

 Open access • Proceedings Article • DOI:10.1109/ICDAR.1999.791790

## Parsing and recognition of city, state, and ZIP codes in handwritten addresses

— [Source link](#) 

U. Mahadevan, Sargur N. Srihari

**Institutions:** State University of New York System, University at Buffalo

**Published on:** 20 Sep 1999 - International Conference on Document Analysis and Recognition

**Topics:** Handwriting recognition and Parsing

Related papers:

- [Online and off-line handwriting recognition: a comprehensive survey](#)
- [Integration of hand-written address interpretation technology into the United States Postal Service Remote Computer Reader system](#)
- [A new method of character line extraction from mixed-unformatted document image for Japanese mail address recognition](#)
- [Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading](#)
- [A system towards Indian postal automation](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/parsing-and-recognition-of-city-state-and-zip-codes-in-48ffdxyoxgx>

# Parsing and Recognition of City, State, and ZIPCodes in Handwritten Addresses

Uma Mahadevan \*

Sargur N. Srihari †

## Abstract

*In this paper, we present a solution to the general vision problem of parsing and recognizing a set of correlated entities in the presence of imperfect information. Our solution mechanism involves the generation of multiple hypotheses in the initial stages of the system, and the use of very-large vocabulary recognition, together with a database of all the valid combinations of the correlated entities, to choose among the hypotheses. We have applied our ideas and techniques to the specific task of identifying the city, state and zipcode fields in handwritten addresses. Given the image of a handwritten address, our algorithm produces a ranking of the 76,121-entry database of valid (city, state, zip) triples in the U.S, and in nearly 75% of the cases, the correct entry for the input address is assigned a rank of at most 10.*

## 1 Introduction

Handwritten address interpretation is a computer vision problem with important commercial applications. The task of interpreting a handwritten address is made extremely challenging due to the enormous variability in handwriting style, the presence of very large lexicons for zipcode, city and street names, and the requirements of very low error rates and real-time speeds.

In this paper, we present a system for the parsing and recognition of city, state, and zip fields of handwritten addresses. There are nearly 76,000 valid (city, state, zip) combinations in the United States, and the problem of city-state-zip (CSZ) identification may be viewed as the process of selecting one triple from the set of 76,000 records present in the database. We present the first system to attempt a problem of this magnitude in earnest. Our algorithm produces a ranking of the entire database of triples, and in nearly 75% of the cases, the correct entry for the input address is assigned a rank of at most 10. Our system is also fairly efficient (it takes about 2s on a 300MHz Sun UltraSparc2).

A salient aspect of our system is that it is able to achieve this performance by using simple tools for segmentation and

digit/character/word recognition. Since reliable tools for handwriting are often difficult to obtain or are slow, the system operates under the assumption of receiving incomplete and imprecise information from the various sub-systems. The power of our system stems from using multiple hypotheses in the initial stages of the system, and the use of correlation between the city, state, and zip fields.

The work presented in this paper is relevant not only to handwritten address interpretation, but also to handwriting recognition in general, because many of the tools developed for segmentation and recognition are applicable for other handwriting domains, such as bank checks, forms, questionnaires, surveys, perhaps even unconstrained handwritten documents.

We briefly comment on the role played by the city, state and zip word(s) in handwritten address interpretation. The zipcode is the most important entity, since its correct recognition will automatically determine the city and state of the destination address. Thus, at first glance, it seems appropriate to focus just on the zipcode and essentially ignore the city and state words that are actually written on the address. In fact, this is the view adopted in the system for address interpretation that has recently been deployed by the Postal Service in various sites across the country (Discover Magazine, July 1997, page 82). The above approach works well in addresses where the zipcode is correctly located and recognized. If there is an error in zipcode recognition, this system attempts to realize that an error has occurred by using street name or PO Box number recognition. The drawback to this error detection technique is that a large percentage ( $\approx 30\%$ ) of the images have to be rejected as having an unrecognizable zipcode, in order to maintain a low error rate.

A better approach to zipcode error detection and possible *correction* is to use the city and state information. These words represent *redundant* information that can be used to verify that a zipcode has been correctly recognized. Additionally, they can be used to select among the top few recognition choices for the zipcode. We feel that both these factors are crucial to obtaining an address interpretation system that achieves low error *and* low rejection rates.

The remainder of the paper is organized as follows. Sections 2 and 3 give the details of our system, and the final two sections present experimental results and conclusions.

\*This work was done as part of this author's doctoral dissertation at CEDAR, SUNY at Buffalo. Current address: PROS Strategic Solutions, 3223 Smith Street, Suite 100, Houston, TX 77006. Email: umahadevan@prosweb.com

†CEDAR, SUNY at Buffalo, 520 Lee Entrance, UB Commons, Amherst, NY 14228. Email: srihari@cedar.buffalo.edu

## 2 Segmentation and Parsing

The input to our system is the image of a destination address presented as a list of connected components. The first step is to segment the address into lines of text. We use previously developed modules for address block location, component generation and line separation.

Our first task is the segmentation of the lines of the address into the constituent words. The process of isolating the words in a line is called *word separation*[6, 4, 5]. Our approach to this problem is to generate the gaps between the adjacent connected components in a line and classify the gaps as *word gaps* and *non-word gaps*. To estimate the gaps, we use a convex hull based method [4].



Figure 1. Address Hypotheses

In classifying gaps as word gaps or non-word gaps, we may adopt one of two approaches: (a) each line is partitioned independently of the others by considering the gaps in that line only, and (b) the gaps in the entire address are considered in the partitioning step, and every classification of the gaps splits the entire address into words. We adopted the second approach, because the inclusion of all the gaps results in a better classification into word and non-word gaps, and more importantly, we obtain a small number ( $\leq 3$ ) of *address hypotheses*, which are partitions of the entire address into words (see Figure 1) Such hypotheses are needed in the later stages of our parsing system.

Given a set of words that comprise the address, our next task is to find the subset of words (if any) that represent the city, state and zip fields. Since we assume that the three fields appear one after another, we are in fact looking for a sub-sequence of words that represent the CSZ block. Statistics collected from a large set of addresses indicate that in over 98% of the addresses, the CSZ block occurs either in the last line, the last two lines or the second-to-last line (examples

are shown in Figure 2). Thus we have three ways of selecting the sub-sequence of words representing the CSZ block. Note that at this stage, we only identify one or more candidate blocks that contain the city, state and zip words. We still have to identify which of these words contain the city, which ones contain the state and the zip.

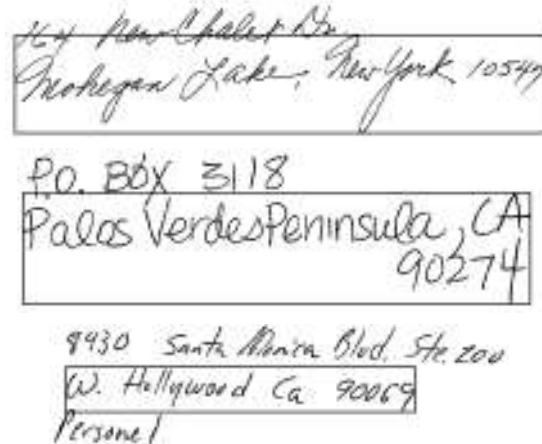


Figure 2. Location of the CSZ Block

Within a CSZ block, there are many possible configurations in which the city, state and zip words can occur, owing to the fact that we have 1-word and 2-word states, multi-word cities and 1-word zipcodes and 2-word zip codes that represent 5-digit and 9-digit codes. In our system, we have developed a model for the CSZ block that indicates the most likely placements of the city, state and zip words within a CSZ block (see [3] for details). In order to employ this model, we perform a coarse classification of the words in a CSZ block, and assign to each word one or more of the tags *text*, *abbreviation* and *digit*. The word tags are used along with the CSZ model to identify the different ways in which a given set of words can be partitioned into city, state and zip categories.

At the end of the segmentation and parsing phase, we produce, on the average, about 5 hypotheses, each of which labels three non-overlapping regions of the address as city, state and zip. Furthermore, we rank the hypotheses based on the results of word separation, and by using coarse “structural characteristics” that are typical of city words, state words, and zip words. Since city recognition and zip recognition are expensive operations involving huge lexicons, we apply the recognition and interpretation steps (described in Section 3) to the three best choices.

## 3 Recognition and Interpretation

Consider a CSZ hypothesis that contains three images, one for the city, one for the state, and one for the zipcode. In the recognition and interpretation phase, we individually recognize each of the three units, and combine their recognition

results using the correlations inferred from the CSZ lexicon (the list of roughly 76,000 possible CSZ triples). Within each individual recognition problem, we perform the following tasks: (a) decompose the word image into a set of sub-images, where each sub-image represents a digit/character, (b) recognize the digit/character sub-images individually, (c) extract the digit/character recognition scores that are relevant to this lexicon entry, and (d) combine them to obtain a final matching score.

### 3.1 Segments, Pieces, and Digits/Characters

In this section, we briefly describe the decomposition of an image into smaller units. The algorithm described is general enough to handle city, state, and zip images. For simplicity of exposition, we use the word “digit” to mean either a digit or an alphabetic character, as the case may be.

First, we obtain a skeleton of the image of the zipcode and decompose the skeleton into tiny units called *segments*. Each segment is intended to represent a stroke or part of a stroke. The purpose of extracting these segments is to identify the break points, that is, the regions in the image where adjacent digits meet. The most likely break points are *ligatures* or connecting segments, shared vertical segments whose upper end point resides in a “valley” region and whose lower end point resides in a “peak” region (see [7, 8] for details). The segments between adjacent break points are collected to form the next higher unit called a *piece*. A piece is intended to represent one complete digit. However, either due to over-segmentation, or because a digit is broken to start with, a piece often represents a portion of a digit rather than a complete digit. Therefore, to apply the digit recognizer, we form *groups* of up to five consecutive pieces. The hope is that there is some sequence of non-overlapping groups that gives the correct decomposition of the zip/city/state image into its constituent digit sub-images. To precisely identify this sequence, we appeal to the lexicon matching algorithm explained in Section 3.2.

In summary, segments are temporary entities that facilitate the location of break points, pieces are non-overlapping sub-images obtained by permanently gluing all the segments between adjacent break points, and groups are possibly overlapping sub-images obtained by tentatively collecting one or more pieces.

The image of each group is sent to the digit recognizer (or character recognizer, for city/state words). In our system, we extract gradient and concavity features [9, 2] from the image of a group and use a linear classification scheme to perform digit and character recognition. The results are stored in a 3-dimensional recognition matrix  $D = d_{ijk}$ , where the entry  $d_{ijk}$  represents the degree to which the  $k$ -th digit class matches the group  $g_{ij}$  formed by combining pieces  $i$  through  $j$ . Similarly, the matrix  $C = c_{ijk}$  stores the character recognition results for city and state recognition.

### 3.2 Lexicon Match

This module takes two inputs—the recognition matrix (the “ $C$ ” or “ $D$ ” matrix defined above) corresponding to a unit (city, state or zip) and the appropriate lexicon. The task of this module is to assign a confidence value to each entry in the lexicon such that a higher value indicates a better match between the lexicon entry and the input unit.

We accomplish this by the following recursive process: tentatively assign a group of pieces to the first digit/character of the given lexicon entry; and recursively assign, in the best possible way, groups of pieces for each of the remaining digits/characters in the lexicon entry; then combine the confidences of the two assignments (“first” and “rest”); repeat this process for different tentative assignments of pieces to the first digit/character, and choose the best overall.

In the implementation of the matching algorithm, we use a *dynamic programming* approach to handle overlapping sub-problems. The sub-problems in our case are matching suffixes of the lexicon entry with a tail set of pieces. Since our recurrence relation does not guarantee optimal sub-structure, we can encounter situations where the best solution to the overall problem does not contain the best solution to a sub-problem. However, such occurrences are very rare. By repeating the matching process, we assign a confidence to every entry in the zip lexicon (or city/state lexicon, depending on the nature of the input unit).

See [3] for details on the dynamic programming formulation, on the use of correlation between the city, state, and zip entries in combining confidences, and on various ideas for the efficient implementation of the dynamic programming algorithm.

## 4 Experiments

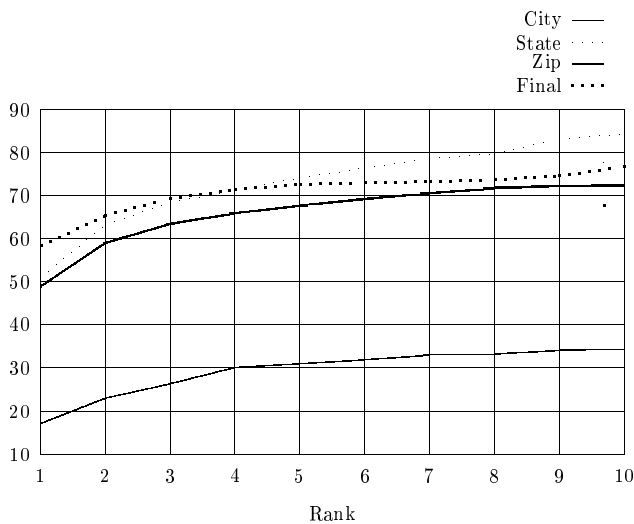
We tested our system on a set of 451 address images. The results are summarized in Table 1. In the table, “Conf” denotes the minimum value of overall confidence required for a triple to be considered a valid answer; the “Reject” column gives the percentage of address images for which no CSZ triple exceeded the threshold; the column labeled “Rank  $\leq k$ ” gives the percentage of address images for which the correct CSZ triple was ranked within the top  $k$  choices; finally, an address image is classified as an error if the correct CSZ triple was assigned a rank above 10.

From the table, we note that in 75% of the cases, we have been able to rank the 76,121-entry CSZ database such that the correct entry is assigned a rank of at most 10. We believe that this is an impressive performance, considering the fact that the process involves word separation, syntax matching, and very-large vocabulary word recognition, and a huge set of possibilities for the final answer. In addition to the high success rate, the system exhibits good thresholding ability. Thus if we accept only those addresses for which the final confi-

Conf	Reject	Rank				
		> 10	≤ 10	≤ 5	≤ 3	≤ 1
0.00	0.0	25.5	74.5	72.5	69.2	58.3
0.40	8.6	17.7	73.6	72.1	69.0	58.1
0.45	20.6	10.6	68.7	67.6	65.2	56.1
0.50	37.5	4.2	58.3	57.9	56.5	50.6
0.51	41.9	3.3	54.8	54.3	53.2	47.9
0.52	46.6	2.0	51.4	51.2	50.3	45.5
0.53	52.1	1.3	46.6	46.3	45.7	41.7
0.54	56.3	0.9	42.8	42.6	41.9	38.6
0.55	58.5	0.4	41.0	40.8	40.1	37.3
0.56	63.6	0.2	36.1	35.9	35.5	32.8
0.58	71.4	0.0	28.6	28.4	28.2	26.6

**Table 1. Experimental results**

dence is high, we can obtain nearly zero error in precisely locating, recognizing and interpreting the city, state and zip fields in the given address.



**Figure 3. Individual and Final Recognition Results (Top 10)**

In Figure 3, we contrast the individual recognition results for the three fields and the final recognition results for the triples in the CSZ database. More precisely, we show the percentage of addresses for which the correct city/state/zip was assigned a rank  $1 \leq 10$ , and the percentage of addresses for which the correct CSZ triple was assigned a rank  $1 \leq 10$ . The graph illustrates that the final results are higher than the individual results for the zip and city, indicating that the use of correlation helps to overcome the errors in the individual recognition results for the zipcode and city fields. Note that although the results for state recognition are higher than the final result (since state recognition is an easier problem), we

still need the results of zipcode and city recognition to obtain the correct CSZ triple associated with a given address.

The current speed of our system is around 2s on a 300Mhz Sun UltraSparc2 machine. Within this time, our system performs line and word separation, syntax matching and multiple word recognitions.

For a more detailed analysis of our experiments, including finer evaluations of the parsing and recognition modules and error analyses, see [3].

## 5 Conclusions

In this paper, we have presented a computer vision system to locate, parse, recognize and interpret a set of correlated entities. The specific example that we have considered is the parsing of city, state and zip words in handwritten addresses. This problem was hitherto unconsidered for the following reasons: it is difficult, time-consuming, error-prone and generally regarded as superfluous to the task of address interpretation. We have presented the design of an efficient system for this difficult problem of parsing and recognizing city, state and zip fields. Furthermore, we have shown that the inclusion of city and state information plays an important role in reducing the errors in zipcode recognition and in the overall task of address interpretation. We believe that our work has paved the way for the creation of a new address interpretation system in which equal importance is given to the parsing and recognition of the city, state and zip fields.

## References

- [1] J. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.
- [2] D. Lee and S. N. Srihari. Handprinted digit recognition: A comparison of algorithms. *IWFHR III*, pp. 153–162, 1993.
- [3] U. Mahadevan. Robust Vision Through Imperfect Eyes: A System for Processing City, State and ZIPCodes in Handwritten Addresses. PhD thesis, SUNY at Buffalo, January 1998.
- [4] U. Mahadevan and R. C. Nagabushnam. Gap metrics for word separation in handwritten lines. *3rd ICDAR*, pp. 124–127, 1995.
- [5] U. Mahadevan and S. N. Srihari. Hypotheses generation for word separation in handwritten lines. *IWFHR VI*, pp. 453–456, 1996.
- [6] G. Seni and E. Cohen. External word segmentation of off-line handwritten text lines. *Pattern Recognition*, 27(1):41–52, 1994.
- [7] S. Seshadri and D. Sivakumar. A technique for segmenting handwritten digits. *IWFHR III*, pp. 443–448, 1993.
- [8] M. Shridhar and A. Badreldin. Context-directed segmentation algorithm for handwritten numeral strings. *Image and Vision Computing*, 5(1):3–9, 1987.
- [9] G. Srikantan, S. W. Lam, and S. N. Srihari. Gradient based contour encoding for character recognition. *PAMI*, 29(7):1147–1160, 1996.