2	12	0	0	6	20	60.0
	3: ML,CL	5	0	66	0	10	81	81.5
	4: HL,CL	7	0	0	79	2	88	89.8
	5: All	0	5	1	1	94	101	93.1
	Total	240	18	79	100	113	550	

Table 5.11: Combination performance of top choices separated by agreement class

Experiment	Combination method	% correct (of k images) in true agreement class				
		None	ML HL	ML CL	HL CL	All
1	Highest confidence	82.0	96.6	86.2	97.5	100.0
	Confidence summation	80.9	96.6	83.9	98.8	100.0
	Weighted confidence summation	80.9	96.6	83.9	98.8	100.0
	Probabilistic confidence estimation	80.9	89.7	88.5	98.8	100.0
	Fuzzy integral	71.9	96.6	78.2	98.8	100.0
	Dynamic selection	71.9	96.6	78.2	98.8	100.0
	k	178	29	87	81	93
2	Highest confidence	78.8	95.0	88.7	92.9	100.0
	Confidence summation	79.8	95.0	95.8	96.4	100.0
	Weighted confidence summation	79.3	95.0	95.8	97.6	100.0
	Probabilistic confidence estimation	79.3	95.0	84.5	98.8	100.0
	Fuzzy integral	71.0	95.0	84.5	96.4	100.0
	Dynamic selection	70.5	95.0	83.1	96.4	100.0
	k	193	20	71	84	100

k = number of test images for which *at least one* of ML, HL, and CL outputs the true object three 'two agree' classes have intermediate performance levels and there is no clear evidence that one class is easier to handle than the other two.

Table 5.11 also shows that the fuzzy integral method and dynamic selection have the lowest performance for the 'none agrees' class. It is because there are a variety of situations that can exist for the class, but the two methods use a fixed set of dynamic relevance values for the class, which only indicates very general behavior of locators when none of the top choices agree.

Chapter 6

Conclusions

This chapter concludes the thesis. Section 6.1 gives conclusions drawn from the research presented in the thesis. Section 6.2 describes the areas of future research.

6.1 Research Conclusions

We developed a theory of document object locator combination. The theory reduces the locator combination problem to a confidence value combination problem. In the process, it offers methods for tackling some important issues of the problem.

Results of locators at different information levels are represented using objective confidence values. This representation has a consistent interpretation and thus allows for combining locators at any information levels. Practical methods for computing objective confidence values for locators at all information levels are developed.

Partially overlapping objects that are output from different locators are handled using object equivalence and object correspondence. A method of determining if two objects from different locators correspond to each other is proposed. Correspondence decision results play a central role in locator combination.

Knowledge of the locator behavior is used in two ways. Objective confidence values themselves contain the knowledge because they are obtained by probabilistic interpretation of the locator behavior. Static and dynamic locator relevance values are extracted from the knowledge and used in confidence value combination and dynamic locator selection, respectively.

Five methods of combining confidence values are proposed: the highest confidence, confidence summation, weighted confidence summation, probabilistic confidence estimation, and fuzzy integral methods. However, from the experimental results, we infer that the selection of the combination method is not the most significant factor that affects the combination performance.

A method of dynamic locator selection is proposed, which uses training image set partitioning by agreement among top candidates from multiple locators. Dynamic locator relevance values are used directly for selecting the most appropriate locator dynamically.

Combination performance is distinguished from location performance improvement due to combination. A measure of combination performance is proposed and shown to be more useful than other measures.

We developed a multiple locator algorithm for address block location to test our theory. The algorithm employs three locators: a machine-printed address block locator, a handwritten address block locator, and a clustering-based locator. The locators are relatively simple and based on different assumptions on input mail piece images. The combination algorithm is a direct implementation of the theory.

We did experiments on address block location using the algorithm. Two separate experiments, both with a training set of 550 mail piece images and a test set of 550 images, were performed. Location performance of the combination is shown to be better than any individual locator performance, for all five combination methods and dynamic selection. When both experiments are taken into account, the combination performance figures ranged from 84.4% to 90.2% considering only the top choice, and from 92.5% to 97.6% considering top two choices. The experimental results show that combining document object locators, where each locator is simple and tuned to different object characteristics, is a promising approach to document object location.

6.2 Future Research

Multiple True Objects

In our research, we assumed that there is only one object of interest in a document image. However, this restriction can be awkward for some problems. Extending the theory to handle cases where there can be multiple true objects can be done by introducing the concept of *meta-object*. A meta-object is defined as a group of objects which together constitute a candidate for the true objects. For example, if we are locating text objects in a document image, a group of candidate text blocks output by a locator is a meta-object. All the operations performed on an object in our theory can be done on a meta-object.

Applications to Other Problems

The proposed theory has been developed for the purpose of combining multiple document object locators. However, we consider it to be general enough to be applicable to problems other than document object location. Specifically, it can be applied to problems in which there is only one object of interest to locate. For example, face location, vehicle license plate location, and hand location in a gesture-based human-computer interface are problems where the constraint of only one true object per image is naturally satisfied. With little modifications in implementation details, our theory can be applied to such problems when multiple locators are available.

Image Partitioning for Dynamic Selection

We used the agreement among top candidates as the criterion for partitioning the training image set and calculating dynamic locator relevance, because it is simple to determine and indicates the overall difficulty of the input image to some extent.

However, image-based approaches to image partitioning that directly relates image features and locator performances still need to be investigated. The features to use in those approaches are bound to be specific to a particular application domain.

Guaranteeing Combination Success

In this thesis, we tested the validity of our theory using experiments on address block location. Although research of this kind is almost always experimental, it would be nice if we can prove that our theory guarantees that a locator combination based on it will be successful. This will involve proving that the location performance of the locator combination is always better than that of the best locator. For such a theoretical proof to be feasible, we will need to put some constraints on the similarity of locator behavior on training images to that on test images. The complementarity of locators on the test set will also need to be assumed to exceed a certain level.

Appendix A

Mathematical Details

This appendix gives some of the mathematical details needed for the experiments. Section A.1 covers methods for finding regression coefficients used in our experiments, Section A.2 presents fuzzy measures and fuzzy integrals, and Section A.3 describes minimization of within-group variance for image binarization.

A.1 Estimating Regression Coefficients

In this section, we describe the least squares method and the weighted least squares method. They are used in this thesis for estimating regression coefficients for linear regression and logistic regression, respectively.

A.1.1 Least Squares

The development of the least squares method is based on [23, pages 529–531]. Suppose we have a multiple linear regression function

$$E(Y) = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k, \tag{A.1}$$

and a set of n data points

$$\{(x_{i1}, \ldots, x_{ik}, y_i) \mid i = 1, \ldots, n\} \tag{A.2}$$

for computing the regression coefficients. The least squares estimates of $\beta_0, \ldots, \beta_k$ are the values $\hat{\beta}_0, \ldots, \hat{\beta}_k$ which minimize the quantity

$$Q = \sum_{i=1}^{n} [\, y - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik})]^2. \tag{A.3}$$

Differentiating the quantity partially with respect to the $\hat{\beta}$'s and equating the partial derivatives to zero, we obtain

$$\begin{aligned} \frac{\partial Q}{\partial \hat{\beta}_0} &= \sum_{i=1}^{n} -2\, [\, y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik})] = 0 \\ \frac{\partial Q}{\partial \hat{\beta}_1} &= \sum_{i=1}^{n} -2 x_{i1}\, [\, y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik})] = 0 \\ &\;\vdots \\ \frac{\partial Q}{\partial \hat{\beta}_k} &= \sum_{i=1}^{n} -2 x_{ik}\, [\, y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik})] = 0 \end{aligned} \tag{A.4}$$

and finally the $k+1$ normal equations

$$\begin{aligned} \sum y &= \hat{\beta}_0 n + \hat{\beta}_1 \sum x_1 + \cdots + \hat{\beta}_k \sum x_k \\ \sum x_1 y &= \hat{\beta}_0 \sum x_1 + \hat{\beta}_1 \sum x_1^2 + \cdots + \hat{\beta}_k \sum x_k \\ &\;\vdots \\ \sum x_k y &= \hat{\beta}_0 \sum x_k + \hat{\beta}_1 \sum x_k x_1 + \cdots + \hat{\beta}_k \sum x_k^2 \end{aligned} \tag{A.5}$$

where we abbreviated $\sum_{i=1}^{n} x_{i1}$ as $\sum x_1$, $\sum_{i=1}^{n} x_{i1} x_{i2}$ as $\sum x_1 x_2$, and so on.

The linear system of equations can be solved for $\hat{\beta}_0, \ldots, \hat{\beta}_k$ using a standard method such as Gaussian elimination. For example, textbooks on numerical analysis like [11] presents several methods for solving linear systems of equations.

A.1.2 Weighted Least Squares

The description of the weighted least squares method is based on [65, pages 418, 585-586]. Suppose we have a multiple logistic regression function

$$\pi = \frac{\exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)}{1 + \exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k)} \tag{A.6}$$

and a set of n data points

$$\{(x_{i1}, \ldots, x_{ik}, N_i, \pi_i) \mid i = 1, \ldots, n\} \tag{A.7}$$

for computing the regression coefficients, where N_i is the number of samples in the i-th observation. The logit transformation $\pi' = \log_e \left(\frac{\pi}{1-\pi}\right)$ linearizes (A.6) to

$$\pi' = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k. \tag{A.8}$$

The weighted least squares estimates of $\beta_0, \ldots, \beta_k$ are the values $\hat{\beta}_0, \ldots, \hat{\beta}_k$ which minimize the quantity

$$Q_w = \sum_{i=1}^{n} w_i \left[\, \pi' - (\hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_k x_{ik})\right]^2 \tag{A.9}$$

where w_i is the weight of the i-th observation.

It can be shown that when N_i is reasonably large, the variance of π'_i is approximately

$$\sigma^2(\pi'_i) = \frac{1}{N_i \pi_i (1 - \pi_i)}. \tag{A.10}$$

Therefore, the appropriate weights according to the weighted least squares theory are the reciprocals of the variances

$$w_i = N_i \pi_i (1 - \pi_i). \tag{A.11}$$

The weighted least squares method assumes that no $\pi_i = 0$ or 1. If these cases occur, or some π_i's are very close to 0 or 1, such π_i's and accordingly w_i's should be modified. One common approach is to modify them as follows:

$$\pi_i = \begin{cases} \frac{1}{2N_i} & \text{if } \pi_i = 0 \\ 1 - \frac{1}{2N_i} & \text{if } \pi_i = 1 \end{cases} \tag{A.12}$$

The rest of the weighted least squares estimation is the same as the least squares estimation method: differentiating Q_w of (A.9) with respect to the $\hat{\beta}$'s, equating the partial derivatives to zero, and solving the resulting linear system of equations for $\hat{\beta}$'s. Specifically, the resulting linear system is

$$\begin{aligned} \sum y &= \hat{\beta}_0 \sum w \quad + \hat{\beta}_1 \sum w x_1 \quad + \cdots + \hat{\beta}_k \sum w x_k \\ \sum x_1 y &= \hat{\beta}_0 \sum w x_1 + \hat{\beta}_1 \sum w x_1^2 \quad + \cdots + \hat{\beta}_k \sum w x_k \\ &\vdots \\ \sum x_k y &= \hat{\beta}_0 \sum w x_k + \hat{\beta}_1 \sum w x_k x_1 + \cdots + \hat{\beta}_k \sum w x_k^2 \end{aligned} \tag{A.13}$$

where we abbreviated $\sum_{i=1}^{n} w_i$ as $\sum w$, $\sum_{i=1}^{n} w_i x_{i1}$ as $\sum w x_1$, $\sum_{i=1}^{n} w_i x_{i1} x_{i2}$ as $\sum w x_1 x_2$, and so on.

A.2 Fuzzy Measures and Fuzzy Integrals

We describe fuzzy measures and fuzzy integrals following the development of Tahani and Keller [90]. Suppose we have a set of information sources $\mathbf{X} = \{x_1, \ldots, x_n\}$. These sources provide measurements of some phenomenon, which can be probabilities or beliefs that some condition holds, scores for some object, or decision values. Let us denote the set of all subsets of $\mathbf{X}$, the power set of $\mathbf{X}$, as $2^{\mathbf{X}}$. A fuzzy measure

$g : 2^{\mathbf{X}} \rightarrow [0, 1]$ is a set function with the following properties:

$$g(\phi) = 0, \tag{A.14}$$

$$g(\mathbf{X}) = 1, \tag{A.15}$$

$$\mathbf{A} \subseteq \mathbf{B} \Rightarrow g(\mathbf{A}) \leq g(\mathbf{B}). \tag{A.16}$$

A fuzzy measure is not a probability measure such that there is no particular requirement that $\sum_{i=1}^{n} g(\{x_i\}) = 1$. We can think of $g(\mathbf{A})$ as the reliability or usefulness of the set of information sources $\mathbf{A}$.

Let the measurement provided by the source x be $h(x)$. In addition, we rearrange the sources so that $h(x_1) \geq \cdots \geq h(x_n)$. Let the set of the first i sources in $\mathbf{X}$ be $\mathbf{A}_i$. That is, $\mathbf{A}_i = \{x_1, \ldots, x_i\}$. Then a fuzzy integral e, with respect to a fuzzy measure g over $\mathbf{X}$, is defined as

$$e = \max_{1 \leq i \leq n} [\min(h(x_i), g(\mathbf{A}_i))]. \tag{A.17}$$

To illustrate the meaning of the fuzzy integral, consider a quantity $\min_{x \in \mathbf{A}} h(x)$, where $\mathbf{A}$ is a subset of sources. It is reasonable to think of the quantity as the safest decision provided by the subset of sources. If we let $\mathbf{A} = \mathbf{A}_i$, then $\min_{x \in \mathbf{A}} h(x) = h(x_i)$. In addition, $g(\mathbf{A}_i)$ represents the degree of importance of the subset of sources $\mathbf{A}_i$. Thus $\min(h(x_i), g(\mathbf{A}_i))$ can be interpreted as the grade of agreement between the best security decision of $\mathbf{A}_i$ and the expected importance of $\mathbf{A}_i$. Therefore, the fuzzy integral searches for the maximum agreement between measurement and source importance.

To use the fuzzy integral we need to specify the fuzzy measure values for all possible subsets of $\mathbf{X}$. This requires specifying 2^n values. However, it is possible to reduce the number of fuzzy measure values to be determined from training data, using the properties of the g_λ-fuzzy measure introduced by Sugeno [88]. A g_λ-fuzzy measure is a fuzzy measure satisfying the additional property: for some $\lambda > -1$ and for all $\mathbf{A}, \mathbf{B} \in 2^{\mathbf{X}}$ and $\mathbf{A} \cap \mathbf{B} = \phi$:

$$g(\mathbf{A} \cup \mathbf{B}) = g(\mathbf{A}) + g(\mathbf{B}) + \lambda\, g(\mathbf{A})g(\mathbf{B}). \tag{A.18}$$

The g_λ-fuzzy measure for $\lambda = 0$ is a probability measure and obeys $\sum_{i=1}^{n} g(\{x_i\}) = 1$.

Suppose $\mathbf{A} = \{x_{i_1}, \ldots, x_{i_m}\} \subseteq \mathbf{X}$ and let $g^i = g(\{x_i\})$. The mapping $x_i \to g^i$ is called a fuzzy density function. By repeated application of (A.18), we can write

$$\begin{aligned} g(\mathbf{A}) &= \sum_{j=1}^{m} g^{i_j} + \lambda \sum_{j=1}^{m-1} \sum_{k=j+1}^{m} g^{i_j} g^{i_k} + \cdots + \lambda^{m-1} g^{i_1} \cdots g^{i_m} \\ &= \left[\prod_{x_i \in \mathbf{A}} (1 + \lambda g^i) - 1 \right] / \lambda, \quad \lambda \neq 0. \end{aligned} \tag{A.19}$$

Since by definition $g(\mathbf{X}) = 1$, we obtain the equation

$$\lambda + 1 = \prod_{i=1}^{n} (1 + \lambda g^i). \tag{A.20}$$

Thus if we know the fuzzy densities, $g^i, i = 1, \ldots, n$, we can construct the g_λ-fuzzy measure. For combination problems, g^i can be regarded as the degree of importance of the source x_i towards the final decision. This means that if we know individual source importance we can compute the importance degree of a set of particular sources, by solving (A.20) for λ and using (A.18). Then we finally use the fuzzy integral to generate a consensus decision.

Given g^i values, expanding (A.20) results in a polynomial equation of order $(n-1)$ in λ. Both Leszczynski [55] and Tahani and Keller [90] prove that for a fixed set of $\{g^i\}, 0 < g^i < 1$, there exists a unique $\lambda \in (-1, \infty), \lambda \neq 0$, which satisfies (A.20). Therefore, we can always find a λ value that will enable us to use the fuzzy integral.

Example Table A.1 shows measurements and fuzzy densities for three sources. Using the values of g^1, g^2, and g^3 in (A.20), we obtain $0.096\lambda^2 + 0.76\lambda + 0.6 = 0$. The value of $\lambda > -1$ for this equation is $\lambda = -0.8894$. Then, using (A.18), we obtain

Table A.1: Example of measurements and fuzzy densities

Source set	Measurement (h)	Fuzzy density (g)
ϕ		0
$\{x_1\} = \mathbf{A}_1$	0.7	0.6
$\{x_2\}$	0.5	0.2
$\{x_3\}$	0.4	0.8
$\{x_1, x_2\} = \mathbf{A}_2$		0.6933
$\{x_1, x_3\}$		0.9731
$\{x_2, x_3\}$		0.8577
$\{x_1, x_2, x_3\} = \mathbf{A}_3 = \mathbf{X}$		1

$g(\{x_1, x_2\}), g(\{x_1, x_3\})$ and $g(\{x_2, x_3\})$. Hence, the fuzzy integral value is

$$\begin{aligned}
\max_{1 \leq i \leq 3} [\min(h(x_i), g(\mathbf{A}_i)] &= \max [\min(h(x_1), g(\mathbf{A}_1)), \min(h(x_2), g(\mathbf{A}_2)), \\
&\qquad \min(h(x_3), g(\mathbf{A}_3))] \\
&= \max [\min(0.7, 0.6), \min(0.5, 0.6933), \min(0.4, 1))] \\
&= 0.6.
\end{aligned}$$

A.3 Minimizing Within-Group Variance

Originally described in [68], Otsu's thresholding method finds a threshold value for an input image using the gray value distribution in the image. There are a few different algorithms to determine the threshold value defined by the method, but any such algorithm results in the same threshold value. We follow the algorithm given in [29, pages 20–23].

A threshold value applied on an image defines two groups of pixels: one group of pixels with values lower than the threshold and the other group of pixels with values higher than the threshold. The algorithm finds the threshold value that results in the most homogeneous groups. The homogeneity of a group can be measured by the variance of pixel values within the group. The algorithm searches for the threshold value that minimizes the weighted sum of group variances, called the *within-group variance.*

Let $P(1), \ldots, P(I)$ be the histogram probability of the gray values $1, \ldots, I$ in an image. Let $\sigma_1^2(t)$ be the variance for the group with values less than or equal to t and $\sigma_2^2(t)$ be the variance for the group with values greater than t. Let $q_1(t)$ and $q_2(t)$ be the probabilities for the first group and the second group, respectively. Let $\mu_1(t)$ and $\mu_2(t)$ be the means for the first group and the second group, respectively. Then, the within-group variance is defined as

$$\sigma_W^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{A.21}$$

where

$$\begin{aligned}
q_1(t) &= \sum_{i=1}^{t} P(i) \\
q_2(t) &= \sum_{i=t+1}^{I} P(i) && \text{(A.22)} \\
\mu_1(t) &= \sum_{i=1}^{t} P(i)/q_1(t) \\
\mu_2(t) &= \sum_{i=t+1}^{I} P(i)/q_2(t) && \text{(A.23)} \\
\sigma_1^2(t) &= \sum_{i=1}^{t} [i - \mu_1(t)]^2 \, P(i)/q_1(t) \\
\sigma_2^2(t) &= \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \, P(i)/q_2(t) && \text{(A.24)}
\end{aligned}$$

The threshold value is the one that minimizes $\sigma_W^2(t)$. This value can be found by a sequential search through all possible values of t.

There is a relationship between the total variance σ^2 and the within-group variance $\sigma_W^2(t)$ that simplifies finding the threshold:

$$\sigma^2 = \sigma_W^2(t) + \sigma_B^2(t) \tag{A.25}$$

where $\sigma_B^2(t)$ is the *between-group variance* which can be computed as

$$\sigma_B^2(t) = q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2. \tag{A.26}$$

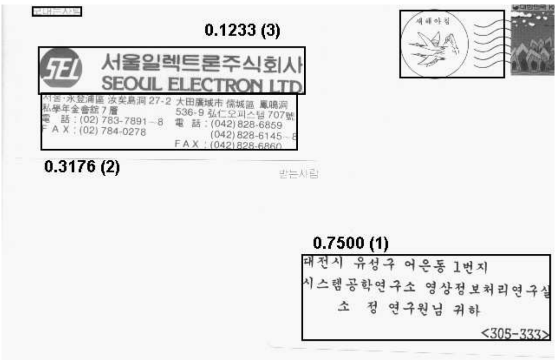
Therefore, the t maximizing $\sigma_B^2(t)$ will be the same as the t minimizing $\sigma_W^2(t)$. We use this alternate method of finding the threshold value because it is simpler. Details on deriving (A.25) and (A.26) are given in [29, pages 21–22].

Appendix B

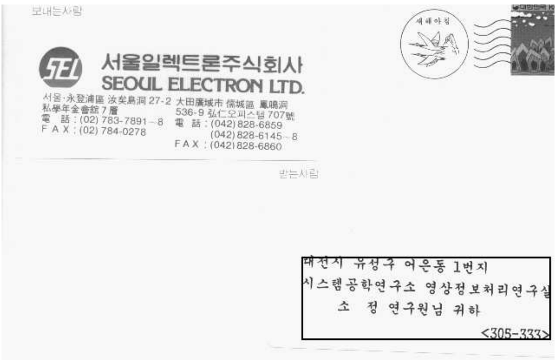
Examples of Address Block Locator Combination

This appendix contains six examples of address block locator combination from 550 test cases of Experiment 1 described in Chapter 5. For each test image, the machine-printed address block locator (ML), the handwritten address block locator (HL), and the clustering-based locator (CL) are applied to produce three sets of candidate blocks. Only those blocks with the objective confidence greater than 0.01 are used for combination. The correspondence of blocks are determined. Based on the decisions and the confidence values, each combination method produce a list of blocks ordered by combined confidence values. The result of dynamic selection is also shown.

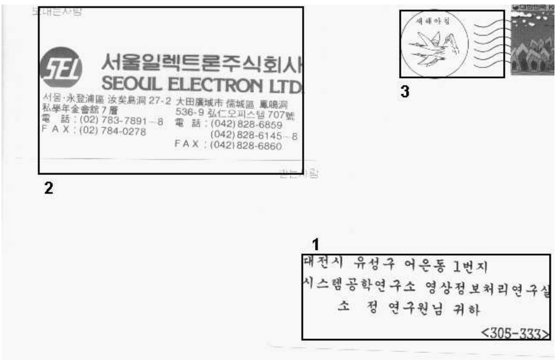
In the examples, we use the abbreviation `m1` for the first block output by ML, `m2` for the second one by ML, and so on. The numbers below five combination methods are the combined confidence values for the blocks, whereas those below dynamic selection are the dynamic relevance values of locators for the agreement class. For a group of equivalent blocks, only the block with the highest objective confidence is output and when a tie exists in final block sorting, objective confidence values are used to break it.



(a) ML



(b) HL



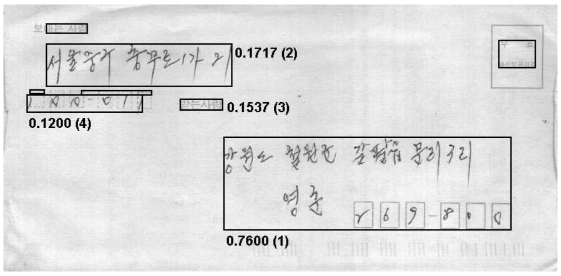
(c) CL

Block	True object?	Objective confidence	Correspondence	Equivalence
m1	yes	0.7500 -> 0.8368	{m1,h1,c1}	{m1,c1}
m2	no	0.3176 -> 0.0355	{m2}	{m2}
h1	yes	0.7167	{h1,m1,c1}	{h1}
c1	yes	1 -> 0.5182	{c1,m1,h1}	{c1,m1}
c2	no	2 -> 0.0109	{c2}	{c2}

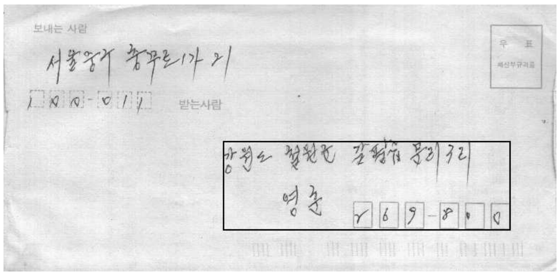
`Agreement class: all agree`

Highest confidence	Confidence summation	Weighted confidence summation	Probabil. confidence estimation	Fuzzy integral	Dynamic selection
m1: 0.8368	m1: 2.0717	m1: 2.0425	m1: 0.9661	m1: 0.8368	m1: 0.9901
h1: 0.8368	h1: 2.0717	h1: 2.0164	h1: 0.9661	h1: 0.8368	h1: 0.9901
m2: 0.0355	m2: 0.0355	m2: 0.0355	m2: 0.0355	m2: 0.0355	c1: 0.9901
c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	

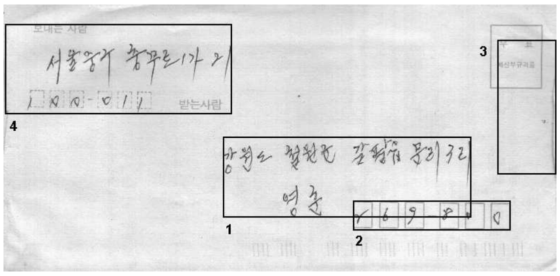
Figure B.1: Combination example 1: All locators make correct top choices which are all corresponding. Combining `m1`, `h1`, and `c1` by any method produces the highest confidence, and all top choices are correct.



(a) ML



(b) HL



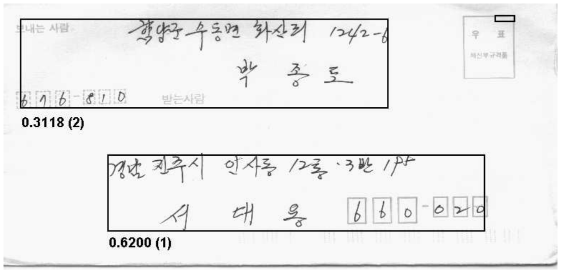
(c) CL

Block	True object?	Objective confidence	Correspondence	Equivalence
m1	yes	0.7600 -> 0.8518	{m1,h1}	{m1}
h1	yes	0.7167	{h1,m1}	{h1}
c1	no	1 -> 0.5182	{c1}	{c1}
c2	no	2 -> 0.0109	{c2}	{c2}

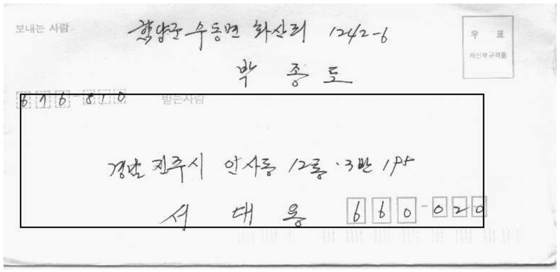
Agreement class: ML and HL agree

Highest confidence	Confidence summation	Weighted confidence summation	Probabil. confidence estimation	Fuzzy integral	Dynamic selection
m1: 0.8518	m1: 1.5685	m1: 1.5352	m1: 0.8393	m1: 0.8518	m1: 0.9500
h1: 0.8518	h1: 1.5685	h1: 1.5289	h1: 0.8393	h1: 0.8518	h1: 0.9500
c1: 0.5182	c1: 0.5182	c1: 0.5182	c1: 0.5182	c1: 0.0500	c1: 0.0500
c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	

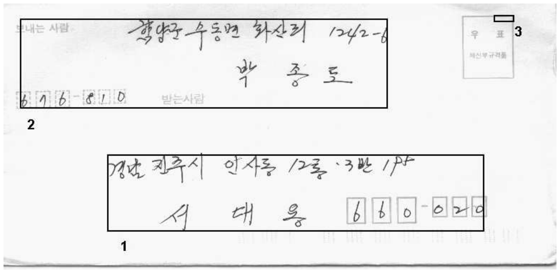
Figure B.2: Combination example 2: ML and HL make correct top choices which are corresponding. Combining `m1` and `h1` by any method produces a higher confidence value than that for `c1`, and all top choices are correct.



(a) ML



(b) HL



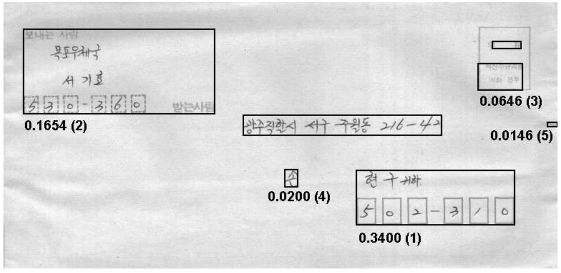
(c) CL

Block	True object?	Objective confidence	Correspondence	Equivalence
m1	yes	0.6200 -> 0.5377	{m1,c1}	{m1,c1}
m2	no	0.3118 -> 0.0334	{m2,c2}	{m2,c2}
h1	no	0.7167	{h1}	{h1}
c1	yes	1 -> 0.5182	{c1,m1}	{c1,m1}
c2	no	2 -> 0.0109	{c2,m2}	{c2,m2}

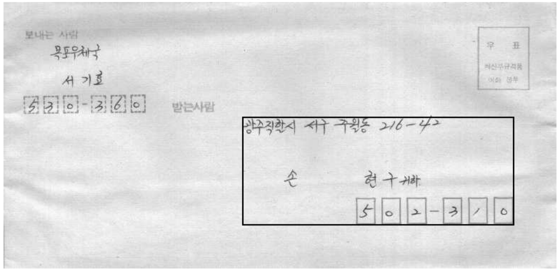
Agreement class: ML and CL agree

Highest confidence	Confidence summation	Weighted confidence summation	Probabil. confidence estimation	Fuzzy integral	Dynamic selection
h1: 0.7167	m1: 1.0559	m1: 1.0559	h1: 0.7167	m1: 0.5377	m1: 0.7407
m1: 0.5377	h1: 0.7167	h1: 0.7167	m1: 0.6713	h1: 0.2037	c1: 0.7407
m2: 0.0334	m2: 0.0443	m2: 0.0443	m2: 0.0197	m2: 0.0334	h1: 0.2037

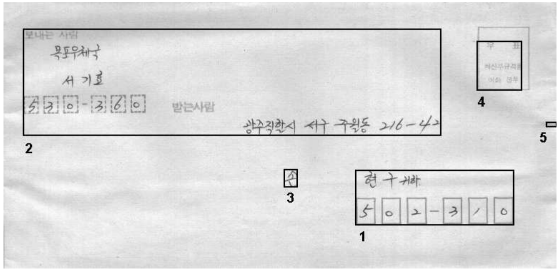
Figure B.3: Combination example 3: ML and CL make correct top choices which are equivalent. Combining `m1` and `c1` by the highest confidence method or probabilistic confidence estimation produces a lower confidence value than that for `h1`, leading to the incorrect top choice.



(a) ML



(b) HL



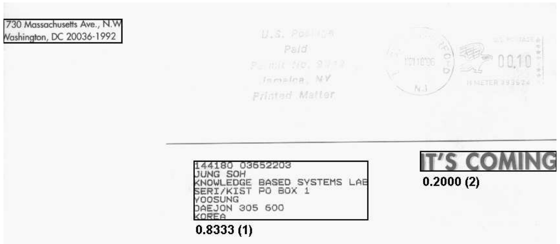
(c) CL

Block	True object?	Objective confidence	Correspondence	Equivalence
m1	no	0.3400 -> 0.0454	{m1,c1}	{m1,c1}
h1	yes	0.7167	{h1}	{h1}
c1	no	1 -> 0.5182	{c1,m1}	{c1,m1}
c2	no	2 -> 0.0109	{c2}	{c2}

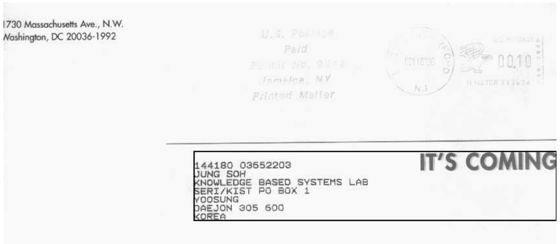
Agreement class: ML and CL agree

Highest confidence	Confidence summation	Weighted confidence summation	Probabil. confidence estimation	Fuzzy integral	Dynamic selection
h1: 0.7167	h1: 0.7167	h1: 0.7167	h1: 0.7167	c1: 0.5182	c1: 0.7407
c1: 0.5182	c1: 0.5636	c1: 0.5636	c1: 0.0981	h1: 0.2037	m1: 0.7407
c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	h1: 0.2037

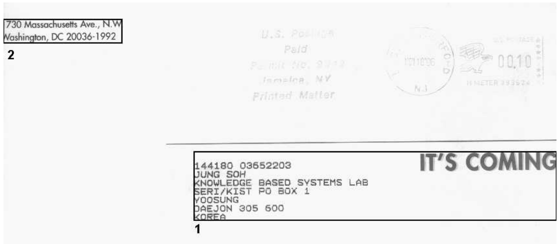
Figure B.4: Combination example 4: ML and CL make wrong top choices which are equivalent. The fuzzy integral method and dynamic selection use misleading dynamic relevance and make the incorrect top choice. Top choices of other methods are correct, because combining `m1` and `c1` produces a lower confidence value than that for `h1`, due to the very low confidence for `m1`.



(a) ML



(b) HL



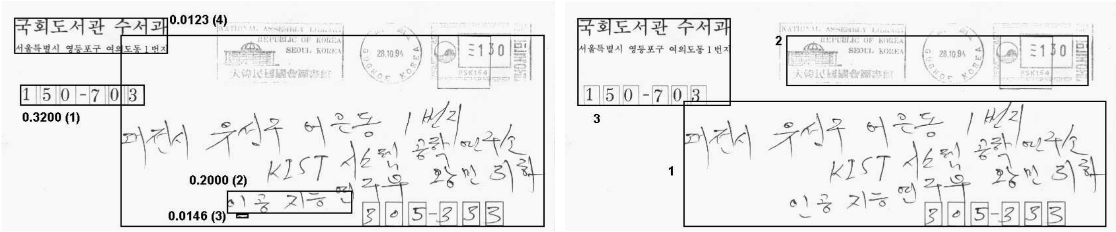
(c) CL

Block	True object?	Objective confidence	Correspondence	Equivalence
m1	yes	0.8333 -> 0.9299	{m1}	{m1}
h1	no	0.7167	{h1,c1}	{h1}
c1	no	1 -> 0.5182	{c1,h1}	{c1}
c2	no	2 -> 0.0109	{c2}	{c2}

Agreement class: HL and CL agree

Highest confidence	Confidence summation	Weighted confidence	Probabil. confidence	Fuzzy integral	Dynamic selection
m1: 0.9299	h1: 1.2349	h1: 1.2300	h1: 0.9318	h1: 0.7167	h1: 0.9318
h1: 0.7167	c1: 1.2349	c1: 1.2281	c1: 0.9318	c1: 0.7167	c1: 0.9318
c1: 0.5182	m1: 0.9299	m1: 0.9299	m1: 0.9299	m1: 0.0227	m1: 0.0227
c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	

Figure B.5: Combination example 5: HL and CL make wrong top choices which are corresponding. Only the highest confidence method makes the correct top choice, because the objective confidence for `m1` is higher than that for `h1` or `c1`.



(a) ML (b) CL

Block	True object?	Objective confidence	Correspondence	Equivalence
m1	no	0.3200 -> 0.0365	{m1}	{m1}
c1	yes	1 -> 0.5182	{c1}	{c1}
c2	no	2 -> 0.0109	{c2}	{c2}

Agreement class: none agrees

Highest confidence	Confidence summation	Weighted confidence summation	Probabil. confidence estimation	Fuzzy integral	Dynamic selection
c1: 0.5182	c1: 0.5182	c1: 0.5182	c1: 0.5182	c1: 0.1615	c1: 0.1615
m1: 0.0365	m1: 0.0365	m1: 0.0365	m1: 0.0365	m1: 0.0365	m1: 0.1231
c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	c2: 0.0109	

Figure B.6: Combination example 6: HL rejects the image and only CL makes the correct top choice. No two blocks correspond and no combination takes place. Since the objective confidence for `m1` is very low, all methods make the correct top choice.

References

[1] L. Anderson, R. Bajcsy, and M. Mintz, "A modular feedback system for image segmentation," Technical Report 110, University of Pennsylvania GRASP Lab., June 1987.

[2] T. Antognini and L. Turnbaugh, "An expert system for the location of destination address in mail of arbitrary complexity," *Proceedings of the Third USPS Advanced Technology Conference*, 279–292, Washington, DC, May 1988.

[3] T. Arbuckle, E. Lange, F. Coudail, T. Iwamoto, M. Takahashi, T. Ishii, H. Mori, K. Kyuma, and N. Otsu, "Face recognition system incorporating classifier combination by fuzzy integral," *Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, 393–394, Iizuka, Japan, August 1994.

[4] T. D. Arbuckle, E. Lange, T. Iwamoto, N. Otsu, and K. Kyuma, "Fuzzy information fusion in a face recognition system," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 3(3), 217–246, 1995.

[5] J. A. Barnett, "Computational methods for a mathematical theory of evidence," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 868–875, Vancouver, Canada, 1981.

[6] N. Bartneck, "Knowledge based address block finding using hybrid knowledge representation schemes," *Proceedings of the Third USPS Advanced Technology Conference*, 249–263, Washington, DC, May 1988.

[7] A. Bergman, E. Bracha, P. G. Mulgaonkar, and T. Shaham, "Advanced research in address block location," *Proceedings of the Third USPS Advanced Technology Conference*, 218–232, Washington, DC, May 1988.

[8] D. Black, *The Theory of Committees and Elections*, Cambridge University Press, 1958, 1963.

[9] R. Brunelli and D. Falavigna, "Person identification using multiple cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(10), 955–966, October 1995.

[10] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, 1984.

[11] R. L. Burden and J. D. Faires, *Numerical Analysis*, PWS-KENT, 1993.

[12] J. Cao, M. Shridhar, and M. Ahmadi, "Fusion of classifiers with fuzzy integrals," *Proceedings of the Third International Conference on Document Analysis and Recognition*, I, 108–111, Montréal, Canada, August 1995.

[13] S.-B. Cho and J. H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Transactions on Systems, Man, and Cybernetics*, 25(2), 380–384, February 1995.

[14] C.-C. Chu and J. K. Aggarwal, "The integration of image segmentation maps using region and edge information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12), 1241–1252, December 1993.

[15] A. De Gaetano, "Approximate reasoning methodologies and their application on a knowledge based system for the "Address Block Location" problem," *Proceedings of the Third USPS Advanced Technology Conference*, 233–248, Washington, DC, May 1988.

[16] A. C. Downton and C. G. Leedham, "Preprocessing and presorting of envelope images for automatic sorting using OCR," *Pattern Recognition*, 23(3/4), 347–362, 1990.

[17] B. Duerr, W. Haettich, H. Tropf, and G. Winkler, "A combination of statistical and syntactical pattern recognition applied to classification of unconstrained handwritten numerals," *Pattern Recognition*, 12, 189–199, 1980.

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

[19] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy, "The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty," *ACM Computing Surveys*, 12(2), 213-253, 1980

[20] K. Etemad, D. S. Doermann, and R. Chellappa, "Multiscale document page segmentation using soft decision integration," Technical Report CAR-TR-761/CS-TR-3444, Center for Automation Research, University of Maryland, March 1995.

[21] K. Etemad, D. Doermann, and R. Chellappa, "Page segmentation using decision integration and wavelet packets," *Proceedings of the 12th International Conference on Pattern Recognition*, II, 345–349, Jerusalem, Israel, October 1994.

[22] J. Franke and E. Mandler, "A comparison of two approaches for combining the votes of cooperating classifiers," *Proceedings of the 11th International Conference on Pattern Recognition*, II, 611–614, The Hague, The Netherlands, August 1992.

[23] J. E. Freund, *Mathematical Statistics*, Fifth Edition, Prentice Hall, 1992.

[24] P. Fua and T. Hanson, "Using generic geometric models for intelligent shape extraction," *Proceedings of the DARPA Image Understanding Workshop*, 227–233, 1987.

[25] P. D. Gader and M. A. Mohamed, "Choquet fuzzy integral in handwritten word recognition," *SPIE Vol. 2660—Document Recognition III*, 309–320, January 1996.

[26] P. D. Gader and M. A. Mohamed, "Multiple classifier fusion for handwritten word recognition," *Proceedings of the 1995 IEEE International Conference on Systems, Man, and Cybernetics*, 3, 2329–2334, Vancouver, Canada, October 1995.

[27] J. Ghosh, S. D. Beck, and C.-C. Chu, "Evidence combination techniques for robust classification of short-duration oceanic signals," *SPIE Vol. 1706—Adaptive and Learning Systems*, 266–276, April 1992.

[28] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), 993–1001, October 1990.

[29] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, II, Addison-Wesley, 1993.

[30] T. K. Ho, "A theory of multiple classifier systems and its application to visual word recognition," Ph.D. Thesis, Department of Computer Science, SUNY at Buffalo, May 1992.

[31] T. K. Ho, J. J. Hull, and S. N. Srihari, "Combination of decisions by multiple classifiers," in *Structured Document Image Analysis*, H. Baird, H. Bunke, and K. Yamamoto (eds.), 188–202, Springer-Verlag, 1992.

[32] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1), 66–75, January 1994.

[33] T. K. Ho, J. J. Hull, and S. N. Srihari, "On multiple classifier systems for pattern recognition," *Proceedings of the 11th International Conference on Pattern Recognition*, II, 84–87, The Hague, The Netherlands, August 1992.

[34] Y. S. Huang and C. Y. Suen, "A method of combining multiple classifiers—a neural network approach," *Proceedings of the 12th International Conference on Pattern Recognition*, II, 473–475, Jerusalem, Israel, October 1994.

[35] Y. S. Huang and C. Y. Suen, "A method of combining multiple experts for the recognition of unconstrained handwritten numerals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1), 90–94, January 1995.

[36] Y. S. Huang and C. Y. Suen, "Combination of multiple classifiers with measurement values," *Proceedings of the Second International Conference on Document Analysis and Recognition*, 598–601, Tsukuba Science City, Japan, October 1993.

[37] Y. S. Huang and C. Y. Suen, "The behavior-knowledge space method for combination of multiple classifiers," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York City, New York, June 1993.

[38] A. K. Jain and S. K. Bhattacharjee, "Address block location on envelopes using Gabor filters," *Pattern Recognition*, 25(12), 1459–1477, 1992.

[39] A. K. Jain and S. K. Bhattacharjee, "Address block location on envelopes using Gabor filters: supervised method," *Proceedings of the 11th International Conference on Pattern Recognition*, II, 264–267, The Hague, The Netherlands, August 1992.

[40] A. K. Jain and Y. Chen, "Address block location using color and texture analysis," *CVGIP: Image Understanding*, 60(2), 179–190, September 1994.

[41] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognition*, 24, 1167–1186, 1991.

[42] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*, McGraw-Hill, 1995.

[43] J. Jelinek, K. Schaper, and A. Patani, "Knowledge-based image understanding system for address block location," *Proceedings of the Third USPS Advanced Technology Conference*, 264–278, Washington, DC, May 1988.

[44] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, 32, 241, 1967.

[45] M. Kanel and A. Thao, "Extraction of binary character/graphics images from gray scale document images," *CVGIP: Graphical Models and Image Processing*, 55(3), 203–217, May 1993.

[46] J. M. Keller and R. Krishnapuram, "Fuzzy decision models in computer vision," in *Fuzzy Sets, Neural Networks, and Soft Computing*, R. R. Yager and L. A. Zadeh (eds.), 213–232, Van Nostrand Reinhold, 1994.

[47] F. Kimura and M. Shridhar, "Handwritten numeral recognition based on multiple algorithms," *Pattern Recognition*, 24(10), 969-983, 1991.

[48] J. T. Koljonen and F. R. Glickman, "Comparative study of artificial intelligence techniques as applied to the location of address blocks on mail pieces," *SPIE Vol. 1095—Applications of Artificial Intelligence VII*, 489–502, 1989.

[49] R. Krishnapuram and J. Lee, "Fuzzy-set-based hierarchical networks for information fusion in computer vision," *Neural Networks*, 5, 335–350, 1992.

[50] L. Lam and C. Y. Suen, "A theoretical analysis of the application of majority voting to pattern recognition," *Proceedings of the 12th International Conference on Pattern Recognition*, II, 418–420, Jerusalem, Israel, October 1994.

[51] D.-S. Lee and S. N. Srihari, "A theory of classifier combination: the neural network approach," *Proceedings of the Third International Conference on Document Analysis and Recognition*, I, 42–45, Montréal, Canada, August 1995.

[52] D.-S. Lee and S. N. Srihari, "Dynamic classifier combination using neural network," *SPIE Vol. 2422—Document Recognition II*, 26–37, February 1995.

[53] J. S.-J. Lee, J.-N. Hwang, D. T. Davis, and A. C. Nelson, "Integration of neural networks and decision tree classifiers for automated cytology screening," *Proceedings of the International Joint Conference on Neural Networks*, I, 257–262, Seattle, WA, July 1991.

[54] S.-W. Lee and K.-C. Kim, "Address block location on handwritten Korean envelopes by the merging and splitting method," *Pattern Recognition*, 27(12), 1641–1651, December 1994.

[55] K. Leszczynski, P. Penczek, and W. Grochulskki, " Sugeno's fuzzy measures and fuzzy clustering," *Fuzzy Sets and Systems*, 15, 147–158, 1985.

[56] J. Lii, P. W. Palumbo, and S. N. Srihari, "Address block location using character recognition and address syntax," *Proceedings of the Second International Conference on Document Analysis and Recognition*, 330–335, Tsukuba Science City, Japan, October 1993.

[57] J. Lii and S. N. Srihari, "Location of name and address on fax cover pages," *Proceedings of the Third International Conference on Document Analysis and Recognition*, II, 756–759, Montréal, Canada, August 1995.

[58] J. Lii and S. N. Srihari, "Use of character recognition and syntax in locating address paragraphs in complex documents," *Proceedings of the 12th International Conference on Pattern Recognition*, II, 374–376, Jerusalem, Israel, October 1994.

[59] Y. Liu, R. Fenrich, and S. N. Srihari, "An object attribute thresholding algorithm for document image binarization," *Proceedings of the Second Internatio nal Conference on Document Analysis and Recognition*, 278–281, Tsukuba Science City, Japan, October 1992.

[60] S. Madhvanath and V. Govindaraju, "Serial classifier combination for handwritten word recognition," *Proceedings of the Third International Conference on Document Analysis and Recognition*, II, 911–914, Montréal, Canada, August 1995.

[61] T. Matsui, T. Noumi, I. Yamashita, T. Wakahara, and M. Yoshimuro, "State of the art of handwritten numeral recognition in Japan," *Proceedings of the Second International Conference on Document Analysis and Recognition*, 391–396, Tsukuba Science City, Japan, October 1993

[62] B. M. Mehtre and B. Chatterjee, "Segmentation of fingerprint images—a composite method," *Pattern Recognition*, 22(4), 381–385, 1989.

[63] T. Murofushi and M. Sugeno, "A theory of fuzzy measures: representations, the Choquet integral, and null sets," Journal of Mathematical Analysis and Applications, 159, 532–549, 1991.

[64] A. M. Nazif and M. D. Levine, "Low level image segmentation: an expert system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(5), 555–577, September 1984.

[65] J. Neter, W. Wasserman, and M. H. Kunter, *Applied Linear Regression Models*, Second Edition, Irwin, 1989.

[66] H. P. Nii, "Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures, Part One," *AI Magazine*, 7(2), 38–53, 1986.

[67] J.-C. Oriot, D. Barba, and J.-C. Salome, "Address block locating method based on transition analysis approach: design and evaluation on flats objects," *Proceedings of the First International Conference on Document Analysis and Recognition*, II, 665–673, Saint-Malo, France, September 1991.

[68] N. Otsu, "A threshold selection method from gray-scale histogram," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8, 62–66, 1978.

[69] P. W. Palumbo. J. Soh, S. N. Srihari, V. Demjanenko, and R. Sridhar "Real-time address block location using pipelining and multiprocessing," *Proceedings of the Fourth USPS Advanced Technology Conference*, I, 73–87, Washington, DC, November 1990.

[70] P. W. Palumbo, S. N. Srihari, J. Soh, R. Sridhar, and V. Demjanenko, "Postal address block location in real time," *IEEE Computer*, 25(7), 34–42, July 1992.

[71] P. W. Palumbo, P. Swaminathan, and S. N. Srihari, "Document image binarization: evaluation of algorithms," *SPIE Vol. 697—Applications of Digital Image processing IX*, 278–285, 1986.

[72] S. Pankanti and A. K. Jain, "Integrating vision modules: stereo, shading, grouping, and line labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8), 831–842, September 1995.

[73] T. Pavlidis, "Why progress in machine vision is so slow," *Pattern Recognition Letters*, 13(4), 221–225, April 1992.

[74] T. Pavlidis and Y. T. Liow, "Integrating region growing and edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3), 225–233, March 1990.

[75] M. P. Perrone and L. N. Cooper, "When networks disagree: ensemble methods for hybrid neural networks," in *Artificial Neural Networks for Speech and Vision*, R. J. Mammone (ed.), 126–142, Chapman & Hall, 1993.

[76] A. Pinz and R. Bartl, "Information fusion in image understanding," *Proceedings of the 11th International Conference on Pattern Recognition*, I, 366–370, The Hague, The Netherlands, August 1992.

[77] B. Plessis, A. Sicsu, L. Heutte, E. Menu, E. Lecolinet, O. Debon, and J. V. Moreau, "A multi-classifier combination strategy for the recognition of handwritten cursive words," *Proceedings of the Second International Conference on Document Analysis and Recognition*, 642–645, Tsukuba Science City, Japan, October 1993.

[78] R. K. Powalka, N. Sherkat, and R. J. Whitrow, "Recognizer characterization for combining handwriting recognition results at word level," *Proceedings of the Third International Conference on Document Analysis and Recognition*, I, 68–73, Montréal, Canada, August 1995.

[79] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, 77(2), 257–286, February 1989.

[80] H. S. Ranganath and L. J. Chipman, "Fuzzy relaxation approach for inexact scene matching," *Image and Vision Computing*, 10(9), November 1992.

[81] G. Rogova, "Combining the results of several neural network classifiers," *Neural Networks*, 7(5), 777–781, 1994.

[82] C. Ronse and P. A. Devijver, *Connected Components in Binary Images: The Detection Problem*, Research Studies, Letchworth, Herts, England, 1984.

[83] M. Salotti and C. Garbay, "A new paradigm for segmentation," *Proceedings of the 11th International Conference on Pattern Recognition*, III, 611–614, The Hague, The Netherlands, August 1992.

[84] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

[85] D. Sher, "Evidence combination using likelihood generators," *Proceedings of the DARPA Image Understanding Workshop*, II, 655–662, Los Angeles, California, February 1987.

[86] S. Shlien, "Multiple binary decision tree classifiers." *Pattern Recognition*, 23(7), 757–763, 1990.

[87] S. N. Srihari, C.-H. Wang, P. W. Palumbo, and J. J. Hull, "Recognizing address blocks on mail pieces: specialized tools and problem-solving architecture," *AI Magazine*, 8(4), 1987.

[88] M. Sugeno, "Fuzzy measures and fuzzy integrals: a survey," in *Fuzzy Automata and Decision Processes*, M. M. Gupta, G. N. Saridis, and B. R. Gaines (eds.), 89–102, North Holland, 1977.

[89] T. Sugiyama and K. Abe, "Image segmentation using both edge and region information," *Proceedings of the IAPR Workshop and Machine Vision Applications*, 127–130, Kawasaki, Japan, December 1994.

[90] H. Tahani and J. M. Keller, "Information fusion in computer vision using the fuzzy integral," *IEEE Transactions on Systems, Man, and Cybernetics*, 20(3), 733–741, May/June 1990.

[91] K. Tumer and J. Ghosh, "A framework for estimating performance improvements in hybrid pattern classifiers," *Proceedings of the World Congress on Neural Networks*, III, 220–225, San Diego, California, June 1994.

[92] *U.S. Census Bureau / NIST First OCR Systems Conference*, Gaithersburg, Maryland, May 1992.

[93] C.-H. Wang, "A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces," Ph.D. Thesis, Technical Report 88-22, Department of Computer Science, SUNY at Buffalo, May 1988.

[94] C.-H. Wang and S. N. Srihari, "A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces," *International Journal of Computer Vision*, 2, 125–151, 1988.

[95] P. Wang, "The interpretation of fuzziness," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 26(2), April 1996.

[96] G. Wong and H. P. Frei, "Object recognition: the utopian method is dead; the time for combining simple methods have come," *Proceedings of the 11th International Conference on Pattern Recognition*, III, 185–188, The Hague, The Netherlands, August 1992.

[97] Y. Xiaohan, J. Ylä-Jääski, O. Huttunen, T. Vehkomäki, O. Sipilä, and T. Katila, "Image segmentation combining region growing and edge detection," *Proceedings of the 11th International Conference on Pattern Recognition*, I, 481–484, The Hague, The Netherlands, August 1992.

[98] L. Xu, A. Krzyżak, and C. Y. Suen, "Associative switch for combining multiple classifiers," *Proceedings of the International Joint Conference on Neural Networks*, 43–48, Seattle, Washington, July 1991.

[99] L. Xu, A. Krzyżak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3), 418–435, May/June 1992.

[100] P.-S. Yeh, S. Antoy, A. Litcher, and A. Rosenfeld, "Address location on envelopes," *Pattern Recognition*, 20(2), 213–227, 1987.

[101] H.-J. Zimmermann and P. Zysno, "Decisions and evaluations by hierarchical aggregation of information," *Fuzzy Sets and Systems*, 10(3), 243–260, 1983.

[102] J. Zupan, *Clustering of Large Data Sets*, Research Studies Press, 1982.