Tangent-V: Math Formula Image Search Using Line-of-Sight Graphs

Kenny Davila[1], Ritvik Joshi[2], Srirangaraj Setlur[1], Venu Govindaraju[1], and Richard Zanibbi[2]

[1] University at Buffalo, Buffalo NY 14260, USA
{kennydav,setlur,govind}@buffalo.edu
[2] Rochester Institute of Technology, Rochester NY 14623, USA
{rj9336,rlaz}@cs.rit.edu

Abstract. We present a visual search engine for graphics such as math, chemical diagrams, and figures. Graphics are represented using Line-of-Sight (LOS) graphs, with symbols connected only when they can 'see' each other along an unobstructed line. Symbol identities may be provided (e.g., in PDF) or taken from Optical Character Recognition applied to images. Graphics are indexed by pairs of symbols that 'see' each other using their labels, spatial displacement, and size ratio. Retrieval has two layers: the first matches query symbol pairs in an inverted index, while the second aligns candidates with the query and scores the resulting matches using the identity and relative position of symbols. For PDFs, we also introduce a new tool that quickly extracts characters and their locations. We have applied our model to the NTCIR-12 Wikipedia Formula Browsing Task, and found that the method can locate relevant matches without unification of symbols or using a math expression grammar. In the future, one might index LOS graphs for entire pages and search for text *and* graphics. Our source code has been made publicly available.

Keywords: graphics search · Mathematical Information Retrieval (MIR) · image search · PDF symbol extraction

1 Introduction

Modern search engines find relevant documents for text-based queries with high efficiency. However, not all information needs are satisfied by text. Most text search engines index graphical elements using textual metadata or ignore graphical elements altogether. To address this, retrieval systems have been created for specific graphic types, but they rely heavily upon notation-specific language models. At the same time, recently developed techniques have been used to extract tables, figures and other graphics automatically from large corpora (e.g., PDFFigures [11] for SemanticScholar[3]), presenting new opportunities for search within and across graphic types.

[3] https://www.semanticscholar.org

R·I·T

LAYOUT AND SEMANTICS: COMBINING REPRESENTATION FOR MATHEMATICAL FORMULA SEARCH

KENNY DAVILA, RICHARD ZANIBBI
COMPUTER SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY

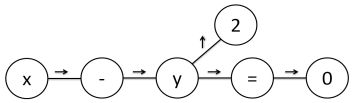
Overview

- The **Tangent-S** search engine finds math formulae using a combination of visual and semantic representations.
- We use a three-layer model with symbol pair indexing, structural alignment and re-ranking by combining multiple similarity scores
- Empirical tests show each layer increasing ranking quality as measured by Bpref and nDCG scores.

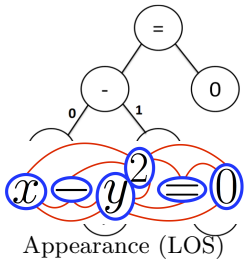
Formula Representation

$x - y^2 = 0$

(a) Expression



Visual Syntax (SLT)
(b) Symbol Layout Tree (SLT)



Appearance (LOS)
(C) Operator Tree (OPT)

Symbol Pair Generatio

Sym-1	Sym-2	Path	Co
V!x	O!−	→	
O!−	V!y	→	
V!y	N!2	↑	
V!y	U!=	→	
U!=	N!0	→	
V!x	V!y	→→	
O!−	N!2	→↑	
O!−	U!=	→→	
⋮	⋮	⋮	

(a) SLT Symbol Pairs

Fig. 1. Formula Structure Representations for $x - y^2 = 0$. State-of-the-art formula retrieval systems use Operator Tree (OPT) and Symbol Layout Tree (SLT) representations (e.g., MCAT [20] and Tangent-S [14]). Our visual search engine uses only domain-agnostic Line-of-Sight (LOS) graphs to represent structure.

Methodology

Layer 1

Candidate Selection: Uses bag of visual symbol pairs from given tree representation

Layer 2

Structural Verification: Finds the largest matching substructure between query and candidate to compute similarity scores

Candidate Selection

Top K candidates selected by Approximate Dice Coefficient of MatchingPairs [3] S_0:

$$S_0 = \frac{2|M|}{|Q| + |C|}$$

With Q and C as query and candidates pairs respectively, and M the set of matching pairs between Q and C.

Largest Structural Match

Finds the largest matching substructure between query and candidate trees [7], considering:

- Wildcards expansion with constrains
- Unification between symbols
- Commutativity for OPTs

We propose a visual graphics search engine, Tangent-V, that is applicable to vector images with known symbols (e.g., in PDF), and raster images with Optical Character Recognition (OCR) output giving recognized symbols locations and labels (e.g., for PNG images). Our method is based upon finding correspondences in Line-of-Sight graphs [7] that represent which symbols 'see' each other along an unobstructed line (see Figure 1). The language model requires only a set of symbols, allowing it to be applied to multiple graphic types such as math, chemical diagrams, and figures using a single index. In addition, our retrieval model supports wildcards that can be matched to any symbol.

Our main concern in this work is testing the viability of this purely visual approach, and comparing this method's behavior to that of notation-specific techniques. We benchmark our system using the NTCIR-12 Wikipedia Formula Browsing Task benchmark [30]. Despite the absence of explicit formula structure or a detailed language model, our approach achieves BPref results comparable to the state-of-the-art Tangent-S [14] search engine, for *both* PDF images (symbols known) and PNG images (symbols from OCR).

2 Background

Our search engine for graphics found in PDF and PNG images is a specialized form of Content-Based Image Retrieval (CBIR). Many CBIR approaches use a Bags-of-Visual Words (BoVW) framework [28], retrieving objects based on image features ('words'). Traditionally, visual words are defined by local image descriptors (e.g SIFT [22] or SURF [6]), and an inverted index of visual words in images is used for lookup. Later CBIR models use Deep Learning techniques [15] to learn local features [25] or even complete image representations such as hashes or embeddings [4,16,8,27].

For images containing notation (e.g., math), the spatial location of a symbol is important because it affects the structure and semantics of the graphic (see Figure 1). Some CBIR techniques consider spatial constraints, for example by locating candidates using spectral models, and then re-ranking the most

promising matches using spatial verification (e.g., using RANSAC [26]). Affine transformations [3,21,19] and elastic distortions [35] are also useful for spatial validation. Other models include spatial information during indexing [36].

To successfully index and retrieve images including notation, our approach combines topology (by indexing the relative locations of symbols) with spatial verification during the retrieval process. In the following, we summarize methods for graphic representation and search, along with recent methods designed specifically for mathematical information retrieval (our application).

Graphics Representation and Search. Graphics may be represented in three ways: *Semantics*, *Visual Syntax* and *Appearance*. Semantic representations encode the domain-specific information represented in a graphic. Figure 1 shows an Operator Tree (OPT) representing the operations and arguments in the expression $x - y^2 = 0$. As another example, table semantics may be represented by an indexing relation from category labels to values [29]. Often the same indexing relation can be represented in a table, plot, or bar graph. Visual Syntax provides a graphics-type-specific representation of visual structure. For example, in Figure 1 we see a Symbol Layout Tree (SLT) formula representation, giving the symbols on each writing line and the relative positions of writing lines. For tables, visual syntax can be defined using a two dimensional grid of cells [32]. For bar charts and scatter/line plots, visual syntax can be represented by the placement of axes, axis labels, bars/lines/points, ticks, and values [2,11,10,1]. Finally, appearance representations describe only objects/symbols and their relative positions. One example is the Line-of-Sight graph see Figure 1 [18,17]). We want search techniques applicable across graphic types, so we use LOS graphs to capture visual structure (see Figure 1).

Mathematical Information Retrieval (MIR). We apply our model to math formula retrieval, placing our work within the field of MIR [31]. Because math expressions are structured, traditional text-based search systems are inadequate for MIR [31]. We distinguish two math formula retrieval modalities: Image-based and Symbolic. For Image-based approaches, symbols and their relationships are initially unknown. Few methods have been proposed for MIR using images directly, and none have used standard benchmarks for evaluation. Zanibbi and Yu [34] used dynamic time warping over pixels projections to search for typeset formula images using handwritten queries. Chatbri et al. [9] use a connected component matching process to cast votes for candidate query matches in images. All of these methods avoid fully recognizing math expressions in images because it is challenging [23].

In contrast, for symbolic approaches, both the symbols and structure are known (e.g., from LaTeX or MathML). The math retrieval tasks at the NTCIR conferences [30] have produced improved symbolic MIR systems. Among other systems, this includes the MCAT [20] and Tangent-S [14] formula search engines that we use for benchmarking in our experiments. Both systems make use of Visual Syntax (SLT) and Semantic (OPT) representations for search, and retrieve formulas using paths in SLTs and OPTs, followed by finer-grained structural analysis and re-ranking.

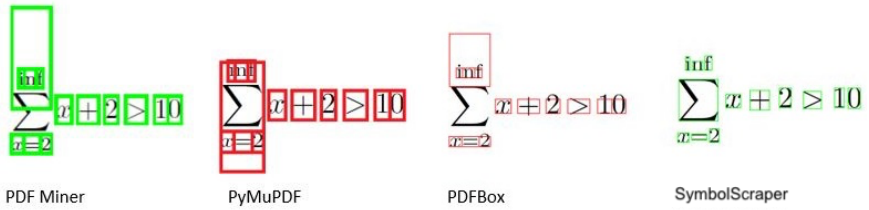


Fig. 2. Symbol Bounding Box Extraction Comparison. SymbolScraper captures the exact location of all symbols. Other tools add or omit character ascender and descender regions, and mislocate large operators (e.g., $\sum$).

Tangent-V [12] generalizes Tangent-S [14], which performs retrieval using symbol pairs in SLTs and OPTs. Tangent-V uses symbol pairs taken directly from images: for PDFs, using symbols extracted directly from the file, and for PNGs using symbols identified with our open-source OCR system [13]. Previously Tangent-V was successfully applied to retrieval of specific handwritten formulas in videos using LaTeX queries [12]. In this paper, we observe the effectiveness of Tangent-V for more general search within isolated formula images.

3 Extracting Symbols from PDF Documents

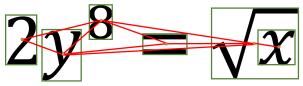
We use an extension of the Apache PDFBox Java library to extract symbol locations and codes from *born-digital* PDF files (e.g., created using Word or LaTeX). Available tools for extracting symbols provide imprecise locations (see Figure 2), or require image processing and/or OCR [5]. Our SymbolScraper tool is open source, and available for download.[4]

In PDF, each character has a vector representation containing a character code, font attributes, and writing line position. However, specific symbol locations must be inferred from font attributes. We identify bounding boxes around symbols using font metrics and the character outlines (*glyphs*) embedded in PDF files. Glyphs are defined by a sequence of line segments, arcs, and lifts/moves of the 'pen' used to draw character outlines. Most characters have a single outline, however some symbols such as parentheses may be drawn using multiple glyphs to support smooth rendering at different scales. To capture these we assume that intersecting character outlines belong to a single symbol.

4 Line-Of-Sight Graphs

To capture spatial relationships between neighboring symbols, we identify symbols and then construct an LOS graph (see Figure 3(a)). After constructing an

[4] SymbolScraper: `https://www.cs.rit.edu/~dprl/Software.html`



(a) LOS graph

(b) OCR results

Actual Symbol	OCR Label	OCR Prob
2	2	0.98
y	y	0.95
8	8	0.70
	&	0.25
=	=	0.99
$\surd$	$\surd$	0.96
x	x	0.97

(c) Inverted Index Entries over Symbol Label Pairs

Labels u	Labels v	Pair ID	Probs. u	Probs. v	3D Disp. dx	3D Disp. dy	3D Disp. dz	Size Ratio	Order
2	y	1	0.98	0.95	0.93	0.36	0.00	0.90	1
2	8	2	0.98	0.70	0.98	-0.22	0.00	1.41	1
&	2	2	0.98	0.25	-0.98	0.22	0.00	0.71	-1
8	y	3	0.70	0.95	-0.77	0.64	0.00	0.64	1
&	y	3	0.25	0.95	-0.77	0.64	0.00	0.64	1
=	y	4	0.99	0.95	-0.99	0.11	0.00	0.76	1
$\surd$	y	5	0.96	0.95	-0.99	0.06	0.00	1.74	1
=	8	6	0.99	0.70	-0.95	-0.32	0.00	1.18	1
&	=	6	0.25	0.99	0.95	0.32	0.00	0.85	-1
$\surd$	8	7	0.96	0.70	-0.99	-0.13	0.00	2.71	1
&	$\surd$	7	0.25	0.96	0.99	0.13	0.00	0.37	-1
=	$\surd$	8	0.99	0.96	1.00	0.00	0.00	0.44	1
$\surd$	x	9	0.96	0.97	0.39	0.08	0.92	2.21	1

Fig. 3. Indexing an image-based LOS graph (a) using OCR results (b). The inverted index entries in (c) map lexicographically sorted symbol label pairs to LOS graph edges with their attributes. All LOS edges with the '8' require two entries, to capture the two OCR label outputs for the symbol ('8' and '&').

LOS graph, we use the graph edges to construct an inverted index over symbol pairs for search.

Symbol Nodes. Each node in an LOS graph represents a candidate symbol with its location and set of labels with confidences. For born-digital PDFs, we use SymbolScraper to obtain these directly from the file (see Section 3).

For binary images, we use connected black pixel regions (connected components, or CCs) as symbol candidates, and run our open-source OCR system [13] trained on 91 mathematical symbol classes (e.g. digits, operators, latin and greek letters, etc.) to obtain the most likely symbol labels. To better capture symbols comprised of multiple CCs such as 'i' and 'j,' we try merging each CC with its two closest neighboring CCs. If one of these merged symbols has a higher classification confidence than the average of the top label confidences for each individual CC, the merged symbol is kept. Note that our OCR model does not recognize all symbols found in our test collection. However, since the images are typeset, a reasonably consistent label assignment is expected for symbols belonging to the same class, allowing the proposed model to describe them well using multiple labels, even if the specific symbol is unknown to the OCR system.

LOS Edges. Once the symbols (nodes) of the LOS graph are defined, two symbols (nodes) are connected if they can "see" each other. We test visibility by drawing lines from the bounding box center of each symbol to the vertices of the convex hull of the other symbol. If one of these lines does not intersect a third symbol's convex hull, then there is a line of sight between the symbols. Starting with a fully connected graph, the LOS graph is generated by pruning edges

between symbols that fail the visibility test.[5] Some graphs include large empty regions, allowing distant elements to see each other, producing a very dense graph. To avoid this, we prune LOS edges more than twice the median symbol distance apart. This substantially reduces both the index size and retrieval times.

5 Indexing Line-of-Sight Graphs

Using LOS graphs edges, we create an inverted index from symbol pairs to LOS edges connected to symbols of the given type (see Figure 3). Figure 3 shows the LOS graph for $2y^8 = \sqrt{x}$, along with OCR symbol confidences and entries for the inverted index.

Symbol Probabilities. For PDF input, symbols are known, and the probability of each label is 1.0. For image input, OCR produces a list of class probabilities for each symbol in decreasing order. The top-n classes are selected until a cumulative probability of at least 80% is obtained, or n class labels have been selected ($1 \leq n \leq 3$). In Figure 3(b), the top class for all symbols has a probability larger than 80% except for the symbol "8." For the "8," we also include the second-highest probability label "&," at which point the cumulative probability is greater than our threshold.

Graph Edge Identifiers. Graph edges have unique global identifiers. When symbols have multiple labels from OCR, their associated LOS edges are entered in the index using pairs of candidate labels. Figure 3(c) shows all 13 index entries for the 9 LOS graph edges. Edge #2 has two entries, in the postings for $(2, 8)$ and $(\&, 2)$. Symbol label pairs for keys are sorted in lexicographic order (i.e., by unicode value). The LOS edge identifiers allow postings for symbols with multiple OCR hypotheses to be merged during retrieval.

Displacement Vectors and Label Order. The relative position of symbol centers are represented using a 3D unit vector $\langle d_x, d_y, d_z \rangle$. The third dimension is non-zero when a symbol center lies within the bounding box of the other (see Figure 4). We fit an enclosing sphere around the bounding boxes of each symbol, and define r_{max} as the larger radius for the two symbols. If symbol centers are at a distance smaller than r_{max}, d_z is computed as:

$$d_z = \sqrt{(r_{max})^2 - \left(\sqrt{d_x^2 + d_y^2}\right)^2} \tag{1}$$

or $d_z = 0$ otherwise. Displacement vectors are normalized and indexed as a unit vector along with their *label order*. The label order indicates whether a given symbol pair is consistent with the direction of the displacement vector (1) or if the displacement vector has been inverted (-1). In Figure 3(c), LOS edge #2 $(2, 8)$ uses an inverted ordering for the combination of labels $(\&, 2)$.

Size Ratios (s_p). We also index the ratio of symbol sizes for an LOS edge. At retrieval time, we prune edge matches with large differences in symbol size ratios. For symbols u and v, $s_p(u, v)$ is the length of the bounding box diagonal

[5] faster algorithms may be used [7].

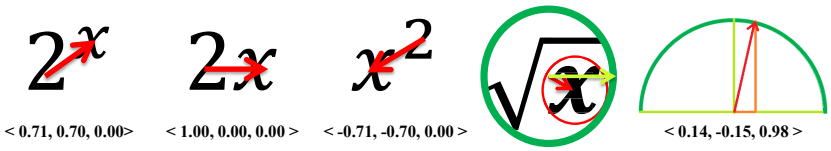


Fig. 4. 3D Unit Vectors Between Symbol Centers. Two directions are enough to represent relative positions for 2^x, $2x$ and x^2. However, the bounding boxes of $\surd$ and x are overlapping in $\sqrt{x}$, so the center of x is projected onto a sphere around $\surd$.

for u divided by the bounding box diagonal length for v. A posting with label order -1 indicates that size ratio has been inverted relative to order of symbols associated with an entry (e.g., for entry $(\&, 2)$ in Figure 3).

Isolated Symbols. To index isolated symbols, we introduce same-symbol pairs using self-edges for symbols in LOS graphs with three or fewer symbols. This allows queries such as x to match small graphs (e.g., x^2). Single symbol 'pairs' have displacement vector $< 0, 0, 1 >$, size ratio 1, and label order 1.

6 Retrieval

Our system uses a two-layer retrieval model. The first layer (the *core engine*) finds all graphs with LOS edges matching the query. Matched graphs are ranked using an edge-based metric, after which the top-k ($k = 1000$) candidates are passed to the second layer (*re-ranking*), which revises scores using an alignment algorithm.

Notation. We define Ω_x as the set of possible symbol identities for LOS graph node x. Given a candidate graph (M), a matched LOS edge for query and candidate symbol pairs (q_1, q_2) and (c_1, c_2) is represented by (Q, C), where $Q = ((\Omega_{q_1}, q_1), (\Omega_{q_2}, q_2))$ and $C = ((\Omega_{c_1}, c_1), (\Omega_{c_2}, c_2))$. The corresponding displacement vectors between symbol pairs on candidate and query edges are unit vectors $\boldsymbol{q}$ and $\boldsymbol{c}$. The conditional probability of symbol class ω given visual features for query symbol q_1 is denoted by $p(\omega|q_1)$.

6.1 Layer 1: Core Engine

LOS edges matching the query are retrieved from the inverted index using pairs of lexicographically sorted symbols. For example, after applying OCR to 'x^2' we obtain one LOS edge with class label lists $\Omega_u = \langle 2 \rangle$ for the '2' and $\Omega_v = \langle x, X \rangle$ for the 'x'. We lookup postings for both $(2, x)$ and $(2, X)$ in the index, merging postings for edges that appear in both posting lists. We support matching wildcards in queries, which are mapped to single symbols on candidates. This is limited compared to domain-specific MIR retrieval models which can match wildcards to sub-graphs [30]. Given a query pair containing a wildcard, we retrieve all index entries satisfying the given pattern. For example, the pair $(X, 2)$

will match all index entries containing a 2 (e.g. $(1,2)$, $(+,2)$, $(2,x)$, etc). Edges containing two wildcards are ignored (these match all index entries).

Retrieved edges in posting lists are filtered, removing candidate edges with large differences in displacement angles and/or symbol size ratios relative to the query edge. First, candidate edge displacement vectors ($\boldsymbol{c}$) with an angular difference of greater than $\pm 30°$ relative to the query ($\boldsymbol{q}$) are removed. Candidate edge C is also filtered if its symbol size ratio is less than half, or more than twice the ratio for the corresponding query edge, given by $s_r(Q,C) < 0.5$, where $s_r(Q,C)$ is:

$$s_r(Q,C) = \frac{min\left(s_p\left(q_1,q_2\right), s_p\left(c_1,c_2\right)\right)}{max\left(s_p\left(q_1,q_2\right), s_p\left(c_1,c_2\right)\right)} \tag{2}$$

The initial edge-based ranking metric is an edge recall, weighted by symbol confidences and differences in displacement vectors. Our edge-based scoring function $S(M)$ for matched LOS subgraph M adds the product of symbol confidences and angular differences, summing over common symbol classes for matched symbols. For a given symbol class ω, we combine the probabilities for that class in corresponding query/candidate symbols p and c using their minimum probability: $f(\omega,q,c) = min\left(\ p(\omega|q),\ p(\omega|c)\ \right)$. We found this produces more stable results than using the product of the probabilities [12]. For a wildcard pair $Q_{\mathrm{w}} = (\{X\},q_{\mathrm{w}}),(\{\omega_j\},q_2))$, matching a concrete pair $C = ((\{\omega_i\},c_1),(\{\omega_j\},c_2))$, we set $p(\omega_{\mathrm{w}}|q_{\mathrm{w}}) = p(\omega_j|q_2)$. This forces our model to prefer wildcard matches only when attached to strong concrete symbol matches.

$$s_\Omega(Q,C) = \sum_{\substack{\omega_i \in \Omega_{q_1} \cap \Omega_{c_1} \\ \omega_j \in \Omega_{q_2} \cap \Omega_{c_2}}} f(\omega_i,q_1,c_1)\ f(\omega_j,q_2,c_2) \tag{3}$$

$$s_\angle(Q,C,\theta) = \begin{cases} \frac{\boldsymbol{q}\cdot\boldsymbol{c}-cos(\theta)}{1-cos(\theta)}, & \text{if } \boldsymbol{q}\cdot\boldsymbol{c} \geq cos(\theta) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$S(M) = \sum_{(Q,C)\in M} s_\Omega(Q,C)\ s_\angle(Q,C,30°) \tag{5}$$

We keep only the top-k ($k = 1000$) matched graphs after computing an optimistic greedy estimation of the maximum $S(M)$ score candidate graphs. For each graph, matched edges are added to $S(M)$ in decreasing order of weighted recall score ($s_\Omega(Q,C)\ s_\angle(Q,C,30°)$ in Eq. 5), while enforcing a 1-to-1 matching constraint between query and candidate edges.

6.2 Layer 2: Re-ranking

For each candidate graph selected by the core engine, connected components from matched edges are identified. To ensure that the connected components match query graph LOS structure, we require components to preserve a one-to-one mapping from candidate to query symbols (nodes). The first row in Figure

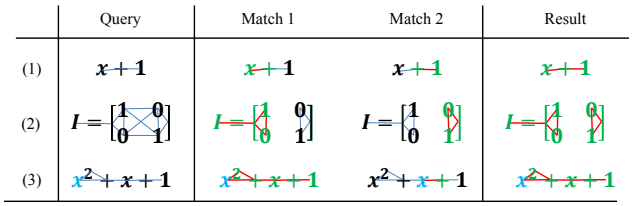


Fig. 5. Structural Alignment Steps. Matching nodes and edges are shown in green/blue and red respectively. (1) Match growing, two connected pairs matching different portions of $x + 1$ are merged into a single larger match. (2) Joining disconnected subgraphs, two partial matches on disconnected subgraphs which are spatially consistent are merged into a single larger match. (3) Incompatible match removal, keeping only the best match.

5 shows two matched candidate LOS edges being merged ("$x+$" and "$+1$") to form a connected component ("$x+1$"). After growing all connected components, the top-M ($M = 50$) components are selected for further processing, each scored using the $S(M)$ metric.

Queries often match disjoint LOS subgraphs. This occurs due to some combination of OCR errors, unmatched symbols, or pruned LOS edges. To connect subgraphs into larger matches, we greedily merge disjoint matches that preserve a one-to-one query/candidate symbol mapping and have a very low *spatial distortion cost* after merging. Using the highest scoring match as the reference match, we compute an affine transformation matrix that will translate and scale the candidate nodes into the query space. Based on the reference match, the center of the bounding box of its candidate subgraph is translated to the center of the bounding box of its query subgraph. Then, the diagonals of the same bounding boxes are used to define a scaling factor, used to re-scale candidate nodes into the query space scale. We then use this transformation matrix to project candidate nodes from the second match into the query space. Representing symbol bounding boxes as 4D vectors (x_1, y_1, x_2, y_2), we compute the average of all euclidean distances between the bounding box of each candidate node from the second match and the bounding box of their corresponding query nodes. Finally, we normalize this average euclidean distance by the average diagonal length of query node bounding boxes, and we use this as our *spatial distortion cost*. If this cost exceeds a threshold ($max_{dist} = 0.5$), matches will not be joined. An example is shown in the middle row of Figure 5, where two components of a matrix are disconnected, but then merged.

This procedure may produce multiple candidate matches. We again apply greedy filtering, selecting the next largest match that does not contain previously selected candidate nodes. In our current evaluation, only one match is allowed per candidate graph; therefore we keep only the highest scoring match. Candidate graphs are sorted by their final scores, with ties broken by sorting by increasing number of unmatched edges.

Table 1. Statistics for different index conditions for the NTCIR-12 MathIR Wikipedia Collection.

	PNG			PDF
Property	**Top-1**	**Top-2**	**Top-3**	**All**
Index Entries	3,923	4,147	4,186	36,593
Graph Edges	10,462,843	10,462,843	10,462,843	9,591,932
Pair Instances	10,462,843	33,532,368	60,232,395	9,728,374
Size on Disk (GB)	2.61	3.09	3.63	1.57
Query Times (seconds)				
Core - Avg (Std)	6.59 (4.79)	10.66 (7.35)	15.77 (12.18)	4.39 (4.09)
Full - Avg (Std)	9.93 (7.40)	14.99 (10.87)	21.84 (19.20)	6.36 (5.47)

7 Evaluation

Benchmark. For evaluation, we use the NTCIR-12 MathIR Wikipedia Formula Browsing Task [30]. The collection contains 591,608 instances of approximately 328,685 unique formulas taken from English Wikipedia. The NTCIR-12 query set has 40 topics, with 20 containing wildcards. During the competition, the top-20 hits from participating systems were pooled, with each scored by two human assessors (university students). Assessors rated hits using 0, 1, or 2 to indicate whether a hit is irrelevant, partially relevant, or relevant. The two assessor scores are then added. 'Fully Relevant' hits are those with a combined assessor score ≥ 3, while 'Partially Relevant' hits are those with a combined score ≥ 1.

Indexing and Retrieval. Using LaTeX, we render each formula in PDF and PNG formats, and then create an index for each. For PNGs, we trained our symbol classifier using classes in the CROHME 2016 dataset [24] with LaTeX-generated synthetic data: the 101 classes were grouped based on similar shapes into 91 classes. For PDFs, we used the extraction tool described in Section 3 to obtain precise symbol locations and classes. For PNG, we constructed three indices for when at most 1, 2, or 3 class labels are permitted per symbol (see Section 5). Metrics for the indices are provided in Table 1.

Both Precision@K and BPref are computed using the official competition relevance judgments for this task, and the `trec_eval` tool[6] (see Tables 2 and 3). Our experimental system had an Intel processor i7-7820X with 64 GB of RAM, and a Nvidia GTX 1080 GPU. Most operations run on a single thread except for vector operations in long posting lists executed on the GPU. Mean query execution times for all indexing and retrieval conditions are shown in Table 1.

Discussion. As one expects, the LOS-based Precision@K values are lower than those obtained by domain-specific state-of-the-art methods for formula retrieval; but this is partly because many formulas without judgments in the top-20 and treated as irrelevant. However, for BPref scores the LOS approach produces more comparable results based on human pairwise preferences. In fact, the LOS model on PDFs achieves slightly better BPref values for Partially Relevant hits for queries without wildcards than domain-specific retrieval models such as Tangent-S [14].

[6] `http://trec.nist.gov/trec_eval`

Table 2. Average Precision@K values per topic for NTCIR-12 MathIR Wikipedia Formula Browsing Task.

		Relevant (%)			Partially Relevant (%)		
		P@5	P@10	P@20	P@5	P@10	P@20
MCAT [20]		**49.00**	**39.00**	**28.25**	**91.00**	**84.00**	**76.87**
Tangent-S [14]		44.00	31.50	21.62	70.00	60.75	51.12
LOS PDF							
All	Core	23.00	18.00	13.00	41.00	33.75	27.13
	Reranked	29.50	22.25	17.37	41.50	38.00	32.37
LOS PNG							
Top-1	Core	23.00	17.25	12.13	41.50	33.75	25.50
Top-2	Core	20.50	16.25	13.00	40.50	33.50	27.37
Top-3	Core	19.50	14.50	11.37	38.00	30.25	24.37
Top-1	Reranked	26.00	19.00	14.50	46.00	37.00	30.50
Top-2	Reranked	27.00	19.75	15.62	47.50	38.25	32.50
Top-3	Reranked	27.50	20.25	16.12	47.00	38.50	33.25

Table 3. Average BPref values per topic for NTCIR-12 MathIR Wikipedia Formula Browsing Task. Results shown for all queries (40) & queries without/with wildcards (20/20).

		Relevant (%)			Partially Relevant (%)		
		All	Concr.	Wild.	*All*	Concr.	Wild.
MCAT [20]		52.02	57.02	**47.02**	53.56	56.98	50.13
Tangent-S [14]		**55.30**	**63.61**	46.99	56.20	58.72	**53.68**
LOS PDF							
All	Core	39.17	48.30	30.04	55.13	60.00	50.26
	Reranked	53.05	59.85	46.26	**56.44**	60.32	52.57
LOS PNG							
Top-1	Core	36.78	49.14	24.42	46.32	55.99	36.64
Top-2	Core	42.05	50.67	33.43	50.97	59.26	42.68
Top-3	Core	40.53	50.33	30.73	51.88	58.05	45.71
Top-1	Reranked	46.04	55.93	36.15	47.51	57.01	38.00
Top-2	Reranked	49.74	58.96	40.52	52.43	**60.86**	44.00
Top-3	Reranked	50.75	59.20	42.30	53.52	59.72	47.31

As expected, PDF results are almost always better than PNG results. We consider PDFs as the better condition for our model, since they have a more accurate label assignment, producing fewer index entries (see Table 1). This means that more unique combinations of symbols pairs are being considered, with shorter postings lists overall. On the other hand, too specific labels for some variations of known symbols (e.g x vs $\hat{x}$) may prevent the system from ranking partial matches properly.

In contrast, PNG results are degraded by noise. Considering only 91 unique symbol shapes can cause problems for out-of-vocabulary symbols. This results in a smaller number of index entries with longer posting lists. Adding extra labels for each symbol causes the index to quickly multiply in size, and produces slower retrieval times (see Table 1). However, we can obtain slight improvements for both Precision@K and BPref for re-ranked results when these extra labels are indexed. This is a trade-off between retrieval time and rank quality, which may worthwhile for applications where higher recall is more important than speed.

The initial core results can be retrieved in shorter times compared to the full model (see Table 1). However, re-ranking helps in almost all conditions, and Pre-

cision@K is always increased after re-ranking. In comparison, the MCAT system takes several minutes on average when unification is used [20]. The Tangent-S system is implemented with several core engine optimizations making it faster (avg of 2.67 s) than our core engine, but it has slower re-ranking times [33,14] with greater variance in execution times than our proposed re-ranking.

We implemented our model using Python. All queries and retrieval conditions were computed using a single process except for GPU-accelerated vector operations. MCAT systems uses 50 processors for variable unification [20]. Our current prototype tests the effectiveness of the retrieval model, and future work includes various low level optimizations that will increase efficiency like the ones used in the pair-based engine of Tangent-S [33]. Overall, our model finds many relevant formulas despite a lack of domain-specific knowledge. We expect our LOS appearance-based model will also provide meaningful results for other graphic types, with little need for domain-specific fine tuning.

8 Conclusion

We have presented our Tangent-V model for visual graphics search, along with its application to retrieving mathematical formulas. Our model considers only symbol labels and their relative positions, without any facility for unification of numbers, identifiers, or variable names. Despite this simple approach, our model finds relevant results, and outperforms existing domain-specific formula search engines in terms of BPref for partially relevant matches. This confirms that appearance alone can provide meaningful formula search results, and we are interested in seeing how this generalizes to other notations (e.g., chemical diagrams and figures). We are also interested in replacing OCR in raster images (e.g., in PNG) with visual feature-based descriptors.

Previously our model was successfully applied to cross-modal search, by matching handwritten versions of formulas taken from course notes in LaTeX [12]. This work confirms that our approach is promising for not just locating specific formulas, but formulas similar to a query.

In the future, we want to explore support for unification of symbols, and modify our scoring metrics to consider the context of a match, preferring identical matches to those surrounded by extra symbols. Finally, our implementation can be optimized in a number of ways, including re-implementing the Python prototype in C/C++, and structuring posting lists to reduce the number of candidate matches considered. Source code for Tangent-V is publicly available.[7]

9 Acknowledgements

We are grateful to Chris Bondy for his help with designing SymbolScraper. This material is based upon work supported by the National Science Foundation (USA) under Grant Nos. HCC-1218801, III-1717997, and 1640867 (OAC/DMR).

[7] `https://cs.rit.edu/~dprl/Software.html#tangent-v`

References

1. Al-Zaidy, R.A., Giles, C.L.: Automatic extraction of data from bar charts. In: Proceedings of the 8th International Conference on Knowledge Capture, K-CAP 2015, Palisades, NY, USA, October 7-10, 2015. pp. 30:1–30:4 (2015). https://doi.org/10.1145/2815833.2816956, http://doi.acm.org/10.1145/2815833.2816956
2. Al-Zaidy, R.A., Giles, C.L.: A machine learning approach for semantic structuring of scientific charts in scholarly documents. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. pp. 4644–4649 (2017), http://aaai.org/ocs/index.php/IAAI/IAAI17/paper/view/14275
3. Avrithis, Y., Tolias, G.: Hough pyramid matching: Speeded-up geometry re-ranking for large scale image retrieval. International Journal of Computer Vision **107**(1), 1–19 (2014)
4. Babenko, A., Lempitsky, V.: Aggregating local deep features for image retrieval. In: Proceedings of the IEEE international conference on computer vision. pp. 1269–1277 (2015)
5. Baker, J., Sexton, A.P., Sorge, V.: Extracting precise data on the mathematical content of pdf documents. In: Towards a Digital Mathematics Library (DML). Masaryk University Press, Birmingham, UK (July 27 2008), iSBN 978-80-210-4658-0
6. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: Computer vision–ECCV 2006, pp. 404–417. Springer (2006)
7. Berg, M., Cheong, O., Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer-Verlag, 3rd edn. (2008)
8. Cao, Y., Long, M., Liu, B., Wang, J.: Deep cauchy hashing for hamming space retrieval. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
9. Chatbri, H., Kwan, P., Kameyama, K.: An application-independent and segmentation-free approach for spotting queries in document images. In: ICPR. pp. 2891–2896. IEEE (2014)
10. Choudhury, S., Mitra, P., Kirk, A., Szep, S., Pellegrino, D., Jones, S., Giles, C.L.: Figure metadata extraction from digital documents. In: 12th International Conference on Document Analysis and Recognition. pp. 135–139. ICDAR '13 (2013). https://doi.org/10.1109/ICDAR.2013.34
11. Clark, C., Divvala, S.K.: Pdffigures 2.0: Mining figures from research papers. In: Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries, JCDL 2016, Newark, NJ, USA, June 19 - 23, 2016. pp. 143–152 (2016). https://doi.org/10.1145/2910896.2910904, http://doi.acm.org/10.1145/2910896.2910904
12. Davila, K., Zanibbi, R.: Visual search engine for handwritten and typeset math in lecture videos and latex notes. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 50–55 (Aug 2018). https://doi.org/10.1109/ICFHR-2018.2018.00018
13. Davila, K., Ludi, S., Zanibbi, R.: Using off-line features and synthetic data for on-line handwritten math symbol recognition. In: ICFHR. pp. 323–328. IEEE (2014)
14. Davila, K., Zanibbi, R.: Layout and semantics: Combining representations for mathematical formula search. SIGIR (2017)

15. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
16. Gordo, A., Almazán, J., Revaud, J., Larlus, D.: Deep image retrieval: Learning global representations for image search. In: European Conference on Computer Vision. pp. 241–257. Springer (2016)
17. Hu, L., Zanibbi, R.: MST-based visual parsing of online handwritten mathematical expressions. In: Proc. Int'l Conf. Frontiers in Handwriting Recognition (ICFHR). Shenzhen, China (2016), (to appear)
18. Hu, L., Zanibbi, R.: Line-of-sight stroke graphs and parzen shape context features for handwritten math formula representation and symbol segmentation. In: ICFHR. pp. 180–186. IEEE (2016)
19. Jégou, H., Douze, M., Schmid, C.: Improving bag-of-features for large scale image search. International Journal of Computer Vision **87**(3), 316–336 (2010)
20. Kristianto, G.Y., Topić, G., Aizawa, A.: The MCAT math retrieval system for NTCIR-12 MathIR task. In: Proc. NTCIR-12. pp. 323–330 (2016)
21. Li, X., Larson, M., Hanjalic, A.: Pairwise geometric matching for large-scale object retrieval. In: CVPR. pp. 5153–5161 (June 2015)
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
23. Mouchère, H., Zanibbi, R., Garain, U., Viard-Gaudin, C.: Advancing the state-of-the-art for handwritten math recognition: The CROHME competitions, 2011-2014. Int'l J. Document Analysis and Recognition (IJDAR) **19**(2), 173–189 (2016)
24. Mouchère, H., Viard-Gaudin, C., Zanibbi, R., Garain, U.: ICFHR 2016 CROHME: Competition on recognition of online handwritten mathematical expressions. In: International Conference on Frontiers in Handwriting Recognition (ICFHR) (2016)
25. Noh, H., Araujo, A., Sim, J., Weyand, T., Han, B.: Largescale image retrieval with attentive deep local features. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3456–3465 (2017)
26. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. pp. 1–8. IEEE (2007)
27. Radenovi, F., Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Revisiting oxford and paris: Large-scale image retrieval benchmarking. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
28. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV. pp. 1470–1477. IEEE (2003)
29. Wang, X.: Tabular Abstraction, Editing and Formatting. Ph.D. thesis, University of Waterloo, Canada (1996)
30. Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topić, G., Davila, K.: NTCIR-12 MathIR Task Overview. In: Proc. NTCIR-12. pp. 299–308 (2016)
31. Zanibbi, R., Blostein, D.: Recognition and retrieval of mathematical expressions. IJDAR **15**(4), 331–357 (2012)
32. Zanibbi, R., Blostein, D., Cordy, J.R.: A survey of table recognition: models, observations, transformations, and inferences. Int'l J. Document Analysis and Recognition (IJDAR) **7**(1), 1–16 (2004)
33. Zanibbi, R., Davila, K., Kane, A., Tompa, F.: Multi-stage math formula search: Using appearance-based similarity metrics at scale. SIGIR (2016)
34. Zanibbi, R., Yu, L.: Math spotting: Retrieving math in technical documents using handwritten query images. In: ICDAR. pp. 446–451. IEEE (2011)
35. Zhang, W., Ngo, C.W.: Topological spatial verification for instance search. IEEE Transactions on Multimedia **17**(8), 1236–1247 (Aug 2015). https://doi.org/10.1109/TMM.2015.2440997

36. Zhang, Y., Jia, Z., Chen, T.: Image retrieval with geometry-preserving visual phrases. In: CVPR. pp. 809–816. IEEE (2011)