Title
Data Mining Historical Manuscripts and Culture Artifacts

Permalink
https://escholarship.org/uc/item/2jx150v7

Author
Zhu, Qiang

Publication Date
2011

Peer reviewed|Thesis/dissertation
UNIVERSITY OF CALIFORNIA
RIVERSIDE

Data Mining Historical Manuscripts and Culture Artifacts

A Dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Qiang Zhu

December 2011

Dissertation Committee:
Dr. Eamonn Keogh, Chairperson
Dr. Stefano Lonardi
Dr. Christian Shelton
The Dissertation of Qiang Zhu is approved:

________________________________________

________________________________________

________________________________________

Committee Chairperson

University of California, Riverside
Acknowledgments

I would like to first express my gratitude to my advisor Dr. Eamonn Keogh, without whom I would accomplish nothing during my Ph.D studying. I had always been considering myself as a perfectionist until working with Eamonn. “You will never spend too much time making a high quality figure”, I still remember this sentence he told me three years ago when I started writing my first paper. The font size and style of the legend, the layout, the color coordination, and even the width of the line, as long as there is still minor flaw, Eamonn would ask me to modify or improve it. His criticism once made me feel frustrated and baffled; however, every time when I compared the final approved version to the initial one, I gradually understood that it was worth every minute’s effort. Eamonn’s exceptional passion to the work has also been consistently propelling me. “I am already a full professor, but I still work from 9 everyday, you don’t have excuse to be lazy”, that is not a brag, Eamonn is the only one I know who goes to school for work on weekends. He set the best example for me that loving what you do is the only way to do great work. I feel tremendously fortunate to be guided by Eamonn, one of the most prolific researchers in data mining. Neither did he provide me the “magic bullet” for publishing, nor teach me how to grab several advanced techniques and combine them to a “new” model; it is his principle of “simple idea works” that impacts my research most, which allows me to dare to follow my own intuition. I cannot include every unforgettable episode in these three years, but please allow me to thank you again, Eamonn, for always being there when I need help!
I would also like to thank other two members in my dissertation committee – Dr. Stefano Lonardi and Dr. Christian Shelton. I took two graduate courses under Dr. Lonardi. His patient and careful teaching helped me prepare fundamental knowledge for my research. Dr. Shelton has provided me many insightful suggestions for my work, and makes my dissertation more solid.

Next, thanks to all my lab mates. Especially my collaborators: Thanawin Rakthanmanon, Abdullah Mueen and Xiaoyue Wang. The input from you inspired me a lot, and I could not have done all my work without you.

It is unimaginable to live in a foreign country without friends, and I feel truly lucky to meet so many of you. People say Ph.D. means “permanent head damage”, with your company, I am glad my head is still running well.

Finally, my deepest thanks go to my parents. Looking back in time, I have been moving further away from home ever since I was fifteen: from crossing the river in my hometown at the beginning, to flying to the other side of the Pacific Ocean now; from going back home once a week to only twice over the past four years. Nevertheless, the support and love from my parents never changes and they have made countless sacrifices for me. They never ask what I can do for them, but always ask what I need from them. Thank you, dad and mom! Thanks for your trust and letting me do whatever I like. This dissertation is dedicated to you.
ABSTRACT OF THE DISSERTATION

Data Mining Historical Manuscripts and Culture Artifacts

by

Qiang Zhu

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2011
Professor Eamonn Keogh, Chairperson

Initiatives such as the Google Print Library Project and the Million Book Project have already archived more than fifteen million books in digital format, over 11% of all the books ever published, and within the next decade the majority of world’s books will be online. While this digitized collection will be an invaluable resource for researchers to browse and search, we feel that the additional step of mining these manuscripts will reveal new insights, knowledge and historical context. Although most of the data will naturally be text, there will also be tens of millions of pages of images, many in color. In this dissertation, we introduce a simple color measure which both addresses and exploits typical features of historical manuscripts. To enable the efficient mining of massive archives, we propose a tight lower bound to the measure. Beyond the fast similarity search, we show how this lower bound allows us to build several higher-level data mining tools, including motif discovery and link analyses. We demonstrate our ideas in several data mining tasks on manuscripts dating back to the fifteenth century.
Compared to the well preserved or already digitized historical manuscripts, there is another category of cultural heritage, rock art, which requires urgent action in order to be explored and archived for prosperity. Rock art is an archaeological term for human-made markings on stone, including *carvings* into stone surfaces (petroglyphs) and *paintings* on stone (pictographs). It is believed that there are millions of petroglyphs in North America alone, and the study of this valued cultural resource has implications even beyond anthropology and history. Surprisingly, although image processing, information retrieval and data mining have had large impacts on many human endeavors, they have had essentially zero impact on the study of rock art. In this dissertation, we consider, *for the first time*, the problem of data mining large collections of rock art. We propose a robust distance measure for unconstrained and complex shapes, a cheap-to-compute tight lower bound, and algorithms based on these two ideas which enable very fast query-by-content and make the otherwise intractable data mining tasks in large collections of rock art tenable.
# Contents

List of Tables ...................................................................................................................... x
List of Figures .................................................................................................................... xi
Chapter 1 Introduction ........................................................................................................ 1
  1.1. Exploiting Historical Manuscripts and Rock Arts ............................................. 1
  1.2. Challenges ............................................................................................................ 7
    1.2.1. How to Obtain the Data ................................................................................ 7
    1.2.2. How to Measure the Similarity ..................................................................... 8
    1.2.3. How to Scale Up to MassiveDatasets .......................................................... 9
  1.3. Contributions ...................................................................................................... 10
Chapter 2 An Efficient and Effective Similarity Measure to Enable Data Mining of Petroglyphs ....................................................................................................................... 12
  2.1. Introduction ........................................................................................................ 13
  2.2. Background and Related Work .......................................................................... 14
    2.2.1. Background on Rock Art ............................................................................ 16
    2.2.2. Background on Image Processing ............................................................... 18
  2.3. Generalized Hough Transform ........................................................................... 22
    2.3.1. Classic Generalized Hough Transform ....................................................... 22
    2.3.2. A New Cell Incrementation Strategy .......................................................... 25
    2.3.3. The Intuition behind Lower Bounding ....................................................... 27
    2.3.4. A Formal Description of the Lower Bound ................................................ 30
    2.3.5. Variants on the Basic Distance Measure .................................................... 32
  2.4. Experimental Results.......................................................................................... 34
    2.4.1. Evaluation of Utility ................................................................................... 35
    2.4.2. Finding Motifs in Large Collections of Petroglyphs .................................. 37
    2.4.3. Evaluation of Accuracy................................................................................ 38
    2.4.4. Evaluation of Speed and Scalability ........................................................... 47
Chapter 3 Using CAPTCHAs to Index Cultural Artifacts ................................................ 52
  3.1. Do Petroglyphs Allow Automatic Feature Extraction? ...................................... 53
  3.2. A Sanity Check of Human Computation to Process Petroglyphs ....................... 54
  3.3. CAPTCHA-ROCK ............................................................................................. 56
3.3.1. A Simple Image-based Stickman CAPTCHA ............................................ 57
3.3.2. The CAPTCHA-ROCK System: Extracting Data from Petroglyphs ....... 59
3.3.3. Quantitative Evaluation of the CAPTCHA-ROCK System ....................... 62
3.3.4. Supporting Similarity Search................................................................. 64
Chapter 4 Mining Historical Manuscripts with Local Color Patches ..................... 68
  4.1. Introduction ........................................................................................................ 69
  4.2. Notation and Background ................................................................................... 71
  4.3. A Lower Bound to the Color Distance ............................................................... 78
  4.4. Mining and Search Algorithms .......................................................................... 82
    4.4.1. Nearest Neighbor Search ........................................................................ 83
    4.4.2. Motif Discovery .................................................................................. 88
  4.5. Empirical Evaluation .......................................................................................... 93
    4.5.1. Case Studies ..................................................................................... 94
    4.5.2. Data Mining Metrics ............................................................................ 106
  4.6. A Novel Framework for Image Retrieval ........................................................ 110
Chapter 5 Conclusions ............................................................................................... 115
Bibliography ............................................................................................................... 118
List of Tables

Table 2.1: The minimal unmatched edge points (MUE) from $Q$ to $C$. ...................... 25
Table 2.2: Algorithm to calculate the column lower bound of GHT by giving the query shape $Q$ and column signature of candidate shape $C$. ................................................. 31
Table 2.3: Comparison of the classification accuracy on 2128 clef symbols. ................ 44
Table 2.4: Comparison of the classification accuracy on 2128 clefs and 1970 accidentals. ............................................................................................................................... 45
Table 2.5: Comparison of the classification accuracy on architectural symbol dataset. .. 46
Table 4.1: The brute force algorithm to search the nearest neighbor of a user-given query $Q$ in a book $B$. ........................................................................................................... 78
Table 4.2: Divide-and-conquer algorithm to update the current nearest neighbor of a given query in a color patch. ...................................................................................... 84
Table 4.3: Fast algorithm using the lower bound distance to search the nearest neighbor NN of a query $Q$ in a book $B$. ......................................................................................... 87
Table 4.4: Algorithm to update the current best motif of size $h \times w$ between two patches ($BinMatrix1$ and $BinMatrix2$) with distance $d$. ......................................................... 91
List of Figures

Figure 1.1: Selected pages containing color images from historical manuscripts on different topics. 
*left* from *The secret book of honour of the Fugger* 1545-1548 [23]. 
*center* from a 1879-1901 text [39] containing illustrations of about 1250 species of butterflies. 
*right* from *A guide to the study of heraldry* 1840 [70].

Figure 1.2: Two maps (of 97 maps) that Smiley admitted he stole from libraries. 
*left* A general map of the middle British colonies in America (1978). 
*right* A map of the United States of North America (1802).

Figure 1.3: A random selection of petroglyphs and pictographs, hinting at their incredible variability, complexity and beauty.

Figure 2.1: 
*left* 'Horse' panel from the Hillaire chamber of the Chauvet Cave in Vallon-Pont-d'Arc, Ardèche, France, which shows a rhinoceros and was drawn more than 30,000 years ago [93]. 
*right* A engraved ochre from Blombos Cave, South Africa, featuring three parallel lines and a row of cross hatching [47].

Figure 2.2: An excerpt from an 1888 government report [78]. The original caption is “Petroglyph in Arizona”.

Figure 2.3: 
*left* An Ibex petroglyph taken from [88] has its two rear hoofs fused. It is not clear if this is an artifact of scanning or the artist’s intent, and it does make a critical difference to graph based methods. 
*center* This bighorn sheep from a classic work [43] has a disconnected leg and horn, which will greatly affect its representation for graph based methods. 
*right* Two petroglyphs from Easter Island are clearly distinct, yet identical in graph based representations.
Figure 2.4: *left*) Two overlaid skeleton traces of the same image of a Bighorn sheep. *right*) The same two images after downsampling.

Figure 2.5: Toy examples of a query $Q$ and a candidate match $C$. Each cell is a pixel, and the dark colors denote edge points of shapes.

Figure 2.6: *left* and *center*) The shape $Q$ is rotated $180^\circ$ around center of mass $R$. *right*) Four vectors of $Q$ form a “star pattern”.

Figure 2.7: Placement of vectors on each edge point of $C$ (*left*) and the final accumulator $A$ (*right*).

Figure 2.8: Four vectors of $Q$ (*left*) and placement of vectors on one cell of $C$ (*right*).

Figure 2.9: We can extract “signatures” from shapes by summing up the number of edge points in each column.

Figure 2.10: *left*) A query image $Q$ has its signature $\text{Sig} Q_x$ extracted. *right*) By noting how many edge points it needs $C$ to have at each column, and how many edge points the column as $C$ actually has, we can derive a lower bound of $D(Q,C)$.

Figure 2.11: *left*) A group-average linkage hierarchical clustering of typical Southwestern USA petroglyphs, with the $D_{\text{clustering}}$ measure. *right*) While the dendrogram to the left shows the full resolution images for clarity, the images input to the distance measure have been binarized, thinned and scaled to fit into a 30 by 30 bounding rectangle.

Figure 2.12: The GHT distance is able to find the intuitive similarity between pairs of anthropomorphic figures, in spite of the different styles of representations.

Figure 2.13: *left*) Five representative motif pairs from the top 52 motifs. *right*) Histogram of the pair-wise distance $D_{\text{motifs}}$ in 2,852 petroglyphs.

Figure 2.14: Examples of 14 categories from NicIcon dataset.

Figure 2.15: Error rate vs. Resolution. Downsampling test on training set of the WD and WI tests in 9 resolutions. Error rate makes little difference once the resolution is larger than $10 \times 10$. Note WI is better than WD here, because the training set of WI contains only 60% of writers while that of WD contains all.

Figure 2.16: Sample digits from Farsi dataset. Note: numbers 2, 3 and 4 are very similar (3 and 4 in the third row are even impossible for humans to distinguish); some digits have different styles (4 and 6); some digits are in bad quality (7, 8 and 9 in the third row).

Figure 2.17: Error rate vs. Resolution. Downsampling test on the training set of Farsi dataset in four resolutions. Note that the error rate varies little when the resolution is greater than $10 \times 10$.

Figure 2.18: Sample clef symbols from the old handwritten musical symbol dataset. *First row:* printed Bass clef followed by 6 samples; *second row:* printed Treble clef followed by 6 samples; *third row:* printed Alto clef followed by 6 samples, in which the last three are very different from the first three.

Figure 2.19: Representative clefs for Bass, Treble and Alto (the last two).

Figure 2.20: Sample symbols from 1970 accidentals. In each row, a printed accidental symbol is followed by six samples in the corresponding class. Four classes are Sharp, Natural, Flat, Double Sharp (from top to bottom).

Figure 2.21: Fifty classes (printed version) of architectural symbol dataset.
Figure 2.22: The Synthetic Petroglyphs Dataset. *First row:* samples of petroglyph templates; *second row:* sample petroglyphs of the basic “human-copied” dataset; *third row:* samples of distorted petroglyphs. Note that for each template, we have copies in different scales, translations, orientations and non-linear distortions. ........................................ 48
Figure 2.23: Time taken for the 1-NN query on eight synthetic petroglyph datasets. For each dataset, the maximal, average and minimal time of 10 runs are reported. Note log scale is used in x axis................................................................. 48
Figure 2.24: Pruning rate of our lower bound algorithm on eight synthetic petroglyph datasets. For each dataset, maximal, average and minimal rates are reported. Note log scale is used in x axis................................................................. 49
Figure 2.25: Percentage of execution time for our lower bound algorithm relative to the brute force algorithm. Note log scale is used in x axis................................................................. 50
Figure 2.26: Time of finding motifs in eight synthetic petroglyph datasets. Note log-log scale................................................................. 50
Figure 2.27: Speed-up of our lower bound algorithm against brute force algorithm of finding motifs in increasingly large petroglyph datasets. For the brute force algorithm, we only ran it on the 10,000 datasets and extrapolated other values. Note log scale is used in x axis. ......................................................................................................................... 51
Figure 3.1: *(left)* Reindeer rock art from Norway, dating to 4200 to 500 BC. The rock carvings have been retouched in bright red by researchers, making them extremely high contrast. *(right)* a segmentation of the image using the Prewitt method, carefully tuned. ......................................................................................................................... 53
Figure 3.2: *First row:* eight original petroglyphs (a to h). *Second row:* corresponding skeletons transcribed by the volunteer SC. *Third row:* skeletons transcribed by the volunteer WY................................................................. 55
Figure 3.3: A group-average linkage hierarchical clustering of all sixteen transcribed petroglyphs. ......................................................................................................................... 56
Figure 3.4: Two examples of CAPTCHAs. In order to solve the CAPTCHA and get access to the next webpage (in this case, offering a free email account) the user must type in 28iVW and jw62K respectively. ......................................................................................................................... 57
Figure 3.5: Four examples of a parameterized Stickman. ......................................................................................................................... 58
Figure 3.6: *(left)* A pair of randomly generated stickmen. *(right)* The distribution of GHT distances between 1,000 pairs of randomly generated stickmen. ......................................................................................................................... 58
Figure 3.7: *(left)* A randomly generated stickman in black and a human tracing of it in red. *(right)* The distribution of GHT distances between randomly generated stickmen and human tracings of them are shown with a finer bucket size (in red), because there is less data. The distribution of GHT distances between two randomly generated stickmen is shown for context (in blue). ......................................................................................................................... 59
Figure 3.8: A CAPTCHA-ROCK consists of two rock art images: one control image and one unknown image. Note that users do not know which one is which. ......................................................................................................................... 60
Figure 3.9: *(left)* A petroglyph from Utah that has been indexed in our database. *(right)* An image of the same petroglyph found on Flickr.com. Could this image be used as a query to retrieve the anthropologist’s annotated version in our database? ......................................................................................................................... 65
Figure 3.10: (left) A query petroglyph that happens to be in our database and its tracing. (right) The five nearest neighbors to the query; the first four all refer to the same image, the correct target.

Figure 3.11: An abstract (top) and animal petroglyph (bottom) which had been traced and issued as queries to our database. The list of the five nearest neighbors to each are shown left to right.

Figure 3.12: (left) An anthromorph is used as a query, and it retrieves another stylized human figure with similar limbs. (right) A petroglyph of concentric circles retrieves a "sunburst".

Figure 4.1: (left) Two pages from *The secret book of honour of the Fugger* 1545-1548 [23]. (right) Two pages from an 1834 text on fish [12].

Figure 4.2: An illustration of some basic notations. (left) A 6×10 page showing a (slightly modified) Benin flag. The 4×3 green window on the left is one of its color patches. (right) A matrix of the same size as the page, with the number indicating the bin_id of corresponding pixels. For simplicity, we set $K = 4$ and assign one bin for each color. Note that for surrounding transparent pixels, we set bin_id = 0. The color histogram can be easily obtained from this matrix, which is \{8,12,1,7\}.

Figure 4.3: A section of images to demonstrate the need for spatial-location invariant distance measures. From left to right: in page 109 of [23], in page 108 of [23], an image of a Death’s Head Moth from 1849 [30], and one from 1860 [91].

Figure 4.4: A section of image pairs that are clearly distinct, but essentially indistinguishable under the distance measure introduced in equation (2).

Figure 4.5: Five hand-colored images from the 1754 version of [80] and five hand-colored images from the 1789 version of the same text. An inspection of a higher resolution color version of this figure in [105] makes it clearer that these images are significantly different in both color and texture.

Figure 4.6: (left) Two example queries with their color histograms. (right) A sample page (cf. Figure 4.2) and its (global) histogram.

Figure 4.7: A query color patch and two pages from [23]. Note Page 2 contains a very similar color patch to the query in the bottom left corner.

Figure 4.8: (left) A query of size $h \times w$. (right) A page of size $H \times W$, which is divided into $2 \times 2$ sub-patches (only two on the left are plotted). Note there is an overlap between them to include all offsets.

Figure 4.9: (left) A query image of an illustration of *Scorpaena scrofa* and its best match in the text [12]. (right) For context, six random other fish from the same text.

Figure 4.10: Three images of real fish from the genus *Scorpaena*. Each of them was used as a query to the text [12], and each of them found the same.

Figure 4.11: (left) A modern idealized version of the coat of arms of Fugger von Babenhausen is used as a query. (center) The best match to query in the 1545 text, *Das Ehrenbuch der Fugger*. (right) A zoom-in of the best match.

Figure 4.12: (left) We can edit the coat of arms discovered in our search by removing the center panel and stretching the two remaining sections to meet in the middle. (right) The resulting image looks more like the query.
Figure 4.13: left) A query coat of arms in a page from The Saxon pedigree book [25]. center) enlarger query and best match. right) the best quartered match to the query found in the same text. ................................................................................................................ 99
Figure 4.14: The coat of arms from Figure 4.13. left queried in two templates. The top row shows results of a left-right template and the bottom row shows result of a quadrant template.......................................................................................................................... 100
Figure 4.15: Examples of reverse search to find origins for patches in quartered coats of arms. ................................................................................................................................ 101
Figure 4.16: left) Photo of the famous “three leopards” (or lions) as the query. center) The best match found in [31]. right) Four additional coats of arms from the same text. 102
Figure 4.17: left) A pair of pages which contains 27 leaf patch pairs, with 6 leaf patches on left page and 7 on the right page displayed. right) another pair contains 170 leaf patch pairs, with 16 and 18 leaf patches on each page............................................................... 103
Figure 4.18: left) A photo of an orange/black butterfly used as a must-include color constraint on motif discovery. right) A pair of tiger butterflies found by the supervised motif discovery algorithm in [49]................................................................................................. 104
Figure 4.19: Two links found between two butterfly books. Note that the technique seems robust to the text variability (color palette, size, etc.). According to an entomologist, the left match is correct at the genus level and the right match is correct at the species level. ......................................................................................................................................... 105
Figure 4.20: Two examples of links found by a motif join. Both matches are correct at the species level ................................................................................................................... 106
Figure 4.21: left) A color patch cut from the Fuggers book and a query made by randomly removing a row slice and a column slice both taking 1/5 area. right) The nearest neighbor found in the book. Note that it is irrelevant to the changing of parameters...................................................................................................................... 107
Figure 4.22: left) “Best execution time to find NN” vs. “Sec” for 5 randomly chosen queries from the Fuggers book. Note only 20,30,40 are tested for Sec >10. right) “Execution time to find NN” vs. “StopDepth” for the same 5 queries when Sec = 2. Note StopDepth = 1 corresponds to the brute-force search (but still with the heuristic page order).............................................................................................................................. 107
Figure 4.23: “Goodness of BSF” vs. “Time” for motif discovery in the Fuggers book. The “goodness” is measured as “eventual BSF/current BSF”......................... 109
Figure 4.24: left) A page from A guide to the study of heraldry 1840 [70], containing a coat of arms called three torteaux with surrounding text “Louis VI King of France”. right) 1st page results of query “Louis VI King of France” in Google image search engine.... 112
Figure 4.25: left) Query patch from the page in Figure 4.24. left. right) Re-ranked images based on the distance to the query (only top images are shown), in the order from left to right, top to bottom. .............................................................................................. 113
Figure 4.26: left) One page from [97], containing one medal with text “spanish american war”. right) 1st page results of query “spanish american war medal” in Google image search engine.................................................................................................................. 113
Figure 4.27: left) Query patch from the page in Figure 4.26. left. right) Re-ranked images based on the distance to the query, in the order from left to right, top to bottom......... 114
Chapter 1

Introduction

1.1. Exploiting Historical Manuscripts and Rock Arts

In 2004, Google began scanning books to make their contents searchable and discoverable online [40]. By October 14, 2010 Google has scanned over fifteen million books [22], over 11% of all the books ever published, which contains over five billion pages and two trillion words, with books dating back to as early as 1473 and with text in 478 languages. Google intends to scan all 129,864,880 unique books in the world [89] by the end of the decade. While this digitized collection will be an invaluable resource for researchers to browse and search, we feel that the additional step of mining these manuscripts will reveal new insights, knowledge and historical context. In a similar vein,
technology writer Kevin Kelly has observed, “the real magic will come in the second act, as each word in each book is cross-linked, clustered, cited, extracted, indexed, analyzed, annotated, remixed, reassembled and woven deeper into the culture than ever before” [55]. While this remark explicitly singles out text, a similar argument can be made for images: although most of the data will naturally be text, there will also be tens of millions of pages of images, many in color (as shown in Figure 1.1).

Figure 1.1: Selected pages containing color images from historical manuscripts on different topics. *left*) from *The secret book of honour of the Fugger* 1545-1548 [23]. *center*) from a 1879-1901 text [39] containing illustrations of about 1250 species of butterflies. *right*) from *A guide to the study of heraldry* 1840 [70].

One of the focuses in this dissertation is to exploit the rich color information often present in historical manuscripts. The reasons why we decided to work on color images come in two-folds. First, compared to images, historical texts are generally harder to read and understand (e.g.: the cursive scripts in Figure 1.1.*left*), and text annotations will probably defy even the state-of-the-art handwriting recognizers. Second, while there is an active research community pursuing data mining of text from historical manuscripts [5][29][32][37][60][82][98][99], historical images have received much less attention.
The utility of efficient mining of massive historical archives is not just limited to the research on history itself, as we shall show in Section 4.5.1:

- A historian can find out how long humans have known fish in a particular genus are poisonous by querying a real image of fish from Google image search in a classic 1834 text about fish in the area he is interested in.
- A genealogist can trace lineages by searching coats of arms which are extremely common in Western manuscripts dating back to fourteenth century.
- A tourist, who wants to know the origin of a coat of arms spotted somewhere in Europe, can take a photo on his iPhone and submit the query to indexed text.
- An entomologist, who wants to find two similar butterflies containing the similar orange color in one specimen of “tiger butterfly”, can issue a must-include supervised motif discovery in a given book.
- A reader, who has questions about butterflies from one book, can reference another heavily annotated text via hyperlinks created by link analyses between two books.

In addition to the above scenarios, mining historical archives could also be useful in the criminal investigation. There is a huge market in the illicit trafficking of cultural properties. According to the 1999 United Nations Global Report, the estimated annual trade in illicit antiquities is around 7.8 billion, ranking just behind drugs and arms as the most profitable black market [19]. As noted by Colin Renfrew, the Director of the McDonald Institute for Archaeological Research, “The single largest source of destruction of the archaeological heritage today is through looting – the illicit,
unrecorded and unpublished excavation to provide antiquities for commercial profit\(^*\).

Among all such criminal cases, the stealing and trafficking of rare manuscripts has grown into an epidemic problem [4][17][19]. Many valuable historical manuscripts are stored at university and public libraries without sufficient supervision, and thieves can often easily cut off individual pages and sell them at online auction website like eBay. For example, in the high profile Smiley case, a rare maps dealer Edward Forbes Smiley III admitted that he had stolen and sold 97 maps (two of which are shown in Figure 1.2.) from collections in the US and UK major libraries, worth an estimated $3 million [4].

![Figure 1.2: Two maps (of 97 maps) that Smiley admitted he stole from libraries. left) A general map of the middle British colonies in America (1978). right) A map of the United States of North America (1802).](image)

The identification of individual pages is difficult, because the marks added by libraries usually have been removed and sometimes it is even hard to tell whether a page was originally from a book. Dealers, collectors and libraries which the thieves admitted have stolen manuscripts from are all required to involve in the recovery process and a page-by-page manually checking is painstaking and time consuming. If all the manuscripts have been indexed in a database, by enabling the fast similarity search, the recovery process can be done much more efficiently. Furthermore, libraries can also perform the link analysis between their archives and those pages sold online to detect if
any page has been stolen (as reported, many libraries have not even realized pages are missing until told so [4]). We believe such actions would become forceful approaches against current rampant online black market.

Compared to the well preserved or already digitized historical manuscripts, there is another category of cultural heritage, rock art, which requires more urgent actions to be explored and archived for humanity. Rock art is an archaeological term for human-made markings on stone, including petroglyphs, *carvings* into stone surfaces and pictographs, *paintings* on stone. Figure 1.3 illustrates some examples of each, which hint at the extraordinary variability of rock art in terms of complexity.

![Figure 1.3: A random selection of petroglyphs and pictographs, hinting at their incredible variability, complexity and beauty.](image)

Petroglyphs and pictographs are one of the earliest expressions of abstract thinking, and the study of this valued cultural resource has implications even beyond anthropology and history. A decade ago, Walt et al. summed up the state of petroglyph research by noting, “*Complete-site and cross-site research thus remains impossible, incomplete, or impressionistic*” [95]. Surprisingly, progress in rock art research has been frustratingly slow in the intervening decade, yet we have seen significant advances in image processing, information retrieval and data mining. In the meanwhile, due to the
nature of rock arts, they are mostly left in outdoor settings, exposed to elements of nature that will erode them inevitably with time.

In this dissertation, we are going to consider, for the first time, the problem of data mining large collections of rock art. By introducing a robotic distance measurement which can capture the similarity of rock art and a tight lower bound to this distance which can be quickly calculated, our proposed algorithms allow efficient and effective mining of rock art. As we will show in Section 2.4:

- Our proposed distance measure can correctly (subjectively) cluster a collection of real petroglyphs from the Southwest USA and seems invariant to several features like the hollow/solid nature (see Figure 2.11).
- It only takes about 18 minutes to find top patterns/motifs from 2,852 real petroglyph images (4,065,526 possible pairs) by our motif discovery algorithm. Interestingly, some of the detected motifs are in fact known to be true meaningful motifs (see Figure 2.13).
- We achieve competitive or superior accuracy for query-by-content compared to state-of-the-art algorithms on four publicly available datasets that are very similar to (at least some kinds of) petroglyphs.
- In the very large datasets (containing up to 1,280,000 images), our lower-bound based algorithms can prune the majority of the calculations and achieve up to 50 times speed up for the nearest neighbor search and 100,000 times speed up for the motif discovery.
Although our distance measure was initially designed for petroglyphs, it is general enough to be applied to different domains. In [79], our distance measure is used to automatically discover repeated shapes in historical archives and the results show its robustness to line thickness, solid vs. hollow shapes, noises and various other distortions. The distance measure can even be used for analyze the mice sounds [100], in which the mice vocalizations are transformed to the visual (spectrogram) space, and similarity search, classification and motif discovery, etc. data mining algorithms on mice song snippets are enabled.

1.2. Challenges

1.2.1. How to Obtain the Data

There are already large collections of digitized historical manuscripts available online, on which our proposed algorithm in Chapter 4 can directly work. In addition to the Google Books Project [40], which already scanned over fifteen million books and intends to scan all books by the end of this decade as we noted at the beginning of this chapter, many libraries and non-profit organizations also provide free access of historical manuscripts to the public. For example, the Internet Archive [53], a non-profit internet library, offers “permanent access for researchers, historians, scholars, people with disabilities, and the general public to historical collections that exist in digital format”. The Munich Digitization Center [72] provides one of the largest and fastest growing digital collections of cultural heritage in Germany, including the famous 500-year-old text Das Ehrenbuch der Fugger (The secret book of honor of the Fugger) which we will
discuss in details in Chapter 4. The Smithsonian Institution Libraries founded in 1846 is a system of 20 branch libraries, with 1,571,114 volumes and manuscripts in 2,109 linear feet held, and its Digital Library [27] offers free access to the public.

In contrast, for rock art, it is a more difficult problem to obtain the data we require for the algorithm proposed in Chapter 2 which assumes the input images are bitmaps with a 1-bit color depth. Although some of existing archives of rock art are already in such data format (for example, anthropologists have been sketching petroglyphs for hundreds of years), most are raw images as we have shown in Figure 1.3. With rare exceptions, petroglyphs do not lend themselves to automatic extraction with current segmentation algorithms for these images. Cracks often present in the rock would probably be recognized as part of petroglyphs without high level semantic context. How to extract a single petroglyph from one image containing multiple objects is another issue. Furthermore, we would like to remind readers that images in Figure 1.3 are among the highest contrast and clearest example of rock art.

1.2.2. How to Measure the Similarity

To define an effective similarity/distance measurement is at the heart of our work, since it is the basis of any higher-level data mining algorithms.

For the shape similarity, feedback from various researchers in the data mining and image processing community suggested us consider Geometric Hashing [96], Hausdorff Distance [52], Chamfer Matching [14], Shape Contexts [11], Fréchet Distance [6], Skeleton Graphs [9], Zernike moments [90], Earth Movers [8], etc., among other hundreds of shape similarity measures [94][101].
Similarly, there are also a number of color descriptors, and lots of measurements based on each of them. Say, for the widely used color histogram, different measures are proposed including simple ones such as Euclidean Distance and Histogram Intersection [87], and more complex techniques such as Histogram Quadratic Distance [34], Histogram Refinement [75], and Color Set Distance [86], etc.

In selection of the most suitable measurement, we should keep in mind special properties of the objects in question. As for the rock art, there does not exist a finite (and relatively small) number of classes; right angles/intersections/circle centers are seldom clearly defined; open/closed boundaries and connected/disconnected shapes are often present, etc. Moreover, simplicity (for example, the ideal measurement should be parameter free or at least not require complicated parameter tuning) and efficiency (which we will discuss in the next section) are two extra issues we should take care of.

1.2.3. How to Scale Up to Massive Datasets

Assume we have solved the above two issues, i.e.: extracted the data in the format that is workable with our algorithms and found effective distance measures which capture the similarity in color and shape, but if we cannot solve the scalability problem (as noted in Section 1.1, there are millions of books and millions of petroglyphs), the practical utility our work would be largely limited.

Let us first see one example in Chapter 4: if we search a 400×400 color patch in a book which contains 500 pages with the size 2000×1500, then the total number of color patches we need to check is more than 881 million, which clearly precludes interactive real-time search. Now if our task is to find out the most similar pair of 400×400 color
patches in the same book, the number of comparisons required is further increased to about $7.75 \times 10^{17}$, an incredibly large search space! Furthermore, in many cases the similarity search is a frequently-called subroutine in a higher level data mining algorithm, and we must drastically improve the performance of the brute force search.

The same issue also exists in mining the petroglyphs (Chapter 2). The time complexity of classic Generalized Hough Transform (see detailed introduction in 2.3.1) which we will use to measure the shape similarity is quite high, and thus limits its applicability for larger datasets.

1.3. Contributions

The main contribution of this dissertation is summarized as follows:

- We considered, for the first time, the problem of data mining large collections of rock art. To capture the similarity of the extraordinarily diverse and complicated data, we introduced an explicit framing of Generalized Hough Transform (GHT) as the distance measure. To scale up to large datasets, we estimated a lower bound distance based on one dimensional signatures extracted from original data, which makes the similarity search in real time and the otherwise intractable task of finding motifs in large datasets tenable. We believe that our work can not only bootstrap additional research in the area of petroglyph mining, but also be easily applied to other domains, such as automatically discover repeated shapes in historical archives [79], analyze the mice sounds [100], etc.
We proposed the *first* real-valued-response CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) for crowdsourcing in data mining. Our CAPTCHA-ROCK system (c.f. Section 3.3) considers inherently real-valued data (photographs of rock art) and expects real-valued responses (mouse movements). With this human computation tool, human efforts spent solving the CAPTCHAs (usually wasted) now can be utilized to help extract useful data from incredibly heterogeneous and noisy datasets.

We considered the problem of mining historical manuscripts with local color patches, and made it work for massive datasets. Other than most of work on indexing color images that consider global atomic images, we focused on the local region-of-interest patches, which we think more flexible and practical. Furthermore, we extended the limited work about local color matching [24][67][87] which only consider *query-by-content* for *main memory search* to more *general data mining* for datasets which are so large that they must be *disk-resident*.
Chapter 2

An Efficient and Effective Similarity Measure to Enable Data Mining of Petroglyphs

Rock art is an archaeological term for human-made markings on stone, including carved markings, known as petroglyphs, and painted markings, known as pictographs. It is believed that there are millions of petroglyphs in North America alone, and the study of this valued cultural resource has implications even beyond anthropology and history. Surprisingly, although image processing, information retrieval and data mining have had a large impact on many human endeavors, they have had essentially zero impact on the study of rock art. In this work we identify the reasons for this, and introduce a novel
distance measure and algorithms which allow efficient and effective data mining of large collections of rock art.

2.1. Introduction

Petroglyphs and pictographs (random selections are shown in Figure 1.3) are one of the earliest expressions of abstract thinking, and a true hallmark of humanity. They provide a rich body of information on several different dimensions, beyond their value as an aesthetic expression. Studies of rock art have implications beyond anthropology and history. For example, a recent study postulates the existence of a now-extinct Australian bat species based on extraordinarily detailed pictographs known to be at least 17,500 years old [77]; petroglyphs have been used in studies of climate change; and the changing inventories of species in the Dampier Archipelago from the Pleistocene to the early Holocene period have been reconstructed partly by petroglyph evidence [7]. However, in spite of these successes, progress in petroglyph research has been frustratingly slow.

A decade ago, Walt et al. summed up the state of petroglyph research by noting, “Complete-site and cross-site research thus remains impossible, incomplete, or impressionistic” [95]. Surprisingly, there has been little change in the intervening decade, yet in the same time frame we have seen significant advances in image processing and data mining. These advances have resulted in fielded applications in domains as diverse as medicine, entertainment, wildlife management, e-commerce, biometrics, zoology [74], etc. Nevertheless, these advances have had essentially zero impact on the analysis of petroglyphs and pictographs.
We believe that this is because the extraordinarily diverse and complex structure of rock art images defies most existing image matching algorithms. Most approaches are simply not suitable to capture the similarity of petroglyphs, and those that are, even in limited cases, do not scale to large collections we need to examine. In this chapter we introduce a novel distance measure for rock art, and show that it can correctly capture the subjective (and where available, objective) similarity between petroglyphs. We show how we can use this distance measure as a basis for several higher-level “data-mining” algorithms, for example finding repeated motifs, clustering, or simply enabling query-by-content.

The rest of this chapter is organized as follows. Section 2.2 contains background information and a discussion of related work. In Section 2.3 we review the Generalized Hough Transform, and show how we can adapt it to produce a fast and robust distance measure for petroglyphs. Finally we test our ideas with a comprehensive set of experiments in Section 2.4.

2.2. Background and Related Work

The earliest petroglyphs have traditionally been associated with the appearance of modern humans in Europe such as the famous example from the Lascaux Cave, France, and an early one (shown in Figure 2.1.left) from the Chauvet Cave, France which dates back to as early as 30,000 years ago [93]. Recent work has shown that the idea of expressing abstract motifs appears much earlier, 77,000 years ago in South Africa (one of
them is shown in Figure 2.1.\textit{right}) [47]. Given this long history, it is one of the most valuable sources of humanity that has persisted to the present time.

Figure 2.1: \textit{left}) 'Horse' panel from the Hillaire chamber of the Chauvet Cave in Vallon-Pont-d'Ardèche, France, which shows a rhinoceros and was drawn more than 30,000 years ago [93]. \textit{right}) A engraved ochre from Blombos Cave, South Africa, featuring three parallel lines and a row of cross hatching [47].

Beyond their value as an aesthetic expression, petroglyphs provide a rich source of information for researchers. Repeated motifs can be identified and traced through time and space, which in turn may shed light on the dynamic histories of human populations, patterns of their migrations and interactions, and even continuities to the present indigenous societies. However, the nature of petroglyphs poses an extremely difficult challenge. As in the case for any other artifacts of history, damage to petroglyphs is permanent and irreversible. However, unlike other artifacts that can be preserved and protected within the confines of a controlled environment in a museum, petroglyphs are mostly left in their natural settings, exposed to elements of nature that will erode them inevitably with time. There is an urgent need to identify petroglyphs and to archive them for humanity.
2.2.1. Background on Rock Art

As we shall show in Section 2.3, our algorithm assumes that the input images are (relatively) low-resolution bitmaps with a 1-bit color depth, one petroglyph per image. However, as Figure 1.3 illustrates, obtaining such images may be non-trivial. With rare exceptions, petroglyphs do not lend themselves to automatic extraction with segmentation algorithms. For example, in the first and third images of Figure 1.3, segmentation algorithms find the “edges” due to cracks in the rock to be more significant that the actual edges of the petroglyphs. Moreover, these images were chosen for this example for their high contrast and clarity; most petroglyphs would be even more challenging. In spite of this, in the next two sections we show how we easily obtained tens of thousands of petroglyphs for this study, and how we plan to have at least one million examples in the very near future.

2.2.1.1. Human Computation to Process Petroglyphs

The last five years have seen a flurry of research on Human Computation (also known as “crowdsourcing”), much of it leveraging off the pioneering work of Luis von Ahn at CMU [1]. The essence of human computation is to have computers do as much work as possible to solve a given problem, but to outsource certain critical steps to humans. These steps are ones which are difficult for computers, but simple for humans. One of the most famous examples is the Google Image Labeler, which is a program that allows the user to label random images to help improve the quality of Google’s image search results. Like many such efforts, human time is donated for free, because the task is
embedded in a fun game, hence the recently coined term, Games with a Purpose, or GWAP [3].

We have exploited and extended some of these ideas to create a system called CAPTCHA-ROCK, which allows human volunteers to “help” computer algorithms segment and annotate petroglyphs. We leave a detailed discuss of CAPTCHA-ROCK system to Chapter 3.

2.2.1.2. Existing Archives of Petroglyphs

Beyond the examples captured by our human computation system, there are several other rich sources of rock art data to be mined. For example, anthropologists have been sketching petroglyphs for hundreds of years, and recent efforts to digitize historical manuscripts have made at least hundreds of books, each with at least a few thousand petroglyph images, freely available on the web. In Figure 2.2 we show an example from the 1888 edition of a series of government reports [78].

![Figure 2.2: An excerpt from an 1888 government report [78]. The original caption is “Petroglyph in Arizona”.

Images of this type can be of particular interest because they may refer to petroglyphs which have long since been destroyed. Furthermore, although the petroglyphs in Figure 2.2 predate photography, it is important to note that because
petroglyphs often do not reproduce well in photographs, the practice of hand drawing or tracing petroglyphs is still used in modern anthropological texts.

2.2.2. Background on Image Processing

An understanding of similarity must be at the heart of any effort to analyze petroglyphs and other cultural artifacts. For example, an image of a horseman incised on a fossilized ostrich eggshell fragment was recently found among eolian deposits in the Gobi Desert, Mongolia [68]. An obvious thing to do with such an image in order to place it in a cultural context is to ask if a similar image exists among the many petroglyphs in the region. Thus, we began this project with careful consideration of shape similarity.

In soliciting feedback and advice for early previews of this work from various researchers in the data mining and image processing community, the feedback obtained was almost always of the form “Very nice, but have you considered using X”, where X was Geometric Hashing [96], Hausdorff Distance [52], Chamfer Matching [14], Shape Contexts [11], Fréchet Distance [6], Skeleton Graphs [9], Zernike moments [90], Earth Movers [8], etc. While we have considered (and in some cases experimented with; see [103]) these distance measures, space limitations prohibit a detailed review and discussion of the pros and cons of each of them. Indeed, the preceding list is only a small subset of the hundreds of shape similarity measures in existence. (See [94][101] and the references therein for an overview.) However, we argue that some of the unique properties of petroglyphs render most of them unsuitable for the task at hand. Consider the following difficulties illustrated by Figure 2.3:
A single atomic petroglyph may contain several disconnected parts. Thus, boundary based methods [56] and graph based methods [9] cannot be applied, at least not directly (c.f. Figure 2.12, which shows an example of a problem which would defeat boundary and graph based methods).

Geometric hashing is a very useful technique for indexing large collections of shapes [96]. However, it is only well defined for machine parts and architectural drawings with many clearly defined right angles/intersections/circle centers, etc. It has not been shown to have utility for more general unconstrained shapes.

There are many specialized distance measures which have been introduced for indexing music notation, Japanese kanji, mathematical symbols, pen-based computing, etc. At least some subsets of these look like at least some subsets of petroglyphs. However, it must be remembered that in these domains there are only a finite (and relatively small) number of possible classes, and we can at least imagine an idealized prototype for each class (i.e. a perfectly drawn square root sign). However, this is not the case for petroglyphs which do not generally fall into discrete classes, and cannot generally be seen as corrupted versions of an idealized template.
Figure 2.3: **left**) An Ibex petroglyph taken from [88] has its two rear hoofs fused. It is not clear if this is an artifact of scanning or the artist’s intent, and it does make a critical difference to graph based methods. **center**) This bighorn sheep from a classic work [43] has a disconnected leg and horn, which will greatly affect its representation for graph based methods. **right**) Two petroglyphs from Easter Island are clearly distinct, yet identical in graph based representations.

Instead of attempting an exhaustive discussion of why we have discounted existing shape distance measures, we will briefly review the positive reasons for why we chose the GHT measure.

- As we shall show, on real, but unlabeled anthropological datasets, the GHT produced subjectively correct answers (cf. Section 2.4.1). Furthermore, on *labeled* datasets which are very similar to petroglyphs, the GHT produces results which are competitive with state-of-the-art approaches.
- As we will demonstrate in this work, we are able to tightly lower bound the GHT, allowing for very efficient searches in large datasets. Moreover, we show that we can make a slight variant of the GHT obey the triangular inequality, thus allowing us to use off-the-shelf data mining algorithms, for example to find motifs [71].
- The GHT makes essentially no assumptions about the data, and thus is defined for open/closed boundaries, for connected/disconnected shapes, etc. This is
important because, as hinted at in Figure 1.3, Figure 2.2 and Figure 2.3, petroglyphs are extraordinarily diverse.

We are now in a position to give some intuition as to why we intend to do data mining on a relatively low resolution of the petroglyph images. Using our CAPTCHA-ROCK, we asked two individuals to trace a petroglyph of a bighorn sheep petroglyph found in Arizona; the resulting two skeletons are shown in Figure 2.4.\textit{left}. The skeletons are on a bitmap of 340 by 250. Although the two images are very similar, less than 3.5\% of the pixels from each image overlap. We can contrast this with the situation after converting the images to a down sampled representation, as shown in Figure 2.4.\textit{right}. Here the images are transformed to a mere 30 by 23 grid representation. However, of the 130 pixels that form each image, 75.6\% of the pixels are common to both.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.4.png}
\caption{	extit{left}) Two overlaid skeleton traces of the same image of a Bighorn sheep. \textit{right}) The same two images after downsampling.}
\end{figure}

In essence, the original image representation has spurious precision. This precision is unwarranted because there is some uncertainty introduced by the human
element of the algorithm\(^1\). The quantizing produced in the downsampling step also introduces some uncertainty, but this is completely dwarfed by original uncertainty. Furthermore, as we shall see, the lower resolution representation has several unique advantages which we can leverage off. In Section 2.4, we provide forceful empirical evidence that appropriate amounts of downsampling significantly improve accuracy in objective tests.

2.3. **Generalized Hough Transform**

We begin by reviewing the classic generalized Hough Transform algorithm and then introduce our modifications and extensions.

2.3.1. **Classic Generalized Hough Transform**

The Hough transform [51][33] is a useful method for two-dimensional shape detection, but it is limited to analytic curves. It was generalized to detect arbitrary shapes in [10][69]; however, these works did not explicitly encode a similarity measure.

We note that there are many variants of the Hough transform, and the notation in the literature is inconsistent. The particular variant of the algorithm we consider, and the notation we will use to describe it, is most similar to Merlin and Farber’s [69], in which shapes are constituted of edge points. Edge points are simply the dark pixels in our one-bit representation of shapes. Suppose we have a candidate shape \( C \) defined as:

\(^1\) For those rare petroglyphs that can be processed without human intervention, there is uncertainty introduced by camera angle, focal length, etc.
and we want to find the best fit of a query shape $Q$ defined in the same way as $C$. That is, given a reference point $R$ in $Q$, to find the best point $R'$ in $C$, if we put $C$ onto $Q$ (where only translation in the plane is allowed) and points $R$ and $R'$ coincide, then the number of matched edge points would be the maximal.

For clarity, we use a very simple example to illustrate the algorithm. Figure 2.5 shows a query shape $Q$ and a candidate shape $C$. Note that the shapes can be disconnected, as in $Q$.

![Figure 2.5: Toy examples of a query Q and a candidate match C. Each cell is a pixel, and the dark colors denote edge points of shapes.](image)

As shown in Figure 2.6, the first step is to mark a reference point $R$ in $Q$ (usually the center of mass of all edge points) and rotate the edge points of $Q$ around $R$ by 180° ($left$ and $right$ of Figure 2.6). We then draw vectors from $R$ to each edge point ($right$ of Figure 2.6). These vectors form a “star-like” pattern which we will use to determine the best fit of $Q$ in $C$. 

\[
C_{[x,y]} = \begin{cases} 
0 & \text{if } [x,y] \text{ is an edge point} \\
1 & \text{otherwise} 
\end{cases}
\]
Figure 2.6: *left and center* The shape $Q$ is rotated $180^\circ$ around center of mass $R$. *right* Four vectors of $Q$ form a “star pattern”.

To find the best alignment of $Q$ to $C$, together with a numeric evaluation of their similarity, we do the following. The “star” vectors are superimposed on each edge point of $C$ (as shown in Figure 2.7.*left*). An accumulator matrix $A$ of the same dimensions as $C$ is used to record the number of vector ends (i.e. the arrowheads) that fall into each cell (Figure 2.7.*right* shows the final accumulator).

![Figure 2.7: Placement of vectors on each edge point of $C$ (left) and the final accumulator $A$ (right).](image)

The cell in $A$ with the maximal value is the best point $R'$ we want to find, and its value equals the maximal number of edge points that can be matched between $Q$ and $C$. This is 3 in our example. Note that while $R$ is the center of mass of $Q$ by definition, point $R'$ is not necessarily the center of mass of $C$.

Based on this maximal value, we can further obtain the *minimal unmatched edge points* (MUE) of $Q$. This is simply the number of edge points in $Q$ minus the number of
maximal matched points. This $MUE$ can be used as a distance measure. In our toy example, with similar shapes, its value is 1. If $Q$ were exactly the same as $C$, the $MUE$ would be 0, meaning $D(Q,C) = 0$. As we shall later see, it can be useful to normalize and adjust this number before using it as a distance measure.

For concreteness we show the algorithm to compute the minimal unmatched edge points in Table 2.1.

Table 2.1: The minimal unmatched edge points ($MUE$) from $Q$ to $C$.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>[MUE] = Classic_GHT ($Q$, $C$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(Rx, Ry) \leftarrow$ center of mass of $Q$;</td>
</tr>
<tr>
<td>2</td>
<td>foreach edge points $(x,y)$ in $Q$</td>
</tr>
<tr>
<td>3</td>
<td>$x \leftarrow 2\times Rx - x; \ Vx \leftarrow x - Rx$;</td>
</tr>
<tr>
<td>4</td>
<td>$y \leftarrow 2\times Ry - y; \ Vy \leftarrow y - Ry$;</td>
</tr>
<tr>
<td>5</td>
<td>add $(Vx, Vy)$ to the set $Vectors$;</td>
</tr>
<tr>
<td>6</td>
<td>endfor</td>
</tr>
<tr>
<td>7</td>
<td>Initialize a matrix $A$ with the same size of $C$ to 0;</td>
</tr>
<tr>
<td>8</td>
<td>foreach edge points $(x,y)$ in $C$</td>
</tr>
<tr>
<td>9</td>
<td>foreach vector $(Vx, Vy)$ in $Vectors$</td>
</tr>
<tr>
<td>10</td>
<td>$A(x + Vx, y + Vy)++;$</td>
</tr>
<tr>
<td>11</td>
<td>endfor</td>
</tr>
<tr>
<td>12</td>
<td>endfor</td>
</tr>
<tr>
<td>13</td>
<td>$MUE \leftarrow$ number of edge points of $Q$ - $\max(A)$;</td>
</tr>
</tbody>
</table>

If $Q$ and $C$ have $S\times S$ pixels, and we denote the number of edge points in $Q$ and $C$ by $N_Q$ and $N_C$ respectively, then the time complexity of this algorithm is $O(N_Q \times N_C + S^2)$.

2.3.2. A New Cell Incrementation Strategy

The classic GHT algorithm can be seen as a cell value incrementation process of the accumulator (as reflected line 8-12 in Table 2.1), and we need to wait for all of the incrementation to finish before we can obtain the value for any particular cell. Here we propose a new cell value incrementation strategy which allows obtaining the cell values
one by one. This will allow us, for the first time, to use a lower bounding strategy for the GHT.

Instead of superimposing vectors on edge points and increasing the value of the corresponding cell, we reverse this process by checking all positions that are possible to increase the value of one particular cell. To achieve this, we need to reverse the direction of the vectors.

Figure 2.8 shows this simple idea (using the same example as in the last section): first we draw vectors from $R$ to each edge point of $Q$, but without rotating $Q$ (left); if we want to calculate the value of a particular cell, say, the one at the third row and second column, we superimpose all vectors onto that cell (right). Then we check every cell with a vector falling into it: if this is also an edge point, we increase the cell value by 1 (because it is guaranteed, when using the classic GHT, that one vector superimposed on this edge point will fall into the target cell). Finally, after checking four cells, we obtain the value 2 for this cell.

![Figure 2.8: Four vectors of $Q$ (left) and placement of vectors on one cell of $C$ (right).](image)

It is obvious that our new cell value incrementation strategy is equivalent to the classic one. However, this strategy has one advantage in that it allows for the implementation of the cell incrementation process in parallel, which avoids nesting for-
loops in the classic GHT (line 8-12 in Table 2.1). In this paper, however, we are not
going to discuss this. We will instead utilize the nice property “obtaining cell value one
by one” as a base to explore a lower bound of minimal unmatched edge points in the next
two sections.

2.3.3. The Intuition behind Lower Bounding

As noted above, the time complexity of the GHT is quite high, and this limits its
applicability for larger datasets. The classic data mining solution to the problem of time
consuming distance measures is to find an efficiently computable tight lower bound to
the distance measure, and to use this bound to cheaply prune off unpromising candidates
[56].

We are now in a position to show the first known lower bound of the GHT-based
distance. Our idea is based on extracting one-dimensional “signatures” from the two-
dimensional query and candidate images. While we extract signatures from both the rows
and columns, for ease of exposition we begin by showing just the column signature,
which we denote as $\text{SigC}_x$.

For a candidate shape $C$ with $m$ rows and $n$ columns, we have:

$$\text{SigC}_x = \{ \sum_{i=1}^{m} C_{[1,j]}, \sum_{i=1}^{m} C_{[2,j]}, \ldots, \sum_{i=1}^{m} C_{[n,j]} \}$$

In other words, we are simply counting all of the edge points in each column of $C$.
For example, the truncated-corner square shape shown on the Figure 2.9.right has $\text{SigC}_x$
$= \{0,0,0,3,2,2,2,3,0,0,0\}$. 
We can extract “signatures” from shapes by summing up the number of edge points in each column.

We can extract these signatures as part of the preprocessing of the images, and store them in an index. At query time, we can use an identical technique to extract a signature, $\text{Sig}_Q$, from the query image $Q$. As shown in Figure 2.10 the only difference is that we truncate any leading or trailing 0’s from the $\text{Sig}_Q$ signature.

As it happens, the MUE distance in this case is 4, a number we can compute using the algorithm in the previous section. However, we can compute a lower bound to this value by looking at just the respective signatures.
We can obtain the intuition behind the lower bound by imagining that \( Q \) “wants” to match perfectly to \( C \), with no missing edge points. As we place “star” vectors to one cell on the center column of \( C \), if \( Q \) “wants” all vectors to fall into edge points of \( C \), a necessary, but not sufficient, condition for this to happen is that the number of vectors falling into each column is less than or equal to the number of edge points in that column. This is equivalent to checking whether each value in a \( \text{Sig}Q_x \) cell is less than or equal to the corresponding cell in \( \text{Sig}C_x \) (as shown in Figure 2.10).

Referring to Figure 2.10, we can see that in the slot \( \text{Sig}Q_{x_1} \) we need two edge points, and the corresponding slot in \( \text{Sig}C_{x_1} \) actually has three. There is no penalty for \( \text{Sig}C_x \) having a surfeit of edge points. In the next slot \( \text{Sig}Q_{x_2} \) we need two edge points, and the corresponding slot in \( \text{Sig}C_{x_{i+1}} \) has the two required edge points.

However, in the slot \( \text{Sig}Q_{x_3} \) we need four pixels, but the corresponding slot in \( \text{Sig}C_{x_{i+1}} \) has only two pixels. Thus, we are guaranteed that no matter how the pixels are arranged, this column will contribute at least two to the number of missed edge points in the accumulator. As we continue, we find that neither of the two remaining slots contributes to the lower bound, because in each case there are at least enough pixels in \( \text{Sig}C_x \) to satisfy \( \text{Sig}Q_x \). Thus, we can say that in this alignment, the lower bound \( \text{LB}(\text{Sig}Q_x, \text{Sig}C_{x_{[4:8]}}) = 2 \).

Note that this lower bound is only for the particular alignment shown in Figure 2.10; if we had shifted \( \text{Sig}Q_x \) one to the left, the lower bound would be 12, and if we had shifted \( \text{Sig}Q_x \) one to the right, the lower bound would also be 12. If we test all alignments,
we must choose the smallest value discovered as the true lower bound for the columns, which we denote as \( LB(SigQx, SigCx) = 2 \).

Finally, as hinted at above, we can do the same thing for the rows, using \( SigQy \) and \( SigCy \). The final global lower bound to \( D(Q, C) \) is then simply the larger of the two individual lower bounds.

### 2.3.4. A Formal Description of the Lower Bound

We expand the intuition presented in the last section to introduce a formal description of the lower bound. We again begin by considering the lower bound for just the columns. The algorithm is formalized in Table 2.2, which takes in a query shape \( Q \) and the column signature of candidate shape \( C \). As described in the previous section, to obtain \( LB(SigQx, SigCx) \), we need to shift \( SigQx \) from the left to the right of \( SigCx \) by aligning the center of mass of \( SigQx \) to each cell of \( SigCx \) (lines 5, 7 and 8 of Table 2.2). In each alignment, we calculate the lower bound for each column of \( C \). Note that when some cells of \( SigQx \) shift out of \( SigCx \), the edge points in these cells cannot find points in \( C \) to match them and then all contribute to the number of missed points (line 9-10 of Table 2.2). Finally, \( LB(SigQx, SigCx) \) is the minimal value of all of these lower bounds (reflected in line 21-23 of Table 2.2).

One important optimization we use here is early abandoning. When calculating the lower bound for a column, if the number of missed points exceeds the current best (smallest) lower bound, we can stop calculations and shift to the next position (line 17-19 of Table 2.2). For a better pruning, we can align \( SigQx \) and \( SigCx \) by their centers of mass first, and then shift stepwise to two sides (omitted in Table 2.2 for brevity).
Table 2.2: Algorithm to calculate the column lower bound of GHT by giving the query shape $Q$ and column signature of candidate shape $C$.

**Procedure** $[LB_x] = LB_{GHT}(Q, SigCx)$

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$SigQx \leftarrow$ column signature of $Q$;</td>
</tr>
<tr>
<td>2</td>
<td>$LBx \leftarrow$ number of edge points in $Q$;</td>
</tr>
<tr>
<td>3</td>
<td>$Rx \leftarrow$ center of mass of $SigQx$;</td>
</tr>
<tr>
<td>4</td>
<td>$left \leftarrow Rx - 1$;</td>
</tr>
<tr>
<td>5</td>
<td><strong>for</strong> $i \leftarrow 1$: length($SigCx$)</td>
</tr>
<tr>
<td>6</td>
<td>$missed \leftarrow 0$;</td>
</tr>
<tr>
<td>7</td>
<td><strong>for</strong> $j \leftarrow 1$: length($SigQx$)</td>
</tr>
<tr>
<td>8</td>
<td>$k \leftarrow (i - left) + (j - 1)$;</td>
</tr>
<tr>
<td>9</td>
<td><strong>if</strong> $k &lt; 1$ $</td>
</tr>
<tr>
<td>10</td>
<td>$missed \leftarrow missed + SigQx[j]$;</td>
</tr>
<tr>
<td>11</td>
<td><strong>else</strong></td>
</tr>
<tr>
<td>12</td>
<td>$delta \leftarrow SigQx[j] - SigCx[k]$;</td>
</tr>
<tr>
<td>13</td>
<td><strong>if</strong> $delta &gt; 0$</td>
</tr>
<tr>
<td>14</td>
<td>$missed \leftarrow missed + delta$;</td>
</tr>
<tr>
<td>15</td>
<td><strong>endif</strong></td>
</tr>
<tr>
<td>16</td>
<td><strong>endif</strong></td>
</tr>
<tr>
<td>17</td>
<td><strong>if</strong> $missed &gt; LBx$</td>
</tr>
<tr>
<td>18</td>
<td><strong>break</strong>;</td>
</tr>
<tr>
<td>19</td>
<td><strong>endif</strong></td>
</tr>
<tr>
<td>20</td>
<td><strong>endif</strong></td>
</tr>
<tr>
<td>21</td>
<td><strong>if</strong> $missed &lt; LBx$</td>
</tr>
<tr>
<td>22</td>
<td>$LBx \leftarrow missed$;</td>
</tr>
<tr>
<td>23</td>
<td><strong>endfor</strong></td>
</tr>
<tr>
<td>24</td>
<td><strong>endfor</strong></td>
</tr>
</tbody>
</table>

In summary, we have:

$$LB(SigQx, SigCx) = \min_{i=1}^{\text{length}(SigCx)} LB(SigQx, SigCx[i-left:i-left+\text{length}(SigQx)-1])$$

To get the final lower bound, we simply run the algorithm in Table 2.2 again, this time with $SigCy$ instead of $SigCx$, and with all column operators changed to row
operations. After then calculating $\text{LB}(\text{Sig}_Q, \text{Sig}_C)$, the final lower bound $\text{LB}(Q, C)$, is simply $\max[\text{LB}(\text{Sig}_Q, \text{Sig}_C), \text{LB}(\text{Sig}_Q, \text{Sig}_C)]$.

The time complexity of our lower bound algorithm is $O(S^2)$. Note that it is independent of the number of edge points in images. As we shall show in Section 2.4.4, a similarity search using the lower bound achieves a one to two order of magnitude speed-up.

### 2.3.5. Variants on the Basic Distance Measure

While the $\text{MUE}$ is in itself a useful distance measure, it is helpful to consider slight variations of it to enable higher-level data mining algorithms. Note that in every case, we can still use the lower bound technique to speed up the high-level data mining algorithms. Below we consider three useful variants, and in the next section we empirically evaluate them.

**Query-by-Content**: In the simple examples we have considered thus far, we have implicitly assumed that the number of edge points in $Q$ and $C$ was the same. While $\text{MUE}$ is surprisingly robust to small deviations from this assumption (say, less than a factor of two differences) it is clear that it has a bias. In particular, images that have relatively numerous edge points simply tend to be somewhat similar to everything. Since any large collection of images will invariably contain a few of these “rich” images, they can distort the results of any nearest neighbor searches. To mitigate this problem we define the nearest neighbor distance from $Q$ to $C$ as:

$$D_{nn}(Q, C) = \begin{cases} 
\frac{1}{N_Q - \text{MUE}(Q, C)} & \sqrt{N_C / N_Q} \quad \text{if } N_C > N_Q \\
\frac{1}{N_Q - \text{MUE}(Q, C)} & \text{otherwise}
\end{cases}$$
Note that we do not use \( MUE \) directly, but the inverse of \( "N_Q - MUE" \) (i.e. \( \text{maximal matched edge points} \)). The term \( \sqrt{N_C/N_Q} \) is an explicit penalty for the problem \( N_C >> N_Q \). Note that we can still use the lower bound of \( MUE \) to lower bound \( D_{nn}(Q,C) \).

**Clustering:** The \( D_{nn} \) measure is perfect for similarity searching, which requires one-to-all matching. However, clustering requires all-to-all matching. In this case, with all things being equal, the \( D_{nn} \) measure would be biased into claiming that two images with many edge points are more similar than two images with few edge points. We can use \( D_{clustering}(Q,C) \) to compensate for this:

\[
D_{clustering}(Q,C) = \sqrt{N_Q \times N_C} \times [D_{nn}(Q,C) + D_{nn}(C,Q)]
\]

**Finding Motifs:** Many data mining algorithms explicitly require a distance measure that obeys the triangular inequality. As a concrete example, we recently introduced an efficient and exact algorithm for finding motifs (approximately repeated patterns) [71], which makes no assumptions about the data or distance measure, other than the triangular inequality. We can modify \( MUE \) to obtain such a distance with:

\[
D_{motifs}(Q,C) = (N_Q + N_C)/2 - (N_Q - MUE(Q,C))
\]

**Proof that \( D_{motifs} \) obeys the triangular inequality**

Here we introduce a new notation \( QC_{mn} \), which is the set containing the maximal number of edge points that match between \( Q \) and \( C \), and \( |QC_{mn}| \) is the size of the set.

Now we can rewrite the motif distance measure as:

\[
D_{motifs}(Q,C) = (N_Q + N_C)/2 - |QC_{mn}|
\]

Given any three images \( A, B \) and \( C \), we’re going to prove:

\[
D_{motif}(A,B) + D_{motif}(B,C) \geq D_{motif}(A,C)
\]
By the distance definition we have:

\[ \left| (N_A + N_B) / 2 - |ABm| \right| + \left| (N_B + N_C) / 2 - |BCm| \right| \geq \left| (N_A + N_C) / 2 - |ACm| \right| \]

That is:

\[ N_B \geq |ABm| + |BCm| - |ACm| \quad (*) \]

To prove this, we use the following two lemmas.

**Lemma 1**: \(|ABm \cap BCm| \leq |ACm|\)

The proof is trivial, because:

\[ ABm \cap BCm \subseteq A, \ ABm \cap BCm \subseteq C \]

And \(AC_m\) contains maximal matched edge points between A and C, so:

\[ ABm \cap BCm \subseteq ACm \]

Then we obtain:

\[ |ABm \cap BCm| \leq |ACm| \]

**Lemma 2**: \(|ABm \cap BCm| \geq |ABm| + |BCm| - NB\)

\[ |ABm \cap BCm| = NB - |ABm \cup BCm| \]
\[ = NB - |ABm \cup BCm| \]
\[ \geq NB - |ABm + BCm| \]
\[ \geq NB - [(NB - |ABm|) + (NB - |BCm|)] \]
\[ \geq |ABm| + |BCm| - NB \]

Combine Lemma 1 and 2, we can easily obtain the inequality (*). \(\square\)

### 2.4. Experimental Results

We have designed all experiments such that they are not only reproducible, but easily reproducible. To this end, we have built a webpage [103] which contains all datasets and code used in this chapter, together with spreadsheets which contain the raw numbers displayed in all of the figures. All of the experiments are performed on a computer with an Intel i7-920 processor and 6.0GB of DDR3 memory.
2.4.1. Evaluation of Utility

We begin with simple sanity checks. We took a collection of petroglyphs from the Southwest USA and extracted fourteen images that would reasonably be grouped into seven pairs. Figure 2.11 shows the clustering obtained by our distance measure.

Figure 2.11: (left) A group-average linkage hierarchical clustering of typical Southwestern USA petroglyphs, with the $D_{\text{clustering}}$ measure. (right) While the dendrogram to the left shows the full resolution images for clarity, the images input to the distance measure have been binarized, thinned and scaled to fit into a 30 by 30 bounding rectangle.
Not only does the measure correctly group the seven pairs, but the higher level structure of the dendrogram correctly groups the images into Bighorn Sheep/Anthropomorphs/Atlatls\(^2\). Note that due to the thinning preprocessing step, the measure seems invariant to the hollow/solid nature of the Atlatls.

In the 1920’s Dr. Stephen Chauvet noticed that many of the petroglyphs discovered on Easter Island showed humans in poses very similar to petroglyphs created by the Harappa culture (in what is now modern-day Pakistan). He noted these similarities in his 1935 text [20], which inspired a flurry of speculation about the origin of the Easter Island peoples\(^3\). It is natural to ask if our proposed distance measure could have “noticed” this similarity. This is a very difficult challenge for a distance measure, because the Harappa culture used stick-figures, whereas the Easter Island petroglyphs used highly stylized outlines. Nevertheless, as we can see in Figure 2.12, our method can capture the intuitive similarities.

![Dendrogram](image)

Figure 2.12: The GHT distance is able to find the intuitive similarity between pairs of anthropomorphic figures, in spite of the different styles of representations.

\(^{2}\) An Atlatl is a spear-throwing device.

\(^{3}\) DNA analyses now show that this speculation was wrong; the Easter Island people are descended from Polynesians.
2.4.2. Finding Motifs in Large Collections of Petroglyphs

We tested whether our distance measure could potentially find meaningful motifs. We first scanned pages 33 to 132 (81-87 were skipped) of the book [85] which contains collections of Indian rock art in southern California, then cut them into 2,852 images to make one petroglyph per image.

With 2,852 images in the database, there are 4,065,526 possible pairs that might be motifs. Figure 2.13 shows a histogram of all pair-wise distances of these pairs. The mean distance between two objects is approximately 120, and we can see visually that we can expect few if any pairs to be closer than 40, a value we denote the motif cutoff.

![Motif Cutoff](image)

Figure 2.13: (left) Five representative motif pairs from the top 52 motifs. (right) Histogram of the pair-wise distance $D_{\text{motifs}}$ in 2,852 petroglyphs.

In Figure 2.13 we also show five representative examples of motif pairs which have a distance to each other of less than the cutoff. A total of 52 pairs, or 0.00128% of the possible candidates, passed this test. Note that these motifs are subjectively very similar. Furthermore, some of the examples show useful invariances. For example, some are invariant to solid/filled boundaries and the “cross” example is invariant to an extra crossbeam.

More interestingly, some of these motifs are in fact known (or at least, heavily speculated) to be true meaningful motifs. For example, Patterson and others have claimed
that the dumbbell shape was used by both Hopi and Navajo tribes to indicate “a conference took place here” [76].

Because of the triangular inequality of $D_{motifs}$, we can combine it with the algorithm recently published in [71] to efficiently find a pair of images whose distance is the smallest in a given dataset. For this petroglyph dataset, when using brute force, it takes nearly 14 hours to discover the top motifs, but it takes less than 18 minutes with our algorithm. Given that it can take months or years to collect a significant number of petroglyphs, it might be argued that there is little effective difference between minutes and hours. However, we are currently building an interactive tool for anthropologists that will allow queries such as “find all motifs within 10 miles of this location” and “find all motifs within 100 miles of this location, with tentative dates post 1100 AD”. If we can support answering such queries in tens of minutes, a researcher can interact with the system and frame/modify hypotheses in a single session. A system that takes tens of hours would be frustratingly slow to use. Furthermore, as we will show in Section 2.4.4, brute-force motif discovery in very large datasets (containing more than one million objects) is truly intractable.

2.4.3. Evaluation of Accuracy

Beyond the 2,852 petroglyphs considered in the last section, there are currently no large collections of objectively labeled petroglyphs (although we will plan to use the human computation tool discussed in Section 2.2.1.1 to create a public database of one million petroglyphs). Therefore, in this section we will test four publicly available
datasets that are very similar to (at least some kinds of) petroglyphs. With these experiments we intend to show:

- Competitive or superior accuracy for query-by-content compared to some state-of-the-art algorithms.
- Relative insensitivity to the amount of downsampling, and that we can learn a good value from training that. These two facts show that our method does not rely on careful parameter tuning.
- As claimed in Figure 2.4, very high resolution imagery hinders rather than helps accuracy.

The first dataset is the NicIcon dataset [73], which contains 24,441 images from the 14 categories shown in Figure 2.14. Thirty-three participants were asked to sketch these icons in different sizes (small, medium and large) and a digital tablet was used to record the data (spatial, time and pressure coordinates). Note that contrary to the original intention for the data and subsequent algorithms, our algorithm only considers the shape (offline data), and completely ignores pen speed and pressure information (online data).

![Examples of 14 categories from NicIcon dataset.](image)

Figure 2.14: Examples of 14 categories from NicIcon dataset.

We did both writer dependent (WD) and writer independent (WI) tests, in both cases, randomly choosing 60% of the data as the training set and the rest as the testing set, the same division as used the original paper [73].
The original offline data is in 234×234 pixels, and keep in mind that we need to first set the amount of downsampling (the only user-specified parameter of our algorithm). Using the training data alone, we tested nine resolutions from 5×5 to 80×80 for both WD and WI tests by the simple one-nearest-neighbor classifier. Figure 2.15 shows the results.

![Figure 2.15: Error rate vs. Resolution. Downsampling test on training set of the WD and WI tests in 9 resolutions. Error rate makes little difference once the resolution is larger than 10×10. Note WI is better than WD here, because the training set of WI contains only 60% of writers while that of WD contains all. This plot suggests that the sampling rate is not critical. The error rate only increased significantly when the resolution was reduced to 5×5, which is clearly highly undersampled for any non-trivial dataset.](image)

We then repeated the experiment on the whole dataset with the downsampling size of 20×20 pixels, which achieved the best error rate in our previous test, and obtained an error rate 4.78% for WD and 8.46% for WI. The dataset creators tested on the online data using three classifiers [73]: the multilayered perceptron, the linear multi-class SVM classifier and a Dynamic Time Warping Based (DTWB) algorithm. The reported error rate for WD varies from 1.94% to 15.61% and 5.3% to 20.01% for WI. Only the DTWB is better than our method, but recall that the DTWB had access to information about the
pen speed, pen pressure, and the direction in which the lines were drawn, all of which were unknown to our algorithm. While the original authors do not measure time for classification, each comparison with the DTWB measure requires DTW calculations to be performed a number of times which are quadratic in the number of line strokes (i.e., the number of pen-ups) in each image, which is clearly very expensive.

We also tested without any downsampling, and the error rate increased dramatically: 31.75% for WD and 35.75% for WI, even worse than the ultra-low resolution of 5×5. This verifies our analysis in Section 2.2.

Another petroglyph-like dataset is introduced by Khosravi and Kabir [57]. This is a very large dataset of handwritten Farsi digits extracted from about 11,942 registration forms. They obtained 102,352 binary images of Farsi digits, and chose 60,000 for training and 20,000 for testing (see samples in Figure 2.16).

![Sample digits from Farsi dataset](image)

Figure 2.16: Sample digits from Farsi dataset. Note: numbers 2, 3 and 4 are very similar (3 and 4 in the third row are even impossible for humans to distinguish); some digits have different styles (4 and 6); some digits are in bad quality (7, 8 and 9 in the third row).

The size of images in the Farsi dataset is smaller than in the NicIcon dataset: the minimum bounding rectangle (MBR) of the largest digits is 54×64 pixels. We also did a
downsampling test first on the training set from 5×5 to 30×30 pixels, using a one-nearest-neighbor classifier. The results are shown in Figure 2.17.

![Graph showing error rate vs. resolution](image)

**Figure 2.17:** Error rate vs. Resolution. Downsampling test on the training set of Farsi dataset in four resolutions. Note that the error rate varies little when the resolution is greater than 10×10.

With the resolution of 20×20 (the best one in the downsampling test), we obtained an error rate of 4.54% for the whole dataset. Borji et al. [15] performed extensive empirical tests on this dataset, testing multiple algorithms, 3-NN, ANN, $\text{SVM}_{\text{polynomial}}$, $\text{SVM}_{\text{linear}}$ and $\text{SVM}_{\text{RBF}}$, each with four parameter choices (two choices of filters times two numbers of orientations). Of the twenty reported error rates, the mean was 8.69%, and only four combinations beat our approach, with a best performance of 2.36%. However, it is important to note that in addition to the two explicit parameter choices, there are at least four other parameters set “in the background” here.

We performed the third test on an old handwritten musical symbol dataset [35]. These symbols were extracted from a collection of old music scores (17\textsuperscript{th}-19\textsuperscript{th} centuries). Figure 2.18 shows samples of 2128 clefs from 24 different authors, illustrating the large deformations and variations among different writing styles.
The authors of [33] first extracted some features from each symbol and used a Dynamic Time Warping Based (DTWB) algorithm as the classifier. To achieve the rotation invariance, features for all considered orientations need to be extracted and compared. Because such a DTW measure is time consuming, they chose one representative (the data sample with the minimum mean distance to the rest of the samples from the same class) for each class (two for Alto clefs due to their huge variability as shown in the third row of Figure 2.18), so only four comparisons were required to determine the class for one input symbol.

We repeated the test in the same way, and this time we skipped the step of setting the downsampling size and simply used the resolution 20×20 (which achieved the best results in the previous two datasets). Four representative symbols for three classes obtained by the GHT measure are shown in Figure 2.19.
In Table 2.3, we compare the classification accuracy of four distance measures. Zernike moments (number of moments = 7) and ART descriptors (radial order = 2, angular order = 11) are also from [35], both of which are robust and invariant to scale and rotation.

Table 2.3: Comparison of the classification accuracy on 2128 clef symbols.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Zernike moment</th>
<th>ART</th>
<th>DTWB</th>
<th>GHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>65.07</td>
<td>72.74</td>
<td>95.81</td>
<td>87.12</td>
</tr>
</tbody>
</table>

Our GHT distance measure is competitive and only worse than the DTWB method, but we should note that we didn’t do any parameter tuning, while the number of features may be an important parameter to the accuracy of DTWB. In addition to this, with the tight lower bound we have on the GHT distance, it is fast enough to do a classic leave-one-out one-nearest-neighbor classification and a 99.58% accuracy was achieved.

An extension of this experiment is to add 1970 accidentals in four classes, which were drawn by eight different authors (shown in Figure 2.20). One may find some sharps and naturals are very similar and can be easily misclassified.

Figure 2.20: Sample symbols from 1970 accidentals. In each row, a printed accidental symbol is followed by six samples in the corresponding class. Four classes are Sharp, Natural, Flat, Double Sharp (from top to bottom).
In this test, we have eight representatives (four clefs and four accidentals) for seven classes, and eight comparisons are required to determine the class. The classification accuracy is compared in Table 2.4:

Table 2.4: Comparison of the classification accuracy on 2128 clefs and 1970 accidentals.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Zernike moment</th>
<th>ART</th>
<th>DTWB</th>
<th>GHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>43.97</td>
<td>52.26</td>
<td>89.55</td>
<td>62.29</td>
</tr>
</tbody>
</table>

Again, our method is only worse than DTWB. When we tested by the leave-one-out 1NN classifier, the accuracy was increased to 98.49% with an acceptable increment in time.

The final dataset we tested is an architectural symbol dataset [83] consisting of 7414 samples in 50 classes (the printed version is shown in Figure 2.21) by 21 users. Each user drew 25 symbols and over 11 samples per symbol.

Mas et al (the dataset creators) tried syntactic approaches [65][66] to recognize these symbols by an adjacency grammar, in which each symbol was described in terms of the primitives that form it and the relations among these primitives. The test was performed on a subset of 10 symbols, and an accuracy of 87.7% was obtained in [65].
Then they further constructed an incremental parser in [66] to improve the performance to 91.78%. However, its main drawback is the complexity. The time complexity of the incremental parser is $O(n^3)$, where $n$ is the larger number between the productions of the grammar and the number of primitives entered into the system. The performance of the parser also relies heavily on the specification of the grammatical rules, and the set of constraints that has been defined. In comparison, our efficiently computed GHT measure doesn’t depend on the number of classes or the relation between them, and it achieved a much better result of 99.29%.

The only experiment we found on the whole architectural symbol dataset was done by Fornés [36]. She compared the classification accuracy on the first five classes and all 50 classes by three methods, one of which (DTW-full) is the same as the one performed on the old handwritten musical symbol dataset we have discussed above. We repeated both tests by the GHT measure, again without any parameter tuning. All results are shown in Table 2.5.

Table 2.5: Comparison of the classification accuracy on architectural symbol dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Zernike moment</th>
<th>ART</th>
<th>DTWB</th>
<th>GHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy on first 5 classes (%)</td>
<td>62</td>
<td>61</td>
<td>97</td>
<td>86.45</td>
</tr>
<tr>
<td>Accuracy on all 50 classes (%)</td>
<td>26</td>
<td>38</td>
<td>87</td>
<td>53.33</td>
</tr>
</tbody>
</table>

The DTW-cyclic is a variation of DTW-full, in which only one feature vector is extracted from each orientation and so the computational cost is reduced. Our method is only worse than DTW-full, and the lower time complexity enables a leave-one-out 1NN
classifier. When this was used, we achieved 100% on the first five classes and 96.18% on all 50 classes.

Having shown that low resolution images can produce high accuracy in our domain and the relative insensitivity of the downsampling size, we have fixed the resolution to 30×30 pixels in all remaining experiments in this chapter.

2.4.4. Evaluation of Speed and Scalability

As noted in Section 2.2, while we currently have only thousands of petroglyphs, we expect to shortly have on the order of a million. Therefore, we tested our algorithm dataset containing more than one million objects. To make this possible, we made our own synthetic petroglyph dataset. We first obtained the twenty-two petroglyphs (samples are shown in the top row of Figure 2.22). Then ten volunteers were asked to duplicate the petroglyphs by drawing them with an HP pavilion tx2510us tablet PC. A total of 250 petroglyphs were created in this way as our basic dataset (samples are shown in the second row of Figure 2.22). We then applied a random second-order Polynomial Transformation to each image in the basic dataset to make [39 79 159 319 639 1,279 2,559 5,119] distorted copies of each (as shown in the third row of Figure 2.22). With this basic dataset, we finally created eight datasets from the size of 10,000 to 1,280,000.
We first did a leave-one-out one-nearest-neighbor test. For each dataset, we randomly picked an image as the testing sample, removed it from the dataset and found its nearest neighbor using our lower bound based algorithm. We repeated this process ten times; Figure 2.23 shows the result.

We can see that the range between the maximal and minimal time is relatively small. When viewed on a normal scale plot (see [103]), we can see that the average running time is linear to the size of the dataset. While this is a test of scalability, we note in passing that the accuracy of this 22-class problem is 100% for all experiments.
It is natural to ask how much of the effectiveness of the search can be attributed to our lower bound. We measured the pruning rate as

\[
pruning\text{rate} = 1 - \frac{\text{number of GHT calculations, lower bound search}}{\text{number of GHT calculations, brute force search}}
\]

for each of the 10 runs; the result is shown in Figure 2.24.

![Figure 2.24: Pruning rate of our lower bound algorithm on eight synthetic petroglyph datasets. For each dataset, maximal, average and minimal rates are reported. Note log scale is used in x axis.](image)

The results show that the pruning is extremely effective, particularly for larger datasets. The average prune rate exceeds 99.0% when examining 80,000 objects, and even the minimal prune rate is more than 96.9% at that point.

We then did the same experiment using the brute force algorithm. Figure 2.25 compares the percentage of execution time for our lower bound algorithm relative to the brute force algorithm. Notice that for the largest dataset, our lower bound time is only 2% of the brute force one.
In addition to query-by-content, we also tested our ability to find motifs. Compared to the tests in Section 2.4.2, our synthetic petroglyph datasets are much larger. Figure 2.26 shows the running time of finding motifs in these datasets.

A brute force algorithm to find motifs requires time quadratic in the size of the dataset. But from a normal scale plot (see [103]), we find that our algorithm scales linearly. This is because we only need to calculate a tiny fraction of the exact distance between two images: even for the smallest dataset with 10,000 objects, we can prune 99.84% of the calculations, and by the time we consider 1,280,000 images, we can prune more than 99.99% of the calculations. In Figure 2.27 we show the explicit speed-up over the brute force search. Even for the smallest dataset, our algorithm is 712 times faster and by the time we see the largest dataset, our algorithm is more than 100,000 times faster.
Figure 2.27: Speed-up of our lower bound algorithm against brute force algorithm of finding motifs in increasingly large petroglyph datasets. For the brute force algorithm, we only ran it on the 10,000 datasets and extrapolated other values. Note log scale is used in x axis.

While these results show that we can make the otherwise intractable task of finding motifs in large datasets tenable, they do not consider the effectiveness. Normally motif discovery cannot be evaluated directly in terms of accuracy, since we assume unlabeled data. However, since we actually know the labels in this case, we can measure the accuracy. For example, when testing the dataset with 80,000 petroglyph images (from 22 classes) over 100 runs on random sets of 80,000 objects (taken from a pool of 1280K), we found that on 99 occasions the labels agreed.
Chapter 3

Using CAPTCHAs to Index Cultural Artifacts

As we shall demonstrate, and as the reader may already be convinced of, in the vast majority of cases, the extraction of meaningful data from photographs of rock art is likely to resist efforts of automatic extraction from images for a long time. In this chapter we show that we can use CAPTCHAs, puzzles designed to tell humans and computers apart, to segment and index rock art. Unlike other CAPTCHAs which operate on inherently discrete data and expect discrete responses, our method considers inherently real-valued data and expects real-valued responses. This creates a challenge which we have overcome by using our GHT distance measure. We demonstrate our system is
capable of acting as a secure CAPTCHA, while producing data that allows for indexing the rock art.

3.1. Do Petroglyphs Allow Automatic Feature Extraction?

A fundamental assumption that motivates the introduction of CAPTCHAAs for annotating petroglyphs is that there is no automatic segmentation algorithm that can robustly segment rock art. To demonstrate this, we conducted a simple experiment on what is probably one of the most amiable images imaginable, the famous petroglyphs of Alta, Norway.

We took one image of a reindeer as shown in Figure 3.1.\textit{left}, and tried segmenting it with six different methods: the Sobel method, Prewitt method, Roberts method, Laplacian of Gaussian method, Zero-cross method and Canny method [108]. In each case we spent fifteen minutes adjusting the parameters to achieve the best (subjectively) feature extraction. The best result, using the Prewitt method is shown in Figure 3.1.\textit{right}.

Figure 3.1: \textit{(left)} Reindeer rock art from Norway, dating to 4200 to 500 BC. The rock carvings have been retouched in bright red by researchers, making them extremely high contrast. \textit{(right)} a segmentation of the image using the Prewitt method, carefully tuned.

Note that while our efforts have paid off in that we have captured much of the animal in question, we are missing a large section of the rump. What is worse, we have
many spurious lines corresponding to cracks in the rock. Of course, it is possible that a
more sophisticated algorithm could be tuned to do a better job; however, this tuned
version is unlikely to generalize to other petroglyphs. Furthermore, it is worth restating
that this example is among the highest contrast, cleanest examples of rock art.

3.2. A Sanity Check of Human Computation to Process Petroglyphs

The GHT (Generalized Hough Transform) distance measure for rock art we
proposed in Chapter 2, and thus all of the higher level data mining algorithms that build
upon it, explicitly assumes that the human computation step can meaningfully extract the
essential shape of the petroglyphs. It is not obvious that this is true. One might imagine
that the subjective notions of the humans in the loop might dwarf any true object
similarity that exists between the petroglyphs. We designed a detailed sanity check
eperiment to test this.

We found eight petroglyphs (examples e and f are technically pictographs) which
can objectively group into four pairs. In the case of examples a through f, there is strong
evidence based on co-location that they were created by the same hand.

We asked two volunteers, whom we denote SC and WY, to annotate the eight
petroglyphs. The volunteers were computer science graduate students, without any
special knowledge or training in either anthropology or art. Neither of the volunteers met
the other, or was allowed to see the other’s work. The annotation was conducted on a
desktop with a mouse. Both volunteers annotated one sample petroglyph provided in our
tool as the training before this experiment.
Figure 3.2 shows the eight original petroglyphs, and the transcribed skeletons produced by the two volunteers.

![Petroglyphs and Transcribed Skeletons]

**Figure 3.2:** *First row:* eight original petroglyphs (a to h). *Second row:* corresponding skeletons transcribed by the volunteer SC. *Third row:* skeletons transcribed by the volunteer WY.

Note that there is significant variation between the two volunteers’ output. The most notable of these is in the two deer denoted e and f. User WY transcribed these as stick figures; however, user SC transcribed them as outlines. Another obvious difference is that user SC explicitly added a stroke for the tail in the bighorn sheep shown in g and h, something that user WY did not see as necessary.

In retrospect, we believe that we can reduce some of this subjectivity by providing a slightly longer and more detailed training session for users. However, we obviously need our similarity measure to be robust to such human factors. We can test to see if it is by hierarchically clustering all sixteen transcribed petroglyphs. We used the clustering distance measure that we introduced in Section 2.3.5.
Figure 3.3: A group-average linkage hierarchical clustering of all sixteen transcribed petroglyphs. The results shown in Figure 3.3 are very reassuring. At the lowest branches of the dendrogram, each of the two transcribed versions of an original petroglyph is linked with each other. Likewise, at the next higher level of the dendrogram, both versions of the birds (lizards/deer/sheep) join together in a sub-tree before merging with the rest of the animals. This dendrogram shows that our distance measure is very robust to human variability and it can capture the essential character of shapes.

3.3. CAPTCHA-ROCK

The results in the previous two sections hint at the fact that:

- If we could only extract the “skeletonized” rock art data, a wealth of opportunities for anthropological data mining would open up.
- The extraction of meaningful data from photographs of rock art is likely to be beyond the capabilities of image segmentation algorithms.

With this in mind, we proposed to extract useful information from unconstrained images of rock art by turning the problem into a CAPTCHA [1]. CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are
tests given by a machine to ensure that a response is generated by a human, not a computer. The most familiar instantiation of them is a sequence of distorted letters that the user must reproduce. Figure 3.4 shows two examples.

Figure 3.4: Two examples of CAPTCHAs. In order to solve the CAPTCHA and get access to the next webpage (in this case, offering a free email account) the user must type in $28iVW$ and $jw62K$ respectively.

These CAPTCHAs operate on inherently discrete data (text, albeit distorted) and expect discrete responses (keystrokes); we can therefore use equality tests to decide if the test was passed, i.e. equals('28iVW', '28iVW')?

In contrast, our method will consider inherently real-valued data (photographs of rock art) and expects real-valued responses (mouse movements). We cannot expect to test for equality. This creates a significant challenge which we have overcome by using the distance measure introduced in Section 2.3.5 to test if a tracing of a petroglyph is close enough to a real pattern to indicate human intelligence.

While our ultimate goal is to introduce a method that will allow us to capture data from real photographs of petroglyphs, for ease of exposition we will begin discussing the problem as if our only intention were to produce an image-based CAPTCHA with artificial data.

3.3.1. A Simple Image-based Stickman CAPTCHA

It is simple to write a program to produce random instances of a “stick figure”; Figure 3.5 shows four examples:
Figure 3.5: Four examples of a parameterized Stickman.

To ensure each stickman is unique (with very high probability), we have parameterized the code. The following features are parameterized:

- The head size and aspect ratio
- The length of the humerus, forearm, femur, tibia and foot
- The angles of knees, elbows, ankle and torso (these may be asymmetric)

There are other elements of a human stick figure that we could represent and parameterize, but this simple model is sufficient for our purposes.

For reasons that we will see shortly, it is useful to ask what the average distance is between two randomly created figures under the GHT distance measure discussed in the previous section. To calculate this, we generated 1,000 pairs of stickmen and calculated the distance between each pair, summarizing the results in a histogram in Figure 3.6.

Figure 3.6: left) A pair of randomly generated stickmen. right) The distribution of GHT distances between 1,000 pairs of randomly generated stickmen.

If we instead produce random stickmen, and ask humans to trace their outline on the screen with the mouse pointer (as in Figure 3.7), we might expect the distances between the generated and traced outlines to be generally smaller.
To verify this, we generated 20 stickmen and asked volunteers to trace them. How well a person can trace the stickmen depends on their dexterity, input device, screen size, etc. Given these variations, we asked three volunteers to trace each stickman on their own personal machines. Figure 3.7 also shows the distribution of these distances.

Figure 3.7: (left) A randomly generated stickman in black and a human tracing of it in red. (right) The distribution of GHT distances between randomly generated stickmen and human tracings of them are shown with a finer bucket size (in red), because there is less data. The distribution of GHT distances between two randomly generated stickmen is shown for context (in blue).

It is easy to see that we could use these results to create a simple stickman CAPTCHA. We could produce a stickman, and ask the user to trace it. If a human traces the stickman, we can be near certain that the distance to the template will be less than 3 (from Figure 3.6). For simplicity, here we assume that the attacker has the code to produce the stickmen, and simply sends a random stickman as his guess. If that is so, his guess will almost certainly be greater than 3 (from Figure 3.6/Figure 3.7) and we can reject his attempt. Of course the attacker could use an image processing algorithm to produce a “customized guess”, and we could counter by imbedding the stickman in a field of distracters and distortions; however, as we shall see, an even better idea is to find “stickmen” in rock art, the subject of the next section.

3.3.2. The CAPTCHA-ROCK System: Extracting Data from Petroglyphs

Motivated by our experiences with the stickmen CAPTCHA, we can now ask: is it possible to design a CAPTCHA system which provides high security (serves as a
CAPTCHA) while collecting useful information about rock art (serves as a Human Computation tool)? The reCAPTCHA [2] proposed by Luis von Ahn is the ideal paradigm to follow. In this system, which is designed to transcribe degraded text from scanned books and newspapers, each test gives the user two words to recognize, one of which is a “control word”, whose answer is known, and the other of which is an unknown word. If the user can correctly type the “control word”, the reCAPTCHA assumes that the inputs come from a human and the answer for the “unknown word” is correct (or at least plausible). Once an “unknown word” receives enough "votes" from a same answer, it can become a “control word”. This system has already been used to transcribe several hundred million degraded words, which OCR systems failed to parse.

We can use the same idea to build our CAPTCHA-ROCK system. An example is shown in Figure 3.8, in which the user is asked to trace both petroglyphs correctly to pass the CAPTCHA.

Figure 3.8: A CAPTCHA-ROCK consists of two rock art images: one control image and one unknown image. Note that users do not know which one is which.

There are still three extra problems/questions we must solve to make this work:

1) How should we build the initial “control image” set?

To frustrate robots that break challenges by simple random guesses, the set of control words in reCAPTCHA contains more than 100,000 items. Do we also need such a
large “control image” set? We believe that the answer is no. We can frustrate an attacker that attempts to simply memorize the entire control image set (with solution tracings) by performing simple scale and translation operations to images. This means that even if the attacker's algorithm correctly detects which image is the control image (perhaps by color), the relative location of the petroglyph within the image may have changed just enough so that even if the attacker sends the right trace, it will be in the wrong location or at the wrong scale.

As we shall show in the next section, our initial experiments show that even for a control set containing only 143 images, by using these simple scale and translation operations, CAPTCHA-ROCK has a pass rate of 0.020% for attackers, even after the attacker has been given a careful tracing for each image in the control set.

2) Is one trace (i.e. one “solution”) per image enough?

Assume two histograms in Figure 3.7 have an overlap. To assure the low pass rate for robots, we have to set a smaller threshold. In this case a false negative occurs: some legitimate attempts by humans would be denied.

Our solution is to store multiple traces for one image. When a tentative solution is submitted by a user, we compare it to all traces and pass it if there is at least one distance below the threshold. Although human traces for the same image vary, by comparing to more than one "interpretation", the possibility of finding a close enough match increases. Note that this will not affect the pass rate for robot significantly. If a random trace is far from one trace, it is also far from other traces of the same image.
Based on our experiments which will be presented in Section 3.3.3, three traces per image improve the human pass rate, without helping the attackers.

3) When can we promote an “unknown image” to a “control image”? Once we have recorded three traces for an “unknown image”, we promote it into the control set. Note that as a control image, the CAPTCHA-ROCK system will obtain more traces for the image. What should we do with these additional traces? We could ignore them, we could add them to the original three traces, or we could temporarily merge the new offering with the other three, expunging the one that has the furthest average distance from the others. We leave these considerations for future work.

3.3.3. Quantitative Evaluation of the CAPTCHA-ROCK System

In this section, we intend to show that:

- Our CAPTCHA-ROCK is very easy for humans to solve (they have a high pass rate) and hard for robots (they have a low pass rate).
- Storing multiple traces for each image helps increase the pass rate for humans, while not affecting the pass rate for robots significantly.
- A small “control image” set is sufficient, at least to bootstrap the system.

We randomly chose 143 images from our rock art image database, and had four volunteers draw traces for each image. The volunteers worked completely independently of each other. We call this initial trace dataset \textit{Trace\_ini}.

For reasons we will see shortly, we performed two rounds of rescaling and 2-dimensional translation to each trace in \textit{Trace\_ini}. In the first round, we rescaled each trace image to 10%~50% of its original size, and translated it in 2 dimensions by
plus/minus 0–3 times of the size in the X and Y axis independently. We call this new dataset \textit{Trace_robot}. In the second round, every 4 traces of the same image in \textit{Trace_init} were performed by a same rescale and transition, and this dataset is called \textit{Trace_human}.

As we assume that there is no automatic algorithm for extracting rock art data, we need to come up with an attack model. We make the pessimistic assumption that the attacker has our entire 143 image database, together with a human trace for each image. These seems to suggest that if the attacker simply submits a random tracing, he would have a one in 143 chance of passing the test, but recall that the images have been rescaled/translated before being presented as a CAPTCHA. Such distortion means that even if the attacker happens to send the correct trace, it will probably not line up with the stored template, and will fail the test.

We first tested the system with one randomly chosen (of four possibilities) trace from \textit{Trace_human} for the “control image”. To model the attacks from robots, each time we picked one trace from \textit{Trace_robot} (but not those from the same person of the challenge trace). There were thus $3 \times 143$ tries for each challenge. As the human input, we picked the other three traces of the “control image” in \textit{Trace_human}.

Using a threshold of seven, only 41 of 245,388 robot tests could pass, a pass rate of 0.014%; whereas 1,632 of 1,716 human tests passed, a pass rate of 94.99%.

Then we tested with three traces for each image. As noted in Section 3.3.2, each input from the user was compared to three traces of the “control image”, and if its distance to \textit{any one} of them is below the threshold, the user passed the test. Each time we picked three traces of a same image from \textit{Trace_human} as traces of the “control image”.

63
To model the attacks from robots, each time we picked one trace from the fourth person in \(\text{Trace}_\text{robot}\). Thus, there were 143 tries for each challenge. As the human input we picked the remaining trace of the “control image” in \(\text{Trace}_\text{human}\).

Using the same threshold of 7, only 22 of 81,796 robot tests could pass, a pass rate of 0.020%; while only 5 of 572 human tests could not pass, a pass rate of 99.13%.

Although our pass rate for robots is slightly larger than the generally accepted figure of 0.01% [21], note that all results are based on the initial control image set, with only 143 images. We expect the robot rate to decrease with more data, while the human rate should stay almost constant. Further recall that these results assume the pessimistic and unrealistic assumption that the attacker has traces for the entire database.

### 3.3.4. Supporting Similarity Search

The major goal of the work in this chapter is to produce a dataset that will enable research by anthropologists. However, a minor goal is to produce a tool for non-specialists to query a database of petroglyphs. This tool could be used to support tourism [26], and to encourage an appreciation of indigenous people’s cultural achievements.

We envision the following scenario: *A hiker on a trail spots a petroglyph, and wants to know if it is known, and if so, what anthropologists and/or tribal historians have said about it. She photographs the petroglyph on her iPhone, traces the outline, and submits the query...*

In order for this query to return the correct answer, our system must have several invariances. Some are trivial, as we are operating on a binary representation of the data,
color and contrast invariance is automatically achieved. However, as shown in Figure 3.9, there need to be at least somewhat invariant in size, angle of view, etc.

Figure 3.9: (left) A petroglyph from Utah that has been indexed in our database. (right) An image of the same petroglyph found on Flickr.com. Could this image be used as a query to retrieve the anthropologist’s annotated version in our database?

To test the feasibility of this scenario we obtained several examples of images of petroglyphs that we know are in our database (referred to the “control image” set in Section 3.3.2), but which were taken on a different day, by a different person, with a different camera etc. To normalize our expectations we also obtained photographs of petroglyphs that are known not to be in our database.

We had volunteers trace these petroglyph images. Note that in each case, these volunteers had not seen the data in the database, and were not familiar with our project.

Recall from the previous section that each petroglyph in our data collection had been traced by four independent volunteers. This means that for each petroglyph we had four models we could use to index it. We could also choose to have only one model for each petroglyph instead, by either averaging all four, or choosing the most typical one.

---

4 For faint petroglyphs changing the contrast/color balance can enhance the petroglyphs visibility [58][64]
However, here we kept all four models, both for simplicity, and because (as we shall see) it is instructive.

Our small dataset in this preliminary experiment does not warrant calculating precision/recall or similar statistics. Instead, we show typical results, and archive all results at [104]. Figure 3.10 shows an example of a query using a (different photograph of) petroglyph that is in our database. The results are quite promising. Note that the query was taken from an image that was not as tightly cropped, and the user issuing the query (rightly or wrongly) traced a hook-like appendage on the left leg of the figure. Furthermore, note that among the four tracings in the database there is significant disagreement. For example, one individual did not trace the head as a circle. In spite of this, query-by-content is clearly successful in this example, as the first four matches are correct (the maximum possible).

![Figure 3.10: left) A query petroglyph that happens to be in our database and its tracing. right) The five nearest neighbors to the query; the first four all refer to the same image, the correct target.](image)

In Figure 3.11, we see two more queries for which the relevant petroglyph (traced from a different photograph) is known to be in the database. For the “wheel” the 1st, 2nd
and 4th matches are correct, and the two others are at least plausible. For the bighorn sheep petroglyph, the first four matches are correct (the maximum possible), and the 5th match is also plausible.

Figure 3.11: An abstract (top) and animal petroglyph (bottom) which had been traced and issued as queries to our database. The list of the five nearest neighbors to each are shown left to right.

Finally, we consider the more difficult case, queries for which we know the relevant petroglyph is not in the database. Here the judgment of quality is subjective. Note also that we might expect to do better and better at this case as the database grows larger and larger. In Figure 3.12 we show two queries and their best matches. In both cases the returned answers are reasonable.

Figure 3.12: left) An anthromorph is used as a query, and it retrieves another stylized human figure with similar limbs. right) A petroglyph of concentric circles retrieves a "sunburst".
Chapter 4

Mining Historical Manuscripts with Local Color

Patches

Initiatives such as the Google Print Library Project and the Million Book Project have already archived more than fifteen million books in digital format, and within the next decade the majority of world’s books will be online. Although most of the data will naturally be text, there will also be tens of millions of pages of images, many in color. While there is an active research community pursuing data mining of text from historical manuscripts, there has been very little work that exploits the rich color information which is often present. In this work we introduce a simple color measure which both addresses and exploits typical features of historical manuscripts. To enable the efficient mining of
massive archives, we propose a tight lower bound to the measure. Beyond the fast similarity search, we show how this lower bound allows us to build several higher-level data mining tools, including motif discovery and link analyses. We demonstrate our ideas in several data mining tasks on manuscripts dating back to the fifteenth century.

4.1. Introduction

In the last decade, millions of books, maps and historical manuscripts have been digitized and made freely available [48]. It is now clear that within a few years a significant fraction of world’s books will be online. While these digitized texts will be an invaluable resource for researchers to browse and search, we feel that the additional step of mining these manuscripts will reveal new insights, knowledge and historical context. In a similar vein, technology writer Kevin Kelly has observed, “the real magic will come in the second act, as each word in each book is cross-linked, clustered, cited, extracted, indexed, analyzed, annotated, remixed, reassembled and woven deeper into the culture than ever before” [55]. While this remark explicitly singles out text, a similar argument can be made for images. Clearly most of the data gleaned from scanned books will be text, but there will also be tens of millions of pages of images. Many of these images will defy automatic annotation for the foreseeable future, and CAPTCHA-based [106] or crowdsourcing efforts [50] are likely to have only limited impacts on these problems. A considerable fraction of the images may, however, be amiable to annotation by algorithms that can exploit the color information which is often available in historical manuscripts.
While there is much work on indexing images by color, most of it assumes that we are indexing atomic images, typically photographs. For example, much work has been done to support queries such as “find photos of sunsets” [62] or commercially useful queries such as “find shoes in this color” [59]. In contrast, here we are interested not in global color queries, but local region-of-interest queries, which we call patches. In Figure 4.1 we show some examples of the type of images we are interested in mining.

Figure 4.1: left) Two pages from The secret book of honour of the Fugger 1545-1548 [23]. right) Two pages from an 1834 text on fish [12].

Here we can imagine an historian might be interested in searching a massive archive to discover examples of the coat of arms of Lucas Fugger, which consists of a golden deer on a light blue background. In Figure 4.1 left we have a local patch that matches this description, but note that the global color of the page is dominated by other colors.

While there is much less work on local color matching, there is some. For example, in [87] the authors proposed a technique called Histogram Backprojection to determine the location of a query object in one image (provided that the query appears in that image), and in [24] the authors described a scale and translation invariant image retrieval system. However, these works, and their many extensions [67] only consider query-by-content for main memory search.
In contrast, we are interested in more general data mining for local color regions, and for datasets which are so large that they must be disk-resident. As we shall show, existing color matching measures work well for historical manuscripts, but the existing works are severely limited in their scalability, especially if we look beyond query-by-content and consider more complex data mining operations.

In Figure 4.1 we show an observation which we plan to exploit to mitigate the scalability problems. Note that while the leftmost page from each manuscript can be considered somewhat “dense”, the rightmost pages are relatively “sparse”. As we shall show in Sections 4.4 and 4.5, by adaptively exploiting this variability of density, we can prune off a huge fraction of the search space, and we can produce much smaller indices, both of which in turn can allow us to answer queries much more efficiently.

The rest of this chapter is organized as follows. In Section 4.2 we discuss the necessary background material. In Section 4.3 we introduce a lower bound that is at the heart of our search technique, and algorithms that exploit it are introduced in Section 4.4. Section 4.5 sees extensive empirical evaluation of our ideas. In Section 4.6 we propose a novel image retrieval framework which utilizes both the existing web image search and our color distance measurement.

4.2. Notation and Background

Before describing our algorithm, we first give the following definitions.

In the domain of our interest, the natural atomic unit is a single image or page:
**Definition 1:** A *Page* is a color image $P$, which is represented as a $H \times W \times 3$ matrix where $H$ and $W$ are the height and width of the page, respectively. Each pixel of the page corresponds to a triplet $(R, G, B)$ in the matrix $P$, denoting the value in the red, green and blue channels, or the pixel may contain $NaN$ to indicate transparency.

We use the RGB color space because it is familiar and we found empirically it works very well in our domain. However, all the lower bounds and algorithms introduced in this work can work in any color space. Note that we allow the pixel to have a $NaN$ value indicating no color in that pixel. We use this value to replace the background color of the page at a preprocessing step. As we shall see later, removing the background color allows us to avoid the trivial solution of matching backgrounds when we want to find repeated patterns (motifs). Automatically finding the background color of an historical text is an important subroutine in other problems (OCR etc.) and is an essentially solved problem for all but the most pathological texts [38][42].

As noted in Section 4.1, we are not interested in page-level queries, instead we want to enable local region-of-interest queries, which are more challenging and useful. We define these lower level units as *color patches*:

**Definition 2:** A *Color Patch* is a rectangular window $CP$ in a page. $CP$ can be specified by four variables: the row $r$ and the column $c$ in $P$ of its top left corner, its height $h$ and width $w$. Then $CP = P(r:r+h-1, c:c+w-1,:)$. Here $1 \leq r \leq H-h+1$ and $1 \leq c \leq W-w+1$. 


Note that our basic primitive is a rectangle. We had considered allowing more
general shaped patches (circles, triangles, etc). However, as we shall show empirically,
rectangles allow the effective retrieval of shapes which are clearly non-rectangular (fish,
butterflies, coats-of-arms, etc), so we restrict our definitions to rectangles for simplicity.

Also note that we can consider a whole page as a color patch, as a special case.

Finally, we define the higher level unit that organizes pages, a book:

**Definition 3:** A Book is a set $B$ containing $N$ pages. Note that due to the
scanning process, and the handcrafted nature of some historical
manuscripts, the size of pages from the same book may not be exactly the
same.

Having this three-level hierarchy (color patch, page and book) in mind, we are
now in a position to define the nearest neighbor of a color patch.

**Definition 4:** Given a color patch $Q$ and a book $B$, the Nearest Neighbor
of $Q$ in $B$ is a color patch $NN$ with the same size of $Q$ and most similar to
$Q$. More formally, for any color patch $CP$ with the same size of $Q$ in $B$,
$\text{Dist}(Q, NN) \leq \text{Dist}(Q, CP)$.

In definition 4, we have not yet given an explicit distance measure between two
color patches. While a number of color descriptors (representations) exist, the Color
Histogram is one of the most widely used. It is easy to calculate, invariant to translation,
insensitive to small changes in viewpoint (angle and distance), and has been proven to be
effective in several content based image retrieval (CBIR) systems [34][86][87].
**Definition 5:** The *Color Histogram* of a color patch $CP$, denoted as $\text{Hist}(CP)$, is a vector which counts the frequency of each possible color in a given *color space*.

In a RGB color space which has 8 bits for each channel, the total number of colors is $(2^8)^3$. Maintaining so many colors is impractical and unnecessary. Furthermore, it is likely to *hurt* accuracy, since two colors that are imperceptibly different to the human eye may map to adjacent but different bins, thus accumulating significant distance. One solution to mitigate this problem is to use a “warping-like” measure [54]; however, this will require dramatically more computational power per distance calculation, and make it difficult to design an effective lower bounding search. Therefore, we use *quantization* to alleviate the abundance of colors problem.

In keeping with the majority of the literature, colors are quantized into $K$ *bins*. The value of $K (<< 2^{24})$ is a user-defined parameter choice, but the exact value does not matter too much from several dozens to several thousands [87].

In Figure 4.2 we illustrate the notations introduced thus far with a toy example.

Figure 4.2: An illustration of some basic notations. *left*) A 6×10 page showing a (slightly modified) Benin flag. The 4×3 green window on the left is one of its *color patches. right*) A matrix of the same size as the page, with the number indicating the bin_id of corresponding pixels. For simplicity, we set $K = 4$ and assign one bin for each color. Note that for surrounding transparent pixels, we set bin_id = 0. The *color histogram* can be easily obtained from this matrix, which is $\{8, 12, 1, 7\}$. 
Having determined that the color histogram is the most suitable descriptor for color patches, we must consider the most appropriate similarity/distance measure for them. Not surprisingly, lots of measurements for color histograms have been proposed in the last three decades [62][81][84], including simple ones such as Euclidean Distance and Histogram Intersection [87], and more complex techniques such as Histogram Quadratic Distance [34], Histogram Refinement [75], and Color Set Distance [86], etc. After extensive testing on our domain of interest we chose the Histogram Intersection, because of both its (relative) efficiency and accuracy. Note however that the focus of this paper is not to make an argument about the “best” color distance measure in general, but rather to show that we can adapt and augment a popular distance measure to solve data mining problems in a domain that has received surprisingly little attention.

**Definition 6:** Given a query color patch \( Q \) and a candidate color patch \( C \),

the Histogram Intersection between them is defined as:

\[
HI(Hist(Q), Hist(C)) = \sum_{i=1}^{K} \min(Hist(Q)[i], Hist(C)[i]) \tag{1}
\]

In the above equation, \( K \) is the number of bins in both Hist\((Q)\) and Hist\((C)\). The \( HI \) value tells how many pixels are in common (falling into the same bins) between \( Q \) and \( C \). We can simply use its inverse as a distance measurement\(^5\):

\[
Dist(Q, C) = \frac{1}{HI(Q, C)} \tag{2}
\]

\( ^5 \) For simplicity, Dist\((Q,C)\) is short for Dist\((Hist(Q),Hist(C))\), the same applies to HI\((Q,C)\) in the equation (2). Without ambiguity, we will use these notations in all the remaining text.
It may be noted that our proposed distance measure does not encode the spatial relationships of the colors within the query region. As it happens, it is this fact that allows us to create tight lower bounds to the measure, thereby enabling our fast search and mining algorithms. However, this is merely a fortunate side effect. The need for spatial relationship invariance emerged from conversations with domain experts. In Figure 4.3, we can see two motivating examples: While the parity of left/right orientation typically does have meaning for some heraldic shields, it is often ignored, as in the examples from a 16th century book shown in Figure 4.3 (left). Likewise, while most organisms are bilaterally symmetric, they may be illustrated at arbitrary orientations, as in the two examples show in Figure 4.3 (right).

![Figure 4.3: A section of images to demonstrate the need for spatial-location invariant distance measures. From left to right: in page 109 of [23], in page 108 of [23], an image of a Death’s Head Moth from 1849 [30], and one from 1860 [91].](image)

In spite of the above, there are some circumstances where color locality matters; Figure 4.4 shows some examples.

![Figure 4.4: A section of image pairs that are clearly distinct, but essentially indistinguishable under the distance measure introduced in equation (2).](image)
In the event that spatial sensitivity is necessary in a particular domain, we can achieve this by dividing the query region into (for example) four quadrants, and defining a new measure as the sum of the four local measures. However, because our domain experts, genealogists, an ichthyologist and an historian of science, did not find a compelling example where spatial sensitivity was critical to the success of query-by-content or a higher order data mining query in their domains, we defer a further discussion of this idea to an online appendix [105].

Our color measure is simple, and the domain of interest does feature the complex issues of staining, fading, degradation, and problems where very subtle distinctions are required. The reader may wonder if the proposed measure is powerful enough to be useful. We will forcefully assuage such doubts in the experimental section of this work. In the meantime we content ourselves with a simple demonstration. Color printing of images became possible only in the 19th century. Before that, color images were produced by using copper engravings to print in black ink, and these images were then hand-colored individually, in a process called aquatinting. We obtained two versions of a classic work on marine life by Louis Renard (b. ca. 1678.) [80]. The two versions were published 35 years apart, and almost certainly hand-colored by two different artists. Nevertheless, as we can see by the clustering of a subset of data from both texts in Figure 4.5, in each case a query image from one book does return the corresponding image from the other.
Figure 4.5: Five hand-colored images from the 1754 version of [80] and five hand-colored images from the 1789 version of the same text. An inspection of a higher resolution color version of this figure in [105] makes it clearer that these images are significantly different in both color and texture.

4.3. A Lower Bound to the Color Distance

We begin by describing the brute force algorithm to search for the nearest neighbor of a query color patch. This allows us to concretely define the search problem and motivates the need for a more efficient solution.

As shown in Table 4.1, the brute force algorithm compares the query with all color patches of the same size, and updates the best-so-far whenever a better match is found.

Table 4.1: The brute force algorithm to search the nearest neighbor of a user-given query $Q$ in a book $B$.

```
Algorithm $[NN] = NN_{BruteForce}(Q,B)$

1  best-so-far = INF;
2  $h = \text{height}(Q); \ w = \text{width}(Q);$
3  foreach page $P$ in $B$
4      $H = \text{height}(P); \ W = \text{width}(P);$
5      for row $\leftarrow 1:H-h+1$
6          for col $\leftarrow 1:W-w+1$
7              $CP = P(\text{row:row+h-1, col:col+w-1,:});$
8          if Dist($Q,CP) < best-so-far$
9              best-so-far = Dist($Q,CP$);
10                 $NN = CP;$
11          endif
12      endfor
13  endforeach
14 endforeach
```
As we can see in the nested for-loop at lines 5 and 6, we need \((H-h+1)\times(W-w+1)\) distance calculations for each page. If a book contains 500 pages with the size 2000×1500, and the size of a query color patch is 400×400 (all typical values), then the total number of color patches we need to check is more than **881 million**. If each distance calculation takes one microsecond, this translates to about 15 minutes. Note that for some problems this may actually be acceptable. Given that it may take hours to scan a single historical manuscript of value, for some data mining/search queries we may be willing to wait overnight for a gem of information that has evaded scholars for centuries. However, such performance clearly precludes interactive real-time search, the number one requested feature from our domain experts. Furthermore, in many cases similarity search is not an end goal in itself, but a frequently-called subroutine in a higher level data mining algorithm. Thus, we must drastically improve the performance of the brute force search.

Attempts have been made to speed up the individual calculations of the Histogram Intersection value with various approximations [87], but the time complexity is still linear to the size of the dataset. However, even if the Histogram Intersection value could be *exactly* calculated one hundred times faster, it would still not allow for a real-time interactive search. What is need is a way to eliminate the vast majority of the calculations altogether. We can quickly see how this might be achieved with a simple example. Figure 4.6.right recalls the toy example shown in Figure 4.2. Consider the two queries shown in Figure 4.6.left; could they have perfect matches in Page-1? It is easy to see that we can answer this question without having to resort to the brute force search,
just by examining their *global* histograms. Note that Query-1 has two red, six green, zero blue and one yellow pixels \{2,6,0,1\}. So in order for it to have an exact match, Page-1 must also have at least this number of pixels in its histogram, and it does, it has \{8,12,1,7\}.

Could Query-2 have a perfect match? The answer is clearly no. With a histogram \{0,4,5,0\} it needs at least 5 blue pixels to have a perfect match, but Page-1 can only provide it with 1.

![Figure 4.6: left) Two example queries with their color histograms. right) A sample page (cf. Figure 4.2) and its (global) histogram.](image)

At the risk of redundancy we will further illustrate this idea with real world examples. In Figure 4.7.*left*, a query color patch featuring a shield with a golden deer on a blue background is shown, and in Figure 4.7.*right*, two candidate pages are shown. Let us first consider Page 1. Assume that we have already encountered a moderately similar object before searching this page, then the *best-so-far* value probably will not decrease after checking all color patches in Page 1. We can make this claim because a global view of the *entire* page reveals that it contains few blue pixels, the dominant color of the query. In other words, even if we perform a histogram intersection between Hist(Query) and Hist(Page 1), the blue bin(s) with a high value in Hist(Query) will have no match. If the
entire page cannot supply enough pixels to match the query, then clearly no color patch inside Page 1 can supply enough pixels to match the query. This idea allows us to prune Page 1 with just a single calculation. As for Page 2, which contains a very similar color patch (only the direction of the deer is different) to the query in its bottom left corner, we cannot prune it after checking the color statistics for the whole page. However, if we had recursively divided the page into two halves (an idea we will shortly develop), the “pruning” idea could still be applied to the top half page, since it does not contain enough blue and golden pixels, and we could focus our search on the bottom half page, where a good match exists.

Now we are ready to formally define the lower bound to the color distance in equation (2). If a color patch $CP(r,c,h,w)$ is contained in another color patch $CP'(r',c',h',w')$; in other words, $r \leq r' < r' + h' - 1$, $c \leq c' < c' + w' - 1$, $h \leq h' + r' - r$, $w \leq w' + c' - c$, then:
\[ \text{Dist}(Q, CP') \leq \text{Dist}(Q, CP) \]  

**Proof:**

Since \( CP \) is contained in \( CP' \), any pixel in \( CP \) also belongs to \( CP' \), thus we have:

\[ \forall i \quad (\text{Hist}(CP')[i] \geq \text{Hist}(CP)[i]) \quad \Rightarrow \]

\[ \forall i \quad (\min(\text{Hist}(Q)[i], \text{Hist}(CP') [i])) \]

\[ \geq \min(\text{Hist}(Q)[i], \text{Hist}(CP)[i]) \]  

With (1) and (4), we have:

\[ \text{HI}(\text{Hist}(Q), \text{Hist}(CP')) \geq \text{HI}(\text{Hist}(Q), \text{Hist}(CP)) \]  

With (2) and (5), we can obtain (3). \( \Box \)

### 4.4. Mining and Search Algorithms

In the last section we briefly hinted at the utility of the lower bound distance in searching through two sample pages; here we will explore more formally how we can incorporate the lower bound into similarity search and data mining.

As noted in Section 4.1, we need to scale to massive datasets, so a disk-based indexing method is required. We can index each page by its color histogram (\( \text{PageHist} \)), which is a very compact representation of the original color image (When setting the number of bins \( K = 6^3 \), for a 24-bit color page of size 2000×1500, the space reduction is about 40000:1). We have shown in the last section that by just one comparison with the \( \text{PageHist} \), an entire page can be pruned (i.e. Page 1 in Figure 4.7). In addition, as we shall
show later, the PageHist can provide us with a very good searching/mining ordering by a simple heuristic strategy.

While the PageHist information can sometimes help us prune off entire pages, it does not contain any information to guide our search within a page (i.e. Page 2 in Figure 4.7), so we also need to obtain color histograms for lower level patches. To prevent the need for repeated transformations from the color space to histogram, we record a BinMatrix for each page (cf. Figure 4.2.right), which provides the bin_id of each pixel, as a preprocessing step. The color histogram of a specific patch in a page can then be easily obtained by a simple summation of its corresponding area in the BinMatrix. BinMatrix is stored on the disk, and retrieved when a search inside its corresponding page is initiated. As we will show in Section 4.5, such retrieval is only necessary for a tiny portion of all pages and so the resulting I/O overhead is marginal.

4.4.1. Nearest Neighbor Search

In the example illustrated in Figure 4.7, we first compare the query to the whole page, and if the matching value is worse than the best-so-far, we can immediately prune the entire page; otherwise, we must continue by comparing the query to lower level patches of this page. Note that this idea lends itself to a classic divide-and-conquer framework.

As with all divide-and-conquer algorithms, we must be careful that data on the edge of the “division line” does not get missed. As Figure 4.8 hints at, we achieve this by having an appropriate amount of overlap between divided sections.
We first show the algorithm (in Table 4.2) which updates the current nearest neighbor in a color patch, given the height, width and histogram of the query, and BinMatrix of the patch. This algorithm will become a subroutine in our algorithm for searching an entire book (see Table 4.3 later in this section).

Table 4.2: Divide-and-conquer algorithm to update the current nearest neighbor of a given query in a color patch.

<table>
<thead>
<tr>
<th>Algorithm UpdateNN_DC(h, w, Hist_Q, BinMatrix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
</tbody>
</table>

In the above divide-and-conquer algorithm, lines 6-15 correspond to the “divide” phase: a patch is divided into $\text{Sec} \times \text{Sec}$ sub-patches, which are assigned $1/\text{Sec}$ of possible offsets in the row-wise and column-wise respectively (line 6 and 7). Lines 2-5 and line 16-18 correspond to the “conquer” phase: if the current patch is small enough, i.e. the area ratio of the query to the patch is above a certain value (line 2), then we simply
perform a brute force search in it (line 3); otherwise, we update the current $NN$ in its sub-patches (line 17), to which the (lower bound) distance is smaller than best-so-far (line 16).

In the example shown in Figure 4.8, a page is divided into $2 \times 2$ sub-patches.

![Figure 4.8: left) A query of size $h \times w$. right) A page of size $H \times W$, which is divided into $2 \times 2$ sub-patches (only two on the left are plotted). Note there is an overlap between them to include all offsets.](image)

The only two parameters that need to be set are $StopRatio$ and $Sec$. We can set $StopRatio$ to any value $\leq 1$. If $StopRatio \leq (h \times w)/(H \times W)$ (the area ratio of the query to the page), then no division would be performed and the algorithm degrades to the brute force algorithm; If $StopRatio = 1$, then we would have to divide the page until the size of the query, which is another undesirable “extreme point”. In this case, when the patch gets very close to the query size, very few sub-patches can be pruned, and it is more efficient

\footnote{NN\_BruteForce in Table 4.2 (search the query in a patch) is only slightly different from the one in Table 4.1 (search in a book), and is thus omitted for brevity.}
to do a brute force search. Although we know the *best StopRatio* has a value between 
\((h \times w)/(H \times W)\) and 1, it is difficult to estimate the optimal value in advance. This is 
because the optimal value depends on both the data (the book) and the user’s query.

However, finding a good (but not necessarily optimal) value for the *StopRatio* is 
easy. Notice that the choice of “valid” *StopRatio* is quite limited, because given the 
height and width of the page and query, and the value for *Sec*, the maximal depth of 
division is a fixed small number. In addition, as we will show in the next section, once 
the depth is above a certain number, the running time does not change much (it only 
increases slightly when approaching the maximal depth), and this holds for queries in 
various sizes and content on different datasets. Given that the value is not critical, we 
learn a good value by experimenting on a small set of queries, and simply hard code that 
value thereafter.

For the second parameter *Sec*, its value can be any integer in the range 
\([2, \min(H-h+1, W-w+1)]\). A larger *Sec* reduces the size of sub-patches, and thus obtains a tighter 
lower bound distance. However, as noted in Figure 4.8, there are overlaps between sub- 
patches, and the area of overlaps increases with the value of *Sec*:

\[
\text{Area(Overlap)} = \text{sum(Area(SubPatch))} - \text{Area(Patch)} \\
= \text{SubH} \times \text{SubW} \times \text{Sec}^2 - H \times W \\
= ((H - h + 1) + (h - 1) \times \text{Sec})((W - w + 1) + (w - 1) \times \text{Sec}) \\
- H \times W \\
\]

(6)

This tells us that the cost of tighter lower bounds is to be forced to check more 
area and more sub-patches. If *Sec* = \(\min(H-h+1, W-w+1)\), we can obtain the tightest lower 
bound, but we also generate the most sub-patches, and the algorithm would be similar to
a brute force one (in which the size of the sub-patch equals that of the query, and so the
lower bound distance is the exact distance). Another drawback of setting a larger Sec is
that the algorithm has to maintain many sub-patches before we can prune them.

Our empirical testing suggests something that has been hinted at in other domains
[28]. Once a lower bound is reasonably tight, making it tighter has little extra value,
especially when searching large datasets. For example, the top left sub-patch in Figure
4.8 can be pruned even if Sec is set to just 2. In Section 4.5 we will show in almost all
cases, setting Sec to 2 can achieve excellent performance, and that the exact value of Sec
is not critical to the performance.

We are now in a position to give the complete algorithm to search for the nearest
neighbor of a given color patch in a book in Table 4.3:

Table 4.3: Fast algorithm using the lower bound distance to search the nearest neighbor NN of a
query Q in a book B.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[NN] = NN_LB(Q, B, PageHist, BinMatrix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>best-so-far = INF;</td>
</tr>
<tr>
<td>2</td>
<td>h = height(Q); w = width(Q); Hist_Q = Hist(Q);</td>
</tr>
<tr>
<td>3</td>
<td>foreach page B{i} in B</td>
</tr>
<tr>
<td>4</td>
<td>LB(i) = Dist(Hist_Q, PageHist(B{i}));</td>
</tr>
<tr>
<td>5</td>
<td>endforeach</td>
</tr>
<tr>
<td>6</td>
<td>Sort pages by I such that LB(I(i)) ≤ LB(I(i+1));</td>
</tr>
<tr>
<td>7</td>
<td>foreach page B{I(i)} in B</td>
</tr>
<tr>
<td>8</td>
<td>if LB(I(i)) ≥ best-so-far</td>
</tr>
<tr>
<td>9</td>
<td>break;</td>
</tr>
<tr>
<td>10</td>
<td>endif</td>
</tr>
<tr>
<td>11</td>
<td>UpdateNN_DC(h, w, Hist_Q, BinMatrix{I(i)});</td>
</tr>
<tr>
<td>12</td>
<td>endforeach</td>
</tr>
</tbody>
</table>

Lines 3-5 first calculate the lower bound distance from the query to each page,
then lines 7-12 search pages in the increasing order of their lower bound distances. Our
experimental results in Section 4.5 will show that by using this simple heuristic strategy,
the eventual nearest neighbor is always found in the first few pages and the algorithm can be stopped at any time after a very short startup time. Once we find a page whose lower bound distance is not smaller than best-so-far (line 8), the algorithm can be terminated (since we can guarantee that there is no better match remaining).

It is obvious that finding a smaller best-so-far early on helps to prune more aggressively. In addition to sorting all pages (line 6), we apply two more optimizations in each page search (omitted in Table 4.2 and Table 4.3 for brevity):

- We can quickly find a good “first best-so-far”. We downsample the query and the first page, find the NN by the brute force algorithm in the downsampled space, and then project back the location of NN to the original page to obtain the “first best-so-far” value.

- We can sort sub-patches based on their distances to the query. Then we check most promising sub-patches first, and can terminate the search in a patch when the lower bound distance to the next sub-patch is not less than the best-so-far.

4.4.2. Motif Discovery

As we shall see in our experimental section, the query-by-content algorithm introduced in the previous section has already proven useful to several domain experts. However, true data mining comes with the discovery of previously unknown patterns. In this section we show that we can further extend our ideas to allow for the discovery of patterns which repeat within or between books. As this idea closely models the idea of DNA motifs and time series motifs [61], we call such patterns color patch motifs.
**Definition 7:** The *Color Patch Motif* is a pair of color patches \( \{CP, CP'\} \) of the given size \( h \times w \) that is most similar among all possible pairs (excluding those residing in the same page) in a book \( B \).

This definition generates an incredibly large search space. If we consider the example in Section 4.3, then the brute force motif discovery algorithm has to compare each color patch in a page to all color patches in other pages. The number of comparisons required is about \( 7.75 \times 10^{17} \). Clearly we need to design an efficient algorithm. Before we design the algorithm for efficient motif discovery, it is helpful to first answer the following questions:

- *Do we really need to compare all these pairs?*

If we learn the experience we gained from the nearest neighbor search, the lower bound distance defined in formula (3) between a query and color patches can be trivially extended to our motif discovery scenario. For two patches \( CP1 \) and \( CP2 \), which are contained in larger patches \( CP1' \) and \( CP2' \) respectively, we have:

\[
\text{Dist} (CP1', CP2') \leq \text{Dist} (CP1, CP2) \quad (7)
\]

This means that if the distance between two patches is larger than *best-so-far*, no better motif can be found between these two patches. Therefore, a similar divide-and-conquer algorithm as the one shown in Table 4.2 can be applied.

- *Is one motif enough?*

The definition above will only return the most similar pair of color patches, a single pattern. If we generalized it to instead find top \( N \) motifs, many trivial matches with a slight shift (say, 1 pixel away) from the best motif would be returned. If \( N \) is not large
enough, then all top motifs may correspond to a single pattern. We can generate a more
diverse top $N$ motif set by answering the next question.

- Do we need to return the exact location of motifs?

One factor contributing to the diversity problem is that we have to return the exact
location of the motifs with the exact user-specified size. If instead we returned two
slightly larger patches, which contain a motif pair of the size specified by users, the end
user would surely not care. As we shall see, this slight relaxation of the problem makes it
much easier to produce an efficient algorithm.

Given this relaxed definition, trivial matches can be combined into these slightly
larger patches (we call them leaf patches below). Astute readers may see that leaf patch
pairs may also be trivial, but should note that the quantity of trivial matches decreases
significantly (e.g.: even for a leaf patch only 9 pixels larger in height and width, it
contains 100 patches of the motif size; and two such leaf patches can contain up to
10,000 trivial matches). Furthermore, as we will show in the next section (Figure 4.17,
etc), by simply plotting all leaf patches, or just showing one leaf patch pair, the patterns
are still quite obvious to the human eye.

Having made these observations, we are now in a position to formalize the
algorithm to update the current best motif between two patches, given the (user-specified)
height and width of the motif, $BinMatrix$ of two patches and the distance between them.

Table 4.4 makes this concrete:
Table 4.4: Algorithm to update the current best motif of size $h \times w$ between two patches ($BinMatrix1$ and $BinMatrix2$) with distance $d$.

```
Algorithm UpdateMotif(h, w, BinMatrix1, BinMatrix2, d)

1. $H = \text{height}(BinMatrix1)$; $W = \text{width}(BinMatrix1)$;
2. if $(h \times w)/(H \times W) \geq \text{StopRatio}$
3.   $Hist1 = BinMatrix1(1:h,1:w)$;
4.   $Hist2 = BinMatrix2(1:h,1:w)$;
5.   if $\text{Dist}(Hist1, Hist2) < \text{best-so-far}$
6.     $\text{best-so-far} = \text{Dist}(Hist1, Hist2)$;
7.   endif
8.   Save $BinMatrix1$, $BinMatrix2$, $d$;
9. return;
10. endif
11. Divide both patches into $\text{Sec} \times \text{Sec}$ sub-patches;
12. foreach sub-patch $p$ in 1st patch
13.   foreach sub-patch $q$ in 2nd patch
14.     if $\text{Dist}(p, q) \leq \text{best-so-far}$
15.       UpdateMotif$(h, w, \text{BinMatrix}_p, \text{BinMatrix}_q, \text{Dist}(p, q))$;
16.     endif
17.   endforeach
18. endforeach
```

When a pair of patches is small enough (line 2)\(^7\), we do not issue a brute force search as we did for the nearest neighbor search in Table 4.2; instead, we only calculate the distance for one pair of sub-patches (which can be the one in the top left corner, as shown in lines 3-4) contained in the leaf patch pair, and use this value to update $\text{best-so-far}$ (lines 5-7). Each leaf patch pair along with their distance are also recorded for further reference (line 8), since we have not checked all but one sub-patch pair of the motif size.

If two patches are not small enough, we divide them into sub-patches as in Table 4.2, and calculate all pair-wise distances. A further check between sub-patches is required.

\(^7\) Here we assume two patches are in a similar size for simplicity. The algorithm can be easily extended to handle the case where their size differs a lot [105].
only when the lower bound distance is not larger than *best-so-far* (lines 14-16), based on formula (7).

By doing this, we save time by eliminating brute force searches between leaf patches, while still guaranteeing that there is no false dismissal: the top one motif *must* exist in one recorded leaf patch pair. The proof is straightforward:

**Proof:**

We denote the top one motif pair as $M_1$ and $M_2$, and two leaf patches containing them as $LP_1$ and $LP_2$.

With formula (7), we have:

$$\text{Dist}(LP_1, LP_2) \leq \text{Dist}(M_1, M_2) \quad (8)$$

With the definition of the motif, we have:

$$\text{Dist}(M_1, M_2) \leq \text{best-so-far} \quad (9)$$

With (8) and (9), we have:

$$\text{Dist}(LP_1, LP_2) \leq \text{best-so-far}$$

Which means, $LP_1$ and $LP_2$ can always pass the test in line 14 of Table 4.4, and thus will be recorded. □

One thing worth mentioning in the above algorithm is that the evaluation of $StopRatio$ is quite different from the one in Table 4.2. Here we do not only care about the *speed* but more about the *quality* of retrieved results. $StopRatio$ has to be close to 1, or the lower bound distance of leaf patch pairs would be too loose and we would not find a similar color patch pair of the user specified size. Given that the “valid” $StopRatio$ is quite limited and its value is discrete, it is not hard to pick one. However, we should also
keep in mind that the judgment for a good $StopRatio$ is subjective and thus we make it adjustable. For example, the user can increase $StopRatio$ if he/she wants to find more similar patterns.

Finally, the framework of the complete algorithm to find motifs from a book is very similar to the one in Table 4.3. We first calculate lower bound distances between all pairs of pages, and sort page pairs in the increasing order of their distances, and then for each pair call $UpdateMotif$ in Table 4.4. The search can be terminated if the distance between two pages is larger than $best-so-far$. The last step is to scan all saved leaf patch pairs, removing those with larger distances than the eventual $best-so-far$.

## 4.5. Empirical Evaluation

We have designed all experiments such that they are easily reproducible. To this end, we have built a webpage [105] which contains all datasets and code used in this chapter, together with spreadsheets which contain the raw numbers displayed in all the figures.

We divide our empirical evaluation into two sections: The first section contains informal case studies to demonstrate the utility of our system, and give the reader an intuition as to the kinds of scenarios in which our ideas can be used. In the second section we evaluate our ideas using the classic data mining metrics of sensitivity to parameters, speed, etc.
4.5.1. Case Studies

4.5.1.1. Query by Content

A historian at UC-Riverside wishes to find out how long humans have known that fish in the genus *Scorpaena* are poisonous. Fortunately, illustrated books on fish have been popular since the 16th century and hundreds of such books are now online. We indexed one such text from 1834, since its title “A Selection of the Most Remarkable and Interesting of the Fishes found on the Coast of Ceylon” refers to the region the historian is interested in. A Google image search found an illustration of one member of the genus, *Scorpaena scrofa*, and, as shown in Figure 4.9, we used this image as a query to our book.

The best match to our query shows a member of the genus, *Scorpaena mile*, and the accompanying text does not mention that the fish is poisonous.

Figure 4.9: left) A query image of an illustration of *Scorpaena scrofa* and its best match in the text [12]. right) For context, six random other fish from the same text.
In this case we were fortunate enough to find an illustration of a member of *Scorpaena*. Suppose we were not so fortunate, could we obtain a good match using a real image of a fish?

To test this possibility, we queried the database with the first three images retrieved in a Google image search for “*Scorpaena*”. Figure 4.10 shows the results.

![Figure 4.10](image_url)

Figure 4.10: Three images of real fish from the genus *Scorpaena*. Each of them was used as a query to the text [12], and each of them found the same.

In the above four scenarios, the average time taken to find the *exact* nearest neighbor was 16.5 seconds (ranging from 5.6 to 29.0 seconds); however, the time to report the page on which the best match occurs took less than 0.1 second, allowing a truly interactive search. Note that in every case our query window size was smaller than the correct matching fish, and in the wrong orientation; however, it is clear in these experiments these are not critical sensitivities.

While we envision our work primarily for finding patterns within and between historical texts (cf. Section 4.1), the results shown in Figure 4.10 emboldened us to
consider examples where there is great utility in querying historical texts with modern images.

Historians and genealogists often need to search manuscripts to trace lineages. In addition to using text, they often use coats of arms as clues. Coats of arms are extremely common in Western manuscripts beginning in the fourteenth century (similar emblems exist in other cultures, for example, the Japanese *Mon*). In some cases a single text may have over 1,000 examples of different coats of arms.

In this case, finding query images is easy, since there is a huge *wiki-like* community of amateur historians that have created clean idealized versions of coats of arms; Figure 4.11. *left* shows an example. Some individual enthusiasts have collected or created more than 50,000 examples [45].

![Figure 4.11: *left* A modern idealized version of the coat of arms of Fugger von Babenhausen is used as a query. *center* The best match to query in the 1545 text, *Das Ehrenbuch der Fugger*. *right* A zoom-in of the best match.]

As we can see in Figure 4.11. *left* the modern images are idealized, “clean” and created with a small color palette. Could such images be correctly matched to real hand-colored instances from 500-year-old texts? To test this we queried a text *Das Ehrenbuch der Fugger* (*The secret book of honor of the Fugger*, referred as *Fuggers* book for short)
below) [23]. This manuscript impressively illustrates the genealogical self-esteem of the Fugger family, a famous dynasty of wealthy merchants. The text contains portraits of 138 members of the family, including the matriarch, together with their coats of arms. Figure 4.11.center shows the best match to the query, and Figure 4.11.right shows an enlarged detail.

While the nearest neighbor is similar, with some editing we can make it more similar. Figure 4.12 illustrates this.

Figure 4.12: left) We can edit the coat of arms discovered in our search by removing the center panel and stretching the two remaining sections to meet in the middle. right) The resulting image looks more like the query.

We did not do the additional step of reversing the direction of the three horns in the bottom left quarter, since the handedness of objects is (typically) irrelevant in heraldry. This result suggests that the coat of arms we found was the original Fugger von Babenhausen coat of arms, augmented (the technical term is Quartered) by the insertion of a new panel. This query suggests that our distance measure is quite robust to “distortions” in the image, especially to the inevitable differences in the color palettes used by modern historians and the color palettes available to medieval artists.

Before moving on, it would be remiss of us not to comment on an invariance that our measure achieved. Both the original and query arms show a woman holding a mitre
(“Pope’s hat”). In the query image the woman is clearly Caucasian and blond. However, in the discovered coat of arms the woman appears to African. This is not the result of discoloration of the original color, or an error by the artist. From external sources we discovered that the black figure is the women mentioned in the Biblical book Song of Songs 1:5 “I am black but comely, O daughters of Jerusalem, ...”. The explanation as to why the query image features a white woman is the obvious and disquieting one. According to Ralf Hartemink, a noted expert on heraldry, “In the 19th and early 20th century many black people were made white, as racism became more common...” [46].

In the above example, the best match is an augmented (quartered) version of the query. Joining (Quartering) different coats of arms is a common method to generate a new coat of arms, usually because of the marriage. Such examples widely exist, for example, the query itself (Figure 4.11.left) consists of three different sub-patches, and the composition of the best match (Figure 4.11.right) is even more complicated. We believe that to exploit possible relations between coats of arms based on quartering is may be useful in tracing lineages. By simply adjusting search parameters, our algorithm is capable of looking for corresponding quartered version of coats of arms. We illustrate the idea by examples in a 1546 text Das Sächsische Stammbuch (The Saxon pedigree book) [25], which contains sketches of about two hundred members of the nobility from the German state of Saxony and theirs coat of arms. Figure 4.13.left and Figure 4.13.right show two pages from this book.
In Figure 4.13.*, we see a coat of arms featuring a black horse on a red background in the bottom right corner (a zoom-in view can be found in Figure 4.13.*). Was this coat of arms ever quartered to form other coat of arms? To answer this question, we first tested one simplest template, where the new coat of arms consists of two coats of arms in the top half and bottom half respectively. Now the definition of *quartered nearest neighbor* of the query becomes a color patch (1) whose height is half of the query and whose width is the same as the query (2) whose color histogram is closest to $0.5 \times \text{Hist}(\text{query})$. Figure 4.13.* shows the best match we found in *The Saxon pedigree book* (a zoom-in view in Figure 4.13.*). Coincidentally, we find that the bottom half of the quartered coat of arms is also from the same page (Figure 4.13.*) as the “black horse”.

Figure 4.13: *left*) A query coat of arms in a page from *The Saxon pedigree book* [25]. *center*) enlarger query and best match. *right*) the best quartered match to the query found in the same text.
We are lucky enough to find a good match in the first try. What if there is no corresponding template existing for the query? We tested two more templates for the leftmost coat of arms in Figure 4.13. The results are shown in Figure 4.14.

Figure 4.14: The coat of arms from Figure 4.13 queried in two templates. The top row shows results of a left-right template and the bottom row shows result of a quadrant template.

We first tested the left-right template, the top row of Figure 4.14 shows the top 2 and top 4 matches to the query (recall that our $NN$ algorithm would report best-so-far matches before finding the final best match). Although neither of them satisfied the testing template, in both cases we found a good match of quartered coat of arms (in other templates). We also tested a quadrant template, and the bottom row of Figure 4.14 shows the top 2 and top 3 matches, one of which satisfied the template while the other one not. Our $NN$ algorithm is fast enough for the user to test multiple templates till he found the desired one.

If the interesting patch appears in a quartered version of coat of arms, a user is likely to ask “what is its origin?”. This reverse process can also be simply achieved by
setting a corresponding searching window and its color histogram. In Figure 4.15, we show three such examples using different templates.

![Figure 4.15: Examples of reverse search to find origins for patches in quartered coats of arms.](image)

To further test the robustness and utility of our distance measurement, we envision the following scenario: A tourist spots a coat of arms somewhere in Europe, and wants to know its origin and whom it belongs to. But there is no instruction or a self-guided tour manual available at hand. He then takes a photo on his iPhone (as seen in Figure 4.16.left) and submits the query...

The invariances we should support for such queries are more challenging, due to the different lighting conditions, contrast settings, angle of view and size, etc when shooting the photo. Suppose we have indexed a text *Leopards of England, and other papers on heraldry* [31] which introduces the history of royal arms with color illustrations. Figure 4.16.center shows the best match to the query (the real photo of a physical shield hung on a wall) in it. Note the considerable differences between them: (1)
the query is vertical while the best match is horizontal; (2) the query is about twice larger than the best match; (3) the golden leopards are more vivid than the best match but without the black edges. Four randomly chosen coats of arms with leopards are also shown in Figure 4.16.right for context.

Figure 4.16: left) Photo of the famous “three leopards” (or lions) as the query. center) The best match found in [31]. right) Four additional coats of arms from the same text.

From the title of the best match we immediately know that this is the coat of arms of England from 1198 to 1340. Reading in more details, we further know that it was first used by Richard I and it is his second Great Seal (since the first one “went astray after his release from captivity in Germany”). Also, Richard I is the first king who bore the three golden leopards on a red field, which has since become a symbol of England. We believe such a tool would be fun to use for people who are ever curious about things they see.

4.5.1.2. Motif Discovery

We first tested our motif discovery algorithm on the 526 page Fuggers book which we investigated in the last section. Due to space limitations, we show two typical examples in Figure 4.17, and archive the complete results in [105].
Figure 4.17: *left*) A pair of pages which contains 27 leaf patch pairs, with 6 leaf patches on left page and 7 on the right page displayed. *right*) another pair contains 170 leaf patch pairs, with 16 and 18 leaf patches on each page.

In the above results, we plotted out all leaf patches (the highly overlapped rectangles), without specifying which two are in a pair. However, as we can see, this does not affect the utility of the patterns. So instead of allowing the algorithm to test all 170 combinations (in Figure 4.17. *right*), we can have the algorithm save time by reporting just one pair, which may not be optimal, but will surely satisfy a user.

Our work on datasets made us realize a limitation of motif discovery. The domain expert in this field wanted to ask questions such as “Are there two or more examples of a shield that is *not* a Fuggers shield, indicating that two or more members of the same family had married into the Fuggers dynasty?”. This translates to “Are there motifs that do not feature pale blue and gold?”. To answer such queries we need to do supervised motif discovery. These essentially reduce to the constraint of *must-include* or *must-exclude* these colors. Such constraints are trivial to include in our algorithm. For example, an entomologist has one specimen of “tiger butterfly” (shown in Figure 4.18. *left*) and wants to find motifs containing the similar orange color. He can simply select an orange patch and set a threshold, say 10%, which is the ratio of pixels from orange bin(s) in the
final motifs. To guarantee this, our algorithm checks whether two patches/pages contain enough orange pixels before searching in them. In Figure 4.18. right we show one such motif pair found in an old butterfly book [49] with about 300 pages. An obvious advantage of supervised motif discovery is that it can prune much more candidates and hence gain huge speed up: the motif discovery was done in less than 5 minutes for the example shown in Figure 4.18.

Figure 4.18: left) A photo of an orange/black butterfly used as a must-include color constraint on motif discovery. right) A pair of tiger butterflies found by the supervised motif discovery algorithm in [49].

4.5.1.3. Link Analyses

We can trivially extend our work on finding motifs within a book, to finding motifs between two books. This allows us to add hyperlinks between the two texts that researchers can follow as they investigate a topic [16]. This is a more challenging problem, since even two logically identical items from different texts may vary much more in the size, color palette used, style, etc. than those from the same manuscript.

We performed link analyses on two old books about butterflies. One of them contains illustrations of about 1250 species [39]. The other is a volume of “new”
butterflies (referred to as “Exotic Butterflies” below) [49]. As the latter text is more heavily annotated, users who have questions about the butterfly in the former book can find helpful references in the latter one if there is a link created.

It took about one hour to “join” these two books, and five pairs of pages were returned. We do not claim our algorithm could find the most similar patch pair in the exact specified size, but again, what we can guarantee is that it must exist in one pair of returned pages. Given there are only five such pairs for this case, it makes no difference to an exact algorithm, only much faster. We show two representative links in Figure 4.19.

Figure 4.19: Two links found between two butterfly books. Note that the technique seems robust to the text variability (color palette, size, etc.). According to an entomologist, the left match is correct at the genus level and the right match is correct at the species level.

These results are very assuring and alleviate our worries about the variability that exists between different books. To further obtain more links, we can decrease the StopRatio as discussed in the last section. Another simple but effective method is to keep joining results between each page of one book and another entire book. Some additional examples can be found in Figure 4.20.
4.5.2. Data Mining Metrics

4.5.2.1. Sensitivity of Parameters

Previously we discussed parameter settings theoretically. Here we intend to show, empirically, that all parameters used by our algorithms are relatively insensitive and that it is easy to learn settings which generate effective and efficient results.

We first investigated the relation between \( \text{Sec} \) and the execution time of our \( \text{NN} \) algorithm. Since for a given query and a fixed \( \text{Sec} \), the execution time varies when using different \( \text{StopRatio} \), we then tested all possibilities (recall that the number of “valid” \( \text{StopRatio} \) is limited) to find the best running time for different \( \text{Sec} \). Figure 4.22.\textit{left} shows the results of 5 randomly chosen queries (distorted and only taking 64% of their original area, see one example in Figure 4.21) in the \textit{Fuggers} book.
Figure 4.21: left) A color patch cut from the *Fuggers* book and a query made by randomly removing a row slice and a column slice both taking 1/5 area. right) The nearest neighbor found in the book. Note that it is irrelevant to the changing of parameters.

We can clearly see that: (1) The best execution time of large *Sec* (20, 30 and 40) is much worse than small *Sec*; (2) The best execution time of small *Sec* does not vary that much, but still slightly increases with *Sec*. 3 out of 5 queries ran the fastest when *Sec* = 2, while only 0.38 and 0.05 second slower than the best (where *Sec* = 3) in the other two cases.

Next we checked the effect of *StopRatio* (i.e. the depth to stop the division) on the execution time when *Sec* = 2. The result for the same 5 queries is shown in Figure 4.22.

Figure 4.22: left) “Best execution time to find NN” vs. “*Sec*” for 5 randomly chosen queries from the *Fuggers* book. Note only 20, 30, 40 are tested for *Sec* >10. right) “Execution time to find NN” vs. “*StopDepth*” for the same 5 queries when *Sec* = 2. Note *StopDepth* = 1 corresponds to the brute-force search (but still with the heuristic page order).
This plot suggests that as long as we do not set the \textit{StopDepth} too small, the execution time can be decreased to a similarly low level by our algorithm. Hence, we set the \textit{Sec} to 2 and \textit{StopDepth} to 8 for all experiments\textsuperscript{8}.

While parameters only affect the running time and our \textit{NN} algorithm can always return the same \textit{exact} answer, both of them (the speed and quality of answers) are related to parameters for the motif discovery algorithm.

First we set \textit{Sec} also to 2, for the same reason as the \textit{NN} algorithm. Moreover, a larger \textit{Sec} is more unpreferred because it has to maintain more sub-patches now.

And we set the algorithm to stop division at the depth whose \textit{StopRatio} is closest to 0.9. With this amount of relaxation, all case studies (e.g.: Figure 4.17 to Figure 4.20) visually show we can still find very similar color patch pairs. In addition, we also tested to stop at a smaller and larger depth, with \textit{StopRatio} ranging from 0.805 to 0.954. Reasonable motif pairs were discovered in all these experiments, and we found a larger \textit{StopRatio} producing fewer but more similar color patch motifs.

The only really important parameter for motif discovery is the motif size. It can be based on the domain knowledge and users’ interests. What we did is to test three different sizes for all experiments, and they all returned similar motifs of corresponding sizes. However, for complicated datasets (e.g.: butterflies), it is difficult to find such a “perfect” value. In spite of this, however, it is clear that our algorithm can find some

\textsuperscript{8} We did the same experiments (as in Figure 4.22) on the “Exotic Butterflies” book, and found the similar results, which we report at [105].
motifs of different sizes (e.g.: Figure 4.19, right), and is efficient enough to search of a range of sizes.

4.5.2.2. Speed and Anytime Property

We randomly picked 10 new queries from the Fuggers book, and reran the NN search with learned parameters. The average running time was 14.1 seconds (ranging from 1.0 to 39.0 sec). However, for 4 out of 10 queries, the best match appeared on the first page returned, and it took less than 0.1 second. Even in the worst case, the best match was on the 6th we retrieved. In comparison, a search using the brute force algorithm took 14 hours. The dramatic speed up is due to the high pruning rate: more than 99.99% of distance calculations were pruned in every query. We archive the NN search result for “Exotic Butterflies” in [105]. Its performance is even better: the best match was located on the first page for 8 out of 10 queries.

While motif discovery may need to run overnight, the brute force algorithm would take years. Moreover, with the heuristic search order, we can always stop at any time after a very short startup time while still obtaining good results (see Figure 4.23), essentially making it an anytime algorithm.

![Figure 4.23: “Goodness of BSF” vs. “Time” for motif discovery in the Fuggers book. The “goodness” is measured as “eventual BSF/current BSF”.

109
4.6. A Novel Framework for Image Retrieval

Case studies in Section 4.5.1.1 have shown that our query-by-content algorithm is robust to color palette, size, orientation, etc. However, in those examples a pre-indexed text from the same domain is always available for the query. What if we do not have such a book/dataset containing similar samples as the query patch? It is almost certain that we cannot obtain satisfying retrieval results, and this is also a common problem for other content-based image retrieval systems [34][44][63][67][81][91][92] which have tried searching by color, texture, shape, layout, etc. As we will show later in this section, even for TinEye [92], the largest reverse image search engine in the market which has indexed more than 1.97 billion images, will fail in a search if the query or any similar sample does not exist in the database. Another drawback of the content-based image retrieval is that low-level features used in CBIR systems are often hard to describe high-level semantic concepts [102], which makes current CBIR systems cannot fulfill to users expectations.

On the other hand, the text-based image retrieval using high-level features (such as keyword, text descriptor) is more close to human’s mind in the semantic perspective. And as the ever expanding internet, one disadvantage of text-based image retrieval that it requires considerable human labor to annotate images has been overcome: web image search engines like Google Images [41] and Bing Images [13] can automatically extract

\* Data provided by the website on July 14th, 2011.
textual information for web images, such as the image file name, the ALT-tag, the URL, and the surrounding text, etc [18]. These image databases are so comprehensive that they can always return some relevant images no matter which (textual) query the user inputs. The main problem is that the retrieval precision is poor, due to the inaccuracy of web annotations. The user has to go through the long list of images to find those desire ones, which is time-consuming, and may easily miss the good match if it is not ranked high enough, say, after the first 10 pages.

The above analysis and comparison of content-based and text-based image retrieval frameworks gives us an intuition that combining these two frameworks may increase both recall and precision. A few such attempts using both visual and textual features have been made, for example in [18], they cluster the text-based image search results based on visual, textual and link analysis information to facilitate user’s browsing. The experiments based on 11.6 million images crawled from the web show their system can cluster results to several semantic categories.

The framework we propose also utilizes text-based image retrieval as the first step, since it provides good recall. Instead of building the system on a subset of images as in [18], we will directly plug the web image search engine into this step. Then we will calculate the color histogram distance defined in formula (2) between the query and each retrieval image to re-rank them, considering our measurement is robust and effective to find similar images if there are good matches in the dataset. Finally, we return the re-ranked images accompanied by web pages where they come from to the user.
The following example further demonstrates how our *text-first* and *color-refined* framework works. Figure 4.24.*left* shows a page of coats of arms from an old text [70]. A reader is interested in one of them with three red circles on a yellow background and wants to find more information about it. He then uses its surrounding text “Louis VI King of France” to query in the Google image search engine. Figure 4.24.*right* shows the 1st page of all 543,000 results. As we can see, most of them are portraits and no coats of arms can be found.

![Figure 4.24: left) A page from *A guide to the study of heraldry* 1840 [70], containing a coat of arms called *three torteaux* with surrounding text “Louis VI King of France”. right) 1st page results of query “Louis VI King of France” in Google image search engine.](image)

While we scroll down and check the remaining images, we do find two very similar images as the query. Can we automatically promote them to the top of the list? We can easily achieve this by comparing query patch (Figure 4.25.*left*) to returning images and re-rank them in the ascending order of color histogram distance (as shown in Figure 4.25.*right*).
Figure 4.25: (left) Query patch from the page in Figure 4.24. (left, right) Re-ranked images based on the distance to the query (only top images are shown), in the order from left to right, top to bottom.

Note that here we consider each candidate image as an atomic item and it only requires one distance calculation for each image. Therefore, the re-ranking is very fast even we perform it on the first hundreds of images.

One additional example is shown in the following two figures. The only difference is that in Figure 4.26, we add “medal” to the original text “spanish american war” as the query, with the hope to narrow down the topics of returning images. But still, the search returns 168,000 results and there is only one image (first one of the second row) hitting the query in the 1st page.

Figure 4.26: (left) One page from [97], containing one medal with text “spanish american war”. (right) 1st page results of query “spanish american war medal” in Google image search engine.
As for the re-ranked results in Figure 4.27, 6 good matches with different variances to the query image are promoted to the first page. Observing that the top one re-ranked image in these two examples is both a hit, we can even add a Google-like “I’m Feeling Lucky” button for the image search engine too.

As a comparison, we also searched the above two color patches (Figure 4.25 and Figure 4.27) in the TinEye image search engine, and it returned 0 result for both of them, hinting the limitation of the content-based image retrieval.

To conclude this section, our proposed image retrieval framework asks for a query keyword(s) and a query image from the user. It first sends the query keyword(s) the web image search engine to ensure that the relevant images are contained in the retrieval image set, and then re-ranks the results based on their similarity of color histogram to the query image. Note that any CBIR technique which is fast enough can be used in the refinement step.
Chapter 5

Conclusions

In this dissertation we considered, for the first time, the problem of mining large collections of rock art. We introduced an explicit framing of the GHT algorithm as a similarity measure, and showed that by lower bounding the measure we can effectively mine large data archives. For the more general problem of creating distance measures for rock art, we also noted some limitations of our current work. Our method does not allow rotation invariance or support partial shape matching. Both of these issues are seen in real rock art databases, and are thus the focus of future research efforts.

To extract the rock art data for the algorithm proposed in Chapter 2 which assumed the input images are bitmaps with a 1-bit color depth, we introduced the
CAPTCHA-ROCK system, which demonstrated *for the first time* a method to crowdsource the annotation and feature extraction of rock art images. We noted several limitations of the CAPTCHA-ROCK system. We could become victims of our own success, given that there are only tens of millions of rock images in existence, but there is a need for tens of millions of CAPTCHAs per day. So even if we capture only a small fraction of the CAPTCHA market, we may run out of rock art images. We believe that we may be able to bypass this issue by generating synthetic rock art images, in a spirit similar to the stickmen shown in Section 3.3.1. In addition, a significant fraction of petroglyph images may not be amenable to our system. Our CAPTCHA is not usable for blind users, and our system may be difficult to use on small screens such as iPhones. However, even if our framework does not succeed as a CAPTCHA, the basic ideas work as a framework to allow volunteers to help annotate data. We tentatively estimate that if every graduate student in anthropology in the US were to donate just one hour a month to the project, all of the world’s rock art could be processed in just a few years.

In addition to the rock art, we also exploited another category of cultural heritage, historical manuscripts. While there is an active research community pursuing data mining of text from historical manuscripts, there has been very little work that exploits the rich color information which is often present. In this dissertation, we introduced a novel lower bound for color matching, and two data mining algorithms that exploit it. We have shown our ideas allow true *data mining*, rather than just *query-by-content* for the domain of historical manuscripts. Furthermore, we also proposed a general image retrieval framework, which combined the advantages of the web image search and our color
measurement. We have made all code and data freely available to the community in order to bootstrap additional research in this area.
Bibliography


[97] Wyllie, Robert E. 1921. Orders, decorations and insignia, military and civil; with the history and romance of their origin and a full description of each. New York Putnam's.


[105] Zhu, Q. 2011. Local color patch Webpage:
http://www.cs.ucr.edu/~qzhu/localcolorpatch

