

# Mining Historical Documents for Near-Duplicate Figures

Thanawin Rakthanmanon    Qiang Zhu    Eamonn J. Keogh

Department of Computer Science and Engineering

University of California, Riverside

{rakthant, qzhu, eamonn}@cs.ucr.edu

**Abstract**—The increasing interest in archiving all of humankind’s cultural artifacts has resulted in the digitization of millions of books, and soon a significant fraction of the world’s books will be online. Most of the data in historical manuscripts is text, but there is also a significant fraction devoted to images. This fact has driven much of the recent increase in interest in query-by-content systems for images. While querying/indexing systems can undoubtedly be useful, we believe that the historical manuscript domain is finally ripe for true *unsupervised* discovery of patterns and regularities. To this end, we introduce an efficient and scalable system which can detect approximately repeated occurrences of shape patterns both within and between historical texts. We show that this ability to find repeated shapes allows automatic annotation of manuscripts, and allows users to trace the evolution of ideas. We demonstrate our ideas on datasets of scientific and cultural manuscripts dating back to the fourteenth century.

**Keywords**—component; cultural artifacts; duplication detection; repeated patterns

## I. INTRODUCTION

The world’s books and manuscripts are being digitized at an increasing rate, and within a few years the majority of the world’s books will be online. Much of the data will be text, most of which is more or less amiable to optical character recognition. However, in addition, there will be perhaps hundreds of millions of pages that contain one or more images. It is clear that these images will be very difficult to process. Indeed, data mining of *modern* photograph images is challenging, and in the case of images from historical manuscripts the challenges are compounded by the problems of fading, staining, wear, insect damage, abrasions, foxing, pencil annotations, and distortion artifacts from the digitization process, etc. [18][19].

In spite of these challenges, it is clear that the wealth of figures from historical manuscripts offer unique possibilities for data mining of important cultural artifacts. While the completely automated extraction of data from these texts will remain a significant challenge for some time to come, in this work, we introduce a specialized sub-routine that *is* achievable and useful. This sub-routine is the automatic discovery of approximately duplicated figures, both *within* and *between* texts.

Our ideas can best be explained with a simple motivating example. In the early part of the 19th century, relatively inexpensive high-powered microscopes became available for the first time. This initiated an explosion of interest in Diatoms, a major group of eukaryotic algae, whose extraordinary shapes delighted and puzzled Victorian naturalists. Consider the two plates shown in Figure 1. They

are typical examples from the perhaps hundreds of books on Diatoms published during the Victorian era [17][22][25].

Thanks to efforts by digital archivists, hundreds of these works, representing over one million individual shapes, have been digitized and placed online. Some of them are scholarly classics, such as W. & G.S. West: *A Monograph of the British Desmidiaceae*[25], which is still referenced in modern scientific texts, and some of them are vanity publications by “gentlemen scholars”.

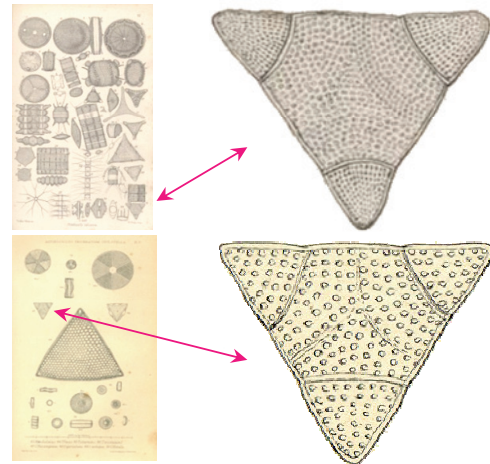


Figure 1. *left*) Two plates from 19th-century texts on Diatoms. Plate 6 of [17] and plate 5 of [22]. *right*) A zoom-in of the same species, *Biddulphia alternans* appearing in both texts.

If we were hosting archives of Diatom images, we might wish to add mutual hyperlinks between each occurrence of *Biddulphia alternans*, since any researcher with an interest in one, will surely have an interest in the other. Here we show just one pair of shapes deserving of a mutual hyperlink; however, in the domain of Diatoms, there are at least a thousand species defined by a unique shape, which could have all their occurrences linked together.

Figure 2 shows another example where linking between two manuscripts is helpful (figure best viewed in color).

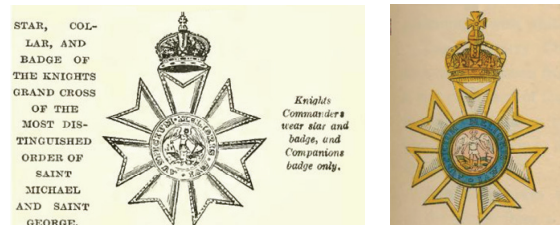


Figure 2. *left*) A figure from page 7 of [7], a 1915 text on peerage. The original text is monochrome. *right*) A figure from page 109 of [3], an 1858 text on honors and decorations.

The 1915 text (*left*) has much greater information and annotations about a medal, but the image is B/W. In contrast,

the older text (*right*) clearly shows that the circular ring is *blue*. In this case, the *combination* of these two figures contains more information than either one on its own.

There are many other examples which demonstrate that linking two shapes within or between books can be of help to historians, genealogists or scientists. However, to the best of our knowledge, this problem has never been addressed. In this work we demonstrate a technique to allow automatic discovery of repeated shapes in historical manuscripts. Beyond the obvious image processing challenges, and the problem of defining an appropriately robust distance measure, the biggest challenge is clearly *scalability*. Even if we have *only* 100,000 shapes in a collection, a brute force “all-to-all” algorithm would require approximately five billion distance calculations, an untenable proposition. Our algorithm is inspired by motif-discovery in bioinformatics [24], as DNA motifs can be considered as near-duplicate *strings*. As we shall show, it can (significantly) adapt bioinformatics algorithms to discover duplicated *shapes* in time linear to the number of “black” pixels in the text.

The rest of this paper is organized as follows. In Section II we introduce all necessary notations and definitions. We introduce an exact, but untenably slow algorithm in Section III. Then, in Section IV we show a very fast approximate algorithm that exploits novel observations and ideas from bioinformatics and image processing. Section V sees a detailed empirical study on real data. We offer conclusions and directions for future work in Section VI.

## II. BACKGROUND AND NOTATION

We begin by introducing all necessary notations and definitions. These notations are illustrated in Figure 5.

### A. Definitions and Notation

Whether we are mining archives of postcards, books, maps, etc., we can see our data source as bitmap *documents*:

**Definition 1:** A *document*,  $D$ , is a matrix with ternary values which are 1, 0, and -1. For any pixel in the given document that is black or white, we will set the corresponding point to be 1 or 0, respectively. The value -1 is reserved for the null or the area outside the original document. A document size  $n$  pixel by  $m$  pixel will be kept by using a matrix of ternary values size  $n \times m$ .

The historical documents may originally be B/W or color. For our purposes we are only interested in shape, so as Definition 1 hints, we binarize all images. The binarization of images in the context of historical manuscripts is a well-studied problem (see [10][12] and references therein) and a relatively easy task for most documents. Nevertheless, we clearly cannot guarantee perfect automatic binarization of large unstructured collections of manuscripts. As we shall show in Section V, our solution to this problem is to use a distance measure and an algorithm that is robust to large amounts of noise and distortions.

Historical documents are often represented by a single surviving instance and many of them have imperfections. As shown in Figure 3, the corners may be burned or worn, or they may have holes due to insect damage [19]. We can support such documents by using null values for the area outside of the document but within the minimum rectangular

boundary of document. Most professional scanners of historical manuscripts use a background color or texture that makes the occurrence of “holes” obvious.

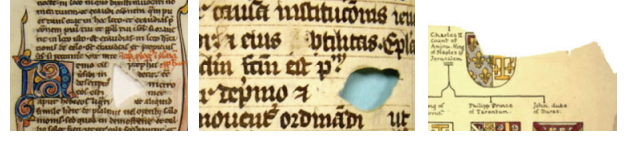


Figure 3. Examples of texts with “holes”.

From the definition above, we regard documents as containing only a single “page.” To apply our algorithm to many pages of a book or many books, we can simply concatenate each page into a long logical document and use a line of null values to separate one page from the others.

We do not expect (or want) to find globally repeated *pages*, so we confine our attention to small regions of interest within a page; these we call *windows*:

**Definition 2:** A *window* is a rectangular area inside a document whose size is specified by the user. It is defined by

$$W_{x,y} = \{d_{i,j} \in D \mid i \in [x, x + s_x) \wedge j \in [y, y + s_y)\},$$

where  $s_x$  and  $s_y$  are a user-defined width and height.

The data inside the window is a ternary value just like the data in the document. While a document may contain many pages, no window is allowed to span two pages. For the rest of this paper, we use the term  $W_a$  interchangeably with the term  $W_{x,y}$  if the starting position of window  $W_a$  is  $(x,y)$ , i.e., in Figure 5,  $W_a = W_{3,2}$ .

There are many ways to measure the similarity between given images. However, most techniques make assumptions about the data. For example, geometric hashing [26] assumes the figures have well-defined points of interest, such as intersections, end-points, areas of maximum curvature, etc. However, even if these assumptions are true, this leads to the non-trivial sub-problem of locating the points of interest. This may be very difficult in our domain of interest. Other distance measures assume that the shapes are fully connected [1], or form closed contours [13]. However, as we shall see, neither assumption generally holds in our domain of interest. SIFT and its variants (G-RIFT, SURF) make less assumptions, but even after tweaking their many parameters, they did not perform well on the problems in Section V.A (to be fair, they are not designed for this domain), so we omit them from further consideration in this work.

Given the above, we need to use a distance measure which is robust to the inevitable noise/distortions we will encounter, and which is general enough to work without any explicit assumptions about the data. Furthermore, as we shall see later, our basic idea to speed up the discovery of repeated figures is to use a “hashing-like” idea from bioinformatics; thus, we need a distance measure that is amenable to hashing.

The recently introduced GHT distance [2][28] is just such a measure. We will explain this in more detail in Section II.B and Section V.A. In the meantime, we use it to define the *distance* between any pair of windows as the following:

**Definition 3:** The *distance* between window  $W_a$  and  $W_b$  is defined as  $\text{dist}(W_a, W_b)$ . We use the GHT distance as a

similarity distance between two windows. Thus, the distance is

$$\text{dist}(W_a, W_b) = \text{GHT}(W_a, W_b).$$

As we can see in Figure 4, our distance measure is offset-invariant. As we shall see, this simple fact allows us to greatly speed up our search algorithm (especially for texts with a lot of “white space”) by significantly reducing the number of distance comparisons needed. We expand on this idea in Section IV.A.



Figure 4. The distance measure we use is offset-invariant, so the distance between any pair of windows, *left*, *center* or *right* above, is exactly zero. This simple fact can be exploited to greatly reduce the search space of motif discovery. Since a pattern from another book that matches one of the above with a distance  $X$  must match all with distance  $X$ , we only need to include any one of the above in our search.

Recall that our task reduces to finding the most similar pair of windows in the document. However, a pathological solution to this would be to have two windows with high overlap, as in the windows  $W_c$  and  $W_d$ , shown in Figure 5.

We call a pair of windows a *trivial match* if its windows have a high degree of overlap.

**Definition 4:** A pair of windows  $\{W_a, W_b\}$  of size  $s_x \times s_y$  is a *trivial match* if its windows are overlapped more than  $\alpha$  times the total area of a window ( $0 \leq \alpha < 1$ ), formally,

$$(s_x - |b_x - a_x|) * (s_y - |b_y - a_y|) \geq \alpha * s_x * s_y \text{ and } (s_x - |b_x - a_x|) \geq 0$$

where  $(a_x, a_y)$  and  $(b_x, b_y)$  are the starting positions of windows  $W_a$  and  $W_b$ , respectively. If  $\{W_a, W_b\}$  is not a trivial match, we call it a *non-trivial match*.

For the special case when  $\alpha=0$ , no overlap between two windows of motifs is allowed; in this case, if windows  $W_a$  and  $W_b$  share one pixels or more,  $\{W_p, W_q\}$  will be a trivial match by the definition. In Figure 5, if we set  $\alpha=0.5$ , both  $\{W_c, W_f\}$  and  $\{W_c, W_d\}$  are trivial matches but  $\{W_a, W_b\}$  is a non-trivial match.

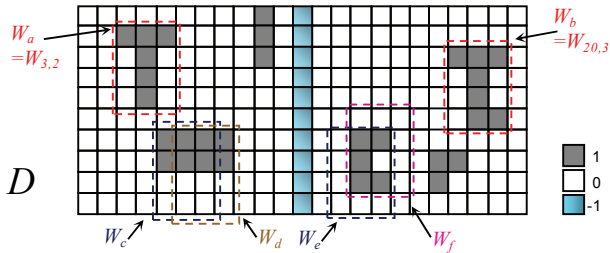


Figure 5. An illustration of our notation. Here the document  $D$  consists of two pages, separated by null values. Intuitively we expect the “T” shape in window  $W_a$  to match the shape shown in  $W_b$ . However, note that the trivial matching pair of  $W_c$  and  $W_d$  (also pair  $W_e$  and  $W_f$ ) are actually more similar, and need to be excluded to prevent pathological results.

Among all possible pairs of windows, we want to find the pair that has the smallest distance between each other and is a non-trivial match. We call this pair the *motif window*:

**Definition 5:** A *motif window* (or just *motif*) is a non-trivial pair of windows  $\{W_a, W_b\}$  such that the distance between

windows  $W_a$  and  $W_b$  is the smallest of all other possible pairs.

$$\text{motif} = \{ \{W_a, W_b\} \mid \min_{W_a, W_b} \text{dist}(W_a, W_b) \}$$

The definitions above assume that we are looking for exactly one near-duplicated figure. However, we can easily generalize this to allow the discovery of multiple motifs. In order to do so we must eliminate some pathological solutions, as shown in Figure 6.

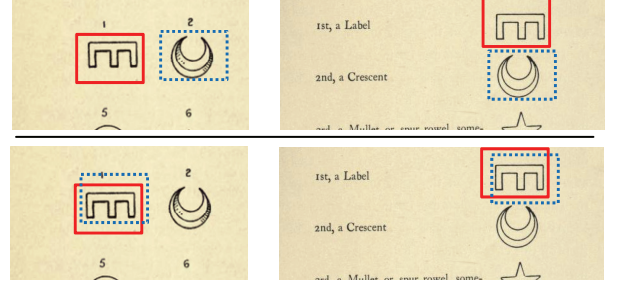


Figure 6. An illustration of a pathological solution to finding the top two motif pairs between two century-old texts. *top*) The desirable solution finds the crescent and label (rotated “E”). *bottom*) A redundant and undesirable solution that we must explicitly exclude is finding one pattern (the label) twice.

To avoid redundant solutions in discovering multiple motifs, we will explicitly exclude *insignificant* motifs from our solution. We define an *insignificant* motif as the following:

**Definition 6:** A motif  $\{W_a, W_b\}$  is *insignificant* if another motif  $\{W_c, W_d\}$  exists such that

- i.  $\text{dist}(W_a, W_b) \geq \text{dist}(W_c, W_d)$
- ii.  $\{W_a, W_c\}$  and/or  $\{W_b, W_d\}$  are trivial matches.

The motif windows are insignificant if at least one of their windows shares a large part with other motifs. However, different motifs can share small parts with each other. Next, we define a *top-k motif window* as the following:

**Definition 7:** A *top-k motif window* is the set of  $k$  most similar pairs of windows, none of which is insignificant.

In Figure 6.*bottom*, only one true motif window is discovered, and the other one is insignificant.

## B. Generalized Hough Transform

The Hough Transform [11] was introduced by Hough as a tool for finding well-defined geometric shapes (lines, curves, rectangles, etc.) in images [8]. The idea was generalized by many others, including Ballard, who introduced the Generalized Hough Transform to detect arbitrary shapes in images [2]. Computing the GHT distance between pairs of windows is relatively expensive. In particular, the time complexity for each GHT calculation is  $O(n_b^2)$ , where  $n_b$  is the number of black pixels in the window.

However, a recent paper by Zhu et al. shows some computational tricks to reduce the amortized time for a *single* comparison, when a higher level algorithm requires *multiple* comparisons (i.e. clustering or query-by-content) [28]. In this work we use the ideas presented by Zhu, but as we shall see, they alone are not sufficient to provide the scalability we require in this domain.



There are two reasons why we chose to use the GHT distance for the problem at hand. Firstly, as shown by Zhu et al., and confirmed by our experiments, the measure is very robust and accurate [28]. Secondly, as we shall see, the method lends itself to being adapted to the random projection framework which is used to solve the motif discovery problem in bioinformatics [24].

### III. EXACT ALGORITHM TO FIND MOTIFS

Given a document  $D$  and (user defined) window size  $s$ , we want to find the top- $k$  motifs in a given document. For simplicity, in the rest of this paper we will explain only how to find the top-1 motif because the extension to top- $k$  is trivial.

#### A. Brute Force Algorithm

We can easily find the top-1 window motif by comparing the distances from all pairs of windows, as shown in Table I.

TABLE I. BRUTE FORCE ALGORITHM

| <b>Algorithm:</b> Brute force algorithm to find the top-1 window motif |  |
|--|--|
| <b>Input:</b>  | $D$ : document<br>$s_x, s_y$ : window size   |
| <b>Output:</b>   | $motif$ : window motif   |
| 1  | $W = \text{set of all windows of size } s_x \times s_y \text{ in } D$                    |
| 2  | $bsf = \infty$   |
| 3  | <b>for</b> $a = 1$ <b>to</b> $ W $   |
| 4  | <b>for</b> $b = a+1$ <b>to</b> $ W $   |
| 5  | <b>if</b> ( $\sim \text{IsTrivial}(W_a, W_b)$ ) $\&\&$ ( $\text{dist}(W_a, W_b) < bsf$ ) |
| 6  | $bsf = \text{dist}(W_a, W_b)$  |
| 7  | $motif = (W_a, W_b)$   |
| 8  | <b>end</b>   |
| 9  | <b>end</b>   |
| 10   | <b>end</b>   |

This simple algorithm uses nested loops (lines 3 and 4) to test all possible pairings of motif windows, checking whether or not two particular windows are trivial (cf. Definition 4), recording the one with the smallest distance. Unfortunately, this algorithm has an obvious flaw which makes it untenable for real problems: it will simply take a great deal of time even for a small document.

Assume that the document is size  $n \times n$  and the user-defined window is size  $s \times s$ . The brute force algorithm must consider every pair of windows, requiring it to compute GHT distances  $O(n^4)$  times. Each GHT calculation takes time  $O(n_b^2)$ , and  $n_b$ , the number of black pixels in a window, can be as large as  $s^2$ ; it is usually larger than  $s$ . Hence, the total running time for the brute force is  $O(s^2 n^4)$ .

To give concrete numbers, suppose the original document is a b/w image of size 1 megabyte, or  $1000 \times 1000$  pixel<sup>2</sup>. The set of all windows,  $W$ , is approximately  $10^6$  (line 1). To find a motif using the brute force algorithm, we need to compute the GHT distances about  $5 \times 10^{11}$  times (line 3-4). This would take approximately  $10^8$  seconds, or 3 years. Because the brute force algorithm cannot find motifs even in a small document in an acceptable amount of time, we will introduce a fast approximate algorithm for this task.

### IV. OUR ALGORITHM

In this section we introduce a sub-linear time motif discovery algorithm. We begin by giving the intuition behind three ideas that we will exploit to make our algorithm

scalable. Later we will show a concrete algorithm that exploits these ideas.

#### A. Intuitions behind our Algorithm

##### 1) Downsampling helps scalability

Our first observation was originally made to help improve scalability, but as a happy side effect it also *greatly* improves accuracy. Figure 7 shows the effect of downsampling on our data of interest.

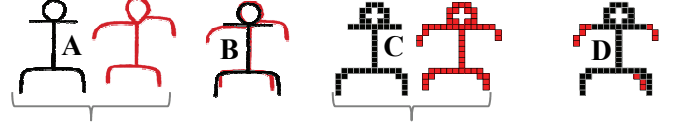


Figure 7. A) Two figures from table 16 of a 1907 text on Native American rock art [14] (one image recolored red for clarity). B) No matter how we shift these two figures, no more than 16% of their pixels overlap. C) Downsampling versions of the figures share 87.2% of their pixels (D).

Because downsampling will greatly decrease the number of windows that must be examined, it will clearly improve efficiency. It is natural to ask if this reduction in resolution will reduce the accuracy. The surprising answer is that the opposite is true; downsampling (except when taken to the extreme) actually improves accuracy by eliminating spurious precision and reducing the shape to its bare minimum. We note that we are not claiming this observation as an original contribution; Zhu et al. pointed this out and demonstrated it with detailed experiments [28]. However, the next two ideas are original and unique to our domain.

##### 2) Random projection further reveals similarities

While the downsampling idea introduced in the last section reduces both the time for a *single* distance calculation and the *number* of distance calculations that must be performed, the number of distance calculations required by the brute force algorithm is still on the order of  $O(n^4)$ . We have just drastically reduced the value of “ $n$ ”. In order to make significant progress on this problem, we need a much faster way to identify (potentially) very similar shapes. Figure 8 shows the intuition as to how we might achieve this.

Assume we have a pair of windows  $\{W_a, W_b\}$  of size  $17 \times 14$ , containing two similar, but not identical figures, whose distance is equal to nineteen (i.e.,  $\text{dist}(W_a, W_b) = 19$ ). For example, in Figure 8 we have two anthropomorphic examples of rock art with this property. Suppose that we randomly choose a single location,  $x = \text{randint}(1:17)$  and  $y = \text{randint}(1:14)$ , and set that pixel to white in both figures. What effect would this have on the distance? There are only two possibilities:

- i) The corresponding pixels in the two windows are already either both white or both black. In either case the distance does not change.
- ii) Exactly one of the corresponding pixels was black, and changing it to white must *decrease* the distance.

From this analysis we can see that “deleting” black pixels (*randomly projecting* windows to a lower dimensional space [24]) must decrease or hold steady the distance between two objects. This is important, because if we manage to decrease the distance to zero, we can find such

zero distance pairs in only linear (in the number of windows) time using *hashing*. This idea is inspired from the well-known hashing technique, min-Hash [4].

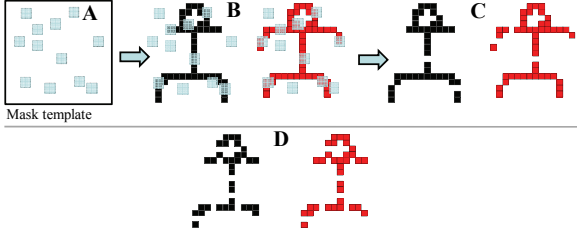


Figure 8. A) If we randomly choose some locations (masks) on the underlying bitmap grid on which the two figures (B) shown in Figure 7 lie, and then remove those pixels from the figures, then the distance between the edited figures (C) can only stay the same or decrease. Several random attempts at removing  $\frac{1}{4}$  of the pixels in the two figures eventually produced two identical edited figures (D).

In the example shown in Figure 8, ignoring one pixel is clearly not enough to make the two figures identical; we actually need to remove at least 19. Furthermore, we need to remove the *correct* set of 19. A simple combinatorial calculation will convince the reader that this is very unlikely to happen if we choose 19 pixels at random. The obvious solution, to ignore *more* than 19 pixels, say 100, contains a problem. If we ignore too many pixels we will also allow two very different figures to hash together.

To some extent, this is a problem we can live with. Even if two very different figures are projected onto a very low dimensional space where they hash together as a *false positive*, we can later check their distance in the original space. The only danger is that if we have too many false positives, then checking them all may not be much faster than a brute force search.

At first blush the problem may seem insurmountable, because we have the extremely delicate task of making *all* similar things identical, without making (too many) different things identical. Fortunately, there is a solution to a nearly identical problem in DNA motif discovery in bioinformatics that we can leverage off [24]. The idea (informally stated) is to be conservative in the number of pixels we remove, but to do multiple independent rounds of projection (*hashing*). This increases the number of true positives, while also reducing the number of false positives.

### 3) Numerosity reduction improves scalability

The final observation we will make has already been hinted at in Figure 4. Even after downsampling, there are many windows that must be explored in order to find a motif (or top- $k$  motifs). The number of windows of size  $s_x \times s_y$  in a document of size  $n \times m$  is quadratic in terms of the document size, or more precisely, is  $(n-s_x+1)(m-s_y+1)$ , which is  $O(n^2)$  when  $n=m$ . Naturally, all of these windows have a great deal of redundancy with their neighbors, and many windows may be totally blank or contain only a handful of pixels. Based on this observation, we can reduce the number of windows in the document dramatically by filtering out all but one representative example of a set of redundant windows, and also filter out windows that do not have enough black pixels to form any meaningful shape. We call the remaining

windows in the document after this process, *potential windows*:

**Definition 8:** A *potential window* is a window whose number of black pixels is at least a threshold  $t$  and not less than other adjacent windows. Then, the set of potential windows  $P$  is defined as

$$P = \{ W_{x,y} \mid \forall \beta, \gamma \in \{-1, 0, 1\} \text{ } sum(W_{x,y}) \geq sum(W_{x+\beta, y+\gamma}) \wedge sum(W_{x,y}) \geq t \}.$$

Let  $sum(\cdot)$  be the total number of black pixels in the particular window;  $sum(W_{x,y})$ . In the case of a null value, we can set  $d_{ij}$  to either 0 or 1.

This idea can be visualized in Figure 9, where potential windows are centered on the peaks of a 3D heatmap.

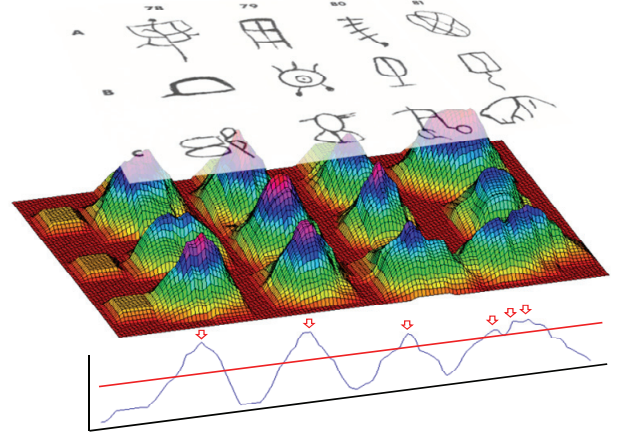


Figure 9. The summation of the number of black pixels in windows. Only windows corresponding to peaks above the threshold (the red line) need to be tested. The arrows show the center position of six potential windows.

We simply set the parameter  $t$  to an average number of black pixels in a specific window size, and the results show that our algorithm works well on this default value. Ties can be resolved by selecting just one potential window, changing the definition from “ $\geq$ ” to “ $>$ ”.

We note that this step also has an analogue in bioinformatics algorithms [22]. Many motif discovery algorithms do a preprocessing step of removing regions of low complexity DNA (for example, a long run of a single amino acid) to both speed up search and eliminate pathological solutions.

### B. Motif Discovery

We are finally in a position to explain our algorithm and how it exploits the three ideas from the last section. In essence, we *downsample* the original book, extract all *potential windows* and hash them with *random projection*. All pairs that collide are inspected in the original space to see if they are true motifs. Our algorithm is described in Table II.

Our algorithm uses four more inputs than the brute force algorithm. The first is the downsampling scale,  $ds$ , and the other three parameters are used in random projection.

In line 1, we downsample the original document  $D$  into the smaller version,  $DD$ , with the scale  $ds$ . While there are many algorithms for rescaling images, we simply downsample by majority voting the values inside  $d_s \times d_s$  pixels

in the original document to create a new pixel in the new document,  $DD$ . Hence,  $DD$  will be smaller than  $D$  by a factor of  $ds^2$ .

The next step is to locate all windows in the new document in line 2. Note that the total search space is reduced from  $O(n^4)$  to  $O(n^4/ds^4)$ . Further note that in our implementation, we do not set  $W$  explicitly. We still need to further reduce the search space, so in line 3 we apply the third idea from the last section. In order to locate the potential windows (cf. Definition 8), for all windows we calculate the total number of black pixels inside that window. We can do this in linear time with respect to the number of pixels in the document. We then locate all potential windows which are at the local maxima of the summation plot, as visualized in Figure 9. Now our search space is massively reduced; for example, in Figure 9, there are less than 30 potential windows among 22,000 original windows.

The number of peaks or potential windows is data dependent. It is possible that there are a lot of small peaks in the document. For example, as in Figure 9, there are 3 potential windows on the right which are created from the same symbol. We solve this problem in line 4 with a simple solution. We align every potential window by moving its center of mass to the center of the window. As a result of this process, potential windows may slightly change their position, so if several windows are aligned to the same position, we pick only one window at the position.

TABLE II. OUR ALGORITHM

| Algorithm: <i>DocMotif</i> |  |
|----------------------------|--|
| <b>Input:</b>              | $D$ : document $s_x s_y$ : window size<br>$ds$ : downsampling scale $it$ : number of iteration<br>$hds$ : hash downsampling scale $mask$ : masking ratio |
| <b>Output:</b>             | $motif$ : window motif   |
| 1                          | $DD = \text{DownSamplingDoc}(D, ds)$   |
| 2                          | $W = \text{set of all windows of size } s_x \times s_y \text{ in } DD$   |
| 3                          | $P = \text{LocatePeaks}(W)$  |
| 4                          | $P = \text{AlignCenter}(P)$  |
| 5                          | $HSig = \emptyset$   |
| 6                          | <b>for</b> $i = 1$ <b>to</b> $it$  |
| 7                          | $hsig = \text{HashSignature}(P, hds, mask)$  |
| 8                          | $HSig = HSig \cup hsig$  |
| 9                          | <b>end</b>   |
| 10                         | $cand = \text{CollidedWindow}(HSig)$   |
| 11                         | $best\text{-}so\text{-}far = \infty$   |
| 12                         | <b>for</b> all pair of windows $(w_a, w_b)$ in $cand$  |
| 13                         | <b>if</b> $(\text{IsTrivial}(w_a, w_b))$   |
| 14                         | <b>continue</b> ;  |
| 15                         | <b>end</b>   |
| 16                         | <b>if</b> $(lb\_dist(w_a, w_b) < best\text{-}so\text{-}far)$   |
| 17                         | <b>if</b> $(dist(w_a, w_b) < best\text{-}so\text{-}far)$   |
| 18                         | $best\text{-}so\text{-}far = dist(w_a, w_b)$   |
| 19                         | $motif = (w_a, w_b)$   |
| 20                         | <b>end</b>   |
| 21                         | <b>end</b>   |
| 22                         | <b>end</b>   |

After all potential windows have been enumerated, we hash them using random projection (line 5-9). This idea is essentially the same as the one shown in Figure 8; we create a hash signature from each figure by randomly removing some pixels from them (line 7). Then, we find all pairs of windows that “collide”; that is, they share the same signature (line 10). To create the hash signature in line 7, we do two steps. First, we *further* downsample all potential windows to a smaller size. The parameter controlling this is called  $hds$ ,

or hash downsampling scale. Then, we randomly remove some pixels, the number of which is controlled by the parameter  $mask$ . We do this random projection  $it$  times to increase the probability that two similar items will collide at least once. Note that only one collision is required to ensure we discover the motif.

In the last step, we create a set of all candidate pairs,  $cand$  (line 10). We can now calculate the true GHT distance of each pair, and the motif is a pair of windows that has the smallest distance, ignoring as always the trivial pairs (line 13). Lines 12 to 22 are essentially the brute force search (see Table I), but over a relatively tiny subset of the original possibilities. Because the GHT calculation is expensive, we avail of a recently published speed-up trick [28]. We calculate the GHT’s *lower bound* first (line 16). If the lower bound is bigger than the *best-so-far*, we do not calculate the expensive GHT distance. Otherwise, we update the *best-so-far* using the GHT distance (line 18-19).

There is one trivial but very useful modification we can make to the algorithm. We can input *two* books instead of one, and insist (by adding an extra test on line 13) that the motif’s two occurrences have one representative from each book. This is an idea hinted at in the first three figures in this work. We may see this as a *motif join* between two texts.

## V. EXPERIMENTAL RESULTS

We have designed all experiments such that they are reproducible, and as such, all data and code are freely available, see [30].

In this section we wish to empirically demonstrate the following: that the GHT distance measure, operating on downsampled data, is appropriate for our domain; that our algorithm can find meaningful motifs in real data, and that this information is useful to domain experts; and that our algorithm is scalable enough to allow mining of large texts. Finally, we wish to show that although our algorithm has several parameters, these are easy to set, and their exact value is not critical to efficiency or effectiveness.

### A. Sanity Check for the GHT Measure

While our experimental case studies (cf. Section V.B) offer compelling anecdotal evidence that our method finds truly similar figures, it would be useful to see *objective* tests. To our knowledge there is no benchmark dataset to test on; however, we can show the suitability of our ideas for objective tests in very similar domains/problems.

We would like the tests to demonstrate two things: that our choice of the GHT as a similarity measure in this domain is warranted, and that the extreme downsampling we perform to improve scalability does not hurt accuracy. We achieve these aims by testing on datasets of hand-drawn figures. Two of the datasets are from a collection of old music scores (17th-19th centuries) [9][20], and thus are very representative of our domain of interest, and the third one is a modern architectural symbol dataset [15], in which various users hand copied symbols, and is thus also very similar to the task at hand. As Figure 10 shows, these are non-trivial problems. In particular, the symbols in the first two datasets come from degraded texts, written by individuals who may have lived centuries apart and in different countries. Rather

than fine-tune our method, we simply hard coded the downsampling to 20x20 for all datasets.



Figure 10. Samples showing the interclass variability in the hand-drawn datasets. *left*) Samples from the music datasets. *right*) Samples from the architectural dataset.

Table III shows the one-nearest-neighbor leaving-one-out accuracy.

TABLE III. THE ACCURACY OF GHT ON 3 HAND-DRAWN SYMBOL PROBLEMS

|                     | # instances | # classes | Accuracy |
|---------------------|-------------|-----------|----------|
| Clefs               | 2,128       | 3         | 99.58%   |
| Accidentals & Clefs | 4,098       | 7         | 98.49%   |
| Architectural       | 7,414       | 50        | 99.29%   |

While others have worked on these datasets, we did not directly compare our results to theirs. The published approaches on these datasets are so slow (an  $O(n^3)$  warping method for the music symbols [9][20], and an  $O(n^3)$  adjacency grammar method for the architectural symbols [15]), that in both cases the authors abandoned any attempt at a full leaving-one-out on the entire dataset, and instead created various smaller subsets (hand crafted and thus difficult to meaningfully compare to). However, our accuracies are so close to perfect in every case that our claim is clearly demonstrated: the GHT on downsampled images is an effective distance measure for these kinds of images.

### B. Motifs between Two Manuscripts

While there is undeniable utility in discovering motifs *within* a single text, the real power of motif discovery will undoubtedly come from the linking of two motifs *between* two or more apparently disparate texts.

Taryn Rampley, a Ph.D. student in anthropology at the University of California-Riverside, is interested in correlating DNA studies of peoples from the Americas with studies of cultural artifacts [21]. In particular, she is looking for evidence of cultural transmission from North America to South America prior to contact with Europeans. While this evidence might be found through jewelry, textiles, weapons or language, this researcher is focusing on petroglyphs (rock art), of which there are several million documented examples in the Americas.

This student gave us a classic reference text on Californian petroglyphs [23], which includes a 104 page petroglyph catalog, containing about 2,852 individual examples of petroglyphs. We scanned this text with an off-the-shelf scanner. Figure 11.*left* shows two representative pages.

Thanks to the Google Book Project, the web is replete with possible texts with which to compare. One such text that caught our attention is a 1907 text by the German ethnologist and explorer Theodor Koch-Grünberg (1872–1924) which discusses the origin and significance of rock art in South America [14]. This text contains 233 images of petroglyphs hand-traced by the author. Figure 11.*right* shows two representative pages.

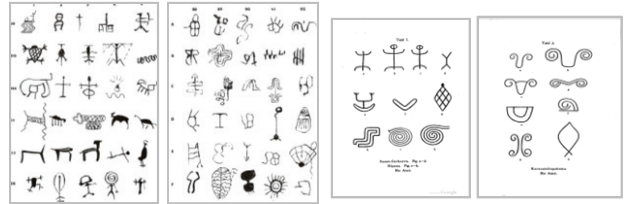


Figure 11. *left*) Two typical pages from Californian petroglyphs [23]. *right*) Two typical pages from [14]. Note that the minor artifacts are from the original Google scanning.

We ran our motif join algorithm on these two texts; Figure 12 shows a selection of the top fifty results.



Figure 12. Five random motif pairs from the top fifty pairs created by joining the two texts [14] and [23]. Note that these results suggest that our algorithm is robust to line thickness, solid vs. hollow shapes, and various other distortions.

While the figure pairs are clearly *somewhat* similar, the anthropologist does not feel that this provides evidence of cultural transmission. If we repeat the experiment by comparing the reference text to petroglyphs from Arizona or Utah, the joins are much more similar. Currently, these conclusions are subjective and tentative; in ongoing work we are working with anthropologists to produce a principled theoretical framework for drawing such conclusions. While we defer detailed scalability results to the next section, we note that this join took approximately one minute.

Before moving on, it is worth re-examining Figure 12 to note the invariances our algorithm has achieved. For example, in the Figure 12.*middle* our algorithm discovered a pair of anthropomorphic figures in spite of the fact that one has a solid head and antenna. To appreciate *why* we can achieve such invariances we invite the reader to review Figure 7 and Figure 8, whose examples we drew from one of these texts [14].

Such robustness is critical if we are to investigate hand drawn texts in addition to the printed texts we consider next. As part of another project on mining cultural artifacts we are also interested in mining the vast literature on genealogy and heraldry that dates back to the 12<sup>th</sup> century [5][6]. Figure 13 shows a typical result in this domain.

In order to make a point about some invariances our distance measure achieved in this domain, Figure 14 shows a zoom-in of the two pairs of discovered motifs shown in Figure 13. Note that in both cases the two members of each motif differ slightly in scale. This is presumably due to differences in the scanning process, since it is likely that the images were produced by the intaglio process, and printed from the same plate. In any case, our method is robust to such minor scale changes.



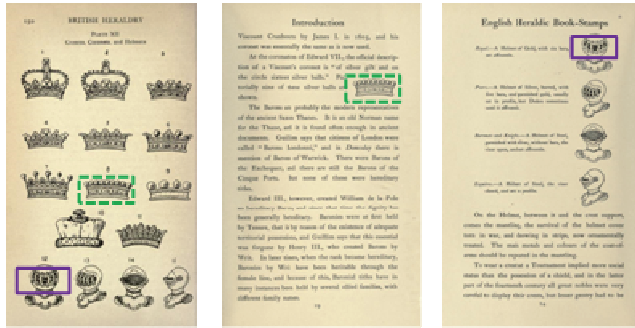


Figure 13. The top two inter-book motifs discovered when linking a 1921 text, British Heraldry [5] (left), with a 1909 text, English Heraldic Book-Stamps, Figured and Described [6] (center), and (right).

Note also that the figures are not identical; for example, the helmet in the later text has additional shading on the right side of the dome and under the chin. Again, our method is robust to this issue.

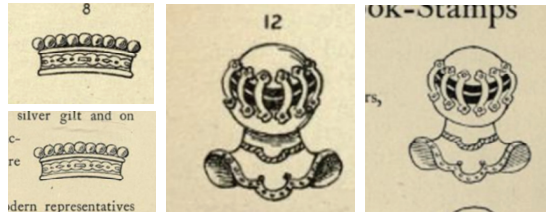


Figure 14. A zoom-in of the motifs discovered in Figure 13. Note that the two helmets differ in size by about 11%, and our algorithm was invariant to this difference.

However, the most interesting point about this example is the (relative) invariance to the user-specified size parameter. Note that as shown in Figure 14, we set a window size and aspect ratio that happens to be perfect to enclose the crown. To enclose the helmet, we really need a window size that is about twice as large and with a more vertical aspect ratio. Nevertheless, in spite of a suboptimal window size we still found the helmet motif. This is not a one-off fortuitous occurrence, but generally true (see additional examples at [30]). So long as the user-supplied size is within a factor of two or so of the motif size, we will robustly find it. If the uncertainty in size is greater than a factor of two, our algorithm is efficient enough to allow range-doubling search.

In the next section we show that the efficiency and accuracy of our algorithm are largely invariant to parameter choice.

### C. Scalability and Noise Tolerance

Testing the scalability of our approach on real data provides us with significant challenges, since the running time of our algorithm depends on the data. For example, suppose a book has a *perfect* motif on pages 1 and 51, but otherwise there are no significant repeated patterns. The time to search a subset consisting of the first 50 pages would be much *greater* than the time taken to search the first 100 pages, since the latter would encounter a high quality *best-so-far* early on. Given this, we test the scalability on an artificial book over which we have perfect control. We made every effort to make a realistic book, but when in doubt we made choices designed to strain our algorithm.

We generated an artificial book using the idea of a 14-segment display that be used to create any English alphabetical character or digit. Figure 15.*left* shows some samples. In our artificial book, each page contains a random selection of 100 characters and the size of each page is 1330x1220 pixels, as shown in Figure 15.*middle*. While it is very unlikely that any random character would be created twice, such an occurrence would greatly favor our algorithm. We therefore further distort the book by two methods: adding a random polynomial warping (modeling a distortion caused by non-contact scanning) to the pages and adding some Gaussian noise, as shown in Figure 15.*right*.

In order to set the parameters for our experiments, we did the following: we created a two-page “book” and spent less than five minutes “playing” with it to find reasonable parameter values. Once we had found these values, we fixed them for *all* data sizes up to 2,048 pages.

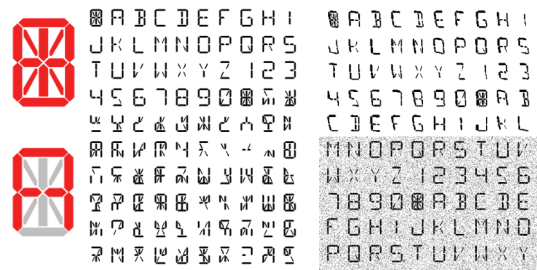


Figure 15. *left*) The 14-segment template used to create characters. We can turn on/off each segment independently to generate a vast alphabet. *middle*) An example of a page which is generated from the process. *right*) A page of the book after adding polynomial distortion (*top half*), and Gaussian noise with mean 0 and variance 0.10 (*bottom half*).

As we can see in Figure 16, our algorithm can find the top motif in a 128-page book in a minute and in a 2,048-page book in half an hour. Note that these times are close to the time taken to scan (at least valuable) books of this size, so they are not unreasonable.

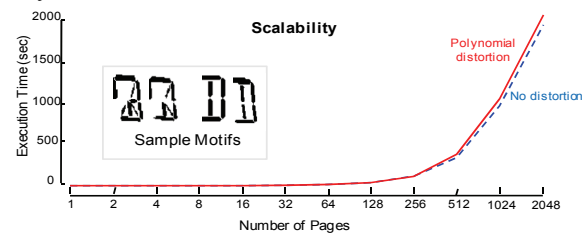


Figure 16. Time to discover motifs in books of increasing size. Our algorithm can find a motif in 512 pages in 5.5 minutes and 2048 pages in 33 minutes. (*inset*) As a sanity check we confirmed that the discovered motifs are plausible, as here (noise removed for clarity).

Note that in this figure and some figures to follow, some lines are difficult to tell apart; however, *this is the point* of these experiments: to show that our algorithm is not sensitive to distortions/noise/parameter choices.

We also test the noise tolerance of our algorithm by generating an artificial book with Gaussian noise added. The mean of the Gaussian noise is set to 0 and its variance is varied from 0 to 0.20. The results in Figure 17 show that our algorithm can tolerate significant noise ( $var=0.15$ ).

When the book contains too much noise ( $var=0.20$ ), the number of potential windows will increase because it is difficult to align all potential windows from a figure into the



same position. Hence, the running time increases. However, this case corresponds to a *very* heavily degraded image.

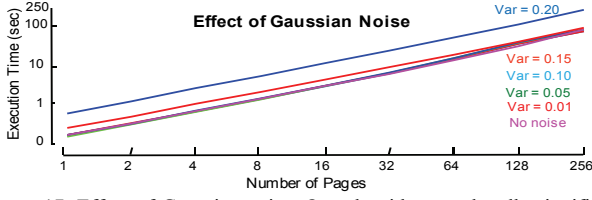


Figure 17. Effect of Gaussian noise. Our algorithm can handle significant amounts of noise. An example of a page containing noise at  $\text{var}=0.10$  is shown in Figure 15.right.

To concretely ground the amount of speed-up our algorithm can achieve we did the following experiment. On a 512-page book, we compared the running times of:

1. Exact motif search over the entire document by applying motif discovery technique in [16]
2. Exact motif search over just the potential windows
3. Our proposed algorithm, *DocMotif*.

The results are shown in Figure 18. We can see that the running time of heuristic search from [16], which is *much* faster than brute force search, rapidly becomes untenable, taking, for example, more than a day for just 8 pages and (an estimated) six months to finish all 512 pages. Our simple trick of only searching over potential windows reduces the search time to just 6.9 hours for the full 512 pages; however, our proposed algorithms take a mere 342.4 seconds.

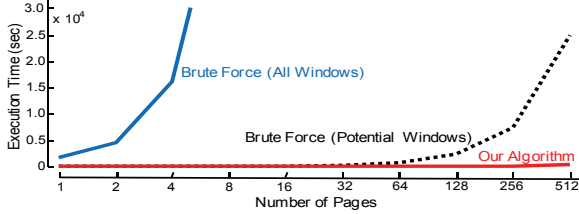


Figure 18. The total execution time of three search algorithms: an exact motif search, an exact motif search on just the potential windows, and our algorithm *DocMotif*.

#### D. Robustness of Parameters

We have an obligation to explain how the choice of parameters affects the speed of motif discovery and the quality of motifs. As we shall see, our algorithm is *not* particularly sensitive to parameter choice. Recall that in the previous sections we have set the parameters based on a few minutes' experience with a two-page sample. Our simple test for parameter sensitivity is to hold three parameters firm, and adjust the other parameters to higher and lower values, to see what effect this has. Figure 19 tells us that for the most part, the algorithm's performance does not rely critically on parameter choices. Of course, this dramatic speed-up would be worthless if the faster algorithms produced inferior results. However, as we shall show *empirically* in Section V.B, the results of all algorithms are virtually identical.

In the random projection process, the length of the hash signature is affected by two parameters, which are the hash downsampling scale  $hds$  and masking ratio  $mask$ . When the signature is shorter, the probability of collisions increases (including false positives that must be checked and dismissed). Thus, when we remove more pixels from a

window by increasing  $mask$ , the running time will increase, as shown in Figure 19.A. As we can easily see in Figure 8, if we continuously remove pixels, eventually all windows will collide to the same shape (with pure white or no content left).

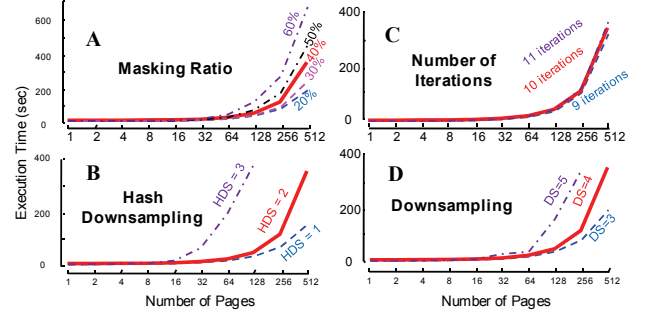


Figure 19. The effect of parameters on our algorithm. We test on artificial books with polynomial distortion and each result is averaged over ten runs. The bold/red line represents the parameters learned from just the first two pages.

Similar to  $mask$  which has a linear effect,  $hds$  has a quadratic effect on the length of the hash signature, so if we change it, the running time may change significantly as shown in Figure 19.B. Recall that  $hds=2$ , meaning that we condense 4 pixels to just one. The number of iterations has very little effect (Figure 19.C), which is also true for the bioinformatics algorithm that inspired us [24].

The downsampling scale  $ds$  parameter (cf. Figure 7) can reduce the search space and increases the quality of the motifs by allowing a greater invariance to noise. Figure 19.D shows that if we fix all other parameters and vary this parameter, the total running time will increase as  $ds$  increases. When  $ds$  increases, the downsampled document, *DD* (cf. Table II), contains fewer pixels and also less information to represent any figure; so after random projection there will be more spurious collisions, increasing the number of false positives that must be checked and thus increasing the running time.

As with any approximate algorithm the quality of the result is important. Hence, we calculate the *quality* of top-20 motifs by using their total distance. Figure 20 shows the average distance for different parameters values, compared to the exact search algorithm.

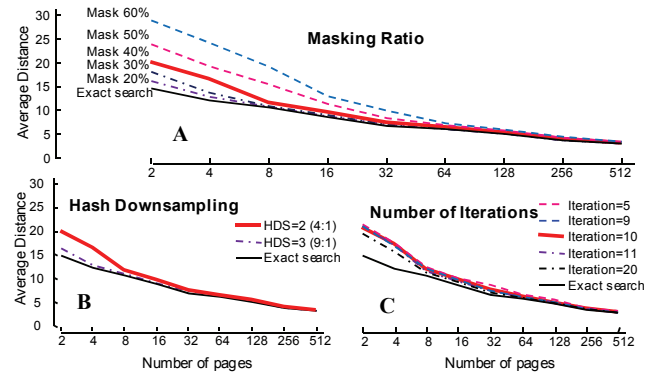


Figure 20. The average distance from top-20 motifs from our algorithm and the exact search algorithm. The bold/red line shows the default parameters. This shows that the quality of motifs is *not* sensitive to different parameter settings and very close to the result from the exact search algorithm.

Here the *quality* of the top twenty motifs is simply the sum of all twenty distances of each motif pair (i.e., Definition 3). As Figure 20 shows, the quality of *DocMotif* is very good under *any* parameter setting, even for small books, but as the size of the book increases, the results are essentially indistinguishable from the exact search, which takes about 67,500 times longer.

### E. Data Mining Palm Leaf Manuscripts

We conclude by noting that our algorithm is currently being evaluated for mining massive (four million leaves) archives of palm leaf manuscripts like the one shown in Figure 21, for medical knowledge.

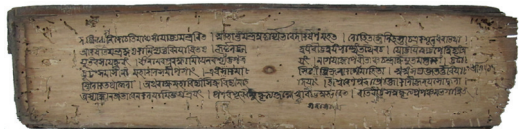


Figure 21: An example of a Palm Leaf Manuscript

Because of space limitations, and because it is an interesting and visual example, we present this example as an one minute long YouTube video [29]. The video demonstrates the speed, robustness and accuracy of our algorithm, even in the face of complex and degraded texts.

## VI. CONCLUSIONS

We have said little about related work thus far because there is little that does exactly what we propose. Xi et al. do consider “*Finding Motifs in a Database of Shapes*” [27]. However, they make two critical assumptions that are not true in our case (or in general): that the individual shapes can be perfectly extracted, and that all shapes can be represented by a closed contour. Of course, our work does borrow heavily from the huge literature on motif discovery in bioinformatics; see [24] and the references thereof. Likewise, we have exploited both the classic ideas of the GHT [2], and the recent extensions by [28]. There is an active community working on computerized historical document analyses [10][12][18]: however, while great many papers address *query-by-content*, including [9][15][20] the task of motif discovery in this domain has not been addressed thus far.

We have shown the first general technique for the unsupervised discovery of repeated patterns both within and between texts. We have demonstrated that our algorithm is scalable, it can produce meaningful results that are useful to domain experts, and its parameters are easy to set. In future work we plan to integrate our ideas with (OCRed) text mining algorithms, and leverage off very recent theoretical results in bioinformatics to remove the need to set any parameters in our algorithm.

## ACKNOWLEDGEMENTS

We would like to acknowledge the financial support for our research provided by the Royal Thai Government and NSF grants 0803410 and 0808770.

## REFERENCES

[1] X. Bai, X. Yang, L. Latecki, and W. Liu, “Learning context sensitive shape similarity by graph transduction,” IEEE TPAMI, 2009.

[2] D. H. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” Pattern Rec 13, 1981, pp. 111-22.

[3] B. J. Burke, Book of Orders of Knighthood and Decorations of Honour of all Nations, London: Hurst and Blackett, 1858.

[4] O. Chum, J. Philbin, and A. Zisserman, “Near Duplicate Image Detection: min-Hash and tf-idf Weighting,” BMVC, 2008.

[5] C. Davenport, British Heraldry, Methuen, 1912.

[6] C. Davenport, English heraldic book-stamps, figured and described, London: Archibald Constable. Ltd, 1909.

[7] C. R. Dod, and R. P. Dod, Dod’s Peerage, Baronetage and Knightage of Great Britain and Ireland for 1915, London: Simpkin, Marshall, Hamilton, Kent. Ltd, 1915.

[8] R. Duda, and P. Hart, “Use of the Hough transform to detect lines and curves in pictures,” Comm. ACM 15, vol. 1, 1972, pp. 11-15.

[9] A. Fornés, J. Lladós, and G. Sanchez, “Old Handwritten Musical Symbol Classification by a Dynamic Time Warping Based Method,” Graphics Recognition, vol. 5046, 2008, pp. 51-60.

[10] B. Gatos, I. Pratikakis, and S. J. Perantonis, “An adaptive binarisation technique for low quality historical documents,” Workshop on Document Analysis Systems, 2004.

[11] P. V. C. Hough, Method and mean for recognizing complex pattern. USA patent 3069654, 1996.

[12] E. Kavallieratou and E. Stamatatos, “Adaptive binarization of historical document images,” ICPR, 2006, pp. 742-745.

[13] E. Keogh, L. Wei, X. Xi, M. Vlachos, S. Lee, and P. Protopapas, “Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures,” VLDB Journal 18, vol. 3, 2009, pp. 611-30.

[14] T. Koch-Grünberg, Südamerikanische Felszeichnungen (South American petroglyphs), Berlin, E. Wasmuth A-G, 1907.

[15] J. Mas, G. Sanchez, and J. Lladós, “An Incremental Parser to Recognize Diagram Symbols and Gestures represented by Adjacency Grammars,” Graphics Recognition, 2006, pp. 252-263.

[16] A. Mueen, E. Keogh, and N. Shamlo, “Finding Time Series Motifs in Disk-Resident Data,” ICDM, 2009, pp. 367-376.

[17] A. Pritchard, A history of Infusoria, including Desmidiaceae and Diatomaceae, British and foreign. Ed. IV. 968. London, 1861.

[18] G. Ramponi, F. Stanco, W. D. Russo, S. Pelusi, and P. Mauro, Digital automated restoration of manuscripts and antique printed books, 2005, EVA.

[19] J. V. Richardson Jr., Bookworms: The Most Common Insect Pests of Paper in Archives, Libraries, and Museums.

[20] G. Sanchez, et al., “A platform to extract knowledge from graphic documents. application to an architectural sketch understanding scenario,” DAS, 2004, pp. 389 -400.

[21] K. B. Schroeder, et al. “Haplotypic Background of a Private Allele at High Frequency in the Americas,” Molecular Biology and Evolution, 2009, pp. 995-1016.

[22] W. Smith, A synopsis of the British Diatomaceae; with remarks on their structure, function and distribution, vol. 1, pp. [V]-XXXIII, pp. 1-89, 31 pls. London, 1853.

[23] Smith, G. A. and Turner, W. G. 1975. Indian Rock Art of Southern California with Selected Petroglyph Catalog, San Bernardino County.

[24] M. Tompa and J. Buhler, “Finding motifs using random projections,” Computational Molecular Biology, 2001, pp. 67-74.

[25] W. West and G. S. West, A Monograph of the British Desmidiaceae, vols. I-V, Ray Soc, London, 1904.

[26] H. J. Wolfson and I. Rigoutsos, “Geometric Hashing: An Overview,” IEEE Computer Science 4, vol. 4, 1997, pp. 10-21.

[27] X. Xi, E. Keogh, L. Wei, and A. Mafra-Neto, “Finding Motifs in a Database of Shapes,” SIAM SDM, 2007.

[28] Q. Zhu, X. Wang, E. Keogh, and S. H. Lee, “Augmenting the Generalized Hough Transform to Enable the Mining of Petroglyphs,” SIGKDD, 2009.

[29] Mining Historical Archives for Near-Duplicate Figures [Video]. May 2011. <http://www.youtube.com/watch?v=rEcEtI9FiyE>

[30] Support webpage. <http://www.cs.ucr.edu/~rakhant/DocMotif>