Line-of-Sight Stroke Graphs and Parzen Shape Context Features for Handwritten Math Formula Representation and Symbol Segmentation

Lei Hu
Department of Computer Science
Rochester Institute of Technology
Rochester, USA
lei.hu@rit.edu

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

***Abstract*—This paper presents a new representation for handwritten math formulae: a Line-of-Sight (LOS) graph over handwritten strokes, computed using stroke convex hulls. Experimental results using the CROHME 2012 and 2014 datasets show that LOS graphs capture the visual structure of handwritten formulae better than commonly used graphs such as Time-series, Minimum Spanning Trees, and k-Nearest Neighbor graphs. We then introduce a shape context-based feature (Parzen window Shape Contexts (PSC)) which is combined with simple geometric features and the distance in time between strokes to obtain state-of-the-art symbol segmentation results (92.43% F-measure for CROHME 2014). This result is obtained using a simple method, without use of OCR or an expression grammar. A binary random forest classifier identifies which LOS graph edges represent stroke pairs that should be merged into symbols, with connected components over merged strokes defining symbols. Line-of-Sight graphs and Parzen Shape Contexts represent visual structure well, and might be usefully applied to other notations.**

***Keywords*-Line-of-Sight graph, symbol segmentation, handwritten math recognition, shape contexts**

I. Introduction

Math expressions are an essential part of scientific communication. Recognizing handwritten expressions written on tablets and other touch-sensitive devices would be helpful in document editing, mathematics education applications, and search engines that support mathematical notation in queries. In this paper, we are interested in recognizing Symbol Layout Trees (SLTs) for expressions, which represent expression appearance by a set of symbols with spatial relationships between them (e.g., *Right-adjacent, Subscript, Above* [1]). SLTs represent information similar to LaTeX formulae, but without formatting information.

Recognizing handwritten formulae requires three main tasks: symbol segmentation, symbol recognition and structural analysis. While often implicit in the literature [1], all tasks require a graph-based representation for handwritten strokes in the expression. These *stroke graphs* constrain stroke and symbol relationships considered while searching for the best interpretation of a formula. An ideal stroke graph contains enough edges to represent all spatial relationships and symbols (i.e., perfect recall), while containing as few extraneous edges as possible (i.e., high precision). We want to be able to express the correct expression, but we also want few additional edges so that we increase the likelihood of producing the correct interpretation.

In this paper, we propose Line-of-Sight (LOS) stroke graphs for representing handwritten formulae written online, and introduce a symbol segmentation technique using LOS graphs and new Parzen window-modified Shape Context features (PSC). We first present existing stroke graph representations in Section II, and then algorithms for constructing LOS stroke graphs in Section III. In Section IV we compare different graph representations using the CROHME competition benchmarks [2], and find that LOS graphs are able to represent the most expressions correctly, while still having a reasonable Precision. In Section V we present our LOS-based segmenter using PSC features, which obtains state-of-the-art symbol segmentation results for the CROHME 2014 data set. We then conclude and identify directions for future work in Section VI.

II. Handwritten Stroke Graph Representations

In this Section we briefly introduce stroke-level graph representations used to parse formulae written online. Nodes represent individual strokes, while edges represent possible relationships between strokes, such as identifying strokes belonging to the same symbol, and identifying spatial relationships between symbols such as right-adjacency ($\mathbf{R}$) or superscript ($\mathbf{Sup}$). Details regarding using stroke graphs to represent formula appearance may be found elsewhere [2].

The graph types below define edge subsets for the *complete* graph with $\binom{n}{2} = n(n-1)/2$ undirected edges between all stroke pairs. The motivation to use more compact graphs is to reduce the number of irrelevant edges, making both training and parsing more efficient and accurate. However, pruning stroke pairs can reduce the space of representable expressions, as we will later show in Section IV.

Time-series. Time-series graphs representing the sequence in which strokes are written are common [3], [4]. Many current systems that parse formula using a modified Cocke-Younger-Kasami (CYK) algorithm consider strokes

in time order [2]. A Time-series graph can be represented using an undirected edge between each stroke and its successive stroke (except for the final stroke). Time-series graphs are unable to directly represent formulae with delayed strokes (e.g., the dot for an 'i' written after writing other symbols) or non-local relationships. These graphs are compact; as sequences they are a restricted form of tree, with $n-1$ undirected edges for n strokes.

k-NN Graph. Others such as Eto and Suzuki [5] have used k-Nearest Neighbor graphs (k-NN). In a k-NN graph, there is an undirected edge from each stroke to each of its k nearest neighboring strokes. This allows strokes that are nearby in space but not necessarily time to be related in the graph (e.g., the dot of an 'i' written after a delay). k-NN graphs are less compact than Time-series, with $O(kn)$ edges. Smaller values of k may produce a disconnected graph, splitting a formula into two or more sub-expressions.

Minimum Spanning Tree (MST). Matsakis [6] uses a stroke graph where edges are defined by the Minimum Spanning Tree (MST) over strokes, based on the distance between stroke bounding boxes. The MST is more compact than k-NN graphs, with $n-1$ undirected edges, and guarantees that the graph is connected. A limitation is that edges for relationships between non-neighboring strokes may be absent.

Delaunay Triangulation Hirata et al. employ Delaunay Triangulation (DT) for stroke graphs [7]. A DT for a set of 2D points S is a triangulation where no point in p is inside the circumcircle of any triangle [8]. It is the dual structure of the Voronoi diagram. Each point in the triangulation is adjacent to all vertices (strokes) in attached triangles. Like MSTs, Delaunay Triangulations guarantee a connected graph, but have more edges: $3n-3-h$, where h is the number of points of the convex hull. This allows more relationships to be represented in a DT than an MST.

In the next Section we propose using a new graph representation, Line-of-Sight graphs.

III. Line-of-Sight (LOS) Graphs over Strokes

We came to consider Line-of-Sight graphs after creating k-NN graphs with large values of k, and finding that expressions with large exponents were having edges between horizontally adjacent symbols pruned (e.g., for $k=2$, in $x^{2a}+1$, the x and $+$ do not share an edge). However, an unobstructed line could often be drawn between strokes with relationships pruned by k-NN in cases like these. A Line-of-Sight graph [8] is a visibility graph defining which nodes can 'see' one another. For our stroke graphs, the LOS graphs define edges for strokes that can be 'seen' from the bounding box center of another stroke, or vice versa. An example LOS stroke graph is shown in Figure 1.

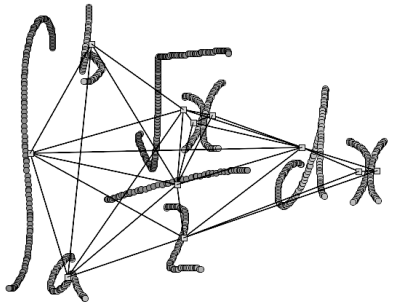


Figure 1: Line-of-Sight (LOS) Graph for a Handwritten Expression. Small square nodes represent bounding box centers for eleven handwritten strokes. Edges represent mutually 'visible' strokes. Two strokes share an edge if an unobstructed line can be drawn from one stroke's bounding box center to a point on the convex hull for the other stroke (see Algorithm 1)

Algorithm 1 Line-of-Sight (LOS) Graph Construction

Input: S, the set of handwritten strokes for an expression

let $E=\emptyset$ be an empty edge set
for each stroke $s \in S$ with b.box center $\mathbf{s_c}=(x_0,y_0)$, **do**
  let unblocked angle interval set $U=\{[0,\ 2\pi]\}$ radians
  for $t \in S-s$, by increasing distance from s, **do**
    let $\theta_{min}=+\infty$, $\theta_{max}=-\infty$
    for each node $\mathbf{n}=(x_h,y_h)$ in the convex hull of t **do**
      let vector $\mathbf{w}=\mathbf{n}-\mathbf{s_c}=(x_h-x_0,\ y_h-y_0)$
      let angle θ between $\mathbf{w}$ and horizontal $\mathbf{h}=(1,0)$ be

$$\theta=\begin{cases}\arccos\left(\frac{\mathbf{w}\cdot\mathbf{h}}{||\mathbf{w}||\,||\mathbf{h}||}\right) & \text{if } y_h \geq y_0\\ 2\pi-\arccos\left(\frac{\mathbf{w}\cdot\mathbf{h}}{||\mathbf{w}||\,||\mathbf{h}||}\right) & \text{if } y_h < y_0\end{cases}$$

      let $\theta_{min}=\min(\theta_{min},\theta)$, $\theta_{max}=\max(\theta_{max},\theta)$
    let hull interval $\mathbf{h}=[\theta_{min},\theta_{max}]$
    if $V=\bigcup_{\mathbf{u}\in U}\mathbf{u}-\mathbf{h}$ contains a non-empty interval
      let $E=E\cup(s,t)\cup(t,s)$ *(s_c 'sees' t)*
    let $U=\bigcup_{\mathbf{u}\in U}(\mathbf{u}-\mathbf{v_0}-\ldots-\mathbf{v_n}),\ V=\{\mathbf{v_0},\ldots,\mathbf{v_n}\}$

return E *(LOS stroke edges)*

Algorithm 1 constructs LOS graphs from handwritten stroke set S. For a given stroke $s \in S$, we consider whether other strokes are visible by incrementally blocking angles covered by strokes. Strokes other than s are sorted by the smallest distance between two sample points: one point is from s, while the other point is from the other stroke. To test their visibility, strokes are represented by their convex hull [8]: the smallest convex polygon containing sample points in the stroke. While 'looking' at each stroke t from s_c, if we find an unobstructed angle range between s_c and the convex hull for t, we define an undirected edge between s and the other stroke (as two *directed* edges). We then remove the visible angle range for t from the set of unblocked angle intervals U. Figure 2 illustrates how visible and blocked angles are defined using convex hulls.

Algorithm 2 recovers missing labeled edges in stroke graph representations, including LOS, to make sure that

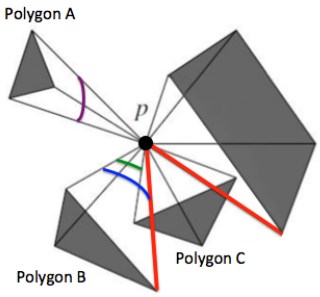


Figure 2: Line-of-Sight Illustration (adapted from [8]). 'Looking' from point p, the purple and blue arcs represent the angle ranges blocked by Polygons A and B, ignoring other polygons in the scene (**h** in Algorithm 1). The green arc represents the visible range of Polygon B (V), taking into account the portion blocked by Polygon C. The red lines are sight lines to vertices blocked by Polygon C.

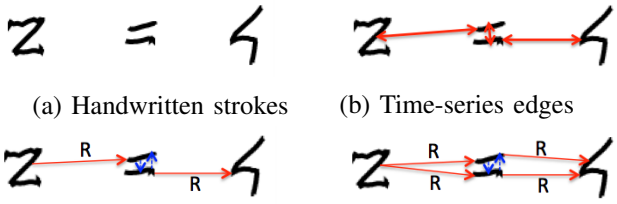
Algorithm 2 Assigning Labels to an Undirected Stroke Graph. *Label *:* merge strokes into symbol; *Label _:* undefined

Inputs:
$G = (S, E)$ an undirected and unlabeled stroke graph
$G_t = (S, E_t, \lambda_{St}, \lambda_{Et})$ a *complete* labeled stroke graph with same node set S (strokes), label functions λ_{St} and λ_{Et}

let $\lambda_S = \lambda_{St}$ *(label strokes)*
let $\phi : S \rightarrow 2^S$ maps strokes to stroke sets for symbols
  Initially, $\phi(s) = \{s\}, \forall s \in S$
let $E_c = E \cap E_t$ be the common edges in G and G_t
for $e = (s_1, s_2) \in E_c$ **do**
  $\lambda_E(e) = \lambda_{Et}(e)$ *(label edge)*
  if $\lambda_E(e) = *$ *(define symbols)*
    let $\phi(s_1) = \phi(s_1) \cup \phi(s_2),\ \phi(s_2) = \phi(s_1)$
    for $(s_i, s_j) \in \phi(s_1) \times \phi(s_1)$ where $s_i \neq s_j$ **do**
      let $\lambda_E(s_i, s_j) = \lambda_E(s_j, s_i) = *$
(refine relationships)
for $e = (s_1, s_2) \in E_c$ **do**
  if $\lambda_E(e)$ is labeled with a relationship (i.e., not * or _)
    for $s_o \in \phi(s_1)$ where $s_o \neq s_1$ **do**
      let $\lambda_E(s_o, s_2) = \lambda_E(s_1, s_2)$
return (λ_S, λ_E) *(stroke and edge labels for G)*

edge labels represent a valid SLT. Connected components of merge-labeled stroke edges are converted into cliques (i.e., all pairs of strokes in a symbol share a merge-labeled edge), and all strokes in a symbol are given the *same* relationship with strokes in other symbols. The algorithm ignores conflicting *undefined* (_) labels and assumes no conflicting relationship labels, which is valid for well-defined SLTs.

Note that if we have a *labeled* stroke graph, we can pass it in both inputs for Algorithm 2 to insure that the SLT representation is consistent, e.g., for recognition results [9]. In fact, the symbol segmenter described later in this paper uses Algorithm 2 in this way.



(a) Handwritten strokes (b) Time-series edges

(c) Edge types (d) Recovered SLT ('$z = 4$')

Figure 3: Recovering Edge Labels with Algorithm 2. Edges in Time-series graph (b) are assigned corresponding labels from Ground Truth in (c). (d) is then produced from (c) by requiring strokes in a symbol to have the same spatial relationship with strokes in other symbols. The Time-series graph in (a) represents the same SLT as in Ground Truth

Figure 3 illustrates recovering an SLT from a Time-series graph for expression $z = 4$, using the corresponding *complete* ground truth graph with labels for all strokes and stroke pairs (identical to Figure 3(d)). In Figure 3(c) and (d), there are two directed * (merge) edges between the two lines in the equals sign.

From a different perspective, Algorithm 2 can test whether a stroke graph has sufficient edges to represent labeled edges in a ground truth SLT stroke graph. We use this to test the expressivity (coverage) of different stroke graph types in the next Section.

IV. GRAPH COVERAGE EXPERIMENTS

Datasets and Performance Metrics. Our experiments test the expressivity (coverage) of LOS and other stroke graph representations. We use data from the Competition on Recognition of On-line Handwritten Mathematical Expressions (CROHME) [2]. CROHME 2012 has 1336 training and 486 test expressions; CROHME 2014 contains more structurally complex formulae, and is larger with 8834 training and 986 test expressions.

For each test expression, all graph representations are passed along with Ground Truth to Algorithm 2. Performance metrics are then computed using the resulting graph. We compare SLT graph coverage using four metrics. First, the number of expressions that can be correctly represented: if all labeled ground truth edges and recovered, the expression is represented correctly (as in Figure 3). At the level of *directed* edges, we also consider the Recall for labeled edges, Precision of selected edges, and F-score.

1) Recall $= \frac{|\text{ Labeled Recovered Edges }|}{|\text{ Labeled Ground Truth Edges }|}$

2) Precision $= \frac{|\text{ Labeled Recovered Edges }|}{|\text{ Graph Edges }|}$

3) F-score $= 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$

Graph Construction and Distance Metrics. MST, k-NN and LOS require distances between stroke pairs for construction. We consider three Euclidean distances:

1) AC: distance between two strokes' averaged center/center-of-mass (AC), the mean of x and y coordinates for the set of stroke sample points.
2) BBC: distance between bounding box centers (BBC).
3) CPP: the smallest distance between two sample points from each of the strokes, the *Closest Point Pair*.

We choose the distance metric and stroke representation based on model selection experiments [9]. AC works best for MST, while CPP distance works best for k-NN. For k-NN, 2-NN achieves the highest F-score, while 6-NN achieves the highest precision for k-NN with a recall higher than 99% [9].

For Delaunay Triangulation, we do not require a stroke pair distance, but instead need to a point to represent strokes, for which we try the center-of-mass (AC) or bounding box center (BBC); AC worked best.

Results. Table I shows results for the CROHME 2012 and 2014 test sets. All graph types have stable edge metrics across the datasets. For complete graphs containing directed edges between all stroke pairs, Recall is perfect, but over 90% of the directed edges are irrelevant (Precision $< 10\%$).

The highest Precision and F-scores are obtained for MST, suggesting these edges frequently belong to the SLT. Time-series graphs have the second-highest precision and F-score values. For expressions on a single baseline, the MST and Time-series graphs may be identical. This high Precision is partly cause by having fewer edges than the other types, which also leads to very low expression coverage ($< 45\%$).

For k-NN, as one expects edge Recall increases with k, while Precision decreases more rapidly. 2-NN obtains the lowest expression coverage for CROHME 2012, while 6-NN obtains the second-highest expression rate for both data sets. Using $k \geq 6$ mostly decreases Precision and F-score. Delaunay obtains the next-highest expression rate. While Delaunay has higher Precision and F-score through producing fewer edges than 6-NN and LOS, its expression coverage is 13%-14% lower than 6-NN.

LOS obtains the highest expression coverage ($> 98.5\%$) with a slightly higher edge Precision than 6-NN of roughly 30%. LOS misses fewer than 0.1% of labeled ground truth edges. Table II summarizes missing edges in the LOS results. Most missing edges are 'Right' relationships. Almost all edges with a merge label can be covered by the Line-of-Sight graph, as related strokes are often close to and can 'see' one another. In Figure 4, we see missing edges caused by using a single 'eye' at the stroke bounding box center, or completely blocked sight lines.

The nearly perfect Recall for merge edges in LOS graphs provides a strong foundation for graph-based symbol segmentation, which we discuss in the next Section. Work on parsing with LOS graphs using visual features may be found in a companion paper [10].

Table I: Coverage Comparison for Stroke Graph Types. Percentage of representable CROHME expressions (SLTs) are shown along with metrics for directed stroke edges

CROHME 2012 Test (486 Expressions)

		Stroke Pair Edges		
	Expr. (%)	**Recall**	**Precision**	**F-score**
Complete	*100.00*	*1.000*	*0.087*	*0.159*
LOS	**98.56**	**0.999**	0.309	0.472
6-NN	89.92	0.994	0.286	0.444
Delaunay	76.75	0.977	0.388	0.555
MST	36.21	0.882	**0.921**	**0.901**
Time-series	31.28	0.878	0.917	0.897
2-NN	24.90	0.879	0.708	0.784

CROHME 2014 Test (986 Expressions)

		Stroke Pair Edges		
	Expr. (%)	**Recall**	**Precision**	**F-score**
Complete	*100.00*	*1.000*	*0.091*	*0.167*
LOS	**98.99**	**0.999**	0.297	0.458
6-NN	95.81	0.994	0.283	0.441
Delaunay	79.11	0.973	0.391	0.558
2-NN	44.93	0.885	0.685	0.773
MST	42.39	0.875	**0.899**	**0.887**
Time-series	40.77	0.868	0.891	0.879

Complete: all stroke pairs, *LOS:* Line-of-Sight
2/6-NN: k-nearest neighbor, *MST:* Minimum Spanning Tree
Delaunay: Delaunay Triangulation

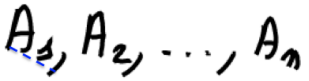
Table II: Number of Missing Edges for Line-of-Sight Graphs. CROHME 2012/2014 Test correspond to Table I

CROHME	Total	*	R	Sub	Sup	Above	Below	Inside
2012 Train	14		14					
2012 Test	13		10				3	
2014 Train	358	2	235	65	24	16	16	
2014 Test	22		22					

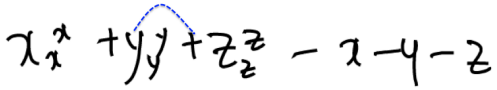
* *merge (into symbol)*

V. Segmentation using LOS Graphs and Parzen Shape Contexts (PSC)

In this Section we present a new technique for segmenting handwritten symbols using LOS graphs and modified Shape Context features [11]. Our segmentation algorithm is simple, using a classifier to identify which directed LOS stroke pair edges correspond to strokes that should be merged. We will now briefly review work on handwritten math symbol segmentation, provide a description of the segmentation



(a) Sight lines for leftmost 'A' and comma blocked (bottom-left of 'A' and comma *can* see one another)



(b) Sight line between leftmost 'y' and the second '$+$' blocked by subscript and exponent y's

Figure 4: Examples of Missing Right-Adjacency (**R**) Edges in Line-of-Sight graphs

algorithm and features, and present segmentation results for the CROHME 2012 and 2014 benchmarks.

Related Work. There have been many graph-based segmentation methods for online handwritten formulae [3], [6], [12]. Toyozumi et al. [12] use a candidate character lattice method, where the closest distance between points on two strokes along with language constraints are used to determine whether strokes should be merged. Matsakis [6] proposes a minimum spanning tree (MST) approach, where each node in the MST represents a stroke, with distances defined by the Euclidean distance between stroke bounding box centers. A limitation is that this technique only considers connected subtrees in the MST for partitioning.

Other methods include Smithies et al. [4] progressive segmentation method, which assumes that symbols are written one-at-a-time. After four strokes are written, the segmenter generates all possible groupings. Strokes from the highest confidence candidate symbol are removed. The process then repeats after another four strokes have been written. Kosmala et al. [13] propose a segmentation method based on Hidden Markov Models (HMM). Discrete left-to-right HMMs without skips and with differing numbers of states are used. A space model is also introduced to represent spaces between symbols. Many recent techniques perform segmentation as a sub-routine while parsing handwritten strokes using an expression grammar, e.g., using a modified Cocke-Younger-Kasami (CYK) algorithm [2].

Hu and Zanibbi [3] classify pairs of strokes in time order as merge/split, i.e., using the Time-series graph for strokes. An AdaBoost classifier with multi-scale shape contexts and symbol classification confidence features is used. In this paper we extend this work, but use random forests applied to Line-of-Sight graphs, do not use classification features, and improve the shape context features.

Stroke Preprocessing and Image Generation. To reduce the effects of sample noise, writing jitter and differences in resolution between expressions, we preprocess strokes and render the expression as a binary image. Preprocessing contains four steps: duplicate point filtering, size normalization, smoothing and resampling.

We first delete duplicate points which have the same (x, y) coordinate as the previous point, because they are uninformative. To reduce the influence of writing velocity and differences in the coordinate range and resolution for different stroke recording devices, we normalize y coordinate values to be in the interval $[0, 1]$, while preserving the width-height aspect ratio for x coordinates. To reduce noise caused by stylus/finger jitter, we smooth all strokes. For each stroke, with the exception of the first and last points, we replace the coordinate of each point by the average of the current, previous, and next coordinate. Finally, we use linear interpolation to resample the expression and render it as a binary image. For the image we use a fixed height of 200 pixels, and then set the width to preserve the width/height aspect ratio of the formula. In generating the image we interpolate ten points between each consecutive pair of sample points, and remove any duplicates.

Segmentation Algorithm. Our algorithm is fairly simple:

1) Construct a stroke LOS graph using Algorithm 1
2) Use a binary classifier to classify all directed edges as * (merge into symbol) or '_' (undefined)
3) Define symbols by converting connected components for * labels into cliques; pass the graph from Step 2 as *both* inputs for Algorithm 2

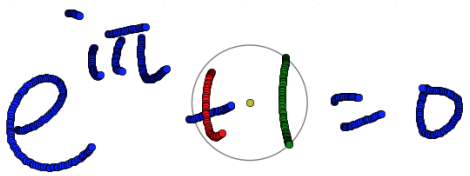
We create a random forest for classification in Step 2, using the Python scikit-learn library [14]. We use 129 features, which are described below. These include the distance in time between strokes ('time gap,' e.g., the first and third handwritten strokes have a time gap of two), Parzen window-modified Shape Context features (PSC) and geometric features.

Parzen Shape Context Features (PSC). A Shape Context characterizes the relative position and density of points (pixels) in an image around a given point using a log-polar histogram [11]. Shape Contexts have been widely used in computer vision for shape matching and classification, as these local representations of appearance and context are often globally discriminative.

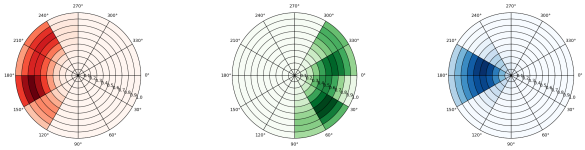
In our work, we use Shape Contexts to characterize the density of points (pixels) in an expression image around two strokes being considered for merging. First, we produce smoother probability distributions with Parzen window estimation, using a 2D gaussian kernel. Second, the shape context region is divided into bins using uniform angles *and* distances from the center of the histogram. Previously, it was found that features using equal rather than the conventional log-polar bin distances perform better when classifying spatial relationships in formulae [15]. The center of the PSC is the average of the two stroke bounding box centers. The radius of the shape context includes the strokes being compared. Note that for distant strokes, the polar histogram may cover the entire expression.

We use *three* separate Parzen window Shape Contexts when classifying directed LOS edges as 'merge' or 'split.' We use a separate PSC for each stroke, and then a third PSC for other strokes in the neighborhood of the two strokes. We do this to improve discrimination by clearly separating point sources. We confirmed empirically that using multiple histograms is beneficial. Figure 5 shows an example of Parzen Shape Context features for classifying a directed LOS edge. In this example, we consider an edge from the vertical stroke of '+' (the parent of the edge) to a nearby '1' stroke (the child of the edge). Red represents points from the parent stroke, green points from the child stroke, and blue points from other strokes in the histogram. Color intensity represents the density of each bin. Note that during segmentation, the reverse edge from the '1' to the vertical stroke of the '+' would also be considered.

PSC features produce a simplified image of the region around a pair of strokes, but with the point sources (parent, child, and other strokes) clearly separated. Polar histograms have higher resolution near their center, which we hoped would be beneficial. However, 2D histograms, convolution masks or other abstracted/compressed image representations might be used to similar or better effect.



Expression with PSC center and perimeter shown



Parent stroke Child stroke Other strokes

Figure 5: Example Parzen Shape Contexts (PSCs). A directed edge from the vertical line in '+' to the '1' is considered. Each PSC has 120 bins, with the PSC radius reaching the furthest parent or child stroke point (pixel). In experiments we use only 30 bins (5 distances $\times$ 6 angles), with a radius 1.5 times the distance to the furthest parent or child stroke point, capturing more context from other strokes

Geometric Features. We also use geometric features from previous work on classifying relationships between stroke pairs, including horizontal distance, size difference and vertical offset [16]; minimum point distance [12]; overlapping area [17]; minimum distance, horizontal overlapping of the bounding box, distance and offset between stroke start and end points, and finally backward movement and parallelity [18]. Parallelity is the angle between two vectors representing strokes, with the vectors defined by the first and last points of each stroke.

We also add some additional geometric features. These include the distance between bounding box centers, distance between centers-of-mass, maximal point pair distance (two points are from different strokes of the stroke pair), horizontal offset between the last point of the first stroke and the starting point of the second stroke, vertical distance between bounding box centers, writing slope (angle between the horizontal and the line connecting the last point of the current stroke and the first point of the next stroke) and writing curvature (angle between the lines defined by the first and last points of each stroke). We normalize all geometric features to lie in the interval $[0, 1]$ except for parallelity, writing slope and writing curvature.

Training. Using CROHME training data, We used greedy search and cross validation with random forest classifiers to determine the parameters of the PSC features [9]. We choose six angles and five distances for the polar histograms (30 bins), and a Parzen window width of $\frac{1}{5}$ of the shape context radius. The shape radius itself is 1.5 times the longest distance between stroke points to the center of the histogram. We also used the CROHME training data to create our random forest merge/split classifier [9]. We use a random forest with 50 trees, with maximum depth 40 for each decision tree. The Gini criterion was used for splitting. There are $n = 129$ features, and $\sqrt{n}$ features (11) are selected to define candidate splits at each decision tree node.

Experimental Results. The classification rate for merging or splitting stroke pairs is quite high: we obtain 98.26% for CROHME 2012, and 97.88% for the CROHME 2014. Table III shows our segmenter obtaining the second-highest reported symbol recall for CROHME 2012 (94.87%), and CROHME 2014 (92.41%), while obtaining the highest F-score for CROHME 2014 (92.43%).

Note that we obtain these results without using OCR or an expression grammar. Many of the systems shown are parser-driven, and use classification and relationship constraints (i.e., context) to refine segmentation. We believe that language constraints would improve our results substantially.

Table III: Symbol Segmentation Metrics for CROHME 2012 Test (only Recall reported [19]) and CROHME 2014 Test

CROHME 2012 Test (486 Expressions)

	Symbol		
	Recall (%)	**Precision (%)**	**F-score (%)**
MacLean et al. [20]	**95.56**		
LOS + PSC	94.87	94.56	94.72
Alvaro [21]	91.95		
Awal et al. [22]	87.75		
Hu et al. [23]	87.51		
Simistira et al.	71.21		
Celik et al. [24]	59.20		

CROHME 2014 Test (986 Expressions)

	Symbol		
	Recall (%)	**Precision (%)**	**F-score (%)**
Alvaro [25]	**93.31**	90.72	92.00
LOS + PSC	92.41	**92.45**	**92.43**
Awal et al. [26]	89.43	86.13	87.75
Yao and Wang [27]	88.23	84.20	86.17
Hu et al. [3]	85.52	86.09	85.80
Le et al. [28]	83.05	85.36	84.19
Aguilar [29]	76.63	80.28	78.41

LOS + PSC: Line-of-Sight Graph using Parzen Shape Contexts w. Random Forest Classifier

VI. Conclusion

We propose a Line-of-Sight (LOS) stroke graph that is able to represent more formulae than Time-series, Minimum Spanning Tree, Delaunay and k-NN graphs. For the CROHME 2012 and 2014 Test sets, LOS graphs omit fewer than 0.1% of necessary directed stroke pair edges, with a Precision of roughly 30%. We have used LOS graphs to

create a symbol segmenter making use of Parzen window-modified Shape Context features (PSC) that obtains state-of-the-art results for the CROHME 2014 Test set (92.43% F-measure) without using OCR or expression grammars. In other work, LOS graphs have been used to obtain surprisingly strong results for parsing handwritten formulae using primarily visual features [10].

Avenues for future work include exploring modified versions of LOS graphs (e.g., relaxing the notion of 'visibility' by allowing strokes to be partially transparent), exploring new graphs and combinations of graph types, incorporating classification and language constraints with our segmenter, and improving Parzen Shape Context features.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science foundation under Grant No. IIS-1016815. We thank Francisco Álvaro for providing code to convert CROHME stroke data to images.

REFERENCES

[1] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *IJDAR*, vol. 15, no. 4, pp. 331–357, 2012.

[2] H. Mouchère, R. Zanibbi, U. Garain, and C. Viard-Gaudin, "Advancing the state of the art for handwritten math recognition: the crohme competitions, 2011–2014," *IJDAR*, pp. 1–17, 2016.

[3] L. Hu and R. Zanibbi, "Segmenting handwritten math symbols using adaboost and multi-scale shape context features," in *Proc. ICDAR*, Aug. 2013, pp. 1212–1216.

[4] S. Smithies, K. Novins, and J. Arvo, "A handwriting-based equation editor," in *International Conference on Graphics Interface*, 1999, pp. 84–91.

[5] Y. Eto and M. Suzuki, "Mathematical formula recognition using virtual link network," in *Proc. ICDAR*, Sep. 2001, pp. 762–767.

[6] N. Matsakis, "Recognition of handwritten mathematical expressions," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1999.

[7] N. S. T. Hirata and W. Y. Honda, "Automatic labeling of handwritten mathematical symbols via expression matching," in *International Conference on Graph-based Representations in Pattern Recognition*, May 2011, pp. 295–304.

[8] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer-Verlag TELOS, 2008.

[9] L. Hu, "Features and algorithms for visual parsing of handwritten mathematical expressions," Ph.D. dissertation, Rochester Institute of Technology, 2016.

[10] L. Hu and R. Zanibbi, "MST-based visual parsing of online handwritten mathematical expressions," in *Proc. ICFHR*, 2016.

[11] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *TPAMI*, vol. 24, no. 4, pp. 509–522, 2002.

[12] K. Toyozumi, N. Yamada, T. Kitasaka, K. Mori, Y. Suenaga, K. Mase, and T. Takahashi, "A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information," in *Proc. ICPR*, Aug. 2004, pp. 630–633.

[13] A. Kosmala and G. Rigoll, "On-line handwritten formula recognition using statistical methods," in *Proc. ICPR*, Aug. 1998, pp. 1306–1308.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] F. Alvaro and R. Zanibbi, "A shape-based layout descriptor for classifying spatial relationships in handwritten math," in *ACM DocEng*, Sep. 2013, pp. 123–126.

[16] Y. Shi, H. Li, and F. Soong, "A unified framework for symbol segmentation and recognition of handwritten mathematical expressions," in *Proc. ICDAR*, Sep. 2007, pp. 854–858.

[17] S. MacLean and G. Labahn, "A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets," *IJDAR*, vol. 16, no. 2, pp. 139–163, 2013.

[18] S. Lehmberg, H.-J. Winkler, and M. Lang, "A soft-decision approach for symbol segmentation within handwritten mathematical expressions," in *International Conference on Acoustics, Speech, and Signal Processing*, May 1996, pp. 3434–3437.

[19] H. Mouchère, C. Viard-Gaudin, D. H. Kim, J. H. Kim, and U. Garain, "ICFHR 2012 competition on recognition of online mathematical expressions (CROHME 2012)," in *Proc. ICFHR*, Sep. 2012, pp. 811–816.

[20] S. MacLean and G. Labahn, "A bayesian model for recognizing handwritten mathematical expressions," *Pattern Recognition*, vol. 48, no. 8, pp. 2433–2445, 2015.

[21] F. Alvaro, J.-A. Sanchez, and J. Benedi, "Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars," in *Proc. ICDAR*, Sept. 2011, pp. 1225 –1229.

[22] A.-M. Awal, H. Mouchere, and C. Viard-Gaudin, "Towards handwritten mathematical expression recognition," in *Proc. ICDAR*, 2009, pp. 1046–1050.

[23] L. Hu, K. Hart, R. Pospesel, and R. Zanibbi, "Baseline extraction-driven parsing of handwritten mathematical expressions," in *Proc. ICPR*, Nov. 2012, pp. 326–330.

[24] M. Celik and B. Yanikoglu, "Probabilistic mathematical formula recognition using a 2d context-free graph grammar," in *Proc. ICDAR*, Sep. 2011, pp. 161–166.

[25] F. Alvaro, J. Sanchez, and J. Benedi, "Recognition of online handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models," *Pattern Recognition Letters*, vol. 35, pp. 58–67, 2014.

[26] A. Awal, H. Mouchère, and C. Viard-Gaudin, "A global learning approach for an online handwritten mathematical expression recognition system," *Pattern Recognition Letters*, vol. 35, pp. 68–77, 2014.

[27] H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. Garain, "ICFHR 2014 competition on recognition of on-line handwritten mathematical expressions (CROHME 2014)," in *Proc. ICFHR*, Sep. 2014, pp. 791–796.

[28] A. D. Le, T. V. Phan, and M. Nakagawa, "A system for recognizing online handwritten mathematical expressions and improvement of structure analysis," in *Proc. DAS*, Apr. 2014, pp. 51–55.

[29] F. JulcaAguilar, N. Hirata, C. ViardGaudin, H. Mouchere, and S. Medjkoune, "Mathematical symbol hypothesis recognition with rejection option," in *Proc. ICFHR*, Sep. 2014, pp. 500–505.