Stroke-Based Performance Metrics for Handwritten Mathematical Expressions

Richard Zanibbi
rlaz@cs.rit.edu
Amit Pillay
aap2731@rit.edu
Department of Computer Science
Rochester Institute of Technology, NY, USA

Harold Mouchère
harold.mouchere@univ-nantes.fr
Christian Viard-Gaudin
christian.viard-gaudin@univ-nantes.fr
LUNAM, Université de Nantes
IRCCyN, France

Dorothea Blostein
blostein@cs.queensu.ca
School of Computing,
Queen's University, Canada

***Abstract*—Evaluating mathematical expression recognition involves a complex interaction of input primitives (e.g. pen/finger strokes), recognized symbols, and recognized spatial structure. Existing performance metrics simplify this problem by separating the assessment of spatial structure from the assessment of symbol segmentation and classification. These metrics do not characterize the overall accuracy of a pen-based mathematics recognition, making it difficult to compare math recognition algorithms, and preventing the use of machine learning algorithms requiring a criterion function characterizing overall system performance. To address this problem, we introduce performance metrics that bridge the gap from handwritten strokes to spatial structure. Our metrics are computed using bipartite graphs that represent classification, segmentation and spatial structure at the stroke level. Overall correctness of an expression is measured by counting the number of relabelings of nodes and edges needed to make the bipartite graph for a recognition result match the bipartite graph for ground truth. This metric may also be used with other primitive types (e.g. image pixels).**

***Keywords*-Performance Evaluation; Math Recognition; Handwriting Recognition; Graphics Recognition**

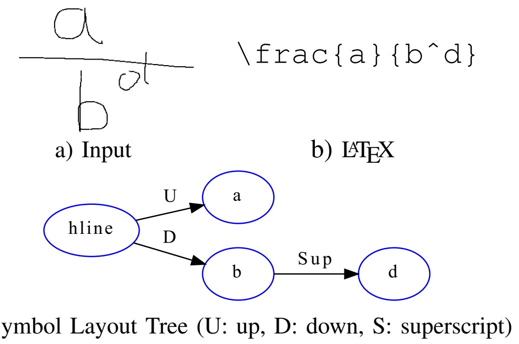
I. Introduction

Evaluating the performance of document analysis systems is an important and difficult problem. As recently summarized by Silva [1], much of the difficulty stems from diversity in goals, input types, input domains, concepts, output granularity, evaluation moments, and evaluation metrics. Our work focuses on addressing performance evaluation issues that arise due to diversity in granularity. Performance metrics are simpler to define for computations in which input and output items have similar granularity. In the case of math recognition, the granularity differs markedly, with a spatial arrangement of strokes or symbols as input, and a hierarchical layout description (e.g. LaTeX, see Figure 1) and/or representation of meaning (e.g. Content MathML, OpenMath) as output. There is a need for standard metrics that permit meaningful comparison of math recognition results [2], [3], both for comparing systems, and for use with machine learning algorithms that optimize system performance.

Our primary contribution is a bipartite graph-based representation of expression structure at the level of primitives (e.g. strokes), that captures errors in both recognized symbols and layout. We were motivated to use a representation at the primitive rather than symbol level, because the distinction between symbol segmentation and structure recognition is sometimes blurred. For example, the configuration "=" can be viewed either as a single symbol consisting of two spatially separated strokes, or as two symbols (short horizontal lines) with one atop another. We consider isolated expressions in this paper, but our method may be adapted in a straight-forward manner for multiple expressions, flowcharts, tables, and even images using pixel regions such as connected components or image patches.

The second contribution of the paper is a set of new error metrics based in our bipartite representation. This includes metrics that characterize overall recognition accuracy, providing a criterion function for expressions, based on strokes as primitives; existing criterion functions use symbols as primitives, as discussed in Section II.

We assume that it is possible to define a ground truth interpretation for a given set of strokes. This excludes undersegmented strokes, in which a single stroke is used to produce two items that must be represented separately in



a) Input

b) LaTeX

c) Symbol Layout Tree (U: up, D: down, S: superscript)

Figure 1. Handwritten Expression Containing Five Strokes and Four Symbols. A LaTeX string (b) or equivalent Symbol Layout Tree (c) may be used to represent symbol arrangement

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.75

the ground truth interpretation (e.g. for a cursive ‘ax’ written using a single stroke, but containing two symbols).

For detailed performance analysis, different metrics are needed to characterize accuracy for specific tasks: segmentation, classification, and parsing. However, in some situations a single value characterizing performance is needed, such as when using machine learning algorithms to optimize system performance as a whole. Thus, in this paper, we discuss several component metrics (section IV) and also discuss methods of combining these into a single overall estimate of performance (Section V).

II. Previous Work in Evaluating Math Recognition

Mathematical expression recognition is an active research field, both for on-line and off-line data [4], [5], [6], Groundthruthed dataset are now available (e.g. [7] (off-line) and [8] (online)). As pointed out by both Lapointe and Blostein [2] and Awal et al. [3], the math recognition domain now needs standard evaluation metrics to support comparisons of existing and newly developed systems.

Most existing approaches to evaluating math recognition compute a distance between the recognized expression and the ground truth, according to different aspects. The expression recognition rate is common, but global and relatively uninformative, as it counts only expressions that precisely match ground truth. Symbol *recognition* rate does not consider symbol *layout*; Baseline recognition checks only if symbols appear on the correct baseline relative to a symbol. Some metrics, such as the average performance index [9] weight errors depending on the depth of nesting for baselines in an expression.

A difficulty in defining an accuracy metric for symbol layout in math, is that the tree-based representation needed to represent symbol layout is unsuitable for use with classical metrics used for text recognition, such as the Levenstein edit distance. One solution is to use tree-edit distance, but this is not used in practice, because of the NP complexity of the existing algorithms to match both tree edges and nodes. An interesting solution by Garain et al. [10] proposes to transform the tree into a token string which allows one to use a edit distance. The drawback of this approach is that it looses some of the edit operations offered by tree edits (like swapping children of a node), leading them to incur high cost in the string-based representation.

Our main contribution is a bipartite graph representation of expression structure at the level of strokes, from which metrics based on Hamming distances may be simply defined, with an intuitive interpretation. Given a set of input primitives for a test expression, our representation prevents the need to match individual strokes, so that only stroke labels and layout relationships between strokes need to be matched, as described in the next section.

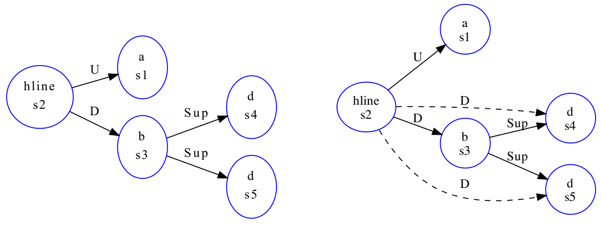
III. Expression Representation

In our approach, the recognizer output and ground truth interpretations must first be converted into bipartite graphs. This is illustrated in Figures 2 and 3. The bipartite representation is shown in Figure 3a), where the nodes of the graph represent each stroke in the expression twice: as an unlabeled input stroke (at left), and with an assigned symbol label and detected relationships (at right, with spatial relationships shown as incoming edges). Note that there are $N(N-1)$ edges in this graph, where N is the number of strokes, and we omit edges from strokes to themselves. For legibility, edges representing ‘no relationship’ are not drawn.

This bipartite graph is constructed from a symbol layout tree: strokes in symbol nodes are split into separate stroke nodes (see Figure 2a), with each stroke possessing the spatial relationships of its associated symbol. All stroke nodes *inherit* the spatial relationships of their ancestors in the layout tree. In Figure 2b, the two strokes corresponding to the ‘d’ inherit the ‘Down’ spatial relationship of the single-stroke ‘b.’ Note that this inheritance applies to all spatial relationships, including containment by square roots, and horizontal adjacency: for example, in ‘$k+m$’, m inherits the ‘Right’ relationship of the $+$ relative to the k.

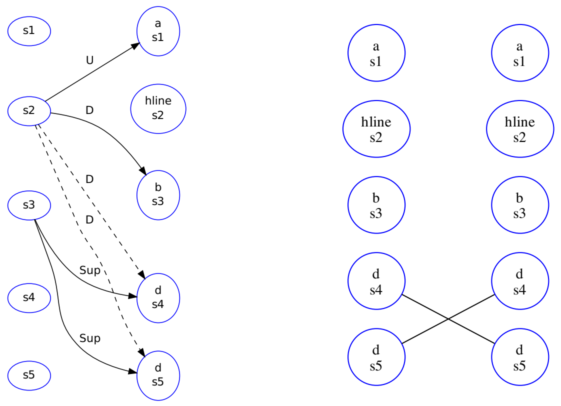
The information presented in the DAG in Figure 2b) can then be converted directly into a bipartite graph, as shown in Figure 3a). Note that strokes with the same set of incoming spatial relationships in Figure 3a are symbols, i.e. *stroke relationships induce a segmentation of strokes into symbols.* It is easier to visualize differences in interpretations using the bipartite representation (as node positions in the graph may be fixed across interpretations), but it is easier to visualize interpreted layout from the DAG representation.

To evaluate symbol segmentation separately from layout, we may use a second bipartite graph: see Figure 3b). An undirected edge is placed between all pairs of *non-identical* strokes belonging to a symbol. Thus a symbol composed of 3 strokes is represented by 6 edges; one isolated stroke corresponding to a symbol is not connected.



a) Stroke Layout Tree b) Stroke Layout DAG

Figure 2. Stroke-level Ground Truth Representations for Expression in Figure 1a). Strokes are labeled using s<*num*>. In b), the stroke layout tree is converted to a DAG by adding incoming spatial relationships at a node to all of its descendents. Spatial relationships are represented by U: up, D: down, Sup: superscript, Sub: subscript, and R: right



a) Stroke labels and layout b) Segmentation

Figure 3. Bipartite Graphs Representing the Expression in Figure 1. Nodes represent strokes, and labeled edges represent spatial relationships. In a), symbol classes are shown using node labels, and spatial relationships using edge labels. This graph represents the same information as in Figure 2b). In b) a segmentation graph is shown, in which strokes belonging to a symbol are connected

IV. Metrics for Specific Error Types

Given two bipartite graphs representing recognizer output and ground truth for strokes in a handwritten math expression, recognition errors may be found directly as disagreeing node or edge labels. A number of examples are provided in Figure 4; incorrect labels and relationships relative to ground truth are shown in red.

a) mislabeled stroke (1 error, Δ_C)
b) misrecognized relationships (2 errors, Δ_L)
c) segmentation error, where the 'd' has been split into two symbols. There are two mislabeled strokes (classification errors), and a spurious spatial relationship between s4 and s5 (3 errors)
d) error similar to that in c), but with the stem of the 'd' misrecognized as being above the fraction line; there is also a missing relationship between s3 and s5 (5 errors)

These errors are summarized in Table I. Additionally, one may count the number of disagreeing stroke pairings in segmentation graphs as illustrated in Figure 3b) (Δ_S). In Figure 4c) and d), two segmentation graph edges from ground truth are missing.

For a set of strokes S and two expressions defined on S represented by bipartite graphs E_1 and E_2, we define the metrics below for specific stroke properties. Let U be the set of all non-identical stroke pairs: $U = \{(p, q) \in S \times S \mid p \neq q\}$, where $|U| = |S||S-1|$.

Each metric below is a Hamming distance, specifically the number of disagreeing labels/relationships. As such, each satisfies the four requirements for a metric [11]: non-negativity, symmetry ($\Delta(E_1, E_2) = \Delta(E_2, E_1)$, $\Delta(E_1, E_1) = 0$, and the triangle inequality: $\Delta(E_1, E_3) \leq \Delta(E_1, E_2) + \Delta(E_2, E_3)$.

- **Classification (Δ_C):** The number of strokes with different symbol labels in the expression graphs E_1 and E_2:
$$\Delta_C(E_1, E_2) = |\{s \in S \mid lab(s, E_1) \neq lab(s, E_2)\}| \tag{1}$$
- **Layout (Δ_L):** Let L_1 and L_2 be the set of labelled edges in expression graphs E_1 and E_2. Layout disagreement is the number of disagreeing edge labels between non-identical strokes:
$$\Delta_L(E_1, E_2) = |U| - |L_1 \cap L_2| \tag{2}$$
- **Segmentation (Δ_S):** This is defined similarly to layout, but using undirected segmentation bipartite graphs (see Figure 3b) B_1 and B_2 constructed on the set of strokes for each symbol relation tree.
$$\Delta_S(E_1, E_2) = |U| - |B_1 \cap B_2| \tag{3}$$

V. Expression-Level Distance Metrics

We now combine our metrics for specific error types (classification, segmentation, and layout) into Expression-Level Distance Metrics that define a single distance measure for two interpretations of an expression. First consider the distance metric $\Delta_B \in [0, 1]$ defined as the number of disagreeing stroke labels and spatial relationships, such as shown in the thumbnail images of Figure 4. This is a Hamming distance, with $|S|^2$ elements in each vector of node/edge labels for a graph.

$$\Delta_B(E_1, E_2) = \frac{\Delta_C + \Delta_L}{|S|^2} \tag{4}$$

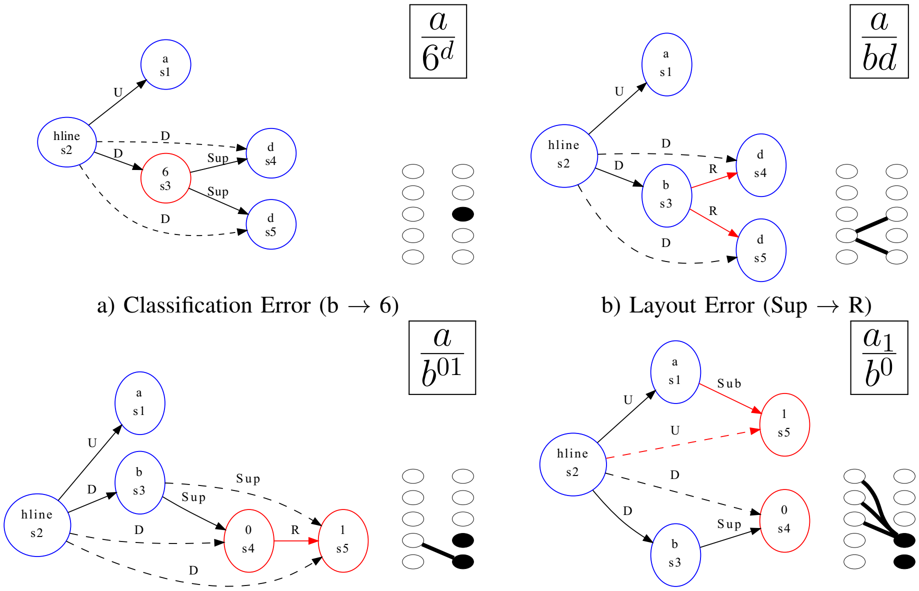
This metric is unweighted, and as a result will produce less distance for classification errors than errors in layout and segmentation (represented implicitly in the layout relationships).

As an absolute measure of the difference between two bipartite graphs Δ_B is sufficient, but one may want to weight errors to make classification errors proportional to segmentation and layout errors. In particular, when comparing algorithms for use in practice, or when using machine learning to optimize the complete recognition system, one may want to weight the different error types.

We define metric $\Delta_E \in [0, 1]$ as the average per-stroke classification, segmentation and layout errors:

$$\Delta_E(E_1, E_2) = \frac{\frac{\Delta_C(E_1,E_2)}{|S|} + \sqrt{\frac{\Delta_S(E_1,E_2)}{|U|}} + \sqrt{\frac{\Delta_L(E_1,E_2)}{|U|}}}{3} \tag{5}$$

We use the square root of the segmentation and spatial relationship distances in order to make them proportional to $|S|$ rather than $|S|^2$ (one could instead divide Δ_L and Δ_S



a) Classification Error (b → 6)

b) Layout Error (Sup → R)

c) Classification and Segmentation (d → {0,1})

d) Classification, Segmentation and Layout

Figure 4. Example Recognition Errors for Expression in Figure 1. In the DAGs errors are shown in red, and by filled nodes and edges in the bipartite graphs. In the bipartite graph thumbnails, nodes correspond to strokes as shown in Figure 3. In part d) there are five errors: the ‘d’ has been separated into two mis-classified strokes, with two spurious spatial relationships (U and Sub), and one missing relationship (the superscript between the ‘b’ and the vertical line in the ‘d’).

Table I
DISTANCE BETWEEN EXPRESSIONS IN FIGURE 4 AND GROUND-TRUTH IN FIGURES 2B) AND 3A)

Fig. 4	Δ_C	Δ_S	Δ_L	Δ_B	Δ_E
a)	1	0	0	0.04	0.067
b)	0	0	2	0.08	0.105
c)	2	2	1	0.12	0.313
d)	2	2	3	0.2	0.368

by $|S-1|$ for the same reason). This prevents differences in segments and spatial relationships from being weighted less heavily than differences in stroke (symbol) classification labels. As each component distance is in $[0,1]$, Δ_E also lies in the interval $[0,1]$.

Δ_B and Δ_E are proper metrics. They are non-negative, symmetric, and the distance from a layout tree to itself is 0. As the square root of non-negative values is an order-preserving monotonic function, the square root of a metric is also a metric. Given that Δ_C, Δ_S and Δ_L are proper metrics, their sum obeys the triangle inequality by definition. Similarly, using the average of their sum does not invalidate the metric property.

Both Δ_B and Δ_E require $O(|S|^2)$ time to compute. In practice, $|S|$ tends to be relatively small, and so the quadratic complexity is not a significant concern. Further, absent spatial relationships need never be explicitly compared: we can simply count labels and relationships present in at least one of the two input graphs.

In order to illustrate and compare these metrics, Table I shows these five distances between errors and its ground-truth. Notice that classification errors are weighted more heavily, and that in general the computed distance/error value is higher for Δ_E than Δ_B.

VI. GENERALIZATION: SEGMENTS AND PIXELS

We have assumed that no stroke corresponds to more than one symbol in the input (i.e. no stroke is under-segmented). This assumption may be removed if we use finer-grained primitives, such as line segments rather than whole strokes. A single stroke containing the two symbols *ax* can then be partitioned, and the resulting segmentation evaluated.

Document images often have some symbols overlapping within a single connected component, such as in: $\frac{y}{x}$, where the fraction line and y may intersect. In this case we can deconstruct connected components into smaller sub-components that correspond to small contiguous regions, or as a more extreme approach, taking pixels to be the primitives.

Using the smallest possible primitives (e.g. pixels) is attractive because under-segmentation cannot occur; however, efficiency may become a problem, as the bipartite graphs/DAGs would be very large. Pixel-level ground truth is imprecise; however, this level of ground truthing is common in computer vision, where it is understood that the human interpretation involved in constructing ground truth results in residual ‘errors’ for decisions within ambiguous regions (e.g.

identifying the specific split point between two connected symbols drawn with a single stroke (e.g. ax)).

With appropriate primitives, the metrics presented may be used as criterion functions for machine learning algorithms. In most cases, losses for errors in stroke labeling and relationships will need to be 'softened' to values in $[0,1]$ rather than $\{0,1\}$, e.g. to avoid discontinuities in the error surface when using algorithms based on gradient descent. These 'soft' errors may be obtained using additional metrics for stroke labels and relationships (e.g. probabilities or fuzzy values).

VII. Conclusion

We have presented new metrics for comparing the similarity of two interpretations of a set of online strokes, with application to pen-based mathematics recognition. Our approach is novel in that it uses strokes rather than symbols as the basis for comparing symbol and structure recognition results. This has the advantage of providing a broader characterization of system performance, allowing expression-level performance to be assessed in terms of input primitives. Our metrics can be efficiently computed, in time $O(n^2)$, where n is the number of strokes. Note that handwritten expressions typically consist of a relatively small number of strokes. The approach can also be easily adapted other pen-based domains, such as recognition of flowcharts, and for use in images.

An open question is whether the metric can be usefully applied when one cannot assume that the sets primitives for two recognition results being compared match (e.g. for Mathematical Information Retrieval (MIR) applications). A related issue is defining metrics for evaluation of mathematical content (i.e. mathematical syntax of a recognized expression); the method presented in this paper addresses only evaluation of layout. As mathematical content is normally represented hierarchically by operator trees, it may be possible to employ a bipartite graph-based approach to evaluation there as well, again using input primitives as the nodes in the graph.

Acknowledgements: This material is based upon work supported by the National Science Foundation under Grant No. IIS-1016815, the Natural Sciences and Engineering Research Council of Canada, the Xerox Foundation, and the Center for Emerging and Innovative Sciences (NYSTAR).

References

[1] A. Sliva, "Metrics for evaluating performance in document analysis: application to tables," *Int'l J. Document Analysis and Recognition*, vol. 14, pp. 101–109, 2011.

[2] A. Lapointe and D. Blostein, "Issues in performance evaluation: A case study of math recognition." IEEE Computer Society, 2009, pp. 1355–1359.

[3] A.-M. Awal, H. Mouchere, and C. Viard-Gaudin, "The problem of handwritten mathematical expression recognition evaluation," in *Int'l Conf. on Frontiers in Handwriting Recognition*, Kolkata, India, 2010, pp. 646–651.

[4] D. Blostein and A. Grbavec, "Recognition of mathematical notation," in *Handbook of Character Recognition and Document Image Analysis*. World Scientific Publishing Company, 1997, pp. 557–582.

[5] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition*, vol. 3, pp. 3–15, Aug 2000.

[6] U. Garain and B. Chaudhuri, *OCR of Printed Mathematical Expressions*. Springer, 2007, pp. 235–259.

[7] S. Uchida, A. Nomura, and M. Suzuki, "Quantitative analysis of mathematical documents," *Int'l J. Document Analysis and Recognition*, vol. 7, no. 4, pp. 211–218, 2005.

[8] S. MacLean, G. Labahn, E. Lank, M. Marzouk, and D. Tausky, "Grammar-based techniques for creating ground-truthed sketch corpora," *Int'l. J. Document Analysis and Recognition*, vol. 14, no. 1, pp. 65–74, 2011.

[9] U. Garain and B. Chaudhuri, "A corpus for OCR research on mathematical expressions," *Int'l J. Document Analysis and Recognition*, vol. 7, no. 4, pp. 241–259, 2005.

[10] K. Sain, A. Dasgupta, and U. Garain, "Emers: a tree matching-based performance evaluation of mathematical expression recognition systems," *Int'l J. Document Analysis and Recognition*, no. 14, pp. 75–85, 2011.

[11] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. Wiley, 2001.