

Building an Intelligent Knowledgebase of Brachiopod Paleontology

By

Yuanliang Meng

Submitted to the graduate degree program in Computer Science and the Graduate Faculty of the
University of Kansas in partial fulfillment of the requirements for the degree of
Master of Science

Dr. Jun Huan, Chairperson

Committee members

Dr. Brian Potetz, Chairperson

Dr. Bo Luo

Date defended: _____

The Thesis Committee for Yuanliang Meng certifies
that this is the approved version of the following dissertation :

Building an Intelligent Knowledgebase of Brachiopod Paleontology

Dr. Jun Huan, Chairperson

Date approved: _____

Acknowledgements

I would like to thank Dr. Brian Potetz for his advice. Many ideas in this thesis are based on his previous research. Even before I became a student in the computer science program, he had introduced artificial intelligence studies to me, including computer vision, computational neuroscience and probabilistic inference. His knowledge and inspiration impelled me to pursue a degree in this area.

I would like to thank Dr. Luke Huan for his help in many ways. I learned a lot in his classes and discussions, and have had a much deeper understanding of machine learning since then. Because of him, I found myself an active student in academic activities of the department, which reinforced my studying.

I would like to thank Dr. Bo Luo for his advice on the IPKB project. A large portion of my thesis is based on the work under his supervision and guidance. When I was in the early phase of the program, he also showed me how to approach research and write articles.

I also thank Dr. Xue-wen Chen. Though not being in the committee now, he was my first advisor and encouraged me to be a graduate student in this department. Without him, I would not have been able to conduct this research.

Last but not the least, I would like to express my gratitude to all other professors, the staff and students. My studying at KU was fruitful and pleasant because of them.

Abstract

Science advances not only because of new discoveries, but also due to revolutionary ideas drawn from accumulated data. The quality of studies in paleontology, in particular, depends on accessibility of fossil data. This research builds an intelligent system based on brachiopod fossil images and their descriptions published in *Treatise on Invertebrate Paleontology*. The project is still on going and some significant developments will be discussed here.

This thesis has two major parts. The first part describes the digitization, organization and integration of information extracted from the Treatise. The Treatise is in PDF format and it is non-trivial to convert large volumes into a structured, easily accessible digital library. Three important topics will be discussed: (1) how to extract data entries from the text, and save them in a structured manner; (2) how to crop individual specimen images from figures automatically, and associate each image with text entries; (3) how to build a search engine to perform both keyword search and natural language search. The search engine already has a web interface and many useful tasks can be done with ease.

Verbal descriptions are second-hand information of fossil images and thus have limitations. The second part of the thesis develops an algorithm to compare fossil images directly, without referring to textual information. After similarities between fossil images are calculated, we can use the results in image search, fossil classification, and so on. The algorithm is based on deformable templates, and utilizes expectation propagation to find the optimal deformation. Specifically, I superimpose a “warp” on each image. Each node of the warp encapsulates a vector of local texture features,

and comparing two images involves two steps: (1) deform the warp to the optimal configuration, so the energy function is minimized; and (2) based on the optimal configuration, compute the distance of two images. Experiment results confirmed that the method is reasonable and robust.

Contents

1	Introduction of Invertebrate Paleontology Knowledgebase	1
1.1	Introduction	1
2	Overview of IPKB	3
2.0.1	Text Processing	5
2.1	Image processing	9
2.1.1	Image segmentation	11
2.1.2	Fossil contour detection	12
2.1.3	Fossil label association	13
2.1.4	Label image recognition	14
2.2	Web access	15
2.2.1	Searching and browsing	15
2.2.1.1	Quick search	16
2.2.1.2	Advanced search	17
2.2.1.3	Browsing	18
2.2.2	Genus record displaying	18
2.3	Related work	20
2.4	Conclusion and future work	22
3	Fossil image matching	23
3.1	Review of image recognition	24

3.1.1	Eigenfaces	24
3.1.2	Deformable template matching	26
3.2	Algorithm for matching fossil images	29
3.2.1	Image model and the objective function	31
3.2.2	Expectation propagation	37
3.2.3	Image preprocessing	39
3.2.4	Construct the warps	41
3.2.5	Texture features	42
3.2.6	Implementation of expectation propagation in deformable templates	45
3.3	Summary	47
4	Experiments and results of image recognition	48
4.1	Demonstration of deformation results	48
4.2	Effect of number of nodes	51
4.3	Experiment and evaluation	52
4.3.1	Experiment 1: matching images	53
4.3.2	Experiment 2: matching images with deformation partially depleted	55
4.3.3	Experiment 3: Calculating score without deformation cost	57
4.3.4	Compare results of experiment 1, 2 and 3	58
4.3.5	Experiment 4: fossil image classification	60
4.4	Summary and conclusions	62
5	Conclusion and Discussions	63
A	The 25 key images used in experiments	67

List of Figures

2.1	Overview of the IPKB system.	4
2.2	The description of individual genera in the treatise.	6
2.3	Image processing in IPKB: (a). the original figure extracted from from the Treatise; Images with the same numeric index belong to the same genus. (b). Opening-by-construction: fossils are masked. (c). Removing fossils from the figure, leaving only labels and captions. (d). Identified label blocks. (e). Using watershed method to split fossils. (f). Detected fossil contours.	9
2.4	A figure from the Treatise. The fossils are embedded in rocks.	10
2.5	The distance matrix. Each column represents a photo and each row represents a label. The value 100 is a dummy distance for those too far away.	13
2.6	The homepage of the web interface.	15
2.7	The webpage for advanced search.	16
2.8	The result of quick search, using "admi" as keyword.	18
2.9	The result of advanced search.	19
2.10	View information of a genus. The photos are on the left side. Google Earth displays the geographical distribution.	20
3.1	Deformation costs of grids. With pairwise connection model, only distances between neighboring nodes are considered. Both cases have the same total cost. With full connection model, all possible pairs of nodes are considered, and the costs differ.	33

3.2	A graphical illustration of expectation propagation. Image curtesy of Dr. Brian Potetz	40
3.3	Cropping images. Complement images are used here to show the margins. Margins are white (as opposed to black here) in real cases.	40
3.4	A visualization of a warp. Each circle represents a node.	42
3.5	Illustration of gray level co-occurrence matrix. In this example, only left-right pairs are counted.	44
3.6	Four directions to count co-occurrences. The arrays are used to represent directions and offsets in the MATLAB function.	44
4.1	Results of deformation (without discretization). All images are from the family Orthida. (a). Dorsal view of genus <i>Nanorthis</i> , used as template image; (b). the same as the template; (c). another specimen of <i>Nanorthis</i> ; (d). dorsal view of <i>Cyrthonotella</i> ; (e) posterior view of <i>Cyrthonotella</i> ; (f). dorsal view of <i>Nothorthis</i> ; (g) a <i>shoshonorthis</i> fossil embedded in rock.	49
4.2	Processing time and number of nodes. Calculation is based on the images shown in Figure 4.1	51
4.3	These two images have only one valid match each (which is themselves).	55
4.4	Matching with no deformation. The image on the left is the template. Nodes are linearly mapped to target images. The grids on the right edge and bottom edge can be slightly different in size from others.	56
4.5	Test images borrowed from Dry Dregers. All images are from the suborder Orthidina. (a). Eridorthis; (b). Glyptorthis; (c). Hebertella; (d). Plaesiomys; (e) Platystrophia; (f). Plectorthis; (g) Retrorsirostra.	60

List of Tables

4.1	Matching results of images in Figure 4.1. Scaling parameter = 2×10^{-3}	50
4.2	Matching results of 25 key images. V.M. means valid matches. # <i>V.M by Chance</i> is defined as the number of images from the figure divided by the total number of images in the 25 figures, and then multiplied by the number of returned items (10).	54
4.3	Matching results of 25 key images with deformation partially depleted. V.M. means valid matches.	57
4.4	Ignore deformation cost in the last step.	59
4.5	Measure of success. It is the sum of reciprocals of the second best V.M. rankings. .	59
4.6	Classification of the fossils used as test images.	61
4.7	Classification results of the images, using weighted 5-nearest neighbor method. Only one case fails.	61

Chapter 1

Introduction of Invertebrate Paleontology

Knowledgebase

1.1 Introduction

Science advances not only from revolutionary discoveries, but also through innovative ideas drawn from accumulated data. Paleontology is not exceptional. The robustness of studies in this field depends on quality and accessibility of data. According to the definition in wikipedia, “Paleontology (also known as palaeontology) is the scientific study of prehistoric life. It includes the study of fossils to determine organisms’ evolution and interactions with each other and their environments”. As a ‘historical scienc’ it attempts to explain causes rather than conduct experiments to observe effects.”

Paleontologists agree that the most authoritative compilation of data on invertebrate fossils is in the *Treatise on Invertebrate Paleontology*¹. The *Treatise* was founded in 1948 and the first volume appeared in 1953. Since then, The Paleontological Institute of the University of Kansas has published 50 volumes, authored by more than 300 contributors worldwide. This encyclopedic work now occupies more than 1.3 meters of shelf space. For paleontologists (and geologists,

¹<http://paleo.ku.edu/treatise/>

biostratigraphers, etc.) the world over, the Treatise holds an almost biblical significance and is to be found in every good library. Systematic paleontologists make bibliographic reference to the Treatise in nearly all of their publications.

Given the rich information in the Treatise, understanding and modeling paleontological data are significant in many areas, such as: understanding evolution; understanding climate change; finding fossil fuels, etc. The vast repository of paleontological data contained in the Treatise needs to be made available in electronic form for present and future workers to extract the greatest possible use from the work. In order to realize the maximum possible benefit from this landmark effort, there is a strong desire within the paleontological community to be able to readily access and use this data. However, it is a non-trivial task to convert large volumes of the Treatise into a structured, and easily accessible digital library. In particular, we face the following challenges: (1) the Treatise contains heterogeneous data objects that are not easy to be associated under a unified framework, in particular, linking text entries with fossil images; (2) although the editorial policies and procedures of the Treatise have always been of the highest standard, it consists of manuscripts from hundreds of paleontologists, which introduces significant inconsistency of styles, formats, and sometimes terminologies.

So far we have achieved a first step towards digitization, organization and integration of the rich information extracted from the Treatise. In this thesis, I will introduce two research projects I participated in, and discuss the problems and solutions. In the first project, we built a system called *Invertebrate Paleontology Knowledgebase (IPKB)*. The ultimate goal is to provide scientists with a general framework that facilitates knowledge discovery activities in invertebrate paleontology. So far the system supports keyword search. In later chapters of this thesis, I will discuss the design of interface as well as search functions. In the second project, we designed an algorithm to compare and classify fossil images. Ultimately this algorithm can be incorporated in the system and make it much more flexible.

Chapter 2

Overview of IPKB

The Treatise on Invertebrate Paleontology is a pandect of all invertebrate fossil genera, together with their taxonomic synonyms, stratigraphic ranges, geographic distributions, and illustrations of the type specimens. In the Treatise, tremendous amounts of data have been accumulated from different resources, and organized in old-fashioned ways. The Treatise is published in separate parts, *Part A* through *W*, and each part contains multiple volumes. Earlier volumes are scanned from printed books, while the newest volumes have PDF files from the publisher. We started from *Part H. Brachiopoda*, which has 6 volumes of paleontological data of brachiopods, a phylum of marine invertebrates with two valves (or “shells”). Fossils are classified in a hierarchical structure (from highest level to lowest): *orders* (e.g. *Lingulida*, *Orthida*); *suborders* (e.g. *Dalmanellidina*, *Orthidina*); *superfamilies* (e.g. *Plectorthoidea*, *Wellerelloidea*); *families* (e.g. *Allorhynchidae*, *Pontisiidae*); and finally *genera* (over 4,000 genera in Part H).

A good system has to possess two characters. One is good useability. It should be able to “get the job done” and do it in a fast, timing manner. The other is aesthetic consideration. The interface should please the eyes and attract people to play with it instead of just tolerating it. Even though we are not delivering a commercial product, we keep the principles in mind.

Although the recent volumes (e.g. all six volumes in Part H) are digitized [Moore & other editors, 2006], they are no more than electronic reprints of paper publications. As we know, the only way to search

in a PDF file is using exact text matching. In the current system we built, however, users are able to submit structured queries. As illustrated before, from a PDF file it is impossible to locate records of certain morphological features and/or geological distributions, across a variety of families. Our advanced search function, however, can find relevant records instantly and rank them according to relevancy. When reading a book, we often wish we could jump from one topic to a related one, as well as connect to some external sources. Our system provides a flexible interface for doing so, too. In addition, traditional publications do not sufficiently take users' experience into account. Modern systems like ours emphasize more on information layout and data presentation methods. Hopefully, information access turns out to be more enjoyable as well. Our long-term goal is to build a system which does not only present data, but also interpret them.

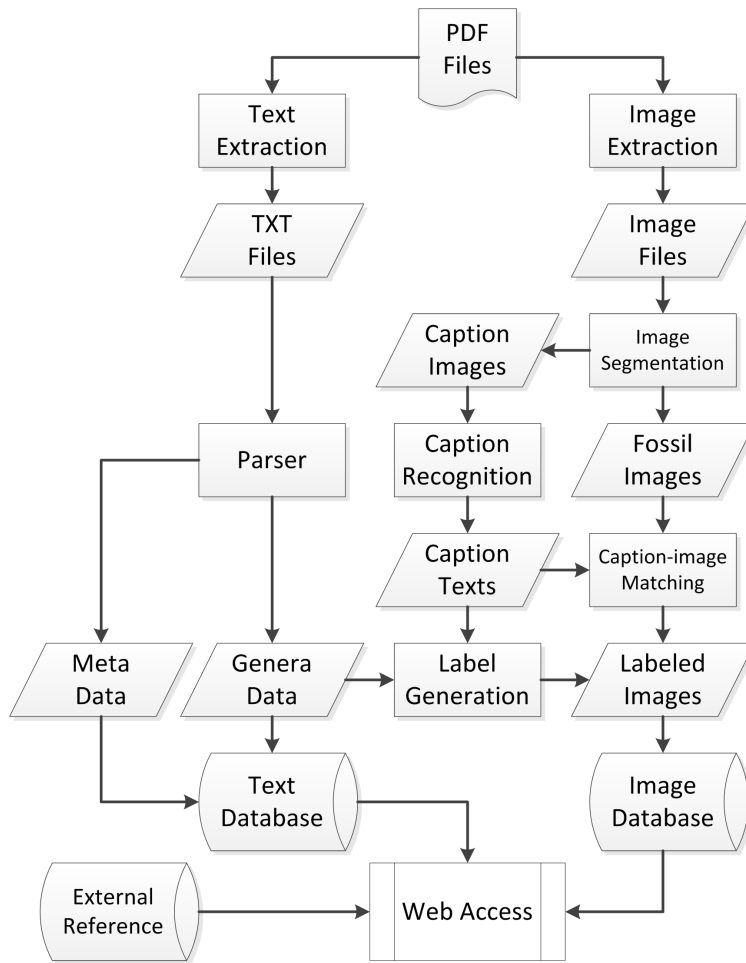


Figure 2.1: Overview of the IPKB system.

Figure 2.1 demonstrates the overall structure of the IPKB system. As shown in the figure, we first extract text and image data from PDF files, and process them separately. Text data is parsed to obtain structured genus records and other meta data (e.g. phylum introduction, glossary, etc). Images are segmented and image captions are recognized and associated with the corresponding segment. Text descriptions of the genus records are then linked with the images. Finally, processed text and image data is provided to the web access module, which includes browsing, searching and genus display as three main components. In the rest of the paper, we use *Part H. Brachiopoda* as examples to describe each module in details. However, the methods are applied to all volumes.

IPKB first extracts raw data from the Treatise, and processes textual and image data objects separately. We have designed an ontology to capture fossil information, and explored ways to link fossil (genera) descriptions with images. Finally, a web interface is designed to present the information, and provide easy browsing and searching functions. With the help of our system, users can achieve some goals that are impossible with a PDF file alone. The following are examples of such scenarios:

Use case 1. A paleontologist wants to browse all genera of superfamily *Plectorthoidea* between geological times *Eifelian* and *Moscovian*.

Use case 2. In Japan, a geologist finds a brachiopod fossil which is subpentagonal and recti-marginate; and the foramen is permesothyrid. He wants to check the geological period it could correspond to.

In the following sections, readers will get to know how the system is designed and implemented.

2.0.1 Text Processing

Part H of the treatise consists 6 volumes with 3,226 numbered pages in total. All volumes are available in PDF format, which was directly generated from the typesetting software. There are 1960 indexed figures. In most cases, a figure contains a number of fossil images, and each image

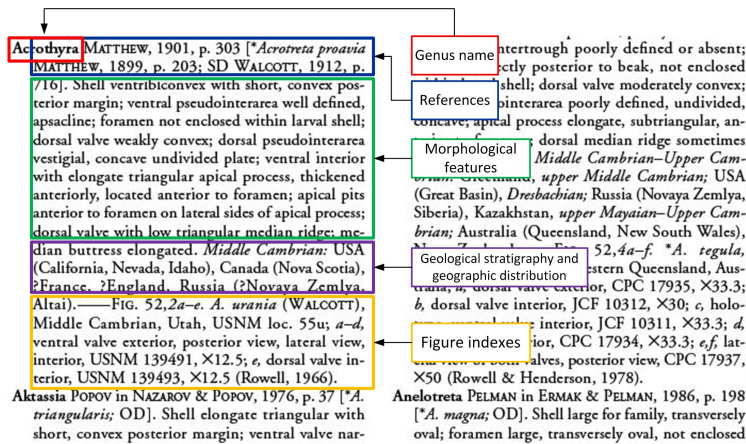


Figure 2.2: The description of individual genera in the treatise.

has a label. In IPKB, text and images are extracted and processed separately. We will discuss text processing in this section, and image processing in next section.

In brachiopod paleontology literature, the basic taxon is usually a *genus*. In the Treatise, each genus record consists of a paragraph of text description, and often a few corresponding images. According to the editorial requirements, genus description should provide the following pieces of information in a fixed order:

1. Genus name; e.g. *Acrothyra*.
2. References; linking to external publication(s), i.e. the original sources describing the genus.
3. Morphological features; e.g. “shell ventribiconvex with short, convex posterior margin; ventral ...” The editorial guidelines also requires the authors to describe features in a stated order, and use limited vocabulary. However, such guidelines are not strictly followed by all the contributors.
4. Geological stratigraphy and geographic distribution; e.g. “Middle Cambrian: USA (California, Nevada, Idaho), ...” Please note that fossils are usually found at multiple locations, and the locations are identified in highly inconsistent manner.
5. Figure indexes; e.g. “FIG. 52, 2a–e.” Multiple images may be included for a genus.

Figure 2.2 shows an example of a genus record. In each paragraph of a genus, missing information is not uncommon and when something is absent, there is no explicit indicator. Currently, we are not exploiting the references, since all the related works are not digitally available, which prevents us from providing any hyperlink. All other pieces of information are crucial and are incorporated in the IPKB system. In addition, every genus is governed by some higher taxa as mentioned before. Names of the corresponding higher taxa are not given in the paragraphs, but they are found in the titles of corresponding chapters and their subordinate sections. These names are important information and are captured too.

We have implemented a Python program to extract the information and save the parsed text in an XML file. The algorithm is based on regular expression matching, which uses keywords and syntax of the text description to split the text paragraph into structured data. Here the syntax does not refer to sentence structures, but merely indicates format patterns, including capital letters and punctuations and so on. Please note, other than pure textual data, we do not have any metadata (e.g. font, size, style, etc.), since we were unable to robustly extract typesetting data from the PDF (possibly due to the typesetting software used to produce the PDF file).

Genus names consist of a number of letters, with the first one capitalized. What follows is an author's name with all capital letters. This is a unique context and can be used in the extraction algorithm directly. The part of geological and geographical distributions starts with some reserved term for geological time, and ends with a long hyphen in the text. Therefore it can be obtained once a list of terms are provided. The beginning of the geological part is also the ending of the morphological description, so clear indicators of boundaries can be used in both ways, forward and backward. Such features also help us to identify and recover incorrectly formatted genus descriptions.

In order to get the names of higher taxa, we need to capture an order name first, and then assume all the genera belong to that particular order until a new order is captured. The same principle applies to suborders, super families, families, and subfamilies. Sometimes one or more taxon names are not available, and we just marked it as "UNKNOWN" in our data set. Occasionally a taxon is

“UNCERTAIN” because it is controversial in literature, and such cases are clearly indicated in the Treatise we work on.

We have designed an ontology to capture the genera data and meta data extracted in text processing. Genera data includes entries of genus records as described above, while meta data includes other descriptive information included in the volumes. For instance, volume information, the glossary, an introduction to the phylum, and so on are considered as meta data. Structured textual data are then stored in XML format.

2.1 Image processing

Image extraction is a time-consuming procedure of data preparation. In order to reduce manual work, we tried to implement automatic methods. Currently we have processed the photos of fossils, corresponding to the genus entries. Figure 2.3 (a) shows an example of the original figures extracted from the PDF files.

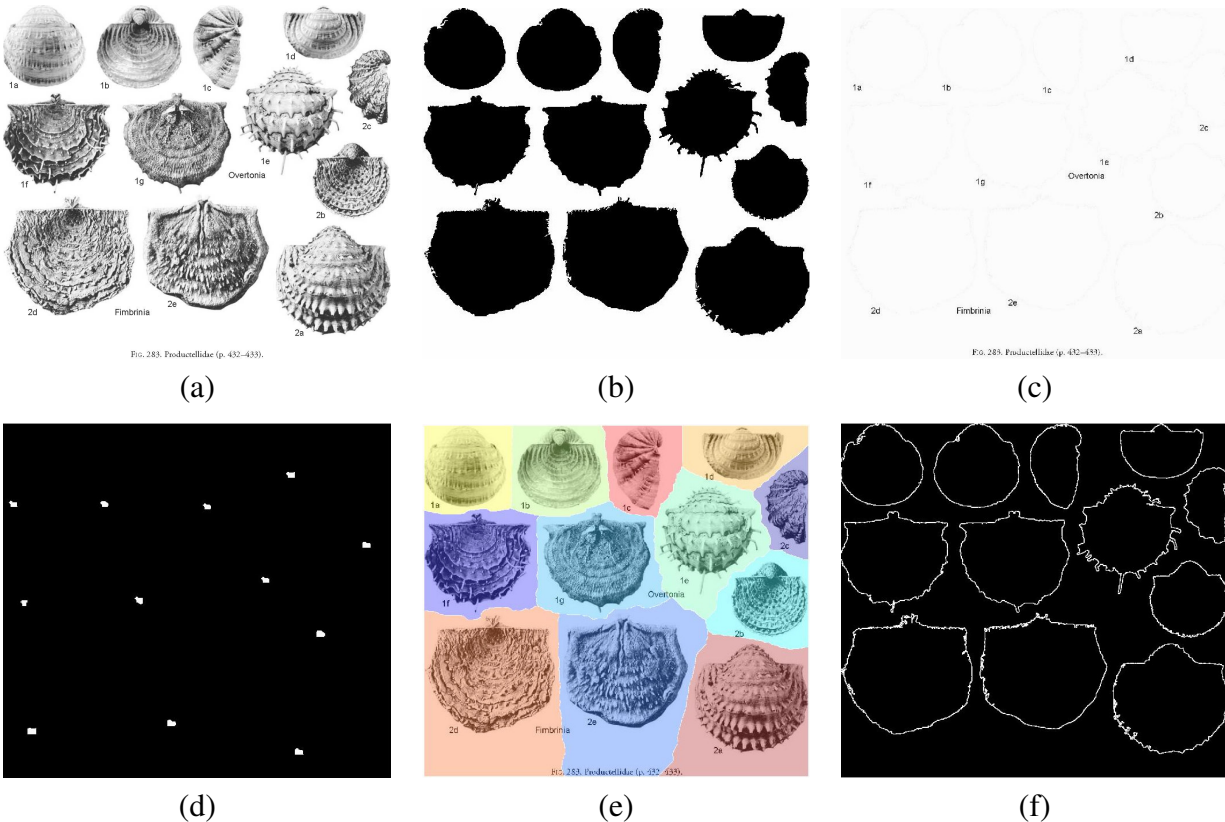


Figure 2.3: Image processing in IPKB: (a). the original figure extracted from from the Treatise; Images with the same numeric index belong to the same genus. (b). Opening-by-construction: fossils are masked. (c). Removing fossils from the figure, leaving only labels and captions. (d). Identified label blocks. (e). Using watershed method to split fossils. (f). Detected fossil contours.

As shown in Figure 2.3 (a), the fossil images in the Treatise are manually selected, grouped and placed in a single image canvas. Each figure in the PDF file is a flattened bitmap image, without any metadata to split individual fossil images, and we do not have access to individual fossil images either. Each figure typically contains images of a few fossils, all in grayscale. The images with the same numeric index belong to the same genus. For instance, Figure 2.3 (a) *1a – e* all belong to

the genus *Overtonia*. Different letters a, b, c and so on usually refer to various views of the same fossil, but they could also be different specimen from the same genus. As a result, the label of an image is typically a number followed by a letter, and occasionally only a number (if the genus has only one fossil photo).

We first need to segment and extract all fossil images from the figures. In order to locate the images correctly, the label under each fossil (e.g. 1a) needs to be identified and recognized too. Moreover, we need to exploit the labels to match the images with the corresponding genus descriptions, hence link genus records with fossil images. Images are first extracted from PDF files using a function provided by Adobe Acrobat. The process is mostly automatic, while manual intervention is often necessary. Segmenting the images and recognizing the labels, however, are much more difficult. We implemented the algorithms with Matlab. The major steps are:

- (1) Segment fossil images and label images;
- (2) Detect contours of fossil images;
- (3) Match fossil images with label images, using their mutual distances in the figure;
- (4) Compare the label images with templates and determine the content;
- (5) Name the fossil images with their label.

In the following we will describe the methods in more details.

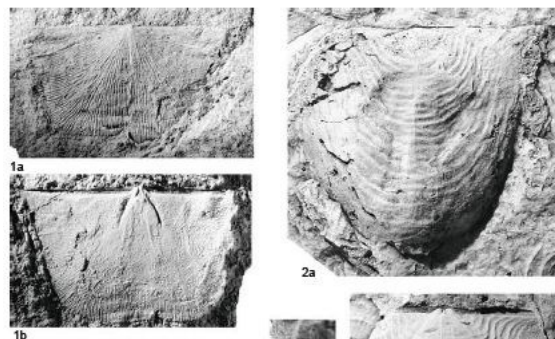


Figure 2.4: A figure from the Treatise. The fossils are embedded in rocks.

2.1.1 Image segmentation

Part H of the Treatise documents Brachiopods, which are marine invertebrates with two shells, hence, the photos exhibit fossilized shells of various shapes. The vast majority of fossil images are presented with a blank background, such as those in Figure 2.3 (a). However, occasionally a fossil can be seen *in situ* i.e. in a rock, as Figure 2.4 demonstrates. In this case, we just treat the fossil with the rock as an entity and do not further segment.

We have implemented an approach that is similar to the Marker-Controlled Watershed Segmentation method [Gonzalez et al., 2009] to separate multiple fossil images in the same figure. First, we use the *opening-by-construction* approach to mark the objects i.e. fossil images as well as the background. This approach consists of an image erosion phase followed by an image reconstruction phase [Vincent, 1993]. In image erosion, we choose a disk-like structuring element object with a diameter of 20 pixels. Anything smaller than 40 pixels in any dimension will be completely eroded. Since the labels and captions are usually about 30 pixels tall, such an operation can retain fossil photos and eliminate labels, markers, embedded captions and other noise. In the image reconstruction, the idea is to use the eroded image as the “seed”, and use the original image as the mask. In our implementation, we reconstructed the background (which includes labels and captions), i.e. the reverse of the fossil photos. Figure 2.3 (b) shows the result of the reconstruction. The background is now bright (i.e. 1) and the fossil photos are dark (i.e. 0).

After the image fossils (i.e. large objects) are identified and the background is marked, we identify the labels (i.e. small objects) similarly. We first mask out the fossils from the original image, as Figure 2.3 (c) shows. Another image opening procedure is implemented to obtain the label blocks. In this case, we used an image erosion followed by an image dilation, and the size of the structuring element object is set to 10 pixels so that embedded text labels are kept. Among the identified text blocks, we retained the smaller ones since we are only interested in labels, not the captions (e.g. “Overtonia” in the figure). The identified labels are shown in Figure 2.3 (d).

Finally, we employed the *watershed* approach to partition the fossil images in the figure, and later they are saved as separate JPG files. The watershed approach is to find local minima in an

greyscale image, and cut them in pieces [Meyer, 1994]. Assuming that pixel intensity values in a digital image represent the altitudes, we can draw an analogy between a grayscale image and a topographic relief. Therefore there are “peaks” with local maxima and “basins” with local minima. The idea of the watershed algorithm is to continuously add “water” into the basins to make them “flood”. When the levels of water rise to the point where two water sources meet, a “barrier” is built. The resulting set of barriers are the watersheds, which separate the regions defined by basins. In our case, the fossils in the figure have been marked with dark pixels (i.e. basins), and the background has been marked with white pixels. The watershed method is used to determine the barrier between individual fossil images (i.e. basins), and hence split them from the figure. Figure 2.3 (e) shows the result of the watershed approach. After segmentation, each fossil image is padded to a square canvas and stored in separate JPG files.

The mathematic details are beyond the scope of this thesis, but interested readers may refer to [Serra, 1982].

2.1.2 Fossil contour detection

In the original figure, fossil labels are manually placed “near” the corresponding fossil image. They are supposed to be placed to the lower left corner of the fossil images, however, this rule is frequently violated. The amount of the images makes it impossible to manually fix such errors. Therefore, we have designed an automatic method to associated fossils and their labels.

Although fossils are segmented and extracted from the figures, we still need an accurate identification of the fossil contours to compute the minimum distance between labels and fossil images. Edge detection has been well studied in the image processing literature, among various methods [Parker, 1997, Gonzalez et al., 2009], we have employed the *sobel* edge detection approach. A sobel operator is a 3×3 kernel, which is applied to the image using 2-D convolution, to compute the (approximate) gradient of pixel intensity. For a 2D image, two kernels are used: one for the

horizontal dimension and the other for the vertical dimension:

$$\mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{I} \text{ and } \mathbf{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{I} \quad (2.1)$$

The magnitude of the gradient can be obtained in the following way: one for the horizontal dimension and the other for the vertical dimension:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \quad (2.2)$$

The edge pixels are those with high values of \mathbf{G} . If the image is masked, as in our algorithm, \mathbf{G} is simply zero for all the non-edge pixels. Figure 2.3 (f) demonstrates the detected contours of the fossil photos.

		Fossils											
		1	2	3	4	5	6	7	8	9	10	11	12
Labels	1	17.493	47.043	100	100	100	100	100	100	100	100	100	100
	2	100	19.235	64.498	100	100	100	100	100	100	100	100	100
	3	100	100	21.19	100	100	100	100	100	100	100	100	100
	4	76.217	45.706	100	18.028	68.622	100	100	100	100	100	100	100
	5	100	100	76.968	100	21.587	74.465	100	100	100	100	100	100
	6	100	100	100	100	100	17.464	100	100	100	100	100	100
	7	100	100	100	88.482	100	100	16.971	98.732	100	100	100	100
	8	100	100	100	100	100	100	100	23.77	100	100	100	100
	9	100	100	100	100	100	100	100	25	100	16.971	100	100
	10	100	100	100	100	100	100	100	100	17.493	100	100	100
	11	100	100	100	100	100	100	100	100	31.385	100	18.385	100
	12	100	100	100	100	100	100	100	69.318	100	100	50.537	16.031

Figure 2.5: The distance matrix. Each column represents a photo and each row represents a label. The value 100 is a dummy distance for those too far away.

2.1.3 Fossil label association

We link labels with fossils based on the distance between them. At high level, each label is associated with the nearest fossil. However, there could be confusion in some cases. For instance, in Figure 2.3, label 1d is placed between two fossils, with similar distance to both of them. In this case, if we can first associate the lower fossil with label 1e, we can confidently link 1d to the

upper fossil. Theoretically, this becomes a *bipartite graph matching* problem, i.e. to find a *perfect matching* with minimum distance.

First, we compute the Euclidean distance between the center pixel of the label block and all pixels of a fossil image contour. The minimum value is saved as the “distance” between the label and the fossil. After all labels loop over all fossils, we obtain a distance matrix. We have extracted the distance matrix for labels and fossils from Figure 2.3, and demonstrate the matrix in Figure 2.5. All distances greater than 100 are set to 100 (and ignored), while distances smaller than 100 are kept as candidates. We scan through the columns to identify the minimum distance between labels and images. We first identify columns with only one candidate (e.g. columns 7, 10, 12 in the figure), and eliminate the corresponding rows from the matrix. For instance, we link fossil 10 with label 9 (column 10) without confusion. Therefore, although label 9 is also close to fossil 8 (column 8), they cannot be associated. We repeat this process until all labels and fossils are linked.

2.1.4 Label image recognition

Since the labels have a limited scope of inventory and are small in size, it is adequate to use templates for recognition. We select some label images, binarize them and save them as templates.

The recognition task is to compute the distances between target images and templates, and the one with the minimum distance is the match. Since the label images are relatively small, we used the most intuitive method: computing the normalized pixel-wise sum of square error (SSE) between images (Equation 2.3). The pairs with the minimum SSE is a match. In reality, the target image and the template image do not often have the same size, so we need to try different positions. Again we used a pixel-by-pixel loop.

$$SSE = \frac{\sum_{(x,y)} (\mathbf{I}_{temp}(x,y) - \mathbf{I}_{targ}(x,y))^2}{\sum_{(x,y)} \mathbf{I}_{temp}(x,y)^2 + \sum_{(x,y)} \mathbf{I}_{targ}(x,y)^2} \quad (2.3)$$

Once we know what the label is, we name the associated fossil image with that label (and the figure index). Then the fossil images are properly named.

2.2 Web access

An important goal of our project is to digitally deliver the Treatise to the research community and the general public. Therefore, we built a web interface to facilitate browsing and advanced searching functions. We have adopted the three-tier architecture, where the database server is powered by MySQL, the Web server is powered by Apache+PHP, and JavaScript is used on the client side. A schema shredding is enforced to convert the XML data generated in Section 3 into relational model to be handled in RDBMS, for its maturity and performance. Figure 2.6 shows the homepage of IPKB website. Since the Treatise is not an open-source product, subscribers need to login to view full volumes.

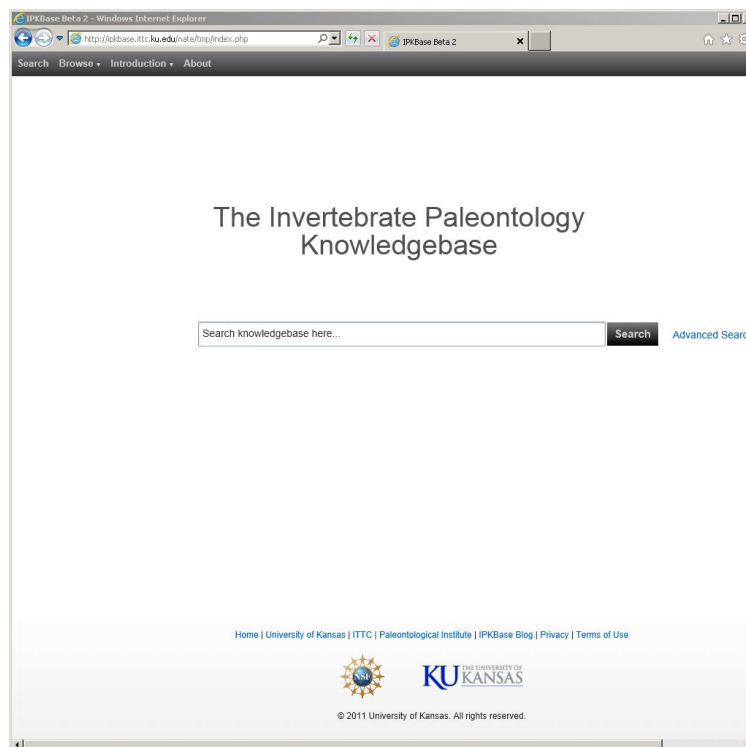


Figure 2.6: The homepage of the web interface.

2.2.1 Searching and browsing

A major function of the web access module is search. At present, we provide two search modes: quick search, which takes free text queries, and advanced search, which takes more complex pred-

icates.

2.2.1.1 Quick search

Quick search accepts a free text query, which could be the name of a genus, or a family, or an order etc., and returns the records that match the name. Nomenclature in paleontology is often confusing even to professionals, hence, we support approximate matching. The algorithm runs in the following way:

- (1) search for exact match of genus name;
- (2) if no results are found, try a natural language full-text search over taxon names;
- (3) if no results are found, try approximate search of genus name;
- (4) if no results are found, ask the user to try another request.

The exact match function only returns entries which contain the exact keyword(s) in the query. The approximate search will allow partial matching. For instance, query “admi” will yield the genus records of “Adminiculoria” and “Adminixtella”, as shown in Figure 2.8. When full-text search is implemented, the records are ranked according to relevance. For more details about relevance, please refer to the discussion on advanced search in next section.

Enter names of species, families, and/or orders:

Enter descriptions (eg. *shell acuminate small*):

Enter places of discovery (eg. *Kansas, USA*):

Enter geological times (eg. *Ordovician: Arenig-Caradoc*)
or use the slider:
From:

To:

Figure 2.7: The webpage for advanced search.

2.2.1.2 Advanced search

Advanced search allows users to send structured queries to IPKB. Users could provide predicates on names, morphological features (descriptions), geological times, and/or geographic locations.

The fossils are dated from lower Cambrian (540 million years ago) to present. In order to help users select a geological time range, we designed a scrollbar with two pointers. In our database, the geological time of fossils are encoded into integers. For a user query, the provided starting and ending times are encoded in the same way, and sent to the database as predicates on a integer field. Moreover, we also send the corresponding geological *period* and *epochs* (if any) of the starting and ending points as keywords in the search query as well. This will yield records that contain geological period and epochs in their morphological descriptions (not in the geological stratigraphy field). This is also useful when people search in a small time range, and helps to provide better ranking.

We have employed the natural language full-text search function in MySQL to support full-text search and similarity-based ranking. In the database, the relevant columns are included in a FULLTEXT index. We have assigned different weights w to different columns to reflect the “importance” of the information in the search context (e.g. genus names are more important than descriptions): 3.0 for taxon names, 1.5 for geological and geographical distribution, and 1.0 for morphological features.

MySQL implements a model similar to TF-IDF in the full-text search function. The *significance* of a term is calculated as:

$$s = \frac{\log(dt f) + 1}{\sum dt f} \times \frac{U}{1 + 0.0115U} \times \log\left(\frac{N - nf}{nf}\right) \quad (2.4)$$

where $dt f$ is the number of times the word appears in a document (i.e. a database record); $\sum dt f$ is the sum of $(\log(dt f) + 1)$'s for all words in the same document. U is the number of unique words in the document (see [Singhal et al., 1996] for more details). N is the total number of documents. nf is the number of documents that contain the term. The combined relevance of a term and a

document is:

$$R = w \times s \times qf \quad (2.5)$$

where qf is the frequency of the term in the query, and in most cases it is 1. It should also be mentioned that MySQL natural language full-text search ignores words shorter than 4 letters by default. In addition, there is a stopwords list which excludes words with too high frequencies.

2.2.1.3 Browsing

The *Browse* function is *de facto* search by names, too. It provides a wizard for users to explore the hierarchical structure of fossil categories. Users see all the orders of brachiopods when the web page is loaded. When an order is clicked on, its subordinate suborders appear, and so on. In this manner, a user does not need to type in anything to locate the genus he wants to view. People can browse all the category names, even if they are not willing to view any specific record.

2.2.2 Genus record displaying

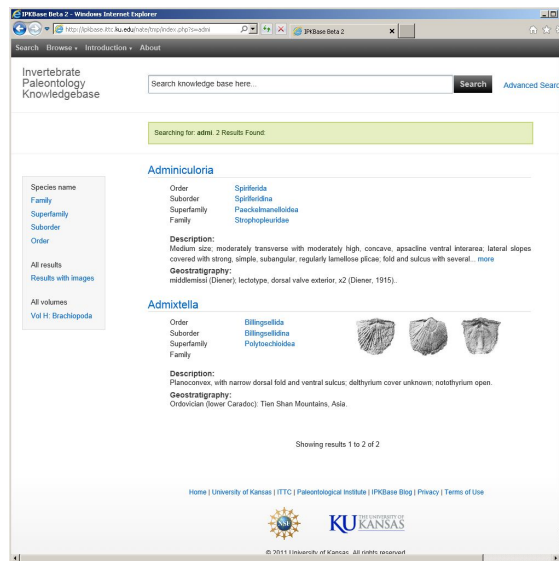


Figure 2.8: The result of quick search, using “admi” as keyword.

In designing the interface to show genus records, we consider the following factors: it must be easy to view and users should be able to perceive the relevant information immediately; there

Nucleata

Order: [TEREBRATULIDA \(view\)](#)
Suborder: [TEREBRATULIDINA \(view\)](#)
Superfamily: [DYSCOLIOIDEA \(view\)](#)
Family: [NUCLEATIDAE \(view\)](#)






Description: 
Small to medium, subrounded to subpen-tagonal in outline; ventribiconvex or planoconvex, anterior...
Geostratigraphy: 
Middle **Jurassic** (Bathonian), **Upper Jurassic (Oxfordian)**-Upper Cretaceous (Cenomanian):
Europe...
Figures: 
Fig. 1423,5a-o. *N. nucleata (Schlotheim), White Jura, Bavaria, Germany; a-b, dorsal and anterior...

Figure 2.9: The result of advanced search.

should be some links that allow users to check related information. The results of quick search and advanced search are essentially delivered in the same format. Up to 10 records (genera) are shown on each page. As shown in Figure 2.8, for each genus record, there are representative photos, if possible; and there are a few lines of textual information. On the top of the text, the name of the genus is shown. Under the genus name, there are names of higher taxa (i.e. family, order etc.) governing the genus. Users can click on a taxon name to view a general description for that taxon. Further down other information can be found, including morphological description, geological and geographic distribution.

In the results of advanced search, the keywords will be highlighted. This feature applies to keywords in geological and geographic distributions, as well as in morphological descriptions. For geological time in particular, the terms marking the beginning and the end of the searched time range will be highlighted. Many of the terms in geological time contain modifiers such as “upper”, “middle” and “lower”. When users are searching for “upper Jurassic”, they certainly do not want to see records with a lot of “upper”s but no “Jurassic”. At the same time, nevertheless, they do want the system to return and highlight records with “Jurassic” even though the modifier is absent. We took cautious treatments for this situation. As we can see from the result, when “Upper Jurassic” is searched for, the strings of “Upper Jurassic” or “Jurassic” are highlighted, but “Upper Cretaceous” and alike are not.

Once a user clicks on the genus name to view the details, a new page containing information of

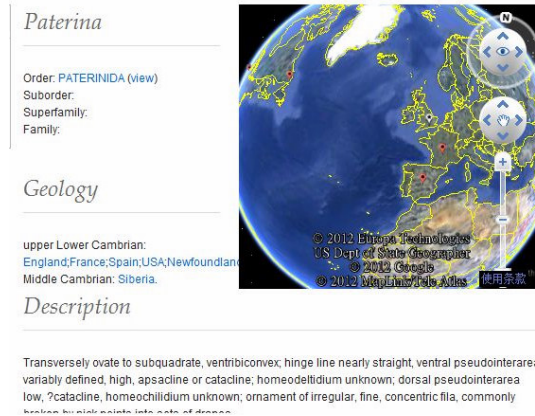


Figure 2.10: View information of a genus. The photos are on the left side. Google Earth displays the geographical distribution.

that specific genus is loaded. All the descriptions and photos of this genus display. Moreover, geographical distribution is crucial information in paleontology. Although it is nontrivial to visualize geological stratigraphy, we have employed Google Earth API ¹ to visualize the sites of discovery of the fossils. The Google Earth API only accepts exact longitudes and latitudes to mark the locations. Therefore we first use the Google Map API to convert the name of locations to coordinates, and then pass it to the Google Earth API. If a certain genus is found in more than one location, multiple markers will show up on Google Earth as well. We also employ the Google Image Search API ² to embed results of Google Image Search in our system. Such plug-ins not only help with data interpretation, but also significantly improves user experiences.

2.3 Related work

In 1990s, digital libraries have been introduced to provide fast and flexible online access to digitalized information repositories (e.g. [Fox et al., 1995, Levy & Marshall, 1995, Choi & Rasmussen, 2006]). Traditional libraries, which house and provide access to collections of books, have made metadata (e.g. catalogs) available and searchable online (e.g. The Library of Congress online catalogs ³).

¹<http://code.google.com/apis/earth/>

²<http://code.google.com/apis/imagesearch/>

³<http://catalog.loc.gov/>

However, access to full content is usually not available over the web since most traditional media are yet to be digitized. Meanwhile, these projects only propose to provide digital access to traditional media, but do not make further explorations on the digitized contents.

We also have digital libraries that focus on a specialized area. For example, bibliographic databases (e.g. PsycINFO, PubMed, arXiv) are repositories for academic publications. On the other hand, Digital Himalaya [Shneiderman et al., 2002] is an anthropology library to digitize, organize and publicize multimedia ethnographic materials from the Himalaya region. The goal is to preserve valuable information in digital format, and provide searchable access to the research community as well as general public.

A number of digital paleontological databases have sprung up over the last decade or so, the best known being the *Paleobiology Database* (PBDB)⁴. This NSF-funded database is the foremost online paleontological database. It differs from the *Treatise* and IPKB in a number of important ways. First, it is populated in a wiki-style process: large numbers of contributors, from a variety of backgrounds, input data with varying degrees of accuracy. In many cases, it is student labor that inputs the data. As a consequence, the quality is highly variable. Second, the data is predominantly text-based, and thus similar to any other queryable dataset; there are few fossil images, and these are random additions by particular workers. In contrast, *Treatise* data is collected, peer-reviewed and verified by the experts in the field, and images of type specimens are an integral feature.

Meanwhile, numerous other paleontological databases can be found on the internet, with each one set up by generally a single worker to reflect their own interests. These vary greatly in scope and utility. For example, the Florissant Fossil Database⁵ is essentially a collection of high-quality images of fossils from this National Park.

⁴PBDB: <http://paleodb.org>

⁵<http://planning.nps.gov/fffo>

2.4 Conclusion and future work

In this chapter, we have discussed our initial accomplishments in IPKB, a project aiming at digitalization, utilizing and sharing of the rich information from the *Treatise on Invertebrate Paleontology*, which is the most authoritative compilation of data on invertebrate fossils. In the project, we first extract raw data from the Treatise, and processes textual and image data objects separately. We have designed an ontology to capture fossil information, and explored ways to segment images, recognize labels, and link fossil (genera) descriptions with images. Finally, a web interface is designed to present the information, and provide easy browsing and searching functions. Google Earth and Google Image Search APIs are employed to help with presentation and improve user experiences.

IPKB provides a solid foundation for future discoveries based on the rich information repository of invertebrate paleontology data. Our next step is to extend the IPKB framework to provide the scientists with a knowledge discovery platform, which hosts complex data analysis, content-based retrieval, modeling, data mining, image processing and understanding, animation and visualization functions.

In the next chapter, I will talk about one of the functions we have just developed: image matching. This application, among others, have not been implemented in our system and is not accessible through the web interface. However I hope users will soon be able to make use of it.

Chapter 3

Fossil image matching

Our data set contains numerous images. In brachiopod paleontology, fossils are primary subjects of studying and thus it is crucial to preserve and present images of the specimen. Researchers would be interested in two functions at least.

One is to use images as “keys” to search in the database. Typing in verbal descriptions as keywords can be both difficult and inefficient. It would be wonderful if one can upload an image directly. After all, “a picture is worth a thousand words”. We can also call this task fossil image recognition.

Another related but different function is to classify new fossils based on their images. Nobody is an expert for everything, and in a relatively specialized field such as brachiopod paleontology, experts are far from abundant, and probably increasingly so. Once we have a fossil image, we may want to know what category it belongs to and the existing data can be utilized to draw some inferences. In this way, our knowledgebase does not only serve as a repository of knowledge, but also becomes an intelligent “thinker” who is capable of performing processes humans can hardly do.

No matter what the purpose is, however, a premise is to define some notion of similarity (or distance) between images. Only if we have a metric to compare images can we discuss recognition or classification. This chapter proposes an algorithm to compute the similarity between images,

which is a foundation of other useful functions. The implementation of this algorithm in a search engine and so on is beyond the scope of this thesis and will only be addressed briefly.

3.1 Review of image recognition

Object recognition has always been a major topic in computer vision research, and it is still a challenging area calling for new techniques. The relatively well studied fields include the recognition of human faces, hand-writings, or vehicles. Some other domain-specific studies are also found in literature, such as recognition of plant leaves.

Facial recognition has drawn a lot of attention because of its importance in security systems in particular. Recognition algorithms can be divided to two families: (1) geometric, which looks at distinctive biometric features, or (2) photometric, which does not specify biometric features *per se* but relies on statistics of images. I will briefly explain two popular approaches here, i.e. the eigenface approach and the deformable template matching approach, as opening words of my algorithm. As a matter of fact, the applications of the methods are not limited to face recognition.

3.1.1 Eigenfaces

The eigenface method developed by [Sirovich & Kirby, 1987] and used by Matthew Turk and Alex Pentland in face classification [Turk & Pentland, 1991] is a significant development. Since it is not directly related to our topic in this article, I will only briefly discuss it here.

Face images are collections of pixels with various intensity values. In other words, all information lies in the brightness and darkness of pixels, as well as their interrelations. The major task, and also the major difficulty of face recognition is to find a computationally economical representation of the “faces”, and thus the similarity between images can be established. It may be hard to find an optimal set of features that represent face images. Statistically, however, the Principal Component Analysis (PCA) can often serve this purpose. The idea is to use a relatively large set of face images to find the eigenvectors of their covariance matrix, and then use those eigenvectors as

bases to represent faces. Those eigenvectors are called eigenfaces when face images are involved, and a relative small number of eigenfaces may be used to span the data space. Hence, *dimension reduction* is achieved. In principle, a standard procedure would be like the following:

- Save images in a matrix $\mathbf{X}_{m \times n}$. Each column $\mathbf{x}_i, i = 1, 2 \dots n$ is an image.
- Compute the mean face over columns $\mathbf{t} = \sum_{i=1}^n \mathbf{x}_i$, and then subtract the mean from each column in \mathbf{X} . The result is a matrix Y in which each row has the mean value of zero.
- Perform singular value decomposition on Y : $\mathbf{Y} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Choose a certain number of column vectors from \mathbf{U} as principal components, or eigenfaces. The first 10 columns of \mathbf{U} will be the first 10 eigenfaces, for example. Correspondingly, the first 10 diagonal elements in the diagonal matrix σ are the 10 largest of all.
- Project the images on the eigenfaces to obtain a compact representation. The number of parameters will then be the number of eigenfaces.

What is given above is just one way to use eigenfaces but other ways are possible. For example, one could compute the covariance matrix, and find the eigenvectors of the covariance matrix. It is intuitively simpler but is more costly from a computation perspective.

No matter which specific techniques are chosen, however, a common idea is to find the principal components of data. Those components are vectors from an orthonormal basis in the data space. They are corresponding to the largest eigenvalues of the data covariance matrix, and thus account for the most variances among data. If we have to represent data images in a compact way, it is naturally a good strategy to maintain as many differences among data as possible.

The eigenface approach is non-parametric and is easy to use. Nevertheless it is solely based on decorrelating second-order dependencies and can be misleading at times. For example, if the data are non-Gaussian and happen to lie on a few non-orthogonal vectors, PCA will fail. In addition, in order to derive the principal components, there has to be a relative big set of training data.

3.1.2 Deformable template matching

The deformable template matching approach (also known as elastic matching, flexible matching or nonlinear template matching) directly matches one image on another. There are many variations of the method, but all of them superimpose a “structure” on the template image, deform the structure in a way to better match the target image, and then compute the similarity (or on the contrary, “distance”) between the images. The deformation procedure is subject to some constraints, and is associated with certain deformation costs. The structure to be deformed can be either *free-form models* or *parametric models*.

The free-form models do not have specific shapes and there are only general constraints with respect to continuity and smoothness, which are often formalized in some *energy functions*. These models are very flexible and are often utilized to capture object boundaries. A very successful model is the Snake [M. Kass & Terzopoulos, 1988], also known as the Active Contour Model. It defines two types of energy functions. The *external energy* function attracts the contours to certain positions in a image, ideally boundaries of objects. However, the *internal energy* requires the contours to maintain a certain level of curvature and length, among other characteristics. The optimal solution is supposed to minimize the combined energy.

The parametric models, however, assume some prior properties of the structures. For example, instead of letting the contours be any possible shapes, we can use a collection of parameterized blocks. Sometimes it does not only reduce computational complexity but also yields better results. A successful approach is *triangulation*, which decompose an object into connected triangles. Since triangles are connected in a tree structure and have relatively simple shapes, the shape of an object can be factorized in a good way and has desirable computation properties.

[Anil K. Jain & Lakshmanan, 1996] proposes an often-cited *General deformation Model*. It is primarily used for shape detection too, but it has close affinity with the model to be discussed in this thesis. Their model has three components:

1. A prototype template which represents the shape of the object class;

2. Transformations that deform the template;
3. A probabilistic model of deformation.

The prototype template is simply an image representing the object of interest. Among a set of images, we want to find a match for the object, and the match is usually not exact. In order to visualize the transformation of the template image, they superimpose a grid-like “planar rubber sheet”, which is often called a warp nowadays, on the template image and allow the warp to be stretched, squeezed and twisted. Originally, the warp consists of a network of small squares, and after the transformation, the edges of the squares may be bent and displaced. However, the four boundaries of the warp should not change.

A transformation is a function which maps a coordinate system to another: $(x, y) \mapsto (x, y) + (D^x(x, y), D^y(x, y))$. The displacement functions $D^x(x, y)$ and $D^y(x, y)$ determine how much transformation is made and it will be used to calculate the probability of a match. They should be continuous and satisfy the boundary condition, i.e. $D^x(0, y) \equiv D^x(1, y) \equiv D^y(x, 0) \equiv D^y(x, 1) \equiv 0$. Potentially there could be many ways to model the displacement functions. In [Anil K. Jain & Lakshmanan, 1996] they proposes the models as follows:

$$\begin{aligned} \mathbf{e}_{mn}^x(x, y) &= (2 \sin(\pi nx) \cos(\pi my), 0) \\ \mathbf{e}_{mn}^y(x, y) &= (0, 2 \cos(\pi mx) \sin(\pi ny)) \end{aligned} \tag{3.1}$$

$$\mathbf{D}(x, y) = (D^x(x, y), D^y(x, y)) = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \frac{\xi_{mn}^x \cdot \mathbf{e}_{mn}^x + \xi_{mn}^y \cdot \mathbf{e}_{mn}^y}{\lambda_{mn}} \tag{3.2}$$

In equation (3.2), $\lambda_{mn} = \alpha \pi^2 (n^2 + m^2)$, $m, n = 1, 2, \dots$ are normalizing constants and the parameters ξ_{mn}^x, ξ_{mn}^y can be viewed as projections of the displacement functions on the orthogonal basis, and measure the deformation. These two parameters are then plugged in a so-called prior probability function. I suggest interested readers to check the original article for details, however, the essential idea is that the prior distribution of a deformed template contains the term $\exp(-\|\mathbf{D}\|_2^2)$,

among other parameters.

Since there is a prior function, naturally there is a “likelihood” function too. According to [Anil K. Jain & Lakshmanan, 1996] the likelihood measures the similarity between the deformed template and the target image. The goal of their research is to match shapes, so the likelihood function contains the descriptors of edges, such as the rotation angle, scaling factor, etc. Ideally, a high level of similarity yields a high probability value. Some other control parameters are also incorporated here, such as smoothing factors. Finally, the prior and the likelihood are combined, and the objective is to minimize the posterior probability.

The concept of a warp serves to visualize the transformation functions. It displays intuitively how straight segments are changed due to stretching, squeezing, and twisting. One can also consider a warp as a sampling of pixels, and a follow-up question is whether we can base our computation on the warp alone, instead of going through all the pixels. It seems to be a good way to get a compact representation and reduce computational complexity.

In the study of [Anil K. Jain & Lakshmanan, 1996], the purpose is to match shapes and thus the edges of objects are compared. In this case one can deal with edge pixels only. However often times we may want to compare both the shapes and the textures, and then we cannot just use edge pixels, or any special collection of pixels. Generally speaking, a warp cannot cover most of the pixels of interest at all. When we compare two images, the content to be matched is distributed everywhere.

Our fossil photos do not have particularly distinctive shapes. They are shells after all, and most are oval, basically. Species that are very distantly related may differ in shape, but shape features alone cannot help too much for classification. Nonetheless the texture features on the shells exhibit a large variety and should be used. It would be nice to design a deformable template that can compare complex features such as texture.

[Slot & Gozdzik, 2008] designs a deformable template model to recognize fingerprints. As we know fingerprints are characterized by ridge patterns. One may tend to believe that prints from the same finger are always the same; however, it is not true. When people press their fingers, there are

always spatial variances of pressure. In short, one cannot reproduce exactly the same fingerprint again and again. Due to the variation, fingerprints need to be aligned for recognition.

In their article, a warp (called a “grid”) is also superimposed on the template image. In this case, the warp can be viewed as connected nodes, just like a graph. A node contains the local features in an image, and all the nodes together represent the whole image. For fingerprints, we are interested in the ridge patterns, which can be considered as a type of texture. [Slot & Gozdzik, 2008] assigns a two-dimensional vector to each node. The vector represents the orientation of the ridge. Under such a formalization, the similarity between two nodes can be measured by a vector inner product. In order to deform the template to find a better match nearby, the authors require the displacement to be along the first principle component of the Jacobian matrix of the vectors in nodes. In other words, the displacement is along the direction of maximum change. The magnitude is proportional to the similarity (i.e. inner product) value between the template node and the target.

The details of the techniques would be irrelevant to our discussion. However, an essential idea proposed by [Slot & Gozdzik, 2008] is that we can construct a warp as connected nodes. Each node stores local features of an image, and the features can be multi-dimensional. The connection between the nodes are elastic, i.e. they can be displaced, but with some cost. The deformation function in their model is rather simple, and is only driven by local features. Finally the deformed template image can be compared with any target image and a similarity value is delivered.

The model adopted in this project shares many characteristics with [Slot & Gozdzik, 2008]. The details will be discussed in next sections.

3.2 Algorithm for matching fossil images

We have argued paleontology is the study of fossils. Needless to say the processing of fossil images is crucial, if we want to build an intelligent knowledgebase. Scholars and general users may want to search for specific images, or classify new images they obtain. They may also be interested in using image as keys to locate textual entries. All the functions are based on the same procedure:

to compare images and associate one image with others by some criteria.

The fossil photos are all extracted from the pdf files and are greyscale images. As mentioned before, they are taken from different views: dorsal view(back side), ventral view (bottom side), and side view. So the same specimen may have several images from different angles. The images vary in size, too. Most are around 200 ~ 300 pixels in height and width, but some can be extraordinary in size. There is also a variety of shapes. As described in the Treatise, the fossil shells can be triangular, oval, rectangular, pentagonal and so on from dorsal and ventral views. Side views often yield very slim images. A few families of shells have special pikes which make them look like porcupines. As far as textures are concerned, on the one hand, the fossils (in particular ventral views) exhibit some crucial organs or systems in certain places; and on the other hand, most fossils have remarkable stripes or flecks. Paleontologists have made efforts to describe all the information in words, as we can find in the Treatise. For non-experts, however, the terminology can be extremely confusing. Even experts' descriptions may not be consistent and authors have different inclinations. English or natural language in general is not very good at describing subtleties of textures so we should not expect verbal descriptions to be highly accurate. Therefore, techniques based on natural language processing always have limitations. Hopefully, computer vision can shed some light.

“There's no two identical leaves”. Brachiopod fossils, even if they are from the same category, always differ from each other. That being said, closely related individuals must have evolved a similar set of organs and systems, and display similar patterns on the surface. We can predict that members of close kinship have near-identical “textures”. This is the fundamental rationale in our algorithm.

It is relatively easy to represent the shapes alone, and we have tried a method using the centroid and radii of an image. Nonetheless we decided not to implement it in the image comparison algorithm so I will not talk about too many details here. A brief description will be provided at the end of the section. In the following, I will focus on the deformable template model we choose to use.

3.2.1 Image model and the objective function

Our algorithm for comparing images has two major steps: (1) deform an image to resemble another image in the most optimal way; and (2) calculate the distance between the two images. When we compare two images, one of them is called the template, and the other is called the target. In this article the template is always the one to be deformed; and the target is the one to be matched on.

In order to perform deformation, we also superimpose a warp on the template image, similar to [Anil K. Jain & Lakshmanan, 1996] and [Slot & Gozdzik, 2008]. A warp contains a number of nodes, and each node is characterized by its coordinate and texture features within the surrounding area (neighborhood). Put it in another way, each node represents a patch of the image. In the deforming process, we treat the coordinates of template nodes as random variables, and each position on the target image is associated with a probability value. Deformation involves relocating nodes to places with different coordinates from their original locations, which is subject to some penalty (decreasing probability). Nevertheless, deformation can also be subject to rewards (increasing probability), if the new places on the target image have very similar textures to the original places on the template image. The goal is to find an optimal configuration for all the random variables, so the template nodes can be mapped on the target image. Thence based on the deformation, we can compute the corresponding distance between two images.

The warps in our model resemble [Slot & Gozdzik, 2008] because we use a node to encapsulate a multi-dimensional vector. That being said, there is a major difference in our model i.e. we do *not* assume pairwise associations between nodes. In most deformable template models, a node is and only is dependant on its neighboring nodes. In our model, on the contrary, it is dependant on *all* other nodes.

Deformation models seek to minimize some energy function, which is an idea borrowed from mechanics. One component of the energy function (often called the “prior”) contains the deformation cost, reflecting how likely a deformation can happen. Generally speaking, structures with high potential energy have low probabilities. In a truss construction, for example, the tension and bending moment on each beam are determined by the (usually very small) deformation of the

beam. Given a beam, the relative displacement between the two ends is sufficient to determine the energy caused by deformation on that particular beam. Summing all the potential energies, we can obtain the potential of the network. Most warp models work exactly in the same way: *only* neighboring nodes generate “forces” and store potential energy. Our model, on the contrary, resembles a fully connected “truss”, where every node supports $N - 1$ beams if N is the total number of nodes. In many deformable template models, the energy function draws an analogy from the laws of mechanics, such as Hook’s Law. Other models do not make such a strong assumption and only impose some general statistical principles. This is also our approach, to be discussed later.

In statistical graphical models, we often use a node to represent a random variable and draw an arch (connection) between two dependant nodes. A pairwise connection model with N nodes has $\Theta(N)$ connections, while a fully connected one has $N^2 - N$ connections. Needless to say a pairwise model is faster in terms of run time; however, there are good reasons for adopting a more sophisticated model. A shortcoming of the pairwise connection is that it is too “short-sighted” and thus incapable of modeling features over a distance, such as the derivative and 2nd order derivative of values of random variables. The only thing a pairwise connection can do is to associated random variables with the ones next to them, and prefers them to have similar values. In an image, pixels often do have long-distance dependencies and demand larger degrees of connections. In Figure 3.1, we see two deformations (from left to right). If the pairwise connection model is assumed, these two deformations have exactly the same penalty because on average, the nodes on the right have the same degree of displacements related to their paired nodes. However, with a fully connected model, these two penalties may not be equal. The relative displacements among all possible pairs are not the same, for example.

Talking about an optimization problem, the first thing coming to mind is an objective function. We may consider our objective as the product of a prior term and a likelihood term. The prior term is a probability function reflecting what a well-formed warp should be. The more a warp is ill-formed, the lower the probability value is. The random variable values of the probability function are coordinates of the mapped nodes on the target image. We usually use $[x, y]$ to denote

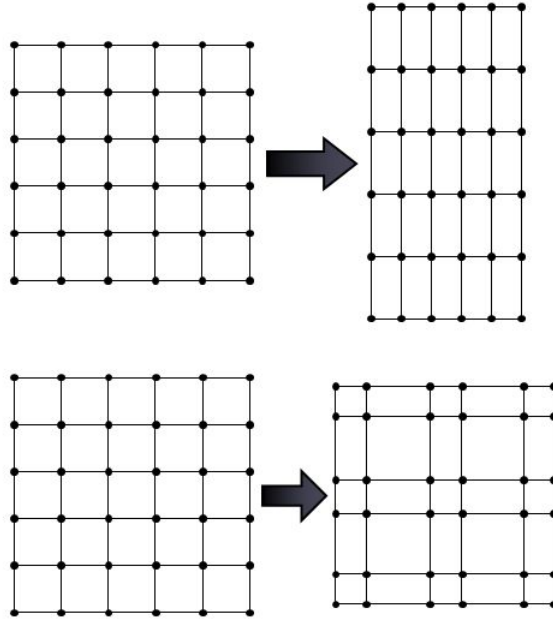


Figure 3.1: Deformation costs of grids. With pairwise connection model, only distances between neighboring nodes are considered. Both cases have the same total cost. With full connection model, all possible pairs of nodes are considered, and the costs differ.

coordinates, however in machine learning literature, we tend to reserve letter x for data and y for labels. Thus henceforth I will use $\mathbf{x}_i = [x_{i1}, x_{i2}]$ to denote the coordinates of a node i on the target image. Also x_{i1} is now corresponding to the row index in a matrix (instead of horizontal coordinate), and x_{i2} is the column index. On a $N \times M$ image, the coordinates of nodes can define a mapping $f : \mathbb{N}^2 \rightarrow \mathbb{N}$:

$$f(\mathbf{x}_i) = f(x_{i1}, x_{i2}) = (x_{i2} - 1)N + x_{i1} \quad (3.3)$$

In other words, the coordinates are corresponding to natural numbers between 1 and MN . Now we can define a probability space (Ω, \mathcal{F}, P) . An outcome in Ω would be an instantiation of a node, or an assignment of coordinates to a node. \mathcal{F} is a collection of events (sets of outcomes) and P is the probability. A random variable \mathbf{x}_i can be viewed as a function that maps an outcome in Ω to a natural number. Every node has a unique identifier i . However, the nodes are always placed on a warp, which is actually a matrix, so equivalently, we can also use the matrix index $[i1, i2]$ on the warp to identify a node. Suppose the warp is $m \times n$, now we can consider the random variable \mathbf{x}_i

as a mapping from $(i1, i2)$ where $i1$ and $i2$ are determined by i , to a natural number in $[1, MN]$. In short, I will use $P(\mathbf{X})$ to represent the joint probability of *all* the random variables, which is also the probability of a deformation. I will also call \mathbf{X} a (multivariate) random variable when it is not confusing, although it may be better to call it a *random field*. Formally, $\mathbf{X} : \mathbb{N}^2 \rightarrow \mathbb{N}$.

Our first assumption is, the random variable \mathbf{X} has a multivariate Gaussian distribution. The mean vector of the Gaussian prior, naturally, is the original positions before deformation, which are given by the template warp. Put another way, the prior assumes no deformation to be optimal. We penalize any deviation from the original positions.

Another reasonable assumption is, nodes that are close to each other tend to have stronger correlations than those far away. If one node moves rightward (maps to a larger natural number), statistically speaking, it's neighbors are very likely to move rightward too. Nodes that are farther away would have a weaker tendency to move rightward. Therefore, the autocorrelation function of X should peak at $(0,0)$ and decay rapidly outward. The autocorrelation function is defined in below (with some simplification):

$$ACF_X(\xi, \eta) = \frac{\frac{1}{NM} \sum_{k=1}^N \sum_{l=1}^M X(k, l) X(k + \xi, l + \eta)}{\left(\frac{1}{NM} \sum_{k=1}^N \sum_{l=1}^M X(k, l)\right)^2} \quad (3.4)$$

Interestingly, the ACF of a signal and the power spectrum of the signal form a Fourier pair: $\mathcal{F}[ACF_X] \sim |\mathcal{F}[X]|_2$. Therefore it is easy to use a filter to control the autocorrelation matrix. As an example, if we want the autocorrelation matrix to be as simple as an impulse (with non-zero value only on the first element), we can just use a filter to make the power spectrum flat, as a uniform matrix. The reason why we want the autocorrelation matrix to be an impulse is that in this case, the covariance matrix of the random variables are diagonal. In other words, a random variable is only correlated with itself. Such a structure is very nice for efficient commutating, which will be discussed later. For now we just need to know that \mathbf{X} is “whitened” before further processing. The aim is to decorrelate the nodes on a warp. In addition, we assume the power spectrum of \mathbf{X} obeys

a power law:

$$P(f) \propto \frac{1}{f^\alpha} \quad (3.5)$$

where $\alpha > 0$ and f is the frequency amplitude. $P(f)$ is the power density. The parameter α is related to the designing of the whitening filter. We assume it to be 2.

Autocorrelation function is closely related to covariance matrix. We consider \mathbf{X} stationary, so the covariance matrix is a *Toeplitz matrix*, and by definition of covariance it is symmetric too. All the elements in the autocorrelation matrix form the first column of the covariance matrix, and the other columns can be obtained by shifting the first column, or using the symmetric property. A Toeplitz matrix can be approximated by a circulant matrix, in which every column has the same elements, but is shifted. A circulant matrix has a nice property: it can be diagonalized by a Fourier basis i.e. $S = F\Lambda F^*$, where F is a matrix of Fourier basis and Λ is a diagonal matrix. Also it can be proved that the diagonal elements of Λ is the power spectrum of \mathbf{X} .

Let $W = F\Lambda^{-\frac{1}{2}}F^*$, it is easy to see that $Cov[W\mathbf{X}] = E[W\mathbf{X}(W\mathbf{X})^T] = I$. Thus, if we transform \mathbf{X} to $W\mathbf{X}$, the covariance matrix is going to be an identity matrix. In other words, after the transformation the random variables are “whitened”. We have just talked about the whitening filter in the frequency domain. The W here can be viewed as a Fourier pair of the filter. In fact, if we take a column of W and convolve it with the random field \mathbf{X} , we can still get the whitened variables.

Formally, we can write the prior (without whitening) as in Equation 3.6. In sum, the prior probability is a Gaussian function. The mean vector μ is the original coordinates, without any deformation. We make an assumption that the autocorrelation function is the inverse Fourier transform of a power law function. We choose the power to be 2. The covariance matrix S is a Toeplitz matrix. The elements in the first column is the same as the elements in the autocorrelation matrix. Gaussian functions are often chosen as priors in statistical inferences. For a discrete Gaussian, the range of a random variable is always between $[0, 1]$ so it is easy to interpret it as a probability. The function is also relatively easy to parameterize.

$$P(\mathbf{X}) \propto \mathcal{N}(\mu, S) \quad (3.6)$$

The random variables are whitened before processing, so their correlations are removed. After whitening, the prior covariance matrix is a scaled identity matrix and thus the function can be factorized, as in Equation 3.7. I use symbol \mathbf{X}_w to denote whitened random variables. In the following text, however, I will omit the symbol for “whitening” and only write \mathbf{X} as long as it is not confusing. At the end of computation, the \mathbf{X} needs to be “dewhitened” by the reciprocal of the whitening filter.

$$P(\mathbf{X}_w) \propto \exp[-\alpha \sum_i (\mathbf{x}_{i_w} - \mu_i)^2] = \exp[-\alpha D_1(\mathbf{X}_w)] \quad (3.7)$$

After talking about the prior, we can now focus on the likelihood. The likelihood term reflects the similarity between the deformed template image and the target image. As mentioned before, a node represents local texture features of an image patch. Therefore a node on the template warp encapsulates a feature vector of texture features to begin with. After deformation, the node has a mapped location on the target image, and will obtain local texture features too. Then we utilize the two feature vectors to compute their distance, and correspondingly, the likelihood. The likelihood term for the whole warp is the product of likelihoods for individual nodes. The details of the texture features will be discussed later. Here we just need to know the likelihood function is in the form of:

$$P(I|\mathbf{X}) \propto \exp[-\beta \sum_i \|\Delta \mathbf{f}_i(\mathbf{x}_i)\|_2^2] = \exp[-\beta D_2(\mathbf{X})] \quad (3.8)$$

We use $\|\Delta \mathbf{f}_i\|_2^2$ to represent the squared L_2 norm distance (Euclidean distance) based on feature vectors of node i . I stands for the target image. Now it is clear that our goal is to optimize the posterior function in Equation 3.9:

$$P(\mathbf{X}|I) \propto \exp[-(\alpha D_1(\mathbf{X}) + \beta D_2(\mathbf{X}))] \quad (3.9)$$

In Equation 3.9, parameter α is related to the covariance matrix of the prior and controls deformation cost. A big value penalizes deformation heavily. Parameter β penalizes texture differences (data differences). After we find the optimal configuration of X , we can plug the value back in

equation 3.9 to find the “probability” value. In reality, however, the probability value is often infinitesimal. It is because our prior and likelihood functions are only proportional to Gaussian functions, but missing the coefficients. Anyway, the probability score is only for human interpretation, so we can boost it by scaling the total cost down. Generally speaking, we want the deformation cost and the data cost to be in the same range, depending on the nature of the data. These parameters are adjusted in experiments, which I will discuss later.

Now the question is how to find the optimal configuration of \mathbf{X} . A naïve approach is to go through all the states of variables and find the pattern that maximizes the empirical posterior directly. However, suppose each node has M possible locations to be placed at, and an image has N nodes, the problem is would be $\Omega(M^N)$ in complexity. In addition, to infer the mode of a distribution from empirical data may have other problems. In the next section I will talk about how to approach a solution.

3.2.2 Expectation propagation

As we know, performing an exact inference is intractable, so we want to find an approximation for $P(\mathbf{X}|I)$. As long as it does not cause confusion, I will write $P(\mathbf{X}|I)$ as $P(\mathbf{X})$ for simplicity. Now we want to find a tractable probability function $Q(\mathbf{X};\theta)$ to approximate $P(\mathbf{X})$. This is a density estimation problem and we seek to minimize the *expectation* of the loss function $\alpha D_1(\mathbf{X}) + \beta D_2(\mathbf{X})$. Equivalently, we can say the aim is to minimize the risk function:

$$R(\theta) = \mathbf{E}_P[-\log(Q(\mathbf{X};\theta))] = - \int P(\mathbf{X})\log(Q(\mathbf{X};\theta))d\mathbf{X} \quad (3.10)$$

It can be proved that minimizing the risk function is equivalent to minimizing the Kullback–Leibler divergence (commonly known as KL-divergence) between P and Q above. KL-divergence is defined in Equation 3.11.

$$KL(P||Q) = \int P(x)\log\frac{P(x)}{Q(x)}d\mathbf{X} \quad (3.11)$$

Obviously, $KL(P||Q) = R(\theta) + \int P(\mathbf{X})\log(P(\mathbf{X}))d\mathbf{X}$. The second term is called the *entropy* of P and it does not affect minimization. Therefore we can transform the optimization problem to minimizing the KL-divergence. By choosing a proper Q from the exponential family, the approximation is relatively easy. Assuming Q has the form:

$$Q(\mathbf{X}; \theta) \propto \exp(\eta(\theta) \cdot \mathbf{T}(\mathbf{X})) \quad (3.12)$$

It can be proved that in order to minimize $KL(P||Q)$, we need to satisfy $E_P[\mathbf{T}(x)] = E_Q[\mathbf{T}(x)]$. In other words, we need to match the moments between P and Q . Generally speaking, the solution is still intractable and this is why we need the method of expectation propagation (EP). The EP technique was firstly proposed by Minka [Minka, 2001a, Minka, 2001b] and has been widely used in computer vision.

For this purpose, however, we need to make some more assumptions. One important assumption is, the distribution can be *factorized* i.e. it can be written as the product of some *potential functions*.

$$P(\mathbf{X}) = \frac{1}{Z} \prod_{i=1}^N \phi_i(\mathbf{x}_i) \quad (3.13)$$

A potential function is a function of random variables, and always yields a nonnegative value. For each potential function $\phi_i(\mathbf{x}_i)$, the random variables are a subset of the random variables in $P(\mathbf{X})$, which is why we call it a factorization. The \mathbf{x}_i here represents a subset of random variables, and thus it has a lower dimensionality than \mathbf{X} . In order to resemble probability density functions, people use a *partition function* Z to ensure the value integrates to 1, which is not important in our case. In the previous section, we already assumed the distribution can be factorized. In our case, \mathbf{X} represents the coordinates of all the nodes. Given N nodes, \mathbf{X} would be $2N$ in dimensionality, because each node has two coordinate variables. Each individual potential function has an argument \mathbf{x}_i , which has two components.

If a distribution factorizes like this, expectation propagation can be a good fit for the model.

As mentioned before, the \mathbf{x}_i in a factor $\phi_i(\mathbf{x}_i)$ has a lower rank and its moments could be computed fast. Therefore, instead of matching the moments of P directly, we may choose to fix q_i , a factor of Q iteratively. Suppose our Q can be factorized as:

$$Q(\mathbf{X}) = \frac{1}{Z} \prod_{i=1}^N q_i(\mathbf{x}_i) \quad (3.14)$$

Such a choice is certainly feasible. For example, we can assume Q is a product of Gaussian functions. Gaussian functions are often chosen for their simplicity.

One may attempt to approximate each ϕ_i with q_i . Although it is doable, it limits the solution to the ones that match the moments of individual q_i only. If the type of distribution of ϕ_i does not fit the chosen q_i (usually Gaussian) well, the errors from individual approximations can accumulate and the result may be a poor approximation of P . After all, our goal is to minimize $KL(P||Q)$ and other *ad hoc* constraints are unnecessary if not erroneous. EP does it in a much more elegant way. Instead of matching the individual factors ϕ_i , we choose a q_i such that

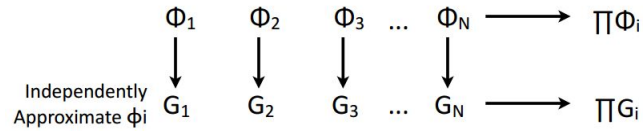
$$q_i = \underset{q}{\operatorname{argmin}} KL(\phi_i \prod_{j \neq i} q_j || Q) \quad (3.15)$$

In this way, we omit one factor of Q at one time, and multiplies it with the empirical factor ϕ_i , and then choose q_i to minimize the KL-divergence between the product and Q . After that, the obtained q_i is plugged back in Q , and thus the approximation function gets updated iteratively.

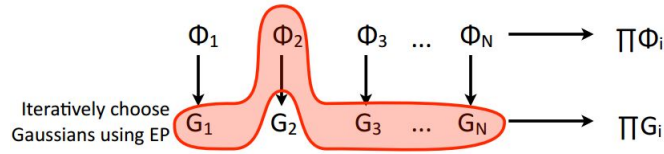
It should be noted that there is no theoretical guarantee that EP will certainly converge to the expectation of P . That being said, EP usually works well in practice and we normally did not meet trouble in our experiment. In the following sections, I will explain how the algorithm is implemented.

3.2.3 Image preprocessing

Before images can be compared, some preprocessing is necessary to assure the quality. We do not use sophisticated techniques to change the original content of the images. Basically we did two



(a) Use Gaussian functions to match individual factors one by one. Expectation propagation does not work this way.



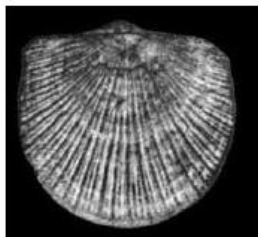
(b) An example of expectation propagation: select the Gaussian function G_2 so that $\phi_2 \prod G_i / G_2$ and $\prod G_i$ match moments.

Figure 3.2: A graphical illustration of expectation propagation. Image courtesy of Dr. Brian Potetz

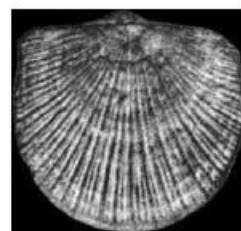
things: cropping the margins and resizing images, if necessary.

Margins (i.e. blank parts) are necessary when we store and display images. When we construct the deformable templates, however, we want to eliminate them. The following steps are implemented in order to crop the images:

1. Detect the border of the fossil;
2. Find the maximum and minimum values of x and y coordinates on the border of the object i.e. the fossil;
3. Crop the image and retain the region that satisfies $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$.



(a) Before cropping



(b) After cropping

Figure 3.3: Cropping images. Complement images are used here to show the margins. Margins are white (as opposed to black here) in real cases.

The algorithm to detect borders has been discussed in Chapter 2 and will not be repeated here. In practice we also add two pixels' margin to each of the extreme point. i.e. If $x_{min} = 6$, we crop at $x = 4$; if $y_{max} = 300$, we crop at $y = 302$.

Most fossil images have similar sizes and do not need to be resized. That being said, we require the height and the width of the template image to be no larger than 300 pixels. Over-sized images are shrunk to fit in 300 pixels. For target images, two options are provided. One can either resize it to match the size of the template before processed, or keep the original size. The former option applies to the situation where “size does not matter”, and the latter penalizes differences in size.

We use MATLAB function `imresize` to resize images. By default, this function uses *linear interpolation*, though some other methods are allowed.

3.2.4 Construct the warps

In this section I will explain how the warps are constructed so images can be deformed.

In the first step, A warp is superimposed on the template, and another one on the target. A warp consists of nodes. Each node has coordinates and a vector of features representing the texture in the local neighborhood. The coordinates are corresponding to their position over the image. The coordinates are also regarded as features *per se* and will be used when we compare images. By default, a warp is 15×15 in size and the nodes are evenly placed. Each node occupies a “neighborhood”, and all the texture features for this node are drawn from this neighborhood only. The next section will explain how the features are computed. The default size of the neighborhood is the same as the node interval. Therefore, the neighborhoods of two adjacent nodes have 50% overlap, which is good in many cases because we want to make sure the image is fully covered.

Figure 3.4 demonstrates a warp over an image, without any deformation. The small circles represents nodes. The lines help to demonstrate the relative positions between nodes. The number of nodes over the target image is set to be the same as the template image. Consequently, the distance between nodes varies, depending on the size of the image. The size of the neighborhoods is still a constant by default.

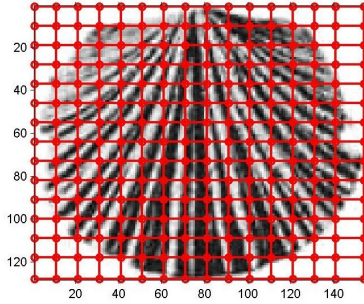


Figure 3.4: A visualization of a warp. Each circle represents a node.

Now, the problem is how to deform the warp. I will discuss how the features of the nodes are computed first, and then explain the algorithm to deform a template.

3.2.5 Texture features

Texture is something we all can perceive but there is no universal representation in mathematic terms. The specific choice of model depends on the nature of data, as well as researchers' personal preferences.

Generally speaking, texture is related to smoothness, coarseness, roughness, and regularity of the surface. From the perspective of vision, texture is determined by illumination and reflectional conditions of the object. When fossil photos were taken, the illumination was carefully controlled so this variable can be ignored. The most straightforward way to analyze texture is to extract power spectrum from the images.

The power spectrum is defined by the magnitude of the spectral components. For example, if we choose 2D Fourier transform to construct the spectrum, the power in a given spatial frequency band would be:

$$P = \sum_{u,v \in D} |F(u,v)|^2 \quad (3.16)$$

where D is the spatial frequency domain of interest, $F(u,v)$ is the two-dimensional discrete Fourier transform of the original image. Note $F(u,v)$ has complex values in general, and $|F(u,v)|$ denotes its magnitude. High power in low frequency domain corresponds to coarse textures— brightness patterns that change over long distances. On the contrary, high power in high frequency domain

corresponds to fast changing patterns, until the frequency is too high to perceive and it appears smooth. The directionality of texture (if any) can also be revealed by the power spectrum.

Besides Fourier transform, other transforms are often implemented to constructing the spectrum, too. In literature there are a lot of discussions on cosine transform, Walsh-Hadamard transform, Haar transform and so on. Different transforms use different basis functions and thus interpret the image in different ways. Usually a *filtering* procedure follows a transform, to highlight or reject certain components. In a broad sense, a transform function can be considered a filter too. The power spectrum method is handy especially if we have some prior knowledge about the textures to be analyzed. When the data are highly diverse, or we are not certain about the characteristics of data, it may be difficult to select the optimal parameters, including the transforms to use.

Another commonly used method, also adopted in this project, is the *gray level co-occurrence matrix (GLCM)* method. In this approach, we construct a gray level co-occurrence matrix of the image first, and then compute some second-order statistic features from the matrix. This approach is flexible and can be used for all kinds of textures. Before creating a gray level co-occurrence matrix, we need to decide how many levels we need. In our project we use 8, so the intensity of image pixels are linearly mapped to 8 levels. i.e. The darkest pixels map to 1, slightly brighter pixels map to 2, and the brightest pixels map to 8. An image is thus converted to an octo-valued matrix. Then we count the pairs of adjacent pixels, and record how many instances of (1, 1), (1, 2), (1, 3)...(2, 1), (2, 2)...(8, 8) are found, respectively, and save the number of instances in a 8×8 matrix, the so-called co-occurrence matrix.

Figure 3.5 shows how to construct a gray level co-occurrence matrix from a hypothetical 5×5 image. The pixel intensity values have already been discretized to integers between $1 \sim 8$. In the figure, only left-right pairs are counted. In our algorithm, however, we count co-occurrences in four directions, as illustrated in Figure 3.6, and construct four separate matrices. Often times, the textures of the fossils do have some directionality and we consider it a good idea to try more directions here. In principle, the pairs do not have to be neighbors. We could set higher “offset” values so long distance co-occurrences are used. Nevertheless using adjacent pairs is the most

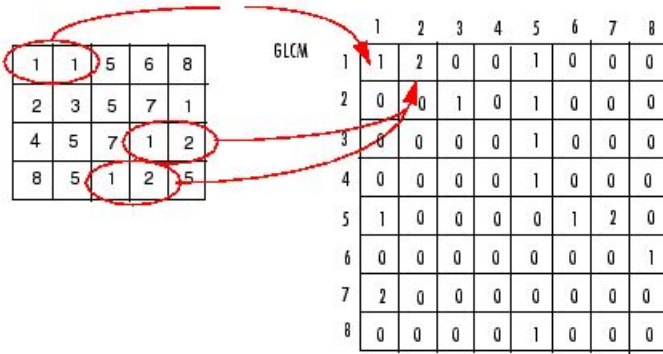


Figure 3.5: Illustration of gray level co-occurrence matrix. In this example, only left-right pairs are counted.

common and we follow the convention too.

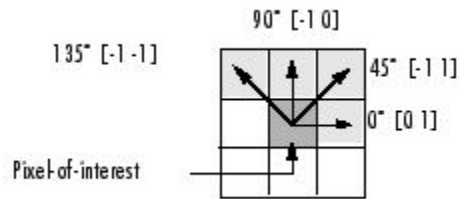


Figure 3.6: Four directions to count co-occurrences. The arrays are used to represent directions and offsets in the MATLAB function.

After co-occurrence matrices are obtained, we can compute the features. For each matrix, we compute 4 values: contrast, correlation, energy and homogeneity. The definitions are given in below:

$$\begin{aligned}
 \textit{Contrast} &= \sum_i \sum_j |i - j|^2 p(i, j) \\
 \textit{Correlation} &= \sum_i \sum_j \frac{(i - \mu_i)(j - \mu_j)p(i, j)}{\sigma_i \sigma_j} \\
 \textit{Energy} &= \sum_i \sum_j p(i, j)^2 \\
 \textit{Homogeneity} &= \sum_i \sum_j \frac{p(i, j)}{1 + |i - j|}
 \end{aligned}
 \tag{3.17}$$

where:

$$\mu_i = \sum_i i \sum_j p(i, j)$$

$$\mu_j = \sum_j j \sum_i p(i, j)$$

$$\sigma_i^2 = \sum_i (1 - \mu_i)^2 \sum_j p(i, j)$$

$$\sigma_j^2 = \sum_j (1 - \mu_j)^2 \sum_i p(i, j)$$

Since we have 4 co-occurrence matrices, and each yields 4 statistic measures as shown above, the feature vector will have 16 components. The range of correlation is $[-1, 1]$, and the range of energy or homogeneity is $[0, 1]$. However, the range of contrast can be $[0, 49]$, though extreme values are rare. In our implementation, all the values are normalized to $[0, 1]$ before processing.

As a result, each node in the warp has 2-dimensional coordinates and 16-dimensional texture features. Therefore, a 18-dimensional vector can fully represent a node. With these vectors, the distance between images is defined. Eventually, for each node we want to have a function that maps the coordinates to a probability value. The following sections will show more details.

3.2.6 Implementation of expectation propagation in deformable templates

Given the idea of EP, the specific implementations can vary. Generally speaking, Gaussian EP performs in four steps:

1. Initialize the mean μ and variance S of the Gaussian function G .
2. Compute the parameters of G with G_i left over i.e. compute $S_{\setminus i} = (S^{-1} - S_i^{-1})^{-1}$ and $\mu_{\setminus i} = S_{\setminus i}(S^{-1}\mu - S_i^{-1}\mu_i)$.
3. Update S and μ to match the mean and variance of $\phi_i(\mathbf{x}_i)G_{\setminus i}(\mathbf{X})$.
4. Update $S_i = (S^{-1} - S_{\setminus i}^{-1})^{-1}$ and $\mu_i = S_i(S^{-1}\mu - S_{\setminus i}^{-1}\mu_{\setminus i})$.

Initialization can be trivial. In the most simple case we can just start with a uniform distribution. In practice, Step 2 can be simplified too. We did not compute $S_{\setminus i}$ and $\mu_{\setminus i}$ specifically, but use S

and μ from the previous iteration (if any) as an approximation. Put in another way, we skip Step 2 and use G to approximate $G_{\setminus i}$ in Step 3. It does not affect the result much, especially because each factor only contains a small subset of variables.

For our problem, if there are N nodes on the warp, the covariance matrix S is $4N^2$ in dimensionality. If N becomes big it will be fairly inefficient to maintain such a high-rank S . Our implementation adopts an algorithm [Potetz & Hajjarbabi, 2013] to reduce the space requirements. As mentioned before, the random variables are “whitened” at the beginning, and thus the covariance matrix of the prior is diagonal. A diagonal matrix is linear with respect to the number of nodes. In the computation, we also constrain S to have the form $W^{-1}D_SW^{-1}$, where D_S is a diagonal matrix and W is the whitening matrix as we discussed before. By maintaining D_S and $W\mu$, computing efficiency is improved. The algorithm *per se* is not a contribution of this thesis, and interested readers can refer to the original article.

Generally speaking, the run time of EP algorithm for a problem like ours is $O(D^3)$ where D is the dimensionality of data. Using the method of Sparse Inverse Covariance EP, the complexity can be reduced to $O(D^{\frac{3}{2}\rho(\mathcal{G})})$, where $1 \leq \rho(\mathcal{G}) \leq 2$, depending on the sparsity of S [Alon & Yuster, 2010]. With whitened EP, however, the complexity is only $O(D)$.

Finally, the algorithm outputs the “best” coordinates for each nodes on the target warp. These coordinate values are discretized so they are corresponding to actual node positions. Recall we do not allow the nodes to be everywhere, to save computation complexity. Instead, they are evenly distributed over the image. When deformation takes place, a target node can only move to one of those proposed positions. After all the relocation is done, we compute the probability score based on both texture similarities between the corresponding nodes, and the degree of deformation made in total. The higher the score, the better the match.

3.3 Summary

In this chapter we have reviewed some techniques of image recognition, especially eigenfaces and deformable templates. Our method is based on deformable templates i.e. we deform an image in order to match it with another image. Their similarity is calculated based on texture features, and deformation is also penalized. Then with the best possible match, we record the similarity score and use it image comparison and recognition. Texture features are represented with some second order statistics of the gray level co-occurrence matrix (GLCM).

We used expectation propagation to find the best match. In order to improve efficiency, we adopt the technique of whitening expectation propagation. Before discussing the algorithm, we also discussed image preprocessing and the construction of warps. The techniques and parameters have already been introduced along with the discussion. In the next chapter, we are going to see the results, with some different set of parameters.

Chapter 4

Experiments and results of image recognition

In this chapter I am going to introduce a set of experiments and the results of the image recognition method based on deformable templates.

4.1 Demonstration of deformation results

To demonstrate how the algorithm works, we can take a look at some images in Figure 4.1. We set parameters $\alpha = 1/16$, $\beta = 2$. The images are all from the brachiopod family *Orthida*, but not necessarily the same genus. The template image (a) is a specimen from the genus *Nanorthis*. The warp happens to be 15×16 , determined by the size of the image. According to our algorithm, each of the target images will be automatically assigned a 15×16 warp too, in order to resemble the template. However the distance between two nodes can vary, dictated by the size of the target image.

Image (b) in Figure 4.1 is the same as the template. As we can see the warp has limited deformation, as expected, but the nodes do not stay in their original positions either. This is an idiosyncrasy of deformation based on expectation propagation. The best match is often not the one that exactly echoes empirical data. Moreover, in real computations, we only allow the nodes to

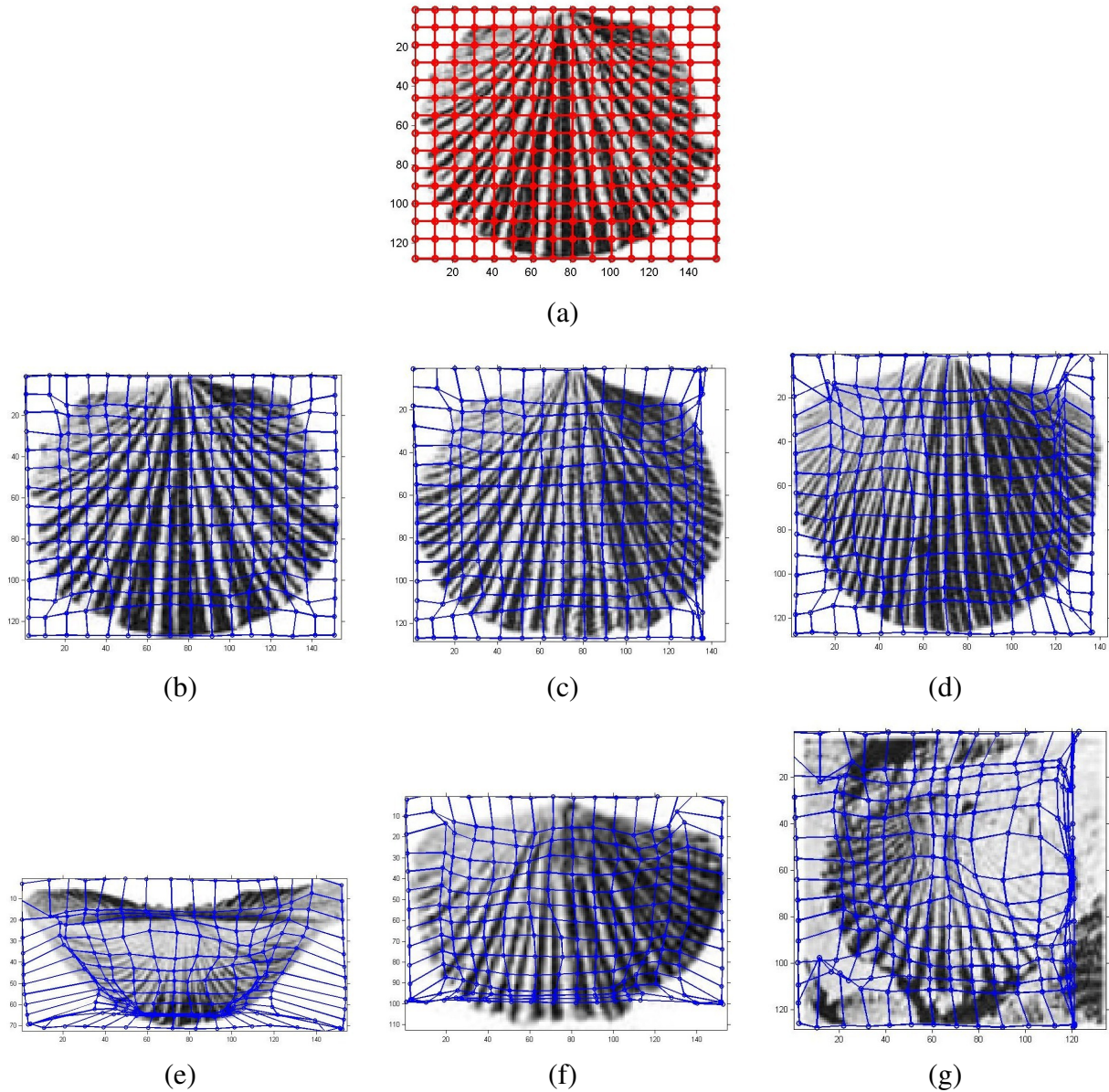


Figure 4.1: Results of deformation (without discretization). All images are from the family Orthida. (a). Dorsal view of genus *Nanorthis*, used as template image; (b). the same as the template; (c). another specimen of *Nanorthis*; (d). dorsal view of *Cyrthonotella*; (e) posterior view of *Cyrthonotella*; (f). dorsal view of *Nothorthis*; (g) a *shoshonorthis* fossil embedded in rock.

move to allocated positions, which are evenly distributed over the target image. We do not allow the nodes to try everywhere, because it would be too time-consuming and unnecessary. Figure 4.1 shows a deformed warp without discretization, just in order to display the effect of deformation better.

Image (c) is another specimen from the same genus. As the figure reveals it is quite similar

to the template image, but with a slightly bigger size. In the program we have a logic variable to control the effect of size discrepancy. If the variable is turned off, images will be resized to match the template size before being processed, which is what we did for the experiments here. Compared to (b), we also see more movements of nodes in (c), which is anticipated because it is already a different specimen.

Images (d)~(g) are specimens from other genera and thus we observe more deformations in general. Image (e) is a posterior view, so the shape greatly differs from the template. Sometimes the textures are very different and even deformation cannot help anyway, and as a result there is little deformation. Sometimes there seem to be fewer nodes on the target image than on the template image. This is only because some nodes map to the same position. There are always the same number of candidate nodes on the target as on the template.

The outmost nodes (“edge nodes”) usually do not move inward because extra high costs are imposed on them. In the implementation, we actually added another prior to constrain the edge nodes specifically. These nodes do not reflect texture features and we consider it appropriate to force them to stay.

Item	Prob. Score	Deformation Cost	Data Cost
(b)	0.999	0.3818	0
(c)	0.906	23.8047	25.4978
(d)	0.820	25.5791	73.9499
(e)	0.376	313.1061	175.6499
(f)	0.807	70.3164	37.0779
(g)	0.684	128.7224	61.2400

Table 4.1: Matching results of images in Figure 4.1. Scaling parameter = 2×10^{-3} .

Table 4.1 shows the numerical results. The probability score reflects how close a match is. The theoretical maximum value is 1 and the minimum is 0. The deformation cost is the penalty for moving nodes. The data cost is the penalty corresponding to texture feature mismatches, after the deformation takes place. Their relationship is illustrated in Equation 4.1. Not accidentally it looks very similar to the posterior probability function shown in Equation 3.9. At this time, however, we use a small constant γ to scale the total cost down, as discussed before. This is only for better

human interpretation and does not really affect the results. For the results in Table 4.1, we set $\alpha = 1/16$, $\beta = 2$ and $\gamma = 1/500$.

$$score = \exp[-\gamma(\alpha D_1(\mathbf{X}) + \beta D_2(\mathbf{X}))] \quad (4.1)$$

4.2 Effect of number of nodes

In practice, the algorithm is reasonably efficient. Running on a regular laptop PC, it takes about 1 ~ 2 seconds to process one image, using the data and parameters shown in Figure 4.1. The run time largely depends on how many nodes we want to use. Using the same images as in Figure 4.1 but with different numbers of nodes, we obtain the results shown in Figure 4.2.

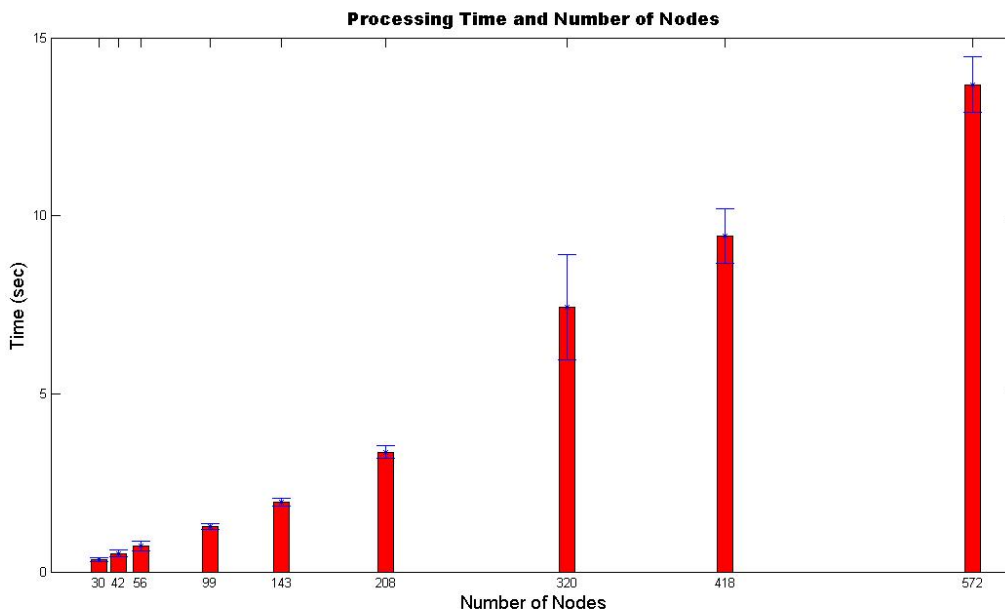


Figure 4.2: Processing time and number of nodes. Calculation is based on the images shown in Figure 4.1

As we can see, the run time increases with the number of nodes, but not exponentially so. As discussed before, the algorithm is suppose to be linear to the number of nodes in theory. In reality it has a higher order than linear, due to some implementation issues. By default we set the interval between two adjacent nodes as 1/14 of overall length in respective dimensions. Usually such a

setup creates about 225 nodes for a typical image in our data set. One may reduce the number by setting bigger intervals between nodes, and then the process can speed up with some possible sacrifice in quality.

Another factor that may affect the speed of computation is the size of neighborhoods i.e. the size of the patch represented by a node. We do not want the size to be too small, because texture features are meaningful only within a reasonably big patch. That being said, we do not want the size to be too big for at least two reasons. First of all, obviously, a big neighborhood demands more time to compute the co-occurrence matrix and the statistics; and secondly, when the neighborhood becomes too big, two nodes will share a large portion of territory and they can hardly be distinguished. Usually we set the neighborhood to be the same length of node intervals, in each dimension. Therefore two adjacent neighborhoods share about half of their pixels. Experiments showed that the time for processing does not change much, when the neighborhood scale ranges between 0.7 to 2 times of node interval.

4.3 Experiment and evaluation

After the discussions on algorithms and parameters, now the question is how good the results can be. Strictly speaking, there is no ground truth for us to evaluate the results, and we have to make some reasonable assumptions. Since the specimens are classified in biological categories (orders, families, genera etc.), we may assume that given a “key image” of known categories, the best matches i.e. those with high probability scores should exhibit a close kinship. At genus level this assumption is largely valid. However if we talk about families and orders, the specimens that belong to the same category do not look particularly similar. From an evolution perspective, organisms sharing a recent ancestry possess some subtle homologous features but they do not necessarily exhibit morphological identity in general.

Currently we have over 22,500 fossil images. They are cropped from about 1,500 figures from the Treatise. Typically, the images from the same figure are closely related. Sometimes they are

different samples of the same genus; sometimes they are different views of the same specimen. Despite that more than one genera can be displayed in the same figure, in this case they belong to the same family, at least. Suppose we use an image as a key to search in the data set, we expect some images originally from the same figure as the key image can yield high scores. In addition, if the image itself is among the target images (i.e. self comparison is allowed), then we should be able to see a near perfect score. This is the philosophy of our first experiment, to test the validity of the algorithm.

4.3.1 Experiment 1: matching images

In this experiment, we randomly selected 25 images of dorsal or ventral views as a set of key images. We use dorsal and ventral views because they represent shapes and textures in the best way. Other than that, We only avoided the images that are too small or hand-sketched (not a photo), because they are not meant to be our focus of study. These 25 images are from 25 different figures of the Treatise. We then collected all the images from those 25 figures as our test set. The key images themselves are included in the test set as well. Totally there are 420 test images. Each of the 25 key images is used as the “template” one by one, and matched against all the test images. In each case, the top 10 matches are recorded. Among the top 10, those that are from the same figure as the key image (including itself) are called a “valid match”. We ran the program on the Cycle4 server of the University of Kansas. If all the data are processed sequentially in a single MATLAB program (which is not necessary), it takes about 5 hours to complete. Table 4.2 shows the valid matches of the 25 images.

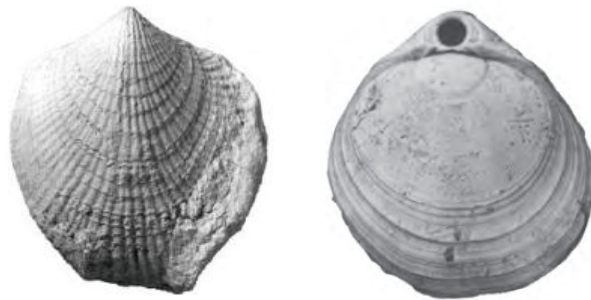
As we can see from Table 4.2, the algorithm always returns some valid match(es) and in the vast majority of cases (21 out of 25), there are more than one of them. We have 420 images in total and the maximum number of images from a single figure happens to be 30 (corresponding to Image No. 21). Therefore as long as 1 valid match is returned, it is already significantly beyond random chance. It is not shown in the table but in all the cases, a valid match also has the top ranking score, and the score is close to perfect. This is not surprising because an image is supposed to have a high

Item	# V.M.	#V.M. by Chance	2nd Best V.M. Ranking
1	3	0.143	4
2	3	0.381	5
3	3	0.119	2
4	3	0.357	5
5	2	0.310	8
6	1	0.214	NA
7	2	0.571	7
8	3	0.333	4
9	2	0.357	2
10	2	0.405	2
11	4	0.524	4
12	3	0.619	5
13	3	0.262	2
14	2	0.429	2
15	6	0.548	2
16	4	0.381	2
17	2	0.286	2
18	3	0.262	2
19	2	0.476	2
20	6	0.667	2
21	4	0.714	2
22	1	0.643	NA
23	2	0.310	8
24	1	0.429	NA
25	2	0.262	3

Table 4.2: Matching results of 25 key images. V.M. means valid matches. # V.M by Chance is defined as the number of images from the figure divided by the total number of images in the 25 figures, and then multiplied by the number of returned items (10).

matching score with itself for any legitimate algorithm. Often times, the second best valid match is ranked No. 2 as well, which is desirable.

Three images among the 25 do not have any second valid match. With a close look, Image No. 6 belongs to a relative small group (random chance 0.214). The photos have some illuminating issue, and probably the specimens are quite diverse as well. Image No. 22 has a very monotonous texture and the quality is not good, which probably does not help. It is unclear why the algorithm fails on Image No. 24, but I took a look at the returned images with high scores and found they do look like the key image a lot. At least it is hard for my eyes to tell which images belong to the same category. In addition, all the returned images for No. 24 have a matching score of over 0.82, which is quite high. Usually it goes down to about 0.7 for the 10th one. Therefore, it may just be “difficult data”.



(a) Image No. 6. The image has an abnormal exposure. (b) Image No. 22. The image has a monotonous texture.

Figure 4.3: These two images have only one valid match each (which is themselves).

In sum, we do see our algorithm returns images that are closely related to the key images. The success is well beyond chance, and is robust to a reasonable extent. The failed cases all seem to be idiosyncratic in one way or another.

4.3.2 Experiment 2: matching images with deformation partially depleted

In another experiment, we want to justify the use of deformation. This time we use the same images as in the previous experiment. However, now we partially deplete the deformation function.

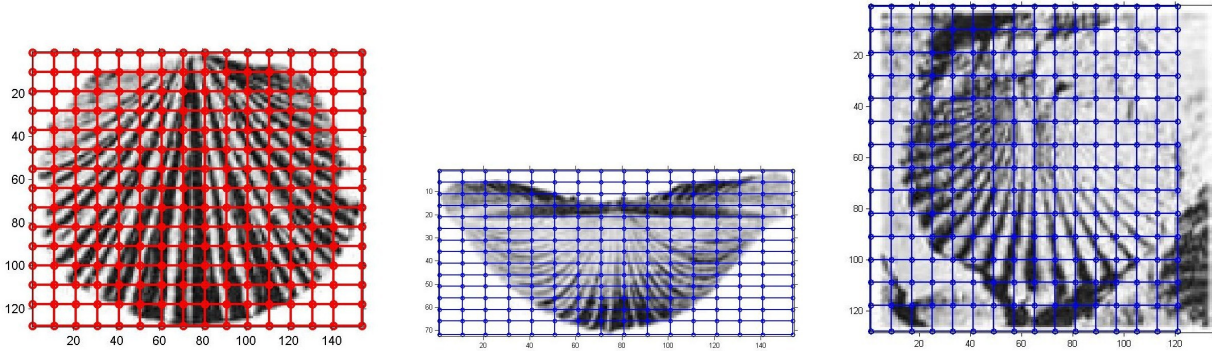


Figure 4.4: Matching with no deformation. The image on the left is the template. Nodes are linearly mapped to target images. The grids on the right edge and bottom edge can be slightly different in size from others.

Specifically, the nodes are still there representing local textures, but they are not allowed to move. The nodes on the template warp linearly map to nodes on the target map based on the shapes and sizes of the images, and the mapping is fixed regardless of texture features. Figure 4.4 is an illustration. Deformation is “partially” depleted because the mapping of nodes still involves some transformation (i.e. deformation) *de facto*, but now the transformation is highly constrained.

In this case there is no doubt that an image will achieve a perfect matching score when it matches against itself. However, our major concern is what the results are in case the template and the target are not identical. We repeated the same procedures as in Experiment 1, but with the manipulations discussed above. The results are shown in Table 4.3. Since the matching scores are model-specific, it is meaningless to compare two sets of scores derived from different algorithms. I just list two critical results: the number of valid matches and the second best ranking of valid matches (if any).

From the table we can see the results are more extreme, compared to the results in Experiment 1. The second best valid matches tend to be ranked either very high (the 2nd position) or very low (near the 10th position). 5 images do not have a second valid match, as opposed to 3 in Experiment 1. This may not seem to be a great change; but if we notice there are 2 cases of 10th position and two cases of 9 position, it is fair to argue the result is not robust. The underlying reason is, with deformation taken away, only exact matches can yield a decent score, with similar patches locating

Item	# V.M.	2nd Best V.M. Ranking
1	3	2
2	4	2
3	3	2
4	2	9
5	2	4
6	1	NA
7	3	9
8	1	NA
9	2	5
10	2	2
11	1	NA
12	2	2
13	3	2
14	2	2
15	6	2
16	4	2
17	3	2
18	5	2
19	2	2
20	7	2
21	4	2
22	1	NA
23	2	10
24	2	10
25	1	NA

Table 4.3: Matching results of 25 key images with deformation partially depleted. V.M. means valid matches.

in exactly the same positions in the respective images. However, with deformation based on EP, we allow patches in different positions to pair. This definitely introduces more flexibility.

It is fair to point out that our test data may not be the best example to show the effect, because in the data set, usually there are photos that do not need much deformation to match. Deformation is more useful when the shapes of the objects are highly unstable.

4.3.3 Experiment 3: Calculating score without deformation cost

This experiment is exactly the same as the first one, but the last step is a little different: we only use data likelihood to compute matching scores, *after* the objective function is optimized. The motivation is to reduce the penalty introduced by shape discrepancies in a quick, easy way. As we know, fossil images are often fragmentary, and sometimes are embedded in rocks. The same fossil

can have different views, too. Fossils of the same category may exhibit unique texture patterns. By focusing on texture features, we hope the results can be more desirable in some circumstances. It should be noted that, the deformation cost has NOT been eliminated from the algorithm. The EP algorithm still uses deformation costs to find the best landing sites for nodes. We only eliminate it in the last step, when the scores are calculated.

As discussed before, our algorithm considers two costs: the deformation cost and the cost based on texture differences. An alternative way to put it is that we have a prior to constrain deformations and a likelihood to compute image similarities. Although we keep the parameters of the prior and the likelihood functions unchanged during the whole process, it may be adequate to tune them *en route*. This experiment can be viewed as doing it in a lazy way: only in the last step and removing the prior completely.

Instead of ignoring the deformation cost in the last step, we could also remove the prior term from the objective function, or set very big prior variances. However, such operations have complicated effects on the results and are hard to control. Not to mention one has to run all the program again before any results can be observed. Removing (or reducing) the deformation cost in the last step serves as an easier trick, and still yields some satisfactory results.

The data and procedures are otherwise exactly the same as in Experiment 1 and 2, and the results are given in Table 4.4. At this time, we see there are only two cases with a single valid match, and all others have more. The second best valid matches also have better rankings than what we observe in Experiment 2. Even though we have three instances of the 9th position, two of them actually have 3 valid matches i.e. there is an extra valid match in the 10th position. So the result is quite robust.

4.3.4 Compare results of experiment 1, 2 and 3

In order to compare the results in an intuitive way, we collected the second best rankings of valid matches, and used the reciprocal of the ranking to measure the level of success. For example, if an image has a valid match ranked in the 4th position, and there is no valid match in the 2nd or 3rd

Item	# V.M.	2nd Best V.M. Ranking
1	3	3
2	3	2
3	3	2
4	3	8
5	3	4
6	2	3
7	2	7
8	4	3
9	3	2
10	2	4
11	3	9
12	3	9
13	2	2
14	2	2
15	7	2
16	2	4
17	3	2
18	2	2
19	2	2
20	5	2
21	3	2
22	1	NA
23	2	7
24	2	9
25	1	NA

Table 4.4: Ignore deformation cost in the last step.

position, we get a score of $1/4 = 0.25$. Summing the scores for all images, we obtain a measure of success. If there is no second best valid match, we count 0. The results are shown in Table 4.5.

Experiment	1	2	3
Measure of Success	8.08	7.87	7.99
Cases without 2nd best V.M.	3	5	2
2nd best V.M. in 2nd position	12	14	11

Table 4.5: Measure of success. It is the sum of reciprocals of the second best V.M. rankings.

With this measure, the original algorithm with a prior term and a likelihood term is still the best. The one with no deformation (Experiment 2), as mentioned fore, often yields extreme results. Dropping prior term in the last step (Experiment 3) is slightly less successful than the original, but it is more stable in a sense, with only two cases having no second best valid match. The relative success among the methods depend on the nature of data too.

4.3.5 Experiment 4: fossil image classification

Ultimately, we want our system to be intelligent, and a useful function is to help us classify fossils. Given a fossil, we often want to know its classification. Unfortunately only a small number of professionals are able to classify brachiopods, and one needs years of training to be adequate. Hopefully the program we designed can be helpful.

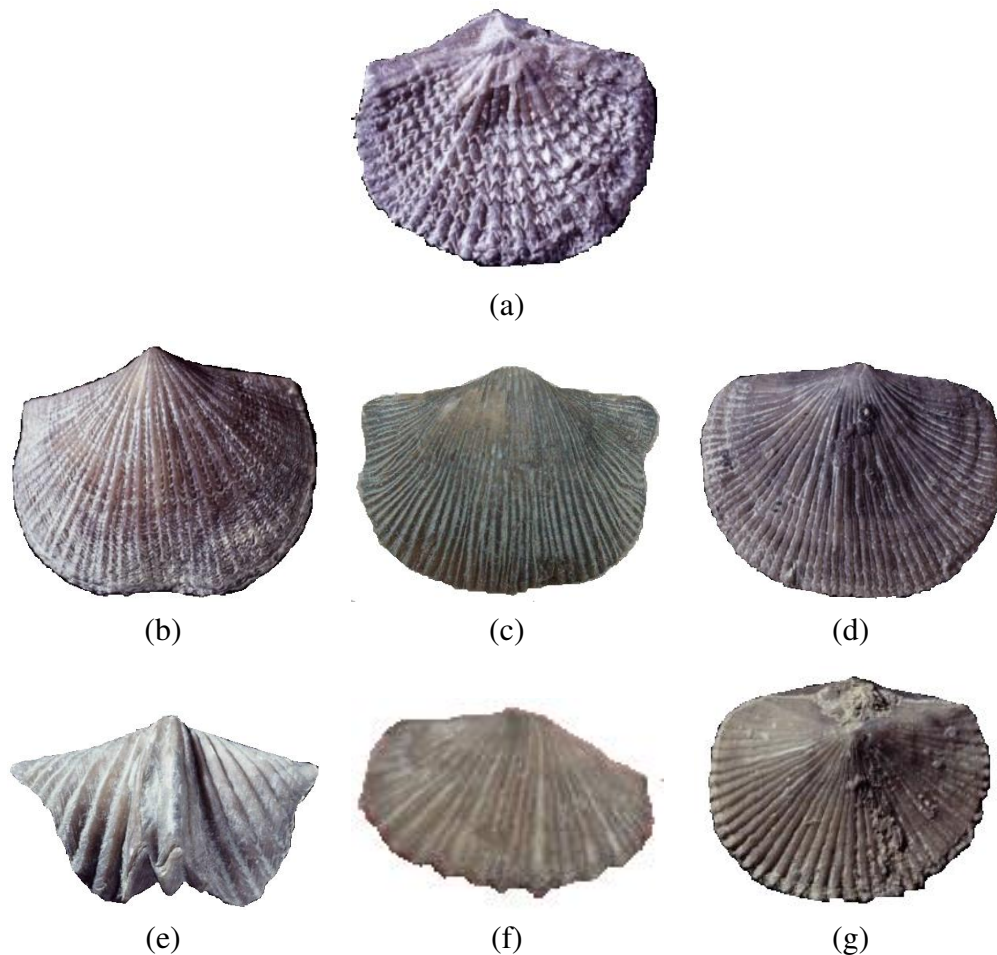


Figure 4.5: Test images borrowed from Dry Dredgers. All images are from the suborder Orthidina. (a). *Eridorthis*; (b). *Glyptorthis*; (c). *Hebertella*; (d). *Plaesiomys*; (e) *Platystrophia*; (f). *Plectorthis*; (g) *Retrorsirostra*.

We definitely want to test if our algorithm and data set can be utilized to identify fossil images from sources other than the Treatise. Thanks to Dry Dredgers <http://drydredgers.org/>, an association of amateur geologists and fossil collectors, for their kindly allowing us to use their photos. We used 7 fossil photos from there, and none of them can be found in our data set. These photos

are taken from different specimen, all from the order ORTHIDA and suborder ORTHIDINA. However, they are distributed in two different superfamilies, ORTHOIDEA and PLECTORTHOIDEA respectively. And they involve four different families as shown in Table 4.6.

Genus	Family	Superfamily
Eridorthis	GLYPTORTHIDAE	ORTHOIDEA
Glyptorthis	GLYPTORTHIDAE	ORTHOIDEA
Hebertella	PLECTORTHIDAE	PLECTORTHOIDEA
Plaesiomys	PLAESIOMYIDAE	ORTHOIDEA
Platystrophia	PLATYSTROPHIIDAE	PLECTORTHOIDEA
Plectorthis	PLECTORTHIDAE	PLECTORTHOIDEA
Retrorsirostra	PLAESIOMYIDAE	ORTHOIDEA

Table 4.6: Classification of the fossils used as test images.

We matched the 7 images with all images we have under the suborder ORTHIDINA. Totally there are 103 of them. Then we perform a score-weighted 5-nearest neighbor classification. Specifically, for each image we consider the 5 best matches, count how many matches belong to each category, and sum the matching scores of each category. The sum of scores under Category C_i divided by the sum of all 5 scores is the “probability” that the image belongs to C_i :

$$P(I \in C_i) = \frac{\sum score(I \in C_i)}{\sum score(I \in \cup C_i)} \quad (4.2)$$

Then the category with the highest score will be the winner: $C_I = \operatorname{argmax}_{C_i} P$.

Our result shows that classification on the superfamily level is quite accurate, but the same cannot be said for family level. Table 4.7 shows the results of superfamily classification.

	Erid.	Glyp.	Hebe.	Plae.	Plat.	Plec.	Retr.
True S.Family	ORTH.	ORTH.	PLEC.	ORTH.	PLEC.	PLEC.	ORTH.
P(ORTH.)	0.803	0.798	0.800	1.00	0.193	0.200	0.602
P(PLEC.)	0.197	0.202	0.200	0.00	0.807	0.800	0.398

Table 4.7: Classification results of the images, using weighted 5-nearest neighbor method. Only one case fails.

4.4 Summary and conclusions

Our Experiment 1 shows that the algorithm is largely successful, with valid matches returned. Experiment 2 highly constrained deformation, and yields very unstable results. Experiment 3 manipulates the deformation cost at the end, and can improve the results in some cases. Experiment 4 shows that the algorithm can deal with innovative data as well, at least it is the case for the samples we have.

Our objective function has two parameters, which control the deformation cost and the texture difference cost, respectively. In order to find the best match, there is actually only one degree of freedom for the parameters, but keeping them separate is a better strategy when tuning the program, because each of them has clear mathematical semantics. In addition, we also use another parameter to scale down the overall cost, so the “probability scores” are in a range easier for humans to interpret. This parameter does not affect results.

Chapter 5

Conclusion and Discussions

In this thesis I have showed some progress in our project of building the Invertebrate Paleontology Knowledgebase (IPKB). In the first step, we extracted and organized data from the *Treatise on Invertebrate Paleontology*, in both text and images forms. So far we have processed the volumes of brachiopods, and others can be processed in the same way. The search engine can perform text-based search in a variety of ways. The web interface is easy to use and browse. We tried to deliver a pleasant tool for a discipline which is considered “dry” by many, so hopefully more people can access and interpret data.

A goal of the knowledgebase is to build the capability of interpreting images independently, not necessarily relying on text. In this thesis I explained a program which matches images and provide a probability score of identity. The algorithm is a deformable template model and we use texture features to measure distances. The objective function contains deformation cost and data difference cost. The algorithm seeks to optimize the objective function using expectation propagation. The preliminary results show that the algorithm is decent. There are ways to tune the parameters, depending on the nature of data and the goal of users.

This IPKB project is still in progress and this thesis is by no means a conclusion of it. An immediate task in near future, probably, is to incorporate the image processing module in the search engine.

References

- [Alon & Yuster, 2010] Alon, N. & Yuster, R. (2010). Solving linear systems through nested dissection. *IEEE 51st Annual Symposium on Foundations of Computer Science*, (pp. 225–234).
- [Anil K. Jain & Lakshmanan, 1996] Anil K. Jain, Y. Z. & Lakshmanan, S. (1996). Object matching using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3), 267–278.
- [Choi & Rasmussen, 2006] Choi, Y. & Rasmussen, E. (2006). What do digital librarians do. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries, JCDL '06* (pp. 187–188).
- [Fox et al., 1995] Fox, E. A., Akscyn, R. M., Furuta, R. K., & Leggett, J. J. (1995). Digital libraries. *Commun. ACM*, 38(4), 22–28.
- [Gonzalez et al., 2009] Gonzalez, R., Woods, R. E., & Eddins, S. L. (2009). *Digital Image Processing Using MATLAB*. Gatesmark Publishing.
- [Levy & Marshall, 1995] Levy, D. M. & Marshall, C. C. (1995). Going digital: a look at assumptions underlying digital libraries. *Commun. ACM*, 38, 77–84.
- [M. Kass & Terzopoulos, 1988] M. Kass, A. W. & Terzopoulos, D. (1988). Snakes: Active contour models. *Int. J. Comput. Vision*, 1(4), 321–331.
- [Meyer, 1994] Meyer, F. (1994). Topographic distance and watershed lines. *Signal Processing*, (pp. 113–125).

- [Minka, 2001a] Minka, T. P. (2001a). Expectation propagation for approximate bayesian inference. *UAI 2001*, (pp. 362–369).
- [Minka, 2001b] Minka, T. P. (2001b). *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology.
- [Moore & other editors, 2006] Moore, R. C. & other editors (1953-2006). *Treatise on Invertebrate Paleontology: Part H*. <http://www.ku.edu/paleo/treatise.html>: Geological Society of America and the University of Kansas Press.
- [Parker, 1997] Parker, J. R. (1997). *Algorithms for Image Processing and Computer Vision*. New York: John Wiley & Sons, Inc.
- [Potetz & Hajjarbabi, 2013] Potetz, B. & Hajjarbabi, M. (2013). Whitenened expectation propagation: Non-lambertian shape from shading and shadow. *post presentation at CVPR 2013*.
- [Serra, 1982] Serra, J. (1982). *Analysis and Mathematical Morphology*. London: Academic Press.
- [Shneiderman et al., 2002] Shneiderman, S., Turin, M., & the Digital Himalaya Project Team (2002). Digital himalaya: an ethnographic archive in the digital age. *European Bulletin of Himalayan Research*, 20(1), 136–141.
- [Singhal et al., 1996] Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 21–29): ACM.
- [Sirovich & Kirby, 1987] Sirovich, L. & Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America, A*, 4(3), 519–524.
- [Slot & Gozdzik, 2008] Slot, K. & Gozdzik, M. (2008). Fingerprint alignment with deformable templates. *19th International Conference on Systems Engineering*, (pp. 395–398).
- [Turk & Pentland, 1991] Turk, M. & Pentland, A. (1991). Face recognition using eigenfaces. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 586–591).

[Vincent, 1993] Vincent, L. (1993). Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2), 176–201.

Appendix A

The 25 key images used in experiments

