
Copyright:


**DOI link to article:**

[http://dx.doi.org/10.1142/S0218001415530031](http://dx.doi.org/10.1142/S0218001415530031)

**Date deposited:**

22/05/2015

**Embargo release date:**

30 March 2016

This work is licensed under a Creative Commons Attribution-NonCommercial 3.0 Unported License
Camera-Based Whiteboard Reading for Understanding Mind Maps

Szlárd Vajda  
Lister Hill National Center for Biomedical Communications, National Library of Medicine  
National Institutes of Health, Bethesda, Maryland, USA  
szilard.vajda@nih.gov

Thomas Plötz  
School of Computing Science, Newcastle University, NE7 1NP, Newcastle upon Tyne, United Kingdom  
t.ploetz@ncl.ac.uk

Gernot A. Fink  
Department of Computer Science, TU Dortmund University, 44221, Dortmund, Germany  
gernot.fink@udo.edu

Mind maps, i.e., the spatial organization of ideas and concepts around a central topic and the visualization of their relations, represent a very powerful and thus popular means to support creative thinking and problem solving processes. Typically created on traditional whiteboards, they represent an important technique for collaborative brainstorming sessions. We describe a camera-based system to analyze hand-drawn mind maps written on a whiteboard. The goal of the presented system is to produce digital representations of such mind maps, which would enable digital asset management, i.e., storage and retrieval of manually created documents.

Our system is based on image acquisition by means of a camera followed by the segmentation of the particular whiteboard image focusing on the extraction of written context, i.e., the ideas captured by the mind map. The spatial arrangement of these ideas is recovered using layout analysis based on unsupervised clustering, which results in graph representations of mind maps. Finally, handwriting recognition derives textual transcripts of the ideas captured by the mind map.

We demonstrate the capabilities of our mind map reading system by means of an experimental evaluation, where we analyze images of mind maps that have been drawn on whiteboards, without any further constraints other than the underlying topic. In addition to the promising recognition results, we also discuss training strategies, which effectively allow for system bootstrapping using out-of-domain sample data. The latter is important when addressing creative thinking processes where domain-related training data are difficult to obtain as they focus on novelty by definition.

Keywords: camera-based document recognition, whiteboard reading, mind map recognition, handwriting recognition, document layout analysis

1. Introduction

The visualization of mental associations has a long history in a variety of challenging processes related to creative thinking and idea finding [2], most prominently used in brainstorming sessions. Examples of which are all kinds of learning processes and problem solving. Concept maps in general, and mind maps [2] in particular, are effective tools for transcribing, organizing and representing ideas and their relations. Mind maps are diagrams
of words, ideas, and tasks, which are linked to a general topic. The latter is typically the
centered root point in a radial graph, where nodes represent the conceptual entities —usually
short texts with a single or just a small number of words each— and connecting edges
visualize their relations.

Arguably, brainstorming is most effective if the participants of creative thinking meetings
can focus exclusively on the idea finding process. Therefore, collaboratively creating
mind maps using traditional means of pens and whiteboard still represents the standard
technique in brainstorming sessions (cf. Fig. 1). However, for archiving and retrieval a
digital representation of the document is usually desirable.

We present a camera-based whiteboard reading system, which processes hand-drawn
mind maps as they are typically created in brainstorming sessions. The system stays com-
pletely in the background, i.e., it does not interfere with the brainstorming process itself,
but provides a digital representation of mind maps. Based on our previous work in the
field [53, 36, 47, 50, 48], this paper is the first of its kind, which presents a complete
camera-based mind map reading system for real-world scenarios starting with the image
acquisition, and ending up with the complete recognition of the mind map.

Our mind map reading system consists of three modules, namely i) image segmentation,
ii) document layout analysis, and iii) word recognition.

Following basic image pre-processing, i.e., normalization and de-noising, the mind-
Camera-Based Whiteboard Reading for Understanding Mind Maps

(a) Image of a mind map hand-drawn on a whiteboard

(b) Digital representation of a mind map

Fig. 2: Example of automatic mind map analysis: Camera-captured image of a mind-map document (left) and desired output, i.e., digital representation of the mind map (right).

map image is segmented into basic document elements – i.e. textual and graphical items constituting the mind map, combining connected component extraction and a statistical classifier. Subsequently, the textual components identified are grouped into text patches using an unsupervised clustering approach, which effectively analyzes the mind map structure by recovering a graph representation of its nodes and edges. All node labels – i.e. the text patches associated with them – are then forwarded to an HMM-based, writer-independent handwriting recognition module, which provides the machine-interpretable transcription of the ideas conveyed by the analyzed mind map. All modules are integrated into a software framework, which represents the interface for both the automatic mind map analysis, and for digital asset management (archiving and retrieval).

In addition to the presentation of the technical contributions, we discuss strategies for robust system bootstrapping in scenarios, where –by definition– the amount of sample data is typically very limited. The latter is reasoned by the fact that brainstorming addresses the creation of novel ideas. Thus annotated training data is typically hard to obtain.

This paper summarizes the results of a long-term research endeavour. Consequently, parts of the results presented here have already been published individually in workshop and conference contributions [36, 47, 48] as well as invited extended versions thereof [50, 49]. This paper puts together the complete set of methods developed, proposes an improved approach to clustering handwritten document elements, and presents new text recognition results using a robust writing model trained on out-of-domain data.

2. Related work

Automatic mind map reading from whiteboard images represents a new application domain, which basically touches three research fields: i) basic image (pre-) processing, ii) automatic analysis of graphical structures, and iii) handwriting recognition.

Fig. 2 illustrates mind map analysis as it is addressed in this paper by means of the input data, which needs to be processed (left: mind map hand-drawn on a whiteboard), and the
Camera-based whiteboard reading needs to deal with the classical image acquisition, and pre-processing procedure as it is standard for computer vision applications. In particular this includes image de-noising, color and illumination normalization etc. [15]. More specific to the final recognition step is the pre-processing of the actual handwriting, where pre-processing operations are applied that attempt to normalize the appearance of the writing with respect to baseline orientation (frequently also referred to as skew), slant angle, and size (cf., e.g., [11]).

Handwritten documents can contain a variety of items such as text blocks, lines, words, figures, tables etc. The primary goal of document structure and layout analysis is to detect these different regions and to identify their functional roles and relationships [32].

Somewhat related to the analysis of mind maps, the recognition of line drawings aims to recover the high-level design from engineering drawings, e.g., to recognize pipes, lines, roads, or rivers in maps [45]. Similar to the use-case addressed in this paper, a limited and well defined set of graphical items needs to be recognized, which includes segmentation and classification. For engineering drawings text / graphic separation is not straightforward since text portions overlap with other objects in the images [46, 45]. Another important application domain for layout analysis is automated document analysis, where the functional elements of documents need to be detected and identified. For example, in postal automation letter envelopes are typically analyzed aiming for the separation of the address block and the stamp [51, 39]. In [31] the structure of business cards is unveiled, whereas [39] addresses the segmentation of legal documents.

The pre-dominant approach for virtually all layout detection and segmentation applications is based on the analysis of connected components (CC) [9]. The detection of connected components is typically based on blob analysis in raw image data employing some form of image segmentation approach [15]. The actual classification of CCs is usually performed by means of straightforward threshold comparison based on certain geometric features like height, width, aspect ratio, pixel density, number of horizontal and vertical segments [46, 31, 9, 34, 17].

For printed text detection a larger variety of methods is available [19]. Some of them use texture, some of them color, while some others are region based. To detect regions Gao et al. [13] employ visual attention models. Even though characters could be not salient, but regions containing text are salient, therefore they apply a second pixel-based filtering after the extraction of global salient regions. Maximally Stable Extremal Regions (MSER) [30] were considered in [17, 55] to detect possible candidate regions by pruning the MSRE tree by filtering out the unlikely regions based on color, size, aspect ratio, and number of holes. A method based on transfer learning involving Adaboost [14] was successfully evaluated on ICDAR2001 scene text detection competition dataset involving windows classification based on Histogram of Oriented Gradient (HOG), number of extended edges in the image, average and variance of stroke width, local binary patterns (LBP), etc. Color distance, color variance is also considered [55] beside spatial distance among possible textual components characterized originally by HOG. Similar attempt to use color is considered by Phan et al. [34] to group together possible candidates. A k-means (k=3) based on color was applied by
Zheng et al. [56] to decide for each pixel, if it belongs to text foreground, to the background or to the noise class. Edge-based methods were considered using different color bands in [34]. An uncommon, but rather interesting method based on skeleton structure classification is proposed in [44], which seems to work up to 70% accuracy for text/non-text separation.

The automatic recognition of handwritten script has been subject to both industrial and academic research for more than 50 years [1, 12] resulting in a variety of approaches and systems for both on-line and off-line processing. The latter is the technological basis for mind map reading as it is addressed by this paper. Images of handwriting –captured after the text has been written– are analyzed with the objective to unveil a textual transcription [35]. Offline handwriting recognition (HWR) techniques follow either a holistic approach, where isolated words are analyzed as a whole (cf. [1] and the references therein). Alternatively, and more widely used, the recognition is performed on character level, which is either based on explicit segmentation (employing all imaginable varieties of pattern recognition techniques, e.g., using neural networks [16]), or performed in a segmentation-free manner. In the latter case temporal models –most prominently Markov Models [38]– are applied.

Automatic whiteboard reading has first been proposed in [54]. Using camera-captured whiteboard images the task has been approached as a special kind of offline handwriting recognition problem combining robust pre-processing and feature extraction methods with Markovian models for representing the appearance and the linguistic structure of the texts to be recognized [53].

The problem of analyzing whiteboard documents has also been also tackled in constrained settings [41, 43] and by using specialized sensing equipment [20, 22].

The Brightboard system [43] continuously observes the whiteboard and grabs a suitable image when the movement of the writer has completed. The image is than analyzed to detect and recognize special marks which can control a computer. A similar system is proposed by the ZombieBoard system [41] which scans the whiteboard for special marks and their corresponding commands using an active camera and a mosaicing algorithm.

Using a pen-tracking device and analyzing pen trajectory data in contrast to images of handwritten script the problem can be significantly simplified at the price of requiring a specialized hardware setup. The approaches presented in [20] and [22] both make use of hardware solutions for pen-tracking. In [20] thus an online recognition approach can be applied for the recognition of text written on a whiteboard. In [22] a multi-touch table is coupled with a pen providing self-tracking capabilities in order to manipulate objects and annotate content. The main drawback of such approaches is the strict requirement of special and costly hardware which can not be easily found in a usual meeting room. Therefore, the usability of the systems is limited to quite special settings.

3. Camera-Based Mindmap Reading

Camera-based whiteboard reading is an extremely challenging task, and can still be considered an open research problem. Our previous work in the field was focused on developing fundamental techniques for offline recognition of whiteboard documents [53, 36, 47, 50, 48]. This previous research also showed that camera-based whiteboard-reading can realistically
be considered as an offline document recognition problem only. The primary reason for this is that whenever whiteboard-documents are created by naive users in realistic scenarios, the pen used for writing is hardly ever visible to an observer. Therefore, camera-based pen-tracking as applied in [6] is not feasible, and the images of the whiteboard content - or patches thereof - have to be processed as offline documents.

In this paper, we focus on the recognition of images of mind maps handwritten and hand drawn on whiteboards addressing the following three central challenges. First, though the task of mind map recognition is well defined on a semantic level, there still exist hardly any constraints with respect to the layout of the considered documents, that could be robustly used for detecting and identifying elementary document units as, e.g., textual items or arrows connecting nodes in the mind map. Consequently, we use machine learning techniques for detection and unsupervised clustering methods for grouping of elementary units in the documents. Secondly, as it is common for special recognition tasks which have not become mainstream yet, we are faced with the fact that only a quite limited amount of domain specific document samples are available. Therefore, for model training we employ large data sets of handwritten material that are related to the task, but are neither directly from the same domain nor of the same rather low document image quality, as it is considered here with camera-captured documents. Thirdly, we investigate methods for increasing the generalization capability of a handwriting recognizer. This can be seen as a supporting measure in order to deal with the fact that training is achieved on out-of-domain data.
3.1. Proposed approach

We chose to base our method for mind map image segmentation on connected component analysis as this representation is rather well suited for handwritten documents, and is quite widely and successfully used in the document analysis field (cf. e.g. [9]). However, the main drawback of these methods usually is the presumption of a certain amount of well organized, well structured text / graphics material, which can serve to build rule-based strategies for distinguishing text from non-text and for identifying different document items. (cf. [46, 9, 3]). Color could have been considered as quite a strong clue to segment text/non-text regions, but in our mind map scenario usually only one marker was used during the creation process for text and non-text alike, therefore we have not seen the importance of the usage of the different color channels. This could have lead us to heuristics such as applied in [17, 56, 34] or even leading us to confusions. In order to avoid as much as possible such heuristics, our method is based on the use of a statistical classifier, namely a neural network for distinguishing between relevant textual and graphical items. The main advantage of such a machine learning approach is that the model can be estimated on annotated sample data automatically*

As mind map documents do not follow a well defined layout structure and may show large variations in format and style, simple layout analysis techniques, as, e.g., profile-based methods, will fail completely. Therefore, it is necessary to use a modeling approach that is able to flexibly adapt to the actual document layout observed. Consequently, we proposed to use methods based on unsupervised learning, as such techniques – to some extent – are able to automatically discover structure inherently given by some set of patterns without requiring prior knowledge about the data to be given (cf. [26]).

In the case of mind map images the main goal of document layout analysis is the discovery of groups of textual elements forming larger text patches. In order to overcome the limitations of the text patch grouping using hierarchical clustering based on the Euclidean distance in our previous work [48] we propose to apply an adaptive clustering to the textual elements identified in the segmentation step. The adaptive clustering is realized by combining a Growing Neuronal Gas (GNG) [10] for the extraction of dense regions of textual items and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [40] for extracting text element clusters that are considered as text patches in subsequent processing.

The text patches identified by the automatic layout analysis are assumed to correspond to the node labels of the mind map considered. In order to recover a digital mind map representation these text-patch images have to be transcribed using a suitable handwriting recognizer. Based on experience acquired in our previous research and following a general trend towards such methods in the document analysis field (cf. [38]) we apply a segmentation-free approach based on hidden Markov models (HMMs). As all machine learning methods this approach offers the advantage that model parameters can automatically be learned from sample data but, consequently, also requires a sufficiently large set of training data to be

*A preliminary version of this work has been published in [50].
available. Unfortunately, in our scenario domain specific training data is not available in the necessary quantity for training a general handwriting recognizer. Therefore, we use out-of-domain data for that purpose.

3.2. Overview

All these proposed strategies are implemented in an integrated software framework, and allow for an automatic reading of handwritten mind maps. Based on a camera image of a hand-drawn mind map a digital representation of the mind map document can be created which can serve as the starting point for further automatic document processing and analysis on a symbolic level. An overview of the automatic mind map reading process is shown in Fig. 3.

In the remainder of this article we will first describe in detail the image acquisition and segmentation approach in section 4. In section 5 our unsupervised approach to the layout analysis of mind map documents will be presented. Afterwards, the development of an offline handwriting recognizer for camera-captured mind map documents will be explained in section 6. The evaluation of the proposed approaches on a challenging data set of mind map images will be presented in section 7, and the results obtained and the implications for future research will be discussed in the final section.

4. Image segmentation

The image segmentation is meant to separate elements written on the whiteboard with a marker from the whiteboard background and noisy image parts, followed by categorizing the written content into different mind map elements (e.g. text, line, circle, arrow).

4.1. Segmentation of the camera image

After the image acquisition by the camera, the objective is to extract only the written content from the whiteboard. This relevant information is then added to a binary region memory (cf. Fig. 1). The region memory represents the current state of written content on the whiteboard and it is robust to changes in the camera image, like illumination or particular users standing in front of the camera. Therefore, the general assumption is that the camera image does not contain anything but the interior of the whiteboard and the camera and the whiteboard are fixed. The system handles images that can consist of three different regions, namely: i) text (indicated by bright blocks in Fig. 1), ii) background (indicated by dark blocks in Fig. 1), and iii) noise (indicated by blocks with grid pattern in Fig. 1).

The applied segmentation is an implementation of the original method proposed in our group by Wienecke et al. [54]. The segmentation is not performed on pixel but rather on block level, to provide a certain robustness w.r.t. to illumination changes and other minor changes in the whiteboard scene. The image is therefore divided into two layers of overlapping blocks. Each block is segmented into one of the formerly mentioned categories based on three features: gradients, gray level and changes between two consecutive images. Based on these measures and the corresponding thresholds estimated on some trial runs, the
whiteboard content is successfully separated from the rest of the scene. For further details please refer to [48, 54].

4.2. Connected component extraction

The upcoming step is the extraction of the connected components. Disregarding probable flaws in the image (e.g. inhomogeneous lighting, or non-opaque marker color) separating the mind map by connected component analysis is reasonable, and no prior knowledge about the whiteboard content is necessary. This choice is motivated by the fact that instead of using complex skeletonization and curve tracing procedures, this system manipulates connected components which can reliably be extracted without any heuristics.

For the CCs extraction the image is binarized first, using Niblack’s local method [33], considering a variant which applies threshold optimization [42], and local thresholding in a 51x51 pixels windows using integral images [52] for efficiency. The height, width, aspect ratio and pixel density of the CCs [9] serve as selection criteria for filtering. Small CCs (5x5) containing only a few pixels or rather large components (70% of the original image height or width) like whiteboard borders are discarded.

4.3. Connected component classification

After the filtering the remaining items (CCs) are classified into: i) text, ii) line, iii) arrow, and iii) circle.

In order to classify the CCs, a feature vector (shape set) composed by 12 measures (i.e. contrast, edge density, homogeneity, number of foreground gray levels, foreground mean gray level, relative amount of gradient orientations, Sobel gradient orientation and magnitude, etc.) is derived to characterize each component. Alternatively, intensities of gradient histograms (values ranging from 0 to 255, equally divided into 16 bins) of the connected components serve also as another type of features (gradient set). Due to the limited dimension of the feature vector, 12 and 16 respectively, the authors have not seen the interest to perform any feature selection strategy. The nature of each feature is different, therefore no high correlation is to be expected.

A Multi-Layer Perceptron (MLP) with one hidden layer is used to perform the classification. No prior knowledge about possible correlations between the extracted feature components has been considered. In consequence, a fully-connected network topology was used, and trained with classical error back-propagation. The number of neurons used in the input layer was defined by the dimensionality of the feature vectors, which is 12 and 16 respectively. Due to the fact that 4 classes were to be identified (i.e. text, line, arrow, circle), 4 neurons were considered in the output layer, and 15 and 20 neurons were used in the hidden layer. The number of hidden neurons was established by several trial runs [50], involving 5,10,15,20,25 neurons. For learning rate we used 0.0001, while for the momentum we considered 0.3.

A complete formal description of the feature set can be found in [50]
5. Document layout analysis

Knowing that characters usually appear closer to each other than the other elements, by clustering they should group with their kind rather than with non-text elements. For this reason we discard in the further processing all non-text elements from the document, and focus the attention only on text snippets.

5.1. Unsupervised layout modeling

Instead of using dendrogram analysis built by single linkage clustering to group the nearest components [47, 50], --our first attempt to tackle this problem --, in this paper a totally new idea is proposed, namely the layout modeling by a self organizing neural network [10, 18]. The aim is to adapt on the fly the modeling to varying layouts, sizes and orientations of scripts.

The different text CCs’ gravity centers, can be represented in a two dimensional Euclidean space. The goal is to model these coordinates with a self organizing network.

The main advantage: There is no need for any labeled training data. The network adapts its neurons and their spatial arrangements to the topology (gravity centers of CCs) of the current document based on competitive learning. Another advantage over other methods (e.g. k-means): There is no need to specify beforehand the number of clusters as the number of words (clusters) differ from one document to another.

The GNG has $N$ units (neurons), each one characterized by its coordinates ($w$), the accumulated error ($\delta_{error}$), the edges between itself and other units and the age of the edge.
Algorithm 1 The GNG applied to the text components.

Require: Init $a, b$ units with random positions $w_a$ and $w_b \in R^2$
Ensure: Init $i = 0$
repeat
    Pick randomly $\epsilon_i$ (one gravity center) $\in R^2$
    $i = i + 1$
    Find nearest and second nearest units $s_1, s_2$
    Increment age of edges for all units emanating from $s_1$
    Calculate error $\delta_{error}(s_1) = ||w_{s_1} - \epsilon||^2$
    Update $\Delta w_{s_1} = \epsilon_b (\epsilon - w_{s_1})$
    Update $\Delta w_{s_2} = \epsilon_n (\epsilon - w_{s_2})$ all direct neighbors
    if $s_1$ connects with $s_2$ then
        Set age of the edge $= 0$
    else
        Set an edge between $s_1$ and $s_2$
    end if
Remove all edges, where age $> age_{max}$
if $mod(i, k) == 0$ then
    {Insert a new unit $r$ if still available}
    Find unit $q$ with $max\{\delta_{error}(s_i)\}$
    Find unit $v$ with $max\{\delta_{error}(s_i)\}$ among $q$’s neighbors
    Insert $r$ between $q$ and $v$ and edges
    Insert edge between $q$ and $r$ and $v$
    {Updates for $u, v, r$}
    $w_r = (w_q + w_v) \times 0.5$
    $\delta_{error}(u) = \delta_{error}(u) \times 0.5$
    $\delta_{error}(v) = \delta_{error}(v) \times 0.5$
    $\delta_{error}(r) = (\delta_{error}(u) + \delta_{error}(v)) \times 0.5$
end if
Decrease error for all units $\delta_{error}(s_i) = \alpha \times \delta_{error}(s_i)$
until $(i < N)$ and $(\sum_{k=1}^{N} \delta_{error}(u_k) < \Omega)$

controlling the topology of the GNG network. The $age_{max} = 80$, controls the maximal time to have an edge between nodes, $\epsilon_b = 0.25$ and $\epsilon_n = 0.0002$ are learning constants, while $\alpha = 0.0005$ serves as a learning factor to diminish the error in the units all over the network. The parameter $k = 100$ controls the speed of growth in the network. The algorithm stops when all the available units are consumed in the network building, and the error is below a certain threshold $\Omega = 0.1$. A short description of the GNG algorithm applied on the gravity centers of the extracted text components is given in Alg. 1. For more details w.r.t. the algorithm, please refer to the seminal work published by Fritzke [10].

The modeling performances of GNG and SOM can be seen in Fig. 4. The gravity centers of the textual items (see Fig. 4b) are approximated. The GNG’s modeling capability (see
Fig. 4d) overcomes the more noisy representation provided by the SOM (see Fig. 4c). The noise introduced by the SOM is due to the fact that in this network the neurons remain always connected, while for GNG the connections might be annulled by removing the edges between different nodes (see Alg. 1). The number of units considered for the modeling is directly proportional to the number of CCs to be modeled. The number of units should be higher (in our case $N = 5 \times$ (number of CCs)) in order to exploit the capabilities of the density clustering which follows. In Fig. 4d a clear distinction can be observed around the different words. Near the words a huge agglomeration of nodes can be spotted, –creating already a visual distinction between the different words in the document.

5.2. Unsupervised word grouping

After the modeling process the different nodes (neurons) are merged into different clusters to establish words. For this purpose the Density Based Spatial Clustering of Applications with Noise (DBSCAN) was considered, a partitioning based clustering method proposed originally by Ester et al. [4].

Instead of some classical clustering based only on distance measures (e.g. hierarchical clustering [50]), this approach considers not only the distance between elements, but also the density of the elements in a certain neighborhood. This allows to recognize the word clusters because within clusters there is a high density of point agglomerations (see Fig. 4d), a considerably higher than outside the clusters.

In the DBSCAN a point $p$ belongs to a cluster, if and only if in the neighborhood of a given radius contains at least a minimum number of points, i.e. the density of the given neighborhood exceeds some threshold. The shape of the neighborhood is defined by the distance measure considered. In our scenario the Manhattan distance will provide a rectangular neighborhood, while the Euclidean distance will define a circle. The latter metric was considered in our clustering task due to the unconstrained layout of the documents.

The reason of an increased number of points (GNG units) considered for the modeling purpose as observed in Fig. 4d is motivated by the density. Instead of clustering the original points (see Fig. 4b), the DBSCAN clusters rather the GNG units (see Fig. 4d) which form dense regions around the words.

The algorithm described briefly in Alg. 2 starts with an arbitrary point $p$ and retrieves all the points density reachable from $p$ w.r.t. $\epsilon$, measuring the neighborhood and $\text{MinPoints}$, counting the minimum number of points necessary to form a cluster. If $p$ is a core point the procedure develops a cluster. If $p$ is a border point, no other points are density reachable from the reference point $p$, thus the algorithm starts to analyze other points in the data set. The parameter $\epsilon$ and $\text{MinPoints}$ are estimated automatically, based on a “sorted $k$-distance plot” [40]. Finally each point in the data set is labeled with a specific cluster identifier. The description of the $\text{ExpandCluster}$ is beyond the scope of the current paper, therefore, for further details please refer [4].

Finally, we map the original data (see Fig. 4b) into this newly created cluster space generated by DBSCAN. Each original data point is labeled accordingly to the nearest unit in the GNG modeling (see Fig. 4c).
Algorithm 2 The DBSCAN algorithm applied to GNG units.

Require: \( p_i \) \( i = 1, 2, \ldots, N \), \( p_i \in \mathbb{R}^2 \) (GNG units)
ClusterID = NextID(Noise)

for \( i = 1 \rightarrow N \) do
  p = SetOfPoints(i)
  if p not clustered then
    if ExpandCluster(SetOfPoints, p, ClusterID, \( \epsilon \), MinPoints) then
      ClusterID = NextID(ClusterID)
    end if
  end if
end for

To reproduce the same topological structure of the mind map document a graph representation of the mind map is generated (see Fig. 3). Text and circle elements are treated as nodes and lines. Arrows are treated as edges. No prior knowledge of which nodes are to be connected by a line, an estimation is necessary. Line components are determined through intersection of the parametric line with neighboring components.

6. Word recognition

The goal of the word recognition stage is the transcription of the text patches extracted during layout analysis, and thus the recovery of the handwritten labels of the nodes within the mind-map documents. As quite commonly and successfully used in the field of handwriting recognition, we apply a segmentation-free recognizer based on HMMs which is developed using the methods and tools provided by the ESMERALDA framework.

In order to be able to train a general purpose handwriting recognizer large amounts of handwriting data need to be available. Therefore, it was clear from the beginning that for our task the handwriting recognition model would need to be trained on out-of-domain data. Though this seems to be technically quite straightforward, an important prerequisite of this approach is that from images of handwritten script of potentially different sizes and pixel resolutions compatible feature representations are obtained. This is true for the geometrical feature representation that we use in our work on handwriting recognition, which was inspired by the feature set proposed by Marti & Bunke. The most important aspect enabling this transferability of feature representations between handwriting data of different resolutions is a quite robust normalization of the apparent size of the writing based on an estimation of the average distance between local minima of the script’s contour which is closely related to an estimate of the average character width.

However, using a compatible feature representation alone does not ensure that recognition models can be transferred successfully between domains. In our first experiments on the recognition of handwriting in camera-captured mind-map images the recognition model was trained on scanned document images, i.e., quite clear and high quality data. This

\(^{a}\) The training data was taken from the well-known IAM database of handwritten English sentences.
model performed rather poorly on the actual task data, a fact that motivated the following modifications to the training of the recognition model.

6.1. Out-of-domain data

Instead of using handwriting material written on paper, and scanned later on to train the recognizer, we decided to investigate the use of data which was also written on a whiteboard, and thus might show similar variations in writing style to the mind-map documents. Such material is available from the IAM Online Handwriting Database [21]. Unfortunately, it consists of on-line data, i.e., pen trajectories recorded by a pen-tracking device. However, offline versions of the data can be rendered with characteristics quite similar to clean offline representations of data from the same source (cf. [23, 36]).

6.2. Context size reduction

As the script images obtained from mind-map documents frequently show artifacts caused by errors in the text extraction and patch grouping stages (see Fig. 6), we decided to reduce the size of the analysis windows extracted when serializing the text-patch images by the sliding-window method (cf. [38]). In our previous works we obtained the best performance especially on scanned handwriting data with a width of 8 pixels. For less sensitivity to image noise we reduced this size to 4, which is the minimum required by our feature extraction method\(^1\).

6.3. Recognition models with reduced complexity

Unfortunately, there is strong evidence that in the case of camera-based recognition of handwritten mind maps with a model trained on data rendered from pen trajectories — even though the handwriting material was also written on a whiteboard — there will be a significant mismatch between the characteristics of training and testing material. Especially in the speech recognition area such mismatch situations usually are tackled by model adaptation strategies. Unfortunately, such an approach again requires manually or automatically annotated adaptation material and, therefore, would reduce our test material to an insignificant size.

Therefore, we decided to explore the use of lower-complexity models instead of model adaptation. In our standard handwriting recognizer for Roman script we use semi-continuous HMMs with a codebook of $2k$ diagonal-covariance Gaussian densities and a set of 80 basic character models with Bakis topology\(^2\). In conjunction with the heuristic to initialize model lengths proportionally to the minimum length of the associated basic unit in the annotation of the training data this configuration leads to a quite complex writing model with approx.

\(^1\)Among others, after image binarization orientations of contours within the analysis windows are estimated, which does not make sense with window widths below 4 pixels.

\(^2\)Bakis models allow self-transitions between states, transitions to the successor state, and the skipping of a single state within a linear sequence.
Table 1: Overview of the data sets used in the experiments detailing the content, the size of the collection and the training/test ratio and the number of writers.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Content</th>
<th>Size</th>
<th># Writers</th>
<th>Training / Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>MindMap</td>
<td>mind maps</td>
<td>30 doc.</td>
<td>11</td>
<td>19 / 11 doc.</td>
</tr>
<tr>
<td>IAM-OnDB</td>
<td>online notes</td>
<td>1,700 doc.</td>
<td>221</td>
<td>5,034 lines / div. sets</td>
</tr>
<tr>
<td>IAM-DB</td>
<td>scanned pages</td>
<td>1,539 doc.</td>
<td>657</td>
<td>6,161 lines / div. sets</td>
</tr>
</tbody>
</table>

2400 model states and a similar number of mixture densities for modeling the respective outputs. In order to reduce the model complexity we investigated both the use of a reduced number of states per basic unit in conjunction with a restriction of the model topology to a linear one and the reduction of the codebook size to $1k$ Gaussians only. Additionally, we investigated a kind of “early-stopping” technique, i.e., we used models after a limited number of re-estimation steps for recognition instead of the ones showing maximum performance on the validation set belonging to the database used for training.

7. Evaluation
This section is dedicated to the evaluation w.r.t. the different system modules described in the previous sections. After the description of the data sets, results of the different modules will be presented.

7.1. Data description
The mind map collection consists of 30 photos taken from mind-map drawings. 11 different writers were asked to freely draw one mind map for each of the topics: i) “holiday”, ii) “party” and iii) “study” (1 writer only sketched 2 and one writer sketched only 1 mind map). The writers were provided with a standard whiteboard marker set, containing four different colors (black, blue, green, red) and a whiteboard eraser. Except for a basic set of words for each topic, which had to be used, and an obligation to add at least 3 other words to the mind map, there were no restrictions in the creativity producing these documents. After a writer had finished his mind map, a photo of the whiteboard was taken with a digital camera set to a resolution of $2048 \times 1536$ pixels (see Fig. 2a). All images were annotated with respect to text, lines, circles/ellipses and arrows. A single annotation corresponds to a rectangle stating the bounding box of the particular graphical element. In the case of text patches the annotations were made on a word-basis. 19 documents were considered for training purpose, while the remaining 11 documents served as test documents. The reduced number of documents is due to the confidential aspect of such mind maps.

Due to the reduced number of mind maps available for test purpose, some well-known data sets were also considered. These data sets do not contribute in the evaluation framework, they serve only the purpose of training our handwriting recognizer. The IAM-OnDB [21]

1 In linear models only self-transitions and transitions to successor states are allowed.
Table 2: Document specific results of connected component classification into text/non-text (shape set).

<table>
<thead>
<tr>
<th>Doc. Id.</th>
<th>Accuracy[%]</th>
<th>Doc. Id.</th>
<th>Accuracy[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97.6</td>
<td>2</td>
<td>92.2</td>
</tr>
<tr>
<td>3</td>
<td>97.7</td>
<td>4</td>
<td>97.3</td>
</tr>
<tr>
<td>5</td>
<td>95.6</td>
<td>6</td>
<td>94.8</td>
</tr>
<tr>
<td>7</td>
<td>82.3</td>
<td>8</td>
<td>96.0</td>
</tr>
<tr>
<td>9</td>
<td>96.8</td>
<td>10</td>
<td>96.4</td>
</tr>
<tr>
<td>11</td>
<td>97.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

is an on-line large sentence database. This on-line data was rendered in order to produce similar quality data as encountered in the mind maps. The IAM-DB is a large English off-line database [29]. A short summary of the data sets can be found in Tab. 1.

7.2. Separation of text and non-text items

The overall classification accuracies for the shape set and the gradient set are 95.7% and 93.0%, respectively [47]. It can be observed that both types of features are suitable to separate text and non-text items. However, the shape features set produces better results. For the sake of clarity, the further results will be limited only to this feature representation. The particular classification scores for each test document (11) can be seen in Tab. 2. The worse results reported for document id. 7 can be explained with the fact, that the quality of the document is not satisfactory and many items (lines, circles) were interconnected during the drawing process, hence huge connected components were extracted and analyzed which lead to significant errors. This particular document contains also some line structure not available in the other documents.

While the text items are recognized with a high precision (98.5%), the arrows are often confused with lines. This confusion can be explained by the fact that just a few arrows are represented in our data set, and there is not much difference between lines and arrows. Similar problems can be encountered for circles, which can be erroneously confused with text items, e.g., "o", "D". Vertical lines are often considered as text snippets, and the other way around. Small text components or letters like "i" or "l" are identified as being lines. Another type of error can be observed once lines touch circles or circles touch letters. In that situation, –due to the nature of the method (CC base recognition), such components are usually miss-recognized (see Fig. 5).

7.3. Clustering of text patches

For the evaluation of the proposed method we use the method introduced in the context of the ICDAR 2005 Text Locating Competition [24]. The evaluation scheme is based on precision and recall [5], deriving these measures from the bounding box coverage between the ground truth document and the analyzed one.
The bounding boxes of the annotated ground truth $T$ and the agglomerated text components $E$ are compared – the larger the overlap of the bounding boxes, the higher the level of match. A match $m_p$ between two rectangles $r, r'$ is defined as the quotient of their intersection area and their union area:

$$m_p = \frac{A(\bigcap (r, r'))}{A(\bigcup (r, r'))}.$$  \hfill (1)

Having a binary answer to whether there is a fitting ground-truth rectangle to an estimated one or not would not cope with partial matches. This is why the quality for a single match $m_p$ in this case lies in the range of $[0, 1]$. In order to calculate these adapted versions of precision and recall the best match between a rectangle within the agglomerations and all rectangles within the set of annotations is taken into consideration – and vice versa. The best match $m(r, R)$ of a rectangle $r$ within a set of other rectangles $R$ is defined as:

$$m(r, R) = \max \{ m_p(r, r') | r' \in R \}. \hfill (2)$$

The recall then is the quotient of the sum of the best matches of the ground truth among the agglomerated areas and the number of all annotated bounding boxes within the ground truth.

$$\text{recall} = \frac{\sum_{r_t \in T} m(r_t, E)}{|T|}. \hfill (3)$$

The precision relates to the quotient of the sum of the best matches of the agglomerated areas among the annotated regions and the number of all agglomerated areas:

$$\text{precision} = \frac{\sum_{r_e \in E} m(r_e, T)}{|E|}. \hfill (4)$$

We evaluated the output of the agglomeration (clustering) using both schemes described above. In the Tab. 3 a detailed list can be found for each document extended also with the F-measure, directly inferred from the precision and recall [5]. The worse score is produced by the document 7, which was identified as failure for the text separation case too. The low results can be explained by the fact that the clustering ends up in some huge components, which do not match anymore the items available in the ground truth document.

The main error source for clustering in general is due to the non-text patches labeled as text or minor text components filtered out based on their size or their confusion with graphics, hence gaps (splitting words) or partial words will be detected. While in some cases, the agglomeration is successful, in some other cases it fails because of some CC recognized as non-text (e.g. R in “Relaxation” or g in “Booking” in Fig. 5) or due some distances which leads to agglomeration or separation (see “Guests” in Fig. 5) of different text items.
Table 3: Precision, recall and F-measure for clustering each document from the test set.

<table>
<thead>
<tr>
<th>Doc. Id.</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.49</td>
<td>0.61</td>
<td>0.54</td>
</tr>
<tr>
<td>2</td>
<td>0.52</td>
<td>0.42</td>
<td>0.47</td>
</tr>
<tr>
<td>3</td>
<td>0.70</td>
<td>0.68</td>
<td>0.69</td>
</tr>
<tr>
<td>4</td>
<td>0.55</td>
<td>0.62</td>
<td>0.59</td>
</tr>
<tr>
<td>5</td>
<td>0.55</td>
<td>0.62</td>
<td>0.58</td>
</tr>
<tr>
<td>6</td>
<td>0.68</td>
<td>0.51</td>
<td>0.58</td>
</tr>
<tr>
<td>7</td>
<td>0.43</td>
<td>0.36</td>
<td>0.39</td>
</tr>
<tr>
<td>8</td>
<td>0.82</td>
<td>0.78</td>
<td>0.78</td>
</tr>
<tr>
<td>9</td>
<td>0.56</td>
<td>0.38</td>
<td>0.46</td>
</tr>
<tr>
<td>10</td>
<td>0.72</td>
<td>0.83</td>
<td>0.77</td>
</tr>
<tr>
<td>11</td>
<td>0.76</td>
<td>0.59</td>
<td>0.66</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.61</strong></td>
<td><strong>0.58</strong></td>
<td><strong>0.59</strong></td>
</tr>
</tbody>
</table>

Fig. 5: Text patch grouping obtained for an exemplary mind-map image.

7.4. Word recognition

In order to be able to properly evaluate word recognition performance for the whiteboard-reading task given only a quite limited amount of domain specific documents, we decided to consider the text patches extracted from the whole set of whiteboard documents, i.e., the
Camera-Based Whiteboard Reading for Understanding Mind Maps

(a) Erroneous agglomeration with other words

(b) Erroneous agglomeration with graphics

(c) Missing "na" in between

(d) Missing "T"

(e) Successful agglomeration

(f) Successful agglomeration

Fig. 6: Typical outcomes of the unsupervised clustering of elementary document items into text patches.

Table 4: Overview of the evaluation results obtained: Model configuration (training data used, topology of basic models, number of model states, codebook and feature window size) and resulting word error rate in percent (right columns).

<table>
<thead>
<tr>
<th>Model configuration</th>
<th>Training</th>
<th>Topology</th>
<th># States</th>
<th># Densities</th>
<th>Window</th>
<th>Task lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>184 1.8k</td>
</tr>
<tr>
<td>IAM-DB</td>
<td>Bakis</td>
<td>2,406</td>
<td>2k</td>
<td>8 px</td>
<td></td>
<td>65.9 65.9</td>
</tr>
<tr>
<td>IAM-DB</td>
<td>Linear</td>
<td>1,374</td>
<td>2k</td>
<td>8 px</td>
<td></td>
<td>62.9 65.9</td>
</tr>
<tr>
<td>IAM-OnDB</td>
<td>Linear</td>
<td>1,296</td>
<td>2k</td>
<td>8 px</td>
<td></td>
<td>52.7 58.5</td>
</tr>
<tr>
<td>IAM-OnDB</td>
<td>Bakis</td>
<td>2,402</td>
<td>2k</td>
<td>4 px</td>
<td></td>
<td>40.0 48.7</td>
</tr>
<tr>
<td>IAM-OnDB</td>
<td>Bakis</td>
<td>2,402</td>
<td>1k</td>
<td>4 px</td>
<td></td>
<td>49.7 49.7</td>
</tr>
<tr>
<td>IAM-OnDB</td>
<td>Linear</td>
<td>1,296</td>
<td>2k</td>
<td>4 px</td>
<td></td>
<td>34.9 42.9</td>
</tr>
<tr>
<td>IAM-OnDB</td>
<td>Linear</td>
<td>1,296</td>
<td>1k</td>
<td>4 px</td>
<td></td>
<td>35.4 42.0</td>
</tr>
</tbody>
</table>

complete MindMap data set\(^6\).

We built different general handwriting recognizers on the IAM-DB and on rendered images of the on-line data provided by IAM-OnDB. The configuration of these HMM-based recognition models is similar to the ones used in our previous research (cf. [36]). However, as explained in the previous section we systematically varied several meta-parameters of the

\(^6\)As the training of the recognition model is performed on out-of-domain data, also in this configuration the test is guaranteed to be writer independent.
models in order to investigate their effect on the generalization capabilities of the recognition systems obtained.

Prior to applying the statistical writing model all text line or text patch images are subject to the usual pre-processing operations, namely skew and slant correction. Additionally, the apparent size of the writing is normalized such that the estimated distance between local contour minima is 25 pixels on average. Subsequently, a sliding window analysis framework is applied with window widths of 4 and 8 pixels. For each window an 18-dimensional feature representation consisting of 9 geometrical features and their approximated temporal derivatives is computed. For modeling the appearance of the writing semi-continuous HMMs with either linear or a Bakis topology for a set of 80 basic character, punctuation, and whitespace units with codebook sizes of 1,024 or 2,048 densities were estimated using the Baum-Welch algorithm. Training was performed for 10 iterations only as informal experiments showed that models trained in order to maximize performance on the respective validation sets — i.e. for 15 to 20 iterations — showed significantly reduced generalization capabilities. Decoding of the model was performed using Viterbi beam-search.

In order to account to some extent for potential text-extraction artifacts and noise in the text-patch images the recognition model consists of a basic lexicon (which includes a whitespace model) and an additional “garbage” model defined as an arbitrary sequence of elementary character models. We used two different lexica. A small task-specific one consisting of only the words found in the ground truth annotation of the MindMap data set (totaling in 183 words) and an extended one which also contains all putative content words heuristically chosen\(^1\) from the training data of IAM-OnDB (1804 words).

On the complete MindMap data set 758 text patches are hypothesized. Evaluation results are however only reported for those 353 patches which were labeled as being readable in the ground truth annotation, i.e., which contained not solely erroneously detected graphical items or complete partial mind-map images. The recognition hypotheses obtained were filtered such that occurrences of “garbage” hypotheses were discarded.

The recognition results obtained for different configurations of the writing model and the two sizes of the task lexicon used are summarized in Tab. 4. It can clearly be seen that training on high-quality data as provided by the IAM-DB will only produce quite poor results on the MindMap data. The performance of the writing model can be improved significantly\(^1\), when using rendered on-line whiteboard documents for training instead. Further significant improvements are possible with the reduction of the size of the analysis window to only 4 pixels making the feature representation less vulnerable to noise. The best results for the task are obtained with a model based on linear HMMs, which only uses approximately half the number of model states.

\(^{1}\)Putative content words were required to be at least 5 characters long and to occur at least 2 times in the training data of IAM-OnDB.

\(^{1}\)An absolute reduction of the error rate of approx. 5 percent is significant at a level of 95%.
8. Conclusion

We developed a camera-based whiteboard reading system, which particularly addresses the analysis of hand-drawn mind maps. Mind maps’ spatial arrangements of handwritten ideas in graph-like formations are important means for, e.g., structuring the results of brainstorming sessions as they are typically held in creative thinking and problem solving processes. Recognizing mind maps from whiteboard images is relevant since it generates digital representations of such hand-drawn documents, which allows for storage and retrieval, i.e., digital asset management.

The technical contributions of this paper consist of: i) a text detection procedure, which automatically extracts handwriting in whiteboard images using a statistical classifier that has been trained on shape features extracted from connected components, thereby avoiding excessive use of thresholds; ii) a novel approach for unsupervised layout analysis that recovers the graph-like spatial arrangements of ideas captured by mind maps using clustering techniques; and iii) unconstrained handwriting recognition using HMM-based recognizers and in particular focusing on parameter estimation procedures that use out-of-domain sample data for effective system bootstrapping. We evaluated the developed system in an experimental evaluation on unconstrained mind map data. The achieved results are very promising for the envisioned application of camera based mind map reading, for example, to automate corporate document workflows with respect to meeting summarization.

Acknowledgments

This work has been supported by the German Research Foundation (DFG) within project Fi799/3. The authors would also like to thank Leonard Rothacker for his valuable support in the implementation of the whiteboard reading system.

References


