University of Glasgow | School of Computing Science

# Graph Models and Maximum Common Subgraph for Character Analysis

Kyle A. Simpson (2029567)

May 12, 2017

## ABSTRACT

*Modern computer vision research depends upon complex vector-space models and machine learning techniques, which can be hard for humans to make sense of. If suitable graph models exist for these problems, then we benefit triply: problem instances become more intuitive, it becomes easier to assess model robustness, and we may use standard graph search and similarity algorithms for classification with little modification. I show the ineffectiveness of current general image graph models against these known similarity metrics, and propose two new models for glyph image modelling. These models are found to underperform compared to the state-of-the-art and computing exact graph similarity is found to be a poor fit for classification tasks, yet the intuitive nature of the models is shown to both allow and inform easy model enhancement.*

## 1. INTRODUCTION

Research into computer vision tasks, such as image and object recognition, is currently dominated by the use of machine learning and vector-space models, while graph encodings are comparatively less well-explored. Machine learning methods typically rely on these vector-space models, applying statistical inference from training data to teach classifiers how to recognise the desired elements or features from an image—and have become popular due to their versatility, effectiveness and accuracy across many problem domains. Some of these vector-space models take a *keypoint* approach, capturing the area around high-contrast regions in images as salient features, or might examine *codings* which describe an image's contents in a dense and very low-level fashion.

In many cases it can be hard to reason about such models' robustness or sensitivity to different phenomena. In a machine learning context, for instance, these are often a function of both the training data and the model itself. The parameters learned by these models aren't structured in a way that allows humans to easily intuit what the model has learned or to comment on the system's correctness, and it can be difficult to discern *why* a particular image might be misclassified. Recent work into *adversarial images* [12] has confirmed this, given that interference can lead to classifications contrary to image content.

Graph models, long-explored in discrete mathematics, provide a simpler way of visualising problems. Systems are broken down into *vertices* and *edges* between vertex pairs—capturing the relationships between objects or key features within a scene. For many domains, this is an intuitive and effective representation, encoding rich semantics in a very natural way which enjoys use in computational chemistry

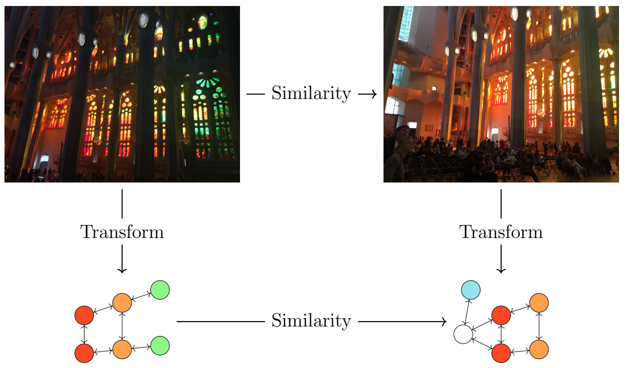


**Figure 1:** ***An example of the expected similarity between two views of the same scene. Similar objects and scenes should theoretically generate similar graphs after a suitable transformation, enabling recognition across images.***

[9, 11, 21], biology [13], and graph database analysis. Furthermore, exact graph search and similarity metrics (e.g. the *subgraph isomorphism* and *maximum common subgraph* problems, respectively) are well-understood and an area of continual research in the field of algorithms.

The main aim of this paper is to explore the hypothesis that, after suitable transformation, image similarity corresponds to graph similarity according to these known metrics; fig. 1 gives a rough illustration of the concept. In particular, this work focuses on character analysis and recognition, considering both the performance of graph models within a domain alongside the usefulness of any chosen analytic techniques. By investigating these questions, this paper contributes:

- A demonstration of the shortcomings of existing image graph work for generic matching (Section 3), by attempting similarity computation on graphs built using common techniques from the literature. This weakness motivates the development of new modelling strategies.

- An algorithm for converting binary images into graphs, designed for use with images containing a clear path structure, such as character glyphs (Section 4). Path curvature is the core feature for matching.

- A matching algorithm for these graphs, produced by modifying the $k\downarrow$ [14] procedure (Section 5). This modified algorithm is used to compute graph similarity as part of a $k$-Nearest Neighbours classifier.

- An alternative graph model, capturing angle path dynamics using a “dual graph”-like transformation (Section 4.2). This model arises in response to a perceived weakness in the first algorithm; this discovery is aided by the intuitive structure of the original graph model.

- An experimental evaluation of the matching accuracy of these models against datasets of handwritten and machine-generated character glyphs (Section 6).

## 2. GRAPHS, SEARCH AND SIMILARITY

First, we must define graphs more precisely. An undirected graph $\mathcal{G}$ may be written $\mathcal{G} = (V, E)$ for a vertex set $V$ and an edge set $E \subseteq \{\{u, v\} : u, v \in V\}$, i.e. each edge $e \in E$ is a set of two vertices from $V$. To access these sets, I define the functions $\mathrm{V}(\mathcal{G}) = V$ and $\mathrm{E}(\mathcal{G}) = E$ to retrieve the vertex and edge set respectively. For some $u, v \in V$, we may write $u \sim_{\mathcal{G}} v$ to mean $u$ and $v$ are adjacent vertices in $\mathcal{G}$ ($\{u, v\} \in E$), and use $N_{\mathcal{G}}(u)$ to refer to $u$'s *neighbourhood*—the set of all vertices in $V \backslash \{u\}$ adjacent to $u$ in $\mathcal{G}$. This definition allows loops, e.g. $u \sim_{\mathcal{G}} u$ with the caveat that $u \notin N_{\mathcal{G}}(u)$. Additionally, the *order* of $\mathcal{G}$ refers to the count of $\mathcal{G}$'s vertices ($\mathrm{Ord}(\mathcal{G}) = |V|$), and the *size* of $\mathcal{G}$ refers to the count of $\mathcal{G}$'s edges ($\mathrm{Sz}(\mathcal{G}) = |E|$). Where $\mathcal{G}$ is clear from context, the relevant subscripts will be elided.

The graphs produced by the techniques I outline are *attributed undirected multigraphs*: vertices and edges may have labels, and each pair of vertices may share multiple edges. Given domains $L_v, L_e$ for vertex and edge labels respectively, we redefine $\mathcal{G} = (V, E, \ell_v, \ell_e)$ for a vertex set $V$, an edge set $E$, a vertex label mapping $\ell_v : V \to L_v$ and an edge mapping $\ell_e : E \to \{(\{u, v\}, l) : u, v \in V, l \in L_e\}$. We can succinctly describe the edges between any $u, v \in V$ with a sorted (non-decreasing) sequence of labels, $\mathrm{seq}_{\mathcal{G}}(u, v)$. This allows us to define basic adjacency, and thus the basic neighbourhood: $u \sim_{\mathcal{G}} v \iff |\mathrm{seq}_{\mathcal{G}}(u, v)| \geq 1$. For matching such graphs, we require three new neighbourhood definitions. Given a sorted label sequence $s$, we may define the *exact neighbourhood* $N^{=}_{s,\mathcal{G}}(u)$ as the set of all $v \in N_{\mathcal{G}}(u)$ such that $\mathrm{seq}_{\mathcal{G}}(u, v) = s$. The *sufficient neighbourhood* $N^{\succcurlyeq}_{s,\mathcal{G}}(u)$ is the set of all $v \in N_{\mathcal{G}}(u)$ where we may map each label in $s$ to a distinct label of equal value in $\mathrm{seq}_{\mathcal{G}}(u, v)$—we say that $s \preccurlyeq \mathrm{seq}_{\mathcal{G}}(u, v)$ if such a mapping exists. Finally, the *overlap neighbourhood* $N^{\circ}_{s,\mathcal{G}}(u)$ is the set of all $v \in N_{\mathcal{G}}(u)$ where at least one label in $s$ may be mapped to a label of equal value in $\mathrm{seq}_{\mathcal{G}}(u, v)$. For simplicity, I shall define the core search and similarity problems using undirected graphs.

Transformation from an image to a graph will create a graph whose structure corresponds to the image's content and invariants. If an image $I$ contains an object, it is thus reasonable to assume that the graph produced from an image of only that object should be reproduced within $I$'s graph model—this may be useful when e.g., searching for a road sign in a view seen by an autonomous car. This form of graph search is known as the *subgraph isomorphism problem* (SIP), which is concerned with finding the exact structure of some pattern graph $\mathcal{P} = (V_{\mathcal{P}}, E_{\mathcal{P}})$ within a target graph $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$—an example is provided in fig. 2. More precisely, for the non-induced variant of SIP the problem is to find an injective mapping $i : V_{\mathcal{P}} \to V_{\mathcal{T}}$ such that $\forall u, v \in V_{\mathcal{P}}, i(u) \neq i(v)$ and $u \sim_{\mathcal{P}} v \Rightarrow i(u) \sim_{\mathcal{T}} i(v)$, preserving adjacency and ensuring no two vertices of $\mathcal{P}$ are mapped to the same vertex of $\mathcal{T}$. The induced variant additionally

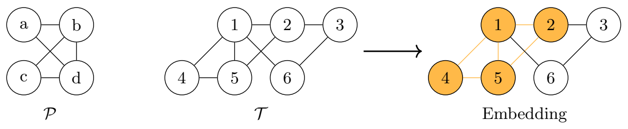


**Figure 2:** ***An example of the subgraph isomorphism problem between two graphs $\mathcal{P}$ and $\mathcal{T}$. $\mathcal{P}$'s embedding within $\mathcal{T}$ is highlighted in orange—this is a valid matching in both the induced and non-induced SIP variants.***

imposes that $\forall u, v \in V_{\mathcal{P}}, u \nsim_{\mathcal{P}} v \Rightarrow i(u) \nsim_{\mathcal{T}} i(v)$, ensuring that non-adjacent vertices in $\mathcal{P}$ must remain non-adjacent in the embedding in $\mathcal{T}$. While SIP is NP-Complete [5, 10], with modern algorithms such as McCreesh and Prosser's Glasgow solver it is computationally feasible for graphs on the order of several thousand vertices [19]. Trends in recent work have focused on increasing performance for larger instances through more costly filtering and pre-processing [28, 1, 19]—for this reason, older approaches such as VF2 [6] remain competitive on smaller instances [20].

In reality, elements of an object's graph structure may not be reproduced in the graph of an embedding due to either occlusion, image distortion or object scale. In this case, it is worthwhile to examine how much two image graphs have in common to identify common elements and substructures. Between two graphs $\mathcal{P}$ and $\mathcal{T}$ this is known as the *maximum common subgraph problem* (MCS), which is the search for the largest subgraph of $\mathcal{P}$ which is isomorphic to some subgraph of $\mathcal{T}$ as in SIP. Figure 3 offers an example instance. While this problem as described is NP-Hard, it is considerably harder in practice than SIP—the current state-of-the-art, McSplit, can operate on graphs of order 35–40 [33], where other approaches are limited to around 30 vertices. Even so, this approach applies only to the induced variant of MCS and largely prevents the addition of side constraints. For such flexibility we must consider either $k\downarrow$ [14], a repeated application of the Glasgow SIP solver excluding $k$ vertices, or a max-clique-based approach using the *association graph encoding* [18]. In this regard there is a trade-off to be made: while the clique-based approach still achieves the best performance on labelled instances [33], $k\downarrow$ requires far less memory to process high-order graphs. I choose to examine and modify $k\downarrow$ for these reasons, but this still requires that the order of any graphs must remain small to perform occlusion-robust matching with this technique.

By computing similar structures between any two graphs as above, we may then quantify how similar they are by considering the order of the MCS. This is in turn a key part of establishing how *dissimilar* a pair of graphs are, and given that MCS order does not take into account the sizes of $\mathcal{P}$ or $\mathcal{T}$ such a metric is more desirable for global classification. We may then use the MCS to compute the number of discrete changes needed to transform $\mathcal{P}$ into $\mathcal{T}$—the *graph edit distance* (GED) [2]. For classification purposes, I consider a simplified expression which disregards edge costs:

$$\mathrm{GED}(\mathcal{P}, \mathcal{T}) = \mathrm{Ord}(\mathcal{P}) + \mathrm{Ord}(\mathcal{T}) - 2\,\mathrm{Ord}(\mathrm{MCS}(\mathcal{P}, \mathcal{T}))$$

## 3. ON EXISTING IMAGE GRAPH MODELS

For graph modelling of generic images, present techniques typically build graphs by applying the *Delaunay triangulation* [8] to a set of interest pixels located within an image.

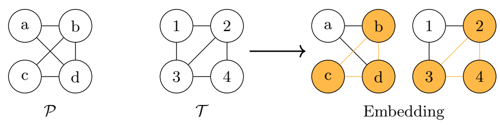


**Figure 3:** ***An example of the maximum common subgraph problem between two graphs $\mathcal{P}$ and $\mathcal{T}$. The MCS's embedding within both $\mathcal{P}$ and $\mathcal{T}$ is highlighted in orange. Note that this embedding is not unique, although for this choice of $\mathcal{P}$ and $\mathcal{T}$ the MCS itself is.***

This process places edges between vertices with coordinate labels to construct a face-connected plane graph; specifically, where each face is a triangle. Crucially, this mesh is subject to one key criterion designed to minimise the incidence of sliver triangles: no point may lie inside the circumcircle of any produced triangle. Efficient algorithms exist for its computation [35, 27]. This avenue is explored briefly by Damiand *et al.* [7], and expanded upon by Samuel, Higuera, and Janodet to make use of structural cues provided by image segmentation algorithms [26]. The authors of these approaches assess the effectiveness of their models by either searching for an image cropping within itself (using *submap* isomorphism) or by quantifying the "loss" incurred by reconstruction from the generated graph. In this regard much is made of matching of graphs *within* an image, and not *between* images; making it unclear whether any discriminative features are captured or modelled in a way that allows matching using MCS algorithms.

To investigate this, I implemented Samuel, Higuera, and Janodet's algorithm in Python, using *scikit-image*[1] functions for segmentation and interest pixel detection. Two experiments were performed:

1. Establishing whether graphs of a test image from the Berkeley Segmentation Dataset [17] (fig. 4a) before and after 180° rotation were isomorphic, and measuring their similarity if not.

2. Measuring the similarity between two adjacent frames of Sergio Leone's *The Good, the Bad and the Ugly* (figs. 4b and 4c).

As the image graphs were expected to be high-order, vertices were labelled with their $(x, y)$ image locations to augment the $k\downarrow$ procedure with Euclidean distance filtering to make matching feasible—vertices could be mapped to one another if their distance was smaller than 1 for the first, or 5 for the second experiment. These distances were chosen due to the expected similarity in each case—while the first experiment governs essentially identical images (once positions have been corrected to negate the transform), in the second case mild variance is expected.

| Experiment | Except-$k$ | Ord($MCS$) | Overlap (%) |
|---|---|---|---|
| 1 | $[61, 111]$ | $[134, 184]$ | 54.6–72.2 |
| 2 | $[38, 217]$ | $[47, 226]$ | 18.1–86.9 |

**Table 1:** ***Observed similarity bounds between real-world images.***

[1]http://scikit-image.org/

Figures 5a and 5b show that the expected isomorphism in the first experiment was *not* observed—both graphs have different order and size. After 5 days runtime, the graph order was not exactly identified in either case, but table 1 shows the obtained bounds. Due to the difficulty of finding an optimal $k$, both of these results have an unacceptably high degree of uncertainty—we cannot draw any conclusive answers on the actual graph similarity for this reason. The level of uncertainty here is strongly tied to the level of filtering provided in each experiment: a higher distance threshold gives more candidate mappings for each vertex, and so weaker filtering. The bound in the first experiment is, however, low enough for two identical images to assert that this transformation is not isotropic. This heavily indicates that the approach's validity and sensitivities depend upon the choice of interest pixel detector. Furthermore, the presence of more unique structures would improve matching performance—I believe that the structure offered by the Delaunay triangulation is too uniform, and thus hinders our ability to compute the MCS. Aside from this, the reliance on perfect segmentation (which is available within the Berkeley Segmentation Dataset) is clear—the second experiment required significant manual parameter tuning to acquire a reasonable segmentation, despite the exceptionally clear foreground-background distinction. Most pressingly, this shows that the current work is wholly inappropriate for the task of image matching with currently available tools: extremely high-order graphs with little discriminative structure are produced, making analysis computationally infeasible.

## 4. ALGORITHMS FOR MODELLING TEXT

There is a clear mismatch between current models for broad matching and the capabilities of modern algorithms. To investigate the value of MCS in image graph matching we must consider a smaller problem domain where instances have a few clear features. To this end, I choose to examine character and handwriting analysis—where the flow and form of glyphs can be captured by measuring the *topological relationships* between disconnected components, *path curvature* between keypoints and some sense of *orientation*. Ideally, this should capture feature locations and relationships in a somewhat *scale-invariant* manner.

**Algorithm 1:** Graph modelling of glyphs

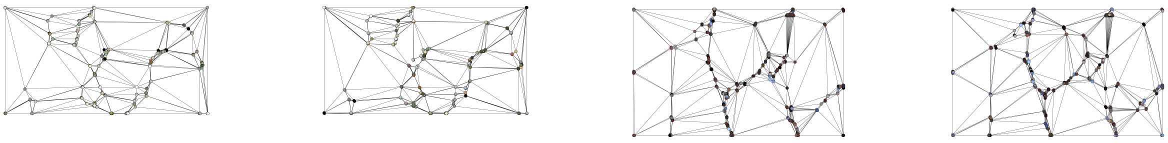
```
GlyphGraph(Image $I$, Float $curve\_thres$)
begin
    $V \leftarrow \{\}$, $E \leftarrow []$
    $B \leftarrow I$ after padding, thresholding and binarisation
    $S \leftarrow B$ after skeletonisation
    $C \leftarrow S$ after connected component labelling
    $n = \max(C)$
    $Keyps, Ends \leftarrow$ FindKeypoints($C, S$)
    $Starts \leftarrow []$
    IdentifyPaths($C$, $S$, $n$, $Keyps$, $Ends$, $Starts$, $V$, $E$)
    OrientGraph($V$, $E$)
    ComponentTopology($n$, $Keyps$, $Ends$, $Starts$, $V$, $E$)
    return $V, E$
```

### 4.1. Graphs from path curvature

For this task, I pose GlyphGraph (algorithm 1) as a solution. An input pixel image is first preprocessed by padding, greyscale conversion, thresholding and binarisation. It is

**(a)** ***BSD test image*** **(b)** ***Film frame 1*** **(c)** ***Film frame 2***

**Figure 4:** ***Images used for evaluation of Samuel, Higuera, and Janodet's [26] approach.***



**(a)** ***BSD test image, normal.*** $|V| = 245$, $|E| = 707$. **(b)** ***BSD test image,*** $180°$ ***rotation.*** $|V| = 247$, $|E| = 705$. **(c)** ***Film frame 1.*** $|V| = 264$, $|E| = 755$. **(d)** ***Film frame 2.*** $|V| = 260$, $|E| = 743$.

**Figure 5:** ***Graph output after boundary walking and Delaunay triangulation.*** $|V|$ ***and*** $|E|$ ***denote the vertex count and edge count, respectively.***

**Algorithm 2:** Keypoint location for algorithm 1

```
FindKeypoints(Image $C$, Image $S$)
begin
    $Keyps, Ends \leftarrow \{\}$
    foreach $label_{x,y} \in C$ do
        if $label_{x,y} < 1$ then
            continue
        $degree \leftarrow$ n° contiguous blocks in $\text{Nhood}_S(x, y)$
        if $degree = 1$ then
            $Ends \leftarrow Ends \cup \{(x, y, label_{x,y})\}$
        else if $degree > 2$ then
            $Keyps \leftarrow Keyps \cup \{(x, y, label_{x,y})\}$
    return $Keyps, Ends$
```

**Algorithm 3:** Path location and labelling for algorithm 1

```
IdentifyPaths(Image $C$, Image $S$, Int $n$, PointSet $Keyps$,
    PointSet $Ends$, PointList $Starts$, VertexSet $V$, EdgeList $E$)
begin
    for $i \leftarrow 1$; $i \leq n$; $i$++ do
        $PathPoints = \{(x, y, l) \in Keyps \cup Ends : l = i\}$
        if $|PathPoints| > 1$ then
            $Starts[i] \leftarrow$ any $p \in PathPoints$
        else
            $Starts[i] \leftarrow$ topmost, leftmost $(x, y, l)$ with
              $l = C[x, y] = i$
        $Paths \leftarrow$ Traverse($Starts[i], PathPoints, S$)
        foreach $(p_{start}, p_{end}, dir_{start}, dir_{end}) \in Paths$ do
            $Splits \leftarrow$ IPAN($S, p_{start}, p_{end}$)
            for $j \leftarrow 0$; $j < |Splits| - 1$; $j$++ do
                $p_1 \leftarrow Splits[j]$
                $p_2 \leftarrow Splits[j + 1]$
                $d \leftarrow \text{dist}(p_1, p_2)$
                $t \leftarrow \text{p\_dist}(S, p_{start}, p_{end}, dir_{start}, p_1, p_2)$
                $V \leftarrow V \cup \{p_1, p_2\}$
                if $curve\_thres * d < t$ then
                    $label \leftarrow$ CURVE
                else
                    $label \leftarrow$ LINE
                $E.\text{append}((p_1, p_2, label))$
```

**Algorithm 4:** North orientation for algorithm 1

```
OrientGraph(VertexSet $V$, EdgeList $E$)
begin
    $zero \leftarrow (0, 0)$
    $north \leftarrow null$
    foreach $v \in V$ do
        if $north = null \lor \text{dist}(v, zero) < \text{dist}(north, zero)$
          then
            $north \leftarrow v$
    if $north \neq null$ then
        $V \leftarrow V \cup \{zero, north\}$
        $E.\text{append}((zero, north,$ NORTH$))$
```

**Algorithm 5:** Topology labelling for algorithm 1

```
ComponentTopology(Int $n$, PointSet $Keyps$, PointSet $Ends$,
    PointList $Starts$, VertexSet $V$, EdgeList $E$)
begin
    for $i \leftarrow 1$; $i \leq n$; $i$++ do
        $eps \leftarrow [(x, y, l) \in Ends : l = i]$
        if $|eps| = 0$ then $eps \leftarrow [(x, y, l) \in Keyps : l = i]$
        if $|eps| = 0$ then $eps \leftarrow [Starts[i]]$
        foreach $(x, y, l) \in eps$ do
            $p \leftarrow (x, y)$
            $best \leftarrow null$
            foreach $\{(x', y', l') \in Keyps \cup Ends : l' \neq i\}$ do
                $v \leftarrow (x', y')$
                if $best = null \lor \text{dist}(v, p) < \text{dist}(best, p)$ then
                    $best \leftarrow v$
            if $best \neq null$ then
                $V \leftarrow V \cup \{p, best\}$
                $E.\text{append}((p, best,$ NEIGHBOUR$))$
```

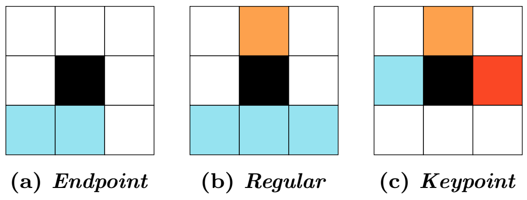

(a) *Endpoint* (b) *Regular* (c) *Keypoint*

**Figure 6:** ***Endpoint and keypoint visualisation. These diagrams represent labellings of contiguous blocks within neighbourhoods of pixels from a path, as in line 7 of algorithm 2—each (non-black) colour represents a different block, and the degree of each is the count of "colours". These have degrees 1, 2, and 3 respectively, which are used in labelling the black pixel at the centre.***

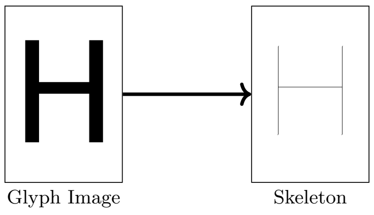


**Figure 7: Skeletonisation** ***processes a binary image by converting regions of arbitrary thickness into 1-pixel thick paths, preserving 8-wise connectivity.***

then *skeletonised* to convert the image into a set of 1-pixel-thick paths, assigning disconnected paths with different labels (lines 4–7); fig. 7 illustrates what is meant by skeletonisation. A preliminary set of interest points is extracted from the image, split into two categories: *endpoints* are pixels in the image skeleton with just one contiguous component in their neighbourhood, while *keypoints* are junctions with at least 3 such components (algorithm 2 lines 1–12, fig. 6).

Algorithm 3 then uses these keypoints to detect and label paths within components. Per component, a depth-first traversal is performed to discover the paths between all constituent interest points (line 9); this traversal prioritises moves to regular pixels over interest points, allowing interest points to be revisited since a path may start and end at the same point. In line 11, these paths are then subdivided using Chetverikov's IPAN algorithm [4] to detect locations of high curvature (using the default parameters provided in the paper). This is a two-pass approach, first testing each point $p$ along a path for the existence of an *admissible* triangle built from $p$, one point preceding $p$, and one point succeeding $p$ in the path—these points must be within a distance range from $p$, and the internal angle $\alpha$ at $p$ must be sufficiently small. If such a triangle exists, $p$ is marked as *high-curvature* and its *sharpness* $\pi - |\alpha|$ is recorded. These locations are then refined by performing non-maximal suppression with the observed sharpness values, leaving only the local maxima marked as splitting points.

Lines 12–22 then classify each path segment: if for a path segment with length $t$ and distance $d$ between that segment's endpoints $d * curve\_thres < t$, then we label it as a `CURVE`, otherwise it is labelled as a `LINE`. I consider $curve\_thres = 1.5$ as the default value. An edge is then added to the graph between these two points, with the given label—modelling path curvature features as desired.

To establish and model orientation, algorithm 4 inserts a new vertex at $(0, 0)$, and places an edge between this vertex and the closest interest point found in the image with a unique `NORTH` label. Algorithm 5 then captures the topological relationship between components. Per-component, *eps* is chosen to be either the set of that component's endpoints, keypoints or its sole start point (in decreasing order of preference). An edge, labelled `NEIGHBOUR`, is then placed between each point in *eps* and the nearest interest point in another component. Note that when outputting the final graph, vertex position data is discarded with the goal of providing scale-invariance.

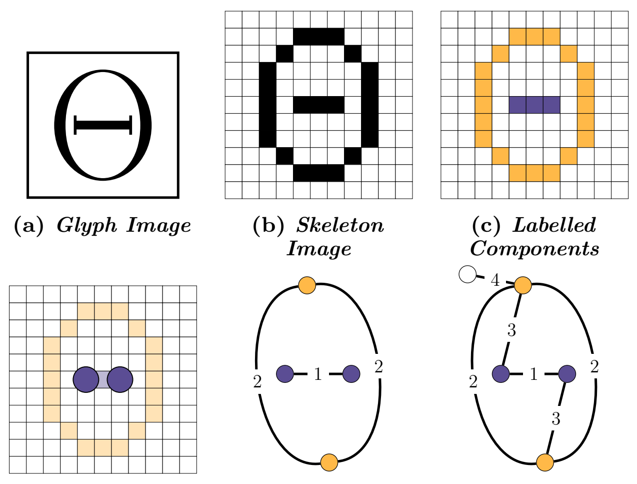
**An example walkthrough** Consider a rough visual example aided by fig. 8—the transformation of an image of the character 'Θ' into a graph. Firstly, our image (fig. 8a) undergoes preprocessing and skeletonisation to capture the glyph's paths (fig. 8b). This binary skeleton then undergoes connected component labelling, and the algorithm identifies two main components present in the image: an outer oval component (orange), and an inner line component (lavender) (fig. 8c).

Each pixel is then examined and is classified as either an endpoint, keypoint or regular pixel according to its neighbourhood (fig. 8d). The line component here features two endpoints, which are detected from their neighbourhoods. In contrast, within the oval component every pixel has two contiguous blocks within its $3 \times 3$ neighbourhood, and so all are deemed to be "regular".

Paths between keypoints within each component are then traced out and labelled to produce the first set of edges (fig. 8e). The inner component is traversed simply, starting at the left endpoint and ending at the right—the path is not split any further, and is classified as a `LINE`. As the outer component has no interest points, traversal begins at the leftmost point on its top row, before moving clockwise. Once the procedure returns to the start-point the IPAN algorithm is run on the traced path, which identifies a location of high curvature on the underside of the oval: creating a new vertex and two path segments. As the path length for each segment is sufficiently larger than the distance between its endpoints, each is classified as a `CURVE`.

The final graph (fig. 8f) is produced by considering how these components relate to one another. To capture the glyph's orientation, a purely logical vertex at $(0, 0)$ is defined with no connection to any image component: a new edge is then placed between the logical vertex and the vertex on the top-side of the oval, its nearest neighbour, labelled as a `NORTH` relation. Finally, we consider topological relationships: the endpoints of the inner component connect to their closest neighbour in the outer component with a `NEIGHBOUR` relation. The reverse also occurs from the vertices in the outer component towards the inner, but duplicate `NEIGHBOUR` relationships are not stored.

**"Indistinct" characters** When working with character images derived from fonts, `GlyphGraph` produces isomorphic graphs for characters which humans would instantly describe as distinct. As shown by fig. 9 for the DejaVu Sans font, due to unexpected "tails" and conjunctions within the binary skeletons the letters 'Z', 'H' and 'K' are jointly isomorphic. What additional information exists within the structure of these characters that can be used to make their graphs more distinct? By looking at the skeletonised forms alongside



(a) *Glyph Image* (b) *Skeleton Image* (c) *Labelled Components*

(d) *Interest Points* (e) *Path Traversal and Splitting* (f) *Final Graph*

**Figure 8:** ***Walkthrough of Θ's image→graph transformation by*** `GlyphGraph`***. Labels have been simplified:*** `LINE` = 1, `CURVE` = 2, `NEIGHBOUR` = 3, `NORTH` = 4. ***Note that the downsampling observed here between figs. 8a and 8b is*** **not** ***part of the algorithm, and is performed only to aid understanding.***

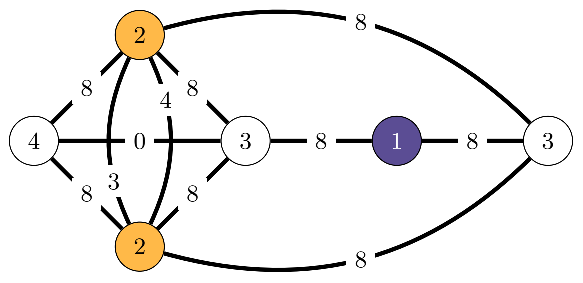


**Figure 10:** ***The dual graph of 'Θ', with path features coloured to match their parent component in fig. 8f. Vertex labels here match fig. 8's shorthand for features, where edge labels denote relative path angles. Note that both '2' edges in the regular representation start and meet at the same point, and thus have two relative angles against each other in the glyph.***

the output graph, the current modelling technique allows us to see features of glyph paths which might be used—in particular, we might wish to include positional data or the relative angle between lines and curves at interest points. The first of these options likely comes at the cost of skew- or scale-invariance, but adapting algorithm 1 to capture the latter class of features seems a worthwhile approach.

### 4.2. A "dual" representation with path direction

In the framework established, it is difficult to capture the relationship between paths since they are modelled by edges. To rework the model and enable these relationships to be captured, I start with a *dual graph*-like conversion: edges in the prior model are converted into identically labelled vertices, placing an edge between these new vertices for each time their corresponding prior edges meet at the same vertex. While this decision appears sensible since these labels *are* the features recognised, to measure (and record) relative path angles it is insufficient to consider this transformation alone—algorithm 1 measures and uses the start and end direction of each path segment, but these are not output. A modification of the initial algorithm is thus required.

First we must divide our labellings into two classes: *standard* features (lines and paths) which have well-defined directions at either end, and *special* features (neighbourhood, north) which do not. In the dual representation, edges between special↔regular edges are simply labelled '8' and special↔special edges are labelled '0'; the directional information becomes useful when considering regular↔regular relationships. These directions are an integer in the range 0–7, numbering the neighbourhood pixels around a path's endpoint clockwise from north, and are assigned based on the location of the first point adjacent to that endpoint. If two paths meet at a vertex, having directions $d$ and $d'$ at that point respectively, then their relationship is labelled $|d - d'|$. Quite usefully, we *can* return to the initial class of model from this new representation: each dual-vertex is transformed to an edge between two unknown vertices, which may be gradually inferred by iterating over the set of edges in the dual graph. Figure 10 shows the dual transformation of the regular model of 'Θ' from fig. 8.

Theoretically, this new approach should increase matching performance. The MCS procedure as I have defined it maximises the order of the produced common subgraph, yet the features in my first model lie on the edges. Running this procedure on a dual graph-like transformation gives similar results to the *maximum common edge subgraph* of the regular model, matching in a way that puts more emphasis on the count of features between a pair of graphs.

## 5. MODIFICATIONS TO K-DOWN

Having seen that both of the proposed models take the form of attributed multigraphs, Hoffmann, McCreesh, and Reilly's $k\downarrow$ requires adaptation to accommodate graphs as I have defined them. The first step in matching these graphs is to determine which sequence-aware neighbourhood (and thus adjacency operator) will be used to impose the constraints

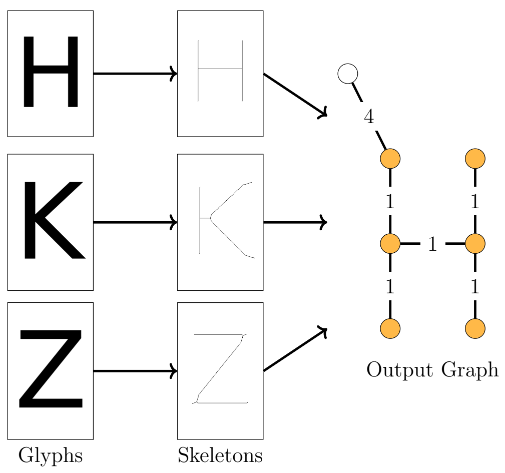


**Figure 9:** ***Characters with isomorphic structure, which are thus "identical"*** `GlyphGraph` ***models.***

arising from edge labels. Each corresponds to a different level of filtering:

- the exact neighbourhood $N^{=}_{s,\mathcal{G}}$ is necessary in the *induced* variant of MCS, so that a pair of pattern and target vertex pairs may only map to one another if they have identical edge sequences;
- the overlap neighbourhood $N^{\circ}_{s,\mathcal{G}}$ provides the bare-minimum level of filtering needed when considering the *non-induced* variant of MCS, allowing a pair of pattern and target vertex pairs to map to one another if there is any overlap in the edge sequences;
- the sufficient neighbourhood $N^{\succcurlyeq}_{s,\mathcal{G}}$, which allows a pattern vertex pair to be mapped to a target vertex pair if every edge in the former can be mapped to an edge of equal value in the latter, offering an oddly asymmetrical mapping when used in MCS—I define this (in the context of SIP) as building an *edge-count-increasing* (ECI) *subgraph isomorphism.*

Algorithm 6 then defines the necessary modifications to $k\downarrow$ to compute the MCS in each case. At the top of search, the matching procedure must enforce that vertices of $\mathcal{P}$ can only be mapped to vertices of $\mathcal{T}$ with equal labels. Additionally, any vertices featuring loops must meet the selected sequence-adjacency criterion (lines 6–7). Per node of constraint propagation, domains are pruned using the chosen sequence-aware neighbourhood—if $v$ is mapped to $v'$ and $v$ and $w$ are adjacent with edge sequence $s$ in $\mathcal{P}$, then $w$ may only be assigned to vertices in the $s$-neighbourhood of $v'$ (lines 15–18).

The rest of the algorithm is unaffected by these additions; by providing both multigraph and simple graph notions of neighbourhood and adjacency, the existing filtering provided by supplemental graphs is maintained. Admittedly, further supplemental graphs could be added, filtering on commonly seen label patterns in paths; although with the relatively low-order graphs produced by the examined models such optimisations would be largely unnecessary for this study.

**Algorithm 6:** Modifications to k$\downarrow$ [14] for attributed multigraphs, given (for a graph $\mathcal{G}$) a vertex labelling function $\ell_{\mathcal{G}}$, a chosen multigraph neighbourhood function $\mathcal{N}_{s,\mathcal{G}}$ and its associated adjacency operator "$\sim_{s,\mathcal{G}}$".

```
1  // top-of-search filtering with sequences on loop
       constraints
2  foreach $v \in \mathrm{V}(\mathcal{P})$ do
3      $D_v \leftarrow \mathrm{V}(\mathcal{T})$
4      foreach $(P, T) \in L$ do
5          $D_v \leftarrow \{w \in D_v : v \sim_P v \Rightarrow w \sim_T w \wedge$
             $S_P(v)\ {}_k\!\preceq S_T(w)\}$
6      $s \leftarrow \mathrm{seq}_{\mathcal{P}}(v, v)$
7      $D_v \leftarrow \{w \in D_v : \ell_{\mathcal{P}}(v) = \ell_{\mathcal{T}}(w) \wedge v \sim_P v \Rightarrow w \sim_{s,\mathcal{T}} w\}$
8      $D_v \leftarrow D_v \cup k$ distinct wildcard vertices

9  // filtering with sequences during propagation
10 foreach $D_w \in D \backslash \{D_v\}$ do
11     $D_w \leftarrow D_w \backslash \{v'\}$
12     foreach $(P, T) \in L$ do
13         if $v \sim_P w$ then
14             $D_w \leftarrow D_w \cap (N_T(v') \cup \mathit{wildcards})$
15     $(P_0, T_0) \leftarrow L[0]$
16     if $v \sim_{P_0} w$ then
17         $s \leftarrow \mathrm{seq}_{P_0}(v, w)$
18         $D_w \leftarrow D_w \cap (\mathcal{N}_{s,T_0}(v') \cup \mathit{wildcards})$
19     if $D_w = \emptyset$ then
20         return false
```

## 6. EXPERIMENTAL EVALUATION

To test the effectiveness of these algorithms for character modelling and classification, three experiments were performed:

1. Firstly, it is necessary to determine whether these graph models allow matching of characters formed in distinct, uniform, machine-generated styles. This is explored via the construction of "confusion matrices" between all graphs of characters in the Latin alphabet from several fonts, both lower- and upper-case. To this end, the order of the maximum common subgraph and graph edit distance are displayed via heat-maps to show character graph (dis-)similarity within and between fonts. For this, I chose three fonts to investigate: *DejaVu Sans*, *Open Sans* (both sans-serif), and *Alegreya* (serif). The aim is to show that the generated graphs are reasonably distinct from one another within a human definition of similarity (i.e. we might expect 'o' and 'O' to produce identical graphs). Furthermore, I investigate whether the modifications introduced in section 4.2 create more distinct glyph models as hypothesised. The experiment is judged to be a success if most character pairs do not exhibit isomorphism within each font, having a GED close to the median graph size. Naturally, a character will be isomorphic to itself *within* a font, and thus another success criterion is that its model is ideally similar to the graph model of that same character from another font.

2. To determine how well these models capture core character features between different styles of writing, I apply a *k-Nearest Neighbours* ($k$NN) classifier to the HWRT database of online handwritten character data [32] (setting $k = 5$ as in [29]). Images (and graphs) were constructed from the online path data using a selection of pen-radii, for $r \in \{1, 5, 9\}$, to explore how different levels of connectivity would affect graph structure and matching accuracy. Specifically, this online data decomposes paths into positions on a canvas with attached timestamps: images were reconstructed by applying Bresenham's line algorithm between these path points, generating a list of locations to place a circle of thickness $r$. This method recreates glyph images at the scale at which they were drawn. Additionally, I attempt to explore how variation of the path curvature threshold during modelling affects classifier performance, examining *curve_thres* $\in \{1.2, 1.35, 1.5, 1.65\}$. For this classifier graph edit distance is the chosen metric, setting a timeout of 30 s per graph comparison. Due to the vast size of the dataset I restrict the database to the set of upper-case Latin alphabet characters, using a total of 2000 randomly selected glyphs split into *test* and *training* data. As the database is already divided into a test and training set, I take subsets of each maintaining the existing proportion. This experiment assesses

| Font | Model | Graph Order | | | Graph Size | | |
|---|---|---|---|---|---|---|---|
| | | Min | Max | Median | Min | Max | Median |
| Alegreya | Regular | 2 | 26 | 11 | 2 | 25 | 11 |
| | Dual | 2 | 25 | 11 | 3 | 37 | 14.5 |
| DejaVu Sans | Regular | 2 | 14 | 6 | 2 | 13 | 6 |
| | Dual | 2 | 13 | 6 | 1 | 14 | 7 |
| Open Sans | Regular | 2 | 17 | 6 | 2 | 16 | 6 |
| | Dual | 2 | 16 | 6 | 1 | 18 | 7 |
| HWRT ($r = 1$) | Regular | 2 | 113 | 7 | 2 | 172 | 6 |
| | Dual | 2 | 172 | 6 | 1 | 398 | 8 |
| HWRT ($r = 5$) | Regular | 0 | 40 | 7 | 0 | 55 | 6 |
| | Dual | 0 | 55 | 6 | 0 | 116 | 7 |
| HWRT ($r = 9$) | Regular | 2 | 59 | 7 | 1 | 126 | 8 |
| | Dual | 2 | 41 | 7 | 1 | 126 | 8 |
| Washington | Regular | 7 | 75 | 33 | 6 | 93 | 42 |
| | Dual | 6 | 93 | 42 | 6 | 170 | 75 |

**Table 2:** ***Aggregate statistics on graph order and size produced from the used datasets.***

whether the features captured within any glyph were consistent between similar images in a way that enabled optical character recognition—a sufficiently high accuracy ($\sim 90\,\%$) would indicate success. Additionally, I assess effectiveness using the *Cohen's Kappa* metric, which takes ground truth classification counts into consideration for effectiveness: a value $\kappa \leq 0$ means that decisions are equal to or worse than random chance, and $\kappa = 1$ indicates perfect agreement with the ground truth.

3. Finally, it is important to position this work in relation to existing graph models which have been applied to handwritten word recognition. For this task, I apply the same classifier to a subset of the George Washington Letters dataset[2] made available by Stauffer, Fischer, and Riesen [29]. I use the same parameters on the classifier ($k = 5$, 30 s timeout), also varying *curve_thres* as in Experiment 2. This experiment allows me to directly compare the modelling efficiency of my graph models combined with the defined GED against Stauffer, Fischer, and Riesen's existing modelling techniques. As these graphs model whole words, graph order is likely to become an issue, harming classifier performance—made aware of this, I consider a lower classifier performance ($\sim 80\,\%$, roughly the result of Stauffer, Fischer, and Riesen) to be successful.

All experiments were run on a Windows 10 Pro x64 machine using the Windows Subsystem for Linux, with an Intel Core i7–920@2.67 GHz and 12 GiB RAM. All code used for these experiments is publicly accessible on GitHub[3]. In all cases, statistics on generated graph order and size were collected in table 2, allowing further commentary on suspected reasons behind matching performance. Similarly, each experiment was attempted at all levels of filtering defined in section 5 to determine the related effects of each on classifier performance.

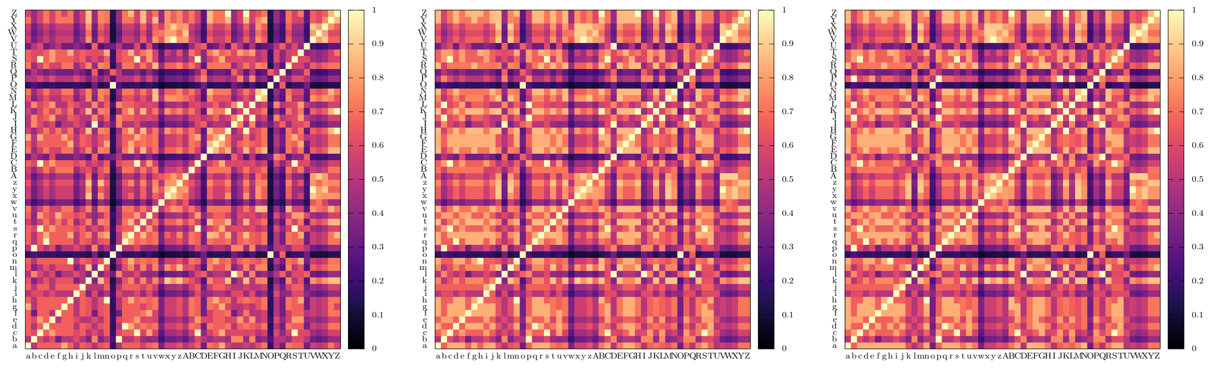
## 6.1. Results and discussion

**Experiment 1** Figure 11 shows the effect of each level of $k\downarrow$ filtering upon character distinctness during matching. Few differences are noted between the edge-count-increasing and non-induced levels of filtering (figs. 11b and 11c respectively), likely as they both build on the use of the non-induced variant of MCS. The induced MCS (fig. 11a), on the other hand, finds the set of characters to be largely more distinct from one another due to its stronger filtering. A more thorough examination reveals that many sets of unexpected isomorphisms remain: {Z, H, K} as explained in fig. 9, {n, h}, {c, s, C, S}, {b, p}, and {q, d} to name several. While many of these are expected due to identical upper- and lower-case forms, several of these suggest that the original model lacks discriminative power and captures too few features; chiefly, these appear to be path length, path angle dynamics and glyph size. Throughout all comparisons, it is clear that many of these graphs still exhibit high similarity to one another (60–70 % in the induced case), implying that many of the character graphs share identical components. This may arise from the fact that the MCS is not *connected*, i.e. I do not impose that there must be a path between every pair of vertices in the MCS; this allows the addition of extra vertices which have no paths to the main components during matching. For instance, in every graph an edge marked `NORTH` between a pair of vertices is guaranteed to exist, and during search may be added to the MCS "for free" if it is disconnected from the current incumbent. In future, it may thus be worthwhile to consider the *maximum common connected subgraph problem* (MCCS), which does impose such restrictions [18].

Having established the induced variant of MCS as the most discriminative, fig. 12 then presents the effects of the dual encoding upon glyph similarity for this level of filtering. Broadly, figs. 12a and 12b show an overall reduction in similarity across the entire space of comparisons—showing that the dual graph representation is successful in making glyph models more distinct from one another. Many of the unexpected isomorphisms *have* been removed by this modification to the algorithm, yet not all; note that {H, K}, {n, h}, {b, p}, and {q, d} remain isomorphic, alongside many of the prior letters with "identical" upper- and lower-case forms. While it is unsurprising that a few such isomorphisms remain (we have not yet modelled the phenomena which actually differentiate them), revisiting fig. 9 sheds some light on the case of 'H' and 'K'. Note that in 'K', the skeleton paths from the right junction start vertically leading to identical path dynamics between the two glyphs, which of course suggests further modifications which could be made to account for *overall* path direction. I do not consider such further modifications here.

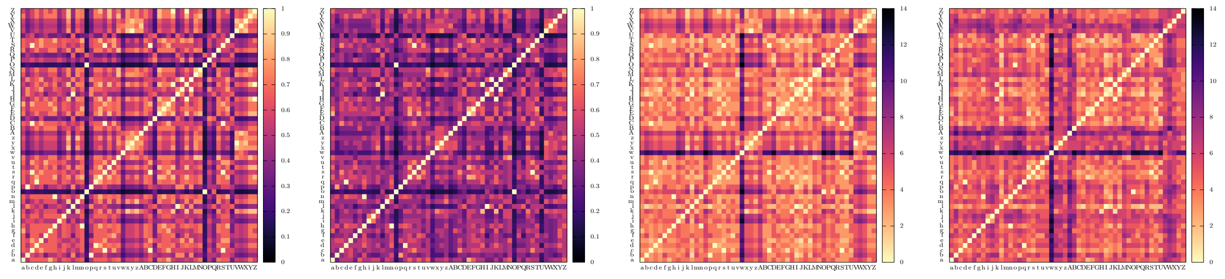
Figures 13 and 14 then demonstrate the observed similarity *between* fonts of different classes, examining DejaVu Sans vs. Open Sans and Alegreya respectively. In the former case, while some portion of the expected isomorphisms were observed it is clear from the visual similarity of the fonts themselves that the model was unable to capture the core features that define many of the glyphs. Somewhat interestingly, changing to the 'Dual' encoding from fig. 13a to fig. 13b *removed* some of these observed isomorphisms, indicating different curvature along the traversed paths. Indeed, curvature may well be the root of many of the other 'missing' isomorphisms, as path-splitting using the IPAN algorithm is likely to have some sensitivity to what humans might appreciate as minor variation. In the case of DejaVu Sans against Alegreya, while some clear glyph isomorphisms are preserved between fonts ({o, O}, and 'Q'), the font graphs exhibit startling dissimilarity. A considerable factor here is the difference in graph order of these font classes; serif fonts display far more variation and ornamentation, particularly on letter path endpoints. This naturally leads to more paths

[2] `http://histograph.ch`
[3] `https://github.com/FelixMcFelix/sip-for-cv-paper`



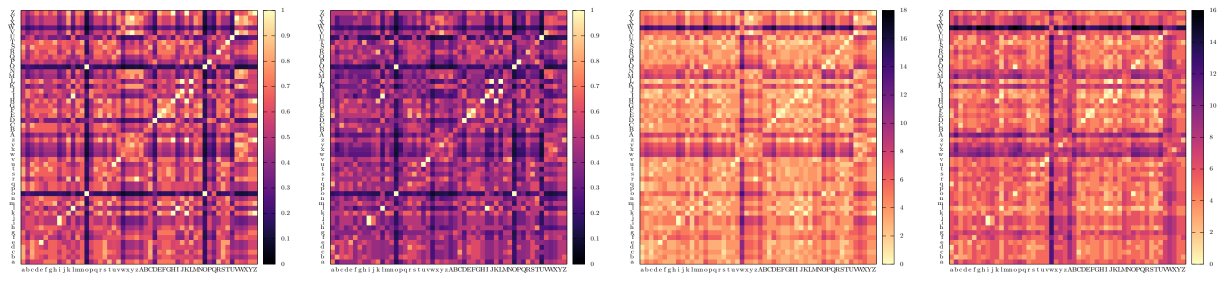
(a) *Induced* (b) *Edge-count-increasing* (c) *Non-induced*

**Figure 11:** ***Heatmaps displaying graph similarity (graph order of MCS) between glyph graphs in the DejaVu Sans typeface, normalised by the order of the largest graph in the comparison. These plots show the effects of the different levels of filtering introduced in section 5. Warmer colours imply more similarity, where cream means isomorphism—an overall darker heatmap shows that characters are seen as more distinct when matching under the given filtering level. It can thus be seen that characters are perceived as being more distinct when computing the induced MCS.***



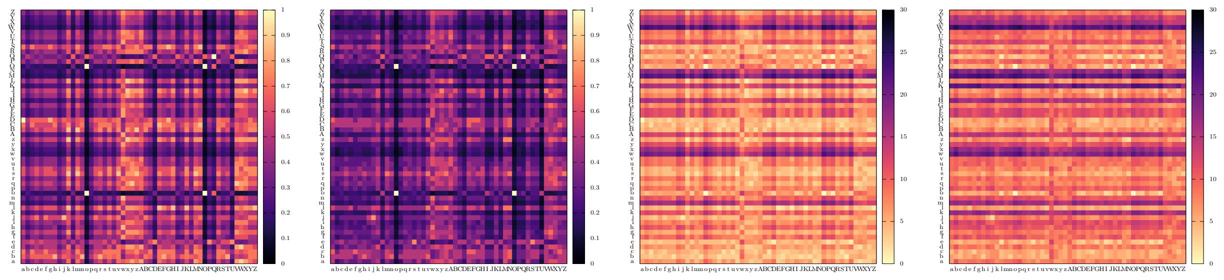
(a) *MCS: Regular* (b) *MCS: Dual* (c) *GED: Regular* (d) *GED: Dual*

**Figure 12:** ***Heatmaps displaying graph similarity (MCS) and dissimilarity (GED) in the DejaVu Sans typeface as in fig. 11, comparing the distinctiveness of character graphs from the 'Regular' and 'Dual' algorithm variants. For GED, lighter colours again imply similarity, where cream implies isomorphism—characters are more distinct in one encoding if lower similarity is observed outside of the diagonal. Crucially, we can see that the 'Dual' encoding leads to more distinctive characters as expected.***



(a) *MCS: Regular* (b) *MCS: Dual* (c) *GED: Regular* (d) *GED: Dual*

**Figure 13:** ***Heatmaps displaying graph (dis-)similarity between the DejaVu Sans and Open Sans typefaces, conveyed as in fig. 12. Ideally, matching characters are expected to be almost isomorphic—high MCS order and low GED are expected along the diagonal. While some isomorphisms and general similarities per-character are preserved between fonts, many of these are lost.***



(a) *MCS: Regular* (b) *MCS: Dual* (c) *GED: Regular* (d) *GED: Dual*

**Figure 14:** ***Heatmaps displaying graph (dis-)similarity between the DejaVu Sans and Alegreya typefaces. Serifs (such as those of Alegreya) modify many characters at the end of paths—ideally, core elements of the letter forms should remain identical. Criteria for analysis match those of fig. 13. Aside from 'o', 'O' and 'Q', neither the 'Regular' nor 'Dual' graphs reliably enable matching between two different typeface styles.***

between these points, producing a rather different structure overall. Given the nature of how `NEIGHBOURHOOD` and `NORTH` relations are formed, the existence of serifs further complicates matching as these relations are more likely in practice to concern endpoints.

While successful within a font, the overall results do not bode well—they indicate that the modelling techniques and/or the similarity measures are not fit for purpose when comparing writing in different styles or even regular variation within a single style. Overall on the specified machine, these experiments typically took around 30 min each to execute all $52 \times 52$ graph comparisons.

**Experiment 2** Tables 3 and 4 show matching effectiveness for the 'regular' and 'dual' graph models respectively, as pen width and filtering level are varied. Some clear trends are visible from this data: we can observe that accuracy due to filtering level follows the same trend as character distinctiveness in Experiment 1, that stricter matching conditions (specifically, the induced variant) lead to greater classifier accuracy.

Curiously, more distinctive character modelling approaches *do not* lead to greater classifier performance. In most cases, excluding non-induced filtering and edge-count-increasing for $r = 5$, the 'regular' graph model outperforms its 'dual' model competitor; while the prior results indicate that stricter matching and modelling should aid classification accuracy, the dual model might be overly sensitive to the angle variations and dynamics which are most likely to appear in handwritten text. In this sense, the dual models are *overfit* to glyphs extracted from fonts. The increased matching performance for non-induced filtering on the dual models can be rather roughly justified—the matching procedure has more "leeway" to match these overfit glyph models against one another, as non-adjacent vertices may be mapped to one another and the lax sequence overlap rules allow many conflicting edge labels to be disregarded. This does not, however, cancel out the fact that induced filtering is unilaterally the best option.

In all cases, it can be observed that a choice of $r = 1$ leads to the greatest matching accuracy—that is, I find that it is better to use forms closest to the online data rather than attempting to enhance path connection features with a larger stroke thickness. This result is somewhat striking, given the related rows of table 2. While median graph order remains roughly consistent over all $r$, we can see that $r = 1$ has the largest outlier graphs and so we might expect it to actually have the *worst* performance as graph comparisons would be more likely to time-out. Larger pen radii, while potentially enhancing the models of glyphs with unintended gaps, run the risk of destroying the separateness of components which were never intended to connect. Interestingly, the contributed change to accuracy relies strongly upon the chosen model: accuracy decreases proportionally with $r$ in the dual encoding, yet consistently sees an increase in accuracy after $r = 5$ in the regular model, although it is difficult to offer commentary on *why* this behaviour is observed.

Table 5 explores the effect of varying *curve_thres* upon accuracy, building upon the best seen classifier parameters (induced, $r = 1$, regular). A lower threshold value means that the algorithm is more likely to label path segments as curves rather than lines, whereas a high threshold will make curve detection less sensitive overall; extremal values in either direction will essentially remove the line-curve distinction. In the given examples, the statistics suggest that a more sensitive curve detector is correlated with higher matching performance (likely to some undiscovered lower bound). Indeed, this trend repeats throughout all tested parameter choices. Why might we observe this? Natural variation in the handwritten samples after path splitting might leave many curves undetected—in this case, a lower curve threshold means this variation is accounted for, and so the model is less sensitive overall to minor curvature variations.

Within each choice of parameters, I examined whether the accuracy *for each label* had any correlation with mean graph order, size, degree, or line-to-curve ratio. While it might be expected that these observed features could be a useful indicator of difficulty within a label class, plotting these features revealed no overly distinct correlations with labelling accuracy. To investigate whether *overall* accuracy had any correlation with these statistics, I considered any parameter combinations with induced filtering upon the regular model. Figure 15 shows the correlations observed between mean graph size and accuracy—larger graph size (i.e. feature count) and fewer `LINE` features correlate with higher accuracy. None of the other statistics exhibited any correlation.

Unfortunately, the more accurate parameter choices come at the cost of execution time. Table 6 contains a guideline set of typical execution times estimated from the filesystem for this experiment, showing that these times grow for more stringent filtering levels, and typically decrease as $r$ increases. This presents a relatively clear accuracy/time trade-off, as

| Filtering | $r$ (px) | Accuracy (%) | $\kappa$ |
|---|---|---|---|
| Non-induced | 1 | 27.15 | 0.220 |
| | 5 | 16.74 | 0.106 |
| | 9 | 22.17 | 0.160 |
| ECI | 1 | 30.32 | 0.253 |
| | 5 | 18.10 | 0.122 |
| | 9 | 23.53 | 0.175 |
| Induced | 1 | 38.46 | 0.347 |
| | 5 | 29.86 | 0.256 |
| | 9 | 30.77 | 0.271 |

**Table 3:** ***Classifier accuracy and Cohen's kappa ($\kappa$) for the 'regular' model against the HWRT dataset when varying pen-thickness ($r$) and filtering class. Here, I use the default*** *curve_thres* $= 1.5$***. A higher accuracy and $\kappa$ both imply better classifier performance.***

| Filtering | $r$ (px) | Accuracy (%) | $\kappa$ |
|---|---|---|---|
| Non-induced | 1 | 28.05 | 0.235 |
| | 5 | 23.08 | 0.180 |
| | 9 | 21.72 | 0.166 |
| ECI | 1 | 26.24 | 0.213 |
| | 5 | 24.43 | 0.200 |
| | 9 | 21.27 | 0.161 |
| Induced | 1 | 34.84 | 0.312 |
| | 5 | 27.15 | 0.233 |
| | 9 | 26.24 | 0.222 |

**Table 4:** ***Classifier accuracy and Cohen's kappa ($\kappa$) for the 'dual model' on HWRT data, as in table 3.***

the induced level of filtering in particular imposes stricter requirements on a solution. Overall, these did not appreciably vary according to the choice of *curve_thres*, with the exception of the settings (induced, $r = 9$, dual), where an increased threshold led to faster classification time.

While the best observed $\kappa$ indicates a "fair" degree of progress above the random baseline, these results are far outmatched by the current state-of-the-art in both graph modelling of glyphs and standard recognition approaches. Furthermore, the costly matching time imposed by the use of exact MCS makes the combined use of this classifier and these models very undesirable in practice, even though the graphs are typically of a suitable order. In these regards, this experiment is deemed a failure. However, this work does not disentangle the effects of the dual representation itself from the effect of the allocated angle labels—this must be explored in future work.

**Experiment 3** From table 7, we can see that the results on the Washington dataset are far from satisfactory, and have $\kappa$ scores so low that they are little different from random chance. A key factor in this lies in the graph statistics of table 2: the regular model has a median graph order of 33, which is pushing close to the limits of $k\downarrow$, keeping in mind that half of all graphs in this dataset will be *larger* than this it quickly becomes clear that many of the individual GED computations timed out. While the median graph order is lower than the most successful approaches of Stauffer, Fischer, and Riesen, many of these graphs likely exceed the capabilities of $k\downarrow$ within a 30 s time window. Noting that the classifier returns a default '0' label for unclassified graphs,

| *curve_thres* | Accuracy (%) | $\kappa$ |
|---|---|---|
| 1.2 | 40.72 | 0.373 |
| 1.35 | 38.46 | 0.347 |
| 1.5 | 38.46 | 0.347 |
| 1.65 | 35.75 | 0.318 |

**Table 5:** ***Classifier accuracy and Cohen's kappa ($\kappa$) for the 'regular' model against HWRT data as*** *curve_thres* ***is varied, with induced filtering and $r = 1$. A lower*** *curve_thres* ***means that edges are more likely to be labelled as curves, and this corresponds to better classification performance. This trend is observed throughout all other examined parameter choices.***

| Filtering | Model | $t_{r=1}$ (h) | $t_{r=5}$ (h) | $t_{r=9}$ (h) |
|---|---|---|---|---|
| Non-Induced | Regular | 0.5 | 0.25 | 0.25 |
| | Dual | 2 | 1 | 0.5 |
| ECI | Regular | 1 | 0.25 | 0.25 |
| | Dual | 2 | 1 | 0.5 |
| Induced | Regular | 6 | 2 | 1 |
| | Dual | 4 | 1 | 0.5–1 |

**Table 6:** ***General runtime statistics for Experiment 2, per model and variant, measured in hours.***

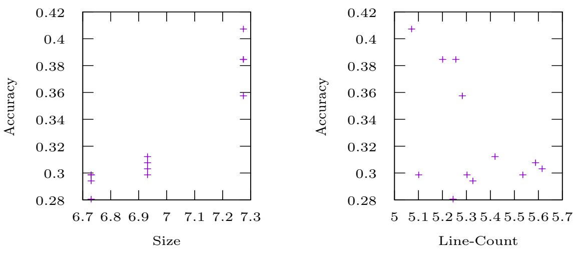


**Figure 15:** ***Plots of mean graph size and count of*** `LINE` ***edges against overall accuracy, for settings with induced filtering on the regular model. In these cases, larger graph size (i.e. feature count) and fewer*** `LINE` ***features correlate with higher accuracy.***

the disturbingly high incidence of this classification confirms that time-outs during computation are the root cause of the problem.

Ultimately, results on the actual effectiveness of this dataset are inconclusive for these reasons—mandating further investigation with a cheaper, approximate GED metric to establish the proposed models' true effectiveness. Furthermore, this issue may have silently influenced the results of Experiment 2, hinting that future reinvestigation of both these studies using the work of Riesen and Bunke [23] may be worthwhile. Given the relatively small dataset size (and the low accuracy), it as not deemed worthwhile or meaningful to attempt to identify the affects of different *curve_thres* on classification performance. Considering run-times, the vastly larger median graph order of the Washington dataset (table 2) ensured that runs of these tests took far longer than the prior experiments, coming in at around 24–36 h dependant on filtering level and the chosen representation.

## 7. RELATED WORK

**General image modelling** The *Scale-Invariant Feature Transform* (SIFT) and similar approaches have historically

| Filtering | Model | Accuracy (%) | $\kappa$ |
|---|---|---|---|
| Non-Induced | Regular | 11.67 | 0.086 |
| | Dual | 5.00 | 0.017 |
| ECI | Regular | 10.00 | 0.069 |
| | Dual | 5.00 | 0.017 |
| Induced | Regular | 8.33 | 0.052 |
| | Dual | 3.33 | 0.000 |

**Table 7:** ***Classifier accuracy and Cohen's kappa ($\kappa$) against the Washington dataset when varying filtering class. A higher accuracy and $\kappa$ both imply better classifier performance. Here, I use the default*** *curve_thres* = 1.5—***due to the very large typical graph order, too few accurate classifications were made to reason about the effects of*** *curve_thres*. ***The current state-of-the-art in graph modelling achieves an accuracy of*** 81.82 % ***with the 'Projection' technique on this task [29].***

been extremely popular vector-space models for image modelling and matching [16, 36]. These techniques typically operate by identifying locations of high variation within a multiscale representation of an image such as a *Laplacian pyramid*, capturing high-dimensional gradient neighbourhoods of these points as features.

*Convolutional neural networks* (CNNs) have been a sizeable area of focus recently, with approaches such as Krizhevsky, Sutskever, and Hinton [15] proving able to accurately learn image labellings from large datasets. These models operate by statistically learning *kernels* from a corpus of training data to perform image convolution, transforming an image several times to eventually provide an output classification. Since these kernels are relatively high-dimensional vectors of real values, these are not typically transformations which can be intuited easily. This reinforces the "black-box" nature of these models, and in combination with work on adversarial images [12] it is clear that the sensitivities and weaknesses of such models are extraordinarily difficult to estimate. While complex networks can require a large volume of training data and have a costly training phase, CNNs achieve very high accuracy above and beyond existing vector-space models while having good performance on modern commodity hardware. Primarily, this is due to the introduction of fast convolution operations on graphics processing units. Notably, the cost of classification *does not typically scale* with the size of the training data set, as the model learned is a *fixed representation.* Further advances by Wei *et al.* have extended such object classifiers with intelligent segmentation algorithms [3] to enable multiple labelling and accurate object location [34].

**Handwriting and character recognition** Techniques in this field are broadly split into two categories; *online* classifiers use direct path input data as captured by a computer, and *offline* classifiers which use image data (typically extracted from some real-world context). Within this framework, the approach I have introduced and evaluated is an offline approach.

Currently, neural networks (both standard and convolutional) are a key model within offline recognition [31, 30]—able to tackle word recognition and spotting to high accuracy in different scripts and language datasets. Aside from this, the use of SIFT-like keypoints for the task of offline Arabic word recognition has been successfully explored within a bag-of-features Hidden Markov Model [25], achieving a lower complexity (and similar effectiveness to) competing techniques. This approach has been further applied to the task of word-spotting from the George Washington dataset [24].

Several existing approaches towards offline graph modelling of handwritten text are provided by Stauffer, Fischer, and Riesen [29]. Their work focuses on analysis of the George Washington letters dataset by a $k$NN classifier using an approximate GED metric [23]. In all cases, vertices are labelled with normalised $(\hat{x}, \hat{y})$ coordinates with edges left unlabelled. The first of their approaches corresponds closely to my own—endpoints and junctions are identified within a skeletonised glyph image, and edges are placed by following paths and creating new vertices for every distance $d$ travelled. Other approaches focus on grid-based segmentation (with edges provided through various techniques) and adaptive segmentations combined with edges from path traversal. They find that the keypoint-based approach has 77 % accuracy at the cost of very high graph order and size, while adaptive segmentations produce lower order graphs with higher ($\sim$ 80 %) accuracy. Interestingly, they note that the use of the Delaunay triangulation for such models results in reduced classifier performance ($\sim$ 63 %) with the highest graph sizes, supporting my earlier findings in section 3.

## 8. CONCLUSIONS AND FUTURE WORK

Throughout this paper, I have shown the shortcomings of current general graph modelling techniques on pixel images, and presented and evaluated two alternative modelling algorithms for a simplified problem domain. Generally, the new glyph graph models achieve admittedly poor performance in comparison with the current state-of-the-art in handwriting recognition and word spotting (graph models and otherwise). A common thread throughout has been the unsuitability of exact MCS algorithms for the task of detecting similarity in feature-rich models, where the per-graph comparison time appears to have a distinct effect on classification accuracy. The sizeable execution time per-comparison compounds with the $k$NN classifier's design, making classification time scale with the size of the training dataset in a very costly manner as model size grows.

The models are, however, as intuitive as hoped. When matching at the character-character level, I've shown that the graph structure is a seemingly natural way to model this information, making it easy not only to see *why* individual misclassifications occur but to suggest modifications which make characters more distinct. This is a double-edged sword to some extent; while the modifications brought by the dual model allow for more effective modelling of glyph images from font data, they cause stark *overfitting* due to hidden assumptions about variation in path starting angle.

Using only GED as the scoring metric may itself be inappropriate for preserving certain character features. Consider the relatively small GED between 'p', 'q', 'b' and 'd', where arguably the largest difference comes from the single orientation vertex and its `NORTH` edge to another keypoint. A more useful distance metric on these models might score the loss or inclusion of certain features to heavily penalise orientation differences, but this leads us farther from standard graph matching algorithms towards domain-specific similarity scores.

Existing ML and keypoint models are effective because they are designed to learn features of arbitrary high-dimensional data which are distinctive and discriminative in their own right. Graph models appear to need to be designed specifically for the task at hand, and for more general matching tasks it is not immediately clear if low-level image modelling in this way is possible since we require an explicit notion of entities and relationships. The explored graph models seem to capture *some* level of discriminative features from their input glyphs, and by varying the defined parameters it was possible to create a classifier operating appreciably above the random baseline. There is perhaps some merit in elements of the outlined techniques. Without attempting to measure performance with an approximate GED metric, it is difficult to say whether this approach requires a complete rethink or simply further tweaking to be truly effective on the assessed tasks. I believe a significant redesign is necessary, though given the differences between my approaches and Stauffer, Fischer, and Riesen [29] I believe it may be worthwhile to investigate how their models fare with the introduction of line/curve semantics.

### 8.1. Future work

The results shown necessitate investigation of these models with an approximate GED metric (such as Riesen and Bunke [23]) to cement this work's position within the literature. In a similar vein, having seen that the induced MCS offers the greatest classifier performance it may be useful to consider a modification of the McSplit algorithm [33] or consider the MCCS rather than MCS—although this would almost certainly provide less benefit than simply approximating the edit distance. There is also some value in modifying the work of existing, successful graph models such as Stauffer, Fischer, and Riesen [29] to see how features like line and curve segment capture might affect classification performance. As alluded to above, for both this model and the proposed modification of the state-of-the-art an investigation of feature-aware custom scoring functions might be of use. This would allow us to penalise the omission of certain features, with the caveat that this would no longer be a standard graph matching problem. The work of Redmond and Cunningham [22] on the *temporal subgraph isomorphism problem* focuses on matching general patterns of edge labels between graphs, rather than performing exact label matching. Rich side-constraints such as these could offer key insight on techniques to accurately match path angle dynamics in a less rigid fashion. These could also be combined with the temporal data found within the HWRT online dataset itself.

If greater performance can be achieved, it may well be worthwhile to consider other problem domains which rely upon path-based structures. One such domain would be the identification of common components and design patterns within *circuit diagrams*. Since there are few ways to build components with known functionality (i.e. logic gates and amplifiers), it may be possible to locate such components and describe the blueprint at a higher level using graph model and search techniques. The intricate structure of these diagrams would likely make MCS computation almost impossible, but if we assume that input images are relatively free of distortion then SIP algorithms become a natural fit for the problem.

The use of approximate similarity measures may enable lower-level graph models for general images, such as using multi-scale keypoint features as vertices with a suitable method of defining edges. While this would lead to higher graph order due to the large volume of feature vectors which SIFT-like approaches can output, approximate measures should allow the use of larger input graphs—potentially allowing the use of graph models in more general computer vision tasks. The specifics of how to approach such a model are very much an open question, however.

**Acknowledgements.** I'd like to thank my supervisors Patrick Prosser, John Williamson and Ciaran McCreesh for their comments and guidance throughout the year. I'd also like to thank my girlfriend Vanessa Stevenson and my family for their continued support.

## 9. REFERENCES

[1] G. Audemard, C. Lecoutre, M. S. Modeliar, G. Goncalves, and D. C. Porumbel. Scoring-based neighborhood dominance for the subgraph isomorphism problem. In B. O'Sullivan, editor, *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, volume 8656 of *Lecture Notes in Computer Science*, pages 125–141. Springer, 2014. ISBN: 978-3-319-10427-0.

[2] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[3] M. Cheng, Z. Zhang, W. Lin, and P. H. S. Torr. BING: binarized normed gradients for objectness estimation at 300fps. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 3286–3293. IEEE Computer Society, 2014. ISBN: 978-1-4799-5118-5.

[4] D. Chetverikov. A simple and efficient algorithm for detection of high curvature points in planar curves. In N. Petkov and M. A. Westenberg, editors, *Computer Analysis of Images and Patterns, 10th International Conference, CAIP 2003, Groningen, The Netherlands, August 25-27, 2003, Proceedings*, volume 2756 of *Lecture Notes in Computer Science*, pages 746–753. Springer, 2003. ISBN: 3-540-40730-8.

[5] S. A. Cook. The complexity of theorem-proving procedures. In M. A. Harrison, R. B. Banerji, and J. D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[6] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1367–1372, 2004.

[7] G. Damiand, C. d. l. Higuera, J. Janodet, É. Samuel, and C. Solnon. A polynomial algorithm for submap isomorphism. In A. Torsello, F. Escolano, and L. Brun, editors, *Graph-Based Representations in Pattern Recognition, 7th IAPR-TC-15 International Workshop, GbRPR 2009, Venice, Italy, May 26-28, 2009. Proceedings*, volume 5534 of *Lecture Notes in Computer Science*, pages 102–112. Springer, 2009. ISBN: 978-3-642-02123-7.

[8] B. Delaunay. Sur la sphère vide. A la mémoire de Georges Voronoï. French. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles*:793–800, 6, 1934.

[9] H.-C. Ehrlich and M. Rarey. Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(1):68–79, 2011. ISSN: 1759-0884.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979. ISBN: 0-7167-1044-7.

[11] S. Gay, F. Fages, T. Martinez, S. Soliman, and C. Solnon. On the subgraph epimorphism problem. *Discrete Applied Mathematics*, 162:214–228, 2014.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014.

[13] H. M. Grindley, P. J. Artymiuk, D. W. Rice, and P. Willett. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *Journal of Molecular Biology*, 229(3):707–721, 1993. ISSN: 0022-2836.

[14] R. Hoffmann, C. McCreesh, and C. Reilly. Between subgraph isomorphism and maximum common subgraph. In S. P. Singh and S. Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* Pages 3907–3914. AAAI Press, 2017.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.* Pages 1106–1114, 2012.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[17] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[18] C. McCreesh, S. N. Ndiaye, P. Prosser, and C. Solnon. Clique and constraint models for maximum common (connected) subgraph problems. In M. Rueher, editor, *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 350–368. Springer, 2016. ISBN: 978-3-319-44952-4.

[19] C. McCreesh and P. Prosser. A parallel, backjumping subgraph isomorphism algorithm using supplemental graphs. In G. Pesant, editor, *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, volume 9255 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2015. ISBN: 978-3-319-23218-8.

[20] C. McCreesh, P. Prosser, and J. Trimble. Heuristics and really hard instances for subgraph isomorphism problems. In S. Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 631–638. IJCAI/AAAI Press, 2016. ISBN: 978-1-57735-770-4.

[21] J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16(7):521–533, 2002.

[22] U. Redmond and P. Cunningham. Subgraph isomorphism in temporal networks. *CoRR*, abs/1605.02174, 2016.

[23] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput.*, 27(7):950–959, 2009.

[24] L. Rothacker and G. A. Fink. Robust output modeling in bag-of-features hmms for handwriting recognition. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 199–204. IEEE Computer Society, 2016. ISBN: 978-1-5090-0981-7.

[25] L. Rothacker, S. Vajda, and G. A. Fink. Bag-of-features representations for offline handwriting recognition applied to arabic script. In *2012 International Conference on Frontiers in Handwriting Recognition, ICFHR 2012, Bari, Italy, September 18-20, 2012*, pages 149–154. IEEE, 2012. ISBN: 978-1-4673-2262-1.

[26] É. Samuel, C. d. l. Higuera, and J. Janodet. Extracting plane graphs from images. In E. R. Hancock, R. C. Wilson, T. Windeatt, I. Ulusoy, and F. Escolano, editors, *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, SSPR&SPR 2010, Cesme, Izmir, Turkey, August 18-20, 2010. Proceedings*, volume 6218 of *Lecture Notes in Computer Science*, pages 233–243. Springer, 2010. ISBN: 978-3-642-14979-5.

[27] D. A. Sinclair. S-hull: a fast radial sweep-hull routine for delaunay triangulation. July 2010. URL: `http://www.s-hull.org/paper/s_hull.pdf` (visited on 03/27/2017).

[28] C. Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artif. Intell.*, 174(12-13):850–864, 2010.

[29] M. Stauffer, A. Fischer, and K. Riesen. A novel graph database for handwritten word images. In A. Robles-Kelly, M. Loog, B. Biggio, F. Escolano, and R. C. Wilson, editors, *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2016, Mérida, Mexico, November 29 - December 2, 2016, Proceedings*, volume 10029 of *Lecture Notes in Computer Science*, pages 553–563, 2016. ISBN: 978-3-319-49054-0.

[30] S. Sudholt and G. A. Fink. Phocnet: A deep convolutional neural network for word spotting in handwritten documents. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 277–282. IEEE Computer Society, 2016. ISBN: 978-1-5090-0981-7.

[31] Z. Sun, L. Jin, Z. Xie, Z. Feng, and S. Zhang. Convolutional multi-directional recurrent network for offline handwritten text recognition. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 240–245. IEEE Computer Society, 2016. ISBN: 978-1-5090-0981-7.

[32] M. Thoma. Hwrt database of handwritten symbols, Jan. 2015. DOI: `10.5281/zenodo.50022`.

[33] J. Trimble, C. McCreesh, and P. Prosser. A partitioning algorithm for maximum common subgraph problems. Unpublished, Feb. 2017.

[34] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. CNN: single-label to multi-label. *CoRR*, abs/1406.5726, 2014.

[35] E. Welzl, P. Su, and R. L. S. D. III. A comparison of sequential delaunay triangulation algorithms. *Comput. Geom.*, 7:361–385, 1997.

[36] J. Wu, Z. Cui, V. S. Sheng, P. Zhao, D. Su, and S. Gong. A comparative study of sift and its variants. *Measurement Science Review*, 13(3):122–131, 2013.