Handwriting Recognition for a Digital Whiteboard Collaboration Platform

Lutz Gericke, Matthias Wenzel, Raja Gumienny, Christian Willems, and Christoph Meinel
Hasso Plattner Institute Potsdam
Prof. Dr. Helmert Str. 2-3, Potsdam, Germany
Email: {firstname.lastname}@hpi.uni-potsdam.de

***Abstract*—The research presented in this paper addresses challenges at the intersection of two disciplines: web based collaboration using digital whiteboards and handwriting recognition. The main focus is on the handwriting recognition in order to enable asynchronous usage of the whiteboard content beyond the existing web portal.**

We present a way to analyze unstructured whiteboard content including drawings, sketches and handwritten text. Our approach uses a recursive extension of the DBSCAN algorithm in order to transfer smaller portions of content to the recognition engine and achieve an appropriate spatial clustering of the content. The adjustment of the configuration parameters, as well as the development of a break condition for the recursion, are shown in detail. We show that it is possible to use an online handwriting recognition engine with offline data and still achieve meaningful results. The presented architecture on the one hand, and the combination of online and offline recognition on the other, ease asynchronous modes of interaction using digital whiteboards.

***Keywords*—remote collaboration; handwriting recognition; digital whiteboard; DBSCAN**

I. Introduction

In the beginning, we developed Tele-Board, which is a digital whiteboard system, supporting different modes of work within the same tool - asynchronous and synchronous as well as co-located and distributed settings (cf. [3]). To make asynchronous working modes more convenient and enable reuse of the produced data, we wanted to find ways of better documenting and computationally "understand" the content. Based on the data that is already archived in our system, we evaluated different approaches in the field of handwriting recognition and their applicability for unstructured whiteboard notes. Applications for the recognized texts are searches on the whiteboard content or reuse in text-applications, e.g. spreadsheets or presentations.

Handwriting recognition has been a major field of interest for researchers in machine learning applications. Recently, some very advanced solutions have emerged on the market. While some solutions use Optical Character Recognition (OCR) to analyze scanned documents, others deal with handwritten text from, e.g., a Tablet PC, usually with the help of an online handwriting recognition using vector data. Typically, these online handwriting recognition systems have a higher recognition rate than offline recognition systems [9], since they can use more features of the writing (e.g., order, pressure or drawing velocity).

Recognition of handwriting is a complex task [10] and therefore, results will not be available to the users instantly. Nevertheless, for collaboration tasks, it is often not a good idea to run an online recognition, because interaction with the user (e.g., choosing recognition alternatives) would capture too much attention. In our application domain - handwriting recognition on a digital whiteboard system - distractions of any kind are not welcome since people are focused on the creative teamwork.

The data we use in the Tele-Board system - which is the underlying application framework for the development described in this article - is vector data. This means that the representation of handwritten texts, drawings, etc., is not only stored as an image but as line strokes consisting of point coordinates. This way, we can benefit from two different fields: an offline recognition that does not distract users and a recognition quality as good as an online handwriting recognition system equipped with more context information. However, we also need an efficient way to analyze the data that has been archived before the introduction of the handwriting recognition within the Tele-Board.

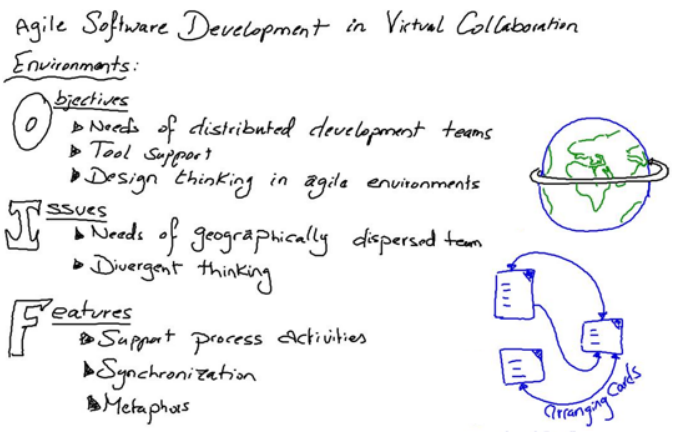

Figure 1. Digitally archived image representation of mixed writing/sketching on a whiteboard

Our approach is an offline handwriting recognition using an existing online handwriting system with very good results even for low-quality writing and different writers. In order to realize this procedure, we have to overcome some problems. The stroke data has to be transferred to the recognition engine

and somehow be replayed there as if it were a live drawing. The problem proves to be even more complex when we go into the details. Whereas in an online handwriting recognition system the input field is often a small area on the screen where single words or phrases can be written, a whole whiteboard as an input field is a much more complex construct (see figure 1). We have to find an automated solution to build clusters of content which can be passed to the recognition engine. Another problem that has to be solved by our presented clustering algorithm is the temporal order, which is not necessarily equivalent to the spatial order of the single strokes.

Towards an understanding of the project's context, we give an architectural overview of the Tele-Board system in the following section. Following, we describe related work in the field of digital whiteboards, offline and online handwriting recognition, as well as clustering algorithms. We then show the clustering algorithm and the details of triggering recognition as well as an evaluation of the recognition results. A conclusion will summarize this article and give an outlook on further research activity.

II. Architectural Overview of the Tele-Board System

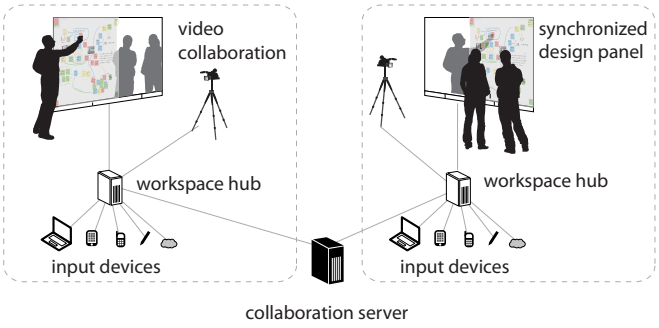

Figure 2. Tele-Board system's overall setup: two locations are connected via a collaboration server, several input devices are used to create content, the whiteboard hardware is used to interact with the whiteboard surface

This section outlines the general architecture of Tele-Board. As shown in figure 2, there are multiple locations (in this case only two) that are connected via a communication server. This server synchronizes whiteboard content between the boards. A touch-sensitive whiteboard input device is used to interact with the content. Several mobile input devices (iPad/iPod/Tablet PC) are used to create content such as handwritten sticky notes or keyboard typed notes from a laptop computer. The communication relies on the open Extensible Messaging and Presence Protocol (XMPP) and whiteboard element data is encoded as XML format. Due to this client server infrastructure, we are able to capture every single synchronization command between the clients and store the data in a database. This allows us to reconstruct every single point in time of the whiteboard content history. For more details on this infrastructure, see [2].

Whiteboards are organized with the help of *projects* and *panels*. A panel describes one whiteboard session with its timeline of events ($p = (e1, e2, e3, ...)$), whereas an *event* can be a NEW, CHANGE or DELETE action on a single whiteboard element (path, sticky note, cluster, etc.). A strict temporal order makes it possible to retrace every element's timeline and its state at a given time. A project is a set of panels ($pro = \{p1, p2, ...\}$) used to organize the different sessions and equip them with a set of permissions [2].

Figure 3. Graphical and XML representation of a sticky note whiteboard element

For the hand writing recognition (HWR) explained here, a path-based representation of each drawn element on the whiteboard is important. Figure 3 shows the representation of a sticky note. It uses SVG-notation to express drawn paths on the sticky note. For sketches directly drawn on the white surface of the board, a similar representation is used. Thus, the input for the handwriting recognition is a list of whiteboard elements describing the state of a panel. A preprocessing step extracts paths from the whiteboard elements while keeping the connection between a single path and the whiteboard element it belongs to.

III. Related Work

A. Digital Whiteboard Systems

Although whiteboard systems - mostly image or video based - have been in the focus of research for over twenty years (e.g., [8], [11], [12], [17], [18]), teams rarely use these systems in corporate or research environments [5]. One reason for this may be that it is still hard to reuse content in office applications, e.g., search within the written content, or otherwise use the full potential of digitalizing the communication data.

The first tools to support creative collaboration of spatially separated teams were *VideoDraw* [19], *VideoWhiteboard* [18] and *Clearboard* [6], each developed in the early nineties. VideoDraw and Clearboard combine synchronous drawing and the ability to observe remote partners at the same time. A desktop-like setup combined with cameras is used to reproduce drawings from one side on the other. A major drawback of almost all these systems is the often missing archiving functionality. One existing system, which also archives the history of a whiteboard, is, e.g., the system by Klemmer et al. [8]. In this example, however, only an image representation of the content is used on an on-request basis.

B. Handwriting Recognition

Research and industry have been working on the problem of handwriting recognition since the 1960s. In the beginning, systems were not powerful enough. In the following years, when research did not focus much on this area [15], Guy Lorette [10] stated in 1999:

[...] the problem of handwriting recognition was initially considered as being very easy to solve, but has later proved to be very difficult.

From the nineties until now, some significant improvements have been achieved, resulting in some systems on the market that can be used in professional environments.

A digitalization of handwriting can be realized by scanning a sheet of paper or by capturing strokes directly during the writing process using a special input device. The online HWR typically uses a continuous stream of coordinates sent from the hardware showing the development of the writing, whereas the offline approach uses a fixed representation of the written content without any temporal information [14], [15]. Our definition of offline data does not necessarily use an image representation of content. From our point of view, the property "offline" represents the fact that the data is somehow archived and the recognition will not be performed during writing.

As research has shown, it is extremely complex to build a full-fledged handwriting recognition system. Therefore, our idea is to use one of the most elaborate solutions, use the content we produce with the Tele-Board system, process the data, and prepare it for further applications. We evaluated some of the most common handwriting recognition tools. A comparison can be found in table I.

TABLE I
COMPARISON OF EVALUATION RESULTS FOR EXISTING HAND WRITING RECOGNITION SYSTEMS

System	Cursive handwriting	Recognition	Writer independence
MyScript Builder	yes	very high	yes
Microsoft Ink	yes	very high	yes
CellWriter	no	very high	no
Lipi Toolkit (Standard Recognizer)	no	unknown	no

Liwicki et al. [9] present an approach for online handwriting recognition on digital whiteboards. The authors have developed a HWR system with a recognition rate that is still too low to be practically useful. It takes huge training sets to optimize such a system, which can often only be done by commercial applications.

The criteria of having a whiteboard solution for a potentially large group of different users who are not willing to complete an hour-long training in handwriting recognition, led to the constraint that the used HWR system must be writer-independent. It also has to be ensured that under difficult settings (whiteboard surface, different hardware that is used with the system) the recognition rate still produces good results.

As a result of our evaluation, we chose the Microsoft Ink API. Using an online handwriting recognition system with offline data leads to the problem of transforming the archived whiteboard content into data that can be passed to the recognition engine. Due to the high structural diversity and complexity of the whiteboard content (see figure 1), it is necessary to separate smaller areas on the whiteboard surface to pass to the recognition engine. Further details are explained in section IV.

C. Clustering Algorithms

This problem led us to the evaluation of clustering algorithms. Clustering algorithms can be distinguished in terms of clustering methodology. Typical categories are: *partitional clustering*, *hierarchical clustering* and *density-based clustering* [4].

Partitional clustering algorithms try to directly decompose the data set into a set of disjoint clusters. K-Means is an example of a partitional algorithm that attempts to find a user-specified number k of clusters that are represented by their center points (centroid) [16]. Problems are the necessity to specify k, the number of desired output clusters in advance [7] and the discovery of clusters with non-convex shapes [4].

Hierarchical algorithms generate a hierarchical decomposition of the data set and do not require the number of clusters to be created. There are two approaches of hierarchical clustering. Agglomerative clustering uses a bottom-up methodology for creating clusters. It starts with all data as individual clusters. The closest pair of clusters is merged at each step until only one cluster or a fixed number of clusters are left. A top-down approach is used by divisive hierarchical clustering. At the beginning, one cluster contains all data. The cluster is then split into sub-clusters. This operation is applied in a recursive manner until each cluster contains a point or there is a fixed number of clusters left [16]. In both cases, it is difficult to determine the point when clusters should stop being merged or split.

Density-based algorithms generate clusters with the help of connectivity and density functions [4]. They are based on the observation that the density of data inside a cluster is considerably higher than outside the cluster [1]. The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm creates disjoint clusters of arbitrary shape. The number of clusters to be created is determined by the algorithm itself based of the given data. Data located in a region of low density is marked as noise and is not assigned to a cluster [1].

IV. IMPLEMENTATION AND INTEGRATION OF THE CLUSTERING ALGORITHMS

A. Integrating the Hand Writing Component as a Web Service

As mentioned earlier, we decided on using Microsoft Ink as a recognition engine. This system is included in the Microsoft Windows operating system (since Windows Vista) [13]. Due to the demand of platform independence, we chose a web service interface to be implemented on a Windows virtual machine running the recognition. The interrelations between the several components of the overall system are shown in figure 4. The Windows server will be accessed via a web service call from the server component, passing all whiteboard data from a specific whiteboard. The recognition plug-in deployed in the XMPP communication server is triggered every time a

client session ends to reanalyze the whole whiteboard. The results, which are returned from the service, are stored in the respective database interlinked with the whiteboard content in order to preserve the relation between original content and recognized results.

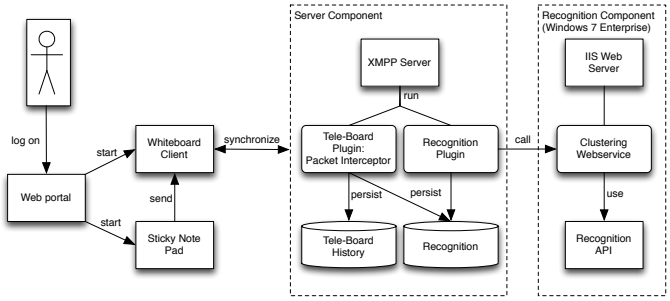

Figure 4. Main recognition components (black) and their interrelation with existing components (gray) of the Tele-Board system architecture

B. Clustering Whiteboard Content

In the case of very large writing areas such as whiteboards, which contain a lot of content such as handwritten text, sketches or drawings, the recognition engine has to overcome limitations. The typical use of the Microsoft Ink system is limited to a small box in which users can write with a tablet PC. A whiteboard surface is much larger and the content much more complex. By processing a complete whiteboard in the recognition system, we encountered various problems and determined that the recognition rate is close to zero.

Besides the volume of data of such a large area, another problem is the effect of temporal discontinuity, meaning that parts belonging together spatially, are not necessarily in a temporal order. In a typical online recognition, this problem does not arise. However, using offline data in an online system demands for the reconstruction of the temporal order.

The content of a whiteboard panel which is archived in the Tele-Board system is typically very diverse and complex. To solve this problem, we want to extract smaller content parts on which to run the recognition, since the smaller the writing area, the more probable it is that handwritten text in this area is written without interruption. These clustered areas also allow for a better relation between recognized text and the area on the whiteboard. A starting point for this clustering is the density of drawn points per whiteboard area. The closer these points are, the higher the probability of a spatial connection between the points and the paths they belong to. Therefore, we need an adequate clustering algorithm.

The requirements for a suitable clustering algorithm can be defined as follows:

- spatial, two-dimensional clustering based on the distance of coordinates
- classification into disjoint clusters
- no fixed definition of a number of resulting clusters
- clusters of an arbitrary shape

1) Naïve Clustering: As a suitable clustering algorithm, we decided to use DBSCAN. Density-based clustering algorithms such as DBSCAN have the advantage that they can identify clusters of an arbitrary shape, but the results heavily depend on the configuration parameters: the minimal number of points to build a cluster *num* and the maximum distance *eps* of two points within a cluster. For instance, choosing a small value of eps typically results in a higher number of clusters.

Using this algorithm with fixed parameters in only one run revealed limitations within the whiteboard domain. As shown in figure 1, very different font sizes may exist, making it difficult to find universal parameters (e.g., larger fonts demand for a larger maximum distance than smaller fonts).

Figure 5 shows what may happen when using the plain clustering algorithm on the existing data (topmost image). The parameters cannot be optimized locally but have to be defined on a global level. This means that the maximum distance parameter can be good for, e.g., a written heading, but is not well suited for standard text. A compatible solution for the whole panel cannot be found in the majority of cases.

We found out that applying this algorithm recursively is an elegant way of dealing with the diversity of the content. It is a solution based on the combination of a density-based DBSCAN algorithm with a divisive hierarchical clustering. This new method is implemented with the web service as shown in section IV-A.

With a recursive approach, the question of an appropriate break condition arises. Our first idea was to start with high eps values on higher levels of the recursion and lower this value until it falls below a certain threshold. The result of this simple break condition can be seen in the second image of figure 5. The recognition results are much better compared to the naïve clustering and the clusters are much more fine-grained and better match the content. For an application such as a search engine that should be able to display the exact position of content found on the panel, this is a major improvement. However, there is still room for enhancement: Some phrases that have been written consecutively are not represented within one cluster but often split over multiple. To address this problem, we came up with a more specialized break condition.

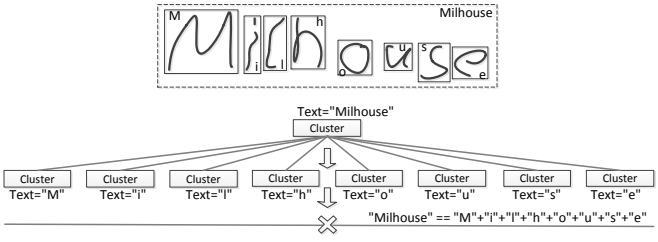

Figure 6. Recursion steps for the clustering algorithm using the advanced break condition

2) Advanced Clustering: As we have seen, results of the recognition will be much more precise if they are recursively clustered and therefore more fine-grained. This recursive ap-

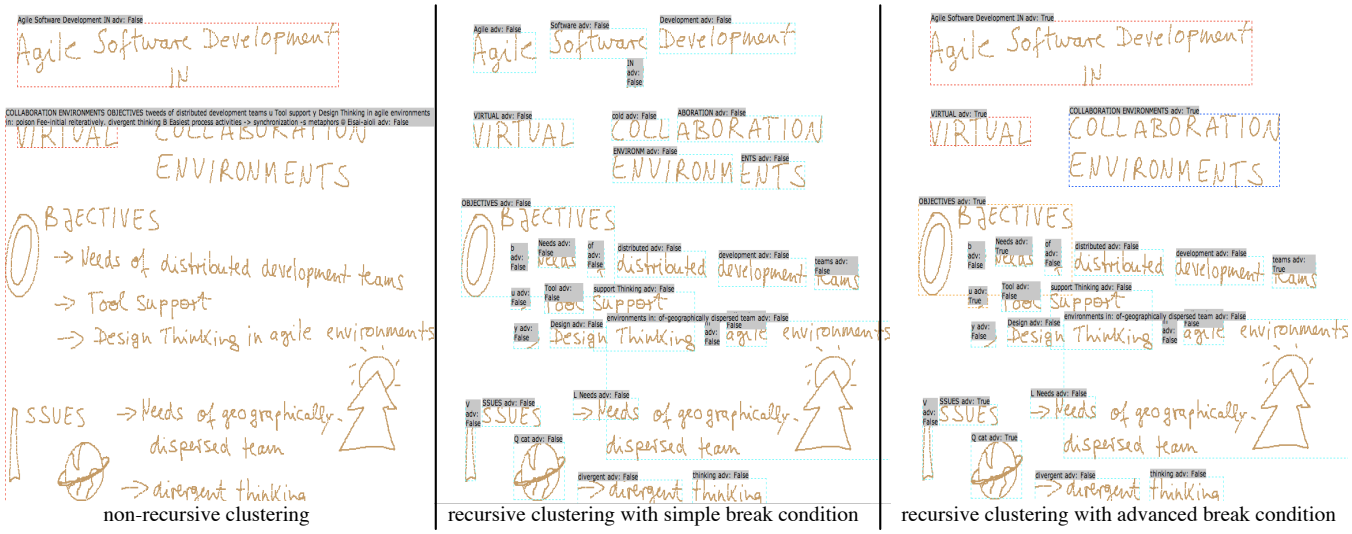

Figure 5. Comparison of three different clustering methods (from top to bottom): simple clustering algorithm without recursion, recursive clustering with a simple parameter-based break condition, and advanced clustering with an adaptive break condition; boxes display the resulting clusters, gray tags in the boxes show the recognition result

proach can better adapt to variations of size and positioning within a whiteboard panel. The break condition has to adapt to the content as well. A fixed threshold parameter cannot be found for every kind of content. The definition of a break condition is difficult because the automatic measurement and evaluation of the quality of a cluster are only partially possible due to missing contextual knowledge. The cluster's quality depends on the quality of the recognition results.

```
function cluster(Cluster, eps) : cluster_list
// fallback break condition
    if thresholdeps >= eps
        return [Cluster]
// run recognition on Cluster
    result = recognize(Cluster)
// lowering maximum distance (eps)
    decrease(eps)
// cluster based on maximum distance threshold
    cluster_list = cluster(Cluster, eps)
    recognition_list = []
// concat recognized results from child clusters
    for i in cluster_list
        recognition_list.add(recognize(i));
    child_recognition = concat(recognition_list)
// decision based on comparison child/parent recognition
    if matches(child_recognition, result)
        return [Cluster]
    else
        return cluster_list
```

Figure 7. Pseudo-code of the advanced break condition

The final approach we came up with is a break condition which uses the recognition results. Figure 6 shows an example. After the recognition of the word *Milhouse*, another recursion step follows that recognizes every single letter of the word ("M", "i", "l", ...). The break condition for this area can then be determined as follows: if the concatenated recognition result of the child recursion level is equal or a subset of the parent recursion level, the clustering can stop for this local branch (cf. figure 7). The concatenation runs from left to right, which works for most languages, but it could also be configured for different languages supported within the Microsoft Ink API.

To ensure that the algorithm does not loop endlessly, a fixed threshold for eps is also applied as a fallback break condition. Figure 5 shows the difference to the previous approaches. The optimization for the recognition of phrases can be seen here. The header text, for example, will be recognized as one cluster. This is an optimal result for a search application because the context of the search term is given instead of just a single word. Another example for improved results can be observed in figure 5 for the phrase *collaboration environments*: in the simple break condition (fig. 5, second picture), the system recognizes the words *cold*, *aboration*, *environm* and *ents*. A search for *collaboration* or *environment* would not lead to a result. In the advanced break condition, the words are recognized as one block and thus can be found by a search engine. The matching between the concatenation of the parts in the deeper recursion level and those on the higher can be fuzzy, but in general we also had good result with simple string comparison. Depending on the respective texts and handwriting, the differences between the clustering algorithms may vary.

A runtime log of the clustering algorithm running on 10 different panels reveals the percentage of how many recursions hit the advanced break condition and the simple break condition respectively. The results show that this advanced break condition not only improves results in certain situations, but it will be hit oftentimes. In an average of 61.2% of the times, the advanced break condition will be used, whereas in 38.8% the fallback break condition (eps threshold) will be used. Summing up, the advanced break condition turns out to be very valuable. Its impact on the recognition rate will be shown in the next section.

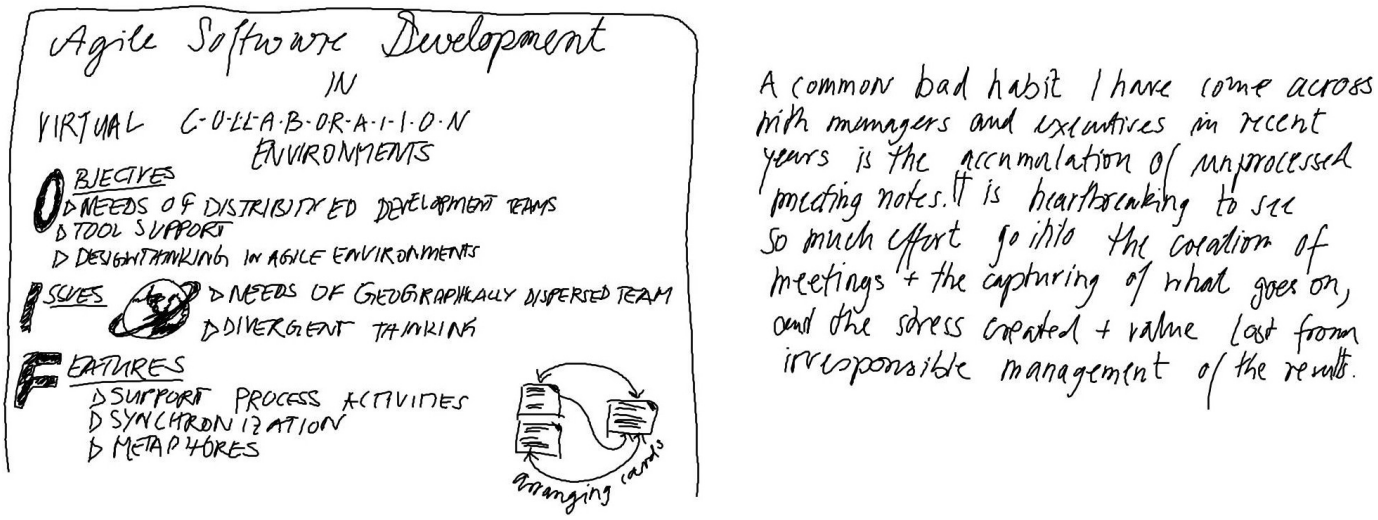

Figure 8. Example panel of one evaluation participant - left (condition 1): mixed notes and drawings; right (condition 2): plain text paragraph

V. Evaluation

To put our implementation in relation to other implementations, but also to reveal the limitations of our approach, we conducted an evaluation of the system's performance. As other approaches in this field do, we measure performance by precision and recall of the recognition. The test scenario consists of two problem domains. Both of them are combined into one whiteboard panel. Condition 1 (left half of Figure 8) mixes handwritten notes together with drawings, which are placed within the text. This makes it difficult for the recognition engine to differentiate between text and drawing and thereby demands a sophisticated clustering algorithm. Condition 2 (right half of Figure 8) is much easier for the recognition system, as content only consists of handwritten text which is typically structured as a well-arranged paragraph.

We asked ten participants to sketch a given whiteboard printout using their typical handwriting and level of detail (especially for the drawings). The textual content of the whiteboard was given as typed text. People were completely free in how to arrange it on the board. They were equipped with a SMARTBoard Interactive Whiteboard[1] as an input device to draw and write.

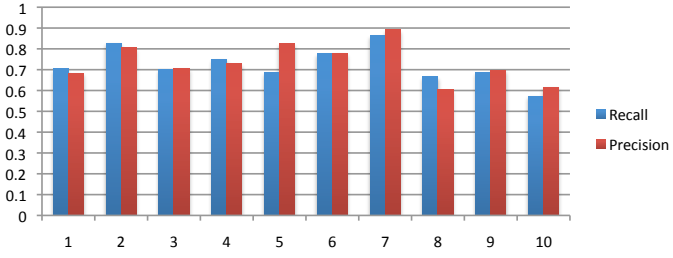

Figure 9. Precision/Recall of recognition results for mixed whiteboard content (condition 1)

[1]SMART Board 600 series interactive whiteboard, http://smarttech.com

Because we had the textual representation of the given content, we could easily compute precision and recall in both cases. For the first condition (see Figure 9), we got an average recall of 73.8% while the precision was 73.3%. This computes to an f-measure of 73.6%. On the other hand, the results of the second condition (see Figure 10) are even better. With an average recall of 91.1%, a precision of 92.8%, and an f-measure of 92.9%, the recognition produces very good results.

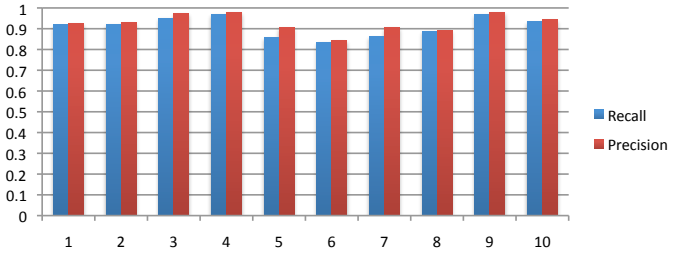

Figure 10. Precision/Recall of recognition results for plain text whiteboard content (condition 2)

Looking at the failures in recognition, it turns out that the second condition could be hardly any better, even with a perfect recognizer: In some cases, people wrote different words (e.g. they mixed up singular and plural). The first condition also showed some particular problems. The headlines had bold printed first letters, which turned out to be unrecognizable in most cases, e.g. in Figure 8 the writer shaded those letters. Secondly, skewed text is also a problem for the recognition engine, e.g. with the phrase "arranging cards". We also did not expect the people to include the dashes in the headline showing "C-O-L-L-A-B-O-R-A-T-I-O-N", which produced differing results. Nevertheless, the process of extracting certain areas of content, which is the main goal of our algorithm, turns out to work properly.

In certain cases, the clustering algorithm returned areas of

whiteboard content with multiple lines of handwritten text one below the other. This sometimes confused the recognition engine so that we came up with another optimization. Within the clusters we generate histograms vertically to find out the text lines within one of the clusters (see Figure 11). Using that information we can split up and translate the stroke data in order to pass it to the recognition engine. For those multiple-lines clusters this optimization gave us significantly better results without influencing the clustering and keeping good results for standard clusters.

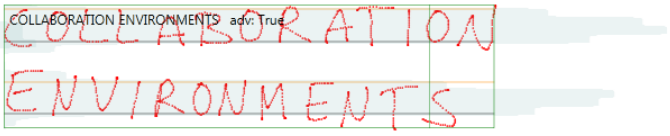

Figure 11. Histogram visualization of multi-line cluster content; green: cluster extents, gray/orange: baseline/topline approximation; gray shading: vertical histogram of stroke points

Comparing our system to the approach presented in [9], we achieve a higher recognition rate. This could be explained by different factors. The recognition in [9] uses image representations of the written text, although a Luidia eBeam device is used to capture the strokes made on the board. This hardware can also be used with Tele-Board. Just like the SMARTBoard, it produces pointer values from drawing events. The term "offline data" tends to be used almost synonymously with image representation. In this paper, we call the usage of line stroke data from an offline source (e.g. a database) also "offline data", but manage to be more efficient in terms of storage and flexibility. Image representations can easily be rendered from the path data, as we also do it in the history browser of our system's management interface (cf. [2]). This architectural decision allows us to achieve better results with less storage capacity used as well as a granularity and accuracy that is difficult to be realized with only the content's image representation.

VI. Summary and Outlook

We presented a novel approach that enables us to benefit from the quality of an online handwriting recognition tool while dealing with offline data. This combination proved to be a very adequate way of dealing with archived whiteboard writing and makes it usable beyond the visual representation (e.g., in search functions).

Figure 12 shows the search results for the term "tool within the Tele-Board web portal. The results include every kind of textual information within the system: a keyboard-typed sticky note, a sticky note with handwritten text and a handwritten text directly on the whiteboard surface. The choice of Microsoft Ink as a handwriting recognition engine turned out to be adequate due to its writer independence and the strong recognition of even poor-quality handwriting (e.g., figure 12 second result).

We tackled a problem many whiteboard systems experience. People use these systems over long distances to communicate synchronously, but it is still cumbersome to document the progress they make. The history solution we presented in [2] is a step towards automated documentation while the handwriting recognition adds even more value to it. People may want to reuse what they created in whiteboard sessions in other documents (e.g., office applications such as PDF files, text documents or tables). So far, this has only been possible using image representations, often only as photographs taken from whiteboards.

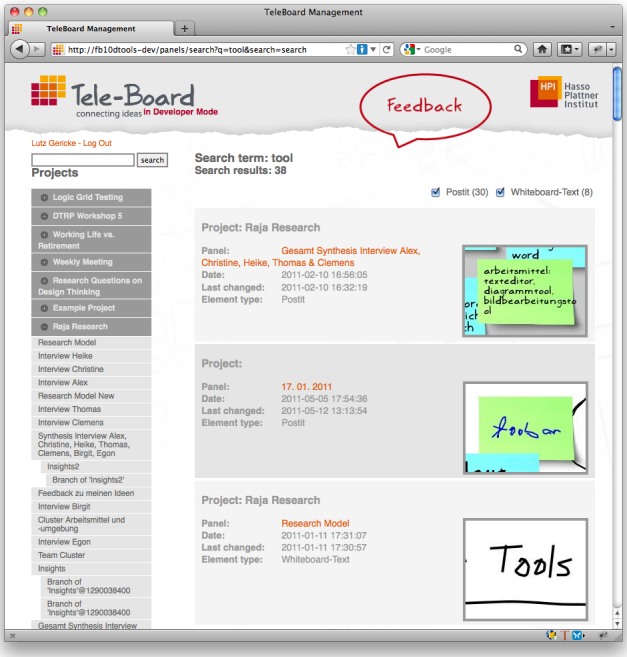

Figure 12. Search result view within web portal

In the future, we have to put more effort into the filtering of the recognition results. In the current state, sketches are treated as regular text and thereby transferred to the recognition engine. The result is often a random character sequence. In a search application, these results do not matter because usually nobody searches for these sequences, but for a final document these parts have to be erased.

What we showed is a promising solution for analyzing recorded whiteboard data and extracting meaning out of the handwritten text, which then can be used in search engine applications. The approach of combining an online handwriting system with archived non-live data benefits from the best of both worlds. It shows the accuracy of online recognition and can be applied with the independence of an offline recognition system without using the client's resources.

Acknowledgment

We would like to thank the HPI-Stanford Design Thinking Research Program for funding and supporting this project.

REFERENCES

[1] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996.

[2] L. Gericke, R. Gumienny, and C. Meinel. Message Capturing as a Paradigm for Asynchronous Digital Whiteboard Interaction. In *6th International ICST Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2010.

[3] R. Gumienny, L. Gericke, M. Quasthoff, C. Willems, and C. Meinel. Tele-Board : Enabling Efficient Collaboration In Digital Design Spaces. In *CSCWD '11*, 2011.

[4] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[5] E. M. Huang, E. D. Mynatt, D. M. Russell, and A. E. Sue. Secrets to Success and Fatal Flaws: The Design of Large-Display Groupware. *IEEE Computer Graphics and Applications*, 26(1), 2006.

[6] H. Ishii and M. Kobayashi. ClearBoard: a seamless medium for shared drawing and conversation with eye contact. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 525–532, 1992.

[7] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31:264–323, September 1999.

[8] S. R. Klemmer, K. Everitt, and J. Landay. Integrating Physical and Digital Interactions on Walls for Fluid Design Collaboration. *Human-Computer Interaction*, 23(2):138–213, Apr. 2008.

[9] M. Liwicki and H. Bunke. Handwriting Recognition of Whiteboard Notes. 2005.

[10] G. Lorette. Handwriting recognition or reading? What is the situation at the dawn of the 3rd millenium? *IJDAR*, 2(1):2–12, 1999.

[11] E. D. Mynatt, T. Igarashi, W. K. Edwards, and A. LaMarca. Flatland: new dimensions in office whiteboards. *Conference on Human Factors in Computing Systems*, 1999.

[12] E. R. Pedersen, K. McCall, T. P. Moran, and F. G. Halasz. Tivoli: an electronic whiteboard for informal workgroup meetings. *Conference on Human Factors in Computing Systems*, 1993.

[13] J. A. Pittman. Handwriting Recognition: Tablet PC Text Input. *Computer*, 40:49–54, 2007.

[14] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:63–84, 2000.

[15] J. R. Prasad and U. Kulkarni. Trends in handwriting recognition. *International Conference on Emerging Trends in Engineering & Technology*, 0:491–495, 2010.

[16] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[17] A. Tang, C. Neustaedter, and S. Greenberg. *VideoArms: Embodiments for Mixed Presence Groupware*, pages 85–102. Springer London, London, 2007.

[18] J. C. Tang and S. Minneman. VideoWhiteboard: video shadows to support remote collaboration. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 315–322. ACM New York, NY, USA, 1991.

[19] J. C. Tang and S. L. Minneman. Videodraw: a video interface for collaborative drawing. *ACM Trans. Inf. Syst.*, 9:170–184, 1991.