Ink Annotations and their Anchoring in Heterogeneous Digital Documents

Xin Wang, Sashi Raghupathy
Tablet PC Ink Parsing Team, Microsoft Corp
One Microsoft Way, Redmond WA, 98007
{xinwang, sashir}@microsoft.com

Abstract

The analysis, especially the anchoring, of free–form ink annotations on mixed–content digital documents is more difficult than that of documents containing only text. In this paper, first, we present our work on analyzing the common user scenarios of ink annotations and their anchoring in such an environment, and we also present a unified way to define and analyze the anchoring of different types of annotations. Then we describe how our ink parsing system integrates the anchoring of and annotation with the recognition of its type. The method can be used in a real–time heterogeneous document analysis system.

1. Introduction

For digital note-taking applications like OneNote or InfiNote, one important goal is to incorporate a wide variety of content like ink, text or images into a single note-taking environment seamlessly. One key difficulty in the implementation of such a system is to handle the ink annotations in a document. For the example in Figure 1, if the user resizes the page, she probably expects the ink blob to stay with the image of the scroll–bar, and the callout to stay with the ink blob. To achieve this goal, the document processing environment must recognize the ink blob and the callout as annotations. It also needs to anchor the callout to the ink blob, and the ink blob to the image.

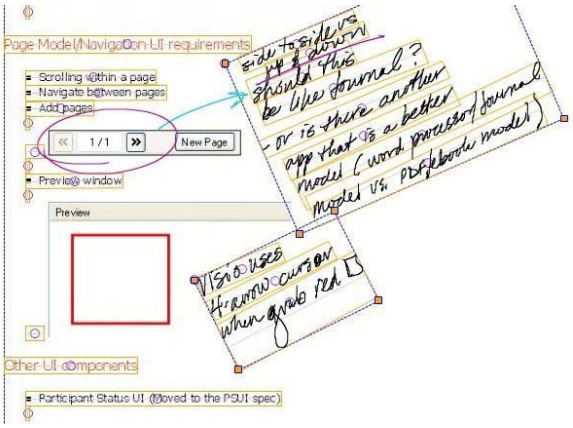


Figure 1. A sample document with a mixture of ink, text and images.

Marshall [3] studied how a user annotates on paper, and how it might impact a digital document environment such as a digital library. Marshall and Brush [4] followed up with research on how to use traditional user interface to faciliate the annotation of a digital document. Shilman and Wei [6] went a step further and investigated how to allow a user to do free–form annotation directly on a digital document of text. They also presented a rule–based annotation classification method that reconciliated multiple hypotheses through heuristically calculated scores.

In [8], we described the architecture and recognition algorithms behind their learning–based ink annotation parsing system in a heterogeneous document environment. But we did not present a detailed analysis and description of the user scenarios of annotations and how to handle their anchoring—a very important part in the parsing of ink annotations.

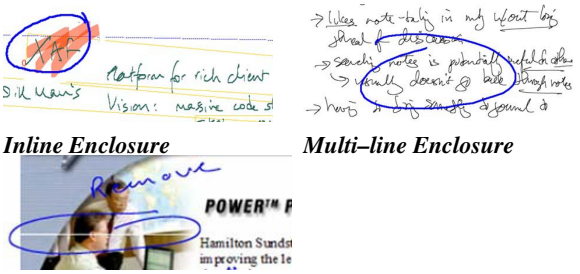
In this paper, we first present our work on the user scenarios of ink annotations and their anchoring in mixed–content documents after examining 17509 real–world digital files with ink annotations. And we also describe a unified way to define and analyze the anchoring of different types of annotation. By doing this, we are able to integrate the anchoring of ink annotation with the recognition of its type ([8]), which helps us in addressing the inherent ambiguity of ink annotations in a mixed–content system.

2. The Anchors of an Annotation

Annotations can "refer" to a wide variety of objects in a document, and consequently have many different types of anchor. For instances, an *underline* can refer to a line of ink

Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)
0-7695-2822-8/07 $25.00 © 2007 IEEE

or text, or only a part of it. It can also refer to an image or other objects in the document. A pair of braces can be used to delimit a couple of lines of ink or text. A callout can point from a word, a line, an image or even another annotation, to some other word, line, image or annotation. It can also simply point from or to a position in the margin or other white spaces. If a free note is a marginalia (e.g. in the margin), it can refer to the entire document or nothing specific; otherwise, it can also refer to some near–by word, line or image.



Inline Enclosure ***Multi–line Enclosure***

Other Type Enclosures

Figure 2. Enclosures

In a word, an anchor α_{anchor} of an annotation $\mathcal{A}_{anno}$ can be any of the following: a span ς_{span}, a position ϖ_{pos}, another annotation $\mathcal{A}'_{anno}$, the entire document φ_{doc}. Here we use *span* to refer to any single or mixture of grammatical units of the underlying document—word, line, paragraph, image, figure, text regions, and etc. As in Figure 2, the span of an enclosure can be: a word as in the example of an inline blob; multiple words as in the example of a multi–line blob; or an image as in the last example. In general, we have

$$\alpha_{anchor} \in \left\{ \varsigma_{span}, \varpi_{pos}, \mathcal{A}_{anno}, \varphi_{doc}, \right\}$$

whereas

$$\varsigma_{span} = \left\{ \begin{array}{l} ink\ word \mid ink\ line \mid ink\ paragraph \\ \mid text\ word \mid text\ line \mid text \\ paragraph \mid image \mid drawing \end{array} \right\}^{+} \quad (1)$$

2.1 Simple Annotations

First, let us consider the anchors of simple annotations that do not refer to another annotation, and how to represent their anchors in the parse tree. The simplest annotations are "range" annotations—horizontal ranges, vertical ranges, and enclosures[6]. As shown in Table 1[1], they can only take one[2] anchor which is usually restricted to the type of a span, ς_{span}. It is natural since most of them are used to mark or to represent a simple action (such as *delete*) on a single *range* of objects. Their function is very similar to the lasso tool in Windows Journal—to *select* an object or multiple objects from the main content of a document. But unlike the lasso tool or other selection mechanisms in traditional document processing environment, they do not require an explicit mode switch [5], especially when the author is editing or composing in the pen mode.

Table 1. Annotations anchored mostly to spans—Horizontal Ranges, Vertical Ranges, and Enclosures

Annotation Type	*Anchors*
Underline	*Mark* (ς_{span})
Strike-through	*Delete* (ς_{span})
Scratch-out	*Delete* (ς_{span})
High-light	*Mark* (ς_{span})
Brace	*Mark* (ς_{span})
Bracket	*Mark* (ς_{span})
MarginalBar	*Mark* (ς_{span})
Parenthesis	*Mark* (ς_{span})
Cross-Out	*Delete* (ς_{span})
Inline Blob	*Mark* (ς_{span})
Multi-Line Blob	*Mark* (ς_{span})
Other Blob	*Mark* (ς_{span})

On the other end of the spectrum, callouts (see Figure 3), which are often used to connect or link two or more objects of the underlying content, can have more than two anchors. As shown in Table 2, *callouts–without –arrow*, *callouts–with–arrow* and *double–ended–callouts* all take two achors. A branch–callouts can have more than three anchors of either the type of a span, ς_{span}, or the type of a position in the document, ϖ_{pos}. Another interesting aspect about the anchors of a callout is that unlike the *range* annotations of Table 1, the relationship between a callout and its anchor is not always the same among its two or more anchors. For example, when anchoring a callout with one arrow, we need to distinguish between its two anchors, and specify which anchor it points from, and which anchor it points to. Similarly,

[1] In the table, we use the convention of $Predicate(\alpha_{annor})$ to represent the anchor of an annotation and the type of relationship it has with the annotation.

[2] As implied by Eq 1, even if the anchor of an annotation consists of multiple words or other grammatical units of the underlying content (Figure 2), we still treat them as a single anchor. Because the *selection* or *marking* relationship denoted by the annotation is the same for all parts of the span. This convention is followed throughout the paper when counting the number of anchors. In system development, to represent an anchor consisting of different types or parts, it might be convenient to use multiple links.

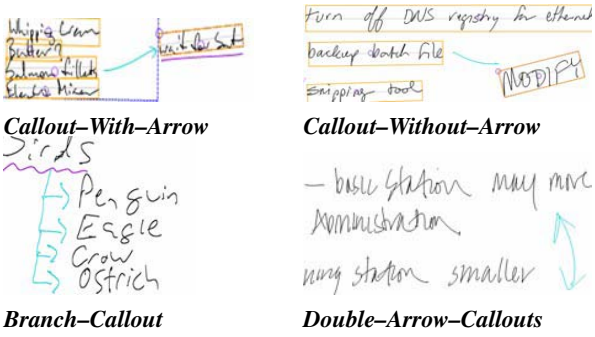


Figure 3. Callouts

for a branch callout, we need to specify which anchor is the root, and which ones are the branches. As shown in Table 2,

Table 2. Callouts

Annotation Type	*Anchors*
Callout w/o Arrow	$Connect\left(o_1, o_2\right)$, $o_1, o_2 \in \left\{\varsigma_{span}, \varpi_{pos}\right\}, o_1 — o_2$
Callout with Arrow	$ConnectFrom\left(o_1\right)$ & $ConnectTo\left(o_2\right)$, $o_1, o_2 \in \left\{\varsigma_{span}, \varpi_{pos}\right\}, o_1 \rightarrow o_2$
Branch Callout	$ConnectFrom\left(o_1\right)$ & $\left\{ConnectTo\left(o_j\right)\right\}_{j=2}^{k}$, $o_1, o_j \in \left\{\varsigma_{span}, \varpi_{pos}\right\}$

for these "directed" callouts, we need to represent their anchors with two different types of link (denoted by predicates $ConnectFrom\ \left(\alpha_{anchor}\right)$ and $ConnectTo\ \left(\alpha_{anchor}\right)$); whereas for callouts–without–arrow, we can simply use one type of link, $Connect\ \left(\alpha_{anchor}\right)$, see Figure 5 and Figure 7. Callout is not the only type of annotation that can take

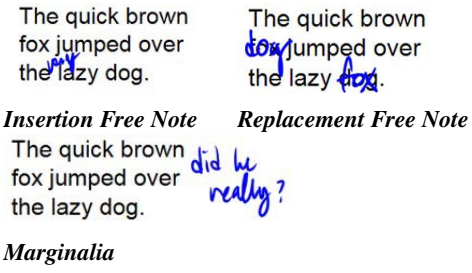


Figure 4. Free Notes

two or more anchors, and have "heterogeneous" relationships with its anchors. For example, free notes [4, 6], as shown in Figure 4, can also have two anchors and needed to be represented by as least two different types of links (see

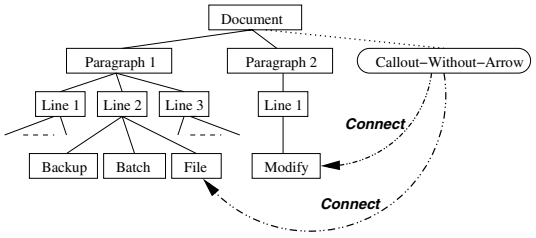


Figure 5. Anchoring the callout without arrow in Figure 3 to the parse tree.

Table 3).

Table 3. Free Notes

Annotation Type	*Anchors*
Marginalia	$Body\left(\varsigma_{span}\right)$, or $Body\left(\varsigma_{span}\right)$ & $Explain\left(\varsigma_{span}, o_2\right)$, $o_2 \in \left\{\varsigma_{span}, \varpi_{pos}\right\}$
Insertion Free Note	$Body\left(\varsigma_{span}\right)$ & $\left[InsertBefore\left(\varsigma_{span}, \varsigma'_{span}\right) \middle\vert InsertAt\left(\varsigma_{span}, \varpi_{pos}\right)\right]$
Replacement Free Note	$Body\left(\varsigma_{span}\right)$ & $Replace\left(\varsigma_{span}, \varsigma'_{span}\right)$

2.2 Composite Annotations

Composite annotations (see Figure 6 and Figure 1) are often used to represent more complicate marking or editing actions, or even action sequences, rather than a simple *selection* or a *selection* with a simple action such as deletion, insertion, or replacement. For example, the last example in Figure 6 represents a complicate action sequence—a replacement, a change of font, and then a movement.

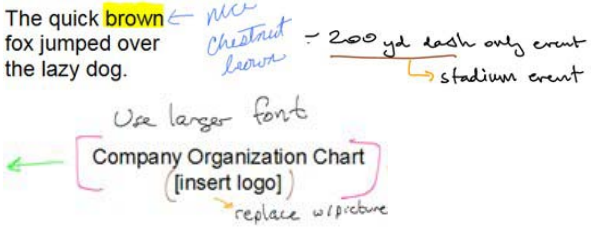


Figure 6. Composite Annotations

As illustrated by these examples, a composite annotation

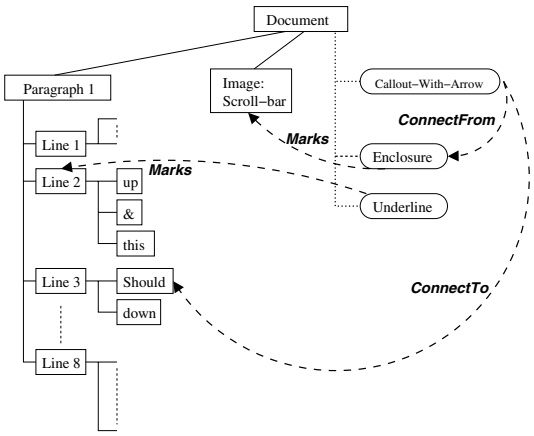


Figure 7. Anchoring annotations in Figure 1 to the parse tree.

usually contains or embeds other annotations. The recursive embedding of annotations can be represented in multiple ways in the parse tree of a document. For example, we can build a subtree of annotations within the parse tree to represent the embedding relationship hierarchically and explicitly. Or since composite annotations are just annotations that have other annotations as anchor, we can represent the above "subtree" structure implicitly through links of anchors. As shown in Figure 7, the callout–with–arrow annotation, is linked to the enclosure that spans the second line of the ink paragraph. The second method actually is more convenient for building ink annotation parsing system in a bottom-up manner.

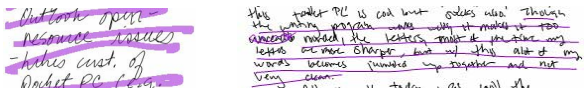


Figure 8. Anchoring a horizontal range could also be an ambiguous and difficult task.

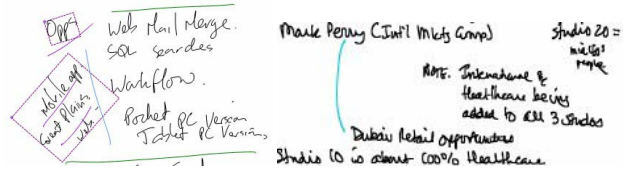


Figure 9. Recognition and anchoring a vertical range could also be an ambiguous and difficult task.

3. Recognition and Anchoring Algorithms

For ink annotations, the same type of ink stroke can be used for multiple purposes. For example, a straight line segment can be used as either an underline, a strike–through, a callout–without–arrow, or a marginal bar. This inherent ambiguity makes the recognition and anchoring of annotations even more complicated than it was described in Section 2. Even for the simplest case of strike–through as shown in Figure 8, the third strike–through in the left example still could be an underline made in a hurry.

Similarly, it is hard to tell at first look whether the vertical lines in Figure 9 are vertical ranges or callouts. Their anchors would be completely different depending on their type. For example, the annotation in the right example of Figure 9 would be anchored to the four lines to its right if it is a vertical range; it would be anchored to the two words "Penway" and "Dubious" if it were a callout.

On the other hand, the ambiguity mentioned above can only be resolved within the context of the specific annotation—the underlying document, other annotations, and specifically, its potential anchors. For example, the annotation in the right example of Figure 9 could not be a vertical range because the four lines to its right are too far away, and of different indentation. Similarly, the vertical line in the left example of Figure 9 is more likely to be a vertical range than a callout.

There are many such "rules" about an annotation and its anchors. Some of them are ambiguous and even contradictory to each other. So instead of capturing such knowledge explicitly in specific rules, we represent them as contextual and geometric features of an annotation hypothesis—soft constraints a learning based classification system can choose to observe or not, for more details, see [8].

In this section, we describe a learning–based method that deals with the type classification and the identification of anchors for an ink annotation simultaneously within context of one other. Since each one provides constraints on the other, even though we generate more hypotheses, the classification task actually becomes easier.

Our method takes both geometric information of the ink strokes of the annotation, and structural and geometric information of the related main content in the classification of an annotation hypothesis.

Each annotation hypothesis H_i is a function of three elements: the set of $n \geq 1$ strokes $\left\{\omega_s\right\}_{s=1}^{n}$, the potential annotation type $\mathcal{T}_{anno}$, and the set of potential anchors $\left\{\alpha_{anchor,j}\right\}_{j=1}^{m}$.

For each annotation hypothesis H_i,

$$H_i = \left(\left\{\omega_s\right\}_{s=1}^{n},\ \mathcal{T}_{anno}, \left\{\alpha_{anchor,j}\right\}_{j=1}^{m}\right), \qquad (2)$$

we uses two types of features, shape features—features that involve only the ink strokes of the annotation, and context features—features that capture geometric and other relationships between an annotation and its potential anchors. For the shape features, we use both the Viola–Jones Filters [7] based image features $\phi_I(\mathcal{A}_{anno})$,

$$\phi_I(\mathcal{A}_{anno}) = \Phi\left(\left\{\omega_i\right\}_{i=1}^{n}\right) \tag{3}$$

and geometric features $\phi_S(A)$ similar to those by Fonseca et al [1] (carefully designed for each annotation type $\mathcal{T}_{anno}$)

$$\phi_S(\mathcal{A}_{anno}) = \Phi\left(\left\{\omega_i\right\}_{i=1}^{n}, \mathcal{T}_{anno}\right). \tag{4}$$

Each annotation type usually imposes its own set of geometric and structural constraints on the annotation and its potential anchors. To capture these constraints and relationships, we also design a separate set of context features $\phi_C(\mathcal{A}_{anno})$ for each type of annotation $\mathcal{T}_{anno}$,

$$\phi_C(\mathcal{A}_{anno}) = \Phi\left(\left\{\omega_i\right\}_{i=1}^{n}, \mathcal{T}_{anno}, \left\{\alpha_{anchor,j}\right\}_{j=1}^{m}\right). \tag{5}$$

With these features, we use AdaBoost.M1 [2] classifiers to eveluate the hypotheses. Even though AdaBoost usually can deal with a large number of features quite efficiently, given that the system deals with multiple classes of annotations, of multiple strokes, with multiple anchoring candidates, it is still impossible to enumerate all possible annotation hypotheses and solve the combinatorial optimization problem globally. Thus extensive pruning of the hypothesis space is necessary to achieve real–time recognition and anchoring of annotations as required by an on–line document system. Fortunately, our experiences showed that with carefully designed pruning heuristics based on domain knowledge about the ink annotation for a digital online document analysis system, a reasonable level of accuracy and performance can be achieved [8].

4. Evaluation

We manually labeled a set of 1294 mixed content documents (similar to Fig 1) to train the AdaBoost classifier. The training set contains both real world documents donated by users of Windows Journal ® and OneNote® 2003 and files collected through specifically designed user scenarios. Altogether they contain about 6974 examples of annotations. Out of these examples, 1413 are set aside for cross–validation. Overall, the algorithm achieves an accuracy of 91.30% on a set of 138 files set aside for evaluation, for more details see [8][3].

[3] In this paper, we did not compare our classification results with other systems mentioned in Section 1. Most of these systems were not designed for ink-on-ink annotations or mixed–content files and could not deal with most of the files in our data sets.

5. Discussion and Conclusion

Ideally, the recognition and anchoring of an annotation should be solved within the context of the entire document in a global manner. We took a more local and bottom-up approach because of the real–time constraint of an online digital document analysis system. The time allocated to parse an annotation when loading a large file is around $1ms$, which is far less the $1s$ or $0.1s$ usually available for most pattern recognition systems.

This presentation is part of an on–going research work on a fast emerging and exciting new area of note-taking. One of our foremost impressions while researching into the user scenarios of mixed content digital documents, is that there is no limit to human ingenuity where their favorite piece of technology is concerned. With the increased adoption the pen-based technolgoies in daily life, we are guaranteed to be overwhelmed with more and more interesting and complicated user scenarios. We are looking forward to enriching our current system to meet the demands of these new user scenarios, and incorporating new methods and techniques developed by the community and contributing to them as well.

References

[1] M. J. Fonseca, C. Pimentel, , and J. A. Jorge. Cali: An online scribble recognizer for calligraphic interfaces. In *AAAI Spring Symposium, Sketch Understanding*, pages 51–58, 2002.

[2] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[3] C. Marshall. Annotation: from paper books to the digital library. In *Proceedings of the ACM Digital Libraries Conference*, 1997.

[4] C. Marshall and A. Brush. Exploring the relationship between personal and public annotations. In *CHI*, pages 812–813, 2002.

[5] E. Saund and E. Lank. Stylus input and editing without prior selection of mode. In *UIST*, pages 213–216, 2003.

[6] M. Shilman and Z. Wei. Recognizing freeform digital ink annotations. In *Document Analysis Systems VI*, pages 322–331, 2004.

[7] P. A. Viola and M. J. Jones. Robust real-time face detection. In *ICCV*, page 747, 2001.

[8] X. Wang, M. Shilman, and S. Raghupathy. Parsing ink annotations on heterogeneous documents. In *3rd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2006.