Grouping Text Lines in Freeform Handwritten Notes

Ming Ye, Herry Sutanto, Sashi Raghupathy, Chengyang Li and Michael Shilman

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

{mingye,herrys,sashir,chengyli,shilman}@microsoft.com

Abstract

Handwritten text lines are prominent structures in freeform digital ink notes and their reliable detection is the foundation to a natural and intelligent interface for note editing and repurposing. This paper presents an optimization method for text line grouping. The global cost function is designed to find the simplest stroke partitioning to maximize the likelihood of the resulting lines and the consistency of their configuration. A dynamic programming algorithm provides an initial segmentation of the time-ordered stroke sequence. Then a local gradient-descent algorithm iteratively evaluates splitting and merging hypotheses to minimize the global cost function. On average, the proposed technique processes each note page in less than a second at a 90% accuracy.

Figure 1. Examples of targeted scenarios.

1. Introduction

This paper addresses the problem of grouping handwritten text lines in freeform digital ink notes. Text lines are the most salient structures in such notes and their reliable detection is the foundation to higher-level layout analysis and semantic parsing [6, 3, 7, 9]. Freeform ink notes are a mixture of complex structures such as blocks of text, drawings, charts and annotations. Fig. 1 shows two note pages donated by real TabletPC users. Such note-taking scenarios are the focus of our study.

Some early ink parsing techniques were developed in the pen-based user interface area. Moran et al [4] use a gesture-driven method to manipulate horizontally aligned text, lists and tables as part of the Tivoli whiteboard project. Chiu and Wilcox [2] propose a dynamic grouping technique for ink and audio notes in which they hierarchically cluster ink strokes based on spatiotemporal distances. UI studies focus more on user interface design and their ink parsing algorithms are usually simple in comparison. Another related area is document image analysis (DIA) which deciphers the structure in scanned images of printed documents [5]. It is important to point out that existing DIA techniques do not readily apply to ink parsing. The strong assumptions DIA makes about document regularity do not hold true for digital ink which is far more variable and heterogeneous [7, 1].

Papers dedicated to ink parsing have just begun to appear in the past several years [1, 3, 6, 7, 9]. Jain et. al. [3] describe a system which classifies strokes into text and non-text, groups text strokes into lines, and partitions the page into text, diagram and table regions. Their line grouping module assumes horizontal baselines and takes a projection-based DIA alike approach [6]. Blanchard and Artires [1] propose a probabilistic feature grammar (PFG) framework and apply it to segmenting an ink page into words, lines and paragraphs. They use a beam search strategy and a genetic algorithm to reduce the prohibitive complexity that is inherent to the PFG framework. Their current system only handles single-column writing pages. Although it remains theoretically possible to add more grammars for more general scenarios, how to handle the additional computational complexity incurred and how to define grammars for complex structures in real ink data such as drawing and annotations will be challenging. The system developed by Shilman et al. [7] is perhaps the first system that is designed for freeform ink and significantly departs from DIA approaches. It groups lines by merging pairs of stroke clus-

ters in four passes in an increasingly aggressive manner, and adds a postprocessing step to correct bullet grouping errors. Although it exploits a lot of good heuristics, the lack of a principled framework limits the system's robustness and extensibility.

Inspired by global optimization approaches to piecewise-smooth visual reconstruction in low-level vision [10, 8], this paper formulates line grouping as an optimal stroke partitioning problem. A global cost function is designed to find the simplest stroke partitioning to maximize the goodness of the resulting lines and the consistency of their configuration. The goodness of a line is measured by its linear regression error and the horizontal and vertical compactness of its strokes. A graph is constructed to connect each pair of neighboring lines and the consistency of the line configuration is measured by the angle difference between neighbors. The complexity of a partitioning is measured by the number of lines. We developed a gradient-descent local optimization method to solve the resulting cost minimization problem. Leveraging the fact that text lines are normally written in approximate time order, we obtain a good-quality initial solution by partitioning the 1D array of strokes using a simplified version of the cost function by dynamic programming. The results from the temporal grouping stage may contain both under- and over- grouping errors. We generate a queue of merging and splitting hypotheses, accept the hypothesis incurring the most global cost decrease, update the line configuration and affected hypotheses accordingly, and iterate until the global cost no longer changes.

The rest of the paper is organized as follows. Section 2 and 3 describe the formulation and optimization technique respectively. Experimental results are shown in Section 4. Section 5 summarizes the paper and points out some future work directions.

2. Formulation

We formulate line grouping as an optimal stroke partitioning problem – given a page of strokes, finding the partitioning of the stroke set to optimize the likelihood of the resulting lines and the consistency and simplicity of the their configuration. There are three likelihood terms and two prior terms in our cost function.

Likelihood of a Line. We compute three features to measure the likelihood of a line. The linear regression error e_{LR} measures the deviations of the stroke points from the fitting line and reflects the linearity of the stroke set. We denote the fitted line segment by $\hat{l}$ (dotted line in Fig. 2). $d_{\mathrm{xmax}}(l)$ and $d_{\mathrm{ymax}}(l)$ are the maximum inter-stroke distances projected onto the fitting line and its orthogonal direction respectively. They reflect the horizontal and vertical compactness of the stroke set.
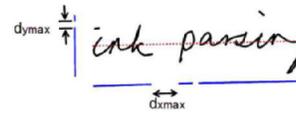
Figure 2. Likelihood of a line. The dotted line is the linear fitting result. d_{xmax} **and** d_{ymax} **are the maximum inter-stroke distances projected onto the fitting line and its orthogonal direction respectively.**
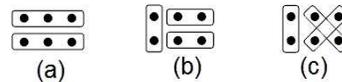
Figure 3. Configuration consistency. Dots represent strokes and boxes represent line grouping. The configuration consistency decreases from (a) to (c).

Configuration Consistency. Consider the example given in Fig. 3 where the dots represent strokes and the boxes represent the line partitioning. The possibility of the results decreases from (a) to (c) as the line configuration consistency decreases.

To measure the consistency of a line configuration, we create a neighborhood graph where the vertices corresponds to the lines and the edges correspond to neighbor relationships between lines. A pair of lines are neighbors if the minimum distance between their fitting line segments falls below a threshold *and* there are no other objects (lines or drawings) lying between them. Fig. 4 gives an example neighborhood graph of six lines. Each line is represented by a gray bar and each edge represented by a solid black line. The edge (dotted line) between Line 1 and 2 is rejected because there is a drawing stroke (curvy line) between them.

We compute each line's configuration consistency $\theta(l)$ as the neighbor-length-weighted sum of the orientation angle difference. We use the weights to encourage lines to be more consistent with longer (usually more reliable) neighbors. More robust functions can be adopted to regulate the
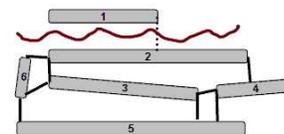

Figure 4. Neighborhood graph. Lines are represented by gray bars and edges by solid black lines. The edge (dotted line) between Line 1 and 2 is rejected due to the drawing stroke (curved line) between them.

angle difference to reduce oversmoothing where neighboring lines do have very different orientation (e.g. between main text and annotation text) [8] .

Model Complexity. We measure the complexity of a partitioning π by its number of lines. This is a simple instantiation of the MDL principle [10].

The Cost Function. The cost of a partitioning π is a weighted sum of the five terms that we have introduced above, i.e.,

$$e_\pi = \sum_{l\in\pi} e_{\text{LR}}(l) + w_1 d_{\text{xmax}}^2(l) + w_2 d_{\text{ymax}}^2(l) + w_3\theta(l) + w_4\cdot 1, \quad (1)$$

where w_i are parameters of the formulation controlling the relative importance of the terms. The optimal line grouping is the stroke partitioning that minimizes this cost.

Remarks. Our cost function design was inspired by the global optimization approaches to piecewise-smooth visual reconstruction problems in computer vision [10, 8]. Piecewise-smooth models make the weakest assumptions about the visual field and have proven the most widely applicable and computationally efficient. These properties are a good fit for our problem. The configuration consistency term is our instantiation of the piecewise-smooth constraint.

Global optimization formulations for piecewise-smooth visual reconstruction may originate from different criteria such as energy functions, Bayesian (MAP, MRF), Minimum Description Length (MDL), but many of them arrive at a similar cost function form – a weighted sum of terms for each segment encoding how well the estimate explains the observed data (likelihood) and how well the estimate matches with some *a prior* knowledge about the solutions (priors) [10]. We adopted the terminology in defining the terms in our cost function.

The way we construct a neighborhood graph and measure the configuration consistency by neighbors' differences bears some resemblance to Markov Random Field (MRF) models [8, 10]. This formulation permits us to evaluate global cost changes locally – any configuration changes to a subset of lines only affects the likelihood terms of these lines and the configuration terms of them and their direct neighbors. This property enables the following local-optimization solution technique.

3. Optimization

We developed an efficient gradient-descent local optimization method to solve the resulting cost minimization problem. Starting from an initial solution from temporal grouping, we iteratively generate local alternative hypotheses and accept the one leading to the largest global cost decrease. Details of these steps are discussed in the follows.

Initial Solution from Temporal Grouping. Based on the fact that most text lines are composed of temporally adjacent strokes, we obtain an initial grouping of lines by par-
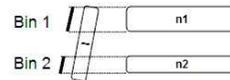

Figure 5. Split a high-configuration-energy line by its neighbors.

titioning the 1D array of temporally ordered strokes. Our cost function uses a subset of terms in the global cost function Eq. 1:

$$e_\pi = \sum_{l\in\pi} e_{\text{LR}}(l) + w_1 d_{\text{xmax}}^2(l) + w_2 \cdot 1, \quad (2)$$

where $d_{\text{xmax}}(l)$ is the maximum inter-stroke distance projected on to the fitting line $\hat{l}$. Such a simplification is enabled by the dramatic reduction of the search space, and it also gives the cost function a recursive structure so that dynamic programming can apply. As a practical speedup, we cut the stroke sequence into spans based on stroke proximity and size similarity, and perform partitioning only within each span.

This step was inspired by the DP line grouping step in [7] but is much more effective – often the times this step alone already produces the correct grouping whereas the DP step in [7] produces smaller stroke clusters. The major reason lies in the model complexity term. [7]'s cost function only contains likelihood terms which opts for over-segmentation.

Generating Alternative Hypotheses. Currently we generate two types of alternative hypotheses: merging a pair of line segments and correcting high-configuration-energy errors, corresponding to the two major error categories of the temporal grouping results.

Under-grouping errors are typical in temporal grouping results due to for example late 'i' dots and 't' crosses. We generating a merging hypothesis for each neighboring pair. High-configuration-energy errors are caused by temporally adjacent strokes belonging to different lines. The most frequent cases are bullets written before or after the list content is filled. As illustrated in Fig. 3.(b), such errors causes high configuration energy in its neighborhood and hence the name. Any line segment l whose maximum angle difference from its neighbors $\{n : n \in N_l\}$ exceeds a threshold is an initial candidate. We find its neighbors which are approximately parallel but have substantial angle differences with l, use these neighbors as bins to split l's strokes (Fig. 5).

Computing Global Cost Changes. Each hypothesis changes an existing configuration π_0 to an alternative π_1 by re-grouping strokes in a local neighborhood. As illustrated in Fig. 6, π_0 and π_1 differ only by the lines colored in dark gray, which we denote as $\{l^0\}$ and $\{l^1\}$ respectively. $\{l^0\}$ and $\{l^1\}$ have the same set of neighbors, which we denote as $N_{\{l\}}$ and show in light gray in Fig. 6. The grouping of $N_{\{l\}}$ remain intact but their neighbor relationships have

changed. Clearly, $\{l\}$ and $N_{\{l\}}$ are the only lines whose configuration are affected by the hypothesis. Therefore, the global cost change from π_0 to π_1 can be evaluated locally.
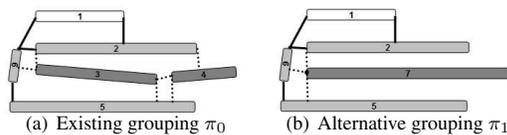

(a) Existing grouping π_0 (b) Alternative grouping π_1

Figure 6. A pair of alternative hypotheses.

Iterations. Given the solution π_0^i at iteration i, we generate a queue of alternative hypotheses $\{\pi_j^i, j = 1, 2, \ldots\}$, accept the hypothesis incurring the most global cost reduction π_*^i, update the line configuration and the hypothesis queue accordingly, and iterate until the queue is empty. The algorithm converged rapidly in our past experiments. Fig. 7 shows our algorithm in action on the example file used in [7] Fig. 5. Our temporal grouping step (iter#0) produces more global results to start with. Then the gradient-descent algorithm corrects one under- or over-grouping error in each iteration. The order in which the errors are corrected is close to the perceived severity ranking, a validation of our cost function design and the gradient-descent strategy. Note that a mistake made in iter#2 was corrected in iter#3. In contrast, [7] handles under- and over-grouping errors separately and each of its four sequential iterative steps uses a different set of rules.
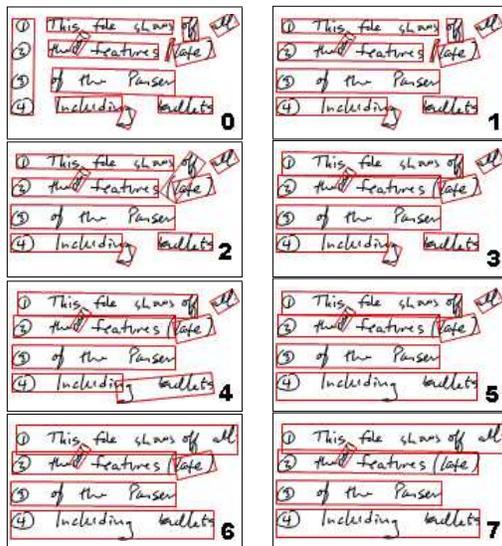
Figure 7. Gradient-descent in action. Iteration numbers are marked in bottom-right corners.

Batch vs. Incremental Mode. So far we have described the "batch" mode of our algorithm. The local gradient-
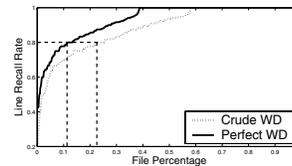

Figure 8. Sorted recall rates for 600 test files.

descent optimization algorithm also provides a good solution to *incremental parsing* – a critical issue for realtime ink analysis. Online ink data are usually generated incrementally – each time the parser is called, the input data may be a page of parsed results plus several new strokes. It is desirable to have some incremental mechanism to "massage" the new strokes into the existing page structure. Our relaxation-like optimization algorithm performs this task naturally. According to our preliminary evaluation, the incremental mode is much more efficient than the batch mode and usually produces consistent results.

4. Experiments

Our experimental data are 600 Windows Journal pages *donated by tens of real TabletPC users*. The collection was chosen to represent key note-taking scenarios. It contains substantial handwritten script in either single-column or multi-column format, in various angles and font sizes, and are often mixed with drawing, charts, tables, annotation to form interesting spatial layout structures. The median and maximum numbers of lines in a page are 15 and 66 respectively. All data have been manually labelled.

The weights in the global cost function Eq. 1 and the temporal cost function Eq. 2 were casually tuned on ten files or so to produce reasonable results. We are currently investigating systematic methods for parameter estimation. Therefore accuracy reported in this paper can be considered as a lower bound of what the system can achieve. The algorithm is implemented in C#. All results reported in this paper were produced in the batch parsing mode on a 3GHz 1GB-RAM Intel PC with tens of other applications running. Our system runs very fast in such settings, spending less than a second elapse time on the pages in Fig. 1.

Since the system is designed to work in mixed writing-drawing scenarios, we evaluate it in two settings: one with perfect (labeled) writing/drawing input (PerfectWD) and the other with a crude writing/drawing classification preprocessor (CrudeWD) that we developed in house. We measure the accuracy of our system by the *recall* metric, which is defined as the number of correct lines divided by the number of labelled lines in each page. Fig. 8 shows the sorted recall rates for the 600 files. In both settings, the recall rate curve quickly rise to 1.0, meaning that the system works

well for the majority of the notes and the errors concentrate on a small set of cases. Even with a crude W/D module in place, about 78% files have less than 20% line grouping errors. The average recall rates for PerfectWD and CrudeWD are **0.93** and **0.87** respectively. The small difference between the two settings shows that the system's performance is fairly stable with the presence of non-text content.

Finally, we show one example in Fig. 9 for qualitative evaluation (cannot show more due to the page limit). The light gray strokes are drawing detected by CrudeWD. The temporal grouping result is almost correct. The gradient-descent step corrects several under-grouping errors, but also introduces over-grouping errors for the call-outs in the middle. Such errors are inevitable for low-level approaches. Nonetheless the results from the low-level module create a good foundation for higher-level semantic modules to bootstrap.
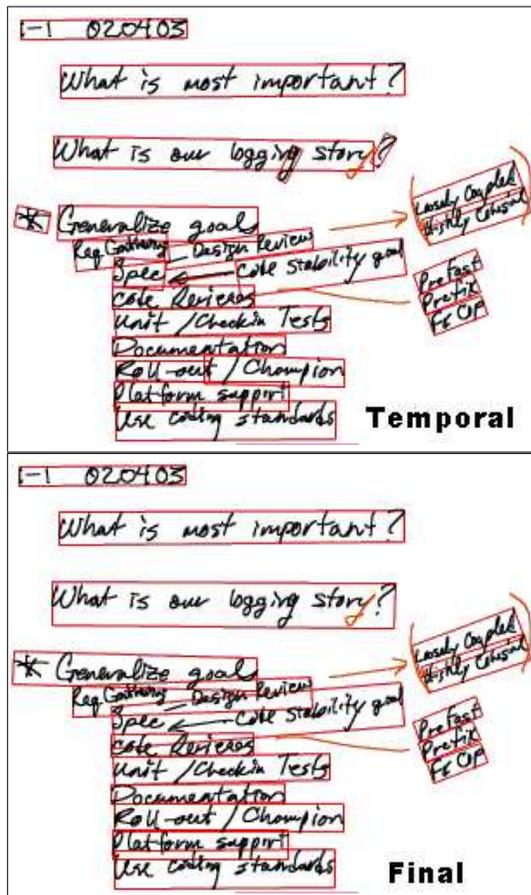

Figure 9. Example for qualitative evaluation.

5. Conclusions

We have presented a global optimization method for grouping handwritten text lines in freeform digital ink notes with a practical solution technique.

The global optimization method has clear advantages over existing heuristic methods. It expresses both prior knowledge and data evidence about the grouping in a unified mathematical formulation and therefore provides an objective criterion for comparing competing hypotheses. Our cost function design drew inspiration from the global optimization approach to piecewise-smooth visual reconstruction in computer vision. This approach has been highly successful in vision and proven in many application domains.

Global optimization frameworks have intrinsic computational complexity which, if not dealt with effectively, would nullify the frameworks' practical value. Another contribution of ours therefore lies in the efficient solution algorithm using temporal properties of online digital ink and gradient-descent local optimization based on a graphical model. The algorithm also provides a natural solution to incremental parsing, an important requirement for realtime ink analysis.

We have tested out our system on a large collection of real user notes containing rich non-text, spatial structures. The average recall rate already reaches 0.87 with a crude writing/drawing classification preprocessor. The processing is fast, taking less than one second on a typical note page even without incremental parsing.

We are currently investigating parameter training for the cost function. After that the proposed method not only can be expanded to include more likelihood and prior terms for higher line grouping accuracy, but also can readily apply to many other ink parsing problems.

References

[1] J. Blanchard and T. Artires, "On-Line Handwritten Documents Segmentation", *Proc. Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 148-153, 2004.

[2] P. Chiu and L. Wilcox, "A dynamic grouping technique for ink and audio notes", *Proc. ACM Symp. on User Interface Software and Technology (UIST)*, pp. 195-202, 1998.

[3] A.K. Jain, A. M. Namboodiri and J. Subrahmonia, "Structure in on-line documents", *ICDAR*, pp. 844-848, Sept. 2001.

[4] T. P. Moran, P. Chiu, W. van Melle, "Pen-Based Interaction Techniques for Organizing Material on an Electronic Whiteboard", *UIST*, pp. 45-54, 1997.

[5] G. Nagy, "Twenty years of document image analysis in PAMI", *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, Vol. 2, No. 1, pp. 38-62, 2000.

[6] E.H. Ratzlaff, "Inter-line distance estimation and text line extraction for unconstrained online handwriting", *IWFHR*, pp. 33-42, 2000.

[7] M. Shilman, Z. Wei, S. Raghupathy, P. Simard and D. Jones, "Discerning Structure from Freeform Handwritten Notes", *ICDAR*, pp. 60-65, 2003.

[8] M. Ye, R. Haralick and L. Shapiro, "Estimating piecewise-smooth optical flow with global matching and graduated optimization", *PAMI*, Vol. 25, No. 12, pp. 1625-1630, 2003.

[9] M. Ye and P. Viola, "Learning to parse hierarchical lists and outlines using conditional random fields", *IWFHR*, pp. 154-159, 2004.

[10] S. C. Zhu and A. L. Yuille, "Region competition: unifying snake/balloon, region growing and Bayes/MDL/energy for multi-band image segmentation", *PAMI*, vol.18, no.9, 1996.