ScribeVideo: Static, Linear Re-Presentation of Blackboard-Style Lecture Videos

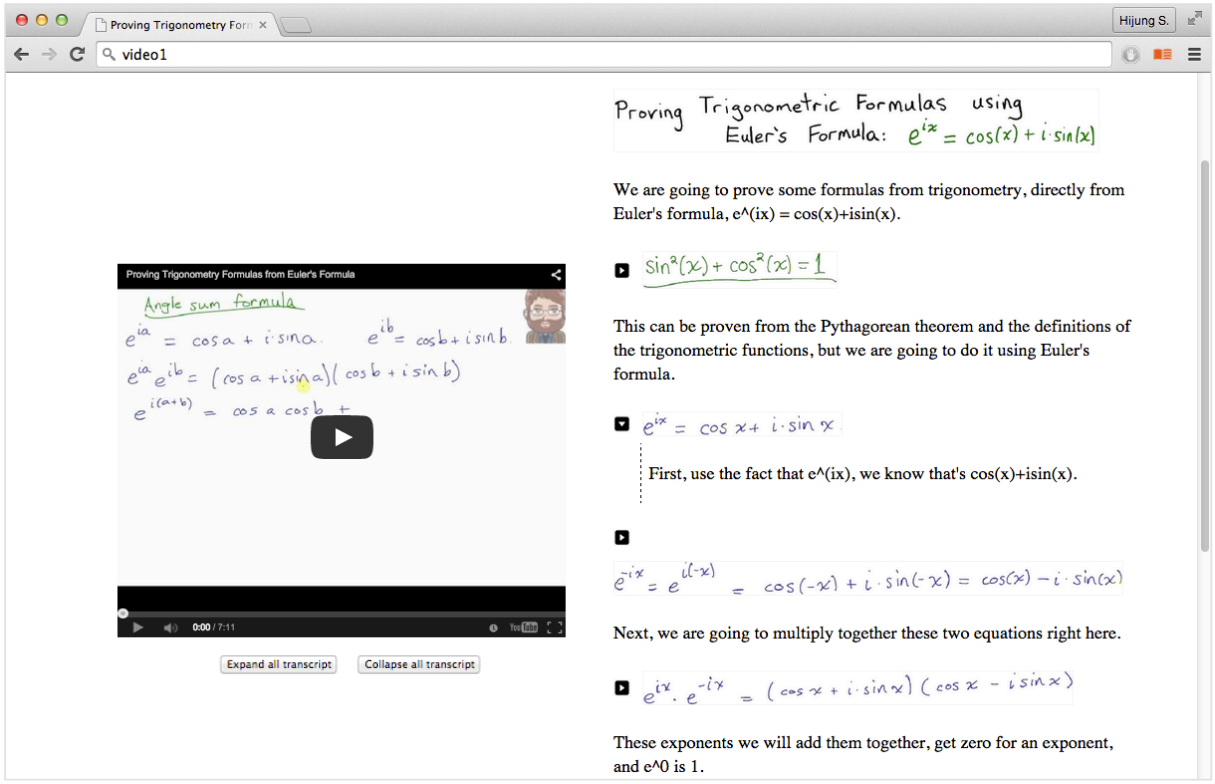

Figure 1: *ScribeVideo interface*

Abstract

Keywords: Video summarization, Video navigation, Lecture videos, Blackboard-style lectures

1 Introduction

With the introduction of online education platforms such as Khan Academy, Coursera, Udacity, edX, and online courses offered by universities, video watching has become an integral part of the students' learning experience. For example, in 2014, the number of universities offering MOOCs doubled to 400 universities offering over 2,400 courses to 16-18 million enrolled students [**?**]. Moreover, *flipped* classrooms, where students watch pre-recorded lectures outside of class, and do more active learning activities within class–are becoming increasingly popular in schools.

Despite the increasingly important and broad role videos are taking in education, video is not an inherently easy medium to navigate dense information. It is difficult for viewers to browse and skim through the underlying content. Viewers have to scrub back-and-forth on a linear timeline slider in order to gain an overview of the topics, or locate specific information of interest. Many platforms, such as Khan Academy and YouTube provide synchronized transcripts, which allow users to click on a word in the transcript and play the video at that location. While transcripts do expose important content from the video, without structured organization and correspondence with visual content, long blocks of text are time-consuming to read and difficult to skim.

Educational videos are different from traditional video contents such as movies. They are usually shorter, more static (less change in scenes), and denser in information, especially textual information. Learning also involves a variety of different interactions with the video. These interactions include: (1) skimming to get a quick overview of the video, (2) searching to find a specific information in the lecture, and (3) re-watching to review specific portions of the video. However, there has been few attempts to adapt video interfaces to support these new types of needs. Exploring the design space of interfaces fitted to the contents and navigation patterns specific to education can facilitate and improve learning experience.

This paper presents ScribeVideo, a static, linear re-presentation of videos to facilitate navigation of blackboard-style lectures.

2 Related Work

Video Summarization Many work on summarizing videos in order to facilitate navigation. [Barnes et al. 2010] Video Tapestry, [Jackson et al. 2013] Panopticon, [Uchihashi et al. 1999], [Hwang et al. 2006][Boreczky et al. 2000] Comic book style layout of keyframes. Content specific:[Ekin et al. 2003] soccer video summarization, [] news footage summarization.

These algorithms have in common key-frame extraction, and layout. They use features such as color histograms to detect keyframes. Educational video content have different characteristics. Most relevant to our work is [Choudary and Liu 2007], which summarizes blackboard-style lectures by creating a panoramic frame of the board.

Video Navigation Interface [Kim et al. 2014a] uses interaction data collected from MOOC platforms to introduce a set of techniques that augment existing video interface widgets. [Pavel et al. 2014] provides a tool to create video digests, structured summaries of informational videos organized into chapters and sections. Most closely related to our work is the NoteVideo interface by [Monserrat et al. 2013], which presents a summarized image of the video composed of clickable visual links to specific places in the lecture.

Improving Transcripts / Subtitles [Hu et al. 2015] optimizing subtitle placement by speaker recognition. [Kurlander et al. 1996], [Chun et al. 2006] subtitle placement for single frame in comic style cinema.

3 Method

3.1 Time-stamped Transcript

Several on-line video lecture platforms (e.g. Khan Academy) provide transcripts. In cases where transcripts were not provided, we used an on-line audio transcription service to acquire a verbatim text transcript. Then, we used a tool from [Rubin et al. 2013] to compute an alignment between the video's audio file and the transcript. The final output is a time-stamped transcript, where each word is annotated with a start and end time.

3.2 Stroke Extraction

Visual content in blackboards-style lectures consists of *strokes* drawn by the instructor on the board. Specifically, a *stroke* is defined as the set of foreground pixels that is drawn during one continuous drawing activity. The method used to extract strokes from video frames is similar to that used by [Monserrat et al. 2013] to extract visual objects in their NoteVideo interface. By comparing the number of foreground pixels in consecutive frames, the start and end time of each drawing activity is detected. A noticeable increase in the foreground pixels signals start of drawing, while no change in the frames marks end of drawing. The difference between the end and start frames gives an image of the stroke drawn during that period. Figure 2 shows examples of extracted strokes from different videos. A typical stroke comprises several characters to several words, or it can also be a part of a figure such as a graph (Figure 2c).

3.3 Hierarchical Grouping of Strokes

We group strokes into hierarchical units: lines and sentence-strokes. A line consists of a set of strokes that *belong together* semantically. For example, a line could be a single row of equations, or a graph including its labels. Figure 3 shows examples of lines. The problem of grouping strokes into lines is analogous to the problem of line breaking, also known as word wrapping [Knuth and Plass 1981].

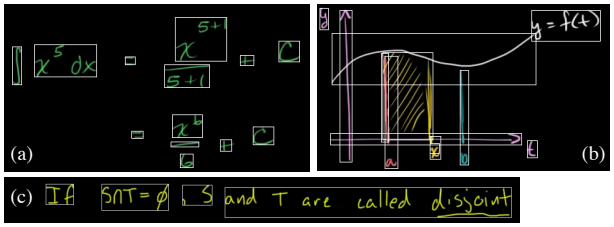

Figure 2: *Examples of strokes extracted from different videos.*

An important difference is that in the traditional word wrapping problem, only a contiguous set of words can be put in the same line. In our case, strokes in a single line can be interspersed by strokes in a different line. This happens, for example, when the instructor goes back and forth between two lines of equations, or between a graph and an equation (Figures **??**).

3.3.1 Lines

We use dynamic programming to group strokes into a set of lines. The algorithm iterates through the strokes, $s_i \in \mathbf{S}$, in the order of their appearance in the video, and keeps track of L_i, the optimal (determined by the scoring function described below) set of lines upto stroke s_i. For each new stroke, s_{i+1}, the algorithm considers all possible previous line breaks L_j, where $j < i+1$. For each L_j, it considers $|L_j| + 1$ possibilities: merging strokes $\{s_{j+1}, ...s_{i+1}\}$ with one of the lines $l_n \in L_j$, or adding a new line $\{s_{j+1}, ...s_{i+1}\}$ to L_j. The optimal solution among the $\sum_{j<i+1}(|L_j|+1)$ possibilities is L_{i+1}.

The scoring function to determine the optimal set of lines is based on a number of observations:

Lines are compact. Strokes that belong together are arranged in a compact figure. We consider three measures of compactness for a line: overall, horizontal and vertical.

- Overall compactness of a line $l$is measured as

$$c_1(l) = \frac{\sum_{s \in l} area(\{s\})}{area(l)} \tag{1}$$

  where $area(\cdot)$ is the the bounding box area of a set of strokes.

- Intuitively, horizontal compactness is related to the maximum horizontal *gap* between strokes within a line. Figure **??** shows an illustration of the of how gaps between strokes are measured. First, we define a horizontal projection function of a set of strokes, S, as

$$proj_h(x, S) = \left|\{s \in S | x_{\min}(s) \leq x \leq x_{\max}(s)\}\right| \tag{2}$$

  where $x_min(s)$, and $x_max(s)$ are the minimum and maximum x-coordinates of the bounding box of s respectively. Then, the maximum horizontal gap of a line l is

$$x_{\text{gap}}(l) = \underset{x_i, x_{i+1} \in proj_h(x,l) \neq 0}{\operatorname{argmax}} (x_{i+1} - x_i) \tag{3}$$

  and horizontal compactness is defined as:

$$c_2(l) = -x_{\text{gap}}(l) \tag{4}$$

- Vertical compactness is defined similarly, in terms of a vertical projection function, $proj_v$, and the maximum vertical gap,

y_{gap}.

$$proj_v(y, S) = \left|\{s \in S | y_{\text{min}}(s) \leq y \leq y_{\text{max}}(s)\}\right| \quad (5)$$

$$y_{\text{gap}}(l) = \underset{y_i, y_{\text{i+1}} \in proj_v(y,l) \neq 0}{\text{argmax}} (y_{\text{i+1}} - y_i) \quad (6)$$

$$c_3(l) = -y_{\text{gap}}(l) \quad (7)$$

Strokes within a line are aligned horizontally. With the exception of certain elements in graphs or figures, most lines comprising equations or words are written left to right. The strokes in such lines are aligned horizontally. We compute the maximum number of horizontally aligned strokes in each line, using the vertical projection function in Equation (5).

$$n_{\text{align}}(l) = \underset{y_{\text{min}}(l) \leq y \leq y_{\text{max}}(l)}{\text{argmax}} proj_v(y) \quad (8)$$

Lines are spatio-temporally separate from each other. This observation is complementary to the first observation, i.e. lines are compact. Whereas strokes that belong together are written close together. We express this property by penalizing overlap between distinct lines, measured by the overlapping area between their bounding boxes.

$$p_{\text{overlap}}(l_i, l_j) = \frac{area(l_i \cap l_j)}{min(area(l_i), area(l_j))} \quad (9)$$

A similar property holds in the temporal domain. After writing a single line and before going to the next line, there is a brief pause while the instructor moves the cursor to the next position or adds some explanation. In order to enforce this property, we compute the temporal distance between two consecutive strokes across line-breaking points.

$$t_{\text{dist}}(s_i, s_{\text{i+1}}) = start(s_{\text{i+1}}) - end(s_i) \quad (10)$$

where $s_i \in l_m$ and $s_{\text{i+1}} \in l_n$ belong to different lines ($m \neq n$), and $start(\cdot)$ and $end(\cdot)$ are start and end times of when the stroke is drawn in the video.

The first four features (compactness and horizontal alignment) are properties of individual lines. Together, these define a scoring function for the *goodness* of a single line.

$$s_1(l) = \alpha_1 c_1(l) + \alpha_2 c_2(l) + \alpha_3 c_3(l) + \alpha_4 n_{\text{align}}(l) \quad (11)$$

The last two features are properties of relationship between multiple lines, and defines a *goodness* score for a set of lines.

$$s_2(L) = -\sum_{l_i, l_j \in L, i \neq j} p_{\text{overlap}}(l_i, l_j) + \sum_{s_i \in l_m, s_{\text{i+1}} \in l_{\text{n}}, m \neq n} t_{\text{dist}}(s_i, s_{i+1}) \quad (12)$$

The final scoring function is a weighted average of the two scores. Figure 3 shows results obtained from our line breaking algorithm. The algorithm successfully segments meaningful groups of strokes from different layouts of figures and equations.

3.3.2 Sentence strokes

If *lines* represent meaningful segments in the video frames, sentences represent semantic units in transcripts. We further group strokes within each line into *sentence strokes* using the temporal correspondence between strokes and transcript sentences. In summary, we have the following hierarchical grouping of strokes: strokes, sentence-strokes, and lines.

Input : list of strokes S
Output: list of optimal lines, $L_{|S|}$
$L_{-1} = -^{''}$ // L_i = optimal set of lines upto i-th stroke
for *each stroke* $s_i \in \mathbf{S}$ **do**
  minscore = $+\infty$
  for $j \leftarrow -1$ ***to*** $i - 1$ **do**
    for $n \leftarrow 0$ ***to*** $|L_j| + 1$ **do**
      // score to merge $\{s_{j+1}, \ldots, s_i\}$ to n-th line of L_j
      // If $n = |L_j| + 1$, $\{s_{j+1}, \ldots, s_i\}$ is a new line
      score $\leftarrow$ line_score(L_j, n)
      if *score* $<$*minscore* **then**
        $optj = j$
        $optn = n$
      end
    end
  end
  $L_i =$ merge s_i to $optn$-th line of L_{optj}
end

3.4 Layout and Formatting

We choose a static and linear format that aggregates the visual contents from the video and the transcript text in order to re-present the video content. A static format facilitates navigation of the content by presenting the entire video content in a single layout and allowing the learners to go through it in their own pace. Moreover, lectures tend to be organized in a sequential structure that naturally lends itself to a linear layout.

We observe two types sentences: sentences that describe the visual content without adding additional information (e.g.reading out the equation on the board), and sentences that provide further explanation (e.g. connection between different equations). The first type of sentence provide redundant information that is already present in the visual content. Figure **??**

4 Results

5 Evaluation

To evaluate the utility of our *ScribeVideo* interface, we performed a comparative user study comparing three interfaces to watch lecture videos: a standard YouTube player with an interactive transcript, the NoteVideo interface, and our ScribeVideo interface.

5.1 Tasks

Learners performed three types of tasks for lecture videos: summarization, information search, and comprehension. These tasks represent realistic learning scenarios using video lectures, and match common evaluation tasks applied in the literature on video navigation interfaces [].

- **Summarization** task requires the learners to write down an overview of the main contents in the video in a short period of time. We gave learners only three minutes to view and summarize videos that were about eight minutes long. We purposely did not give enough time to watch the entire video so as to motivate the learners to quickly browse through its content. This is a useful task, for example, when the learner wants to get a quick overview of the lecture without watching the entire video, or when the learner wants to quickly review previously watched material. Summaries written by learners are compared to a gold standard list of key contents manually

created together by two referees, including the author of this paper. Using this list, we score the learners' summaries by the number of topics that they cover.

- **Search** tasks involve finding a specific piece of information in a video. The questions included both looking for a visual cue (e.g. *"Find the point in the lecture where the height of the graph of $f(x)$ is denoted with the variable h."*) and textual cues (e.g. *"Find the point in the lecture where the double angle formula for $cos(2x)$ is stated."*). These tasks emulated situations when a learner remembers something from the lecture and wants to find where it appeared in a video, or situations when the learner only wants to access specific information from the video and skip the rest. Performance was measured by the time task completion time as well as accuracy (distance between the learner's answer and the actual location of the specified information).

- **Comprehension** tasks involve both finding or identifying relevant information and understanding the content. To gauge knowledge gain from watching videos, we gave learners a 5-question pre-test (before watching) and a 5-question post-test (while watching). The two sets of questions tested the same type of knowledge but were asked in slightly different ways (i.e. one question would be asked in a *true-or-false* form, and the other question would be in a *fill-in-the-blank* form). The difference in the number of questions answered correctly int he pre- and post- tests was recorded, as well as the task completion time for the post-test..

5.2 Study Protocol

5.3 Results

To be done. Ideally, Learners find information faster and more accurately using our interface. Learners complete higher quality summaries using our interface. Learners score higher in comprehension tasks using our system, and finish the task faster.

6 Conclusion

Acknowledgements

References

BARNES, C., GOLDMAN, D. B., SHECHTMAN, E., AND FINKELSTEIN, A. 2010. Video tapestries with continuous temporal zoom. *ACM Transactions on Graphics (TOG) 29*, 4, 89.

BORECZKY, J., GIRGENSOHN, A., GOLOVCHINSKY, G., AND UCHIHASHI, S. 2000. An interactive comic book presentation for exploring video. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, ACM, 185–192.

CHI, P.-Y., LIU, J., LINDER, J., DONTCHEVA, M., LI, W., AND HARTMANN, B. 2013. Democut: generating concise instructional videos for physical demonstrations. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, ACM, 141–150.

CHOUDARY, C., AND LIU, T. 2007. Summarization of visual content in instructional videos. *Multimedia, IEEE Transactions on 9*, 7, 1443–1455.

CHUN, B.-K., RYU, D.-S., HWANG, W.-I., AND CHO, H.-G. 2006. An automated procedure for word balloon placement in cinema comics. In *Advances in Visual Computing*. Springer, 576–585.

EKIN, A., TEKALP, A. M., AND MEHROTRA, R. 2003. Automatic soccer video analysis and summarization. *Image Processing, IEEE Transactions on 12*, 7, 796–807.

HU, Y., KAUTZ, J., YU, Y., AND WANG, W. 2015. Speaker-following video subtitles. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 11*, 2, 32.

HWANG, W.-I., LEE, P.-J., CHUN, B.-K., RYU, D.-S., AND CHO, H.-G. 2006. Cinema comics: Cartoon generation from video stream. In *GRAPP*, 299–304.

JACKSON, D., NICHOLSON, J., STOECKIGT, G., WROBEL, R., THIEME, A., AND OLIVIER, P. 2013. Panopticon: A parallel video overview system. In *proceedings of the 26th annual ACM symposium on User interface software and technology*, ACM, 123–130.

KIM, J., GUO, P. J., CAI, C. J., LI, S.-W. D., GAJOS, K. Z., AND MILLER, R. C. 2014. Data-driven interaction techniques for improving navigation of educational videos. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, ACM, 563–572.

KIM, J., NGUYEN, P. T., WEIR, S., GUO, P. J., MILLER, R. C., AND GAJOS, K. Z. 2014. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, ACM, 4017–4026.

KNUTH, D. E., AND PLASS, M. F. 1981. Breaking paragraphs into lines. *Software: Practice and Experience 11*, 11, 1119–1184.

KURLANDER, D., SKELLY, T., AND SALESIN, D. 1996. Comic chat. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 225–236.

MONSERRAT, T.-J. K. P., ZHAO, S., MCGEE, K., AND PANDEY, A. V. 2013. Notevideo: Facilitating navigation of blackboard-style lecture videos. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1139–1148.

MONSERRAT, T.-J. K. P., LI, Y., ZHAO, S., AND CAO, X. 2014. L. ive: an integrated interactive video-based learning environment. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, ACM, 3399–3402.

MONSERRAT, T.-J. K. P., LI, Y., ZHAO, S., AND CAO, X. 2014. L. ive: an integrated interactive video-based learning environment. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, ACM, 3399–3402.

PAVEL, A., HARTMANN, B., AND AGRAWALA, M. 2014. Video digests: a browsable, skimmable format for informational lecture videos. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, ACM, 573–582.

RUBIN, S., BERTHOUZOZ, F., MYSORE, G. J., LI, W., AND AGRAWALA, M. 2013. Content-based tools for editing audio stories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, ACM, 113–122.

UCHIHASHI, S., FOOTE, J., GIRGENSOHN, A., AND BORECZKY, J. 1999. Video manga: generating semantically meaningful video summaries. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, ACM, 383–392.

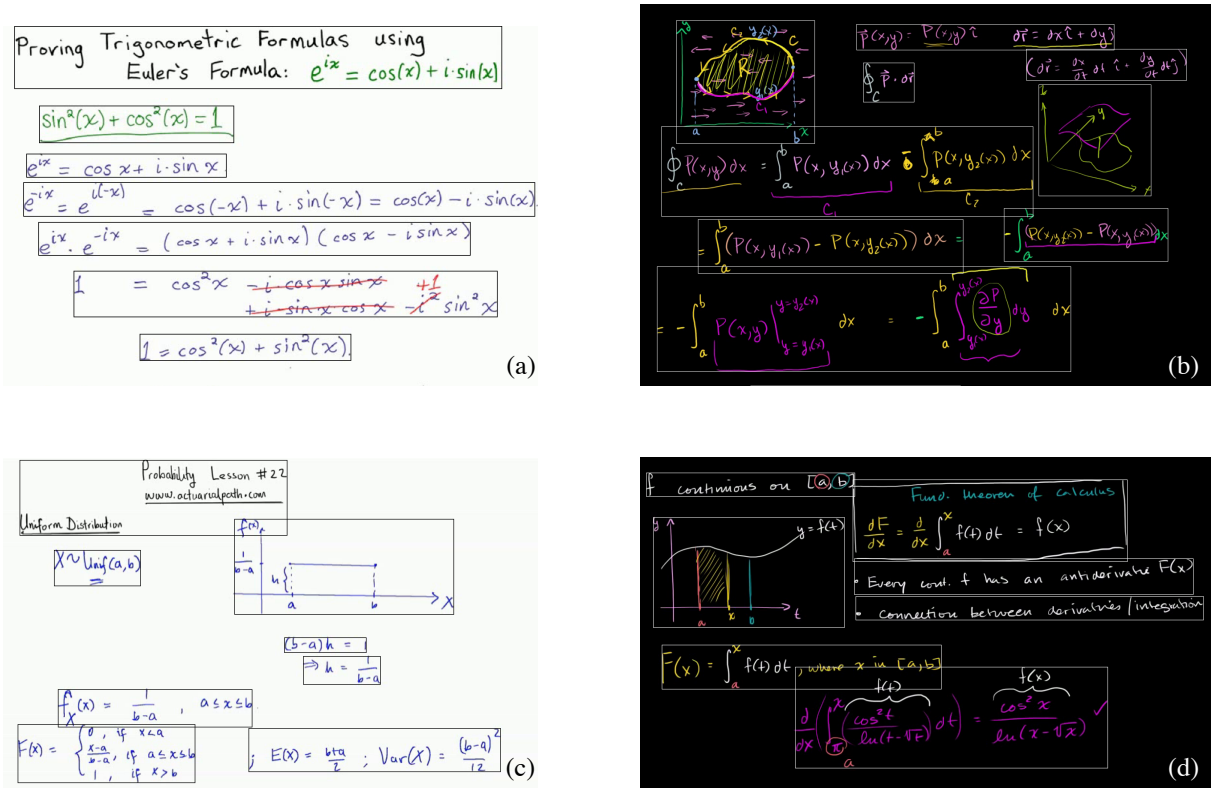

Figure 3: *Examples of lines (i.e. set of strokes that belong together semantically) output from our line-breaking algorithm. Our algorithm successfully identifies meaningful groups even from complex layouts with a mix of equations, figures and graphs.*