ADDRESS BLOCK SEGMENTATION USING ENSEMBLE-CLUSTERING TECHNIQUES

by

Mustafa Idrissi

Master Thesis

Media & Knowledge Engineering

Faculty of Electrical Engineering,

Mathematics and Computer Science

Delft, February 2008

TU Delft
Technische Universiteit Delft

Prime Vision
We bring sense to vision

Preface

This thesis is result for my Master graduation at Media & Knowledge Engineering group Faculty of Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology. The research presented in this thesis was performed at Prime Vision Company, Delft during the time 01 March – 31 December 2007. Prime Vision is a Dutch company, which specializes in automated OCR (Optical Character Recognition) for both postal and financial institutions. Within the Research and Development team at Prime Vision, I was asked to investigate some traditional clustering techniques as well as ensemble techniques for the address-block segmentation problem.

Acknowledgements

I want to take this opportunity to express my gratitude to all the people for their contribution and support they give me to accomplish this thesis. First I would like to thank all the people at Prime Vision, for contributing to my very pleasant stay at Prime Vision. There are too many names to write here but I can assure you, I will not forget any one of you! The work presented in this thesis is my own, but I could not have accomplished this without the help of others. Therefore I want to express my gratitude to the following people for their contribution and support:

- My supervisor, Wouter Homan, at Prime Vision for helping me with both small and big problems, for his suggestions, and for the weekly discussions that were both enjoyable and useful.
- My supervisor at TuDelft university, Leaon Rothkrantz, for his time and useful suggestions on my work.
- Eddy Thans and Michiel de Rijcke, for giving me the opportunity to do my thesis at Prime Vision.
- My co-supervisor Wim de Waard, for his help, suggestions, and for taking his last days at Prime Vision to read my report and provide me with useful feedback.
- And not to forget my room mates at Prime Vision, Hein van de Ploeg and Sergey Verzakov, for their support and beautiful time.
- And finally, to my family, for supporting me financially as well as mentally, and for always being there for me.

Merci beaucoup,

Mustafa Idrissi, December 2007

Abstract

Address-block segmentation using ensemble-clustering techniques
Mustafa Idrissi (1029894)
Delft, November 2007

Media & Knowledge Engineering
Faculty of Electrical Engineering, Mathematics,
and Computer Science
Delft University of Technolog
Mekelweg 4, 2628 CD Delft, The Netherlands

Graduation committee:
Dr. Drs. L.J.M. Rothkrantz
Dr. Ir. C.A.P.G. van der Mast
Ir. H.J.A.M. Geers
Ir. Wouter Homan

The large volume of mail pieces that must be handled every day, the intensive process of mail handling, and the increase of the manual processing costs have made automatic sorting systems very important and economically attractive. One of the challenges to make the postal automation technology more accurate is to improve the accuracy of the address-block locating task. This task can be decomposed into: address-block segmentation, which involves segmenting the mail piece into different regions and address-block selection, which involves selecting the segmented region that satisfies the optimal destination address-block among all the segmented regions. In this thesis we focused on the segmentation part of the address-block locating task. We investigated whether some clustering techniques and ensemble clustering techniques will help to improve the accuracy of address-block locating. For this a workbench with several clustering methods as well as ensemble methods was developed. Very little prior knowledge of the images is required. The implemented system has been evaluated on mail piece images captured live from real postal pieces at the postal sorting centers. The results of this approach will be described and illustrated with tests carried out on different images (parcels, magazines, postcard, etc.) where there are no fixed position for the address-block, postmarks and stamps. A ground-truth strategy is employed to evaluate the accuracy of segmentation. The agglomerative based clustering and their ensemble version achieved the best clustering results. However they suffer from high run-time. Therefore further development will be needed to achieve better computation time.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The Dutch postal service processes approximately 17 million postal pieces every day [20, 21]. Due to this large volume of mail pieces that must be handled every day, intensive process of mail handling, and the increase of manual processing costs have made automatic sorting systems very important and economic attractive. Such postal automation is a machine vision system, which scans each mail piece in turn, followed by address-block locating, recognition, interpretation, and finally determining the destination address of that mail piece. Figure 1 shows an image of a postal sorting centre.

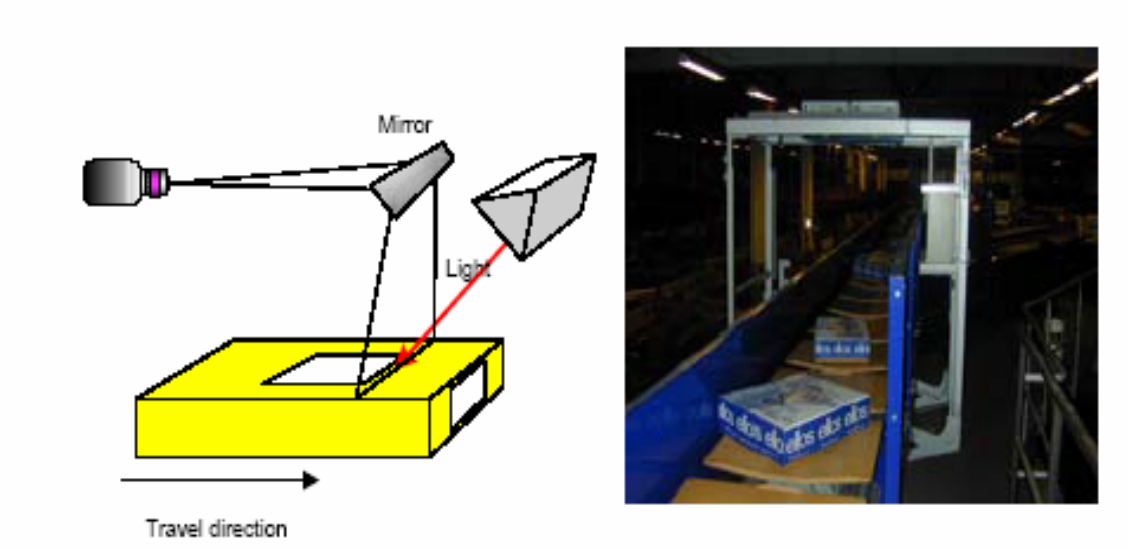


Figure 1-1: Image of the sorting center.

The availability of sophisticated optical character recognition (OCR) systems for recognizing both handwritten and machine printed characters with high accuracy makes it possible to explore full address information on mail pieces, which improves the efficiency of an automated mail sorting machine. The overall process of the postal automation system can mainly be decomposed into four main sub-tasks [16, 18, Prime Vision system]:

1. Image acquisition;
2. Address-block locating;
3. Line and word separating; and
4. Address parsing and recognition.

The main task of the image acquisition module is to obtain a digitized image of the mail piece. The obtained image is grey-scaled if necessary. This may be followed by some pre-processing techniques like noise removal to improve the image quality, or reducing the resolution of the image to decrease the computational requirements. The pre-processed image is then converted to a binary image. Connected components are then generated using this binary image together with one of the existing connected component labeling. The next step is to segment the obtained connected components of the mail piece image into different clusters, which satisfy some pre-determined criteria (like overlap, distance between components, etc). Each of these clusters can contain one or more components. Next, the obtained clusters are classified as text-blocks, image/graphic-blocks, and noise-blocks according to pre-determined criteria (such as pixel-density, histogram analysis, different size of related features, etc…). All the other blocks are ignored and only the text-blocks are used for further processing. By applying some confidence assignment function on these text-blocks, the text-block with the highest confidence is selected as the Destination Address-Block (DAB). After locating the destination address, the address-block is split into its text lines and words. These lines and words are then sent to the OCR for the further processing. Finally the textual information of the destination address is obtained, interpreted and its elements are labeled such as street name, street number, ZIP-code, and city name using some knowledge about address-block syntax. A flow chart of this overall process is shown in figure 1-2.

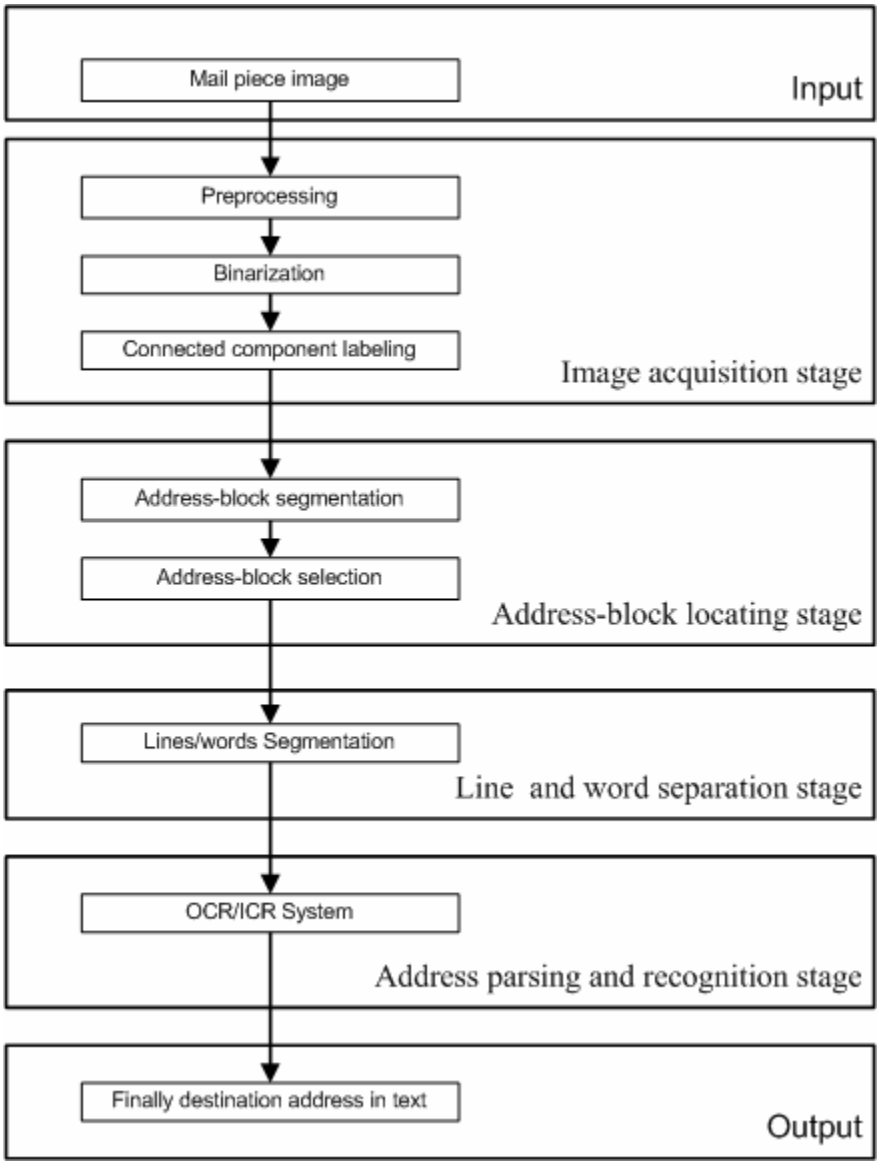


Figure 1-2: Flow chart of the overall process.

The aim of this thesis is to find out whether or not some clustering techniques will help to improve the segmentation task of the address-block locating stage. For this, we will mainly concentrate on the address-block segmentation stage.

1.1 RESEARCH CONTEXT

One of the challenges to make the postal automation technology more accurate is to improve the accuracy of automatic determination of the position and orientation of the destination address-block in the image of a mail piece such as a letter, magazine, or parcel [7]. This will allow

larger volumes of mail pieces to be processed more accurately and quickly, which will lead to an economic benefit. Intuitively this means improving the address-block locating sub-task of the postal automation. This sub-task can further be divided into two main tasks:

- **Clustering or segmentation task:** This involves dividing the image into several regions (like address-block region, stamp region, etc.) such that each region corresponds to a unique block. There are numerous ways to segment an image into regions. One method that is frequently used in document image segmentation is clustering, or also called unsupervised classification. The general goal of clustering is to group a given set of data into clusters such that objects within a cluster have high similarity in comparison to one anther.
- **Selecting or classification:** Address-block selecting involves selecting the cluster that satisfies the optimal destination address block among all the clusters. Due to the absence of cluster labels (unsupervised situation) this problem is much harder than the situation where clusters are labeled (supervised situation). The common strategy of most approaches uses some confidence assignment techniques based on some prior-knowledge (e.g., position and orientation of the blocks, the writing style of the blocks, etc.).

It is well known that standardized mail pieces allow the automation of postal sorting in general and especially address-block locating to be processed quickly and more efficiently. However, the physical attributes of mail pieces vary greatly. Different mail pieces have different sizes, layouts, colors, texture, and often no fixed item's position. To complicate the problem further, the address-block may be handwritten or machine printed, and the address-block may appear mixed up with postmarks or stamps [12]. These factors affect the address-block locating task and make the address-block locating process sensitive for errors. The errors that may occur during the address-block locating stage can be divided into: (i) segmentation errors; (ii) selection errors; (iii) combination of segmentation and selection errors. Ranganath and Chipman [15] classified several types of errors associated with segmentation as follows:

- **Missing objects:** For the problem of address-block segmentation this error type is not directly associated with the segmentation task, but rather with the preprocessing

task. However, this results in one or more objects having been discarded during the preprocessing task (e.g. because of the chosen threshold some text objects may became invisible). Because of this some regions may be affected with missing objects (see figure 1-3a).

- **False objects:** This error type is also associated with the preprocessing task rather than with segmentation task. Generally, this involves extraneous marks (noise) or shadows that may be segmented as regions, which do not correspond to any scene. However, these extraneous objects that normally are not a part of the original image may affect the segmentation task. An example of this is shown in figure 1-3b.
- **Fragmented objects (over-segmentation):** This type of error is directly associated with address-block segmentation task and results in a single object being segmented into more than one region. Intuitively, this means that part of address-block is missing (like the zip-code or house number), because it has been segmented as a different region. This is shown in figure 1-3c.
- **Merged objects (under-segmentation):** This error type has also directly to do with address-block segmentation and means that two or more objects may be combined into one region. Consequently, this means that an object is added to the address-block, which should not be a part of that region. See figure 1-3d for an example.

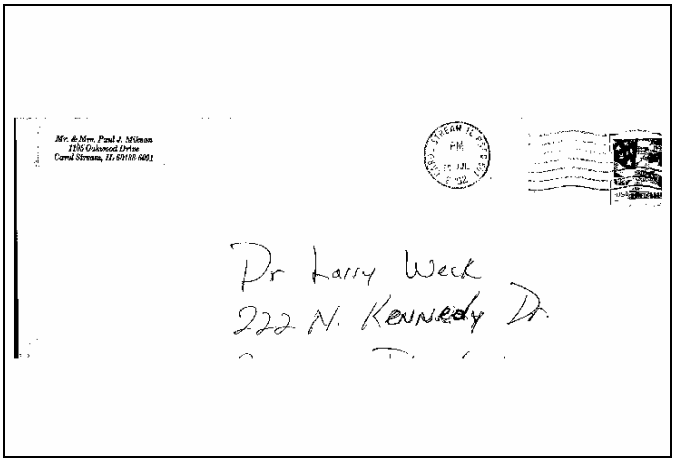


Figure 1-3a: Missing objects.

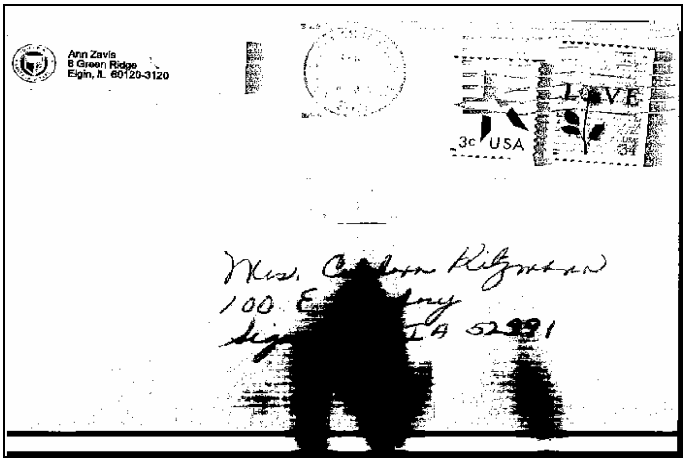


Figure 1-3b: False objects.

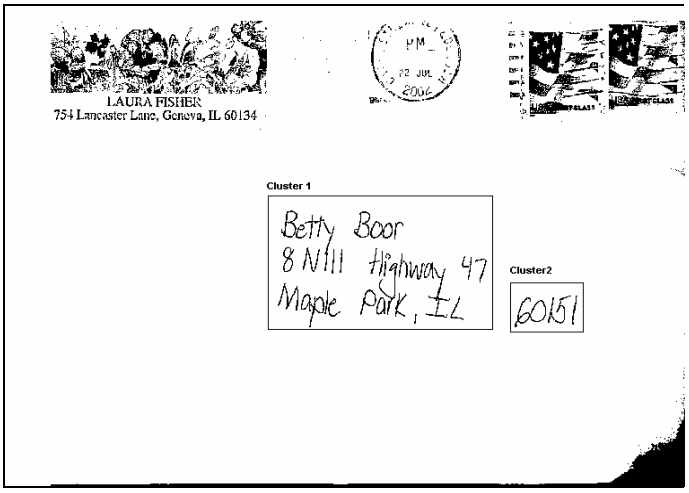


Figure 1-3c: Over-segmentation.

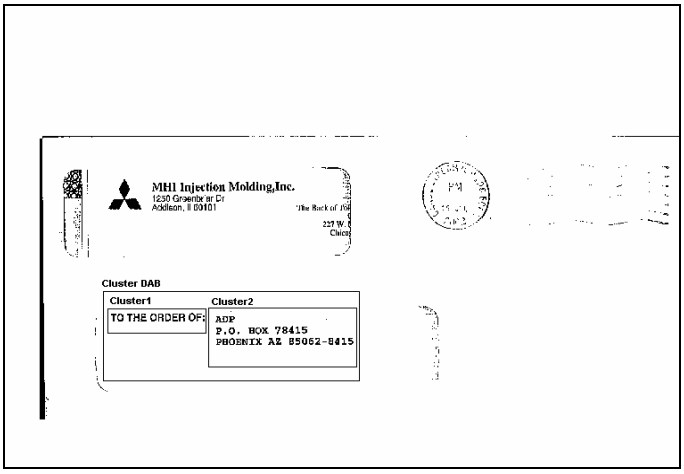


Figure 1-3b: Under-segmentation.

The third error is in terms of address-block locating fatal, where the last one is not. This is because the extra information in the DAB may be removed during the address interpretation. On the other hand information lost in the DAB may lead in failed address recognition, certainly if the core elements like house number and zip code are missing [5]. Beside these segmentation errors the selection stage is also not free from errors. There are some cases where good segmentation may still cause the selection stage to produce wrong destination address block. This is in most cases a consequence of the following:

- Destination Address block may be scraped, which may cause the selection process to select the return address instead of destination address. This means that the mail piece will go back to the sender.
- Some words may be added to the return address to indicate that this is now the destination address. For a human being this is easy to see, but for a postal automation this is a hard problem.

The current segmentation system suffers from all the above errors. Accordingly, the error level varies widely with the difficulty of the mail piece. For example, parcels suffer more from

errors than letters. This is because parcels contain in comparison to envelopes more text, graphics and logos.

An interesting research is to investigate whether some clustering techniques will lead to more accurate results (destination address) for the domain of address-block locating by overcoming, or at least decreasing the number of segmentation errors that may occur during the segmentation stage of the address-block location process. Since the selection errors are hard to overcome, because of the variety that they occur in, our focus for this thesis will be on the address-block segmentation part.

1.2 RESEARCH QUESTION

Clustering techniques have been used in many areas, including artificial intelligence, data mining, image processing, marketing, medicine, biology, pattern recognition, and so on. In medicine, clustering is used, for example, to automatically build taxonomy of diseases, symptoms of diseases, or cures for diseases. Several clustering algorithms are used to obtain segment labels for each pixel of an image. Another application of clustering is to better understand gene functions in the biological processes in a cell. The main idea of this analysis of gene expression data is to detect groups of genes that have similar expression patterns. Another growing application area is customer relationship management, where data collected from different sources, like web surfing, cash register transactions, and call center activities is used to obtain customer behavior that can help to optimize marketing, bundling, and pricing strategies. Some companies such as banks, insurance companies, etc. may also apply clustering techniques in order to detect groups of high-risk customers. Clustering techniques have also been used in document image segmentation, in order to cluster document objects into blocks (e.g. text, image, table, etc…).

However, one major issue using clustering techniques is deciding how to choose the algorithm that will cluster the objects quick and accurately using some similarity measure. Different algorithms may give different results. Moreover, the same algorithm may give different results for different datasets, different domains, or different algorithm's parameter settings. Therefore finding the best clustering algorithm for a certain problem is not an easy job. An alternative approach demands efficient methods that would benefit from combining the strengths of many individual

clustering results in order to improve the clustering accuracy. This is the focus of research on clustering ensembles, seeking a combination of multiple clustering techniques that provides improved overall clustering of the given data. Cluster ensembles are beneficial in several aspects. They can lead to better average performance across different domains and data sets. Combining multiple clustering results can lead to a solution unattainable by any single clustering algorithm. These ensemble techniques have also been used for different problem domains, and they have showed good results.

The primary goal is to investigate whether some clustering techniques will efficiently cluster the connected components of a given mail piece and lead to more accurate clusters. This can be modeled as shown in figure 1-4. First, the mail piece's image is pre-processed and the connected components are defined using some connected components techniques. In this thesis we assume that the pre-processing and connected components extraction is already done by existing techniques. The connected components are then presented to the single clustering algorithm as well as to the ensemble-clustering algorithm. This makes the single clustering algorithm come up with some clustering results. The ensemble-clustering algorithm will first come up with different base clustering (a set of generated clusters). Using some criteria those base clusters are integrated to more suitable clusters. Finally, to evaluate which algorithm best fitted to the problem of address-block locating the results of both algorithms are compared to each other and to the truth clustering.

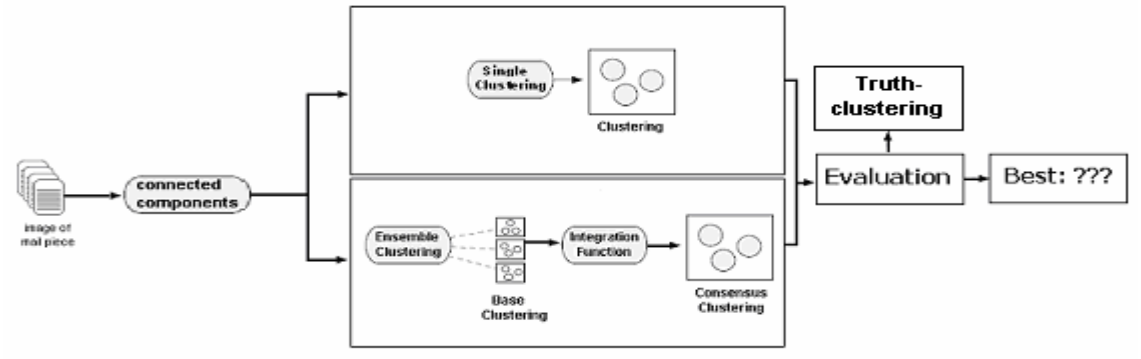


Figure 1-4: Flow chart of the research approach.

Having this model and the previously problem motivation in mind, the research question for this thesis can be defined as follows:

"Can some single or ensemble clustering technique help to make the clustering process of the address-block locating more efficient and accurate?"

1.3 GOALS

To give an answer on the research question several issues have to be investigated and implemented. The goal of thesis will be separated into research objectives and implementation objectives. The enumeration of *research objectives* can be defined as follows:

- Examine the existing clustering techniques in general and especially the ensemble clustering techniques. The result of this research will be presented in the research assignment report.
- Given a set of several connected components (character-blocks, image, graphic and other objects that may appear on a mail piece) examine the results of address block locating using the single cluster technique, which will be chosen based on the research assignment investigation.
- Given a set of several connected components (character-blocks, image, graphic and other objects that may appear on a mail piece) find out whether our assumption that some ensemble cluster technique (e.g. combination of different parameters, different feature, etc…) will lead to more accurate results in comparison to a single cluster technique is justified. Likewise the ensemble techniques will be chosen based on the research assignment investigation.
- One general question among clustering techniques is whether the underlying assumptions (obtained clusters shapes, number of clusters, etc.) of a cluster algorithm are satisfied at all for the considered data set. There are some validation techniques discussed in the literature including external, internal and relative measures. The question for this thesis is to investigate which of the validation technique is best applicable for measuring the quality of the clustering. In case

where none of the existing validation techniques works some predetermined validation criteria will be used to measure the quality of the clustering.

- Find a function that will improve the selection stage of the address block locating problem. This question is optional and will only be examined if permitted by time.

Consequently, the enumeration of *implementation objectives* can be defined as follows:

- Since the current system was built in Delphi, a workbench with the chosen single clustering algorithm for address-block segmentation will also be implemented in Delphi.
- In order to test ensemble-clustering technique on the address block segmentation problem the chosen ensemble techniques will also be implemented in Delphi and included in the workbench.
- Finally, some evaluation technique will be implemented to evaluate the proposed techniques. Within evaluation the results of both techniques will be compared with each other and with the current system.

1.4 STRUCTURE OF THE THESIS

The remainder of this thesis is structured as follows: Chapter 2, provides a literature review of related work on address-block locating. The background theory on clustering techniques and ensemble techniques will be discussed in chapter 3. Chapter 4 will describe the proposed model. This will be followed by the description of the model implementation on chapter 5. Experiment results will be given in chapter 6 and followed by discussion on chapter 7. Finally conclusion and suggestion for future research is given in chapter 8.

Chapter 2: Related work on address-block segmentation

Several authors have dealt with the problem of locating address-blocks in images of mail piece. The existing methods are based on artificial intelligence techniques [1, 2, and 3], on geometrical characteristics [4, 5, 6, and 7] or on texture analysis by filtering [8, 9, 10, 11, 12, 13, and 14]. This chapter contains a brief summary of an approach reported in the literature of these classes.

2.1 ARTIFICIAL INTELLIGENCE APPROACH

Wang et al [2,3] developed an expert system for automatic sorting of mail pieces. This system is capable for multi-spectral images, i.e., gray scale, color, infrared, or color under ultraviolet illumination. The address-blocks in this system are located by: *"the address block locating subsystem"* (ABLS). The ABLS consists of six major components:

- **Mail Statistical Database (MSD):** The MSD contains the statistical information of the geometric features of all meaningful information-blocks on various mail pieces. Specially, this includes the probability that a destination address block and a return address block are in a certain location on the image, the average and standard deviation of the aspect ration, number of text lines, and the address block length of a typical hand and machine generated destination address block. This database has been generated from a careful analysis of a large number of mail pieces.
- **Tool box:** The toolbox contains some image processing tools for both colored and gray level images, such as adaptive thresholding to convert such images into a binary image, bottom-up segmenter for machine-generated text as well as hand-generated text, in order to group characters into words, lines and blocks. It contains also regularity discrimination that discriminated between machine printing and hand-writing, icon detector to detect the rectangular postal icon, handwriting detection that detects the likely regions of hand generated text, shape analyzer to measure the degree of rectangularity of a region, texture discriminator for distinguishing address blocks from

miscellaneous text blocks, and ultra-violet detection to detect postage indicia under UV light.

- **Rule-based inference engine:** This component acts as an interpreter of all the rules associated with each tool and performs forward or backward reasoning on different rule modules.
- **Control mechanism:** The control mechanism is responsible for checking the termination condition, selecting an appropriate tool from the toolbox, combining new evidence, and updating the context. The tool selection is based on the benefit/cost estimation of each tool in the toolbox. The utility of each tool in the current context is estimated and the one with the maximum utility is selected. The selected utility is applied next and its utility is reset to zero before it is applied.
- **Control data:** The control data used by the control system to provide information about interdependencies between tools and criteria for accepting a block as the destination address block.
- **Blackboard:** The blackboard component contains the geometric attributes of blocks obtained from low-level image processing of different type of images, the current context, and the confidence levels for the labeling hypothesis. The blackboard module divides a complex problem into individual subtasks, and a specialized tool treats each subtask. The ultimate goal is to integrate knowledge from different sources to achieve a common goal.

The ABLS may use one or more of the various image-processing tools to process the input image of the mail pieces. The evidence achieved from these tools is integrated using a blackboard system. Each piece of evidence is accompanied by a confidence value, which is a measure of the degree to which the evidence supports the hypothesis that a particular block is the destination address block (DAB). Based on the available evidence, a score is assigned to each candidate DAB. Finally, the candidate with the highest score is selected as the DAB. In order to evaluate the

performance of ABLS, experiments were conducted using a database containing of 174 complex mail pieces images. These experiments showed that 81% were classified as a success, i.e., the DAB has the highest rank. In 4% of the cases were partial success, i.e., the DAB was identified but the DAB contains insufficient information to correctly sort the mail pieces. The ABLS could not recommend any block in 5% of the cases and in the remaining 10% the wrong candidate was identified as DAB. The average CPU time was 10.4 minutes on a Sun-3 system.

2.2 GEOMETRIC FEATURE APPROACH

The algorithm presented in [6] was developed for parcel image sets, which contain images with well-separated address blocks as well as non-separated/ incorrectly separated address block in different orientations. The process flow of the algorithm can be composed in the following main steps:

- Extract contours of connected components.
- Extract contour features, where each contour corresponds to a point in a feature space.
- Cluster points in the feature space.
- Using heuristic rules, find clusters corresponding to the address block.
- Separate contours of the chosen cluster together with any close clusters.

For each contour the following features are considered:

- $mean\_x = (right + left)/2$.
- $mean\_y = (bottom + top)/2$.
- $size = \max((right - left), (bottom - top))$.
- $avg\_stroke\_width = \dfrac{Area\_inside\_contour}{0.5 * Perimeter}$.

  Using the assumption that the whole address is written with the same pen or is printed with the same font, this feature is very useful in separating address block text from the rest of the image.

- $\max\_stroke\_length = \text{largest interval traced in one of } 8 \text{ directions.}$

  This feature reflects the length of the stroke. The interval traced in some particular

direction would include a sequence of pixels having transitions with the direction as well as with two neighboring direction.

In order to cluster points in the feature space a logarithmic distance between two contours in the feature space as sum of the following feature distances has been used:

- $\frac{|mean\_x_1 - mean\_x_2|}{\min(size_1, size2)}$.
- $\frac{|mean\_y_1 - mean\_y_2|}{\min(size_1, size2)}$.
- $\log\left(\frac{\max(size_1, size_2)}{\min(size_1, size_2)}\right)$.
- $\log\left(\frac{\max(avg\_stroke\_width_1, avg\_stroke\_width_2)}{\min(avg\_stroke\_width_1, avg\_stroke\_width_2)}\right)$.
- $\log\left(\frac{\max(avg\_stroke\_length_1, avg\_stroke\_length_2)}{\min(avg\_stroke\_length_1, avg\_stroke\_length_2)}\right)$.

To group similar features into the same cluster several clustering methods have been tested, such as k-means algorithm. The authors conclude that such methods tend to incorrectly remove few contours from address block, or to add some single unrelated contours to the address block cluster. Due to success achieved with agglomerative clustering algorithms type, an algorithm of this type was used. A threshold θ was defined. Next, the clustering results need to satisfy the following two conditions:

- If the distance between two contours is less than θ: $dist(c_i, c_j) \le \theta$ then c_i and c_j are in the same cluster.
- The distance between any two clusters Cl_k and Cl_l is bigger than θ, that is for any $c_i \in Cl_k$ and $c_j \in Cl_l$: $dist(c_i, c_j) > \theta$.

Using this condition an algorithm was implemented to add contours to already existing clusters, form a new cluster, and if necessary merge clusters. The algorithm takes $O(n^2)$ time, where n is the number of contours. In order to decide which cluster is the most probable destination address-block cluster the authors used the following heuristic rules:

- Address block cluster should have at least 10 contours.

- There should be contours of size bigger than some threshold in the cluster.
- Choose the cluster satisfying 1 and 2 with the largest area as a candidate address block cluster.

To test their proposed algorithm the authors used 1684 images. Approximately 1/3 of these images did not contain a full destination address block, another 1/3 contained well-separated destination address blocks with possible noise. In order to evaluate the total performance a HWAI[1] system developed in CEDAR with and without address block-locating algorithm was used. The HWAI experiment without address block was able to identify 240 images. This number was increased to 270 in the HWAI experiment with address block locating. The increase in performance is due to successfully separated address block in some images, and removed noise in other images. There were about 10 images that were recognized in the first experiment, but not in the second experiment. This was due to information loss (mostly zip code, which sometimes stands far away from other parts of the address block) during the address block locating.

2.3 TEXTURE ANALYSIS FEATURE APPROACH

An address-block segmentation approach[10] based on evidence generation by lacunarity associated with a region-growing algorithm was introduced by Facon. and Menoti. Lacunarity is used as evidence to distinguish between background and objects. The block diagram for this segmentation approach can be divided in the following four main steps:

- **Feature extraction:** This is performed on an input image I_{IN} by means of lacunarity generation feature image I_{FE}. Lacunarity is a multi-scale measure describing the distribution of gaps within a texture: the greater the range in gap size distribution, the more lacunar the data. To measure lacunarity, Allain's and Cloitre's algorithm *'gliding box'* was used. The gliding-box samples an image using overlapping square windows of length r. Lacunarity is then defined in terms of the local first and second moments, measured for each neighborhood size for every pixel in the image. The following

[1] Handwritten Address Interpretation. For more information see: http://www.cedar.buffalo.edu/hwai/rcr/RCRhwai.html)

equation was used in order to extract I_{FE} from I_{IN} :

$L(r) = 1 + \frac{\text{var}(r)}{mean^2(r)}$.

- **Feature Normalization:** New features I_{FN} are devised by non-linear normalization from I_{FE}, in order to enhance extracted features and singularities (discontinuities) between background and objects. This normalization is needed because of the variations of hand written objects sizes; background illumination in the image and the distribution in I_{FE} are very sparse and non-uniform. This directly affects lacunarity distribution. In order to take this in to account the following non-linear normalization was used:

  $I_{FN} = \arctan\left(\frac{I_{FE}}{(k * std(I_{FE}))}\right)$,

  where $std(I_{FE})$ is the standard deviation of I_{FE} and *k* is a multiplication factor.

- **Saliency identification:** A saliency map of a given image describes the degree of saliency importance of each position in the image, which is a particular quantification of the desirable smoothness and length properties, and relies on curvature estimates and gaps. For this the authors performed a Otsu's thresholding algorithm on the I_{FN} to produce I_{SI}, which contains enough evidence for the segmentation of objects.

- **Region growing:** This is applied on the evidence in I_{SI} in order to recover all remaining pixels belonging to the segment, the object of interest, yielding the final segmentation I_{OUT}. Thus, each point in I_{SI} will be selected if the gray value i_v falls inside $\lambda\%$ of the image distribution (Gaussian). This is given as follows: $i_v \le I_\mu - Z_{50\%-\lambda} * I_\sigma$, where, I_μ and I_σ are the global mean and the global standard deviation of the image respectively, $Z_{50\%-\lambda}$ is the normalized point for probability of $50\% - \lambda$. In fact, the objects to recover are the $\lambda\%$ (in Gaussian distribution) darker ones. After verifying all points i_v was indicated through I_{SI}, only ones that hold the above equation will be stored. If the selected point holds $i_v \le i_g$ (where i_g is the greatest grey value of each initial saliency so far), its neighbor point will be en-queued if they hold $i_v \le I_\mu - Z_{50\%-\lambda} * I_\sigma$. This process will be terminated when there are no more points in the queue.

In order to evaluate their proposed segmentation method a database of 200 complex postal envelope images, with no fixed position for the handwritten address blocks, postmarks and stamps was used. All the images are digitized at 200 dpi, gray-scaled, and approximately 1500x2200 pixels. Also a ground-truth segmentation regarding each class (handwritten address block, postmarks and stamps) has been generated for each envelope image. A segmented score was computed by comparing the original pixels at the same location in the ground-truth images with the segmented ones. The best results are achieved with $r= 3$ $k=2$ and λ =10% (and $Z_{50\%-\lambda}$ = 1.28) see Table 2-1.

Table 2-1. The best results for with $r= 3$ $k=2$ and λ =10% (and $Z_{50\%-\lambda}$ = 1.28).

Object	**Accuracy pixel by pixel** $(\mu \pm \sigma)$
Address block	97.52% ± 5.72%
Stamp	31.94% ± 15.10%
Postmarks	88.07% ± 16.79%
Noise	0.51% ± 0.75%

The experiments with different results for r conclude that by increasing r the address block accuracy decreases, the noise increases, and time complexity increases since it is $O(r^2 n)$ in lacunarity feature extraction. The influence of the multiplicative factor of non-linear normalization was also tested, and this showed that feature extraction and proposed nonlinear normalization based on standard deviation are robust. Finally, the influence of parameter λ has been analyzed. By observing the results regarding accuracy for address block and noise shows that increasing/decreasing λ increases/decreases both the address block and noise accuracy. Considering 0.51% a good noise rate on average, λ = 10%.

Chapter 3: Theory on single and ensemble clustering

A strict definition of learning is hard to find although most sources use the definition of Nardendra and Thathachar, namely: *"Learning is defined as any relatively permanent change in behavior resulting from past experience, and a learning system is characterized by its ability to improve its behavior with time, in some sense towards an ultimate goal."*

The field of machine learning strives to gain the capability to solve problems by learning. Broadly speaking machine learning can be divided into three main classes:

- **Supervised learning:** This kind of learning is called supervised because there is a teacher who shows the system the correct association between input and output. In supervised learning each example is a sample of input-output patterns. The description of data is a function of the input being able to approximate the output with a good compromise between generalization and complexity.
- **Reinforcement learning:** In reinforcement learning there is a teacher, but in contrast to supervised learning this teacher does not give the desired response. Rather a scalar reward or punishment according to the quality of the generated output. In this case, each experience is a triplet of stimulus, response and corresponding reinforcement.
- **Unsupervised learning:** This kind of learning is called unsupervised in the sense that the learner must find a description or structures inside data without any other external information. In contrast to supervised learning in unsupervised learning there is no teacher. Unsupervised learning can be thought of as finding some patterns in the input data.

In the category of unsupervised learning, one of the primary tools is clustering. An alternative approach of clustering demands efficient methods that would benefit from combining the strengths of many individual clustering results in order to improve the clustering accuracy. This is the focus of research on clustering ensembles, seeking a combination of multiple clustering techniques that provides improved overall clustering of the given data. In order to distinguish

between traditional clustering and ensemble clustering, single clustering may be used to refer to the traditional clustering.

This section will give a short description of each of these clustering types. First, we will describe the difference between classification and clustering. Next, we will indicate what clustering is, which will be followed by clustering components description. After that, some clustering issues mentioned in the literature will be summarized. Finally, ensemble clustering will briefly be discussed.

3.1 CLASSIFICATION VS. CLUSTERING

In order to clarify what clustering is, it is important to understand the difference between clustering (unsupervised classification) and classification (supervised classification). In supervised classification, the algorithm is provided with a collection of labeled (pre-classified) patterns. The goal is to classify a newly yet unlabeled pattern into the class whose description best fits the pattern. Typically, the given labeled (training) patterns are used to learn the descriptions of classes that in turn are used to label a new pattern.

Clustering is similar to classification in that data are grouped together. However, unlike classification, clustering analyzes data objects without consulting class labels or any training data. Generally class labels are generated using clustering. The objects are clustered or grouped based on some similarity or dissimilarity (existence of different types or features) criteria between data, such that clusters of object within the same cluster have high similarity in comparison to one another and are very dissimilar to objects in other clusters. Each cluster can than be viewed as a class of objects, from which rules can be derived.

3.2 WHAT IS CLUSTERING?

Clustering, or cluster analysis, is an important technique in the rapidly growing field known as exploratory data analysis and is being applied in a variety of engineering and science disciplines. Clustering aims to organize a collection of data items into clusters, such that items within the same cluster are more similar to each other than to items in the other clusters. This similarity can be expressed in very different ways, according to the purpose of the study, to domain-specific

assumptions and to prior knowledge of the problem. In contrast to classification, clustering is usually performed when no information is available concerning the membership of data items to predefined classes. Due to the absence of predefined classes, clustering is traditionally seen as part of unsupervised learning. Clustering algorithms are geared toward finding structure in the data, i.e. group data into clusters of similar objects.

A cluster is comprised of a number of similar objects collected or grouped together. Some cluster definitions are documented by Brian Everitt [43] as follows:

- A cluster is a set of entities, which are alike, and entities from different cluster are not alike.
- A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less that the distance between any point in the cluster and any point not in it.
- Cluster may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.

The last two definitions assume that the objects to be clustered are represented as points in the measurement space. While it is easy to give a functional definition of a cluster, it is very difficult to give an operational definition of a cluster. This is because the definition and the goals of clustering vary from case to case due to the fact that objects can be clustered or grouped into clusters with different purposes in mind. Therefore a large number of clustering definitions can be found in literature, from simple to complex. The simplest definition is shared among all and includes one fundamental concept: "*the grouping together of similar data items into clusters*". Another commonly used definition in literature is given by Jain and Dubes [24] as follows: *"Cluster analysis organizes data by abstracting underlying structure either as a grouping of individuals or as a hierarchy of groups. The representation can then be investigated to see if the data group according to preconceived ideas or to suggest new experiments"*.

An example of clustering is depicted in figure 3-1, where the input data are shown in figure 3-1a, and the desired clusters are shown in figure 3-1b. Hence, objects belong to the same cluster are given with the same label.

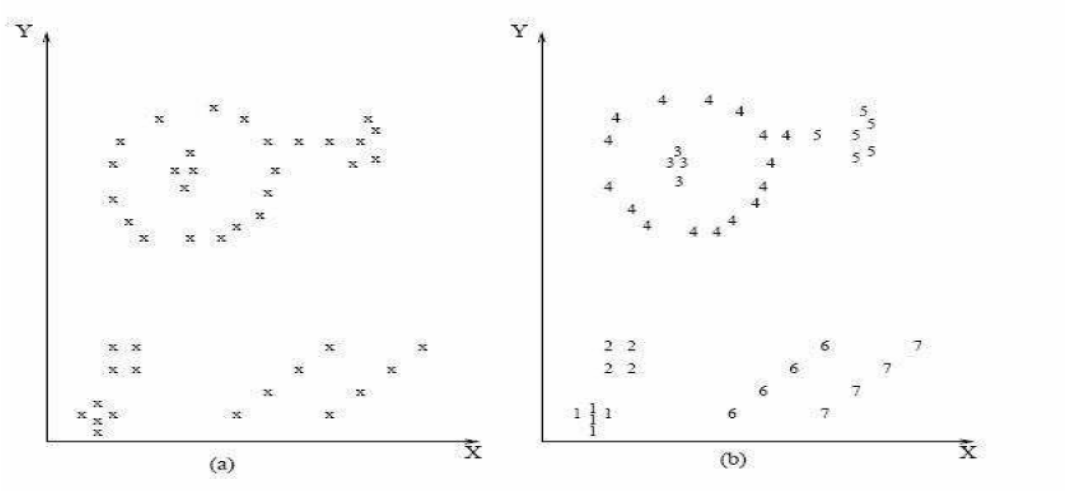


Figure 3-1: Clustering example.

3.3 CLUSTERING COMPONENTS

Jain and Dubes[27] divided clustering activities into the following 5 steps (see figure 3-2):

1. Representing the data objects by patterns;
2. Defining proximity measure between patterns;
3. Grouping objects according to the proximity measure;
4. Abstracting/validating the data; and
5. Interpreting the output.

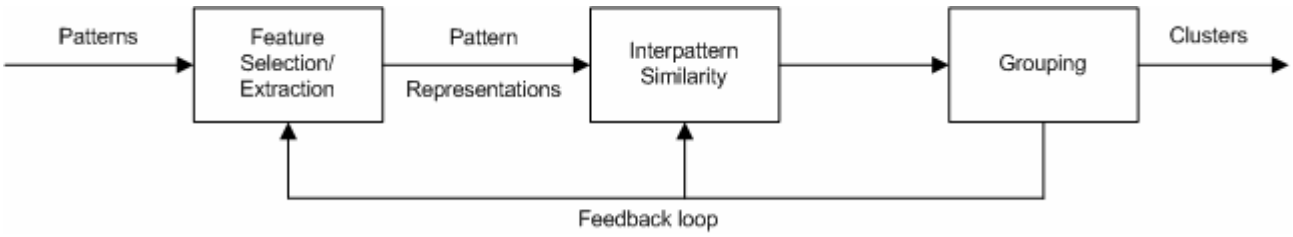


Figure 3-2: Components of a clustering task.

In this section, we describe each of these components. The concept of a feature and similarity measures will be described for different types of features. The two major categories of clustering algorithms, hierarchical and partitional, will also be introduced.

3.3.1 Pattern representation

Typically this consists of defining a measurement vector also referred to as feature vector, which will represent each object to be clustered. The set of features used to represent an object makes up the object's feature vector. Generally, these features may be of various types such as numerical, binary and categorical. For example, in a medical diagnosis system patients may be represented by features such as their age (numerical), gender (categorical), and smoking habits (binary). Suppose a set of *n* data points (objects), $[o_1, o_2, ..., o_n]$ which have to be clustered, and each object consists of *m* features (measurements), this feature vectors can then be represented in the following pattern matrix:

$$\begin{bmatrix} o_{11} & o_{12} & \cdots & o_{1m} \\ o_{21} & o_{22} & \cdots & o_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ o_{n1} & o_{n2} & \cdots & o_{nm} \end{bmatrix}$$

Where o_{ij} is the j^{th} feature of the object o_i , $1 \le i \le n$ *and* $1 \le j \le m$.

Some features are more useful than others. Therefore, feature selection may be used to select the most effective subset of the original features. Feature selection is the process of identifying the most effective subset of the original feature set to be use in clustering. Instead of directly using a feature, one can decide to normalize it first. This process of extracting more reduced features from a feature vector is called feature extraction. Feature extraction is the use of one or more transformations (e.g. feature normalization) to generate useful and novel features from original ones. Feature selection/extraction may optionally be performed during the pattern representation. Both typically lead to a better clustering, and makes the clustering algorithm more efficient in both time and space complexity.

3.3.2 Proximity measurements

A proximity measure measures the closeness between the feature vectors of objects. The proximity can either be measured using a similarity or dissimilarity (distance) measure depending

on the pattern representation. Similarity and distance are essentially synonymous and can usually be interchanged (i.e. the distance between elements measures their dissimilarity). Distance measure like Euclidean distance can often be used to measure dissimilarity between two patterns, whereas similarity measures can be used to characterize the conceptual similarity between patterns.

Given a proximity measure, one can convert the pattern matrix representing the feature vectors into an *n x n* matrix of the proximity between each object:

$$\begin{bmatrix} 0 & & & & \\ d_{21} & 0 & & & \\ d_{31} & d_{32} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d_{n1} & d_{n2} & \cdots & d_{nn-1} & 0 \end{bmatrix} \text{ or, } \begin{bmatrix} 1 & & & & \\ s_{21} & 1 & & & \\ s_{31} & s_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ s_{n1} & s_{n2} & \cdots & s_{nn-1} & 1 \end{bmatrix}$$

Where d_{ij} denotes the distance between the i^{th} and the j^{th} object and s_{ij} denotes the similarity between the i^{th} and the j^{th} object. Note that d_{ij} must satisfy the following condition:

1. $symmetry: d_{ij} = d_{ji}$;
2. $positivity: d_{ij} \geq 0, \forall i, j$;
3. $triangle\,inequality: d_{ij} \leq d_{im} + d_{mj}, \forall i, j, m$; and
4. $reflexivity: d_{ij} = 0 \; iff \; o_i = o_j$, *this is called a metric.*

Likewise, s_{ij} should satisfy the following condition:

1. $symmetry: s_{ij} = s_{ji}$;
2. $positivity: 0 \leq s_{ij} \leq 1, \forall i, j$;
3. $triangle\,inequality: s_{ij}\, s_{jm} \leq \lfloor s_{ij} + s_{jm} \rfloor s_{im}, \forall i, j, m$; and
4. $reflexivity: s_{ij} = 1 \; iff \; o_i = o_j$, *this is called a metric.*

Since proximity between feature vectors is fundamental to the definition of a cluster, the selection of a good proximity measure is critical for the success of a clustering algorithm. For this reason, there are numerous proximity measures available. The selection of a measure is partially driven by the types and scales of features to be clustered. The most well-known proximity measures for different types are discussed below.

Numerical features

The numerical features can be discrete or continuous. These kinds of features are typically measured using distance. Given two objects o_i and o_j represented by numerical feature vectors, the most common distance metrics are derived from the Minkowski[24,27]:

$$d_{ij} = \left(\sum_{k=1}^{m} \left| o_{ik} - o_{jk} \right|^{n} \right)^{\frac{1}{n}}.$$

A common distance metric, City-block distance can be obtained when n=1:

$$d_{ij} = \sum_{k=1}^{m} \left| o_{ik} - o_{jk} \right|.$$

Likewise the most common distance metric can be derived when n=2:

$$d_{ij} = \left(\sum_{k=1}^{m} \left| o_{ik} - o_{jk} \right|^{2} \right)^{\frac{1}{2}}.$$

Other distance metric will be discussed in chapter 4.

Binary features

A binary feature takes only two values. For this kind of feature, a similarity measure is the commonly used measure to determine the similarity between a pair of objects. However, dissimilarity measures can still be obtained by simply using $d_{ij} = 1 - s_{ij}$. The proximity indices between two objects o_i and o_j can be derived from the following contingence table:

Table 3-1. Contingence table.

	$o_j = 1$	$o_j = 0$
$o_i = 1$	n_{11}	n_{10}
$o_i = 0$	n_{01}	n_{00}

Where, n_{00} and n_{11} represent the number of simultaneous absence or presence of features in both objects, and n_{01} and n_{10} represent the presence of features in only one object. Given this, the most common matching coefficients between objects o_i and o_j are as follows:

- $S(i,j) = \dfrac{n_{11} + n_{00}}{n_{11} + n_{00} + \alpha(n_{10} + n_{01})}$, here, $\alpha = 1$ give the Simple matching coefficient, $\alpha = 2$ the Roger and Tanimoto measure and $\alpha = \dfrac{1}{2}$ the Gower and Legendre measure.
- $S(i,j) = \dfrac{n_{11}}{n_{11} + \alpha(n_{10} + n_{01})}$, here, $\alpha = 1$ provides us with the Jaccard coefficient, $\alpha = 2$ with the Sokal and Sneath measure and $\alpha = \dfrac{1}{2}$ with the Gower and Legendre measure. Notice that these measures focus only on the presence of features while ignoring the effect of feature's absence. This is mainly used in case where only the state described as '1' has a high weight in comparison with the other; i.e. the n_{11} matches are much more important than the n_{00} matches.

Nominal features

Nominal features are considered to be the weakest type, in that their numbers are simply used as names; the numbers themselves as well as the ordering of the numbers are meaningless. However, many features, especially features of data collected from human subjects are often of the nominal types. In contrast to binary features, nominal feature may have more then two states. For example, an image pixel may be described by one of the following color: {black, red, blue, green, brown, etc…}. There are two ways to determine the matching coefficient for these kinds of types. One way is to map the nominal features into a set of binary features and then apply one of the binary's matching coefficients discussed above. Another more effective way is by utilizing the following matching criterion[25]:

$$s_{ij} = \frac{1}{m}\sum_{k=1}^{m} s_{ijk} \text{ , where } s_{ijk} = \begin{cases} 0, & \text{if } i \text{ and } j \text{ do not match.} \\ 1, & \text{if } i \text{ and } j \text{ match.} \end{cases}$$

Ordinal features

Ordinal features are similar to the nominal features with the additional feature that the ordering of their numbers is meaningful. For example, the number '1' from the ranking {1, 2, 3} has only a meaning in relation with the other numbers '2' or '3'. In terms of measuring dissimilarity, ordinal features are treated in the similar way as numerical features.

Mixed features

In practice, most problems generally do not consist of homogenous feature types, but rather they consist of mixed feature types. Several ways can be used to deal with these kinds of mixed types. One way is to map all the mixed variables into the interval (0,1) and then use some dissimilarity function like the Euclidian distance to compute the matching coefficient. Another way is to transform the variables into binary variables and then use some binary similarity function to compute the similarity. However, these methods have the information loss as drawback. Another simple way is to group the features of the same types together and then separately compute the similarity for each group. In order to obtain the overall similarity results, the set of group similarity results have to be comparable. A more powerful alternative was described by Grower as follows:

$$S(i,j)=\frac{\sum_{k=1}^{m}\eta_{ijf}S_{ijk}}{\sum_{k=1}^{m}\eta_{ijk}},$$

where η_{ijf} is a 0-1 coefficient, based on whether the feature of two objects is missing or is zero, S_{ijf} describes the similarity for the k^{th} feature, and *m* is number of features.

3.3.3 Grouping objects

This is what is often referred to as clustering by using a clustering algorithm. Several clustering algorithm have been developed to solve different problems in specific fields. However, there is no universal clustering algorithm that can be used to solve all problems. Therefore, it is important to carefully investigate the characteristics of the problem to be cluster, in order to select or design an appropriate clustering algorithm. There are many different criteria's, which provide significant distinctions between the clustering algorithms and assist by selecting the suitable algorithm for a certain problem. Different criteria will usually lead to different taxonomies of clustering algorithms. Moreover even a short list of some well-known clustering algorithms can fit into several sensible taxonomies. A rough but widely agreed clustering algorithms taxonomy is given by Jain and Dubes in [27] as shown in Figure 3-3. The clustering algorithms taxonomies are usually obtained considering the following characteristics:

- **Hierarchical vs. Partitional:** This characteristic has to do with the output representation. Hierarchical clustering algorithm is a nested sequence of partitions, whereas a partitional algorithm is a single partition of the collection of items into clusters. Another difference is that the hierarchical clustering uses a proximity matrix to determine relationships between objects, whereas partitional clustering uses a pattern matrix. A partition of the data can be derived from a hierarchy by cutting the tree of clusters at some level.
- **Monothetic vs. Polythetic:** This characteristic relates to sequential or simultaneous usage of the data features. A monothetic clustering algorithm uses the features sequentially (e.g., one feature at a time), whereas a polythetic approach uses all the features simultaneously. The major drawback of the monothetic approach is that it generates a large number of clusters, namely 2^d, where d is the dimensionality of the patterns and for a large value of d these generated clusters are uninteresting and difficult to interpret. Due to this drawback of monothetic algorithms, the majority of clustering algorithms used in practice are polythetic.
- **Hard vs. Fuzzy:** This characteristic indicates the degree of membership of a data point to a cluster. In hard or crisp clustering algorithms, each point belongs to exactly one cluster, whereas in fuzzy clustering algorithms different data points have different degrees of membership to different clusters. The membership functions are generally normalized so that the sum of the membership degree of a point to clusters always sums to '1'. For example, in case of hard clustering a data point belongs to one and only one cluster, whereas in case of fuzzy clustering a it may belong for 80% to a cluster and 20% to another cluster.
- **Deterministic vs. Stochastic:** Deterministic clustering algorithm uses deterministic steps during the clustering process (e.g., there is only one possible labeling). In contrast to deterministic, stochastic algorithms uses random steps to find the best clustering, which is typically done through a random search of the state space consisting of all possible labels.

- **Incremental vs. Non-Incremental:** This aspect is related to the initialization of the data points. The incremental algorithms can be initialized with some data and incrementally refined points (e.g., data points are assigned to the cluster they belong to when they are arrived), while the non-incremental approach mainly expects that the whole data set is ready before applying the algorithm.

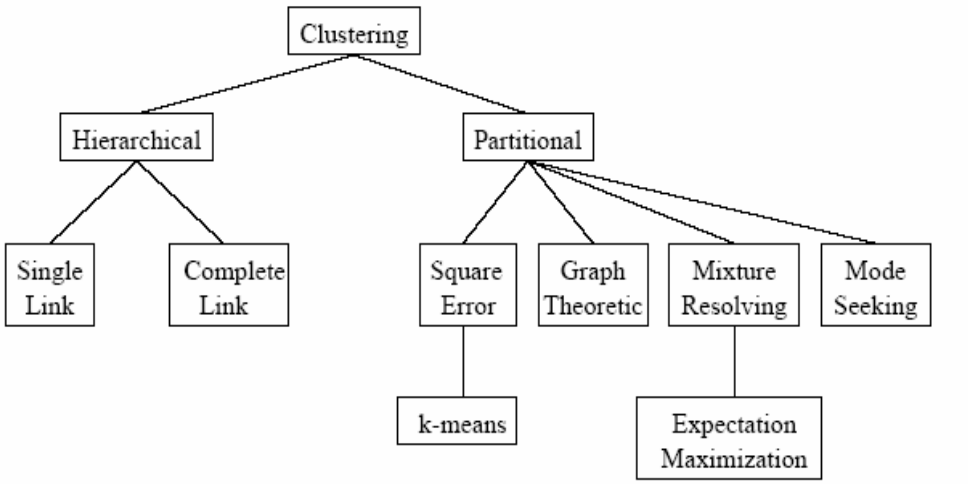


Figure 3-3: Taxonomy of clustering methods.

A short description of the two main classes of clustering algorithms is discussed below.

Hierarchical algorithm

A hierarchical algorithm yields a tree or dendrogram representing a nested set of partitions and the similarity levels at which the set changes. A dendrogram consists of layers of nodes that represent a cluster. These clusters are represented by lines connecting nodes, which are nested into one another. The result of a hierarchical algorithm is shown in figure 3-3. According to the way the tree or dendrogram is constructed, hierarchical algorithms can mainly be classified in one of the following two classes:

- **Agglomerative (bottom-up) algorithms:** These kinds of algorithm starts with each data point in a separate cluster and successively merge these clusters according to a distance measure. The algorithm will stop when all objects are in a single cluster or when some

stop criterion is met. Because each data point starts in a separate cluster, they also called bottom-up algorithms.

- **Divisive (top-down) algorithms:** Divisive algorithms follow an opposite strategy, in that they start with all the data in one cluster and successively split clusters into smaller ones, until each cluster falls in single cluster or some predefined stop-criterion is satisfied. Due to this clustering strategy they also referred as top-down algorithms.

Partitional clustering

Partitional clustering algorithms aim to directly obtain a single partition from the data set, without any sub-partition like the case in hierarchical algorithms. In general, partitional clustering algorithms attempt to minimize a cost function or an optimality criterion, which reflects the agreement between the data and the partition (i.e., how good a partition fits with respect to the data set). Most of these algorithms use cluster prototypes to obtain the evaluation of a given partition. Each prototype is assumed to be a typical representative of the group of similar data points. In an ideal case, each prototype will take the shape that best describes its cluster. In practice, however, most algorithms assume some simple shapes in order to optimize the computational cost. Some important categories of partitional clustering algorithms will be discussed below:

- **Algorithms using the squared error:** These kinds of algorithms rely on representing each cluster by a prototype and attempt to minimize the cost function, which is the sum over all the data points of the squared distance between the data points and the prototype of the cluster it is assigned to. Typically these prototypes are the cluster centroids, such as the case in the popular k-means algorithm.
- **Graph Theory-Based algorithms:** These approaches consider the whole data set as graph, where the nodes of a weighted graph corresponds to data points in the pattern space and the edges reflect the proximities (similarity/dissimilarity) between each data point pair. The graph can be simplified to an un-weighted threshold graph considering a proximity matrix defined as follows:
  $D_{ij} = \begin{cases} 1 \ \ if \ D(o_i, o_j) < \tau \\ 0 \ \ otherwise \end{cases}$, where τ is the threshold value.

- **Mixture Densities-Based Algorithms:** These algorithms rely on the assumption that the clusters were generated according to several probability distributions. Clusters may have arbitrary shape and data points and different clusters may be generated by different distribution. These different distributions can either be derived from different types of density functions (e.g., Gaussian distribution) or by the same type of density function with different parameters. The goal is then to estimate the parameters of each distribution model.
- **Model-Based Algorithms:** These kinds of algorithms rely on finding the best approximation of model parameters that best fit the data set. In general, artificial neural networks techniques are used for these types of algorithms. Accordingly, some authors classified these algorithms as neural network-based algorithms. This type of algorithm can either be partitional or hierarchical. This depends on the data model and the way this model identifies partitions.

3.3.4 Data abstraction

This is the procedure of using or evaluating the clustering results obtained by the clustering process. Many clustering algorithm will always generate clusters, no matter whether the data contains clusters or not. In addition, different clustering approaches usually lead to different clusters. Moreover, the same algorithm with different initial parameters or different input order may affect the finale results. Therefore, effective evaluation standards and criteria are important to provide the users with a degree of confidence for the clustering results derived from the used algorithms. These validity assessments should be objective and algorithm independent. They should also provide the user with answers to the questions such as how many clusters are hidden in the data, whether the clusters obtained are meaningful or just artifact, or why some algorithm is more useful instead of another? Generally, there are three types of validation studies: external indices, internal indices and relative indices. External indices use a human reference classification as a standard to validate the clustering solutions. Typically, they are based on some pre-specified structure, which is the reflection of prior information on the data. Internal indices are not dependent on any external information. In contrast, they examine the clustering structure directly from the

original data. Relative indices can be derived from internal indices by evaluating different clustering results and comparing their scores.

3.3.5 Output interpretation

This is the process of extracting a simple and compact representation of a data set. Here, simplicity is either required for making the further processing more efficient or making the obtained output easyly interpretable by humans. In the clustering context, this is a compact description of each cluster, usually in terms of cluster prototypes or representative patterns such as the centroid [44]. The major goal is to provide the experts in the relevant fields with a meaningful insight from the original data, so that they can interpret the data and solve the problems encountered.

3.4 CLUSTERING ISSUES

Choosing a proper clustering algorithm for a particular problem depends on the type of the data features as well as on the nature of the problem. Accordingly, many clustering algorithms can deal only with a particular size and feature type of a data set. Consequently, some researchers noted some issues, which have to be considered while choosing a good clustering algorithm. Some of the general issues regarding clustering algorithms are as follows:

- **Scalability:** Many algorithms, like some hierarchical clustering algorithms are not able to handle larger datasets. Therefore, a clustering algorithm should have an efficient time and space complexity in order to be able to deal with large datasets.
- **Ability to deal with a mixture of feature types:** In practice, the datasets of many applications consist of features with varying types. Consequently, a good clustering algorithm should be able to handle mixture types when computing the proximity between two objects.
- **Ability to find arbitrary cluster shapes:** Many clustering algorithms cannot handle all kind of cluster shapes, but tend to find only some specific cluster shapes. For example k-means tend to find only spherical shapes. Thus, an algorithm should be able to find clusters of arbitrary shapes.

- **Minimal requirements of domain knowledge**: Many clustering algorithms require some predefined parameters (e.g., number of clusters in k-means algorithm) or some stop-criterion (e.g., number of iterations in k-means) in order to cluster the dataset. However, a good clustering algorithm should be insensitive to these kinds of domain knowledge requirements in order to improve the cluster quality.
- **Ability to deal with noisy datasets**: In practice, the datasets are often not errorless, but they typically contain noisy patterns and outliers. The clustering algorithm should be robust enough to minimize the effect of this noise.
- **Insensitivity to order of dataset:** Some algorithms may produce different clusters when they are presented with different ordering of the same data. A clustering algorithm should therefore be insensitive to the order of the data.
- **Ability to handle datasets with high dimensionality:** The number of features (dimensions) in the feature space for many clustering problems are very large, and many clustering algorithms can handle only small dimensions. A good clustering algorithm should not only be able to handle large datasets, but also be able to handle data with high dimensionality.
- **Interpretability and usability of clustering results:** A clustering algorithm should come up with usable and interpretable clustering results (easy to understand results).

3.5 ENSEMBLE CLUSTERING

As already mentioned, a large number of clustering algorithms exist, but because of the varying characteristics of the datasets and clusters, there is not a single clustering algorithm that can handle every given dataset successfully, i.e., which is able to identify all kinds of cluster-shapes and structures that are encountered in practice. For example, k-means can only handle numerical datasets and have difficulties to deal with complex cluster-shapes. An alternative approach to overcome these drawbacks and address many of the clustering issues mentioned in the previous section is to integrate multiple clustering results. This idea of integrating multiple data sources (e.g., multi-sensor data fusion) and/or approaches (e.g., multi-classifier in learning systems) to obtain better results is found in several disciplines and applications. In clustering techniques applications

this is the research on cluster ensembles. Cluster ensemble attempts to find more accurate and robust clustering results by combining the clustering results of a single or multiple clustering algorithms. Ensemble typically consists of two parts: an ensemble constructor and a consensus function. The ensemble constructor is used to refer to the diversity strategy used to generate diversity clusters or also called base clusters from the original data. The consensus function is then used to combine these base clusters into a final single clustering result. These ensemble techniques require two key issues to be addressed. Firstly, how to generate a collection of base clusters from which the ensemble is composed? Secondly, how can one combine these base clusters to produce the final output? Several ensemble generator methods as well as consensus functions exist to do this. Figure 3-4 shows a general process-flow of an ensemble clustering:

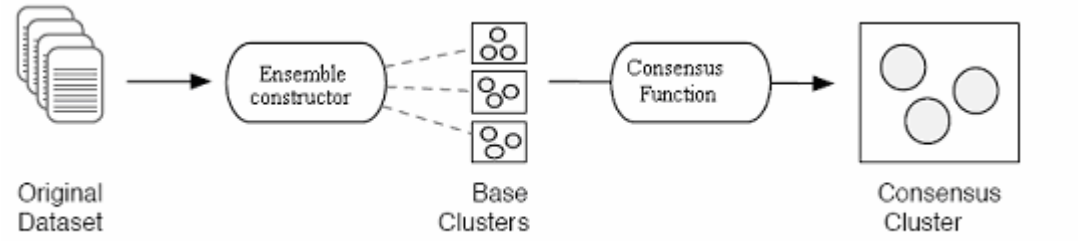


Figure 3-4: The process-flow of the cluster ensembles.

Cluster ensembles are based on the idea of combining multiple clustering results of a given dataset $X = \{X_1, X_2, ..., X_n\}$ to produce a solution that is often unattainable by any single clustering algorithm. These techniques generally follow a process as illustrated in Figure 3-4 and consist of two distinct phases.

- **Ensemble constructor:** The input data set can be represented as a *nxd* pattern matrix or *nxn* proximity matrix. Given a data set of n instances $X = \{X_1, X_2, ..., X_n\}$, an ensemble constructor generates a set of base clusters, represented as $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$, where *r* is the number of base clusters (the ensemble size). Each base cluster in Π is a set of clusters $\pi_i = \{C_1^i, C_2^i, ..., C_{k(i)}^i\}$, where $X_i = C_1^l \cup C_2^l \cup ... \cup C_{k(i)}^l, \forall \pi_i$, and k_i is the number of clusters in the i^{th} base cluster. These base clusters can either be hard or

soft clusters. In this report the ensemble constructor will be referred as generation function.

- **Ensemble consensus:** Once a collection of base clusters has been generated, a suitable consensus function $f : \Pi = \{\pi_1, \pi_2, ..., \pi_r\} \rightarrow \sigma$ has to be designed in order to combine the base clusters and find a final consensus cluster $\sigma_i = \{C_1, C_2, ..., C_M\}$. In order to find this final consensus cluster σ, a consensus function utilizes information from the base clusters in $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$, such that the objects in a cluster of σ are more similar to each other relative to objects in different clusters of other clusters. Integration function will be used in this report to denote the ensemble consensus.

Based on this idea several ensemble generation functions as well as integration functions have been designed for different problem domains. Taxonomy of the commonly used ensemble generation and integration methods as mentioned in [37] is shown in Figure 3-5.

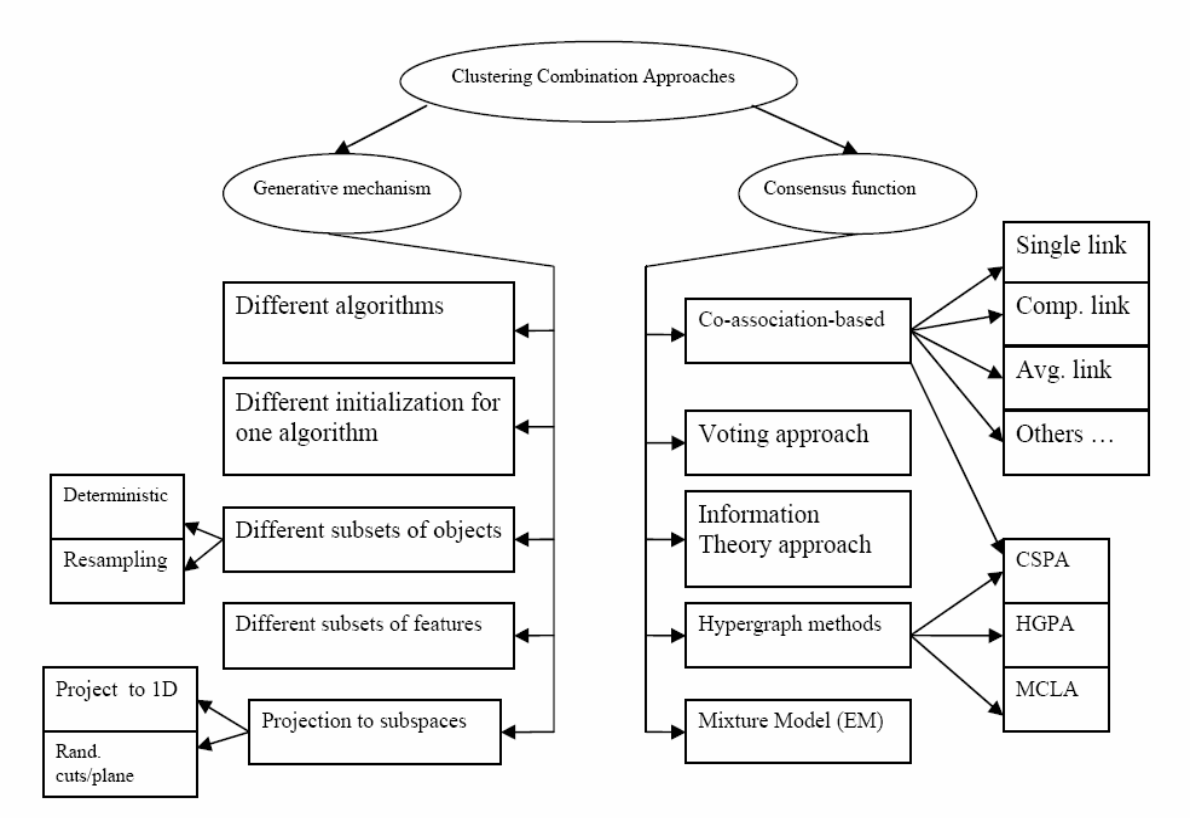


Figure 3-5: Taxonomy of different cluster ensemble.

3.5.1 Ensemble generation

Research has demonstrated that supervised ensembles are most successful when they are constructed from a set of accurate classifiers whose errors lie in different parts of the data set. Similarly, unsupervised ensemble constructors typically seek to encourage diversity in order to improve quality of the information available in the integration phase and to optimally integrate clustering ensembles in a robust and stable manner. A variety of strategies have been proposed to achieve this goal. Generally these include:

- **Using different clustering algorithms.** As already mentioned, some clustering algorithms may perform well for certain datasets relative to other clustering algorithms. Therefore, instead of running the risk of picking an unsuitable clustering algorithm, one can use different clustering algorithms to produce and combine the result of each algorithm to produce the final output.
- **Using different initialization or other parameters for one clustering algorithm.** The output results of some clustering algorithms rely on the initialization of the algorithm's parameter, i.e., different initialization may lead to different results. Therefore, combining results of different initializations for a certain clustering algorithm may lead to a robust and stable final result. An example of such algorithms is the k-means algorithm.
- **Using different subsets of features to represent a pattern.** Each pattern may be represented using different features, for example, an image can be represented by its pixel, histogram, location and parameter of perceptual primitives or 3D scene coordinates. Many clustering algorithms are not able to handle all feature types when computing the similarity coefficient between two data points. Consequently, using different features to represent a pattern may help to improve the quality and robustness of the final result.
- **Using different subsets of the original dataset.** This is suitable in cases where the data set to be clustered is large, and constraints on execution time or memory space may affect the architecture of the algorithm.

- **Using different orders of the dataset.** Some on-line clustering algorithms like BIRCH are sensitive to the data order. Therefore, this strategy may help to improve the final output.

3.5.2 Ensemble integration

Once the ensemble constructor generates a collection of base clusters, one can produce a single solution by combining these base clusters. However, it has been observed that the success of a supervised ensemble technique depends not only on the presence of a diverse set of base classifiers, but also on the ability of the integration method to exploit the resulting diversity. Similarly, the choice of a suitable consensus function for combining the ensemble base clusters will greatly affect the accuracy of the final clustering solution. Several different approaches have been proposed in the literature for performing the task of ensemble integration. The most popular has been to use information provided by the base clusters to derive a new measure of similarity between data objects, i.e., without using the original data set. A brief review of several known consensus functions is given below.

- **Co-association based:** Similarity between a pair of patterns (co-association values) o_i and o_j can simply be estimated by counting the number of clusters shared by these objects in all the base clusters $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$, i.e., how many times o_i and o_j are put in the same cluster in the clustering ensemble. This similarity between two patterns expresses the strength of co-association of patterns by a matrix containing the values (co-association matrix) and it can be defined as follows:

$$S_{ij} = S(x_i, x_j) = \frac{1}{R}\sum_{k=1}^{R}\delta\left(\pi_k(x_i), \pi_k(x_j)\right), \text{ where } \delta(a,b) \equiv \begin{cases} 1, \text{ if } a = b. \\ 0, \text{ if } a \neq b. \end{cases}$$

  These kinds of consensus functions operate on the co-association matrix. Numerous similarity-based clustering algorithms can be applied to the co-association matrix in order to obtain the final clustering output. Examples of these algorithms include single-link, complete-link and average-link.

- **Voting approach:** The voting approach assumes that the label correspondence problem is solved and then uses a majority vote to determine the final clustering output.

However, this label correspondence problem is exactly what makes the clustering procedure difficult. Therefore, some heuristic approximation is needed to generate consistent labeling. The main idea is to rearrange the cluster labels such that best agreement between the labels of two clusters is obtained. All the clusters from the ensemble must be re-labeled according to their best agreement with some chosen reference cluster (partition). The reference cluster can either be taken from the ensemble clusters, or from a new clustering of the original data set. The complexity of the re-labeling process is $k!$, which can be reduced to $O(k^3)$ in cases where the Hungarian method is employed for the minimal weight bipartite matching problem. However, these kinds of consensus functions suffer from a number of drawbacks including high complexity, the heuristic character of objective function and uncertain statistical status of the consensus solution.

- **Information theory approach:** This approach maximizes the mutual information (MI) between the empirical probability distribution of labels (clusters) in the consensus clustering and the labels (clusters) in the ensemble (base clustering). The optimal combined clustering should share the most information with the original clustering results. Assuming that the clusters in the ensemble are independent, mutual information can be expressed as the sum of pair wise mutual information between the target and given clusters. Using the classical Shannon definition one can determine the quality of the consensus clustering by the amount of information it shares with the given cluster. However, maximizing the consensus with this classical definition is difficult. Therefore, an alternative solution can be obtained with the qualitatively similar generalized mutual information. The complexity of this consensus function is $O(kNB)$, where k is the number of clusters in the components of the combination, N is the number of data points, and B is the number of clusters to be combined. Though these types of algorithms can be potentially trapped in a local optimum, its relatively low computational complexity allows using multiple restarts in order to choose a quality consensus solution with minimum intra-cluster variance.

- **Hypergraph methods:** The clusters of different ensembles are represented by hyper-edges on a graph whose vertices correspond to the objects to be clustered. The problem of consensus clustering is then to find the minimum-cut of a hyper-graph. A k-way minimum cut of this hyper-graph gives the required consensus clustering. These algorithms seem to work effectively for approximately balanced clusters. Though the hyper-graph partitioning problem is NP-hard, several efficient heuristics have been proposed to achieve the k-way min-cut partitioning problem. This includes the CSPA, the HGPA and the MCLA, with a computational complexity estimation of $O\left(kN^2B\right)$, $O\left(kNB\right)$, and $O\left(kN^2B^2\right)$, respectively.
- **Mixture model (EM):** This method relies on solving the cluster integration problem in the space of cluster labels via expectation maximization (EM). All the data is assumed to be independent and identically distributed. This makes it possible to represent the log likelihood function for the unknown parameters. The objective of consensus clustering can then be formulated as a maximum likelihood estimation problem. The best fitting mixture density for a given data can be found by maximizing the likelihood function with respect to the unknown parameters. Each component in the density distribution corresponds to a cluster in the target consensus clustering, and is assumed to be a multivariate, multinomial distribution. The consensus clustering can be found as a solution to the maximum likelihood problem for a given clustering ensemble. This maximum likelihood problem can be solved using the EM-algorithm.

Chapter 4: Proposed system

This chapter presents the design of the address-block segmentation workbench system. First, an overview of the system is given, followed by the functional requirements of the system. Subsequently the non-functional requirements of the system will be discussed and finally the pseudo requirements of the system will be discussed.

4.1 OVERVIEW

The address-block segmentation workbench system is a system, which provides the functionality to segment a mail piece image using several single clustering algorithms and ensemble algorithms. The workbench provides also the functionality to compare and evaluate the segmentation results. The main components of the system's architecture are as shown in figure 4-1:

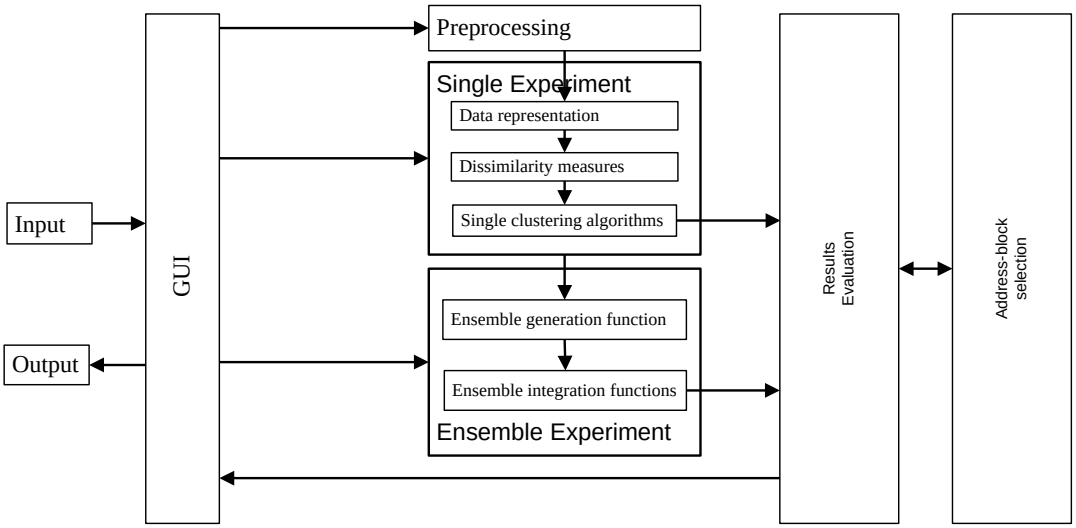


Figure 4-1: Proposed system architecture.

First, the GUI is used to insert the mail image to be segmented and the related ground-truth segmentation. The pre-processing component is then used to obtain a binary version of the image. Afterwards connected components are derived from this binary image. The user can then define several single or ensemble experiments to segment the image. In case of an experiment with single clustering, the user needs to define the connected components features to be used (data

representation), the way to measure closeness between those features, a certain clustering algorithm to group the feature with, and clustering algorithm parameters if necessary. In case of segmentation with ensemble experiments the user needs to define a diversity strategy to generate base clusters and an integration function to combine those base clusters to the final output. The diversity can for instance be obtained from several single experiments with the same single clustering algorithm and different features, different single clustering algorithms, and so on. The defined experiments are then sequentially applied on the image to gain segmentation results. Afterwards, the obtained results are evaluated using the ground-truth segmentation and classification results. Finally, the system returns the evolution result for each experiment. Both the segmentation results and the evaluation are then available to be used. This section provides a brief explanation of each component.

4.1.2 Input data

This component contains the inputs of the system. The inputs that are needed are mail piece image and a ground-truth-segmentation file also called labeling file, which contains several information of the input image. The next paragraphs will give a short explanation of both inputs.

Mail piece image

This is a digital image, which is captured live from a real mail stream. Normally, address-block segmentation is an online (real-time) process, which means that mail pieces are sequentially captured and sent to the address-block segmentation system. However, instead of this, an offline approach will be used for this project. This means that the images have to be read from a database, which contains different kinds (parcels, postcards, magazines, etc…) of mail piece images that are already captured. The images can be grey-level or RGB-images. Some example images are shown in figures 4-2a, to 4-2d.

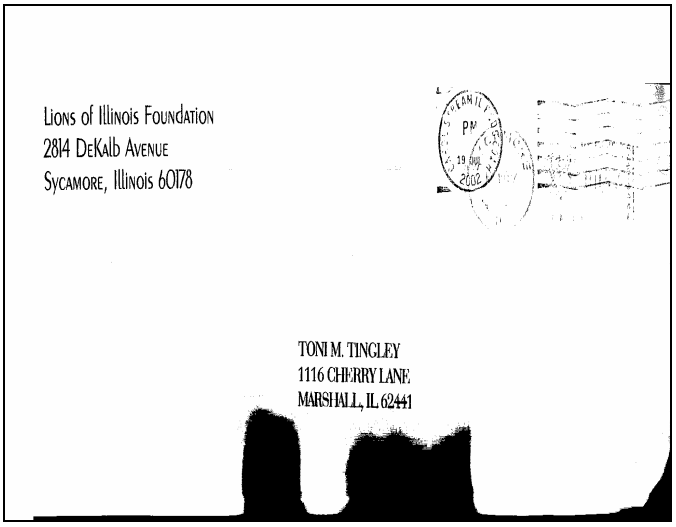


Figure 4-2a: Image of a normal letter.

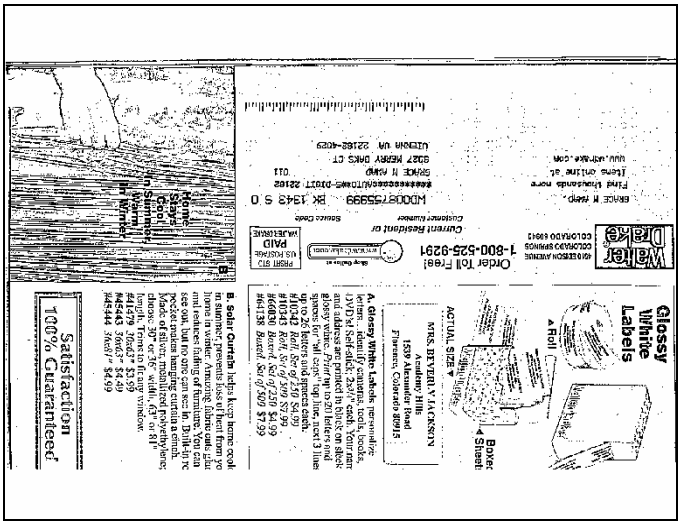


Figure 4-2b: Image of a magazine.

Figure 4-2c: Image of parcel on a tray.

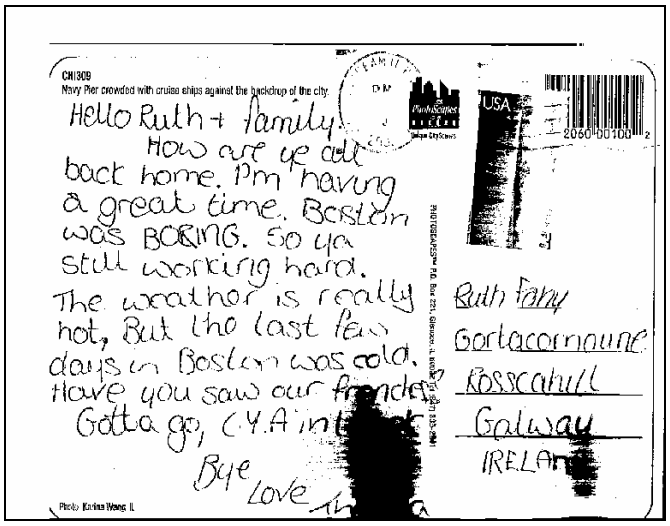


Figure 4-2d: Image of postcard.

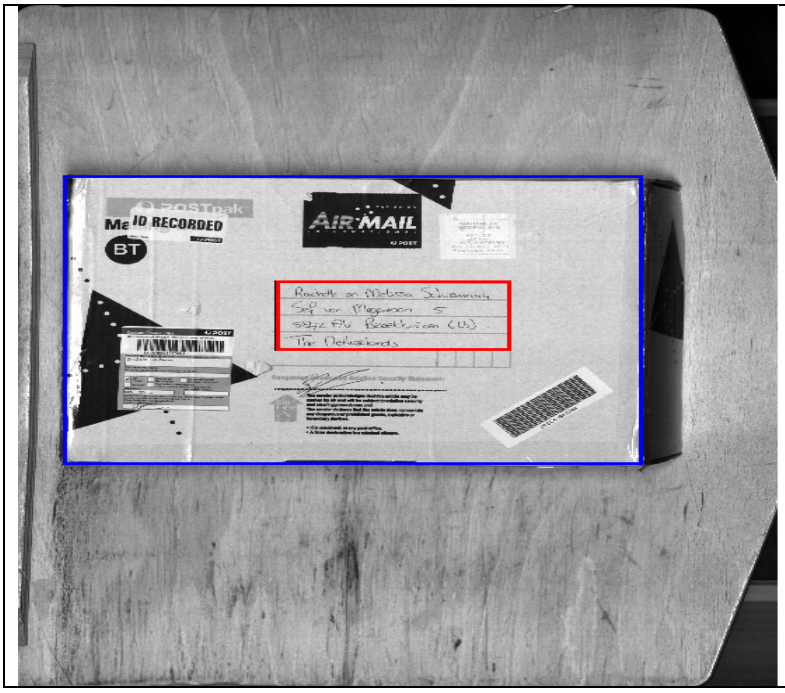
Ground-truth segmentation

This is a CSV-file, which contains all kinds of information about the image. This labeling file is obtained using a labeling-tool and human-analysis of the image. The labeling-information attributes that are relevant for this project are as follows:

- **Document ID:** This attribute holds a unique id of the image.

- **Address Block Coordinates:** This attribute represents the bounding box coordinates of the address-block (i.e., center of gravity for the x-coordinate, center of gravity for the y-coordinate, width, height, and angle).
- **Crop Coordinates:** This attribute is only needed in case of parcels and it represents the coordinates of parcel without parcel holder. This is because in case of parcel the image contains not only the parcel but also a tray (see figure 4-2c). Note that the position of the parcel is detected by another process using either a hardware or software approach.
- **Writing Style:** This attribute describes the writing style (i.e., machine-print or hand-writing) in which the DAB is written.

An example of the labeling is show below in figure 4-3. The image is shown left and the explanation of corresponding attributes is on the right. The Document id is not described in the example below, because it is just a long unique text and therefore not necessary to be mentioned.



Address-block coordinates: 2838,1404,240,89, -1.457832, this corresponds to the red-block.
Crop Coordinates: 2322,1892,1972,1489, -1.469942, this corresponds to the blue-block.
Writing Style: Hand-writing, this corresponds to the text style in the red-block.

Figure 4-3: Image labeling example.

4.1.2 The GUI

The main tool GUI allows the user to interact with the system. The user interface contains facilities for the user to perform the following actions:

- Insert a single image file or an image database;
- Insert a labeling file;
- Define one or more address-block segmentation experiments with single clustering;
- Define one or more address-block segmentation experiments with ensemble clustering;
- Define which address-block segmentation experiments to run, to evaluate and whether the results have to be showed;
- Applies internal all the experiments that are defined to run on one or more (auto-run) images;
- Shows the segmentation results of all experiments that are defined on the debug area; and
- Shows the evaluation results of all experiments that are defined with evaluate on the debug area.

More explanation of the user-interface and these facilities can be found in the user-interfaces paragraph of the next chapter.

4.1.3 Preprocessing

The preprocessing component consists of two main components, namely: Binarizor and connected component labeling. A short description of each component is given below.

Binarizor

The goal of this component is to obtain a binary image by preprocessing the input image (e.g., gray-scaled or color image) such that the objects in the image are separated from the background. A binary image has two states, one of these states will indicate the foreground objects, i.e., hand written text, printed text, figures, etc, while the complementary state will correspond to the background. Depending on the application domain, the foreground can be represented by black and the background by white or visa versa. Separating the foreground from the background of an

image is an important preprocessing step in any image-analysis [40]. Its purpose is to acquire some useful information in the image for higher-level image processing. Thresholding is widely used in image-analysis and generally its process is to determine first a grey threshold according to some objective criteria and then assigns each pixel to one class (e.g. foreground; usually black) if its grey-level is below the determined threshold otherwise it is assigned to the other class (e.g. background; usually white). There are a lot of methods available, which can be used for thresholding an image. Sezgin, M., and Sankur, B. [40] described in their paper thresholding algorithms, analyzed them, evaluated their performances, and categorized them according to the image information contents they exploited into six classes:

- **Histogram shape-based methods:** This kind of algorithm analyzes, for example, the peaks, valleys and curvatures of the smoothed histogram.
- **Clustering-based methods:** This class of algorithm either uses a clustering analysis to cluster the input image samples into two parts, background and foreground, or a mixture of two Gaussians to model the input image.
- **Entropy-based methods:** This kind of algorithm refers to algorithms that use the entropy of the foreground and background regions, and the cross-entropy between the original and binary image.
- **Object attribute-based methods:** This kind of algorithm selects the threshold value based on some attribute quality or similarity measure (such as fuzzy shape similarity, edge coincidence, grey-level moments, texture, etc.) between the input image and the binary image.
- **The spatial methods:** This class of algorithm uses higher-order probability distribution and/or correlation between pixels. It utilize not only grey value distribution but also dependency of pixels in a neighborhood, for example, in the form of context probabilities, correlation functions, co-occurrence probabilities, local linear dependence models of pixels, 2-D entropy, etc.
- **Local methods:** This kind of method adapts the threshold value on each pixel based on the local image characteristics; i.e., a threshold value is calculated at each pixel,

which depends on some local statistics like range, variance, or surface-fitting parameters of the pixel neighborhood.

Because of the time limitation, a thresholding algorithm that is already implemented by Prime Vision will be used for this project. This is a method, which is based on the local method principles as described above.

Connected component labeling

This component gets a binary image and uses some connected component labeling method to obtain the connected components from the binary image. Connected component labeling is commonly used to refer to the task of assigning unique labels to each object (a group of directly or indirectly connected pixels) in an image [41]. These labels are used for distinguishing and referencing the objects and they form a fundamental step for any subsequent analysis procedure. This makes connected components labeling one of the most fundamental algorithms of almost all applications in pattern recognition. For example, before one can segment, classify or identify objects or regions in an image, groups of similar pixels have to be identified and labeled. Generally each group of pixels is referred to as an object. These objects allow one to compute the information required for subsequent processing, such as area size, height, width, and perimeter.

There are a number of different ways to define the connectivity of pixels in an image. The simplest approach repeatedly scans the image and assigns labels to each pixel until the labels for the pixels no longer change. There are two scans that can be distinguished: a forward scan which assigns labels to pixels from left to right and top to bottom, and a backward scan which assigns labels to pixels from right to left and bottom to top. Connected components labeling methods are classified into three different categories, namely:

- **Multi-pass algorithm:** These algorithms can control the number of iterations, by alternating the direction of scans, or directly manipulate the equivalence information.
- **Two-pass algorithm:** These kinds of algorithms first scan the image to assign labels and record the equivalence information of each label assigned to an object, then this equivalence information is analyzed in order to determine the final labels. Finally, the labels are assigned to the object pixels by doing a second pass through the image.

- **One-pass algorithm:** The Algorithms based on this approach first scan the image to find an object pixel which is not labeled and then assign the same label to all connected object pixels in one pass.

For this project a connected component-labeling algorithm, which is already implemented by Prime Vision will be used. This is a one-pass approach based algorithm and its process is as follows: First run-length encoding is applied to get a compressed representation of the binary image; i.e., instead of describing each pixel individually, only the start and the end of each run of pixels is described. This representation is then used to drive contours from it. A contour is an object in a binary image. There are two different contours: an outer contour which describes a black area and an inner contour, which describes a white area. An inner-contour always lies within an outer contour. Each inner contour can have only one parent. Finally contours are used to obtain connected components. Generally, a connected component is an outer-contour with al its inner contour childes. A connected component has the following features: Circumference, surface of all black pixels, black-white ratio, skewed ellipse, number of inner contours. It inherits also the following features from a contour: bounding box and surface of center of gravity. An example of one connected component with two inner contours and one outer contour, and one connected component with one outer contour is shown in figure 4-4.

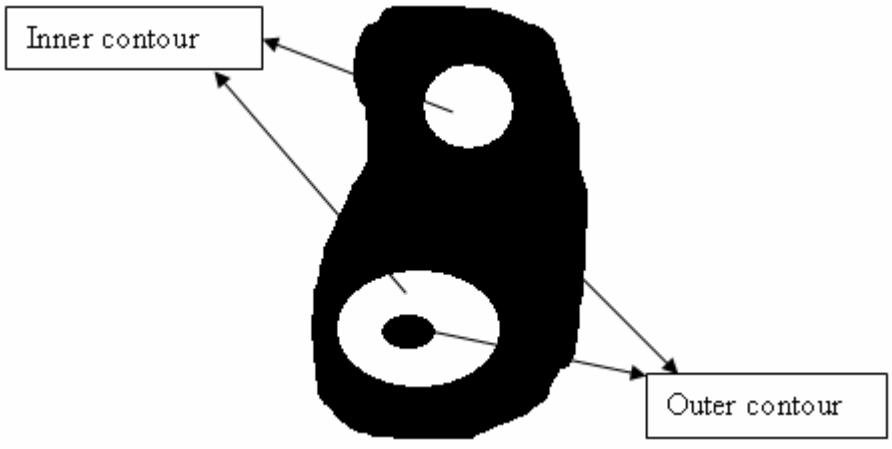


Figure 4-4: Example of connected components and contours.

4.1.4 Data representation

At this stage a feature vector will be defined using the connected components obtained from the preprocessing stage. Given a set of *n* connected components, $[o_1, o_2, ..., o_n]$ each consisting of *m* features (measurements), a feature vector can then be represented as follows:

$$\begin{bmatrix} o_{11} & o_{12} & \cdots & o_{1m} \\ o_{21} & o_{22} & \cdots & o_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ o_{n1} & o_{n2} & \cdots & o_{nm} \end{bmatrix}$$ Where o_{ij} is the j^{th} feature of the object o_i , $1 \le i \le n$ *and* $1 \le j \le m$.

The connected component's features that will be used are as follows:

- The bounding box coordinates (x_0, y_0, x_1, y_1);
- Center of gravity x-coordinate;
- Center of gravity y-coordinate;
- The bounding box width;
- The bounding box height;
- Skewed ellipse;
- Circumference;
- Number of inner contours;
- Black-white ratio (i.e. a ratio between the surface of the outer contour and the inner contours);
- The black area (i.e. surface of the outer contour without the inner contours);
- Dx, we constructed this feature from different features, such that it denotes the minimal x-coordinate distance between two object pairs; and
- Dy, this is the same as Dx except that it denotes the minimal y-coordinate distance between two object pairs.

All the features mentioned above are numerical features. However, not all the connected components are relevant to be clustered. This is because some connected components correspond to noise and others to large objects like, lines, blocks, etc. Using feature selection one can choose to distinguish some features from a set of candidates. Generally this task of feature selection is an optional task. Therefore, this task will be implemented such that it's user defined; i.e. the user can

choose to use it or not to use it. In order to exclude graphic, noise and other irrelevant connected components from the candidate's set the following rules will be used:

- If the width and the height of the connected component are less than T_w and T_h respectively, its size is too small, and therefore it is labeled as noise.
- If the connected component's width to height ratio is larger than R_w, the height to width ratio is larger than R_h, or the density of the connected component is larger than T_d, then the connected component is labeled as graphic.
- If the black area of the connected component is larger than T_{ba} and the number of inner contours of the connected component are larger than T_{nc}, then the connected component is labeled as image.

All the thresholds mentioned above will be empirically chosen. The above rules are based on the classifying rules mentioned in [4] as well as the rules already used in the existing functionality.

4.1.5 Dissimilarity measures

Once the feature vector is defined, one needs to choose a way to measure the closeness between two object pairs. As already mentioned this closeness can be measured using either a similarity or dissimilarity measure depending on the data representation. In this project we will use a dissimilarity measure. This is because all the features that will be used for this project are numerical and these kinds of features are typically measured using dissimilarity. There are several ways to measure dissimilarity. One common way is distance. Distance can be determined in several ways depending on the type of the attributes. In order to test the influence of different distance functions the most common distance functions will be implemented for this project. Given a set of n data points o_i and o_j represented by a numerical feature vector with length *m* a distance between the i^{th} and the j^{th} object, denoted by d_{ij} we can summarize the distance function to be implemented in this project as shown in Table 4-1.

Table 4-1. Dissimilarity functions to be implemented.

Measures name	**Description**	**Formula**
Euclidean distance	Euclidean distance is one of the most common used distance functions. Euclidean distance calculates the root of square differences between coordinates of a pair of objects.	$d_{ij} = \left(\sum_{k=1}^{m} \left| o_{ik} - o_{jk} \right|^2 \right)^{\frac{1}{2}}$
City-block distance	City-block distance also known as Manhattan distance, represents distance between points in a city road grid and calculates the absolute differences between coordinates of two object pairs.	$d_{ij} = \sum_{l=1}^{m} \left| o_{ik} - o_{jk} \right|$
Chebyshev distance	Chebyshev distance also called maximum value distance or sub distance calculates the absolute magnitude of the differences between coordinates of a pair of objects.	$d_{ij} = \max_{1 \le k \le m} \left| o_{ik} - o_{jk} \right|$
Minkowski distance	This is the generalized metric distance. When n=1, it becomes city-block distance and when n=2, it becomes Euclidean distance.	$d_{ij} = \left(\sum_{l=1}^{m} \left| o_{il} - o_{jl} \right|^n \right)^{\frac{1}{n}}$
Canberra distance	Canberra distance determines the sum of series of fraction differences between coordinates of a pair of objects. Each term of fraction differences has a value between 0 and 1.	$d_{ij} = \sum_{k=1}^{m} \frac{\left| o_{ik} - o_{jk} \right|}{\left| o_{ik} \right| + \left| o_{jk} \right|}$
Bray Curtis Distance	Bray Curtis distance is a normalization method that views the space as grid like the city-block distance. This distance has an additional property, which is if all coordinates are positive, the distance value will be between zero and one.	$d_{ij} = \frac{\sum_{k=1}^{m} \left| o_{ik} - o_{jk} \right|}{\sum_{k=1}^{m} \left(o_{ik} + o_{jk} \right)}$

Pearson correlation	Pearson coefficient is standardized angular separation by centering the coordinates to its mean value. The Pearson correlation has a value between -1 and +1 and measures similarity rather than dissimilarity. However, one can still obtain the dissimilarity by subtracting the similarity value from one.	$d_{ij} = (1 - s_{ij})/2$, where similarity, $s_{ij} = \frac{\sum_{k=1}^{m}(o_{ik} - \bar{o}_i)(o_{jk} - \bar{o}_j)}{\sqrt{\sum_{k=1}^{m}(o_{ik} - \bar{o}_i)^2 \sum_{k=1}^{d}(o_{jk} - \bar{o}_j)^2}}$
Cosine similarity	Cosine similarity represents cosine angle between two vectors. It also measures similarity rather than dissimilarity. Therefore, higher value of angular separation indicates that the two objects are similar. In order to obtain the dissimilarity the similarity value has be subtracted from one.	$d_{ij} = (1 - s_{ij})$, where similarity, $s_{ij} = \cos\alpha = \frac{o_i^T o_j}{\|o_i\| \|o_j\|}$

4.1.6 Single clustering algorithms

After the data representation has been well defined and the dissimilarity measure has been chosen one needs to choose a clustering algorithm for clustering the objects. As already mentioned in the previous chapter clustering algorithms are generally classified as hierarchical versus partitioned. Several clustering algorithm of each class have been developed to solve different problems in different fields. Therefore, it is important to carefully select or design an appropriate clustering algorithm that satisfies the characteristics of the problem to be clustered. In order to determine which of these classes is more suitable for the address-block segmentation problem, different clustering algorithms of both classes need to be implemented. However, because of the huge number of clustering algorithms versus the project time, it will be impossible to implement all the clustering algorithms. Therefore, only the following clustering algorithms have been selected to be implemented: k-means, expectation maximization, iso-data, single-linkage, complete-linkage, and average weighted-linkage. The Prime Vision clustering algorithm (multi clustering algorithm)

will also be used for this project. A justification for the clustering choice and a description of each clustering algorithm is given below.

Expectation-maximization (EM)

The Expectation-maximization algorithm, or EM algorithm, is a mixture densities-based algorithm, which relies on the assumption that the clusters were generated according to several probability distributions. Clusters may have arbitrary shapes and data points in different clusters may be generated by different distribution, which can be derived from different types of density functions (e.g., Gaussian distribution), or by the same type of density function, but with different parameters. The goal is to estimate the parameters of the distribution by fitting k component density functions $p_i(o|\theta_i)$ to a data set *D* with m features. Let $o \in D$ be a data point of data set *D*, the mixture model probability density function evaluated at x is:

$p(o) = \sum_{i=1}^{k} p_i(o|\theta_i) p_i(o)$, where $p_i(o|\theta_i)$, $i = 1,...,k$ denotes the cluster or density functions modeling the data points of the i^{th} cluster, k is the number of clusters, $\theta_i = (\theta_1,...,\theta_k)$ represents the specific parameters needed to compute the value of $p_i(o|\theta_i)$ and $p(o)$ denotes the fraction of a data point x belonging to the i^{th} cluster and sums to one. The probability of membership of a data point 'o' in cluster 'i' can be computed as follows:

$$p_i(cc) = \frac{p_i(o) p_i(o|\theta_i)}{\sum_j p_j(o) p_j(o|\theta_j)}.$$

By assuming that each cluster is generated by a Gaussians probability distribution, we can compute the density function for cluster i^{th} as follows:

$p_i(o|\mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(o-\mu_i)^2}{2\sigma_i^{\,2}} \right)$, where μ_i is mean object and σ_i denotes the standard deviation. Let further $\phi = \{p_i, \mu_i, \sigma_i\}\ i = 1,...,k$ represent a collection of mixture model parameters. The quality of a given set of parameters is a measure of how well a certain mixture model fits the data. This is quantified by the log-likelihood of the data given the mixture model

$$L(\phi) = \sum_{cc \in CCs} \log\left(\sum_{i=1}^{k} p_i(o) p_i(o|\mu_i, \sigma_i) \right).$$

The EM algorithms can now be defined as follows:

1. Define the number of the desired clusters *k* and the maximum number of iteration allowed *MaxIter*.
2. Choose initialization parameter settings ϕ: randomly choose *k* prototypes from the set data points as μ_i, set $p_i(o) = \frac{1}{k}$.
3. Let t = 0.
4. For each data point $o \in D$
   - ❖ E-step: Compute the membership probability of x in each cluster $i = 1,...,k$
     $$p_i^{t+1}(x) = \frac{p_i^t(o) p_i^t\left(o \middle\| \mu_i^t, \sigma_i^t\right)}{\sum_j p_j^t(o) p_j^t\left(o \middle\| \mu_j^t, \sigma_j^t\right)}.$$
   - ❖ M-step: Update the model parameters
     $$p_i^{t+1}(o) = \sum_{x \in D} p_i^t(o),$$
     $$\mu_i^{t+1} = \frac{\sum_{x \in D} p_i^t(o) o}{\sum_{x \in D} p_i^t(o)},$$
     $$\sigma_i^{t+1} = \frac{\sum_{x \in D} p_i^t(o)\left(o - \mu_j^{t+1}\right)^2}{\sum_{x \in D} p_i^t(o)}.$$
   - ❖ Stopping Criteria: If $\left|L(\phi^t) - L(\phi^{t+1})\right| \le \varepsilon$ or the maximum number of iteration is reached, stop. Else set $t \leftarrow t+1$ and go to step 4.

K-Means

The k-mean and its variants are the most popular clustering algorithm. It usually uses an objective function that attempts to minimize the intra-cluster variance or the sum-of-squares between the data point and the prototypes (i.e., centroids) of the cluster it is assigned to. This is given in the following formula:

$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| o_i^{(j)} - C_j \right\|^2$, where $\left\| o_i^{(j)} - C_j \right\|^2$ is the chosen distance measure between a data point $x_i^{(j)}$ and the j^{th} cluster prototype C_j.

Based on the above formula, the k-means algorithm can be described in the following steps:

1. Define the number of the desired clusters *k* and the maximum number of iteration allowed *MaxIter*.
2. Initialize the k- prototypes randomly or based on some prior knowledge.
3. Assign each data point to the closest cluster prototype C_w,
   i.e. $o_j \in C_w$, *if* $\|o_i - C_w\| < \|o_i - C_i\|$ *for* $i = 1,...,n, i \neq w,$ *and* $i = 1,...,k$.
4. Recalculate the entire cluster prototype using the current partition.
5. Repeat Steps 3 and 4 until the prototype no longer move (i.e. $C_{w+1} - C_w \leq \varepsilon$) or the maximum number of iteration is reached.

K-means has been used in solving many practical problems, because it is very simple, easy to implement, and its time complexity is $O(nkr)$, and since *k* (number of desired clusters) and *r* (number of iterations) are generally much less then N (number of data points), it can also be used for large data sets.

ISO-Data

ISO-Data stands for Iterative Self-Organizing Data Analysis Techniques. The ISODATA algorithm is similar to the K-means algorithm with the distinct difference that the ISO-DATA algorithm allows different number of clusters while the k-means assumes that the number of clusters is known a priori. It applies also some further refinements by splitting and merging of clusters. The clusters are merged if either the number of data points in a cluster is less than a certain threshold or if the prototype of two clusters is closer than a certain threshold. On the other hand clusters are split into two different clusters if the cluster standard deviation exceeds some predefined threshold and the number of data points of the cluster is twice the threshold of the minimum number of data points in a cluster. The ISO-Data can then be described as follows:

1. Define the following parameters:The number of cluster to start with *k*.
    a. The Maximum number of iteration allowed *MaxIter*.
    b. The Maximum number of data points allowed in a cluster $T_{\max}$.
    c. The Minimum number of data points allowed in a cluster $T_{\min}$.
    d. The standard deviation threshold value (needed for split operation) T_{std}.
    e. The pair-wise distance threshold value (needed for the merge operation) T_{dis}.
2. Initialize the k- prototypes randomly or based on some prior knowledge.
3. Assign each data point to the closest cluster prototype C_w,
   i.e. $o_j \in C_w$, *if* $\|o_i - C_w\| < \|x_i - C_i\|$ *for* $i = 1,...,n,\ i \neq w,$ *and* $i = 1,...,k$.
4. Discard clusters with less than $T_{\min}$ data points, i.e., reassign the data points of those clusters to the closest cluster prototype, remove those clusters, and decrease *k*: $k \to k-1$.
5. Compute the average distance of data points from their corresponding cluster prototype D_{avg}^{j}.
6. Compute the overall average distance of the data points from their respective cluster prototypes D_{avg}.
7. Split the cluster if the splitting rules are satisfied, i.e., if the number of data points in a cluster > 2 * $T_{\min}$, the standard deviation > T_{std}, and $D_{avg}^{j} > D_{avg}$ and increase *k*: $k \to k+1$.
8. Merge two clusters, which satisfies the merging rules, i.e., if pair-wise distance > T_{dis} and number of objects in the new cluster < $T_{\max}$ and decrease *k*: $k \to k-1$.
9. Recalculate the entire cluster prototypes using the current partition.
10. Repeat from step 3 until the prototypes no longer move (i.e. $C_{w+1} - C_w \leq \varepsilon$) or the maximum number of iterations is reached.

Single-Linkage

Single-linkage also known as nearest neighbor is an agglomerative algorithm. This kind of algorithm starts with each data point in a separate cluster, and successively merges these clusters according to the proximity matrix and a predefined distance measure. The algorithm will then stop

when all the data points are in a single cluster or when some stopping-criteria is met. Given as set of *n* data points $[o_1, o_2, ..., o_n]$ one can define the proximity matrix as follows:

$$\begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n-1} & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n-1} & d_{2n} \\ d_{31} & d_{32} & \cdots & d_{3n-1} & d_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn-1} & d_{nn} \end{bmatrix}$$ where, $d_{ij} = dis(o_i, o_j)$ denotes the dissimilarity measure between o_i and o_j.

The hierarchical clustering uses the proximity matrix to determine relationships between objects. However, the diagonal entries of the proximity matrix are ignored since all data points are assumed to have the same degree of proximity to themselves $d_{ij} = 0, \forall i = j$ and all proximity matrices are assumed to be symmetric $d_{ij} = d_{ji}$. Therefore, the above matrix will be converted to a lower-triangle matrix: $\begin{bmatrix} 0 & & & & \\ d_{21} & 0 & & & \\ d_{31} & d_{32} & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d_{n1} & d_{n2} & \cdots & d_{nn-1} & 0 \end{bmatrix}$.

The single-linkage algorithms can now be summarized by the following steps:

1. Begin with n singleton clusters having level $L(m) = 0$ and sequence number m = 0. Calculate the proximity matrix for the n clusters.
2. Find the cluster pair with the minimal distance $d_{rs} = \min d_{ij}$ in the proximity matrix.
3. Increment the sequence number: m = m +1. Merge the clusters corresponding to r and s into a single cluster to form the next cluster m. Set the level of this cluster to $L(m) = d_{rs}$.
4. Update the proximity matrix, by removing the rows and columns corresponding to clusters r and s and adding a new row and column corresponding to the newly formed cluster. The proximity between the new cluster and the old cluster is defined in this way: $d_{rs} = \min(dis(o_{ri}, o_{sj}))$.
5. Repeat steps 2 and 3 until all data points are in the same cluster.

Complete-Linkage

The complete linkage clustering algorithm or also called farthest neighbor, is the opposite of single linkage. Its algorithm steps are similar to the steps of the single-linkage with the distinct difference that the complete-linkage defines the new proximity in step 4 as the distance between the most distant pair of data points, $d_{rs} = \max\left(dis\left(o_{ri}, o_{sj}\right)\right)$.

Average-linkage

The average-linkage or also known as un-weighted pair-group (UPGMA) using arithmetic averages is similar to the single-linkage except that it defines the new proximity in step 4 as the mean distance between all possible pairs of data points in the two clusters. This can be described in a formula as: $d_{rs} = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dis\left(o_{ri}, o_{sj}\right)$.

Multi Clustering algorithm

Multi clustering algorithm is a clustering algorithm used by Prime Vision. This algorithm is an agglomerative-based algorithm. It first removes all the data points that are bigger or smaller than a certain threshold value. The remaining data points are then assigned to a single cluster. Next all data points that satisfy some predefined distance threshold are merged together. This process is repeated iteratively until all the data points are merged. Finally, all the obtained clusters are tested on some feature (e.g., size) and only the clusters that survive this test are added to the final cluster collection.

Justification clustering choice

The large number of existing clustering algorithms makes it hard to select a suitable algorithm. Therefore, the clustering algorithms selected for this project are based on the following characteristics:

1. Commonly used in different areas;
2. Already used for the problem of address-block segmentation;
3. Implementation complexity;
4. Literature availability; and
5. Run-time complexity.

The taxonomy of the clustering algorithm discussed in chapter 3 was used to select candidate algorithms. The commonly used algorithms of each algorithm class (i.e., bottom-up, top-down, squared error, etc) were selected as a candidate algorithm. Based on these, candidate algorithms were rated (see table 4-2) and the ones with the highest scores were finally selected.

Table 4-2. Candidate clustering algorithms score.

Algorithms	**Char. 1**	**Char. 2**	**Char. 3**	**Char. 4**	**Char. 5**
EM-algorithm	high	no	average	good	low
K-means	high	yes [e.g. 6]	good	good	low
Agglomerative-algorithms	high	yes [e.g. 6]	average	good	high
chameleon	average	no	bad	bad	high
Min-Span-Tree	low	no	bad	bad	high
SOFMs	average	no	bad	average	average
Birch	low	no	bad	bad	low
Mean-shift	low	no	average	average	average

According to the scores in table 4-2 we selected: EM-algorithm, k-means, single-linkage, and complete-linkage. One common drawback of the k-means is that it requires the number of clusters a pre-knowledge. Therefore, we decided also to implement ISO-Data in order to see whether this algorithm will help overcome this drawback. We decide to implement only the most common agglomerative-algorithms, i.e. single-linkage, average-linkage, and complete-linkage. Moreover, Multi-clustering was also selected, because we want to compare its results with the other algorithms.

4.1.7 Ensemble generation function

Generation function refers to diversity strategy used to generate base clusters. As already mentioned there are several ways to obtain this diversity. In order to evaluate as much diversities as possible we will implement the following diversity strategies:

- Using different clustering algorithms;
- Using different initialization or other parameters for one clustering algorithm;
- Using different subsets of features to represent a pattern;
- Using different subsets of the original data; and
- Using different order of data set.

It is important to note that this diversity will be obtained using different single experiments. Therefore at least two single experiments are needed to ensure diversity.

4.1.8 Ensemble integration functions

The integration function is the task of combining the base clusters, which are obtained at the generation stage. This task can be viewed as a clustering task itself. Typically, each cluster in the combination is represented as a set of labels assigned by a (single) clustering algorithm. Given a collection of base clusters that have been generated using one of the generation methods mentioned above, a suitable integration function $f : \Pi = \{\pi_1, \pi_2, ..., \pi_r\} \rightarrow \sigma$ is needed in order to combine this base clustering and find a final consensus cluster $\sigma = \{C_1, C_2, ..., C_M\}$. A consensus function utilizes information from the base clustering $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$ to find this final consensus cluster σ. Based on this, several integration function have been designed for different problem domains. Selecting a suitable integration function is very important for the accuracy of the clustering results, but hard to realize. Unfortunately, not all the integration functions discussed in chapter 3 are well documented in literature. For this reason and because of the time limitation, only the integration functions that were well documented in literature and are realizable within the time offered for this project; were selected to be implemented. The integration functions selected for this project are: information theory based, co-association based, and overlapping based. This last one is designed based on some

base clusters analysis of several experiments. Each of these functions will be explained separately below.

Information theory based

Using the assumption of independence of base clusters, mutual information can be formulated as the sum or pair-wise mutual information between the target and the given base clustering. Let $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$ denote a set of r base clusters. The optimal final consensus cluster should share the most information with the original base clusters. In information theory, mutual information is a symmetric measure to quantify the statistical information shared between two distributions. Let A and B be the random variables described by the cluster labeling π_a and π_b, with k_a and k_b groups respectively. Let $H(A)$ and $H(B)$ denote the entropy of A and B respectively. The mutual information between A and B, denoted as $I(A,B)$ can be defined as follows [42, 45]:

$$I(A,B) = \frac{H(A) + H(B)}{2}.$$

Based on this, a [0,1]-normalized version of the mutual information can be defined as [42, 45]:

$$NI(A,B) = \frac{2I(A,B)}{H(A) + H(B)}.$$

Using $I(A,B) = \sum_{a \in A} \sum_{b \in B} p(a,b) \log \frac{p(a,b)}{p(a).p(b)}$ a normalized mutual information criterion $\phi^{(NMI)}$ is defined as: $\phi^{(NMI)}(\pi_a, \pi_b) = \frac{2}{n} \sum_{i=1}^{k_a} \sum_{j=1}^{k_b} n_i^j \log_{k_a * k_b} \left(\frac{n_i^j * n}{n^j * n_i} \right)$, where n is the number of data points, n_i is the number of data points in cluster C_i according to π_a, n^j is number of data points in cluster C_j according to π_b, and n_i^j is the number of data points that are in cluster C_i according to π_a as well as in cluster C_j according to π_b.

The Average Normalized Mutual information (*ANMI*) between as set of *r* base clusters Π, and a base cluster $\tilde{\pi}$ is defined as follows [42,45]:

$$\phi^{(NMI)}(\Pi, \tilde{\pi}) = \frac{2}{r} \sum_{i=1}^{r} \phi^{(NMI)}(\tilde{\pi}, \pi_i).$$

Accordingly, the optimal consensus clustering π_{k-opt} should be defined as the one that has average mutual information with all individual base clusters $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$ given desired *k*. Thus the

integration function for this project is Average Normalized Mutual Information (*ANMI*). Then, π_{k-opt} is defined as:

$$\pi_{k-opt} = \arg\max_{\tilde{\pi}} \sum_{i=1}^{r} \phi^{(NMI)}(\tilde{\pi}, \pi_i)$$, where $\tilde{\pi}$ goes through all possible k-clusters.

Co-association based

The Co-association based integration function operates on the co-association matrix. This matrix contains the strength of co-association of two object pairs. The similarity between objects can be estimated by the number of clusters shared by two objects in all the cluster of an ensemble (base clusters). Given a set of *r* base clusters $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$, an nxn co-association matrix *Co*, whose $(i, j)^{th}$ entry is given by:

$$Co_{ij} = S(o_i, o_j) = \frac{1}{R} \sum_{k=1}^{H} \delta(\pi_k(o_i), \pi_k(o_j)), \text{ where } \delta(x, y) \equiv \begin{cases} 1, \text{ if } x = y \\ 0, \text{ if } x \neq y \end{cases}.$$

Intuitively, C_{ij} measures how many times o_i and o_j share the same cluster in the clustering ensemble. This matrix can be viewed as a new proximity matrix, which is superior to the original proximity matrix. In order to get the final consensus cluster, one can apply one of the similarity based clustering algorithms to the obtained co-association matrix. In our case we will use one of the similarity based clustering algorithms (i.e., single-linkage, complete-linkage, and average-linkage) that we plan to implement for this project.

Overlapping Based

The overlapping based integration function is our own integration function, which is based on overlapping of cluster in the base clusters. This simple integration function iterates through all base clusters. All the cluster pairs that satisfy the overlapping threshold are merged together, while the remaining clusters are added as a single cluster. The result of this operation is then returned as the final consensus clustering. The overlapping rate can be computed using a Jaccard's coefficient:

$$Overlaping(C_i, C_j) = \frac{n_i^j}{n_i^j + n_i + n_j}$$, where n_i^j denotes the number of data points that are in C_i as well as in C_j, n_i is the number of data points that are only in C_i, but not in C_j, and n_j is the number of data points that are only in C_j, but not in C_i.

For a given set of r base clusters $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$, this integration function can be summarized as follows:

1. Add each cluster in base cluster as a single cluster to the consensus cluster $\sigma = \{C_1, C_1, ..., C_m\}$, i.e., all the cluster of π_k, $k = 1,2,...r$.
2. Find cluster pair $C_i \in \sigma$ and $C_j \in \sigma$ that share a certain percentage of data point, i.e. $Overlaping(C_i, C_j) \geq T$.
3. Merge cluster C_i and C_j together.
4. Repeat until all cluster in σ are examined.

4.1.9 Results evaluation

This component evaluates the results of the proposed system. In order to do that, we need to evaluate the clustering results, the time each clustering algorithm takes to obtain the clustering results, and the address-block selection evaluation, i.e., this is needed to check whether the classifier was able to classify the destination address-block among the obtained clustering results.

Clusters validation

This is a measure that indicates how good some clustering's results are. As already mentioned, there are three types of validation that can be distinguished: external indices, internal indices and relative indices. The external type uses some pre-defined classification as a standard to validate the clustering solutions, while the other two types examine the clustering solutions directly from the original data. Because of the availability of ground-truth segmentation (true clustering) of each mail piece image, external indices seem to promise a better validation result than the others. They use predefined data instead of only some examination of the original data. This has motivated us to implement an external cluster validation for our project. In this case, we would like to be able to compare the true clustering and the clustering produced by a certain algorithm. There are numerous methods to obtain this. Mainly these methods are based on error measure (difference measure), point pair counting, variation of information, or on statistic measure (data-points

correlation). Generally all criteria for comparing clusters can be computed given the so-called confusion matrix. Assume that we have two clusters $C = \{C_1, C_1, ..., C_k\}$ and $C' = \{C'_1, C'_1, ..., C'_l\}$. The confusion matrix M is a *kxl* matrix, whose $(i, j)^{th}$ element is the number of points in the intersection of clusters C and C', i.e. $m_{ij} = \left| C_i \cap C'_j \right|$, $i = 1,..., k$ *and* $j = 1,...l$. An important class of criteria for comparing clusters is based on counting the pairs of points on which two clusters agree or disagree. Each pair of data points falls in one of the four categories labeled as N_{11}, N_{10}, N_{01}, and N_{00}. The category N_{11} and N_{00} represents the number of simultaneous absence or presence of the pairs of points in the same cluster in both C and C'. While, the categories N_{10} and N_{01} count the pairs of data points present only in the same cluster C or C'. All four categories can be obtained from the confusion matrix:

$N_{11} = \frac{1}{2}\left[\sum_{i,j} m_{ij}^2 - m\right]$, where $m = \sum_i m_i$, m_i = size of a certain cluster C.

$N_{10} = \sum_{i=1}^{l}\sum_{j=1}^{k}\sum_{k=j+1}^{k} m_{ij} m_{ik}$.

$N_{01} = \sum_{i=1}^{k}\sum_{j=1}^{l}\sum_{k=j+1}^{l} m_{ji} m_{ki}$.

$N_{00} = N - N_{11} - N_{01} - N_{10}$, where N is the total number of point pairs.

The proportion of point pairs on which the two clusters agree can be given by the Jaccard index as:

$$\delta(C, C') = \frac{N_{11}}{N_{11} + N_{10} + N_{01}} .$$

To evaluate our clustering results we will make use of the method described above.

Run-time

This indicates how long a certain clustering-algorithm needs to cluster a given data set. The time needed for clustering will be used to compute the run-time of each clustering algorithm.

Address-block selection validation

This indicates whether the Prime Vision classifier was able to select the correct destination address block. For this, the results of the clustering algorithm will be sent to the classifier, which will return selection results. These results will then be compared with the ground-truth

segmentation of the image. For this Jaccard index will also be used to compute the proportion of points on which the two clusters agree.

4.1.10 Address-block selection (Classifier)

Among the entire segmented clusters only one cluster corresponds to the DAB. This component is used to select among all the segmented clusters the cluster that has the highest confidence value to represent the DAB cluster. In order to do that, each cluster is evaluated and confidence value, which is a measure of the degree that a particular cluster is the DAB, is assigned to it. The cluster with the highest confidence score is assumed to be the DAB.

Several authors used different methods to achieve the DAB cluster. These methods can be classified as:

- **Heuristic based methods:** These methods uses heuristic rules, like DAB should have a least 10 objects in it or its area should be bigger than some threshold.
- **Geometric based methods:** These methods are based on geometric features, like the cluster position, cluster width, height, etc.
- **Artificial intelligence based methods:** These methods use, for example, statistical information of the geometric features to train a neural network and this neural network is then used to evaluate clusters and select the DAB cluster.

A classifier based on the last approach has already been implemented for the existing system. Its process is as follows: it extracts all kinds of features from the clusters and statistical information of these features is then used to achieve the DAB. This classifier will be used as the classifier for this project. However, it is important to note that this classifier has been trained for the Multi clustering algorithm only.

4.1.11 Output

This component represents the output of the system, which can be distinguished in clustering results and evaluation results. The clustering results will extract a simple and compact representation of the data set, i.e. a compact description of each cluster in terms of bounding boxes. To represent these clusters bounding boxes we will make use of the so-called debug mode of the

Prime Vision tool. Figure 4-5 shows an example of clustering results output with three clusters, denoted by bounding boxes.

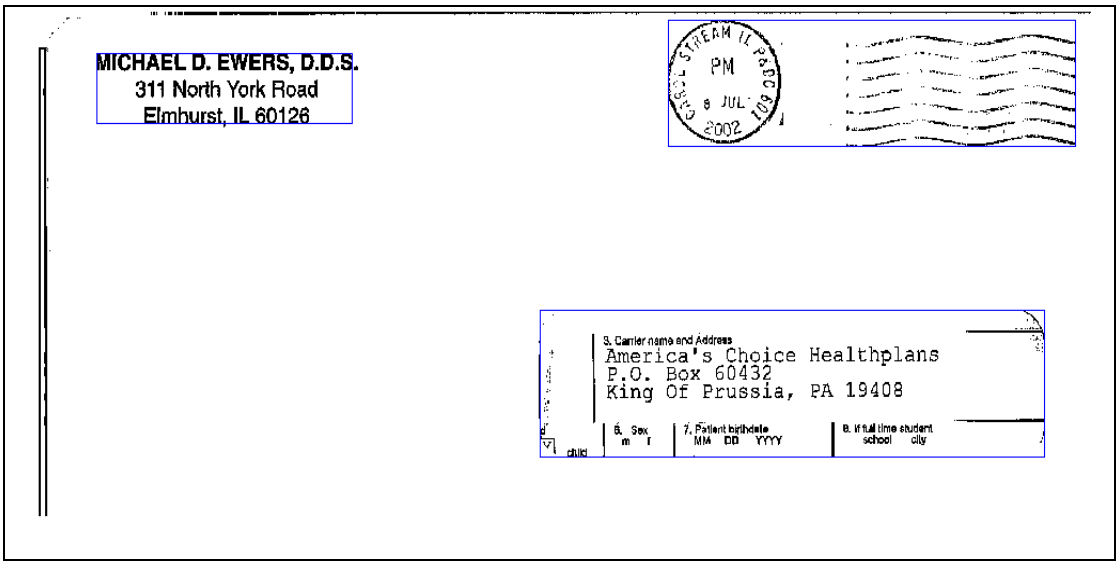


Figure 4-5: Example of clustering results.

The evaluation results will be stored as a comma separated file (CSV) which will contain information of the image to be segmented as well as the validation results. The attributes that are needed to be stored in the CSV-file are:

- Labeling Information, i.e., document name or ID and writing style.
- Validation information obtained at the evaluation results, i.e., clusters validation, run-time, and classifier validation.
- Segmentation experiment settings, i.e., experiment settings related to a certain validation.

4.2 FUNCTIONAL REQUIREMENTS

Functional requirements describe the interaction between the system and its users and external systems independent of its implementation. This section will discuss the functional requirements related to the proposed system. The requirement are gathered based on the system needs, which are discussed with the future user. Consequently, a list of functional requirements related to the proposed system can be summarized as follows:

- Give the user the ability to:
  - create, delete and modify address-block segmentation experiment with single clustering.
  - create, delete and modify address-block segmentation experiment with ensemble clustering.
  - run a segmentation experiment.
  - evaluate a segmentation experiment.
  - show segmentation results.
  - validate the results with the ground-truth segmentation.
  - choose different data features to use for clustering.
  - filter the dataset.
  - create random subsets of the data.
  - specify clustering algorithm's parameters.
- Provide the user with:
  - different single clustering to choose from.
  - different dissimilarity measure to choose from.
  - different integration functions to choose from.
  - user-friendly interfaces.

4.3 NONFUNCTIONAL REQUIREMENTS

Nonfunctional requirements describe user-level requirements that are not (directly) related to the functionality of the system. Likewise this requirements obtained after discussion with future user. The relevant nonfunctional requirements for the proposed system are presented in this section.

User interface and human factors

A user-friendly interface will be provided for the system users. The user-interface will be designed such that only the functionality that is needed is shown and any other functionality will be hidden or disabled. The level of expertise required for the users will be average computer knowledge. The proposed system will only be used by the Research and Development team of Prime Vision. Most of these users have high knowledge of computers.

Documentation

The development process of the system will be well documented in this report. This will provide programmers and users a better understanding of how the system will perform in various different ways.

Performance characteristics

The performance of the system relies on the existing functionality, which will be used as well as the new algorithms to be implemented. This means that the achieved response time and accuracy are determined by the efficiency of both the existing system and the proposed system. The typical and extreme load will depend on how many experiments are running.

Hardware consideration

The System will require a computer with at least Microsoft windows 2000 platform. To view the evaluation file Microsoft Excel or some text editor will be needed. The system is only compiled and tested on Microsoft windows 2000 platform. This means that the system may not function well on other platforms.

Error handling and extreme conditions

The system is designed to handle various kinds of user input errors by providing an error message box when invalid input is entered. The part that has an error on it will be made red. This will make it easy for the user to see which part has to be corrected.

Quality issues

Alpha testing will be performed in order to help produce a quality system that will meet all the functionalities. Beta testing will also help by gathering user information and feedback to increase the quality of the system.

System modifications

System documentation as well as source code comment will serve as a guideline for future modifications.

Physical environment

Practically, the overall address read system operates in the sorter center. But, because the proposed system is only for research purposes it will be used in-house at Prime Vision.

Security issues

Because the system is implemented within the prime-vision software a Dongle key will be required in order to run the system. A dongle is a piece of hardware that attaches to a computer in order to make a piece of secure software run.

Resource issues

The system requires a mail piece image for the segmentation part and a labeling file (ground truth segmentation) for the evaluation part. Also, several developed components of the Prime Vision software tool are required.

4.4 PSEUDO REQUIREMENTS

The existing software is written using Borland Delphi 7.0. Due to this all related software implemented for this thesis will also be written using the same programming language. Rational Rose Student Edition and Microsoft Vision 2003 are used for system modeling. For the result evaluation the Data Analysis tool of Microsoft Excel 2003 will be used. Finally, for the documentation Microsoft word 2003 is used.

Chapter 5: Proposed system implementation

This chapter describes the actual implementation of the system. The system is implemented in accordance with the design and algorithms specified in the previous chapter. First, the use case model of the system will be provided. Next, the UML class diagrams for the system will be illustrated with a briefly explanation of the core parts. Afterwards, the user interface will be discussed and some screenshots of it will be shown. Finally, an explanation of how the system was tested will be discussed.

5.1 USE CASE MODEL

A use case diagram describes the functional aspects of a system. More specifically, it describes the behavior of the system as seen from an actor's point of view. Actors are external entities (a person, database, hardware, etc…) that interact with the system. Figure 5-1 shows the use case diagram for the final workbench system as presented.

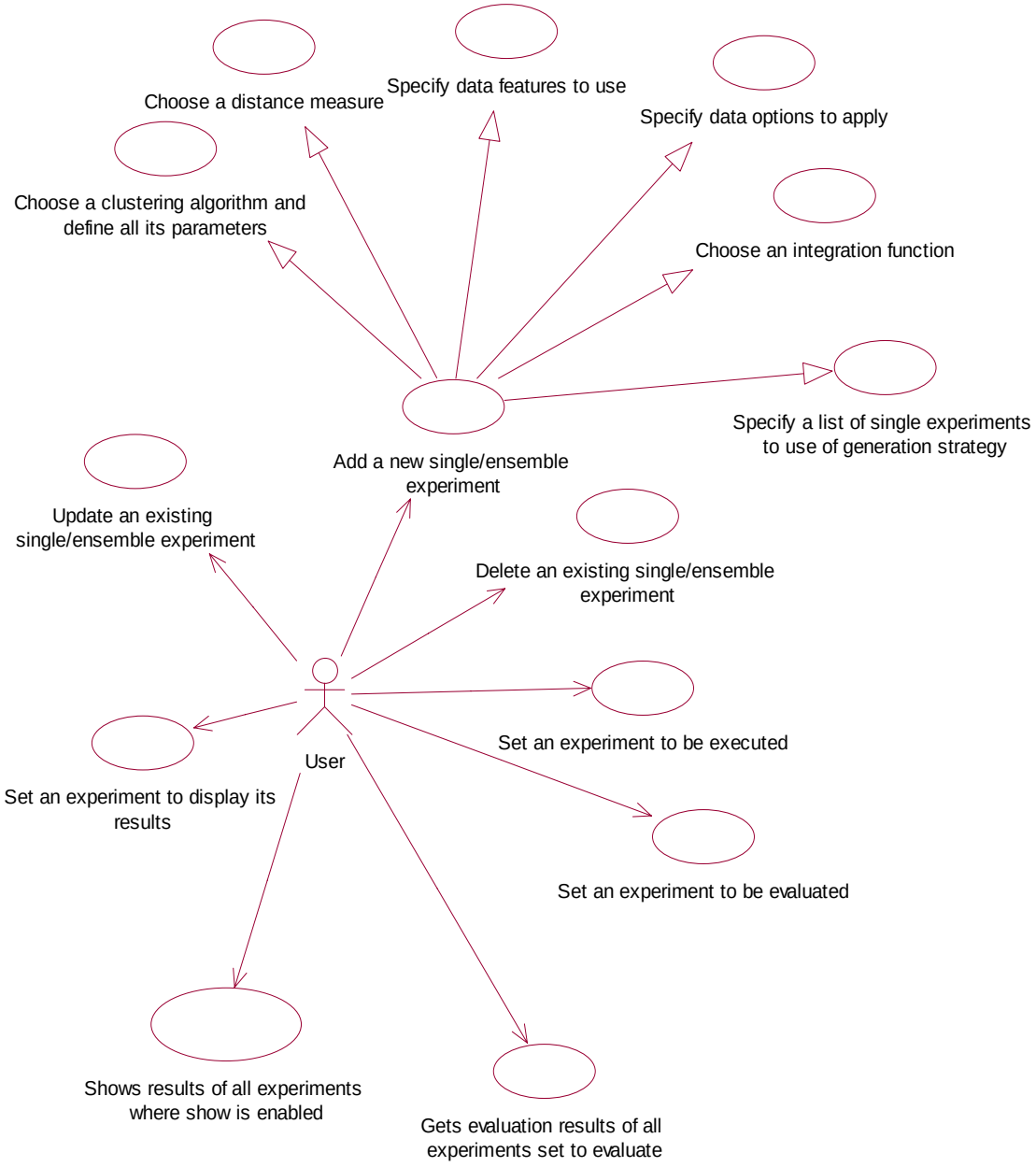


Figure 5-1: Use case diagrams.

5.2 CLASS DIAGRAMS

A class diagram is a type of structure, which identifies structures of a system by showing its classes, their attributes, methods, and relationships between the classes.

The class diagrams are based on the system specification from the company Prime Vision. The segmentation system is divided into six subsystems that provide different parts of the system functionality. During the implementation we tried to keep the implementation of each subsystem generic and as object oriented as possible. A general overview of the subsystem structure is shown in figure 5-2. The specific implementation details for every sub-system are described in separate sections. Whenever needed, the interaction between the classes is discussed as well. However, since there are many classes and each class consists of several attributes and methods, we will not discuss each class in details but rather a short description of each class will be given. A detailed explanation will be given in case the implementation differs significantly from the system as described in chapter 4.

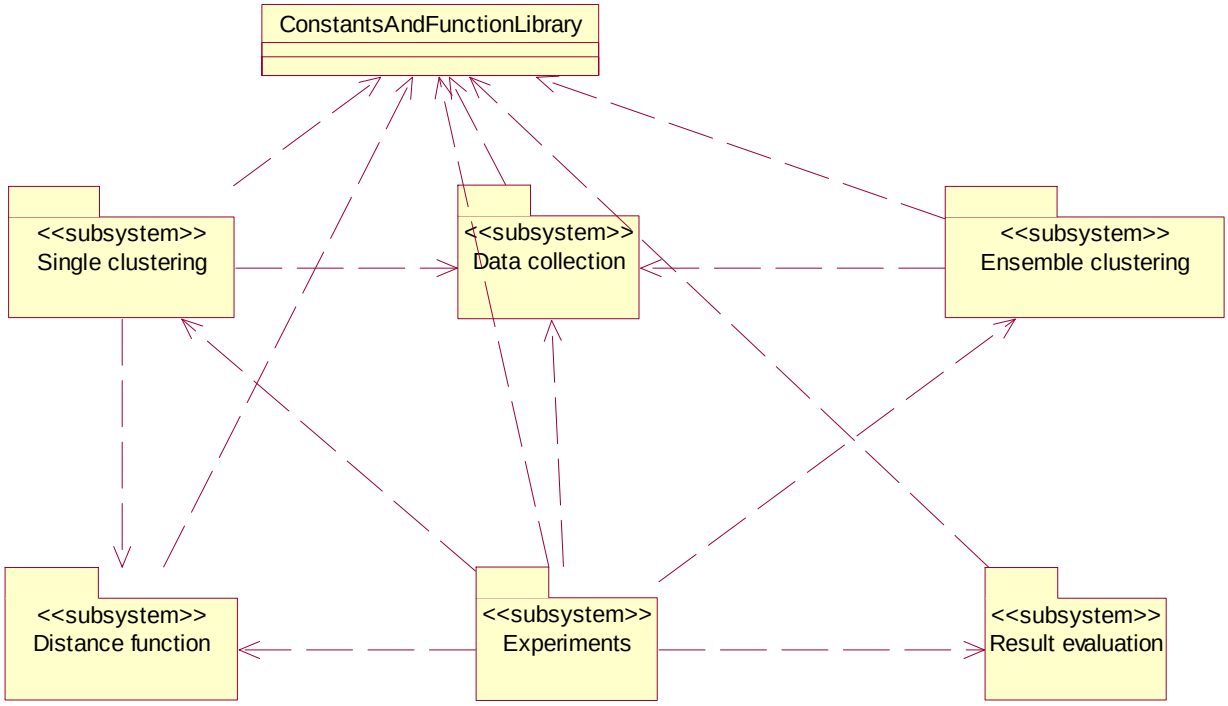


Figure 5-2: System structure of the implemented model.

5.2.1 Data collection subsystem

This subsystem was needed because we could not use the existing data structure. Therefore we implemented our own data structure. The main goal of this subsystem is to handle all the data

functionalities. The most data related functionality could be reached by calling methods from the classes of this system. For a better understanding, the specific implementation of each class member of this system will be described separately in this section. Figure 5-3 shows all the classes related to this system.

ConnComp

This is a record, which holds connected component features for each individual connected component of an image. This record is an existing feature, which is defined in the ConnComps class. However to simplify the model understanding it has been modeled within the class diagram as an individual class.

ConnComps

This class is an existing class, which contains a set of conncomp records, i.e. a set of connected component features for each connected component of the image. It contains also all functionality needed to handle this set of connected components, such as save features, get a certain feature, and so on.

FilterConncomps class

The main goal of this class is to exclude the non-relevant connected components, such as noise, lines etc. from the connected components set. This is what is mentioned in system design section 4.1.4 as feature selection. We mentioned also in the same section that this is an optional task. Therefore we implemented this in such a way that the user can choose to apply or not apply this task. When the user chooses to apply this task a new sub set is created from the original one by excluding all the non-relevant connected-components. Otherwise the original set of connect components is returned without any change. For the excluding operation we used the rules discussed in the design section 4.1.4. The threshold for each rule is defined based on some experimenting.

DataObject class

This class is used to create a data object from a connect component. One of the important features of this data object is the connected component feature subset. This feature subset is a sub

set of the original connected component's feature set. The goal of this is to allow the user to define which connected component's features to use for the clustering process. For this we used a feature weight vector. This vector is filled in accordance to the user's choice. The feature subset is then obtained by multiplying the chosen feature with its associated weight and excluding all the features that are not chosen.

DataCollection class

This is one of the important classes. It represents actually a collection of data objects. This collection can be filled using a set of connected components obtained from the *"FilterConncomps"* class. In this case a data object is first created for each connected component and then it is added to the collection. However, the data collection can also be filled with already created data objects. This class is implemented such that it can represent a set of data objects related to the input or to a certain cluster (a cluster is actually a sub set of input data). The class provides several functionalities need to add, remove, and show its data object members. Each data collection contains a reference data object, which is needed to support the cluster prototypes required for some clustering algorithms. It contains also a bounding box, which represents data objects belonging together in the form of rectangle. The coordinates of the bounding box are defined according to the most top, bottom, left, and right data object. The mean data object is also one of the important feature of each data collection. To compute the mean, a virtual object is first created with the average features. The data object that has the features closely to the virtual object's features is then picketed up as the mean object.

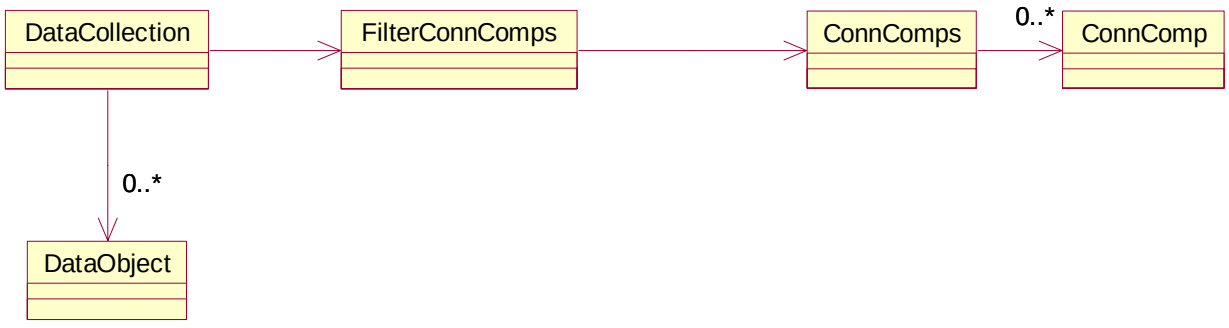


Figure 5-3: Data collection subsystem.

5.2.2 Distance function subsystem

The main goal of this subsystem is to compute the distance between the feature subset of two data objects. All the distance functions discussed in chapter 4 section 4.1.5 were implemented in order to achieve this. The distance function 'OldDistanceFunction', which is not mentioned in section 4.1.5 is also implemented. With this distance function we tried to simulate the distance function currently used by Prime Vision. The class diagram for this subsystem is shown below in figure 5-4.

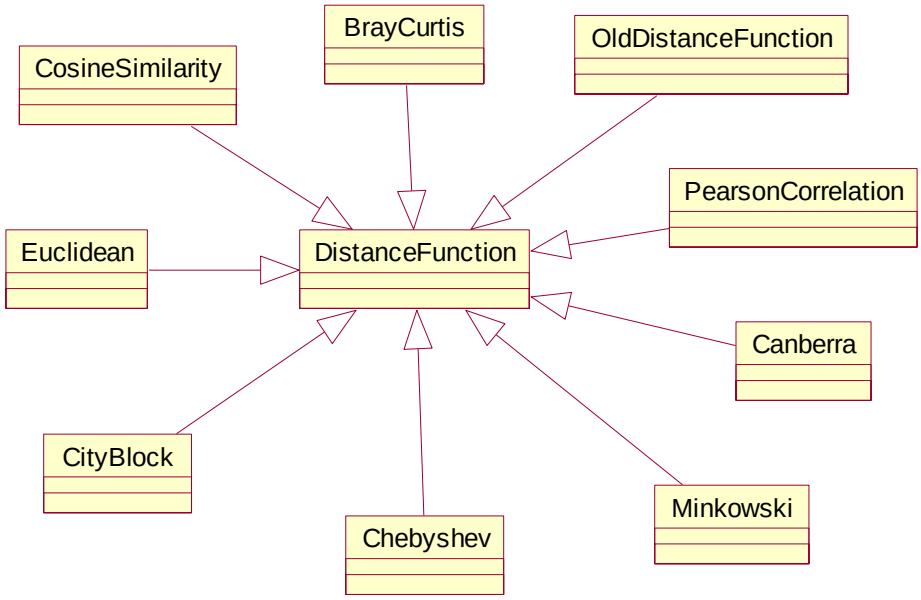


Figure 5-4: Distance function subsystem.

5.2.3 Single clustering subsystem

This subsystem, as shown in figure 5-5, provides a set of single clustering algorithms discussed in section 4.1.6. Almost all the algorithms are implemented as they are discussed. However, there is an additional change made to the initialization step of the partitional clustering algorithms (i.e., K-means, EM, and ISO-Data). We implemented for this three methods, namely:

- **Regular random:** This is method selects k data objects randomly from the data collection as its k-prototypes. Where k is the number of clusters defined by the user.

However, one problem we had with this method is that the clustering results are not reproducible. This makes it difficult to evaluate the clustering results.

- **Offset method:** This method was implemented overcome the problem we had with reproducibility of clustering results. First, it computes the offset, which is the number of data objects divided by the number of clusters defined by the user. Based on this offset it selects every offset a data object as candidate prototype until the number of clusters is reached. For example, if we have data collection D ={1,2,3,4,5,6,7,8,9,10, 11,12} with 12 data objects and our number of clusters is 3 then the offset is 12/3=4. This means that the 1, 5, and 9 will be selected as the candidate prototypes.
- **Balanced method:** This method is almost the same as offset, except that its offset is divided by two and it starts with the median data object of the data collection. Based on this it interactively selects once a data object left of the median object and once right of the median data object. For the above example it means that the offset will became 4/2 = 2 and the selected data objects will be 6, 4, and 8.

All these methods can be chosen by the user. A clustering algorithm gets a data collection and a distance function. With this input it uses its clustering strategy to group data object into similar clusters. Each cluster is represented by a data collection and a vector of data collections represents all the clusters. The clustering algorithms provide also the functionality to use clusters from other clustering algorithms instead of data collection. For example, one can apply k-means algorithms to data collection and then apply single-linkage algorithms to the clustering results obtained by the k-means. Summarized all the clustering algorithms mentioned in chapter 4 were implemented. The multi clustering algorithm is also modified so that it can be used in this project.

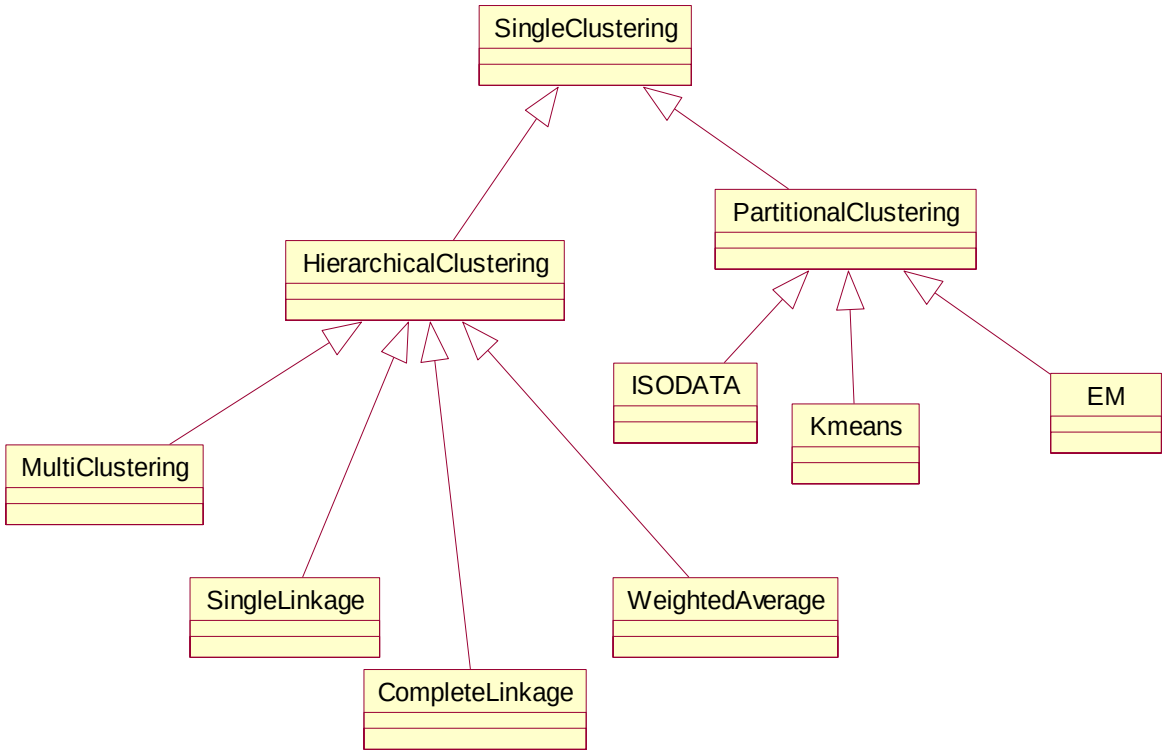


Figure 5-5: Single clustering subsystem.

5.2.4 Ensemble clustering subsystem

This subsystem, as shown in figure 5-6, is the implementation result of sections 4.1.7 and 4.1.8. It works as follows: the ensemble generation uses one or more single clustering algorithms to generate base clusters. Those base clusters are then integrated to final consensus cluster by the ensemble integration.

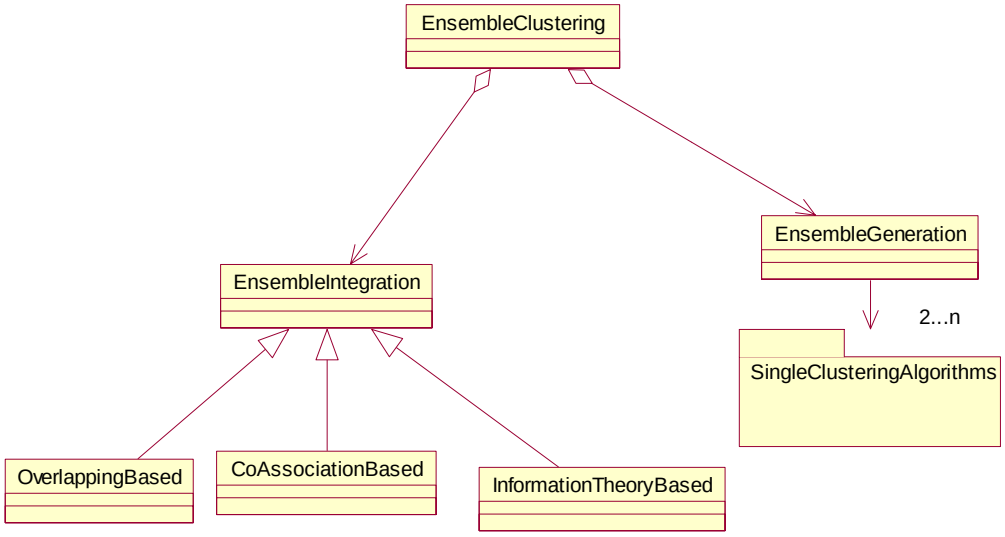


Figure 5-6: Ensemble clustering subsystem.

All the diversity strategies discussed in 4.1.7 are implemented. An enumeration of all these generation functions implemented with an explanation on each is listed below:

- **Using different clustering algorithms**: We implemented the system such that the user can define which algorithm to use for a certain segmentation experiment. For instance defining two segmentation experiments one with k-means and another with multi clustering algorithm. The clustering results can then be used to generate base clusters.
- **Using different initialization parameters for one clustering algorithm:** The algorithms are implemented such that the user can modify or use the default initialization of the algorithm's parameters. The parameters 'number of cluster', 'min objects in a cluster', and ' max objects in a cluster' can also be defined as a certain percentage of the entire number of data objects. This strategy makes it possible to define two segmentation experiments with the same clustering algorithm but different clustering algorithm's parameters.
- **Using different subset of features to represent a pattern:** The system is implemented such that the user is able to define for each segmentation experiment which features to use.

- **Using different subsets of the original data:** We implement for this diversity strategy three methods, namely: Data inside boundingbox, random data with offset, and random data. The first method 'Data inside boundingbox' makes it possible to use only data objects inside a certain bounding box defined by the user. The second method uses an offset defined by the user as iteration-steps and selects iteratively the data objects to be used. This method is almost like the initialization method of the partitional clustering algorithms, except that the offset is entered by the user. The last method selects randomly data objects to be used without any prior knowledge. However, a problem with this algorithm is that its results are not reproducible.
- **Using different order of data set:** In order to test whether diversity can be obtained by ordering data objects differently two methods are implemented, namely order by center of gravity for x (xcent) and y-coordinate(ycent).

Furthermore, we implemented all the integration functions as described in section 4.1.8. We chose the threshold of the overlapping integration function after several experiments $T = 0.8$. In order to create a co-association matrix for the co-association based method we tried first the regular method (as mentioned in section 4.1.8), which fills the nxn co-association matrix by counting for each data objects the number of its appearance. However, this is a real time-consuming process. We achieved better results with a method similar to the method used in [42] for hyper-graph methods. This method works as follows: First the set of base clusters $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$ is mapped into a suitable matrix representation. For each $\pi_i \in D^n$ an nxk_i matrix $H_i \in D^{nxk_i}$ is extracted. Where *n* is the number of data objects and k_i is the number of clusters in the i^{th} base cluster. The entries of *H* represent the binary membership of data objects for each in the base. Suppose a set of 3 base clusters and 5 data objects the matrix *H* can then be defined as follows:

$$\begin{vmatrix} & \pi_1 & \pi_2 & \pi_3 \\ o_1 & 1 & 2 & 2 \\ o_2 & 1 & 2 & 2 \\ o_3 & 2 & 3 & 1 \\ o_4 & 2 & 3 & 1 \\ o_5 & 3 & 1 & 1 \end{vmatrix} \Leftrightarrow \begin{vmatrix} & H_1 & H_2 & H_3 & H_4 & H_5 & H_6 & H_7 & H_8 \\ o_1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ o_2 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ o_3 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ o_4 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ o_5 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{vmatrix}$$

Where π_1 consist of three clusters associated with H_1, H_2 and H_3. The base cluster π_2 consist also of three clusters that represented by H_4, H_5 and H_6. Finally π_3 consist of two clusters denoted by H_7 and H_8. An nxn co-association matrix *Co* can now be computed by multiplying the matrix H with its transpose $Co = HH^T$. The hierarchical clustering algorithms that may be used for the integration process use all dissimilarity instead of similarity. Therefore each entry of the matrix is subtracted from the number of data objects to obtain dissimilarity. However, the diagonal entries of the matrix are ignored since the same object cannot appear as a pair data object. The data objects appearance is also assumed to be symmetric. Therefore a lower-triangle matrix will be used instead of a full matrix. Finally for integration process we applied a single-linkage algorithm because of its good results on the matrix. For the information theory based integration we computed the *ANMI* between all base clusters. The base cluster that shares the most information with the other base clusters was then shown as the final consensus cluster.

5.2.5 Result evaluation subsystem

This subsystem provides the functionality to evaluate both single and ensemble clustering algorithms. Its implementation is based on section 4.1.9. During the implementation an additional change has been made to the clusters validation method. Instead of the number of object we used the area of the object. The reason for this change is to make small objects (like noise) have less influence than the core objects. In order to validate the classifier's results we modified the existing classifier such that it can be used for our project. A brief description of the evaluation process is as follows: the evaluation class gets the clustering results, the run-time of the clustering algorithm and the ground-truth segmentation (DAB-cluster). Based on this it computes the proportion of points on which the DAB-cluster agree with the clustering results. The cluster with the highest proportion is then chosen as the cluster closest to DAB-cluster. Next, the clustering results are sent to the

classifier to obtain the cluster which is classified as DAB. Finally in order to see whether the classified cluster is the cluster closest to the DAB they are compared with each other. The classes related to this subsystem are shown in figure 5-7.

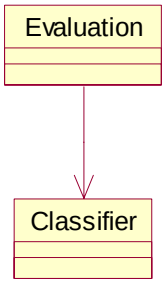


Figure 5-7: Result evaluation subsystem.

5.2.6 Experiments subsystem

The main goal of this subsystem is to provide the user with the functionality needed to create segmentation experiment with single or ensemble clustering. Based on user interaction this subsystem uses the other subsystems to create segmentation experiment. Except for the mainToolGUI all the classes of this subsystem are new classes. The mainToolGUI class is an existing functionality, which provides the user, via a GUI the functionality to input data, see the output results, and switch to the experiments frame. However, some functionality needed for this project has been added to this class. The experimentsFrameUnit class is the main GUI for the segmentation system. It uses the other classes to provide the user with all the functionality needed for the segmentation experiments, such as add, delete, edit, save the experiment settings, and so on. To define a segmentation experiment it presents the user with a form (i.e. a SingleExperimentSettingsFormGUI in case of single experiment and EnsembleExperimentSettingsFormGUI in case of ensemble experiment), which allows the user to define all the settings needed for a certain segmentation experiment. Within the single experiment form the user can define the data option to use (i.e. filter data, subset method, and data order), define the features, define the distance function, and define the clustering algorithm and all its parameters. The ensemble experiment form allows the user to select a list of predefined single experiments for the generation process and an integration function to be used for the integration

process. Once an experiment is defined its settings are saved to text file and added to an experiment settings list. A control-panel is also created for each defined experiment in order to allow the user to control the experiment. The experiment settings list is then used to create the segmentation experiments. Note that only the experiments that are set to run (i.e. the run-checkbox is checked) will be created and applied for the segmentation process. The results of each segmentation experiment will then be sent to the evaluation subsystem for evaluation. Finally, both the segmentation results and the evaluation results are provided. A class diagram with all classes related to this subsystem is shown in figure 5-8.

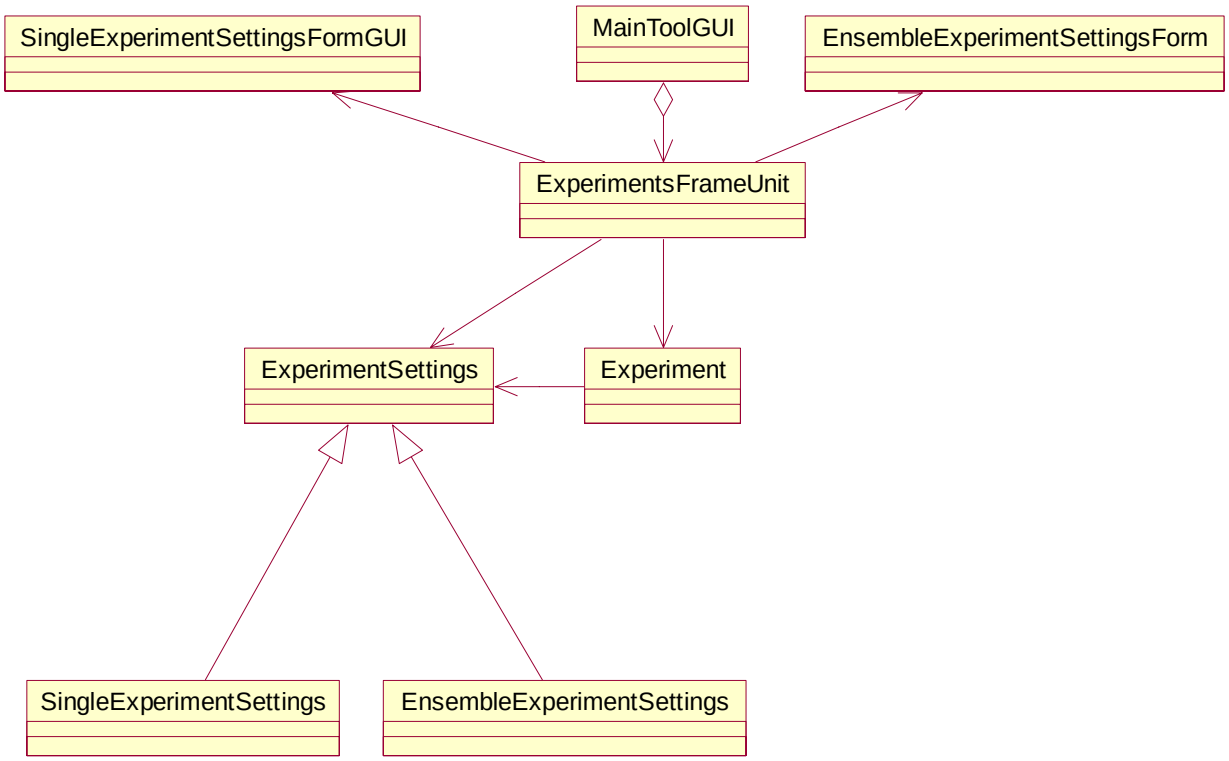


Figure 5-8: Experiments subsystem.

5.3 USER INTERFACE

One of the important aspects that determine the user's acceptance of a certain software product is the design of its graphical user interface (GUI). Generally, this GUI affects the amount of effort and time the user needs to accomplish a certain task. Moreover, the end user may refuse a

good software product with great features because of its complex or bad GUI. Therefore, a common approach used within GUI design is prototyping, which is an iterative process between the designer and the end user. The end user evaluates each design presented by the designer, and based on the end user's feedback the design is modified until an agreement is reached. However, this is a very time-consuming process and since the proposed system is only for research purposes we decides not to go through this time-consuming process. Our main focus lies on the system implementation. Therefore a different approach of designing the user interface was used. First we analyzed the existing user-interface according to the functional requirements. Based on this analysis we discovered that it will take a lot of time to design the user interface from scratch. Therefore, we decide to use the existing functionality supported by the existing GUI and add only those functionalities that are not supported yet. For these new functionalities we analyzed the functional requirements of the system. Since the users are familiar with the existing GUI, we tried to keep user interaction needed for the new functionalities as close as possible to the existing one. Moreover, during the design we tried to organize the user interface in a meaningful and useful ways based on clear and consistent models that are recognizable to the users. For example, the related things were putted together using group-boxes. In order not to confuse the user, we also tried to show only the models that are needed and hide or disable any other model (functionality). For example, if the user want to add a single experiment with K-means clustering algorithm, only the models (such as input boxes related to k-means parameters, etc.) related to this will be shown. The GUI consists of different sub-interfaces in form of frames and forms of which the user can switch according to the task to be performed. Each of these sub-interfaces is related to some functionality. This section attempts to give a brief description of all the sub-interfaces that are relevant for this project.

5.3.1 Source area

This sub-interface is an existing functionality, which makes it possible to see the original version of the loaded source image. It contains also a control sub-interface (as shown in figure 5-9), which allows the user to perform different tasks. Note that this control sub-interface consists almost in all the other sub-interfaces. Therefore, it will only be explained within this sub-interface. Both

sub-interface as shown in figure 5-9 contains several functionalities, but we will only explain those functionalities that are in scope of this project. Table 5-1 gives an explanation of the screenshot.

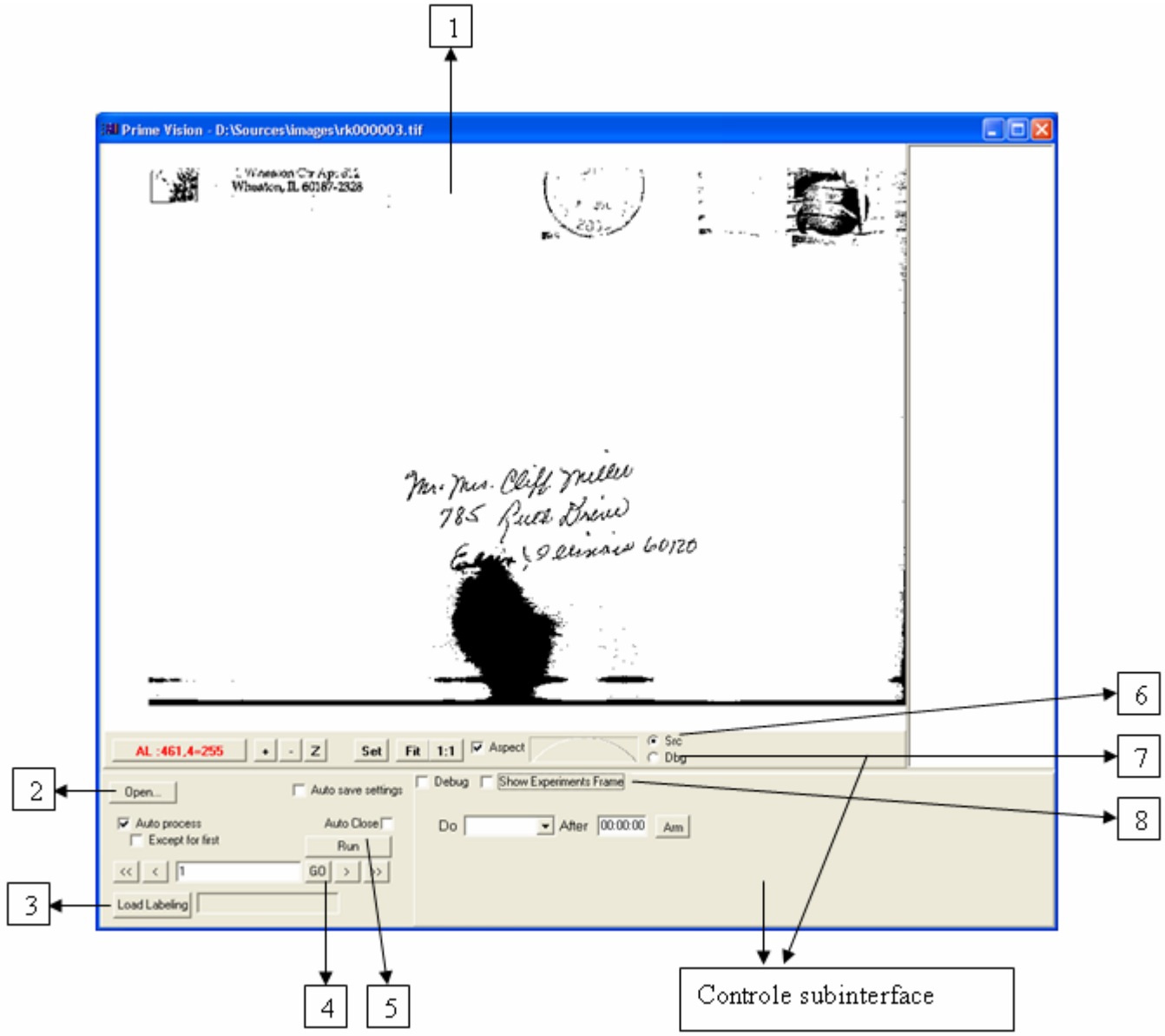


Figure 5-9: Interface of the source area.

Table 5-1. Explanation of the source area interface.

Item number	Explanation
1	This area is used to display the source image.
2	With this button the user can load a single image, an image list, or database file containing images.
3	This button can be used to load a labeling file associated with a set of images.
4	The “go” button is used to apply a certain task to the current displayed image. In case of this project this means applying the segmentation experiment(s) defined by the user on the image.
5	This button is the same as the “go” except that it applies the same task to a list of images instead to only the current image. These have been used during the experimenting stage to apply a certain experiment to an image database/ image list.
6	This can be used to switch back from debug mode to the source area.
7	Likewise, this can be used to switch from source mode to debug mode.
8	Finally, this has been added the interface to allow the user to show or hide the experiment sub-interface.

5.3.2 Debug mode Area

This sub-interface as shown in figure 5-10 is also an existing functionality. Its main purpose is to allow the user to see the preprocessed image in the so-called debug mode. This debug mode provides the user with detailed graphical representation of internal data during processing. However, since not all the available functionalities are relevant for this project we will only point out the functionalities that are relevant for this project. This is summarized in table 5-2.

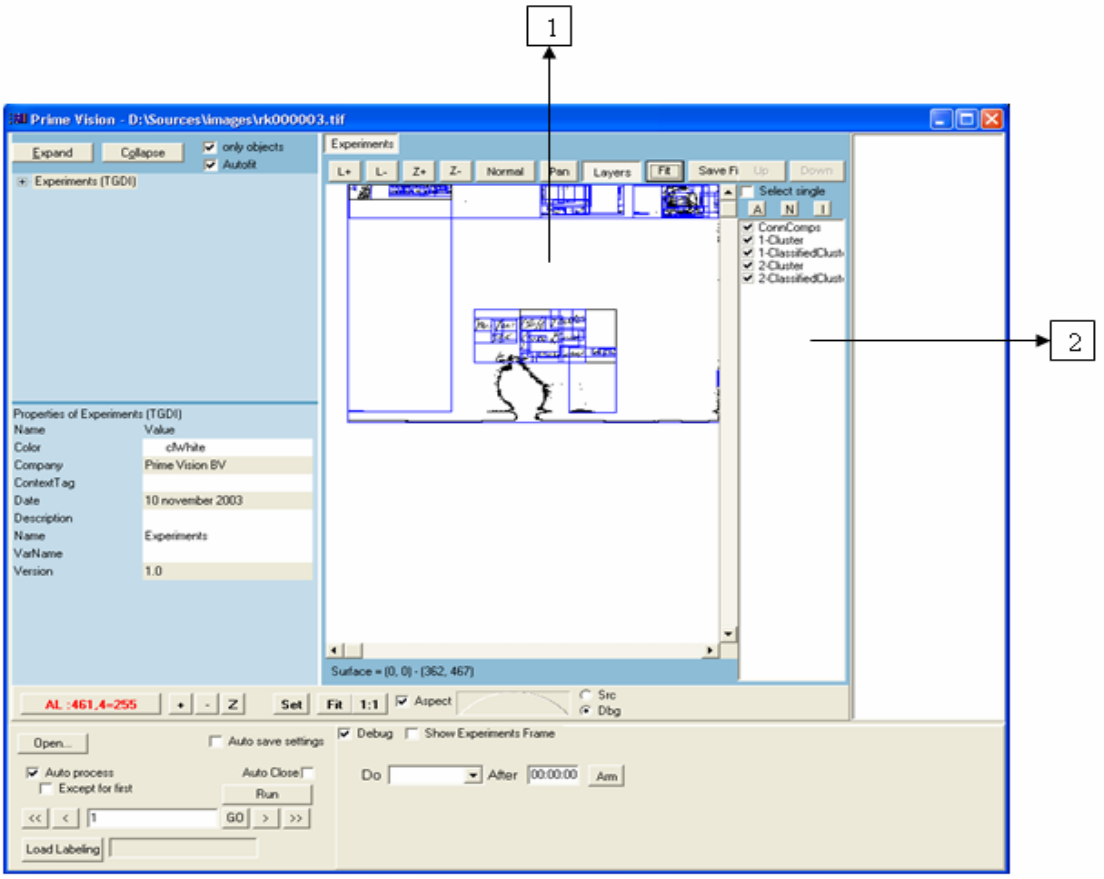


Figure 5-10: Interface of the debug mode area.

Table 5-2. Explanation of the debug mode interface.

Item number	Explanation
1	In this area the user sees the preprocessed image with same additional graphical data such as, bounding boxes.
2	In this area all the defined layers are shown. A layer is a simplistic plane of information. Each layer contains specific graphical objects of a specific part of the process. For instance, the layer denoted with “conncomps” in the figure 5-10 contains the preprocessed image, “1-cluster” layer contains the clusters bounding boxes related to segmentation experiment 1, and the layer “1-classifiedCluster” contains the bounding box of the cluster classified as destination address block for experiment 1. Note that each layer covers the layer above it.

5.3.3 Segmentation experiment's area

This sub interface as shown in figure 5-11 is one of the sub-interfaces that have been added in order to support the segmentation. It is the main interface for the new functionally, i.e., it controls all the other new sub interfaces that have been added. Table 5-3 gives an explanation of different items of this sub interface.

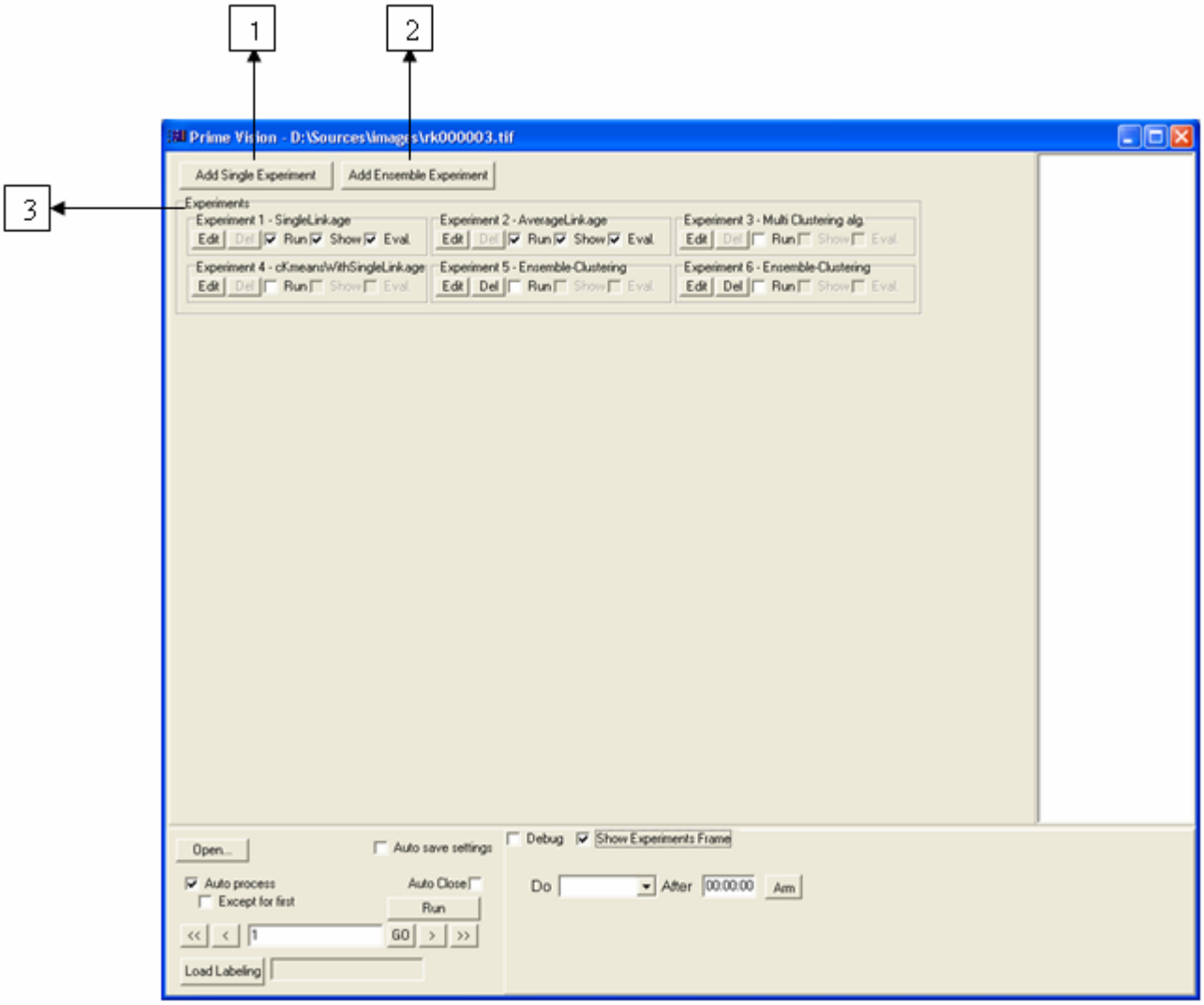


Figure 5-11: Interface of the segmentation experiments area.

Table 5-3. Explanation of the segmentation experiments area interface.

Item number	Explanation
1	This button allows to user to define a new single segmentation experiment.
2	This button allows the user to define a new ensemble segmentation experiment.
3	This shows a list of experiments with their control panels. Each time an experiment (single/ensemble) is added a control panel is dynamically created and associated with that certain experiment. This allows the user to edit, delete, and set the state of an experiment to run, show, or evaluate. Run indicates whether the experiment should be applied during the segmentation process. Show defines whether the experiment's results of a certain experiment should be available to be seen in debug mode. Finally, Eval. indicates whether a certain experiment should be evaluated when it is applied.

5.3.4 Single experiment's form

This sub interface appears when the user chooses to add new single experiment or edit an existing one. It contains all the functionality needed to define a single experiment. The layout of this interface changes dynamically according to the interaction of the user. For example, if the user clicks on the checkbox 'Random data with offset' an input box appears where the user can type the offset value. A screenshot of this interface is shown in figure 5-12. The content of this interface is explained in table 5-4.

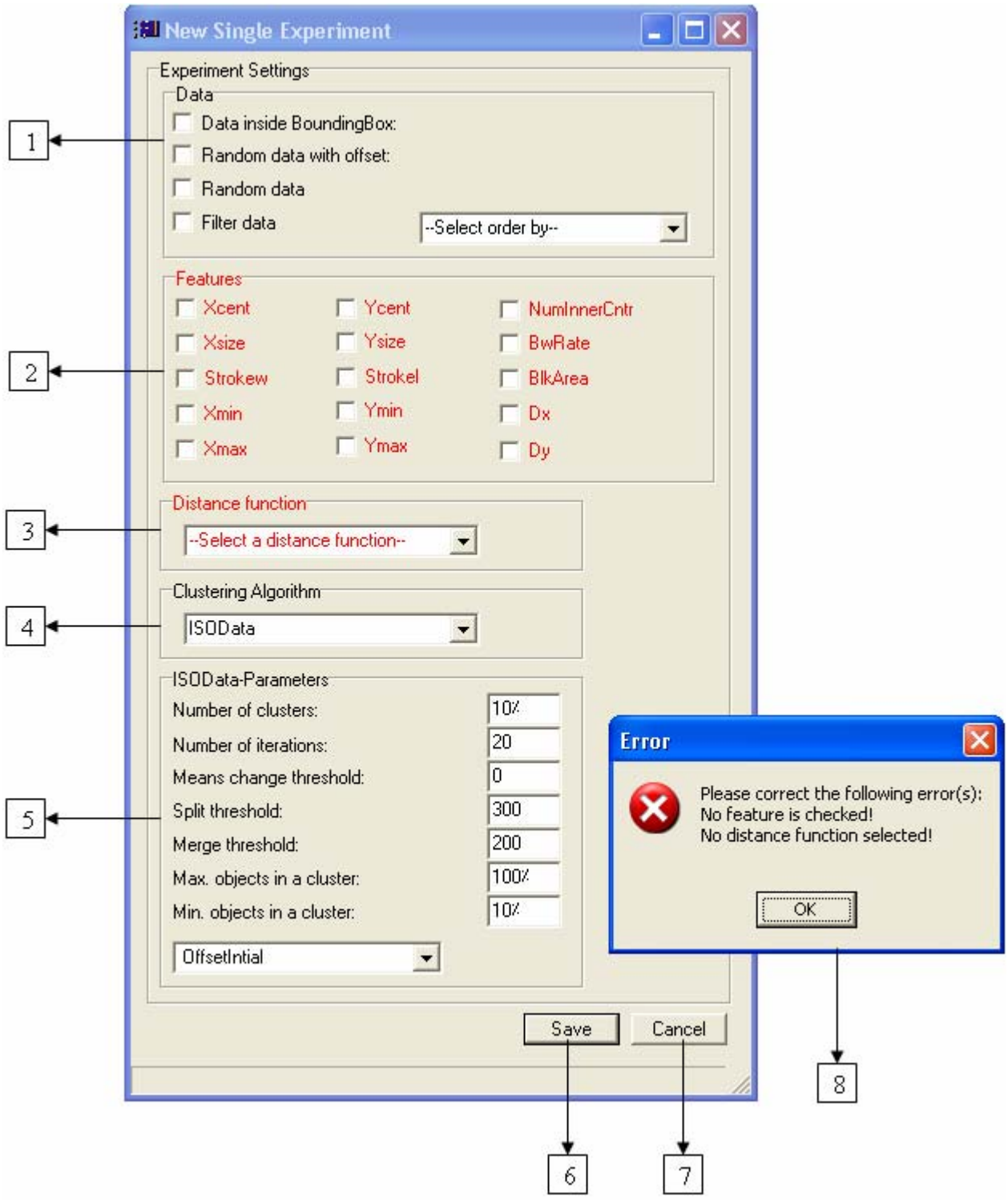


Figure 5-12: Interface of the single segmentation experiment.

Table 5-4. Explanation of the single experiment interface.

Item number	Explanation
1	This area contains all the user-defined data options that have been discussed in the previous chapter. The layout of this area adapts dynamically to the user's choice. If the option "Data inside boundingBox" is chosen a group of four input boxes appears, which allows the user to specify a bounding box. The same counts for the option "random data with offset", when this on is chosen an input box appears where the user can insert the offset value.
2	This area contains all the used features. The user can choose one or more features of the connected components to be clustered. Once a feature is chosen a little input box appears to its right where the feature weight can be entered.
3	This area contains all the implemented distance functions where the user can choose from.
4	This area contains all the available (single) clustering algorithms, which can be chosen by the user.
5	This area contains according to the chosen clustering algorithm in 4, all the parameters that can be modified by the user. The user can either modify these parameter's values or agree with the default values.
6	This button saves a new experiment or saves the modification of an existing experiment.
7	This button allows the user to cancel the process. Nothing will then be changed or added.
8	This is an error message box, which appears when the user forgets to define all the required settings. Additionally, the text of the area containing the error is made red to make error tracing easy.

5.3.5 Ensemble experiment's form

This sub interface as shown in figure 5-13 appears when the user chooses to add a new ensemble experiment or edit an existing one. It contains all the functionality needed to define an ensemble experiment. An explanation of this interface is described in table 5-5.

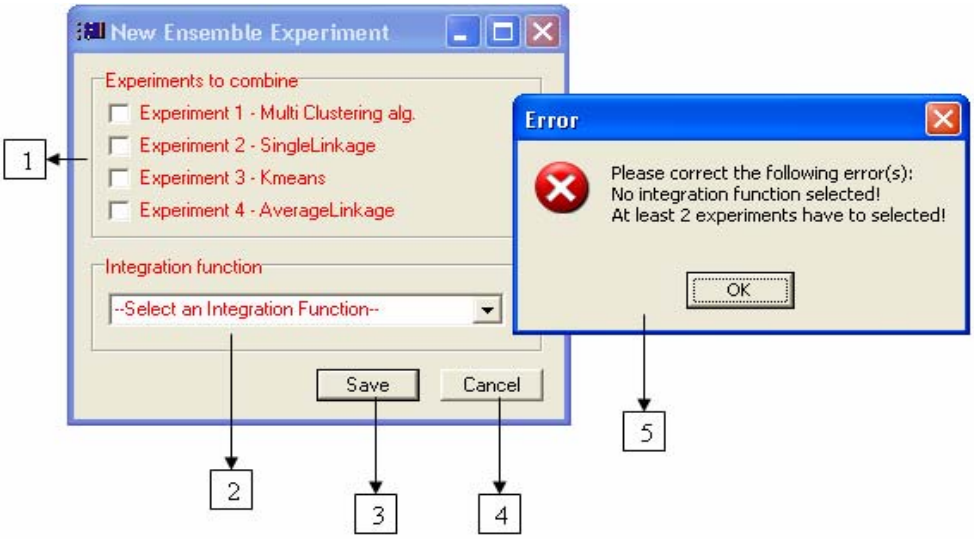


Figure 5-13: Interface of the ensemble segmentation experiment.

Table 5-5. Explanation of the ensemble experiment interface.

Item number	Explanation
1	This area lists all the available single experiments that the user can choose for the ensemble generation task. At least two experiments have to be chosen.
2	This area contains all the implemented integration functions where the user can choose from.
3	This button saves a new experiment or saves the modification of an existing experiment.
4	This button allows the user to cancel the process. Nothing will then be changed or added.
5	Like the single experiment, this appears when the user defines an experiment incorrectly. Likewise the text of the area containing the error is made red.

5.4 TESTING

The objective of testing is to ensure the new or updated functionality performs as intended. Testing should reveal any potential deviation from expected behavior of the code. The goal is to obtain high-qualified software. During the implementation stage, all the source code of the implemented classes were reviewed and tested to find potential faults. Some tests conducted during the implementation are described below.

5.4.1 Unit testing

During unit testing one tries to reveal faults in participating objects (e.g., method or class) with respect to the functional requirements of the object. Each class that has been produced, reviewed, and verified for correct syntax, was separately tested to establish errors within its bounds. To achieve this several debug outputs were implemented to test several methods of the class. However, because of time constraints, not every method of a class could be tested individually, so only the core methods were tested individually. If a method is composed of several sub-methods behaves correctly, it is assumed that the sub-methods behaved correctly as well. The data flow was tested to ensure that the information properly flows into and out of the class unit being tested. The local data structure was also tested to ensure that data stored temporarily maintains its integrity for all stages in an algorithm's execution. Boundary conditions are tested to ensure that the modules perform correctly at boundaries created to limit or restrict processing. Finally, different independent paths through the control structure were considered to ensure that all statements were executed correctly.

5.4.2 Integration testing

During integration testing one tries to find faults while testing the individually tested components together as one module. Once individual units that form a module have been finished and separately tested, they were integrated together and tested as one module. All the modules are tested to ensure that an individual unit does not have a bad impact on one another.

5.4.3 System testing

This kind of testing focuses on testing all the modules together as one single system. During our project several tests were preformed to ensure that the complete system complies with both functional and non-functional requirements. Throughout these tests the system was tested on several mail piece images of different types. Based on the use cases the system functionality was also tested. The results were also compared to the results obtained with the existing segmentation algorithm of Prime Vision.

5.4.4 Usability testing

Usability is one of the important aspects, which concerns on the user's understanding of the use case model as well as on the user interface details (e.g., the look, layout, etc…). A usability test focuses on the usability parameters, like the easiness to learn the system, the time needed to accomplish a certain task, or the rate of errors a user makes when accomplishing a task. In large software development, usability test follows three stages namely: **scenario test** where visionary scenarios of the system are tested on understanding's level by one or more users, **prototype test** where a system prototype is evaluated by a different user, and **product test** where the version of the end product is evaluated by different users.

In case where different users with different skills and backgrounds will use the system, usability test will be a must. However, since the proposed system is only for research purposes, usability test will not be carried out. Instead of this, the graphical user interface will be well explained.

Chapter 6: Experiments and results

An experimental study has been performed to evaluate our system. In this section we will describe those experiments and their results. First an explanation about the dataset is provided, afterwards the experimental setup will be described, the results will then be presented, and finally the experiments discussion will be provided.

6.1 EXPERIMENT DATA SET

Data set is one of the most important fundamentals of any type of research. Wrong datasets can affect the results of the research and ultimately lead to bad results. For this, there are several data sets available for us to use. However, in order to make sure that we use the correct data set, we decided to choose the dataset that is frequently used by Prime Vision to test their new functionalities. The data set consists of 50.000 different United States (US) mail piece images of different type, size, layout, writing-style, etc. All images are live captured from real postal mail at a postal sorting center in US. To create a small dataset 2000 different images were randomly selected. The US data set does not contain images of parcels. For this reason we decided to use also a dataset containing only parcel images. This dataset contains 1600 different parcel images live captured from real parcels at a Dutch postal sorting center. The ground-truth segmentation is available for almost all the images. However, since the labeling process is a manual task some image may be badly labeled or not labeled at all. Moreover, the data set may contain images that were badly captured (e.g. too dark, no address-block at all, etc…). One can exclude those bad images from the data set, but since this is also a problem for the real-time segmentation system, we decided to keep them in our data set. One main reason for this is because we want to make sure that the implemented algorithms can deal with real life situations.

6.2 EXPERIMENTS SETUP

Our experiments focus is on comparing the quality of the segmentation results produced by our algorithms with both the existing algorithm and the ground-truth segmentation. We conducted four types of experiments, namely:

- Different algorithms;
- The same algorithm with different features;
- The same algorithm with random data; and
- The same algorithm with random parameters.

Based on some experiments it turned out that all the algorithms related to this project are not sensitive to the data order. Therefore, data order related experiments are not conducted. We also experimented with different features in order to examine their influence. Based on these experiments it turned out that too many features together decrease the accuracy. We achieved the best results with the features 'xcent' and 'ycent'. Therefore we will use those features for our experiments. Likewise we analyzed the influence of different distance functions and we achieved the best results with the Euclidean distance function. Accordingly we will use the Euclidean distance function as our distance function. Table 6-1 summarizes all the algorithms used in the experiments with their parameter settings as mentioned in 4.1.6.

Table 6-1. Clustering algorithm settings.

Number	**Name**	**Settings**
1	EM-Algorithm	The parameter k was set to 10%, MaxIter was set to 20, ε was set to 0.01, and as initialization method we choose offset method.
2	K-Means	The same settings as EM-algorithm were used except for the parameter ε. For the k-means this was set to 0.
3	ISO-Data	The parameter k was set to 50%, MaxIter was set to 20, and ε was set to 0. Furthermore, the parameter $T_{\min}$ was set to 10%, $T_{\max}$ was set to 100%, T_{std} was set to 300, and T_{dis} was set to 200. Likewise the offset method was chosen as the initialization method.
4	Single-Linkage	No parameters to set.
5	Complete-linkage	No parameters to set.
6	Average-linkage	No parameters to set.
7	Multi clustering alg.	No parameters to set.

For our experiments we used 3 personal computers with a Pentium 4 2.2 GHz. Processor and 512MB memory. The exact experiments are explained below.

6.2.1 Experiments with different algorithms

The main goal of this experiment type is to evaluate both the quality of each algorithm as individual and as a set of algorithms together. We used for this all the implemented algorithms. The diversity needed for the ensemble approach was achieved by the solutions obtained from a set of algorithms combined together. The implemented integration functions were selectively applied on the base clusters for the combining process. The exact experiments used for this type of experiment are:

- **Single experiments on the US dataset:** For this, we used in total seven single experiments, namely: EM-Algorithm, K-means, ISO-Data, Single-linkage, Complete-linkage, Average-linkage, and Multi Clustering algorithm.
- **Ensemble experiments on US dataset:** In order to evaluate the ensemble approach for the US dataset we defined the following experiments:
  - **OverlappingBased 1+2+3:** For this we used the first, the second and the third single experiments to generate base clusters and overlapping based integration function for the integration process.
  - **OverlappingBased 4+5+6:** For this we used the 4^{th}, the 5^{th}, and the 6^{th} single experiments to generate base clusters and overlapping based integration function for the integration process.
  - **OverlappingBased 2+7:** For this we used the second and the 7^{th} single experiments to generate base clusters and overlapping based integration function for the integration process.
  - **InformationTheoryBased 2+7:** For this we used the second and the 7^{th} single experiments to generate base clusters and information theory based integration function for the integration process.
- **Single experiments on the Parcels dataset:** For this we used the K-means, ISO-Data, Single-linkage, Average-linkage, and Multi Clustering algorithm.

- **Ensemble experiments on the Parcel dataset:** For this we defined the following experiments:
  - **OverlappingBased 2+4:** For this we used the second and the 4th single experiments to generate base clusters and overlapping based integration function for the integration process.
  - **OverlappingBased 4+6:** For this we used the 4th and the 6th single experiments to generate base clusters and overlapping based integration function for the integration process.
  - **OverlappingBased 3+4+7:** For this we used the third, the 4th, and the 7th single experiments to generate base clusters and overlapping based integration function for the integration process.
  - **InformationTheoryBased 3+4+7:** For this we used the third, the 4th, and the 7th single experiments to generate base clusters and information theory based integration function for the integration process.

6.2.2 Experiments with the same algorithm and random parameters

With this experiment type we want to evaluate the influence of different initializations. For this, we used the k-means as described in table 6-1 with different numbers of clusters. This was also used as a method of generating the base clusters for ensemble approach. Diversity of the clusters is achieved by the solutions obtained after different initialization of the algorithm. Based on this the following experiments were defined:

- **Single experiments with k-means**: For the US dataset we varied the number of clusters from 3 to 10. For the parcel dataset k-means was defined with 3, 9, and 12 as the number of clusters. The number of iteration is set to 20 and the mean's change threshold is set to 0. Further the offset-initialization is selected as method of mean initialization.
- **Ensemble experiment on US dataset**: K-means experiment with the following number 3, 6, and 9 as number of clusters were used to generate base clusters. In

order to integrate those base clusters we used the overlapping based integration function.

- **Ensemble experiment on Parcel dataset**: For this, k-means with the following number of clusters: 3, 9, and 12 were used as ensemble generation. For the integration stage we tested all the three integration functions that are implemented.

6.2.3 Experiments with the same algorithm and random data

The main goal of this type of experiment is to evaluate whether better results can be obtained with different subsets of the original dataset. In order to evaluate this, we applied the same single-linkage clustering algorithm on random data with different offsets. This approach was then used as a method generating the base clusters for the combination. Diversity was achieved by the solution obtained with random data of different offsets. In accordance with this we defined these algorithms:

- **Single experiments on the US dataset:** For this we defined two experiments with single-linkage algorithm and the same settings, but with different random data offsets. We choose for one experiment offset 3 and for the other 5.
- **Ensemble experiment on the US dataset:** The single experiments described above were used for the ensemble generations. All the three integration functions were then applied for the ensemble integration process.

6.2.4 Experiments with the same algorithm and different features

In this experiment type we want to evaluate the influence of different features with the same clustering algorithm. For this, we defined several experiments pairs with different features. Those experiments were then used in order to obtain diversity for the ensemble method. The exact experiments we used are:

- **Single experiments on US dataset:** We defined in total three pairs of experiments, namely:

- An experiment pair with single-linkage clustering algorithm and different features. For one experiment we used xcent and ycent as features and for the other Dx and Dy as features.
- An experiment pair with complete-linkage clustering algorithm and different features. For one experiment we used xcent and ycent as features and for the other xsize and ysize as features.
- An experiment pair with average-linkage clustering algorithm and different features. For one experiment we used xcent and ycent as features and for the other Dx and Dy as features.

❖ **Ensemble experiment on the US dataset:** Each single experiment pair was used a method of generating the base clusters for the ensemble approach. The overlapping based integration was then applied on those base clusters to obtain the final results.

6.3 EXPERIMENT RESULTS

This section presents and analyzes the experimental results we obtained. It is important to note that each set of results presented in this section was obtained independently through a separate set of experiments. We analyzed for each experiment the correct score of the segmentation, the number of correctly classified images, the maximum run-time, and the average run-time. Note that the correctly classified images denote the number of images where the classified cluster (i.e., DAB) is the same as cluster closest to the labeling cluster. The correct segmentation rate is a value varying from 0 to 1, which indicates the overlapping of the segmentation with the ground-truth segmentation. Where 0 denotes no overlapping and 1 denotes 100% overlapping. The histogram illustrates for each correct segmentation rate the number of images.

6.2.1 Experiments with different algorithms

The results that we obtained from this type of experiments are described below. First the experiments on US dataset are provided followed by the experiments on Parcels dataset.

Experiments on the US Dataset

Figure 6-1 shows that hierarchical clustering (i.e. single-linkage, complete-linkage, average-linkage, and multi clustering algorithm) obtained the best results. EM-algorithm had the worst segmentation rate. The reason for this is because this algorithm suffers mostly from under-segmentation, i.e. it contains clusters that are to large. In figure 6-2 we see that the results are even worse when combining bad clustering algorithms. Contrarily is shown in figure 6-3 where combining good clustering algorithms gives an improved result. In figure 6-4 we see that combining good clustering algorithm with bad clustering algorithm decreases the quality. Likewise in figure 6-5 we see that the quality is not only affected by the bad clustering algorithm but also by the type of integration function. Table 6-2 summarizes the average run-time, the maximum run-time, and the number of correctly classified images for each experiment. The hierarchical and ensemble clustering algorithms achieved the highest number of correctly classified images. Unfortunately one drawback of this kind of algorithms is the high computational time.

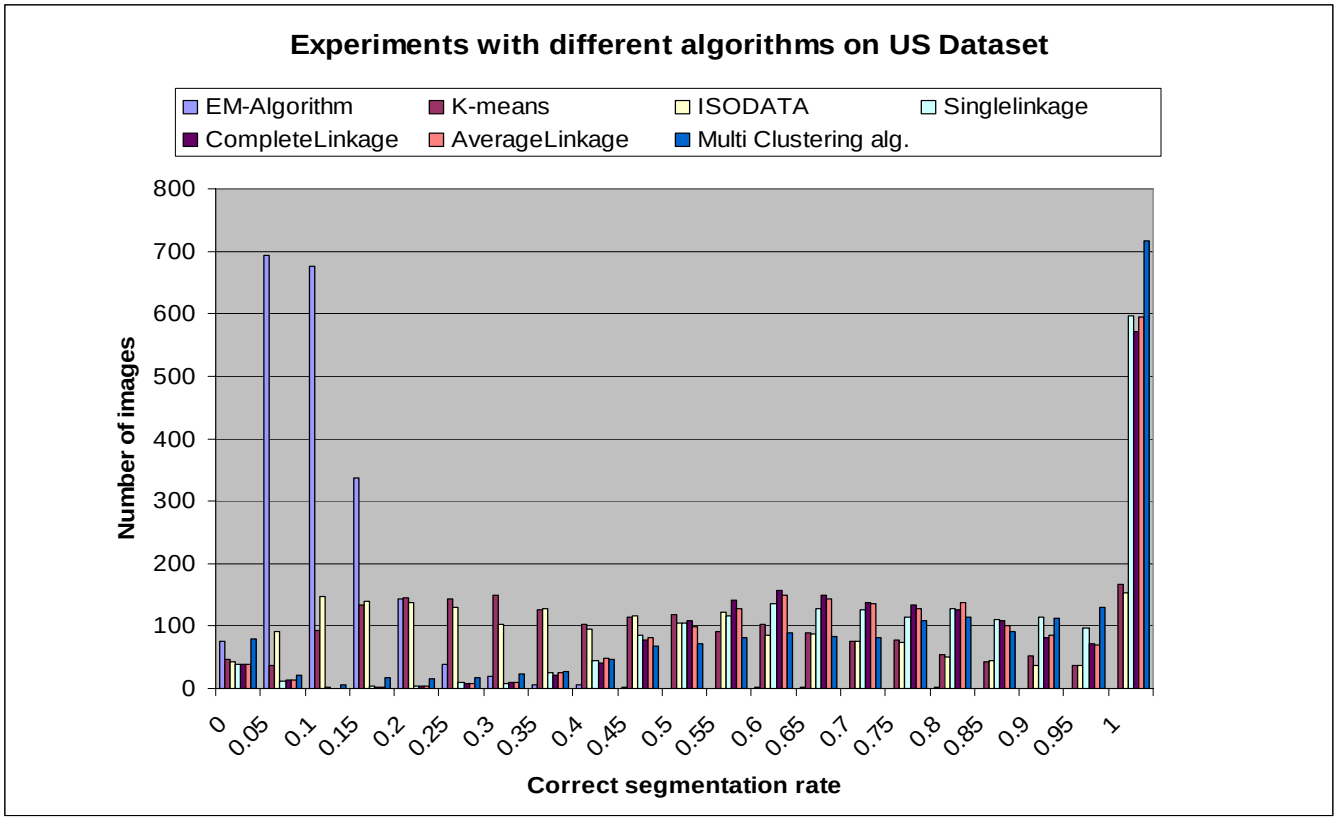


Figure 6-1: Histogram of the single experiments with different algorithms on US dataset.

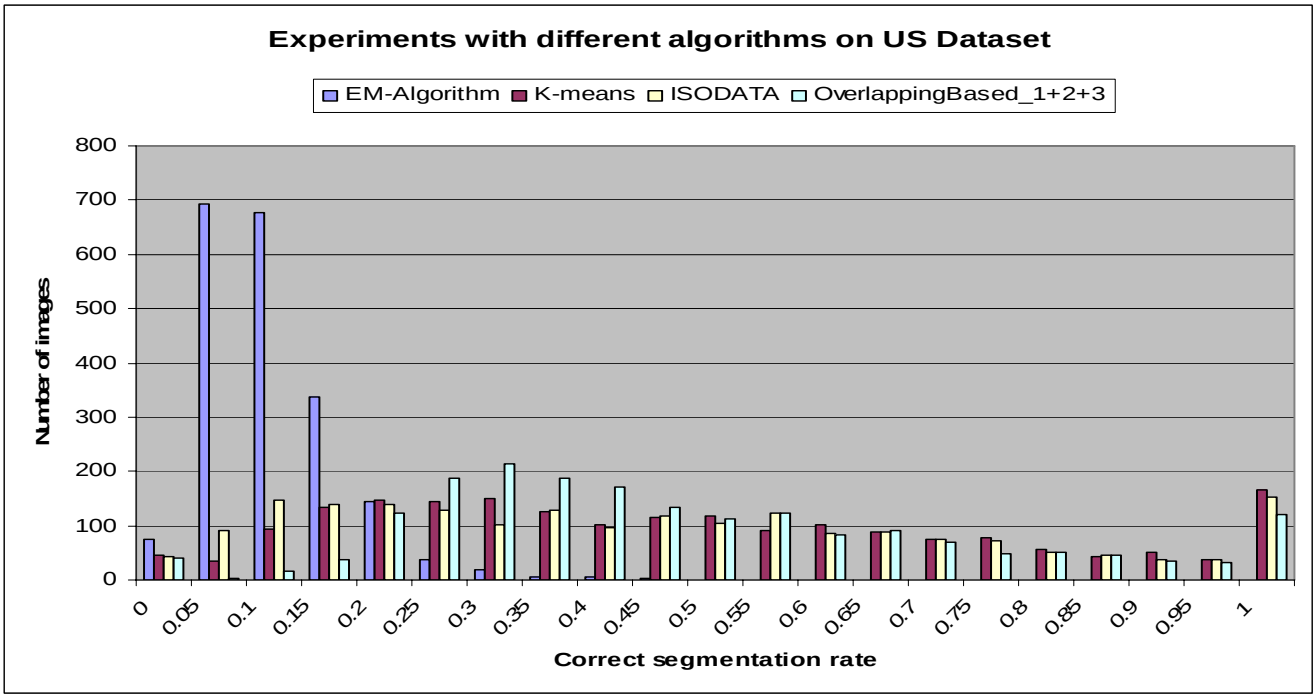


Figure 6-2: Histogram of the overlapping experiment 1 with different algorithms on US dataset.

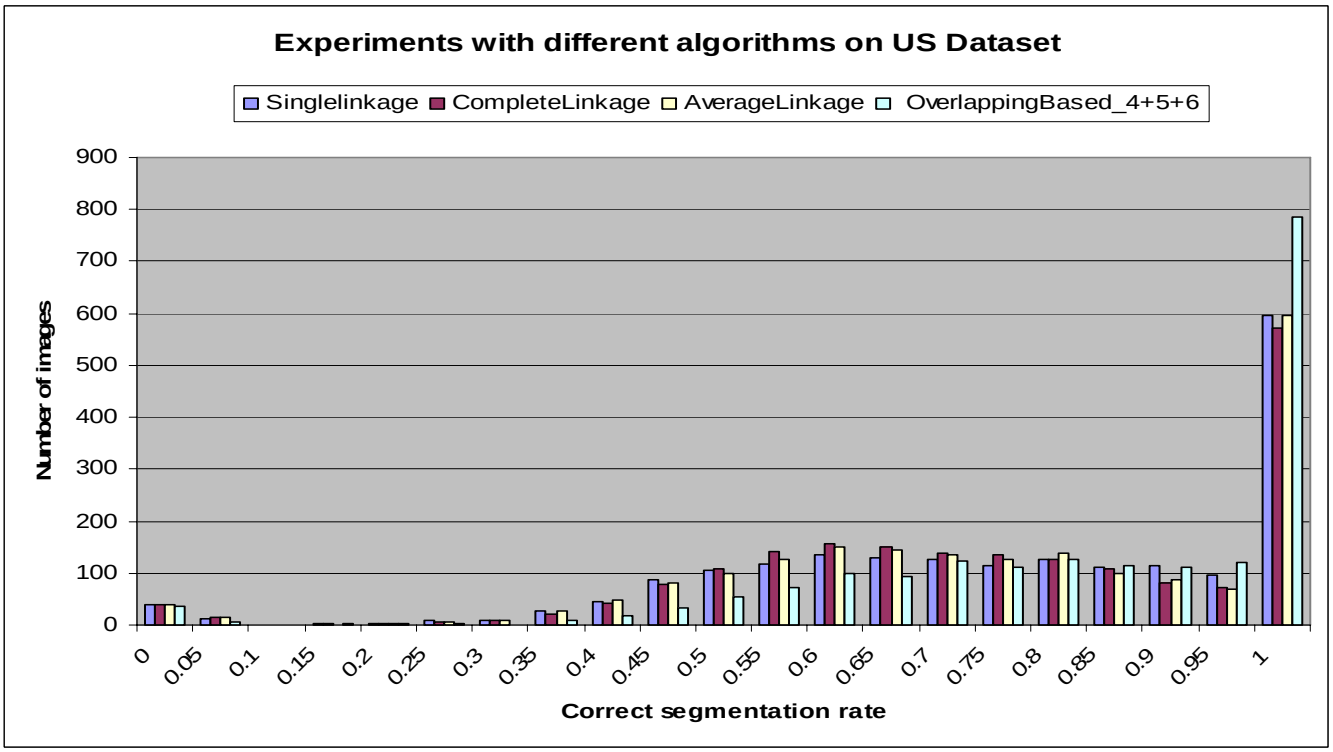


Figure 6-3: Histogram of the overlapping experiment 2 with different algorithms on US dataset.

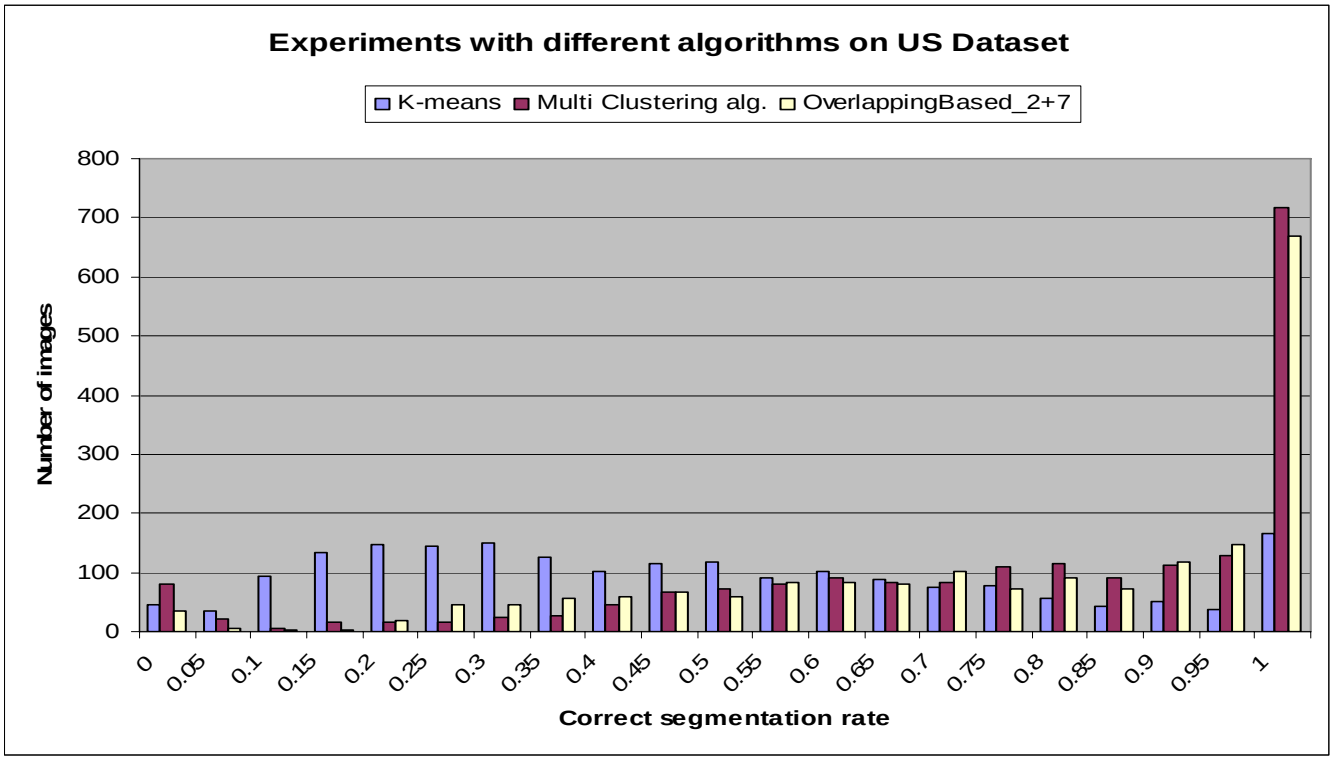


Figure 6-4: Histogram of the overlapping experiment 3 with different algorithms on US dataset.

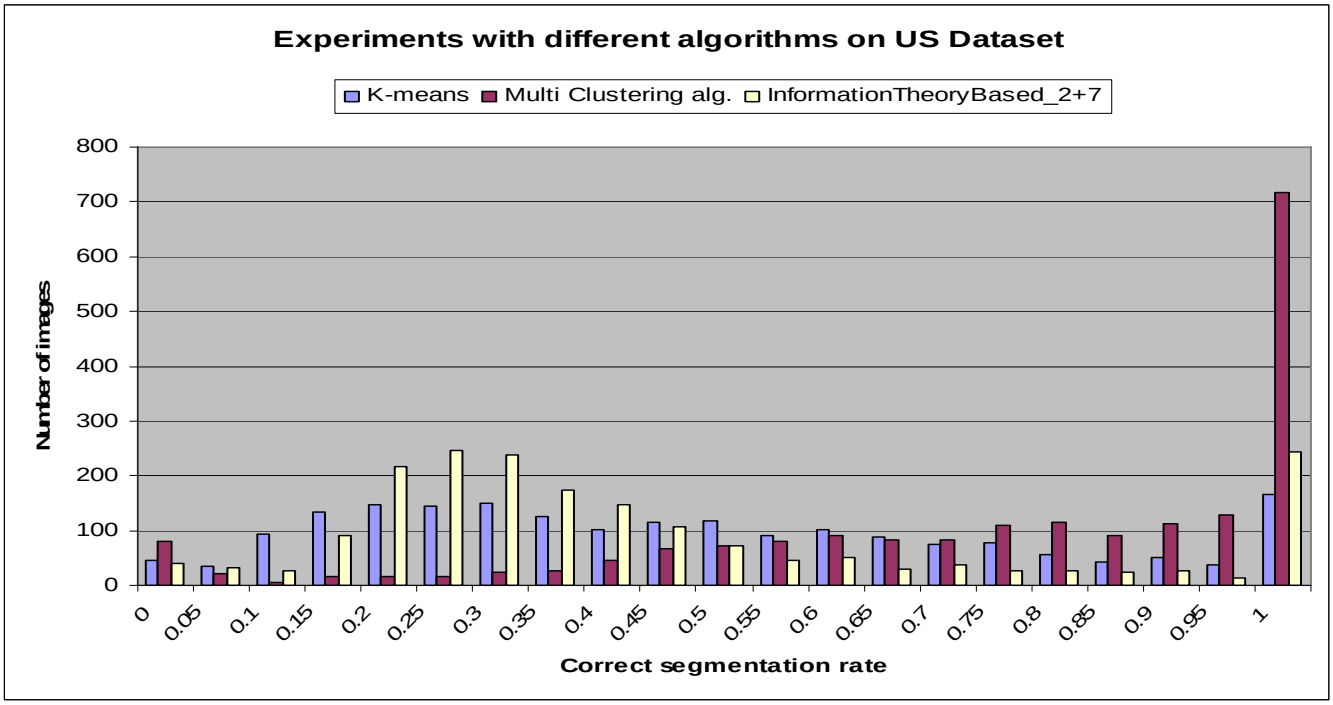


Figure 6-5: Histogram of the information theory experiment with different algorithms on US dataset.

Table 6-2. Different algorithms experiments on US dataset.

Experiment	Average run-time (seconds)	Maximum run-time (seconds)	Correct classified images
EM-Algorithm	3,59	75,30	1
K-means	4,08	111,25	216
ISODATA	7,66	646,92	159
Singlelinkage	14,44	1081,91	392
CompleteLinkage	14,15	1034,33	369
AverageLinkage	14,42	1451,54	383
Multi Clustering alg.	0,14	1,77	527
OverlappingBased_1+2+3	10,86	699,38	233
OverlappingBased_4+5+6	35,86	2379,31	164
OverlappingBased_2+7	3,59	95,39	494
InformationTheoryBased_2+7	3,59	97,05	388

Experiments on the Parcels Dataset

In Figure 6-6 we see that hierarchical clustering obtained also for the parcels dataset the best results. Like the US dataset we see in figure 6-7 that combining bad clustering algorithm with a good one doesn't improve the quality. Combining good clustering algorithms (see figure 6-8 and 6-9) gives also better results for this dataset. However, in figure 6-10 we see that the quality depends also on the integration function. Table 6-3 summarizes the average run-time, the maximum run-time, and the number of correctly classified images for each experiment. Likewise, here the hierarchical and ensemble clustering algorithms achieved the highest number of correctly classified images. Unfortunately, they suffer even more from higher computational time.

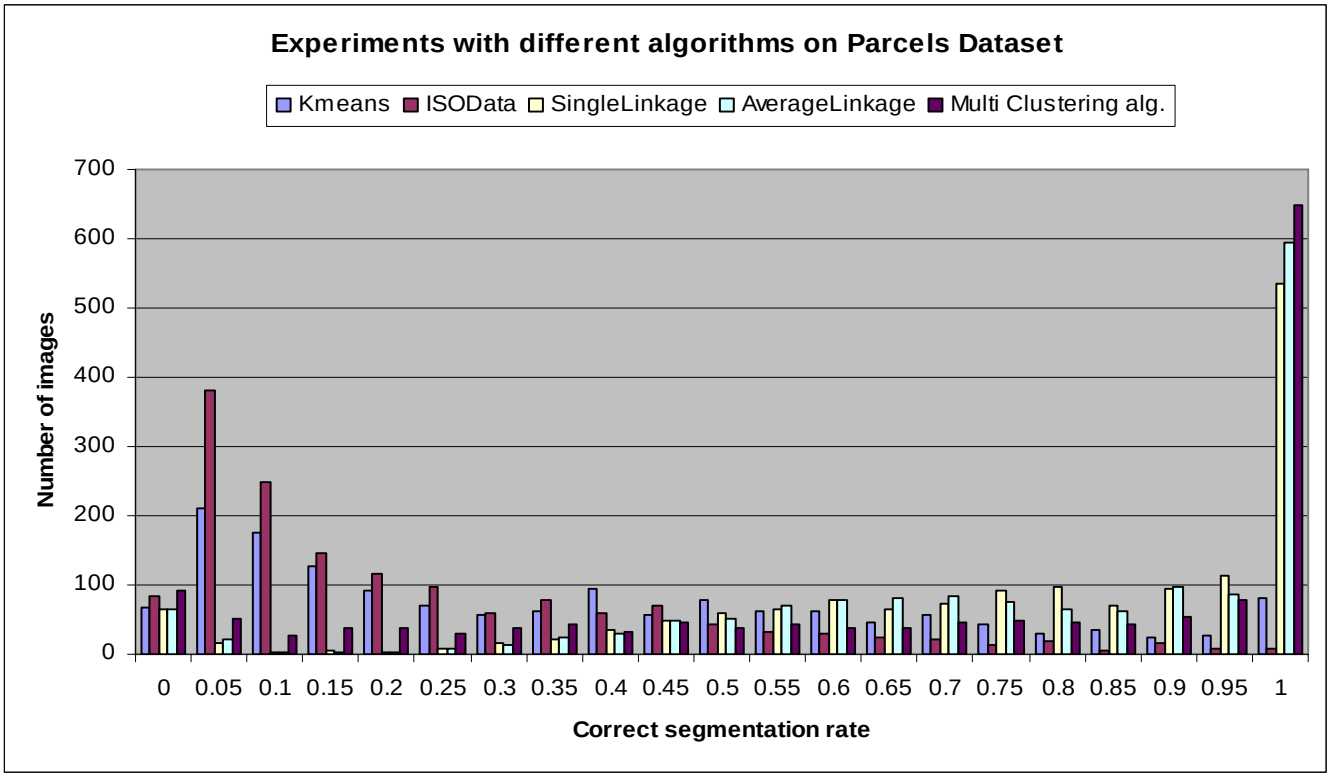


Figure 6-6: Histogram of the single experiments with different algorithms on Parcels dataset.

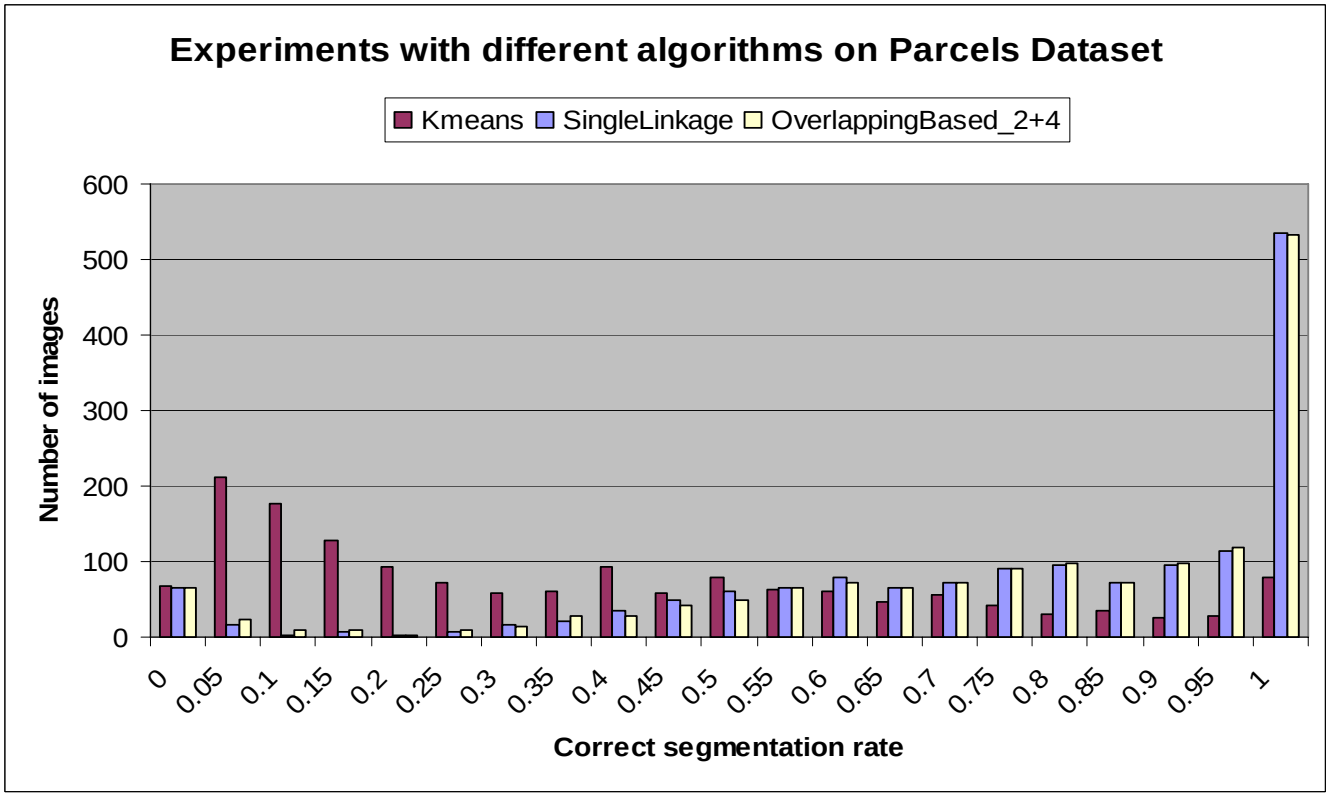


Figure 6-7: Histogram of the overlapping experiment 1 with different algorithms on Parcels dataset.

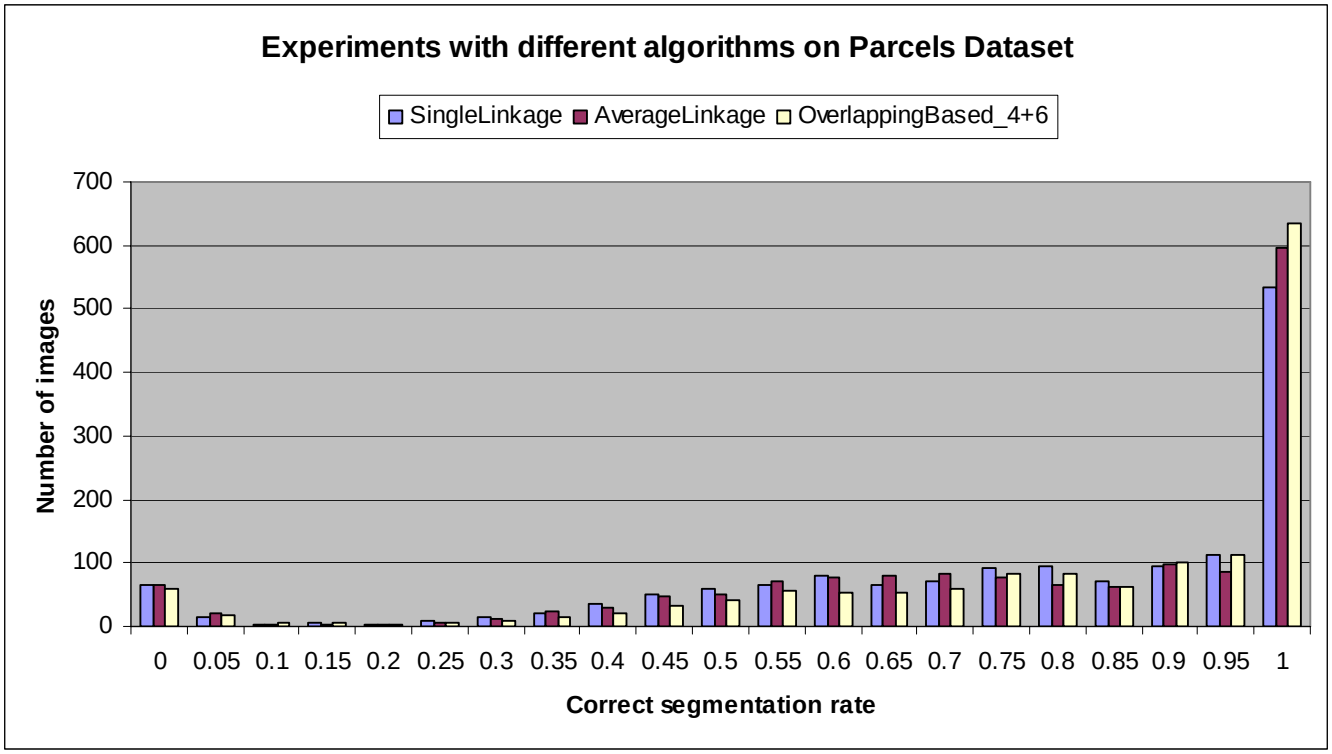


Figure 6-8: Histogram of the overlapping experiment 2 with different algorithms on Parcels dataset.

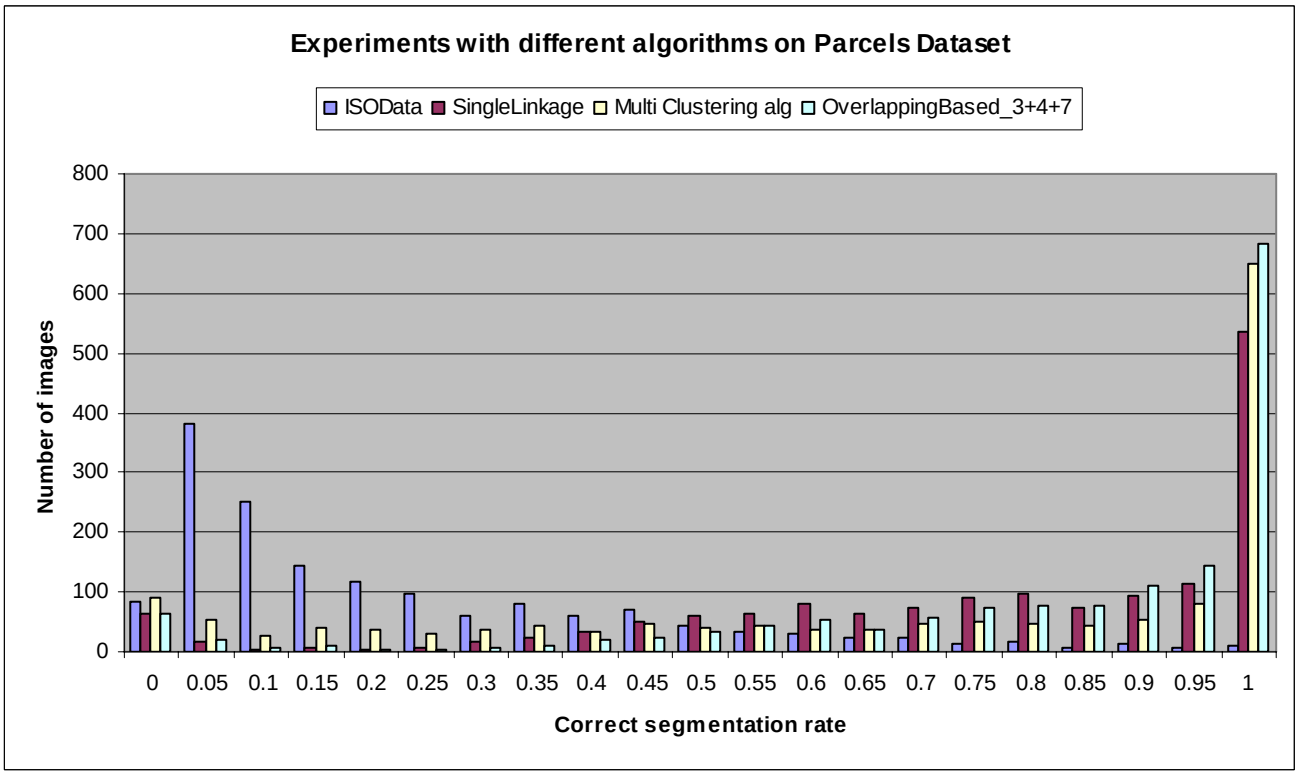


Figure 6-9: Histogram of the overlapping experiment 3 with different algorithms on Parcels dataset.

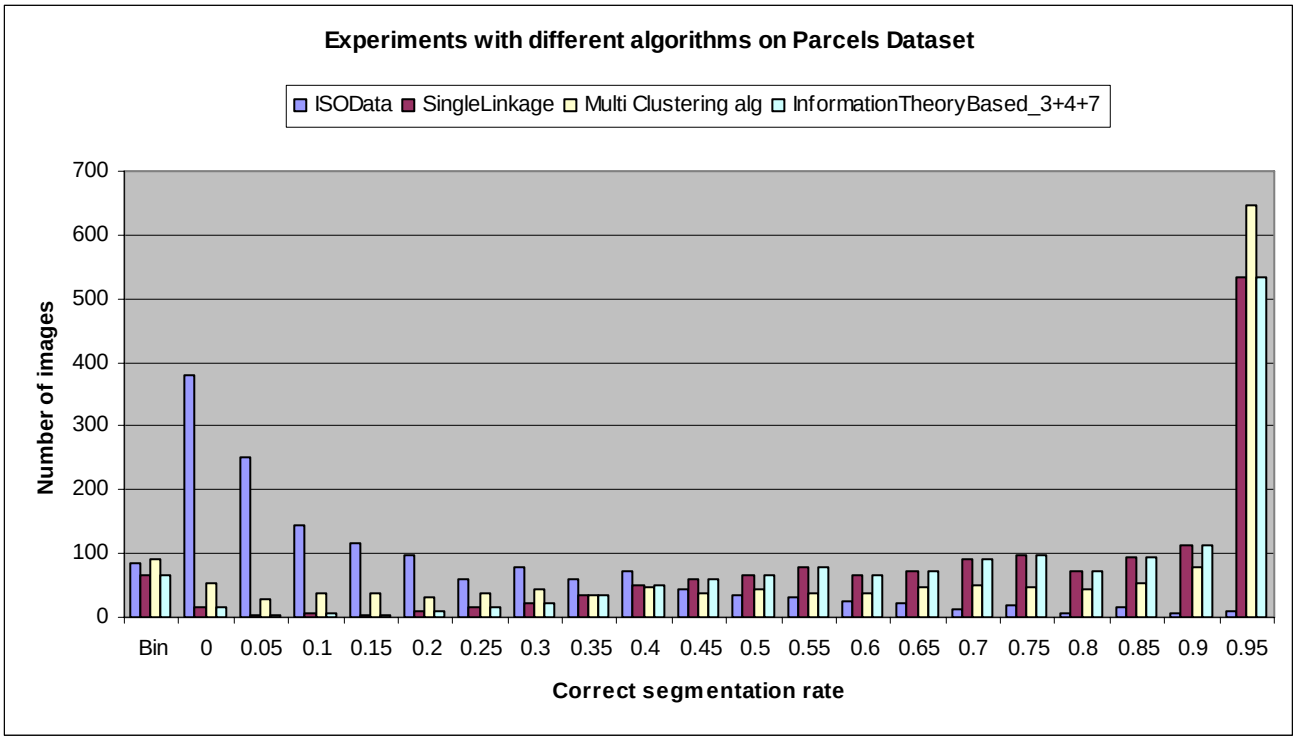


Figure 6-10: Histogram of the information theory experiment with different algorithms on Parcels dataset.

Table 6-3. Different algorithms experiments on Parcels dataset.

Experiment	Average run-time (seconds)	Maximum run-time (seconds)	Correct classified images
SingleLinkage	18,20	1011,76	431
Kmeans	0,65	14,92	104
ISOData	7,81	832,23	15
AverageLinkage	17,98	1842,84	441
Multi Clustering alg.	0,26	13,22	537
Overlappingbased_2+4	2,22	2117,90	401
Overlappingbased_4+6	22,63	1387,09	401
Overlappingbased_3+4+7	32,10	2476,83	274
InformationTheoryBased_3+4+7	32,99	3102,48	431

6.2.2 Experiments with the same algorithm and random parameters

The experiment's results based on algorithms with random parameters for both datasets are described below.

Experiments on US the Dataset

In Figure 6-11 we see that varying the number of desired clusters affects the clustering results. The best results were obtained with 6 and 9 as number of clusters. Figure 6-12 shows that a combination of clustering algorithm with different parameters increases the result's quality. However, the results are still below the results achieved with any of the hierarchical clustering algorithms. One reason for this that partitional clustering algorithm predefine the number of cluster and since different mail piece images may contain different number of clusters this predefined number of clusters is not always satisfied by all the existing mail piece images. Table 6-4 summarizes the average run-time, the maximum run-time, and the number of correctly classified images for each experiment. We see that the run-time increases with the number of clusters. This is because a high number of clusters need to do more object assignment and re-assignment. The ensemble clusters have a higher computational time than the other single experiments, because it uses a combination of clustering algorithms to generate its base clusters. The number of correctly classified images increases however with the number of clusters until 9 and then it start to decreases.

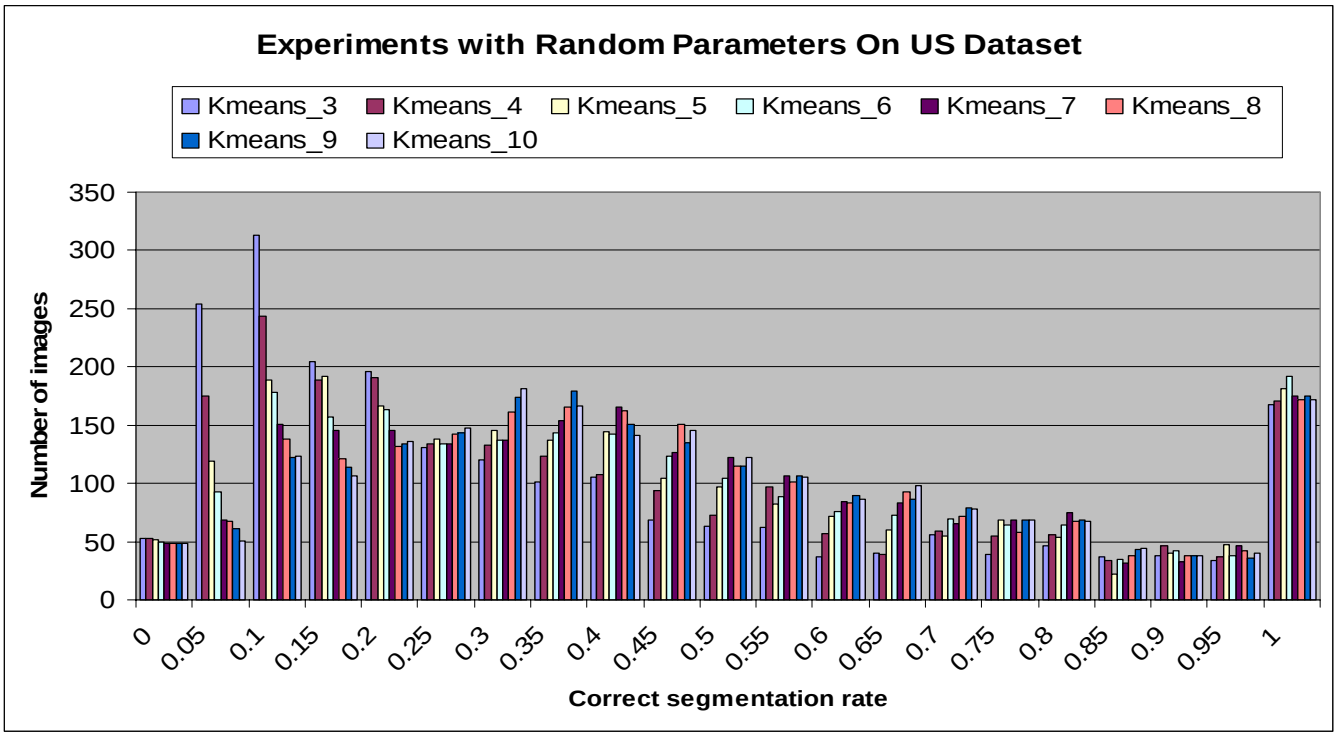


Figure 6-11: Histogram of single experiments with random parameters on US dataset.

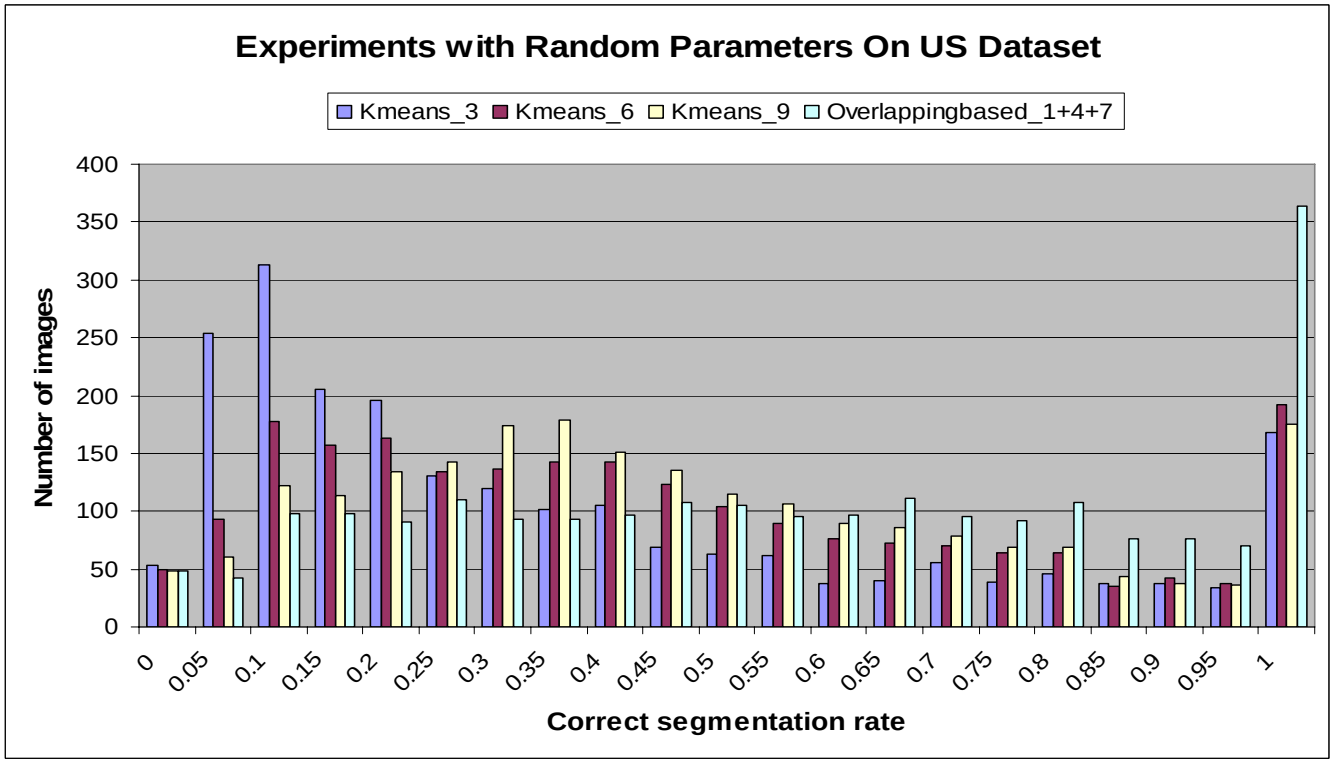


Figure 6-12: Histogram of the overlapping experiment with random parameters on US dataset.

Table 6-4. Random parameters experiments on US dataset.

Experiment	Average run-time (seconds)	Maximum run-time (seconds)	Correct classified images
Kmeans_3	0,29	2,55	127
Kmeans_4	0,27	2,14	169
Kmeans_5	0,31	2,39	201
Kmeans_6	0,34	2,66	245
Kmeans_7	0,38	2,94	266
Kmeans_8	0,38	3,03	289
Kmeans_9	0,44	3,44	294
Kmeans_10	0,48	3,66	268
Overlappingbased_1+4+7	3,25	42,09	175

Experiments on the Parcels Dataset

Figures 6-13 and 6-14 shows that a combination of clustering algorithm with different parameters increase the result's quality. However, the overlapping based integration function achieved better results than the other integration functions. Moreover, in figure 6-15 we see that the ensemble results are even less than the single clusters. Table 6-5 summarizes the average run-time, the maximum run-time, and the number of correct classified images for each experiment. Also for this dataset we see that larger number of clusters and the ensemble experiments have a higher computational time. Likewise k-means with 9 as number of clusters, achieved the highest number of correctly classified images.

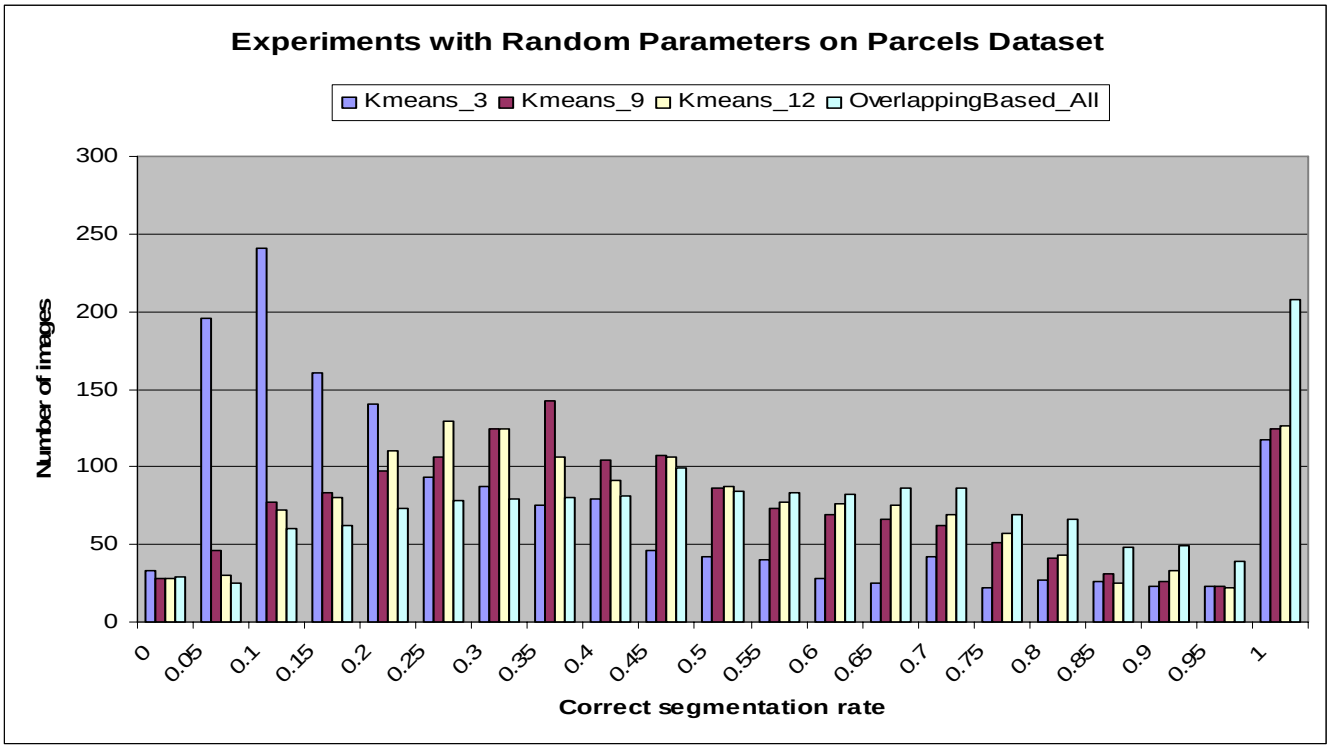


Figure 6-13: Histogram of the overlapping experiment with random parameters on Parcels dataset.

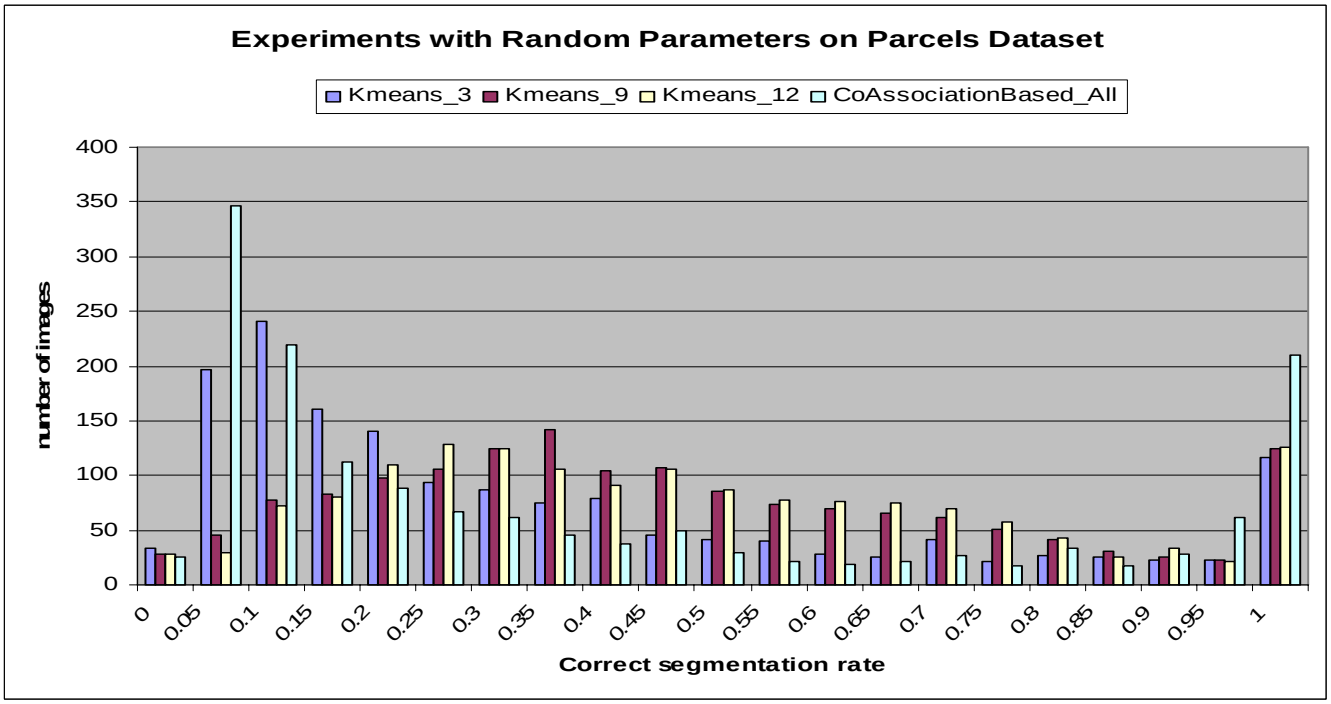


Figure 6-14: Histogram of the co-association experiment with random parameters on Parcels dataset.

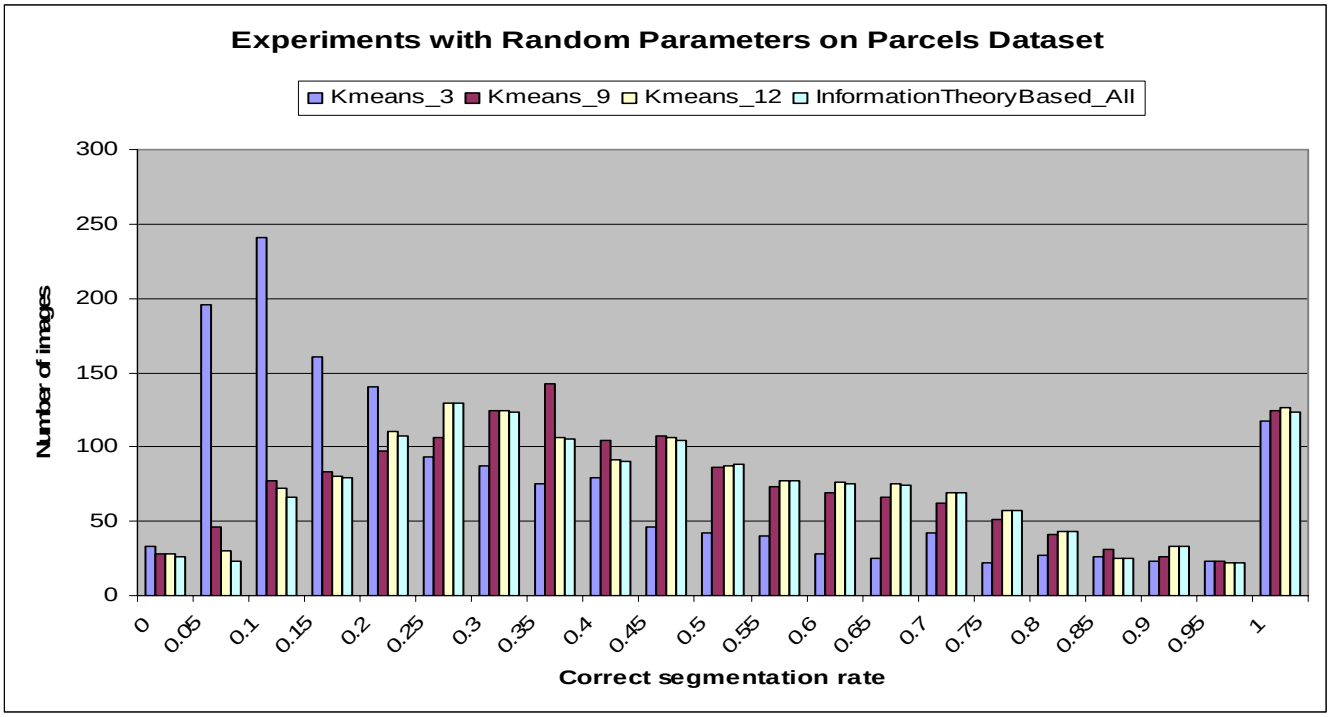


Figure 6-15: Histogram of the information theory experiment with random parameters on Parcels dataset.

Table 6-5. Random parameters experiments on Parcels dataset.

Experiment	Average run-time (seconds)	Maximum run-time (seconds)	Correct classified images
Kmeans_3	0,464	4,17	86
Kmeans_9	0,635	4,47	204
Kmeans_12	0,78	5,44	203
Overlappingbased_All	1,80	17,95	148
CoAssociationBased_All	16,74	304,65	66
InformationTheoryBased_All	2,19	16,42	201

6.2.3 Experiments with the same algorithm and random data

In figures 6-16, 6-17 and 6-18 we see that none of the ensemble approaches was able to improve the clustering quality. Moreover, the accuracy of single-linkage clustering algorithm is decreased because of the random dataset. One reason for this is that random dataset may exclude core objects, which are members of the DAB. Due to this the results clusters and the ground-truth segmentation share less or no objects at all. Table 6-6 summarizes the average run-time, the maximum run-time, and the number of correct classified images for each experiment. We see that the computation time is much less than for the normal case where the whole dataset is used. The reason for this is because the number of objects to be clustered is much less than in the normal case. The number of correctly classified images is also very low for those type of experiments.

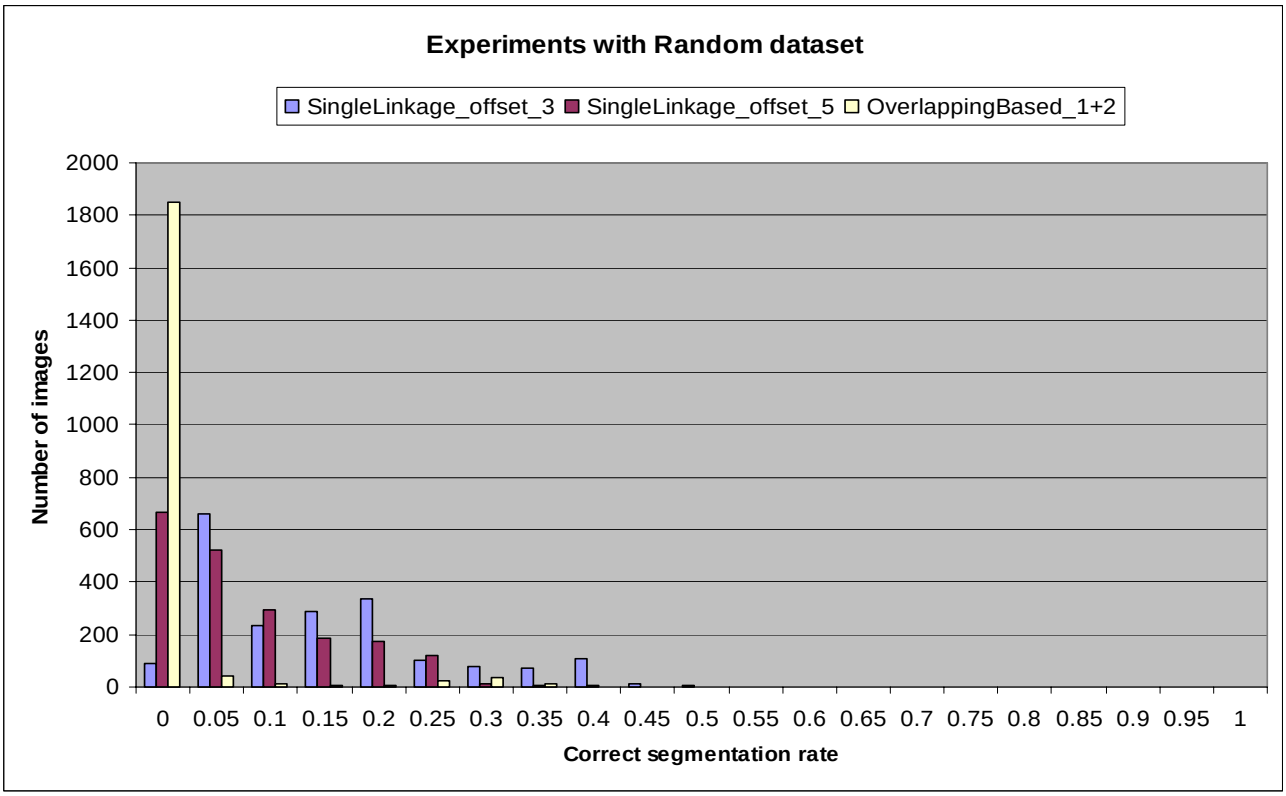


Figure 6-16: Histogram of the overlapping experiment with random dataset on US dataset.

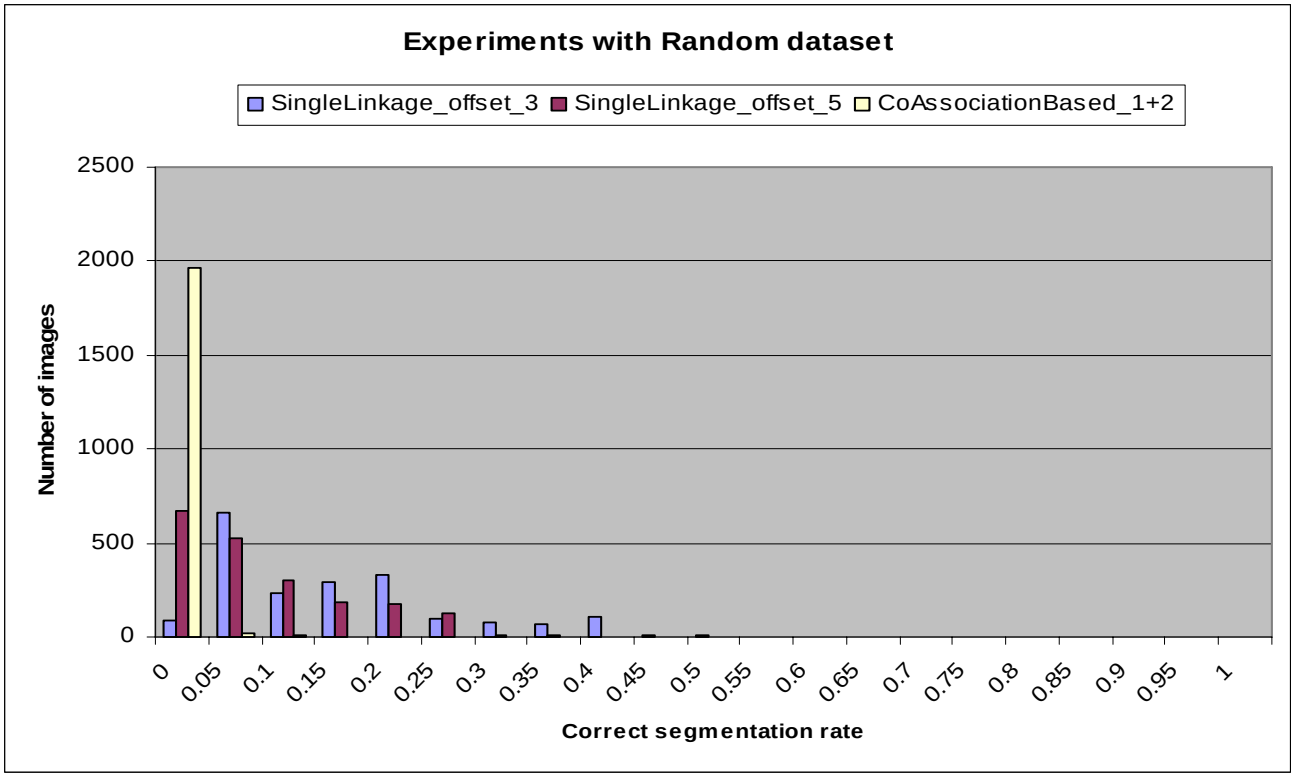


Figure 6-17: Histogram of the co-association experiment with random dataset on US dataset.

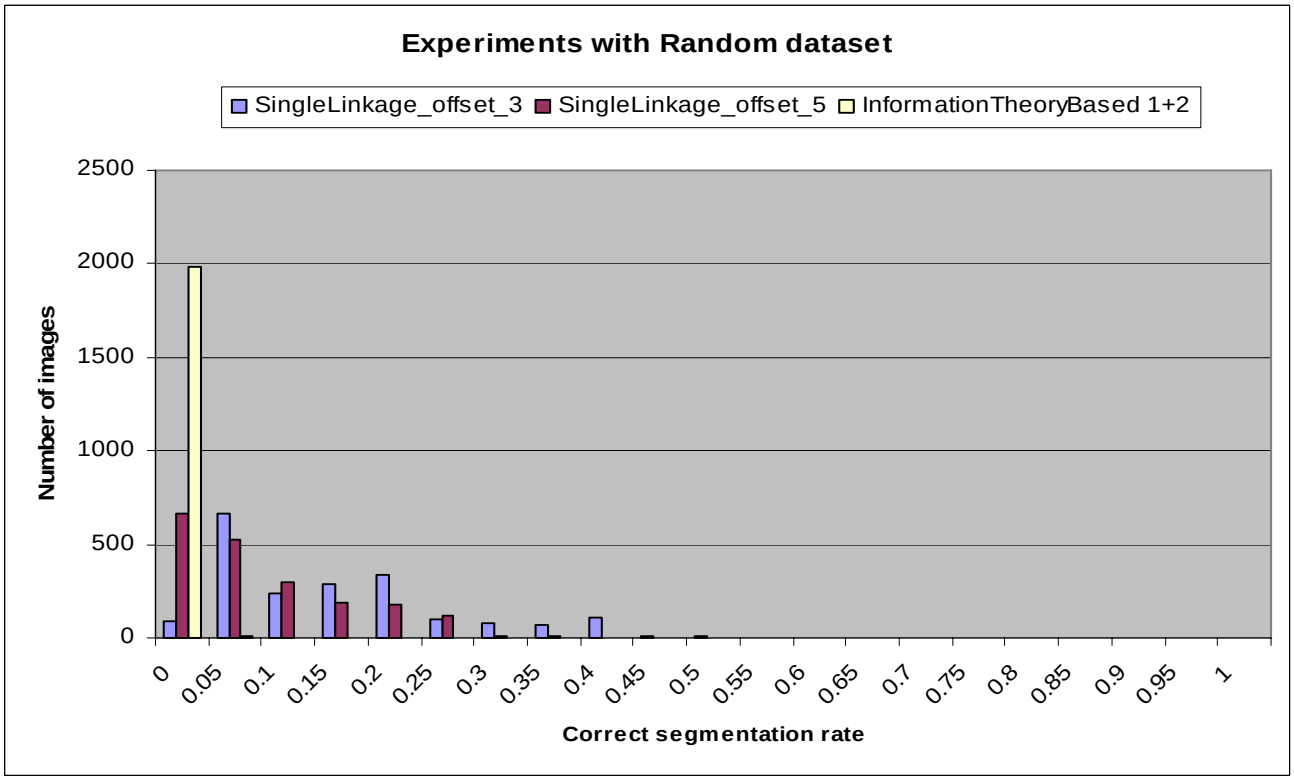


Figure 6-18: Histogram of the information theory experiment with random dataset on US dataset.

Table 6-6. Random dataset experiments on US dataset.

Experiment	Average run-time (seconds)	Maximum run-time (seconds)	Correct classified images
SingleLinkage_offset_3	0,02	0,29	153
SingleLinkage_offset_5	0,02	0,032	6
OverlappingBased_1+2	0,05	0,70	3
CoAssociationBased_1+2	0,02	0,53	3
InformationTheoryBased_1+2	0,02	0,47	3

6.2.4 Experiments with the same algorithm and different features

In figures 6-19, 6-20 and 6-21 we see that the ensemble of different features was able to improve the clustering results. However, this sacrifices in cost of the run-time. Table 6-7 summarizes the average run-time, the maximum run-time, and the number of correct classified images for each experiment. Also here we see that the computation time is much higher for the ensemble experiments than for the single experiments. We see also that the number of correctly classified images is less of the ensemble experiments. The reason for this is because the ensemble experiment contains more clusters compared to single clusters and this makes it hard for the classifier to select the correct DAB.

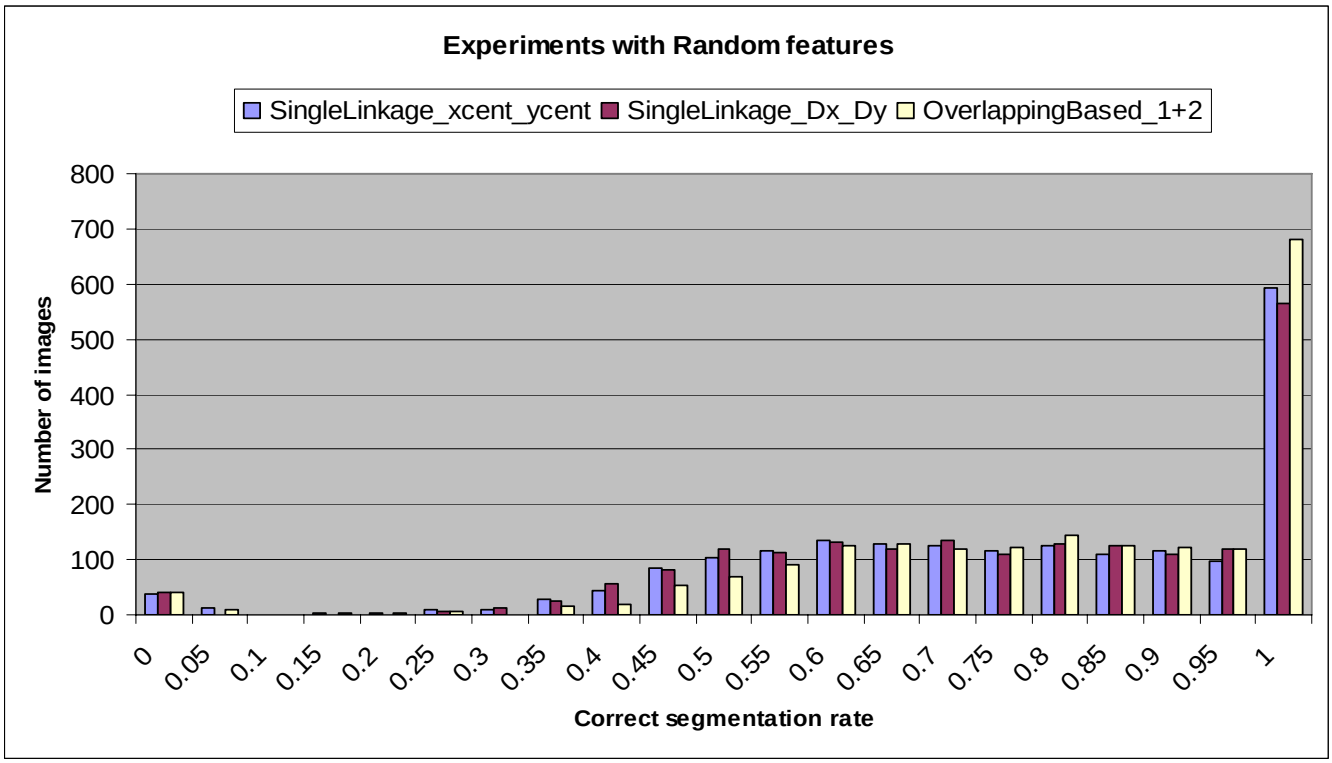


Figure 6-19: Histogram of the overlapping experiment 1 with different features on US dataset.

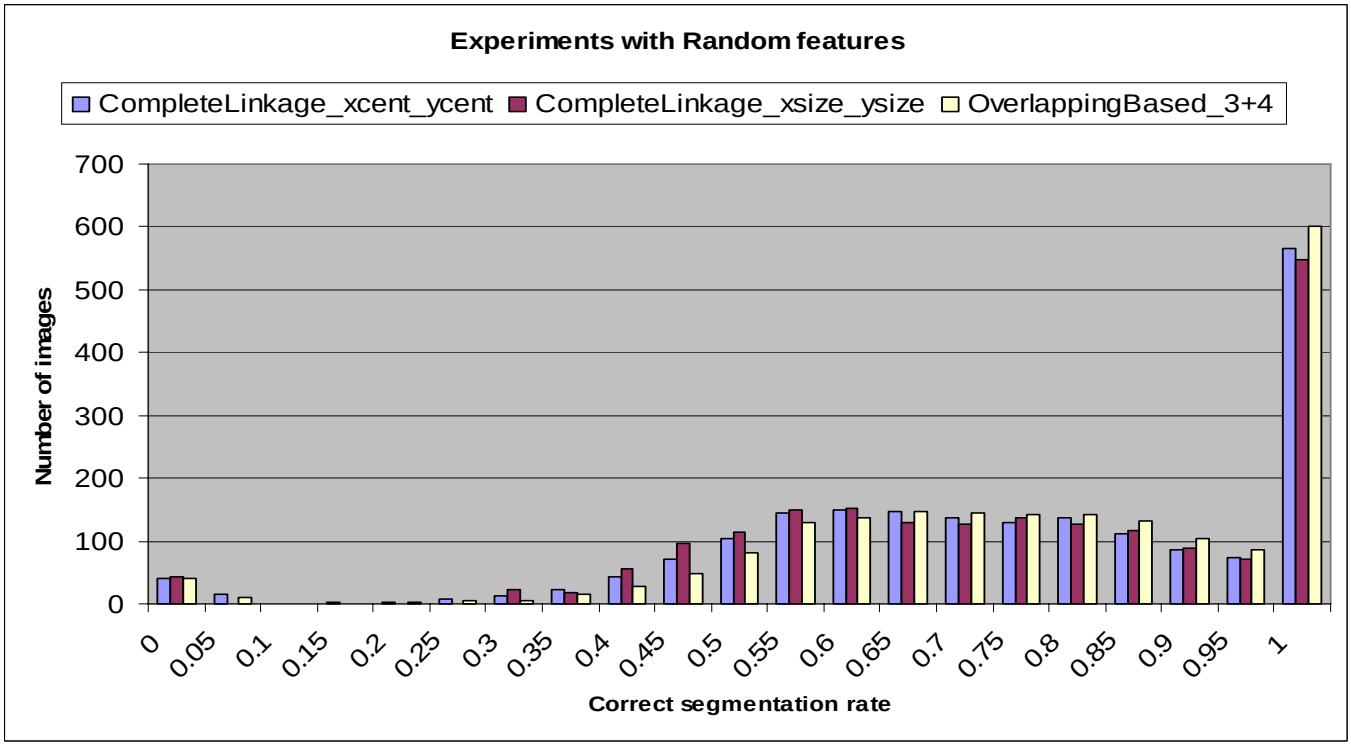


Figure 6-20: Histogram of the overlapping experiment 2 with different features on US dataset.

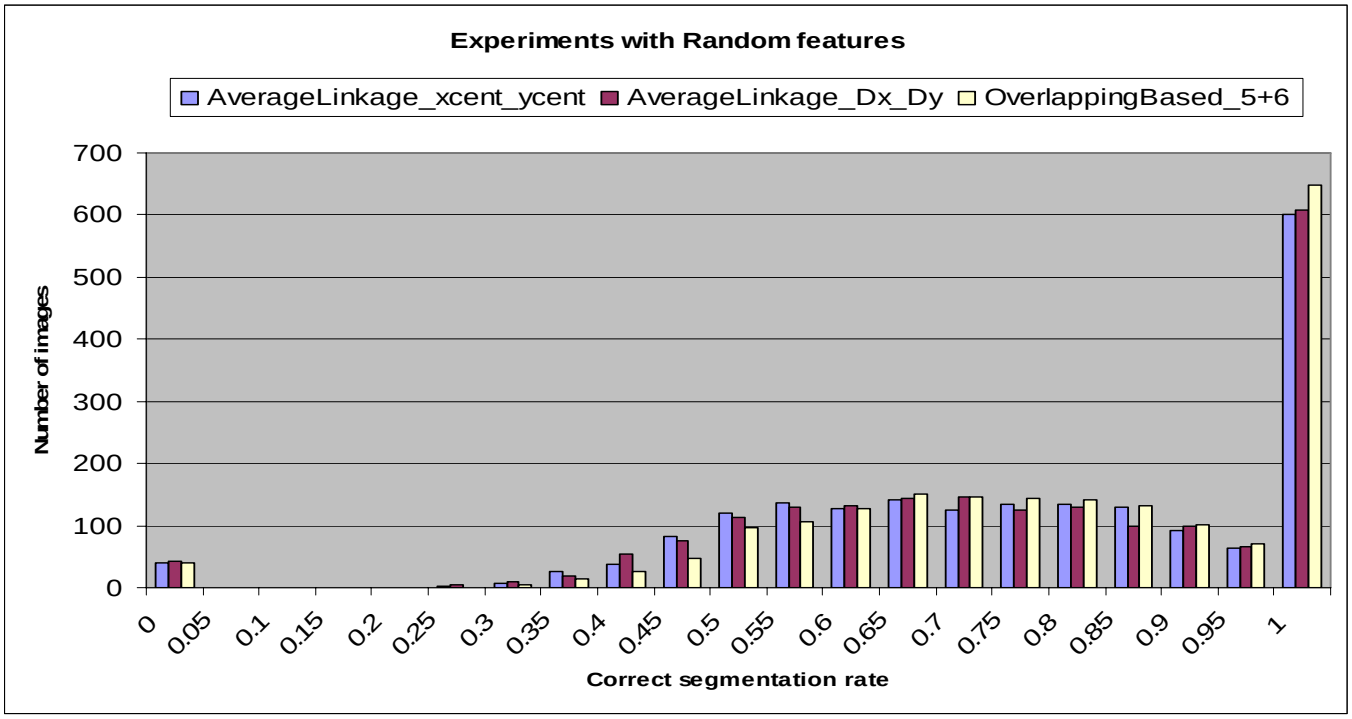


Figure 6-21: Histogram of the overlapping experiment 3 with different features on US dataset.

Table 6-7. Different features experiments on US dataset.

Experiment	Average run-time (seconds)	Maximum run-time (seconds)	Correct classified images
SingleLinkage_xcent_ycent	11,86	989,02	512
SingleLinkage_Dx_Dy	19,13	1011,76	504
CompleteLinkage_xcent_ycent	11,68	987,97	494
CompleteLinkage_xsize_ysize	18,93	1231,90	475
AverageLinkage_xcent_ycent	23,55	1247,51	502
AverageLinkag_Dx_Dy	18,87	1230,88	504
OverlappingBased_1+2	39,52	2396,98	275
OverlappingBased_3+4	43,11	2381,91	230
OverlappingBased_5+6	46,64	2629,24	245

6.4 EXPERIMENTS DISCUSSION

In this chapter we will discuss the experimental results. In the previous chapter we showed that hierarchical agglomerative clustering algorithms obtain better results than the partitional clustering algorithm. The reason for this is because partitional clustering algorithms request some priori knowledge, like the number of desired clusters. Since different mail piece images have different textures, different number of objects, and no fixed position for the address-block, postmarks and stamps it is difficult to predefine the number of desired clusters. As we have seen in the previous chapter our hierarchical clustering obtained good results close to the results of multi clustering algorithm. It is important to note that multi clustering algorithm was highly tuned over the years for address-block segmentation, whereas our algorithms are kept as general as possible; i.e. as they are described in the literature. With some tuning our hierarchical algorithms may be improved to obtain even better results than the multi clustering algorithm. Likewise the classifier provided better results for the multi clustering algorithms than for our algorithms. One of the reasons for this is that the classifier was trained using multi clustering algorithm. Another reason is that our hierarchical algorithms contain clusters that are almost similar to each other, which makes it for the classifier difficult to select the correct one. The high run-time has to do with the data structure usage. This means that multi clustering works direct on connect components data structure, whereas our algorithms use this data structure to create a new data collection. Another factor for the high rum-time is the merge process. The merge process for the multi clustering algorithm is based on some knowledge (i.e. only the clusters that satisfy some predefined knowledge are merged), whereas our hierarchical algorithms are implemented as described in the literature.

For the ensemble approach we have seen that not all diversity strategies are useful for address-block segmentation. However, we have seen that when the correct integration function is used it improved the clustering results. Therefore the choice of a suitable integration function for combining the base clusters will greatly affect the accuracy of the final clustering results. We obtained the best results with overlapping based integration function and hierarchical clustering algorithms as method for diversity, which are even better than the results obtained with the multi

clustering. The partitional clustering algorithms as method for diversity lead to a solution unattainable by the single partitional clustering algorithm, but unfortunately it was still below the results obtained with any hierarchical clustering algorithms. The reason for the less number of classified images has to do with even lager number of both clusters as well as the similar clusters. Unfortunately, the ensemble experiments have a higher run-time compared to the single experiments. The reason for this is that ensemble clustering experiments use a set of single experiments to generate the base clusters.

Chapter 7: Conclusion and suggestion for future research

This chapter gives a very brief summary about the main goal of this thesis and exposes conclusions according to the different stages that have been followed to obtain our goal: literature review, theory, design, implementation and evaluation. Finally we look at the opportunities for future research.

7.1 CONCLUSION

We started this thesis with the following research question:

"Can some single or ensemble clustering technique help to make the clustering process of the address-block locating more efficient and accurate?"

Based on this question, we formulated our research objectives. Those objectives consist both of a theoretical part and an implementation part. In order to achieve those objectives we started with a literature review of address-block locating, here the work of several authors that dealt with the problem of locating address-blocks on mail piece images was reviewed. We pointed out that the existing methods might be classified into artificial techniques, geometric characteristics and texture analysis by filtering. Afterwards, the information about clustering and ensemble clustering was gathered. Based on this information a brief description of clustering was provided. Next the components that make a clustering process were discussed. Finally ensemble clustering and its components were described. Based on this theory and analysis of the current functionality we designed our system needed for our research. With the design model and functional/non-functional requirements in mind the proposed system was implemented. First the system was divided into sub-systems and each sub-system was separately implemented and tested. Afterwards all the sub-systems were integrated to form a complete system and finally a user interface was provided in order to interact with the system. Finally, this system was used to evaluate several single/ensemble segmentation experiments on mail piece images. For this evaluation two different databases were used, one with 2000 images and another with 1600 images. Based on these datasets we defined

several single as well as ensemble experiments. According to the results of these experiments we can answer our research question as follows:

- Hierarchical based single clustering algorithms are the best candidate for our problem. However, they are not always efficient. Therefore some modification will be needed in order to make them more efficient.
- Ensemble techniques can improve the accuracy of the clustering process, especially with hierarchical clustering algorithms as method for diversity. But they suffer also from high-computational time.

Our main contribution in this thesis is to provide a system for the address-block segmentation that makes it possible to apply several single and ensemble segmentation experiments to a mail piece image, to evaluate results of different segmentation experiment based on the ground-truth segmentation, and to see the segmentation results of the segmentation experiments. We can conclude that this has been successfully accomplished.

7.2 FUTURE RESEARCH

There are many interesting questions in this context that are worth being addressed for further work. A few of them are listed below:

- Because of the time constraints we evaluated only some segmentation experiments. Therefore, it will be a good idea to evaluate more segmentation experiments using the implemented system.
- We have seen in this thesis that hierarchal-based clustering algorithms obtained better clustering results than partitional clustering algorithms. Therefore, it is a good idea to research more of those kinds of algorithms. Graph-based algorithms are suitable to hierarchal-based and are worth to be researched.
- Consider other integration functions that may be more successful in exploiting diversity. For example, for the diversity obtained with random data distance based integration function may be more suitable.

- A drawback of both hierarchal-based algorithms and ensemble clustering algorithms is the high computational time. An interesting future work is to research whether this computational time can be decreased.
- In this project we have seen that it is very difficult to select a suitable clustering algorithm. Therefore, it will be a good idea to use one of the available data-mining tools (e.g. Tanagra, CLUTO, Matlab, etc…), to assist selecting a suitable clustering algorithm. This will save a lot of time, because instead of implementing a clustering algorithm, which may or may not work, only some communication interface for the data-mining tool is needed. Using this it will be easy to evaluate several clustering algorithms for the address-block segmentation problem.
- Because the preprocessing stage may affect the clustering process, it may be worth researching whether this stage can be improved.
- Based on some experiments it turned out that position related feature overpowered the other features. Therefore feature normalization may be useful to lose this overpower. The simplest type of normalization can be obtained by subtracting feature with their means.

References

[1] T. Kagehiro, M. Koga, H. Sako, and H. Fujisawa, “Address-Block Extraction by Bayesian Rule”, ICPR (2): 582-585, 2004.

[2] S.N. Srihari, CH. Wang, P.W.Palumbo, and J.J. Hull, “Recognizing Address Blocks on Mail Pieces: Specialized Tools and Problem-Solving Architecture”, Al Magazine, pages 25-40, winter 1987.

[3] CH. Wang, P.W. Palumbo, and S.N. Srihari, “Object Recognition in visually complex environments: An Architecture for Locating Address Blocks on Mail Pieces”, Proc. Ninthe Intl. Conf. on Pattern Recognition, Rome, Pages 365-367, 1988.

[4] T. Liu, X. Ding, Q. Fu, and Z. Ren, “A Fast Algorithm of Address Lines Extraction on Complex Chinese Mail Pieces”, SPPRA: 38-43, 2006.

[5] S.H. Jang, S. Jeong, and Y. S. Nam, “Locating Destination Address Block in Korean Mail Images”, ICPR (2): 387-390, 2004.

[6] V. Govindaraju and S. Tulyakov, “Postal address block location by contour clustering”, ICDAR: 429-43, 2003.

[7] Y.K. Chen and J.F. Wang, “Locating the Destination Address Block on Images of Complex Mail Pieces”, Journal-chinese institute of engineers, VOL 24; PART 6, pages 761-770, 2001.

[8] K. Roy, S. Vajda, A. Belaïd, U. Pal, and B. B. Chaudhuri, “A System for Indian Postal Automation”, ICDAR: 1060-1064, 2005.

[9] K. Roy, U. Pal, S. Vajda, and B. B. Chaudhuri, “A System towards Indian Postal Automation”, In Proc. Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp.580-585, 2004.

[10] J. Facon, D.Menoti, and A. A. Araújo, “Lacunarity as a texture measure for address block segmentation”, Proceedings of the X Iberoamerican Congress on Pattern Recognition – CIARP 2005.

[11] L. F. Eiterer, J. Facon, and D. Menoti, “Postal Envelope Address Block Location by Fractal-Based Approach”, SIBGRAPI: 90-97, 2004.

[12] E. A.Yonekura and J.Facon, “Postal Envelope Segmentation by 2-D Histogram Clustering through Watershed Transform”, ICDAR: 338-342, 2003.

[13] E. A.Yonekura and J. Facon, “2-D Histogram-Based Segmentation of Postal Envelopes”, SIBGRAPI: 247-253, 2003.

[14] D. Menoti, D. L. Borges, J. Facon, and A. de S. Britto Jr.,”Segmentation of Postal Envelopes for Address Block Location: an approach based on feature selection in wavelet space”, ICDAR: 699-703, 2003.

[15] H.S. Ranganath and L.J. Chipman, “Fuzzy relaxation approach for inexact scene matching”, IVC 10, 631-640, 1992.

[16] S.N. Srihari, CH. Wang, P.W. Palumbo, and J.J. Hull, “Recognizing Address Blocks on Mail Pieces”, AI Magazine 8(4): 25-40, 1987.

[17] R. Wolf and J. C. Platt, “Postal Address Block Location Using a Convolutional Locator Network”, NIPS: 745-752, 1993.

[18] J. Wen, A. J. Baerveldt, and F. Ade, “Location of address labels on postal parcels using neural networks based on multifeatures”, industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON apos; 91., 1991 International Conference on Volume , Issue , 28 Oct-1 Nov 1991 Page(s):1480 - 1484 vol.2

[19] H. Walischewski, “Learning Regions of Interest in Postal Automation”, ICDAR 1999: 317-320, 1999.

[20] R.J.N. Kalberg, G.H. Quint, and H. Scholten, “Automatic interpretation of Dutch addresses”, ICPR92 (II: 367-370), WWW Version 9208 BibRef, 1992.

[21] TNT site, http://www.tntpost.nl/overtntpost/bedrijf/cijfers/index.aspx, WWW version 2007

[22] T. Kärkkäinen and S. Äyrämö, “Robust clustering methods for incomplete and erroneous data”, in Proceedings of the Fifth Conference on Data Mining, pp. 101–112, 2004.

[23] R.J. Little and D. B. Rubin, “Statistical analysis with missing data”, JohnWiley & Sons, 1987.

[24] A. K. Jain and R. C. Dubes, “Algorithms for clustering data”, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[25] R. Xu and D. Wunsch, “Survey of Clustering Algorithms”, IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645-678, 2005.

[26] P. Berkhin, “Survey Of Clustering Data Mining Techniques”, Tech Report (2002).

[27] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data Clustering: A Review”, ACM Computing Surveys, Vol. 31, No. 3, 1999.

[28] G. Karypis, E.H. Han, and V. Kumar, “CHAMELEON: A Hierarchical clustering algorithm using dynamic modeling”, IEEE Computer, vol. 32, no. 8, pp.68-75, August 1999.

[29] S. Kotsiantis and P.E. Pintelas, “Recent advances in clustering: A brief survey”, WSEAS Transactions on Information Science and Applications, 1(1): 73–81, 2004.

[30] V. N. Cherkassky and F. Mulier, “Learning from data: Concepts, Theory and Methods”, Wiley & Sons, New York, 1998.

[31] R. O. Duda and P. E. Hart, “Pattern Classification and Scene Analysis”, Wiley, 1973.

[32] V.N. Vapnik, “The nature of statistical learning theory”, Springer- Verlag New York, Inc., New York, NY, USA, 1995.

[33] A. K. Jain and M. H. C. Law, “Data Clustering: A User's Dilemma”, PReMI: 1-10, 2005.

[34] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu, “An efficient k-means clustering algorithm: Analysis and implementation”, IEEE Trans. Pattern Analysis and Machine Intelligence, 24 (2002), 881-892, 2002.

[35] A. Gionis, H. Mannila, and P. Tsaparas, “Clustering Aggregation”, in Proceedings of ICDE'2005 pp.341-352, 2005.

[36] Y. Qian and C.Y. Suen, “Clustering Combination Method”, ICPR 2000: 2732-2735.

[37] Z. He, X. Xu, and S. Deng, “Clustering Mixed Numeric and Categorical Data: A Cluster Ensemble Approach”, publication: http://www.angelfire.com/mac/zengyouhe/, 2002.

[38] A. Reynolds, G. Richards, B. Iglesia, and V. Rayward-Smith, “Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms”, Journal of Mathematical Modelling and Algorithms, Volume 5, Number 4, pp. 475-504(30), December 2006.

[39] H. Kargupta and P. Chan, “Advances in Distributed and Parallel Knowledge Discovery”, AAAI/MIT Press, Cambridge, MA 2000.

[40] M. Sezgin and B. Sankur, “Survey over image thresholding techniques and quantitative performance evaluation”, Journal of Electronic Imaging 13(1), 146– 165, January 2004.

[41] K. Wu, E. Otoo, and K. Suzuki, “Two Strategies to Speed up Connected Component Labeling Algorithms”, LBNL-59102, 2005.

[42] A. Strehl and J. Ghosh, “Cluster ensembles: A knowledge Reuse Framework for Combining Partitionings”, IEEE, 2004.

[43] B. Everitt, “Cluster Analysis”, Heinemann Educational Books, 1974.

[44] E. Diday and J. C. Simon, “Clustering analysis”, In K. S.Fu, editor, Digital Pattern Classification, pages 47–94. Springer Verlag, 1976.

[45] A. Strehl, “Relation-based Clustering and Cluster Ensembles for High-dimensional Data mining”, Phd thesis, the university of Texas at Austin, may 2002.

[46] B. Bruegge and A. Dutoit, “Object-Oriented Software Engineering”, Prentice Hall, 2000.

[47] T. Kowalski, “Borland Delphi 7”, Hanser Fachbuchverlag, Dec 2002.

Appendix A

PAPER

ADDRESS BLOCK SEGMENTATION USING

ENSEMBLE-CLUSTERING TECHNIQUES