博士論文

Retrieval and Drawing Assistance for Manga

(漫画の検索と描画支援)

48-137407

松井　勇佑

指導教員　相澤　清晴

大学院　情報理工学系研究科　電子情報学専攻

東京大学

This dissertation is submitted for the degree of

Doctor of Philosophy

December 2015

Acknowledgements

First and foremost, my deepest gratitude is to my advisor, Professor Kiyoharu Aizawa, for his continuous support of my PhD study. During the six years I have spent in Aizawa-Yamasaki lab, he always helped and guided me to the right direction.

I would like to express my great appreciation to my co-advisor, Professor Toshihiko Yamasaki. I am deeply grateful to him for the long discussions, not only for research, but also all the things for my life.

I am grateful to my mentor at MSRA, Dr. Takaaki Shiratori. Through the research with him, I have learned how not to give up until the very end.

I would like to express my gratitude to my menter at MSR, Dr. Yingpeng Chen and Dr. Zengyou Zhang. I have learned from them how to achieve the goal of a project with a team.

I would also like to thank all members of the Aizawa-Yamasaki Laboratory.

Finally, I would like to dedicate this thesis to my wife, Dr. Asako Kanezaki.

Abstract

Manga are Japanese style comics, i.e., visual media by which ideas are expressed via images and texts. In Japan, the manga industry serves an extremely large market, and *electronic manga* (*e-manga*) are also becoming popular. E-manga are distributed not in print but electronically via online stores, and the market for e-manga is rapidly growing. For example, Amazon Kindle Store, which is one of the largest e-manga platforms, sells more than 130,000 e-manga titles.

Let us move on to the general topic of computer visuals, e.g. target images that are not manga but naturalistic images, a massive quantity of data (photos, texts, sensed data, activity logs, etc.) is uploaded to the Web every day. Because of the advent of such large-scale data and of efficient algorithms, incredible applications are becoming a reality such as image recognition and image captioning. The advent of big data is not limited to naturalistic images. The number of e-manga titles is also rapidly increasing. In the face of the massive quantities concerned, can we create amazing applications like those for naturalistic images? And if this is possible, what kind of new values can be generated? This is a central topic of this thesis.

To answer the question, we propose **a fundamental component**, **a theoretical improvement**, and **a data-driven application**.

- **Fundamental component: sketch-based retrieval architecture**: To handle large scale data, the most basic operation is searching. In this thesis, we build a sketch-based manga retrieval framework. The framework is fast and memory efficient; it takes only 70 ms to search 21,142 pages, with a 204 MB memory footprint. From this result, we concluded that retrieval, the fundamental process for large-scale manga data, was achieved.

- **Theoretical improvement: efficient search using hash tables**: Next, we develop a fast nearest-neighbor search method using hash tables. To handle more data, e.g., one million pages, further analysis and improvement of a core part of the system, the nearest-neighbor search, is required. To cope with million- or billion-scale data, we develop a product quantization-based hash table: the

product quantization table (PQTable). The PQTable produces exactly the same results as a linear PQ search, and is 10^2 to 10^5 times faster.

- **Data-driven application: drawing assistance**: Finally, we create an application using large-scale image data. Essentially, drawing is a difficult task for novices. We present an interactive drawing system with visual feedback. The user studies indicated that the proposed system help novices draw their own sketches.

According to the proposed methods, this thesis will contribute to the fundamental technology for manga image processing, the theoretical improvement with regard to searches, and will create its application using a large-scale image database. We expect that this thesis will provide a promising future direction for research into manga image processing.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Manga

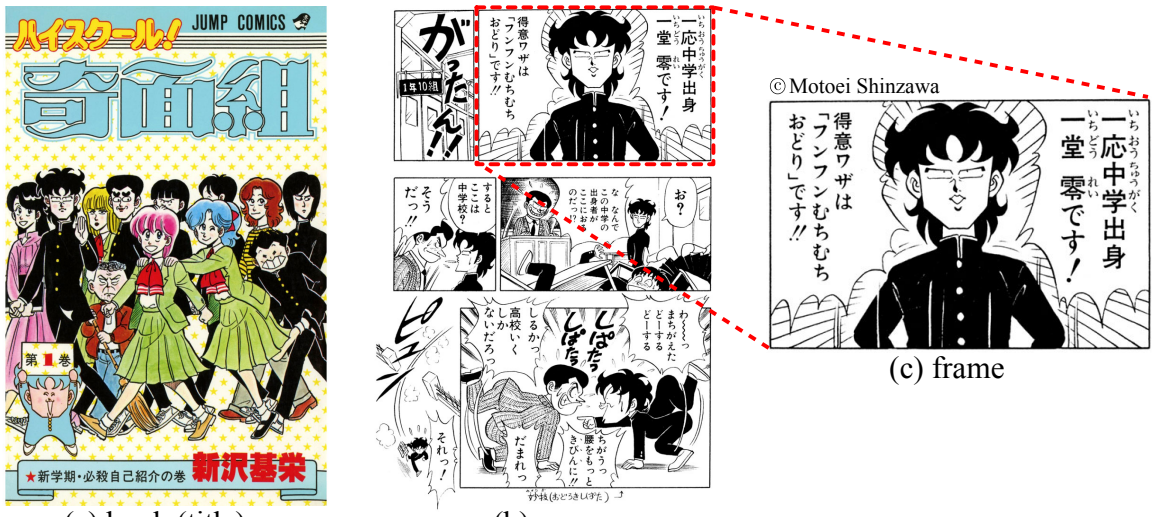
Manga are Japanese style comics, i.e., visual media by which ideas are expressed via images and texts. Manga were originally born and raised in Japan, and are currently popular in many parts of the world. The visual characteristic of manga is their unique form of expression. Unlike usual paintings in which visual elements are represented by color, manga images are usually line drawings made up of black lines on a flat white ground. To express textures and shades, special preprinted patterns (screen tones) are commonly used. Figure 1.1 shows an example of a manga title. In terms of narrative content, manga cover a broad range of categories of stories, e.g., sports, love romance, science fiction, fantasy, etc.

In Japan, the manga industry serves an extremely large market. Statistics show that manga sales in 2014 reached the amount of 370 billion JPY[1]. Furthermore, manga attract considerable attention not only in Japan but also in many other countries. The Japanese Ministry of Economy, Trade and Industry attaches much importance to manga in its "Cool Japan / Creative Industries Policy"[2], which aims to dramatically expand the sales of attractive Japanese goods and services worldwide.

Nowadays, *electronic manga* (*e-manga*) are becoming popular. E-manga are distributed not in print but electronically via online stores, then read on electronic devices such as smartphones, tablets, PCs, etc. The market for e-manga is rapidly

[1] 出版科学研究所，出版月報，2014年2月号

[2] Cool Japan / Creative Industries Policy. Retrieved on May 19, 2015, from http://www.meti.go.jp/english/policy/mono_info_service/creative_industries/creative_industries.html



(a) book (title) (b) page (c) frame

Figure 1.1: Manga examples. (a) A manga book (title) consists on average of about 200 pages. (b) A page (an image) is composed of several frames (rectangular areas). (c) Characters, text balloons, and background are drawn in each frame. Typically, manga are drawn and printed in black and white.

growing. For example, Amazon Kindle Store, which is one of the largest e-manga platforms, sells more than 130,000 e-manga titles[3].

Because e-manga are a form of digital images, image processing can be applied. For example, if a Gaussian filter is applied to a page of e-manga, it will be blurred in the same manner as for naturalistic images. Therefore, it is possible to apply various techniques developed in computer vision and image processing fields to e-manga. However, no such attempts have been made. The only ways for readers to enjoy e-manga are currently the usual ones of purchasing and reading them.

1.2 The era of big data

Let us move on to the general topic of computer visuals, e.g. target images that are not manga but naturalistic images, usually taken by a camera. As of the 2010s, it has been said that the era of big data has arrived. A massive quantity of data (photos, texts, sensed data, activity logs, etc.) is uploaded to the Web every day. For example, Facebook reports that more than 350 million photos are uploaded every day [110]. At the same time, powerful deep learning algorithms are currently being

[3]Amazon.co.jp Kindle Comic. Retrieved on September 27, 2015, from http://amzn.to/1KD5ZBK

intensively developed. With such algorithms, big data can be handled, and useful predictive systems can be learned. Because of the advent of such large-scale data and of efficient algorithms, incredible applications are becoming a reality. One example is that of image recognition. A number of authors claim that their methods achieve higher scores than human beings in image recognition tasks such as ILSVRC [42]. Another example is caption generation. Given an image, algorithms can predict a caption that explains the content of the image [35].

The advent of big data is not limited to naturalistic images. As described in Section 1.1, the number of e-manga titles is rapidly increasing. In other words, big *manga* data is becoming available. In the face of the massive quantities concerned, can we create amazing applications like those for naturalistic images? And if this is possible, what kind of new values can be generated? This is a central topic of this thesis.

1.3 Research Challenges

The technical challenges related to processing e-manga stem from the difference in nature between naturalistic images and manga. Manga images are created not by capturing a real scene using a camera, but by manually drawing lines on white paper. Therefore, manga images are constructed according their own rules (e.g., parts of faces such as eyes are abstracted to manga-style icons), and do not follow the rules of the 3-dimensional (3D) real world. Therefore, the question of whether or not effective methods for naturalistic images are also useful for manga has not yet been answered.

In particular, from the viewpoint of computer visuals, the greatest differences between naturalistic images and manga images can be summarized as follows: (1) Gradients of intensity. In many cases, manga are drawn using black curved lines on a white background. Hence, manga contain neither color information nor consecutive gradients of intensity. Because of this property, we cannot model and leverage a statistical profile of gradients of intensity, which is the standard way for image segmentation (e.g., grab cut [92]). (2) 3D information. Objects drawn in manga do not need to follow the geometric relationship of the real 3D world. In many cases, objects in manga completely break the restrictions posed by 3D rules. For example, if a rigid arm of a robot is drawn on one page, another page might show that the arm is extremely deformed and exaggerated. Therefore, it is difficult to identify that these two arms are the same only by using 3D image processing techniques.

This simple example implies that image processing techniques using 3D information cannot be directly applied to manga objects.

1.4 Contributions

In the face of large-scale manga data, how can we handle this and create new values? This is a central topic of this thesis. To answer the question, we propose **a fundamental component**, **a theoretical improvement**, and **a data-driven application**.

- **Fundamental component: sketch-based retrieval architecture**: To handle large-scale data, the most basic operation is searching. In our manga case, given an instance (character, object, etc.) of the manga, the most fundamental action is to find a similar instance from a manga database. Without searching for a similar manga instance, no method that handles manga images can be conducted effectively. This task is so-called image-based image retrieval, and to our knowledge, existing applications of this kind have not yet been applied to manga. In this thesis, we build a sketch-based manga retrieval framework. The proposed framework consists of (1) EOF feature description, (2) feature compression and search using product quantization, and (3) effective preprocessing including skipping margins. The framework is fast and memory efficient; it takes only 70 ms to search 21,142 pages, with a 204 MB memory footprint. From this result, we concluded that retrieval, the fundamental process for large-scale manga data, was achieved.

- **Theoretical improvement: efficient search using hash tables**: Next, we develop a fast nearest-neighbor search method using hash tables. By the proposed sketch-based retrieval framework, we achieved retrieval for manga images. To handle more data, e.g., one million pages, further analysis and improvement of a core part of the system, the nearest-neighbor search, is required. To cope with million- or billion-scale data, we develop a product quantization-based hash table: the product quantization table (PQTable). The PQTable produces exactly the same results as a linear PQ search, and is 10^2 to 10^5 times faster when tested on 10^9 SIFT data points (SIFT1B data). In addition, although state-of-the-art performance can be achieved by the previous inverted-indexing-based approaches, such methods require manually designed parameter setting and

much training, whereas our method is free from these. Therefore, the PQTable offers a practical and useful solution for real-world problems.

- **Data-driven application: drawing assistance**: Finally, we create an application using large-scale image data, i.e., a data-driven application for manga. Essentially, drawing is a difficult task for novices. We present an interactive drawing system with visual feedback. The proposed system enables users to retrieve and refer to a sketch image stored in a database, and provides them with various new strokes as suggestive or deformation feedback. User studies indicated that the proposed system helped novices draw their own sketches. This is one of the applications that can be achieved in a data-driven way using a large-scale image database.

According to the proposed methods, this thesis will contribute to the fundamental technology for manga image processing, the theoretical improvement with regard to searches, and will create its application using a large-scale image database. We expect that this thesis will provide a promising future direction for research into manga image processing.

Chapter 2

Related work

[56]

Chapter 3

Fundamental Component: Sketch-based Retrieval Architecture

3.1 Introduction

Retrieval is one of the fundamental operations for handling a large scale data. Without an efficient retrieval, data driven applications cannot be performed. However, current e-manga archives offer very limited search support, i.e., keyword-based search by title or author, or tag-based categorization. They are not suitable for large-scale search and searches cannot take the images (contents) of manga into consideration. For this reason, applying content-based multimedia retrieval techniques to manga search would have the potential to make the manga-search experience more intuitive, efficient, and enjoyable.

There are three unique challenges associated with content-based manga retrieval. (1) **The manga image description problem**. First, the visual characteristics of manga images are very different from those of naturalistic images. For example, as shown in Figure 1.1, manga images are usually line drawings comprising black lines on a flat white background. Therefore, manga images often do not have varying gradient intensities, unlike naturalistic images. Traditional local-feature characterizations, such as scale-invariant feature transform (SIFT) [76], may not be suited to describing manga images. (2) **Retrieval–localization problem**. Furthermore, a manga page comprises several frames (rectangular areas). It is therefore necessary to retrieve not only an image, but also a part of the image. This is a combined problem of retrieval and localization, and solutions must allow fast processing to achieve large-scale retrieval. (3) **Query modality problem**. The final challenge is to find a way of

efficiently querying manga images. Suppose that a user wants to retrieve the face of the boy in Figure 1.1(c). Usually, we do not have annotations that describe the specific region, so we must somehow find a way of querying this region. Because manga images are mostly black-and-white drawings, users should have access to some form of drawing, such as examples of manga images or sketches of their own. For this reason, it is important to provide a natural sketch-based interface for manga retrieval.

In this chapter, we propose a content-based manga retrieval system that addresses the above three challenges.

- First, we propose a manga-specific image-describing framework, namely an objectness-based edge orientation histogram (EOH) with product quantization. The system is composed of three steps: labeling of margin areas, EOH feature description [71], and approximate nearest-neighbor (ANN) search using product quantization (PQ) [56]. A comparative study confirmed that the proposed method can efficiently describe manga images, and can yield better accuracy than the state-of-the-art sketch-based retrieval methods [106, 97]. In addition, we evaluated the proposed method in the instance detection framework, and verified that the proposed method can achieve both retrieval and localization from 21,142 manga pages in less than a second with reasonable accuracy.

- Second, we study the feasibility of a sketch-based interface as a more natural way for people to interact with manga content. Sketch-based interfaces for retrieving multimedia contents have been explored previously [106, 33, 20], and we conjecture that they are particularly suited to manga. In addition, based on the sketch interaction, we built two interactive reranking schemes for an intuitive manga search experience: **relevance feedback** and **query retouch**.

An overview of the GUI implementation of the proposed system is shown in Figure 3.1. Given a query sketch (Figure 3.1(a)), the system retrieves similar areas from a manga dataset in real time, and shows the top results in thumbnail windows (Figure 3.1(c)). The retrieval is performed automatically after each stroke is completed. If a user clicks on one of the retrieved results in the thumbnail windows, a page containing the result is shown in a preview window (Figure 3.1(d)). In this case, the page containing the first-ranked result is shown.

For evaluation, we built a novel dataset of manga images drawn by professional manga artists, Manga109, which consists of 109 comic books with a total of 21,142 pages. Manga109 will be very beneficial for the multimedia research community

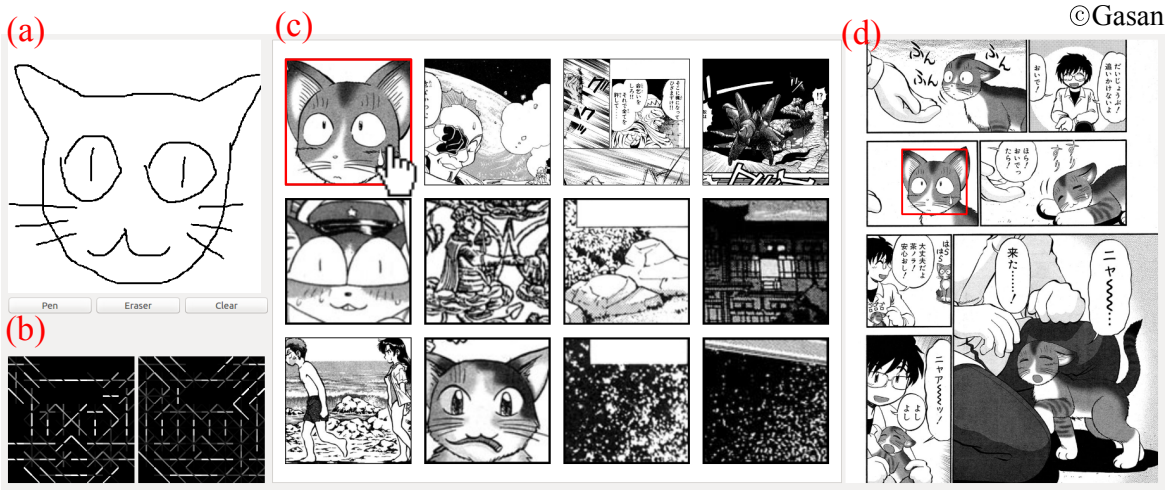


Figure 3.1: An interface for the retrieval system. (a) A canvas for the user's sketch. (b) Visualized EOH features, where the left figure shows a query and the right shows the retrieved result. (c) Thumbnail windows for the retrieved results. (d) A preview window of a manga page.

because it has been difficult for researchers to use "real" manga in experiments due to the copyright problem. Research so far has been limited to the small scale. We made this dataset with permission from 94 professional creators. The dataset covers a wide range of manga genres, and is publicly available for academic use.

Our contributions are summarized as follows:

- We propose a sketch-based manga retrieval system, which enables us to retrieve a manga image in 70 ms from 21,142 pages using a notebook PC. The method consists of a manga-specific image description system, and a sketch-based interactive query scheme, including relevance feedback and query retouch.

- We provide a publicly available manga dataset, **Manga109**, which includes 109 manga titles containing 21,142 pages that will be useful for manga image-processing research.

For applications, the proposed method can be integrated in a tablet or a PC to retrieve purchased manga titles stored in each user's computers, or implemented as a Web-based interface to search manga titles at e-manga services such as Amazon. The system is useful in several scenarios. (1) A user wants to find a manga title that contains an object that is hard to represent by words, but easy to represent with a sketch query, such as "Japanese-style clothing" (Figure 3.7a). (2) A user does not have a specific title in mind, but tries to find interesting manga, such as one in which

cute girls appear. In these cases, our proposed query interactions can be useful (Figure 3.15a). (3) After the initial search is performed using text queries, a user can rerank the retrieved results by sketch interactions, e.g., a user selects "Dragon Ball" titles by typing "Dragon Ball", and then refines results by drawing some characters.

3.2 Related Work

We discuss related studies in content-based manga retrieval by categorizing them into three challenges: the manga description problem, the retrieval–localization problem, and the querying problem. Additionally, we introduce research related to manga.

3.2.1 Manga image description

In the literature of content-based image retrieval, many methods have been proposed to describe images. A query and dataset images are converted to some form of vector representation to evaluate their similarity. A typical form is the bag-of-features (BoF) representation [100], in which local features such as SIFT [76] are extracted from an image and quantized into a sparse histogram. The distance between a query histogram and a dataset histogram can be computed efficiently using an inverted index structure. The BoF form has been studied widely because it achieved successful image description with fast computation.

Various extensions of BoF have been proposed so far, including query expansion [27], soft voting [86], and Hamming embedding [54]. The state-of-the-art in this line are the vector of locally aggregated descriptors (VLAD) and Fisher vector (FV) approaches [57, 102, 31], which are interpreted as generalized versions of BoF [94]. However, all of the above methods were designed for natural images for both query and dataset. Thus, special attention has been paid to cases in which a query is a line drawing, i.e., a sketch.

If a query is presented as a form of sketch, several special features tailored for sketch queries have been proposed. BoF using a histogram of oriented gradients [28] was proposed [97, 33, 130]. Because sketch images usually have a flat background of uniform white pixels, several methods tried to fill such blank areas with meaningful values and interpreted them as a feature vector; e.g., distance-transformed values from edge pixels [112] and Poisson image-editing-based interpolation values [46, 47]. To reduce the impact of noisy edges, a line segment-based descriptor was

proposed [117]. Chamfer matching, which is a successful method for computing the similarity between two contours of objects, was incorporated in the sketch-based retrieval framework [106, 20, 89].

In the manga feature description task, we compared the proposed method with (1) BoF, (2) large-window FV as the state-of-the-art of BoF-based methods [97], and (3) compact oriented chamfer matching (Compact OCM) as the state-of-the-art of chamfer-based methods [106].

Note that deep-learning techniques are becoming quite popular in image recognition because of their superior performance. These technologies are being applied to the sketch processing. However, the effectiveness of such methods has not yet been discussed sufficiently. For example, Yu and colleagues [124] reported that traditional methods such as FV are still competitive for a sketch recognition task, and therefore such deep learning-based features are beyond the scope of this thesis.

3.2.2 Retrieval and localization

As shown in Figure 1.1, our objective is to find not only a similar page, but also identify a similar region in the retrieved image. This kind of task has been tackled in spatial verification-based reranking methods [69, 85, 121, 55, 19, 129, 128]. These methods first find candidate images by BoF, and rerank the top results by spatial information such as relative positions among extracted SIFT features. However, such methods cannot be applied to manga for two reasons. First, the manga image's structure is more complex than that of a natural image. A single manga page usually consists of several frames, each of which is a totally different visual scene (see Figure 1.1(b)). Therefore, a manga page can be interpreted as a set of images if we treat each frame as a single image. This is a much harder condition than those usual in image processing tasks such as retrieval or recognition, which assume that at least an image is a single image rather than a set of images. Second, BoF does not work well for manga as shown in the experimental section.

Another related area is object-detection methods, in which the position of a target object is identified in an input image. A number of methods have been proposed so far. Among them, contour-based object localization methods [99, 118] are closely related to our manga problem because an object is represented by its contour. In such methods, an instance of a target object such as "giraffe" is localized from an image by matching a query contour of giraffe, which is learned from training data. If we apply such methods to all dataset pages and select the best result, we might achieve both retrieval and localization at once. However, this is not realistic because:

(1) these methods usually require learning to achieve reasonable performance, i.e., many giraffe images are required to localize an instance of giraffe, but a query is usually a single sketch in our case; and (2) localization usually takes time and cannot be scaled to a large number of images.

The proposed method makes use of a simple approach: window-based matching. It is also slow, but we formulate the search and localization as a single nearest-neighbor problem, and solve it using ANN methods.

3.2.3 Querying

How to query multimedia data has been an important problem in content-based multimedia retrieval [59]. Although many approaches have been proposed, including keywords, images, and spatial/image/group predicates [101], we focus on sketches drawn by users. Sketch-based interaction is the most natural way for humans to describe visual objects, and is widely adapted not only for image retrieval, but also in many applications such as shape modeling [84], image montage [22], texture design [63], and as a querying tool for recognizing objects [97, 32].

3.2.4 Manga

From an image-processing perspective, manga has a distinctive visual nature compared with natural images. Several applications for manga images have been proposed: colorization [88, 108, 95], vectorization [66], layout recognition [109], layout generation [45, 17], element composition [18], manga-like rendering [87], speech balloon detection [90], segmentation [3], face tailored features [25], screen tone separation [52], retargeting [80], and manga-like summarization for video [44, 26, 60]. The research group at Université de La Rochelle constructed a comic database [40], which we mention in Section 3.5, and analyzed the structure of visual elements in a page to understand the comics content semantically [91].

Several studies have explored manga retrieval [105, 29, 104]. In these methods, inputs are usually not pages but cropped frames or characters, and evaluations were conducted using a small number of examples. In addition, the runtimes of these methods have not been discussed fully. Therefore, it is not known whether these methods can scale to a large dataset. On the other hand, the proposed method can perform a retrieval from 21,142 pages in 70 ms.

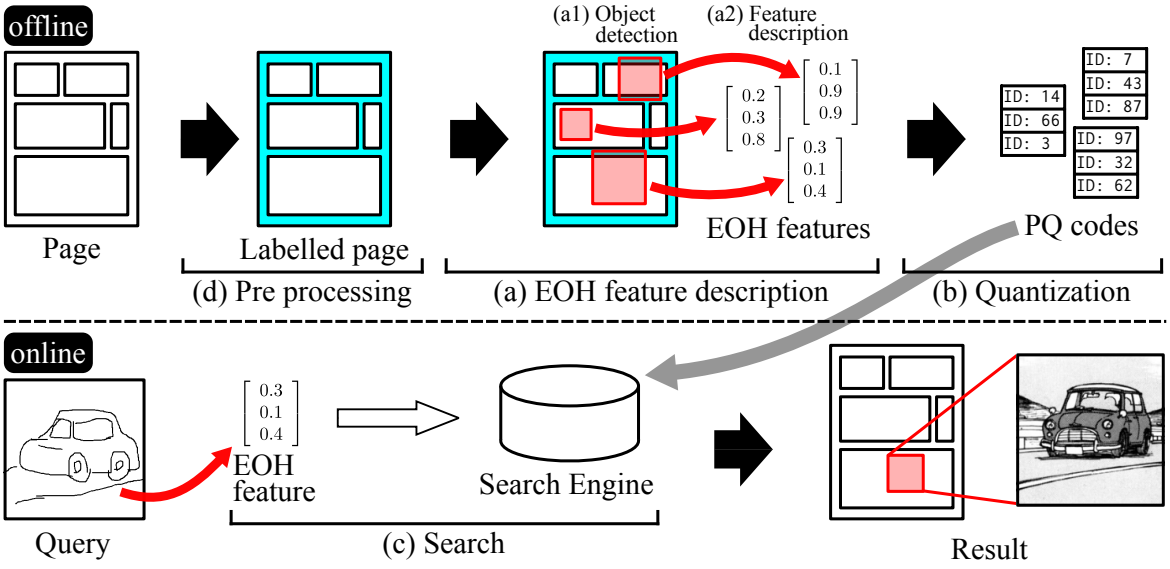


Figure 3.2: Framework of the proposed system.

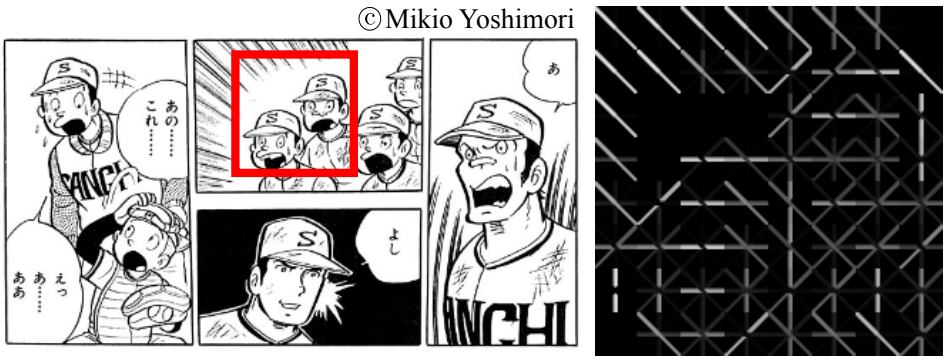
3.3 Manga Retrieval System

In this section, we present the proposed manga retrieval system. Figure 3.2 shows the framework of the system. First, input pages are preprocessed offline (Figure 3.2(d). This is discussed in Section 3.3.4). Next, region of interest (ROI) areas are detected and EOH features are extracted (Figure 3.2(a), as described in Section 3.3.1). These features are then compressed into PQ codes (Figure 3.2(b), as discussed in Section 3.3.2). At query time, given a query sketch, similar EOF regions are retrieved (Figure 3.2(c), as discussed in Section 3.3.3). The system is designed to solve both the manga image description and retrieval–localization problems.

The proposed framework is based on the window search paradigm; i.e., several features are extracted from dataset images using a generic object detector, and a query feature is compared with all features. Compared with the traditional image retrieval problem, such a window system is usually too slow to handle a large number of features. We formulate the problem as a single ANN search, and solve it using PQ, with effective preprocessing including the labeling of margin areas. Note that we delete screen tone (texture) areas in advance during the preprocessing [52].

3.3.1 EOH feature description with object windows

We first show a core part of the system, feature description (Figure 3.2(a)). We employ EOH features [71], which compute an EOH for a square target ROI. Given



(a) (b)

Figure 3.3: An example of an EOH feature. (a) An image and a selected area. (b) The visualized EOH features extracted from the area in (a). As can be seen, the visual characteristics of a manga page can be captured using such a simple histogram.

a page, candidate areas of objects are detected automatically (Figure 3.2(a1)), then EOH features are described from selected areas (Figure 3.2(a2)). The retrieval is performed by matching an EOH feature of the query against the EOH features in the dataset. Figure 3.3 shows an example of the feature representation. The EOF feature of the red square area in Figure 3.3(a) is visualized as shown in Figure 3.3(b).

The candidate areas of objects are detected by generic object detectors [2, 113, 24], which are successful ways to replace a sliding window paradigm. (Figure 3.2(a1)). Intuitively, these detectors compute bounding boxes, which are more likely to contain any kinds of objects. These bounding boxes are considered as candidates for object instances, and a processing task such as image recognition is applied only to the bounding box areas, which greatly reduces computational costs compared with a simple sliding window approach.

We found that objectness measures can detect candidate manga image objectives more effectively than the sliding window approach. We constructed an experiment using a small dataset in which ground-truth objects (a head of a man) are annotated. The detection rate of a few methods were evaluated as shown in Figure 3.4. BING [24], selective search [113], and sliding window (baseline) [79] are compared. From the results, we concluded that selective search is the best approach for this dataset. In the rest of this section, we use selective search for our object detector. Note that a horizontally or vertically long bounding box is divided into several square boxes with overlaps because we restrict the shape of object areas to a square.

After object areas have been detected (typically, 600 to 1000 areas are selected from a page), their EOF features are described (Figure 3.2(a2)). The selected square area is

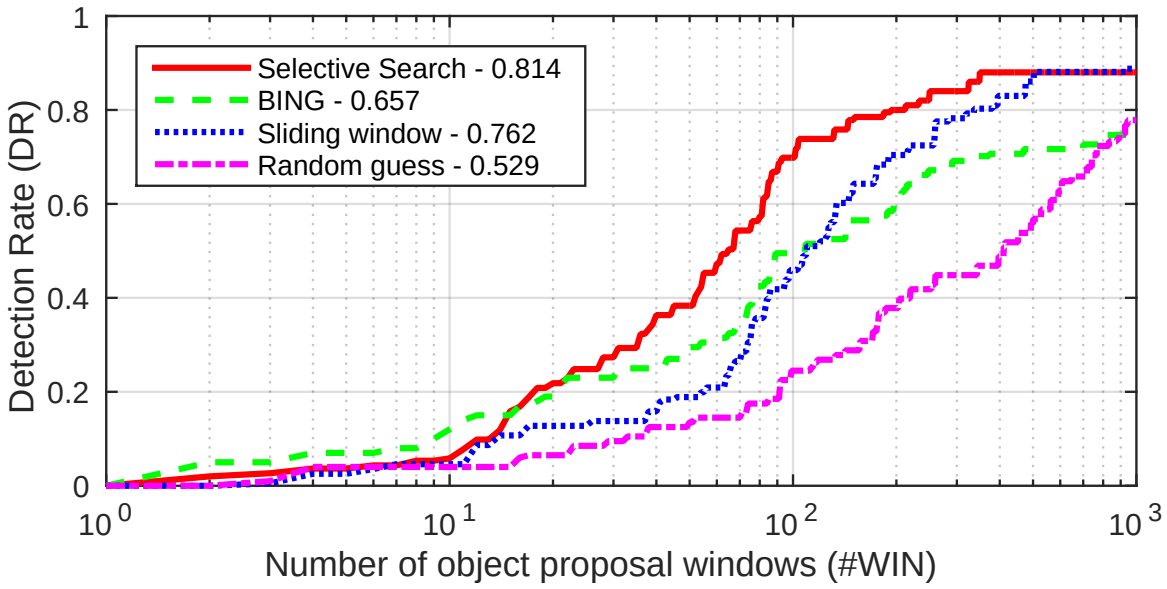


Figure 3.4: Detection rate (DR) given #WIN proposals, which is a standard evaluation metric for evaluating objectness [24, 2, 113]. DR shows the percentage of ground-truth objects covered by proposals, which are generated by each method. An object is considered covered only when the PASCAL overlap criteria is satisfied (intersection over union is more than 0.5 [34]). The area under the curve (AUC), which is a single-value summary of performance for each curve, is shown on the legends. We evaluated methods using the dataset used for the localization task, see details on Section 3.6.2.

divided into $c \times c$ cells. The edges in each cell are quantized into four orientation bins and normalized, and the whole vector is then renormalized. Therefore, the dimension of the EOH vector is $4c^2$. Note that we discard features in which all elements are zero. For manga, the features should be robust against scale changes because we want to find areas of any size that are similar to the input sketch. For example, if a certain character's face is the target, it can be either small or large. To achieve matching across different scales, we simply accept any sizes of patches, but restrict the shape of an area to square. Whatever the patch sizes, the same-dimensional EOH features are extracted and used for matching.

By describing EOF features of candidate areas, a page is represented by a set of EOH features:

$$\mathcal{P} = \{\mathbf{x}_i | i \in \{1, \ldots, N\}\} \tag{3.1}$$

where $\mathcal{P}$ means the page, $\mathbf{x}_i$ denotes an EOH feature for the ith window, and N is the number of features extracted from the page. Note that integral images are utilized to enable the fast computing of features in the same manner as in [71]. In the comparative study, we show this simple representation achieves better accuracy than previous sketch-based retrieval methods, and we confirm that EOH-based description gives a good solution to the manga image representation problem.

In summary, given a page, candidate areas are detected by selective search, and EOH features are extracted, then the page is represented as a set of EOH features. This window-based representation is simple and efficient at finding a matching area in the page because a feature implicitly contains its position in the page, which is particularly useful for manga pages, and so this approach effectively solves the retrieval–localization problem.

3.3.2 Feature compression by product quantization

Given a set of EOH features, we compress them into binary codes using PQ [56], which is a successful ANN method, as shown in Figure 3.2(b). Applying PQ greatly reduces the computational cost of matching and reduces memory usage. After the features are compressed to binary codes, the approximate distances between a query vector and dataset codes can be computed efficiently by simple lookup operations (asymmetric distance computation (ADC) [56]). Therefore, the search can be performed efficiently even for large number of dataset vectors.

We now briefly describe the PQ algorithm. We represent $\mathbf{x} \in \mathbb{R}^D$ as a concatenation of M subvectors: $\mathbf{x} = [\mathbf{x}^1, \ldots, \mathbf{x}^M]$, where each $\mathbf{x}^m \in \mathbb{R}^{D/M}$. We define a subquantizer

$\mathbf{q}^m(\mathbf{x}^m)$ for $\mathbf{x}^m$ as follows:

$$\mathbf{x}^m \mapsto \mathbf{q}^m(\mathbf{x}^m) = \underset{\mathbf{c}^m \in C^m}{\arg\min} \left\| \mathbf{x}^m - \mathbf{c}^m \right\|. \tag{3.2}$$

The subcodebook C^m consists of K D/M-dimensional centroids learned by k-means [75] in advance. Therefore, $\mathbf{q}^m(\mathbf{x}^m)$ maps a subvector $\mathbf{x}^m$ to a code word $\mathbf{c}^m \in C^m$.

A product quantizer $\mathbf{q}(\mathbf{x})$ is defined as a concatenation of subquantizers:

$$\mathbf{x} \mapsto \mathbf{q}(\mathbf{x}) = \left[\mathbf{q}^1\left(\mathbf{x}^1\right), \ldots, \mathbf{q}^M\left(\mathbf{x}^M\right) \right]. \tag{3.3}$$

Therefore, an input $\mathbf{x}$ is mapped by $\mathbf{q}(\mathbf{x})$ to a code word $\mathbf{c} = [\mathbf{c}^1, \ldots, \mathbf{c}^M] \in C = C^1 \times \cdots \times C^M$.

The resultant code word $\mathbf{c}$ is encoded by a tuple $(i^1, \ldots, i^M)$ of M subcentroid indices. Because each i^m is an integer, i.e., $i^m \in \{0, \ldots, K-1\}$, the code $\mathbf{c}$ can be represented by $M \log_2 K$ bits. Typically, K is made a power of 2 so that $\log_2 K$ is an integer. As in other methods, we set K as 256 in this thesis ($\mathbf{c}^m$ is represented by 8 bit). Note that M, which shows the level of quantization, is set as 8 or 16, which yields the length of the code as 64 and 128 bits, respectively.

We now return to our EOH feature description. Recall the page is represented as $\mathcal{P} = \{\mathbf{x}_i | i \in \{1, \ldots, N\}\}$. Each feature $\mathbf{x}_i$ is quantized as $\mathbf{q}(\mathbf{x}_i)$, and the page is represented as

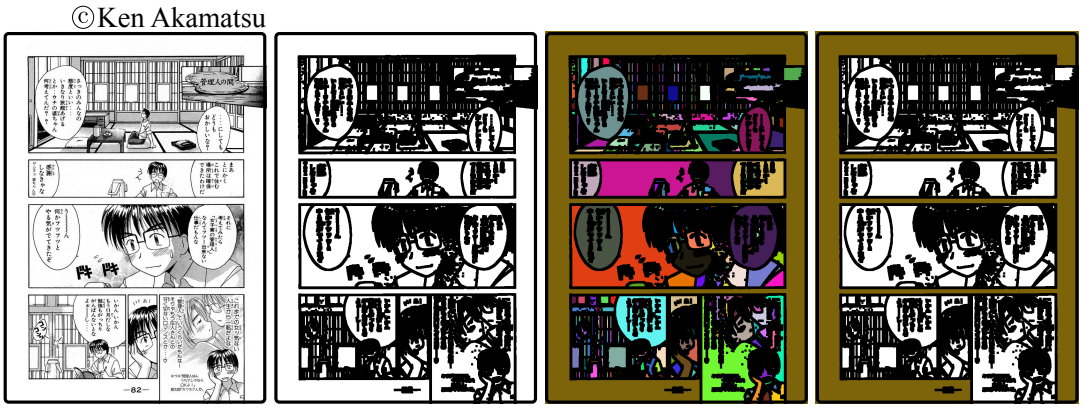
$$\mathcal{P}_{PQ} = \left\{ \mathbf{q}(\mathbf{x}_i) \middle| i \in \{1, \ldots, N\} \right\}, \tag{3.4}$$

where $\mathcal{P}_{PQ}$ is a set of quantized EOH features for a page. Because each $\mathbf{q}(\mathbf{x}_i)$ is recorded as a tuple of M subcentroid indices, which take $8M$ bits, $\mathcal{P}_{PQ}$ is stored by $8MN$ bits.

3.3.3 Search

Let us describe how to retrieve the nearest PQ-encoded feature from many manga pages (Figure 3.2(c)). Suppose there are P manga pages, and the quantized EOH features of the pth page are represented as $\mathcal{P}^p_{PQ} = \left\{ \mathbf{q}(\mathbf{x}^p_i) \middle| i \in \{1, \ldots, N^p\} \right\}$, where N^p means the number of EOH features from the pth page, and $p \in \{1, \ldots, P\}$. Given an EOH feature $\mathbf{y}$ from a query sketch, the search engine computes the nearest neighbors using:

$$\langle p^*, i^* \rangle = \underset{p \in \{1,\ldots,P\},\ i \in \{1,\ldots,N^p\}}{\arg\min} d_{AD}\left(\mathbf{y}, \mathbf{x}^p_i\right), \tag{3.5}$$



(a) (b) (c) (d)

Figure 3.5: (a) An input page. (b) Erosion is applied to white regions to thicken the lines. (c) White-connected areas are labeled with the same value. (d) The margin areas are selected.

where $d_{AD}(\cdot,\cdot)$ measures an approximate Euclidean distance between an uncompressed vector and a compressed PQ code [56]. This can be computed efficiently by lookup operations (ADC [56]).

By computing d_{AD} from $\mathbf{y}$ to each $\mathbf{x}_i^p$ and selecting the smallest one, p^* and i^* are computed. This is a simple linear search problem for PQ codes, which can be solved efficiently; e.g., 16 ms for searching one million 64-bit codes. Then we can say that the i^*th feature from the p^*th page is the nearest to the query sketch. The point here is that the problem is separated into a feature description and an ANN search. Therefore, even if the number of features is very large, we can employ the ANN techniques. In our system, searching from 21,142 pages takes 793 ms (without parallel implementation) or 70 ms (with parallel implementation) on a single computer using a simple PQ linear scan, which is fast enough. If much faster searches are required, we can employ more sophisticated ANN data structures [62, 8].

3.3.4 Skipping margins

Although PQ computing is fast, effective preprocessing of the manga pages of the dataset is essential (Figure 3.2(d)). A manga page comprises a series of frames and the interframe spaces (margins) are not important. For efficient searching, margin exclusion from retrieval candidates should be performed before extracting features. We label the margins as shown in Figure 3.5. First, the lines of the manga page image (Figure 3.5a) are thickened by applying an erosion [98] to the white areas (Figure 3.5(b)), thereby filling small gaps between black areas. Next, the

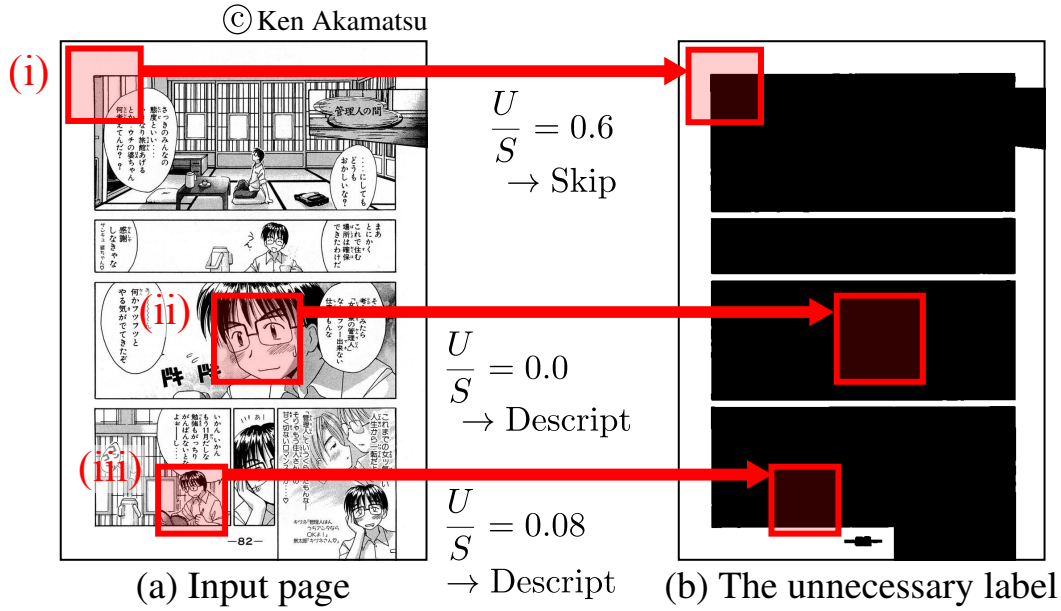


Figure 3.6: (a) Input image. (b) Its margin labels. In case (i), the red area (i) in (a) is skipped because $U/S = 0.6 > 0.1$. In case (ii), in contrast, the corresponding area is all black, and the feature is therefore extracted. In case (iii), $U/S = 0.08 < 0.1$, and an EOH feature is extracted.

white-connected areas are labeled by connected-component labeling [78] as shown in Figure 3.5(c), where different colors for the labels have been used for visualization. Finally, areas are selected as margins by finding the most frequent label appearing in the outermost peripheral regions (Figure 3.5(d)). Because interframe spaces tend to connect to the outer areas, the method succeeds in most cases.

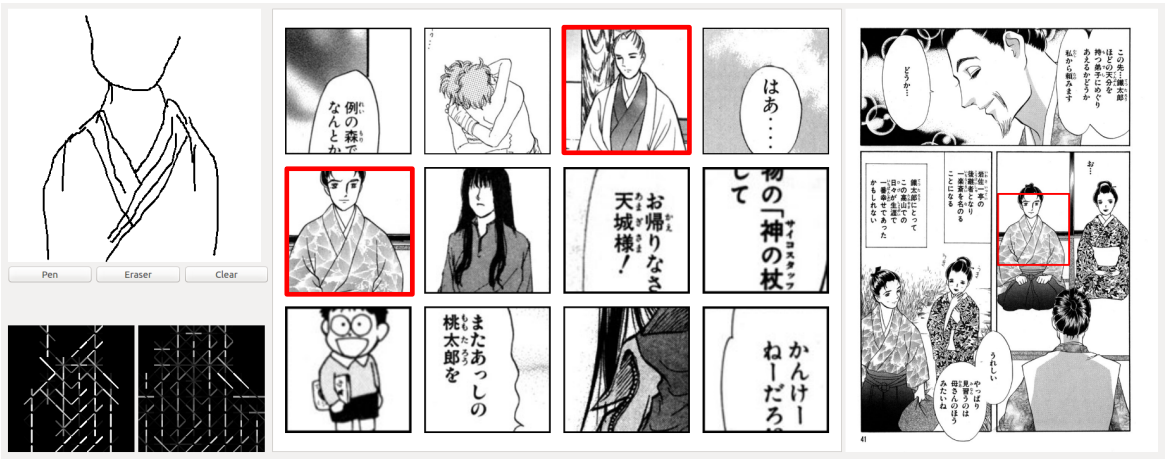
Let us define the patch area as $S(\mathbf{x})$, and the margins as $U(\mathbf{x})$ (e.g., the colored areas shown in Figure 3.5(d)). We extract a feature only if its area ratio U/S is less than a threshold, which we set to 0.1. Equation (3.4) is therefore rewritten as

$$\mathcal{P}_{PQ} = \left\{ \mathbf{q}(\mathbf{x}_i) \middle| \frac{U(\mathbf{x}_i)}{S(\mathbf{x}_i)} < 0.1, \;\; i \in \{1, \ldots, N\} \right\}. \tag{3.6}$$

Intuitively, this means that if an area belongs to the margin, it is skipped in the feature extraction step. An example of the process is shown in Figure 3.6.

3.4 Query interaction

Querying is a difficult issue for manga retrieval. We call this the query modality problem of manga retrieval. We cannot employ textual or tag data because EOH



(a)

(b)

(c)

Figure 3.7: Relevance feedback. (a) A user draws strokes and finds a target (Japanese-style clothes) in the third and fifth results (red borders). (b) By selecting the region in the retrieved page, another retrieval is performed automatically with this as the query. (c) A wide range of images of Japanese-style clothes is obtained.

Figure 3.8: Query retouch. (a) Results of relevance feedback. Target characters have red borders. (b) The user adds strokes and draws glasses on the target character. Other target characters with glasses are then successfully retrieved (red borders). Note that users can erase lines using an eraser tool. (c) Both relevance feedback and query retouch can be conducted multiple times.

features do not involve such information. For a natural and intuitive interface, we prefer sketch-based queries. Because manga itself comprise sketches drawn by authors, sketching is compatible with manga.

In addition, we can make use of sketching not only for the initial query, but also for additional interaction with the retrieved results. The queries that can be performed using our framework are summarized as follows: (1) **sketch querying**, the proposed sketch-based retrieval described above; (2) **relevance feedback**, which reuses the retrieved results; and (3) **query retouch**, in which the results of relevance feedback are modified and reused.

Relevance feedback: We propose a simple interaction to reuse retrieved results, which is inspired by the relevance feedback techniques proposed in the information retrieval literature [81]. Users can reuse a retrieved result simply by selecting a region in the retrieved manga page, as shown in Figure 3.7. With relevance feedback, even novice users can use professional manga images as queries. Note that a user can use any region in the page as a query.

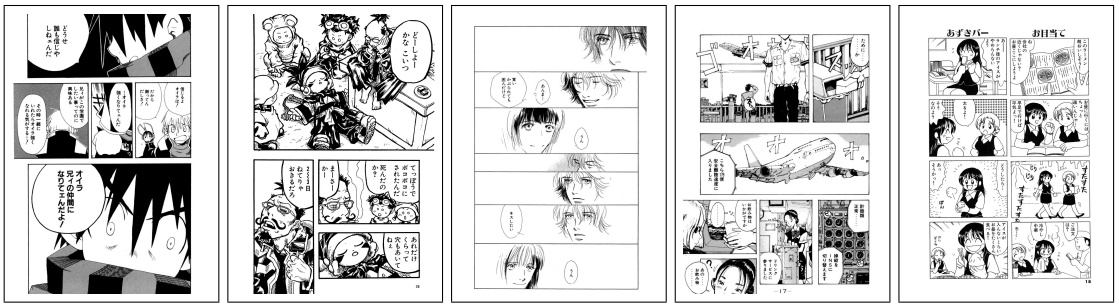
Query retouch: We propose a new methodology for modifying and reusing the results. In query retouch, we can modify either the initial sketch or a query taken from the results of a retrieval (Figure 3.8). As the query is changed by adding lines or partial erasure, the results will change immediately. As a result, we can intuitively change the retrieval results in the direction that we want.

According to Mei and colleagues [81], these interactions are classified as "interactive reranking," and are especially useful for the visual search problem because "*visual data is particularly suited for interaction, as a user can quickly grasp the vivid visual information and thus judge the relevance at a quick glance.*" [81].

3.5 Manga dataset

In this section, we introduce a new dataset of manga images, namely the **Manga109** dataset for evaluation. The Manga109 dataset consists of 109 manga titles, and is made publicly available for academic research purposes with proper copyright notation. Figure 3.9 shows example pages from the Manga109 dataset.

A large-scale dataset of manga images is important and useful for manga research. Although several papers related to manga image processing have been published, fair comparisons among the proposed methods have not been conducted because of the lack of a dataset with which the methods can be evaluated. In preparing a dataset, the most serious and intractable problem is copyright. Because manga is artwork and



(a) Gakuen Noise, vol.1, p.117 ©Daisuke Inohara (b) PLANET7, vol.2, p.36. ©Shujip Takeya (c) Aosugiru Haru, p.26. ©Momoko Okuda (d) Hanzai Kousyounin Eitarou, vol.1, p.17. ©Takashi Ki Minegishi (e) OL Lunch, p.18. ©Youko Sanri

Figure 3.9: Example images from the Manga109 dataset. Each caption shows the bibliographic information of an image (title, volume, and page).

protected by copyright, it is hard to construct a publicly available dataset. Therefore, researchers have collected their own manga images or used manga images drawn by amateurs. These are not appropriate for manga research because: (1) the number of manga images is small; (2) the artistic quality of the manga images is not always high; and (3) they are not publicly available. The Manga109 dataset, on the other hand, clears these three conditions. It contains 21,142 pages in total, and we can say that it is currently the largest manga image dataset. The quality of the images is high because all titles were drawn by professional manga authors.

As is widely known, well-crafted image datasets have played critical roles in evolving image processing technologies, e.g., the PASCAL VOC datasets [34] for image recognition in the 00s, and ImageNet [93] for recent rapid progress in deep architecture. We hope the Manga109 dataset will contribute to the further development of the manga research domain.

Image collection: All manga titles in the Manga109 dataset have been previously published in manga magazines[1], i.e., they were written by professional manga creators. The manga titles are in an archive "Manga Library Z" run by J-comi. It has more than 4000 titles, most of which are currently out of print. With the help of J-comi, we chose 109 titles from the archive that cover a wide range of genres and publication years. We obtained permission from the creators to use them for academic research purposes. Thus researchers can use them freely with appropriate citation for research challenges, including not only for retrieval and localization,

[1]Japanese manga is usually published in a manga magazine first, which contains many manga titles. Then it comes out in independent book form for each manga title.

but also for colorization, data mining from manga, and so on. The manga titles were originally published from the 1970s to the 2010s. The Manga109 dataset covers various kinds of categories, including humor, battle, romantic comedy, animal, science fiction, sports, historical drama, fantasy, love romance, suspense, horror, and four-frame cartoons. Please see the details (titles, author names, and so on) in our project page [67].

Dataset statistics: The dataset consists of 109 manga titles. Each title includes 194 pages on average, with a total of 21,142 pages. The average size of images is 833×1179, which is bigger than the image sizes usually used for object recognition and retrieval tasks, e.g., 482×415 pixels on average for ILSVRC 2013 [2]. This is because manga pages contain multiple frames and thus require higher resolutions.

Relation to other datasets: *eBDtheque* [40] is a publicly available comic dataset. It consists of 100 comic pages with detailed meta-information such as text annotations. Compared with eBDtheque, our Manga109 dataset contains a much larger number of pages. Manga109 is useful for large-scale experiments, whereas eBDtheque is beneficial for small but detailed evaluations such as object boundary detection.

3.6 Experimental Results

In this section, we present three evaluations: (1) a comparative study with previous methods for manga description (Section 3.6.1); (2) a localization evaluation (Section 3.6.2); and (3) a large-scale qualitative study (Section 3.6.3). These three experimental results verify that our system presents solutions to the corresponding manga image description problem, the retrieval–localization problem, and the query modality problem. In the comparative study and the localization evaluation, we used a single-thread implementation for a fair comparison, and employed a parallel implementation for the large-scale study.

3.6.1 Comparative study

A comparative study was performed to evaluate how well the proposed framework can represent manga images compared with previous methods such as those introduced in Section 3.2. We compared our proposal with a baseline (BoF with large window SIFT [97]), the state-of-the-art of BoF-based methods (FV [97]), and the state-of-the-art of chamfer-based methods (Compact OCM [106]). All experiments

[2] http://www.image-net.org/challenges/LSVRC/2014/index#data

Table 3.1: Image statistics for comparative study. All images are cropped frames from the Manga109 dataset.

target type	difficulty	#ground truth	#dataset	#query
Boy-with-glasses	easy	67	8563	30
Chombo	normal	178	8563	30
Tatoo	hard	81	8563	30

were conducted on a PC with a 2.8 GHz Intel Core i7 CPU and 32 GB RAM, using C++ implementations.

Frame image dataset: For evaluation, we cropped frames from 10 representative manga titles from the Manga109 dataset[3]. There were 8889 cropped frames, and the average size was 372×341. We used these frames for retrieval. Note that we used frames instead of pages for this comparison because the features of BoF, FV, and Compact OCM are only comparable within a frame. Although the frame is less complex than the page, retrieval is not easy because the size of frames varies greatly, and the localization problem still exists as shown in Figure 3.10c, where the target (a head of a boy) is small and the frame includes other objects and backgrounds.

Target images: To make the comparisons, we chose three kinds of targets: *Boy-with-glasses* (Figure 3.10c), *Chombo* (Figure 3.10f), and *Tatoo* (Figure 3.10i), which are from manga titles *Lovehina*, *Mukoukizu no Chombo*, and *DollGun*, respectively. *Boy-with-glasses* is an easy example, *Chombo* is much harder because the face might be either to the right or left, and *Tatoo* is the most difficult because it is so small compared with the frame. These target images are treated as ground truths.

Query images: To prepare query sketches, we invited 10 participants (seven novices and three skilled artists, such as a member of an art club), and asked them to draw sketches. First, we showed them query images for 20 s. Next, they were asked to draw a sketch for the query. Each participant drew all target objects; therefore, we collected $10 \times 3 = 30$ queries. Examples of queries are shown in Figure 3.10. We used a 21-inch pen display (WACOM DTZ-2100) for drawing.

Results of comparative study: Using the queries, we evaluated each method using standard evaluation protocols in image retrieval: recall@k and mean average precision [16]. Note that the ground-truth target images were labelled manually. All frame images were used for the evaluation, and the images of the different targets were regarded as distractors. The statistics of the frame images are shown in Table 3.1.

[3] *Lovehina vol.1*, *Gakuen Noise*, *DollGun*, *Hanzai Kousyounin Minegishi Eitarou*, *Bakuretsu KungFu Girl*, *Aosugiru Haru*, *OL Lunch*, *Mukoukizu no Chombo*, *Highschool Kimengumi vol.1*, and *PLANET7*

Figure 3.11 shows the results for each target. Note that this task is challenging, and all scores tend to be small. The queries from users are not always similar to the target. On the contrary, some novices even drew queries that were dissimilar to the target, as shown in Figure 3.10d and Figure 3.10f. Still, in all cases, the proposed method achieved the best scores. In particular, the proposed method outperformed the other methods for *Boy-with-glasses*. Note that BoF and FV received almost zero scores for the *Tatoo* case because they are not good at finding a relatively small instance from an image. From these experiments, we can say that BoF-based methods do not satisfy the purpose in manga retrieval.

Note that we did not apply any approximation steps for a fair comparison. Compact OCM measures an original chamfer distance without an approximation (a sparse projection). We did not compress a feature using PQ in the proposed method.

About quantization: When we applied PQ to the features, the score decreased according to the quantization level, as shown in Figure 3.12. There is a clear trade-off between the rate of compression and accuracy. From the result, we accepted $M = 16$ as a reasonable option; i.e., a feature is divided into 16 parts for PQ compression and encoded to a 16-byte code. Note that, interestingly, there is not a clear relation between accuracy and the number of cells (c^2). The feature description becomes finer with a large number of cell divisions, but it does not always achieve higher recall. This indicates that some level of abstraction is required for the sketch retrieval task. We accepted $c = 8$ for all experiments, i.e., a selected area is divided into 8×8 areas for feature description.

Parameter settings and implementation details: We show parameter settings and implementation details used for the evaluation. For BoF, SIFT features are densely extracted where the size of a SIFT patch is 64×64 pixels, with a 2×2 spatial pyramid, and the number of vectors in the dictionary is set to 1024. The final number of dimension is 4096. For FV, a Gaussian Mixture Model with 256 Gaussians was used, with a 2×2 spatial pyramid. The dimensions of SIFT are reduced from 128 to 80 by PCA. For BoF and FV, we leveraged the same parameter settings as in Schneider and colleagues [97], with the `vlfeat` implementation [114]. For selective search, we employed the `dlib` implementation [64], with little Gaussian blurring before applying selective search. To train code words for BoF, FV, and PQ, we randomly selected 500 images from the dataset. Note that they were excluded and not used for testing. To eliminate small patches, we set a minimum length of a patch as 100 pixels and discarded patches that were smaller than that.

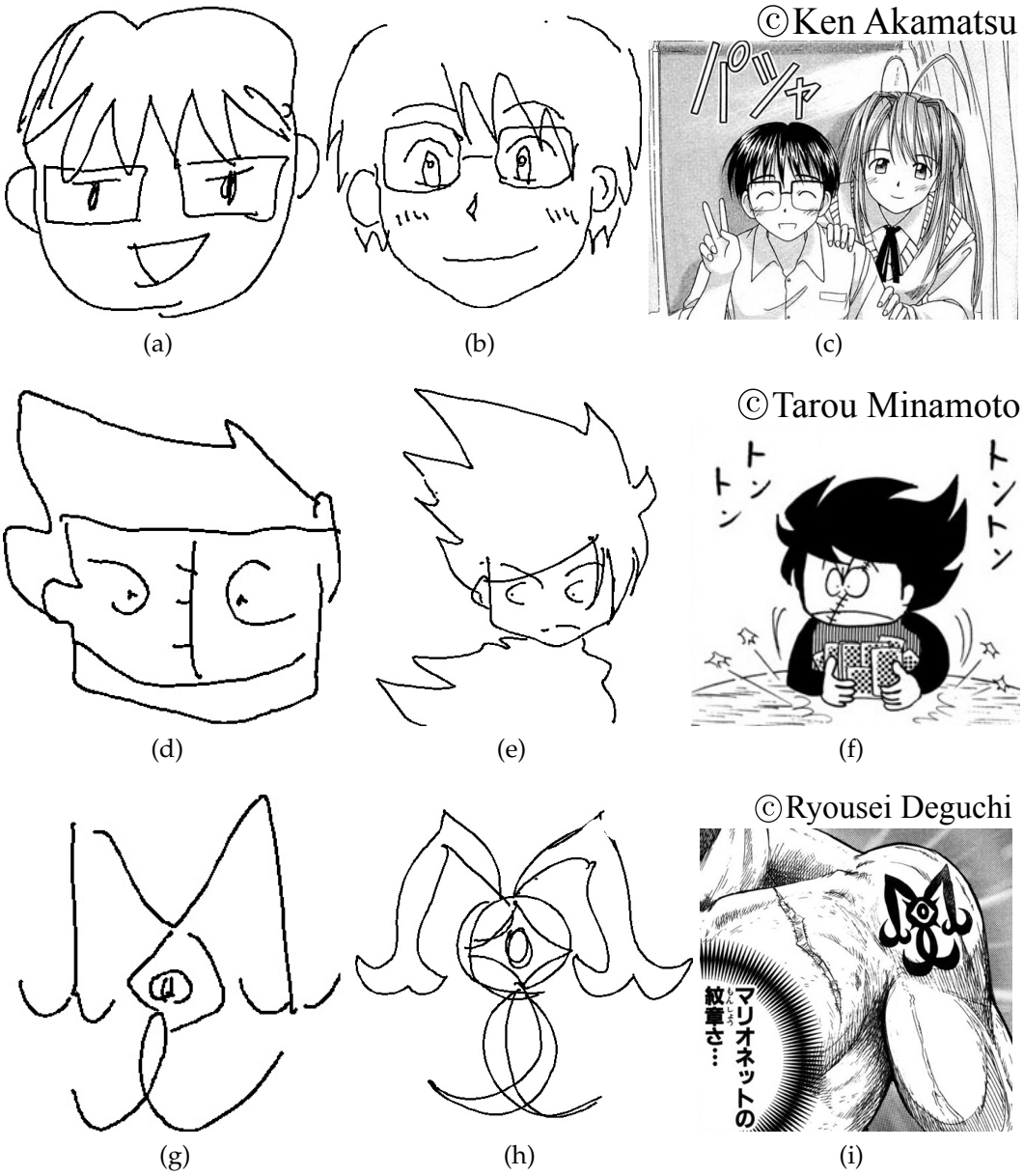
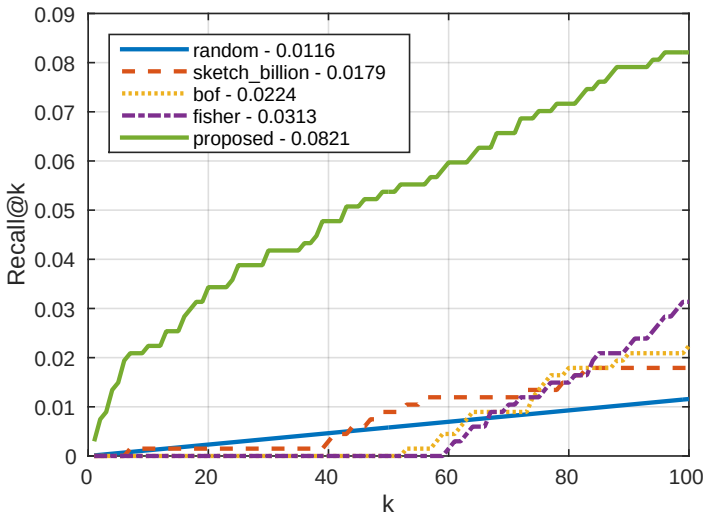
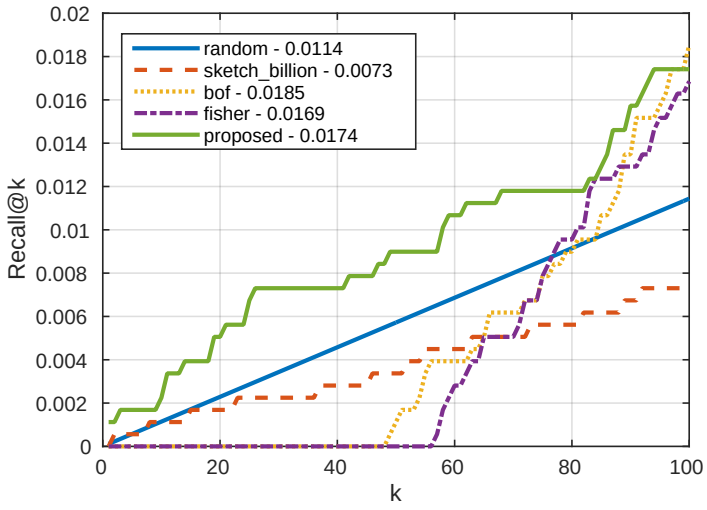


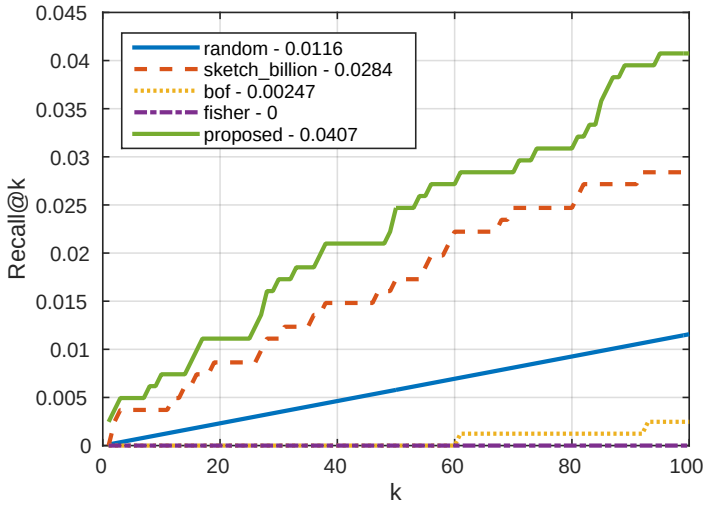
Figure 3.10: Targets for the comparative study. Left to right: query sketches by novice artists, skilled artists, and ground-truth images. Top to bottom: targets, *Boy-with-glasses*, *Chombo*, and *Tatoo*.



(a) Boy-with-glasses



(b) Chombo



(c) Tatoo

Figure 3.11: Results of the comparative study. Values in the legend show Recall@100.

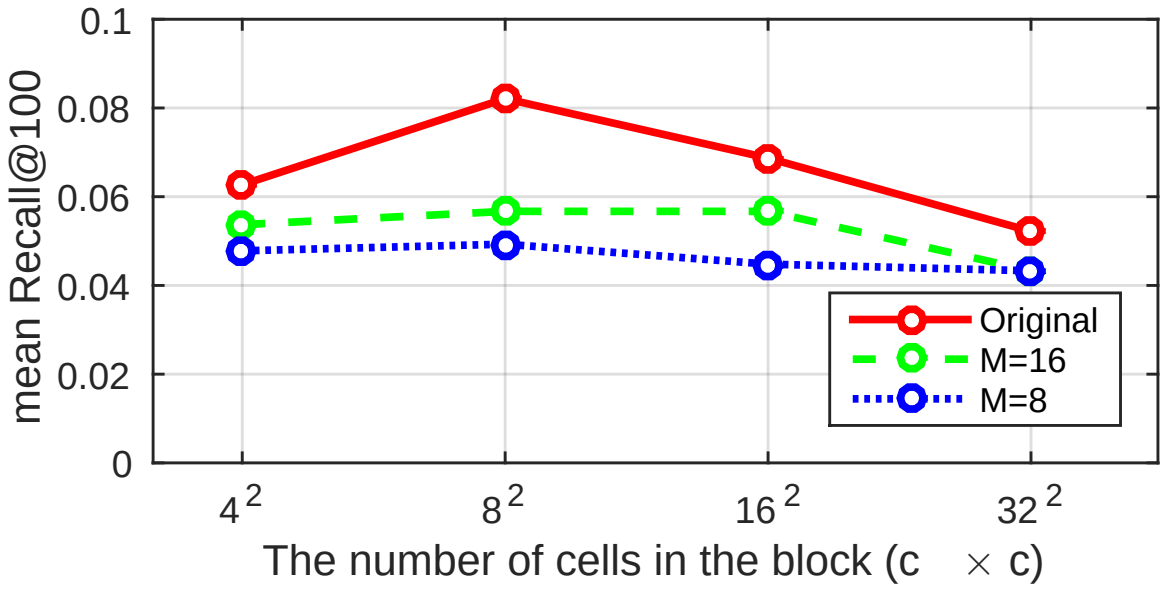


Figure 3.12: The effect of feature compression by PQ. The Y-axis represents the retrieval performance. The X-axis shows the number of cells. Each line corresponds to a compression level. As the features are compressed by PQ (i.e., the feature is represented by a smaller number of subvectors (M)), the score decreases compared with the original uncompressed feature.

3.6.2 Localization evaluation

Next, we evaluated how well the proposed method can localize a target object in manga pages. The setup is similar to that for image detection evaluation [34].

Images: As query sketches, we used the Boy-with-glasses sketches collected in Section 3.6.1. The ground-truth windows were manually annotated from Lovehina. A total of 69 windows was annotated. For evaluation, we prepared two datasets. (i) A Lovehina dataset, which is a title of *Lovehina vol.1*, and consists of 192 pages, including the pages containing ground-truth windows. (ii) The Manga109 dataset, which is the sum of all manga data, consists of 109 titles with a total of 21,142 pages. Note that the Lovehina data is included in the Manga109 dataset, so (i) is a subset of (ii).

In contrast to the previous comparative study, localization is performed on a page. Therefore, this is an object localization task: given a query image, find a target instance in an image. In our case, however, we must find the target from many manga pages (21,142 pages, for Manga109).

Evaluation criteria For evaluation, we employed a standard PASCAL overlap criterion [34]. Given a bounding box (retrieved result) from the method, it is judged

Table 3.2: Results for localization evaluation (single thread implementation).

	#image	#patch	mAP@100	memory	runtime
Lovehina	192	138K	1.12×10^{-2}	2.21 MB	11.6 ms
Manga109	21,142	14M	1.43×10^{-4}	204 MB	331 ms

to be true or false by measuring the overlap of the bounding box and ground-truth windows. Denote the predicted bounding box as B_p, and the ground-truth bounding box as B_{gt}, the overlap is measured by:

$$r = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}. \tag{3.7}$$

We judged the retrieved area is true if $r > 0.5$. If multiple bounding boxes are produced, at most one among them is counted as correct.

For each query, we find the top 100 areas using the proposed retrieval method from the dataset (Lovehina or Manga109), then the retrieved areas are judged using Equation (3.7). Then we can compute the standard mAP@100 from the result (true/positive sequence).

Result of localization evaluation: We show the results in Table 3.2. With our single implementation, searching the Manga109 dataset (14M patches) took 331 ms. This is fast enough, and was further improved (70 ms) using a parallel implementation as discussed in Section 3.6.3. We also show theoretical values of memory consumption for EOH features (#path×8M). The whole Manga109 dataset consumes only 204 MB. As can be seen from mAP, this task is difficult. There are possibly hundreds of thousands of candidate areas in the windows (138K for Lovehina, and 14M for Manga109), but only 69 ground-truth areas. Examples of retrieved results are shown in Figure 3.13. For the Lovehina data, the first result is a failure but the second is correct. For the Manga109 dataset, the first success can be found at the 35th result. The first result shares similar characteristics to the query (wearing glasses) even though the result is incorrect.

3.6.3 Large-scale qualitative study

In this section, we show a qualitative study of retrieval from the Manga109 dataset. The whole system was implemented using a GUI as shown in Figure 3.1.

We employed a parallel implementation using the Intel Thread Building Library, and the average computation time was 70 ms for the Manga109 dataset (21,142

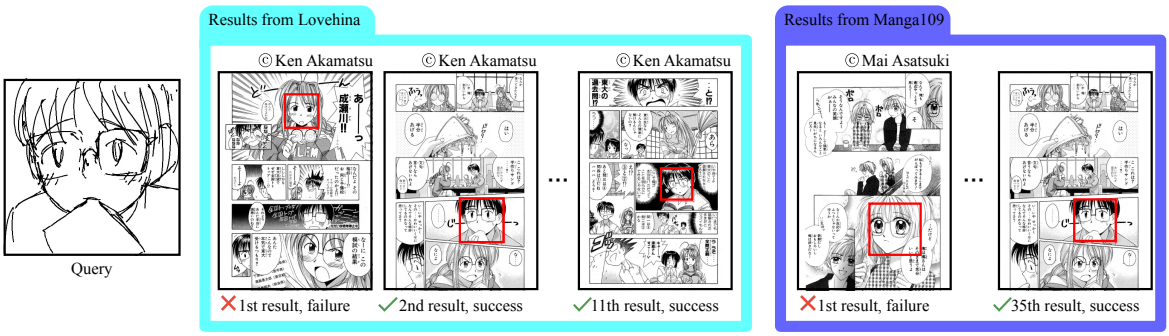


Figure 3.13: Examples of localization experiments for the Lovehina dataset and Manga109 dataset.

images). The parallelization was straightforward. Neighbors are computed for each manga title in parallel. In the implementation, we selected the most similar feature from a page (not keeping all features per page), then merged the results.

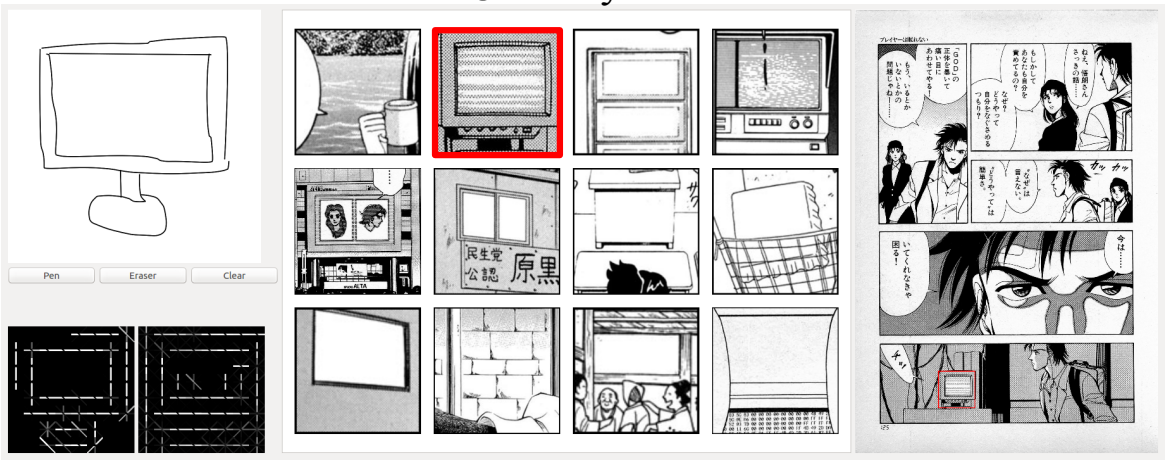
Qualitative study using a sketch dataset: We qualitatively evaluated the proposed method using a public sketch dataset as queries. We used representative sketches [32] as queries. The 347 sketches each had a category name, e.g., "panda."

Figure 3.14a and Figure 3.14b show successful examples. We could retrieve objects from the Manga109 dataset successfully. In particular, the retrieval works well if the target consists of simple geometric shapes such as squares, as shown in Figure 3.14b. This tendency is the same as that for previous sketch-based image retrieval systems [33, 106]. Figure 3.14c shows a failure example, although the retrieved glass is similar to the query.

As can be seen in Figure 3.14c, text regions are sometimes retrieved and placed at the top of the ranking. Because users usually do not require such results, detecting and eliminating text areas would improve the results, which remains as a future work.

More results by relevance feedback: We show more results from queries of character faces using the proposed relevance feedback in Figure 3.15. We see that the top retrieved results were the same as (or similar to) the characters in the query. In this case, all the results were drawn by the same author. Interestingly, our simple edge histogram feature captured the characteristic of authors. Figure 3.15b shows the results of "blush face." In Japanese manga, such blush faces are represented by hatching. By the relevance feedback, blushed characters were retrieved from various kinds of manga titles. These character-based retrievals are made possible by content-based search. This suggests that the proposed query interactions are beneficial for manga search.

(a) Trousers

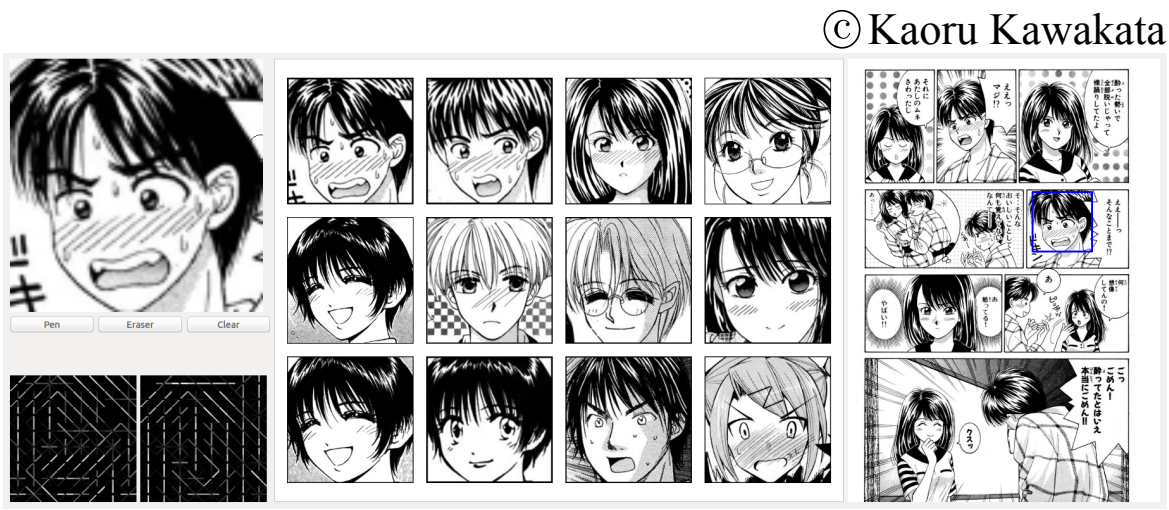


(b) Computer monitor

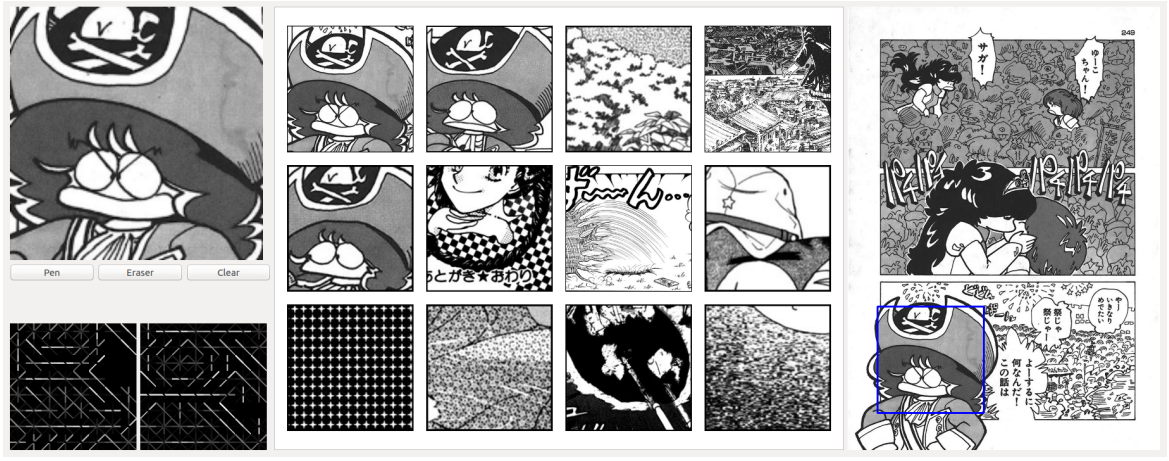
(c) Hourglass

Figure 3.14: Results of the subjective study using representative sketches [32] as queries.

(a)



(b)



(c)

Figure 3.15: More results for relevance feedback from Manga109 dataset.

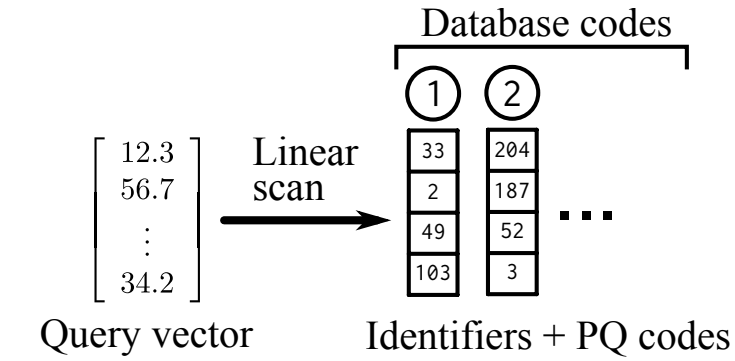
Chapter 4

Theoretical Improvement: efficient search using hash tables

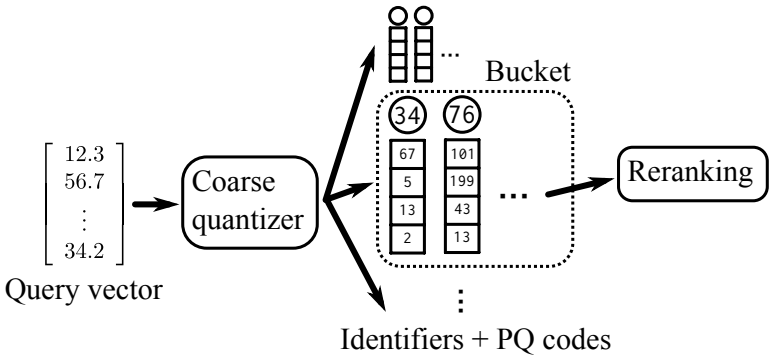
In Chapter 3, we proposed the sketch-based manga retrieval system. To handle more data, further analysis and improvement of a core part of the system, nearest neighbor search, is required. In this chapter, we propose a fast and practical solution to the large-scale nearest neighbor problems.

4.1 Introduction

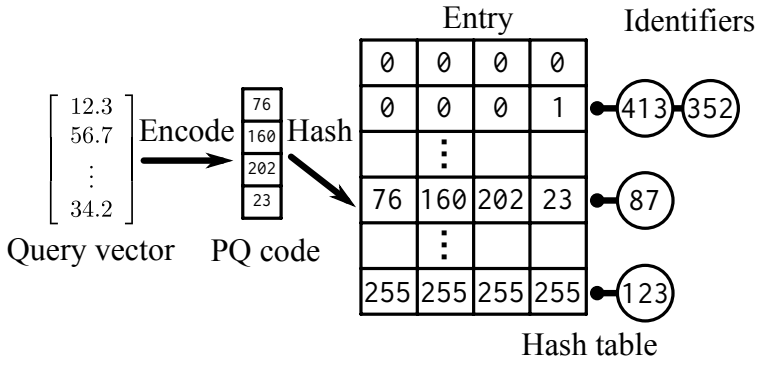
With the explosive growth of multimedia data, compressing high-dimensional vectors and performing approximate nearest neighbor (ANN) searches in the compressed domain is becoming a fundamental problem in handling large databases. *Product quantization (PQ)* [56], and its extensions [82, 37, 6, 125, 116, 43, 9, 126], are popular and successful methods for quantizing a vector into a short code. PQ has three attractive properties: (1) PQ can compress an input vector into an extremely short code (e.g., 32 bit) that enables it to handle typically one billion data points in memory at once; (2) the approximate distance between a raw vector and a compressed PQ code can be computed efficiently (the so-called *asymmetric distance computation (ADC)* [56]), which is a good estimate of the original Euclidean distance; and (3) the data structure and coding algorithms are surprisingly simple. By quantizing database vectors into short codes in advance, vectors similar to a given query can be found from the database codes by a linear comparison of the query with each code using ADC (see the linear ADC scan in Figure 4.1a).



(a) Linear ADC scan



(b) Short-code-based inverted indexing system



(c) Proposed PQTable

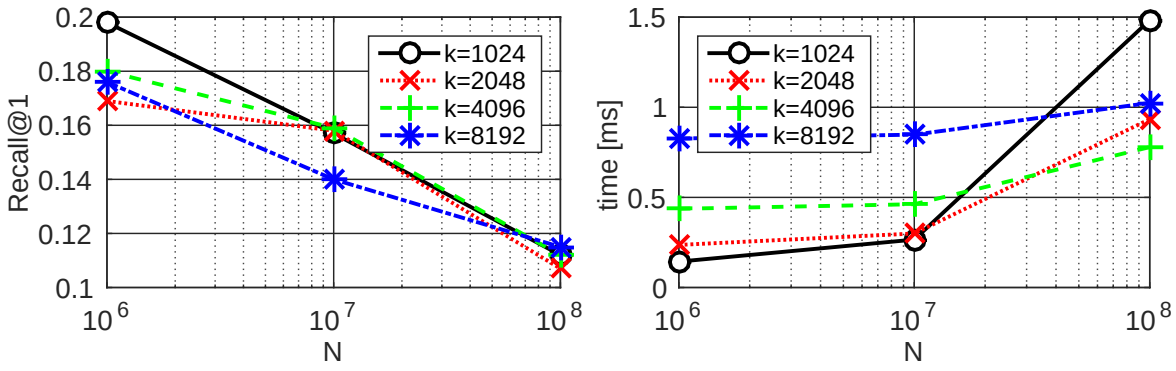
Figure 4.1: Data structures of ANN systems: linear ADC scan, short-code-based inverted indexing systems, and PQTable.

Although a linear ADC scan is simple and easy to use, ANN search by linear ADC scanning is efficient only for small datasets because the search is exhaustive, i.e., the computational cost is at least $O(N)$ for N PQ codes. To handle large (e.g., $N = 10^9$) databases, *short-code-based inverted indexing systems* [5, 122, 37, 62, 8, 7] have been proposed, which are currently state-of-the-art ANN methods (see Figure 4.1b). Such systems operate in two stages: (1) coarse quantization and (2) reranking via short codes. A database vector is first assigned to a bucket using a coarse quantizer (e.g., k-means [56], multiple k-means [122], or Cartesian products [5, 53]), then compressed to a short code using PQ [56] or its extensions [82, 37] and finally stored as a posting list [1]. In the retrieval phase, a query vector is assigned to the nearest buckets by the coarse quantizer, then associated items in corresponding posting lists are traversed, with the nearest one being reranked via ADC [2]. These systems have been reported as being fast, accurate, and memory efficient, by being capable of holding one billion data points in memory and conducting a retrieval in milliseconds.

However, such inverted indexing systems are built by a process of carefully designed manual parameter tuning. The processing time and recall rates strongly depend on the number of space partitions in the coarse quantizer, and its optimal size needs to be decided manually, as shown in Figure 4.2. For example, this simple example shows that $k = 1024$ is the fastest setting for $N = 10^7$, but the slowest for $N = 10^8$. These relationships do not become clear until several parameter combinations for the coarse quantizers are examined. However, training and testing coarse quantizers is computationally expensive. For example, we found that, to plot a single dot of Figure 4.2 for $N = 10^9$, the training took around a day and building the inverted index structure took around three days, using 16-core machines. This is particularly critical for a recent per-cell training ANN system [7, 62], which requires even more computation to build the system. It should be pointed out that achieving state-of-the-art performances by ANN systems depends largely on special tuning for the testbed dataset, e.g., SIFT1B [56]. There is no guarantee that we can always achieve the best performance by such systems. In real-world applications, it often happens that cumbersome trial-and-error-based parameter tuning is required, which might well turn ordinary users away from these systems.

To achieve an ANN system that would be suitable for practical applications, we propose the *PQTable*, an exact, nonexhaustive, NN search method for PQ codes (see Figure 4.1c). We do not employ an inverted index data structure, but find similar PQ

[1] In practice, the residual difference between the query vector and the codeword is encoded.
[2] There are smarter versions of this computation [8].



(a) Recall@1 (b) Runtime per query

Figure 4.2: Performance of a short-code-based inverted indexing system (IVFADC [56]), showing the relationship between the database size, N, and Recall@1 or runtime per query for various k, which is the number of space partitions for the coarse quantizer. The search range w is set to one. Note that the search range is also an important parameter, which has been chosen manually.

codes directly from the database. This achieves exactly the same accuracy as a linear ADC scan, but requires a significantly shorter time (10.3 ms, instead of 18 s, for the SIFT1B data using 64-bit codes). In other words, this section proposes an efficient ANN search scheme to replace a linear ADC scan when N is sufficiently large. As discussed in Section 4.4, the parameter values required to build the PQTable can be calculated automatically. Note that, as discussed in Section 4.5, PQTable requires larger memory footprints than storing PQ codes linearly.

The main idea of the PQTable is the use of a hash table, where the PQ code itself is used directly as an entry in the hash table (see Figure 4.1c). That is, identifiers of PQ codes in the database are directly associated with the hash table by using the PQ code itself as an entry. Given any new query vector, it can be PQ coded and identifiers associated with the entry are retrieved.

There are advantages and disadvantages in the use of inverted indexing systems as well as for our proposed method. Inverted indexing systems are accurate and fast, but they do require manual parameter tuning and time-consuming training. The PQTable can achieve the same recall rates as a linear ADC, but its performance (accuracy, runtime, and memory usage) may lag the latest inverted indexing systems with the best parameters. The PQTable does not require any parameter tuning or training steps for the coarse quantizers, which are important aspects of real-world applications.

Relationship with previous methods

Hamming-based ANN systems: As an alternative to PQ-based methods, another major approach to ANN are the Hamming-based methods [119, 38, 115], in which two vectors are converted to bit strings whose Hamming distance approximates their Euclidean distance. Comparing bit strings is faster than comparing PQ codes, but is usually less accurate for a given code length [41].

In Hamming-based approaches, bit strings can be linearly scanned by computing their Hamming distance, which is similar to linear ADC scanning in PQ. In addition, a multitable algorithm has been proposed to facilitate fast ANN search [83], which computes exactly the same result as a linear Hamming scan but is much faster. However, a similar querying algorithm and data structure for fast computing have not been proposed for the PQ-based method to date. Our work therefore extends the idea of such a multitable algorithm [83, 39] to PQ codes, where a Hamming-based formulation cannot be applied directly.

Extensions to PQ: Recent reports [125, 6, 9, 126] have proposed a new encoding scheme that is regarded as a generalized version of PQ. Because the data structure used in these approaches is similar to PQ, our proposed method can also be applied to such approaches. In this thesis, we discuss only the core of the PQ process for simplicity.

4.2 Background: Product Quantization

In this section, we briefly review PQ [56].

Product quantizer: As introduced in Section 3.3.2, a product quantizer is defined as follows. Let us denote any $\mathbf{x} \in \mathbb{R}^D$ as a concatenation of M subvectors: $\mathbf{x} = [\mathbf{x}^1, \ldots, \mathbf{x}^M]$, where we assume that the subvectors have an equal number of dimensions D/M, for simplicity. A product quantizer $\mathbf{q}(\cdot)$ is defined as a concatenation of subquantizers:

$$\mathbf{x} \mapsto \mathbf{q}(\mathbf{x}) = \left[\mathbf{q}^1\left(\mathbf{x}^1\right), \ldots, \mathbf{q}^M\left(\mathbf{x}^M\right)\right], \tag{4.1}$$

where each subquantizer $\mathbf{q}^m(\cdot)$ is defined as

$$\mathbf{x}^m \mapsto \mathbf{q}^m(\mathbf{x}^m) = \underset{\mathbf{c}^m \in C^m}{\arg\min} \left\|\mathbf{x}^m - \mathbf{c}^m\right\|. \tag{4.2}$$

Each subcodebook C^m comprises K D/M-dim centroids learned by k-means [75]. Therefore, $\mathbf{q}(\mathbf{x})$ maps an input $\mathbf{x}$ to a codeword $\mathbf{c} = [\mathbf{c}^1, \ldots, \mathbf{c}^M] \in C = C^1 \times \cdots \times C^M$.

Because $\mathbf{c}$ is encoded as a tuple $(i^1, \ldots, i^M)$ of M subcentroid indices, where each $i^m \in \{0, \ldots, K-1\}$, it can be represented by $M \log_2 K$ bits. Typically, K is set as a power of 2, making $\log_2 K$ an integer. In this thesis, we set K as 256 (this is the typical setting used in many papers, for which $\mathbf{c}^m$ is represented by 8 bits), and M is set as 1, 2, 4, or 8, which yields a length for the code of 8, 16, 32, or 64 bits, respectively.

ADC: Distances between encoded vectors and a raw vector can be computed efficiently. Suppose that there are N data points $\mathcal{Y} = \left\{\mathbf{y}_j \in \mathbb{R}^D \middle| j = 1, \ldots, N\right\}$ that have been encoded and stored. Given a new query vector $\mathbf{x} \in \mathbb{R}^D$, the squared Euclidean distance from $\mathbf{x}$ to a point $\mathbf{y}$ is approximated by asymmetric distance (AD):

$$d_{AD}(\mathbf{x}, \mathbf{y})^2 = \sum_{m=1}^{M} d(\mathbf{x}^m, \mathbf{q}^m(\mathbf{y}^m))^2. \tag{4.3}$$

This can be computed as follows. First, $\mathbf{x}^m$ is encoded by its corresponding sub-codebook, C^m, thereby generating a distance table whose size is $M \times K$, where each (m, k) entry denotes the squared Euclidean distance between $\mathbf{x}^m$ and the k-th subcentroid in C^m, namely $\mathbf{c}_k^m$. Suppose that a subvector $\mathbf{y}^m$ of each $\mathbf{y}$ is encoded in advance as $\mathbf{c}_k^m$, i.e., $\mathbf{q}^m(\mathbf{y}^m) = \mathbf{c}_k^m$. It follows that an (m, k) in the distance table means $d\left(\mathbf{x}^m, \mathbf{c}_k^m\right)^2 = d(\mathbf{x}^m, \mathbf{q}^m(\mathbf{y}^m))^2$. We can then compute Equation (4.3) by simply looking up the distance table M times and summing the distances. The computational cost of the whole process is $O(KD + NM)$, which is fast for small N but still linear in N.

4.3 PQTable

As shown in Figure 4.1c, the proposed system involves a hash table within which the PQ code itself is used as an entry. In the retrieval phase, a query vector is PQ coded and identifiers associated with the entry are retrieved. This is very straightforward, but there are two problems to be solved.

The empty-entries problem: Suppose that a new query is PQ coded and hashed. If the identifiers associated with the entry of the query are not present in the table, the hashing fails. To continue the retrieval, we would need to find candidates for the query by some other means.

The long-code problem: Even if we can find the candidates and continue querying, the retrieval is not efficient if the length of the codes is very long compared to the size of the database, e.g., 64-bit codes for 10^9 vectors. Figure 4.3 shows the relationship between the number of database vectors and the probability of finding

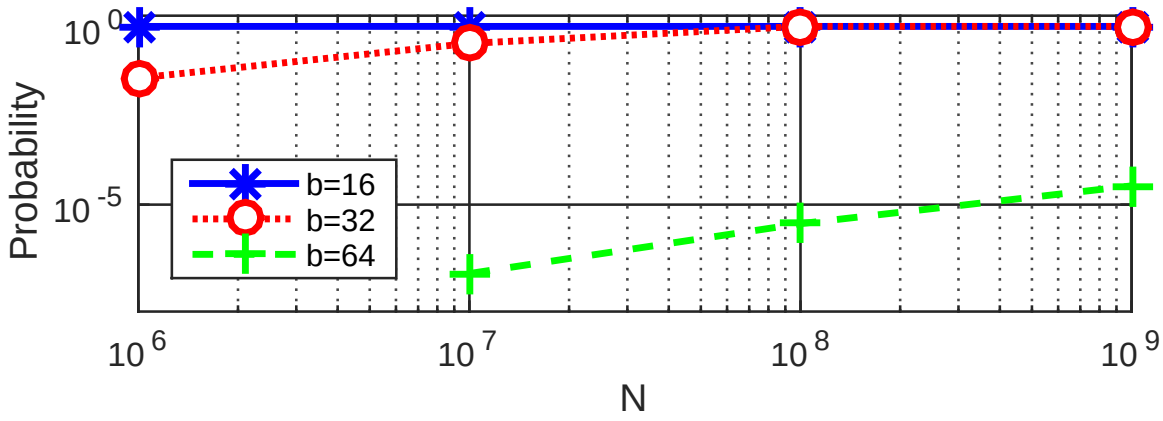


Figure 4.3: The probability of finding the first nonempty entry (the first identifier). Plots are computed over 50 queries from the SIFT1B data for various code lengths b.

the first nonempty entry. If the length of the codes b is 16 or 32 bits, there is a high probability that the first identifiers can be found by the first hashing. However, for the case where $b = 64$, candidate entries need to be generated 10^5 times to find the first nonempty entry. This is caused by the wide gap between the number of possible candidates, 2^b, which measures the number of entries in the table, and the number of database vectors. This imbalance causes almost all entries to be empty[3].

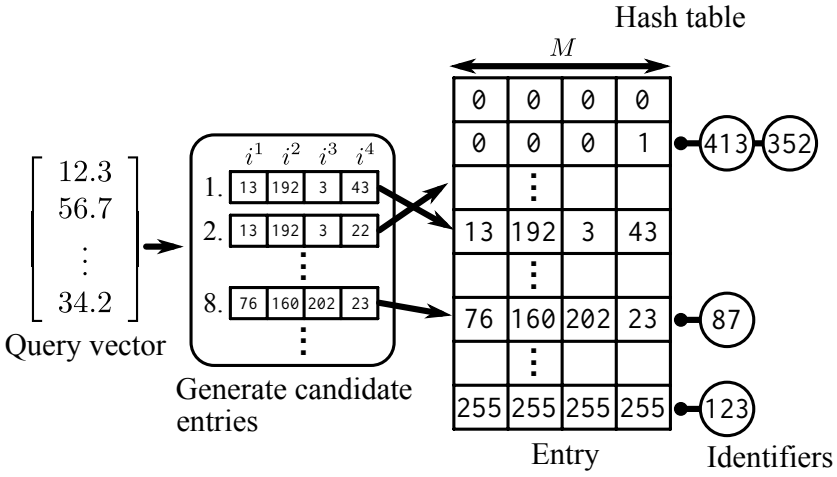
To handle these two issues, we first present a **candidate generation scheme**, which is mathematically equivalent to a multisequence algorithm [5], to solve the empty-entries problem. Then we propose **table division and merging** to handle the long-code problem.

We first show the data structure and a querying algorithm for a single table. To be able to use a single table, we need to handle the empty-entries problem, so we present a candidate-generation scheme to solve it. Then, because a single table cannot handle large codes (the long-code problem), we propose table division and merging to handle codes of any length.

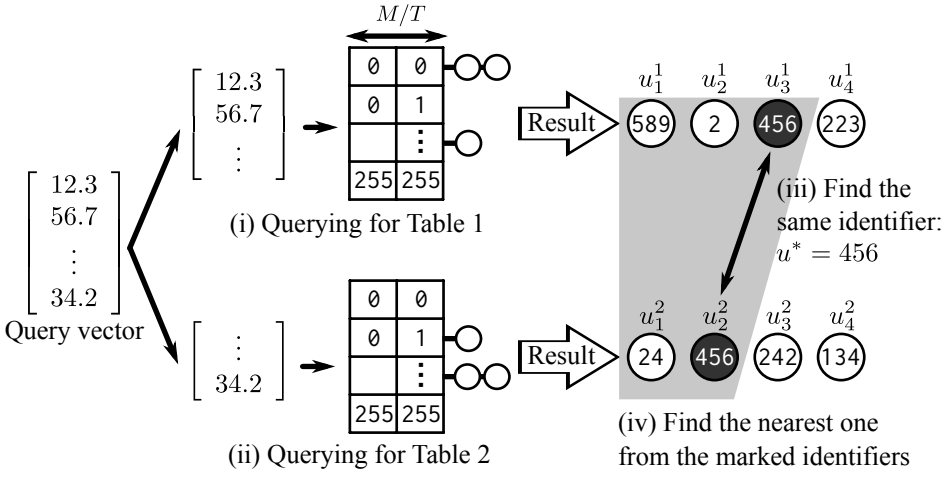
4.3.1 Single Hash Table

The basic idea of the proposed method is to use a hash table, as shown in Figure 4.4a. We prepare a hash table comprising the PQ codes themselves, i.e., a table entry is a tuple of 8-bit integers. During offline processing in advance, the j-th database vector $\mathbf{y}_j$ is compressed by PQ. The resultant PQ code $(i_j^1, \ldots, i_j^M)$ is used directly as an entry

[3]This is a similar problem to that of designing a coarse quantizer for an inverted indexing-based ANN system, but we found an optimal solution empirically, as discussed later.



(a) Single table ($M = 4$). # entries is $(2^8)^M = 2^{32}$



(b) Multitables ($M = 4, T = 2$). # entries for each table is $(2^8)^{M/T} = 2^{16}$

Figure 4.4: Overview of the proposed method.

in the hash table and its identifier j is associated with the entry. If several vectors are associated with the same entry, as shown for "413" and "352" in Figure 4.4a, both of them are stored. Note that the distances from the query to these vectors are the same because they are quantized as the same code. In online processing, an input query vector $\mathbf{x}$ is compressed to $(i^1, \ldots, i^M)$ by PQ, and identifiers associated with the PQ code of the query are searched.

If an identifier is found ($^\exists j, i^m = i^m_j$ for all m), it is obviously the nearest entry in terms of AD because the codeword that is the nearest to the query is $(i^1, \ldots, i^M)$, by definition. A problem arises if there are no identifiers associated with the PQ code of the query, e.g., (13,192,3,43) is the PQ code of a query for which there is no associated identifier in the hash table in Figure 4.4a. This is the empty-entries problem described above. In such cases, we need to find a candidate entry for the PQ code, which will be the second nearest to the query in terms of AD. To find the top K nearest identifiers, the search continues for candidates until K identifiers are found, giving a set of results that are sorted automatically in terms of AD.

Given a query vector, how can candidate entries be generated? We present a candidate-generation scheme (the "Generate candidate entries" box in Figure 4.4a). Given a query vector, it produces candidate codes that are sorted in ascending order of AD. This scheme is mathematically equivalent to the higher-order multisequence algorithm proposed by Babenko and Lempitsky [5]. They used this algorithm originally to divide the space into Cartesian products for coarse quantization. We found that it can be used to enumerate all possible PQ code combinations in ascending order of AD.

We review the multisequence algorithm [5] using our efficient implementation in Algorithm 1. It consists of two steps: initialization (Init) and code generation (NextCode).

Initialization: For the initialization, given a query vector $\mathbf{x}$ and codewords C, $\mathbf{x}$ is divided into subvectors $\mathbf{x}^m$, and coded by corresponding subcodewords C^m. When coding, the distances from the $\mathbf{x}^m$ to each subcodeword $\mathbf{c}^m_k \in C^m$ are recorded in $d\_table[m]$, and sorted. Note d_table is an $M \times K$ 2D array consisting of tuples. Each tuple is composed the ID[4] of the centroids and a distance to the query. Then, the indices of the closest subcodewords are collected and used to construct the nearest code, which is then kept in $cand$. Note $cand$ is a priority queue composed of tuples. Each tuple consists of a code and its distance to the query, sorted by the distance,

[4]To avoid confusion, we use "ID" to represent the ID-th centroids of subquantizers and "identifier" to denote database vectors.

where the distance is automatically computed by looking up *d_table* when pushing[5]. *cand* holds candidates of the final output. The computational cost of the initialization is $O(K(D + \log K))$ for encoding and sorting, which is negligible for a large database.

Code generation: To generate a code, the nearest code to the query is popped from the priority queue, which consists of candidates of the nearest code. For generation of the next-nearest candidates, new candidates of codes are computed as follows. Suppose the popped *code* is constructed by IDs, from $(d\_table[1][k_1], \ldots, d\_table[M][k_M])$. We construct M *new_code*s, where each of them is constructed by IDs from ($d\_table[1][k_1]$, $\ldots$, $d\_table[m][k_m + 1], \ldots, d\_table[M][k_M]$), for $m = 1 \ldots M$. These M *new_code*s are new candidates, and kept in the priority queue, *cand*. The NextCode( ) function is called whenever a new candidate is to be generated.

Pseudocode for an algorithm that queries a single table is shown in Algorithm 2, where $\mathbf{x}$ denotes the input vector, *table* is the PQTable filled with database PQ codes, and K is the user-specified number of output identifiers. The output $u[\,]$ is a sequence of identifiers ordered in terms of AD between the query and the database vectors. *cand* is a priority queue composed of PQ code, and used for the multisequence algorithm. The norm $|\cdot|$ of a sequence means the number of elements of the sequence. Hash($\cdot$) fetches items using the input code as an entry and returns a set of associated identifiers. Insert($\cdot$) simply adds items to the end of the array. We omit *d_table* and returned *cand* from NextCode, for simplicity.

4.3.2 Multiple Hash Table

The single-table version of PQTable may not work when there are long PQ codes, such as $8 \leq M$ (64 bits). This is the long-code problem described above, where the number of possible entries is too large for efficient processing. The number of entries is $(2^8)^M$ for an $8M$ bit code, and the number of database vectors can be at most one billion. Therefore, most of the entries will be empty if M is 8 or greater ($(2^8)^8 = 2^{64} \sim 1.8 \times 10^{19} \gg 10^9$).

To solve this problem, we propose a table division and merging method. The table is divided as shown in Figure 4.4b. If an $8M$-bit code table is divided into two $4M$-bit code tables, the number of the entry decreases, from $(2^8)^M = 256^M$ for one

[5]Note that this priority queue does not contain duplicate codes as a result of checking the code when pushing a new tuple into the priority queue, using another supplemental hash table. This "checking" step is a different implementation to the original method [5], where they required an M-dimensional array to handle this, requiring much more memory. Note that the results of the two algorithms are exactly the same.

Algorithm 1: Multisequence algorithm [5] with our efficient implementation.

Function Init
  Input: $\mathbf{x} = [\mathbf{x}^1, \ldots, \mathbf{x}^M]$ // query vector
    $C = C^1 \times \cdots \times C^M$ // codewords
  Output: *cand* // priority queue
    d_table[ ][ ] // 2D array
  cand $\leftarrow \emptyset$
  d_table[ ][ ] $\leftarrow \emptyset$
  for $m \leftarrow 1$ to M **do**
    for $k \leftarrow 1$ to K **do**
      $\mathbf{c}_k^m \leftarrow k$-th center from C^m *d_table*$[m][k] \leftarrow$ tuple$(k, d(\mathbf{x}^m, \mathbf{c}_k^m)^2)$
    SortByDist(*d_table*$[m]$[ ])
  code $\leftarrow$ Collect IDs from *d_table*$[m][1]$
    for all m and construct code.
  cand.Push(*code*)
Function NextCode
  Input: *cand*, *d_table*[ ][ ]
  Output: *cand*, *code*
  code $\leftarrow$ *cand*.Pop( )
  Suppose *code* is constructed by collecting
    IDs from *d_table*$[m][k_m]$ for all m.
  for $m' \leftarrow 1$ to M **do**
    next_code $\leftarrow$ Collect IDs from
      d_table$[m][k_m + a]$ for all m and construct
      code where $a \leftarrow 1$ if $m = m'$, else 0. *cand*.Push(*next_code*)

table to $(2^8)^{M/2} = 16^M$ for two tables. By properly merging the results from each of the small tables, we can obtain the correct result.

If we have b-bit PQ codes for the database vectors, we prepare T b/T-bit tables. During offline processing in advance, the j-th PQ code is divided by T and inserted in the corresponding table. In online processing, we first compute ranked identifiers from each table in the same way as for the single table case (see Figure 4.4b (i, ii)).

Denote the ranked identifiers from the t-th table as $\{u_n^t\}$, where $t = 1, \ldots, T$, and $n = 1, 2, \ldots$. We can then check the identifiers from the head ($n = 1$) until the same identifier u^* is found from all of the tables, i.e., $u^1_{n_1^*} = u^2_{n_2^*} = \cdots = u^T_{n_T^*} \equiv u^*$, where n_t^* means u^* is found in the n_t^*-th element of the t-th ranked identifiers (see Figure 4.4b (iii)). If the same identifier is found, we pick up all of the checked identifiers, i.e., $u_1^t, \cdots, u^t_{n_t^*-1}$ for all t, referred to as the **marked identifiers** (shaded gray in Figure 4.4b)

Algorithm 2: Querying for a single table.

Input: $\mathbf{x}$, *table*, K
Output: $u[\,]$: ranked identifiers

```
1 u[ ] ← ∅                                              // array
2 cand ← Init(x)                                  // priority queue
3 while |u| < K do
4 |   code ← NextCode(cand)
5 |   {u'} ← table.Hash(code)
6 |   u.Insert({u'})
```

and compute their AD to the query. It is guaranteed that all identifiers whose AD is less than $d_{AD}(\mathbf{x}, \mathbf{y}_{u^*})$ will be included among the marked identifiers (see the next paragraph). Therefore, by simply sorting the marked identifiers in terms of AD, we can find all of the nearest identifiers in the database with AD less than $d_{AD}(\mathbf{x}, \mathbf{y}_{u^*})$ (see Figure 4.4b (iv)). This process is repeated until the desired number of output identifiers is obtained.

We prove that all identifiers where their asymmetric distance (d_{AD}) is less than $d_{AD}(\mathbf{x}, \mathbf{y}_{u^*})$ are included in the marked identifiers. Hereafter, we denote the ADC from the t-th sequence as $d^t_{AD}(\mathbf{x}, \mathbf{y})$, which leads to

$$\sum_{t=1}^{T} d^t_{AD}(\mathbf{x}, \mathbf{y})^2 = d_{AD}(\mathbf{x}, \mathbf{y})^2. \tag{4.4}$$

Here, we introduce a proposition.

Proposition: Given a query vector y, suppose there are sequences of identifiers from each table, where each sequence is sorted in ascending order by its distance to the query (d^t_{AD}). If we find the same identifier u^* from all of the sequences, then any vectors $\mathbf{y}_u$ in the database where $d_{AD}(\mathbf{x}, \mathbf{y}_u) < d_{AD}(\mathbf{x}, \mathbf{y}_{u^*})$ have been already included in the marked identifiers.

Proof: The proposition is proved by contradiction. Suppose there is an identifier $\bar{u}$, where $d_{AD}(\mathbf{x}, \mathbf{y}_{\bar{u}}) < d_{AD}(\mathbf{x}, \mathbf{y}_{u^*})$, and it is not included in the marked identifiers. Because $\bar{u}$ is not included in the marked identifiers, $\bar{u}$ must appear after u^* in each sequence. This leads to $d^t_{AD}(\mathbf{x}, \mathbf{y}_{u^*}) < d^t_{AD}(\mathbf{x}, \mathbf{y}_{\bar{u}})$ for all t, because each sequence is sorted in ascending order. By summing up all t, it leads to $d_{AD}(\mathbf{x}, \mathbf{y}_{u^*}) < d_{AD}(\mathbf{x}, \mathbf{y}_{\bar{u}})$, which contradicts the premise.

Pseudocode for querying multiple PQTables is shown in Algorithm 3. Unlike the single-table case, *tables*[ ] is an array of tables. *u_dist*[ ] and *marked*[ ] are each

Algorithm 3: Querying for multitables.

Input: $\mathbf{x}$, *tables*[ ], K
Output: *u_dist*[ ] : ranked identifiers with distances

```
1  u_dist[ ] ← ∅                                                    // array
2  T ← |tables|
3  cands[ ] ← Split(Init(x), T)                                     // priority queues
4  marked[ ] ← ∅                                                    // array
5  while |u_dist| < K do
6      for t ← 1 to T do
7          code ← NextCode(cands[t])
8          {u^t} ← tables[t].Hash(code)
9          for u ∈ {u^t} do
10             Check(u)
11             if u has been checked T times then
12                 {i'} ← Pick up checked identifiers
13                 for i ∈ {i'} ∪ u do
14                     marked.Insert(tuple(i, d_AD(i)))
15                     UnCheck(i)
16                 PartialSortByDist(marked, K)
17                 u_dist.Insert(marked.Less(d_AD(u)))
```

an array of tuples, comprising an identifier and its AD to the query. *u_dist*[ ] is the final result and *marked*[ ] is used as a buffer. Split$(\cdot, T)$ divides the priority queues into T smaller priority queues. Check$(\cdot)$ and UnCheck$(\cdot)$ are used to record how many times identifiers have appeared in the process. They are implemented by an associative array in our method. Because each identifier is included once in each table, and therefore can be checked at most T times. An identifier that has appeared in all tables can be detected by seeing if it has been checked T times or not. PartialSortByDist$(\cdot, K)$ sorts an array of tuples in terms of distance, obtaining the top K results at a computational cost of $O(N \log K)$, where N is the length of the array. The function *array*.Less(d) returns those elements whose distances are less than d. Note that $d_{AD}(i)$ is an abbreviation of $d_{AD}(\mathbf{x}, \mathbf{y}_i)$.

4.4 Empirical Analysis for Parameter Selection

In this section, we discuss how to determine the value of the parameter required to construct the PQTable. Suppose that we have N PQ-coded database vectors of

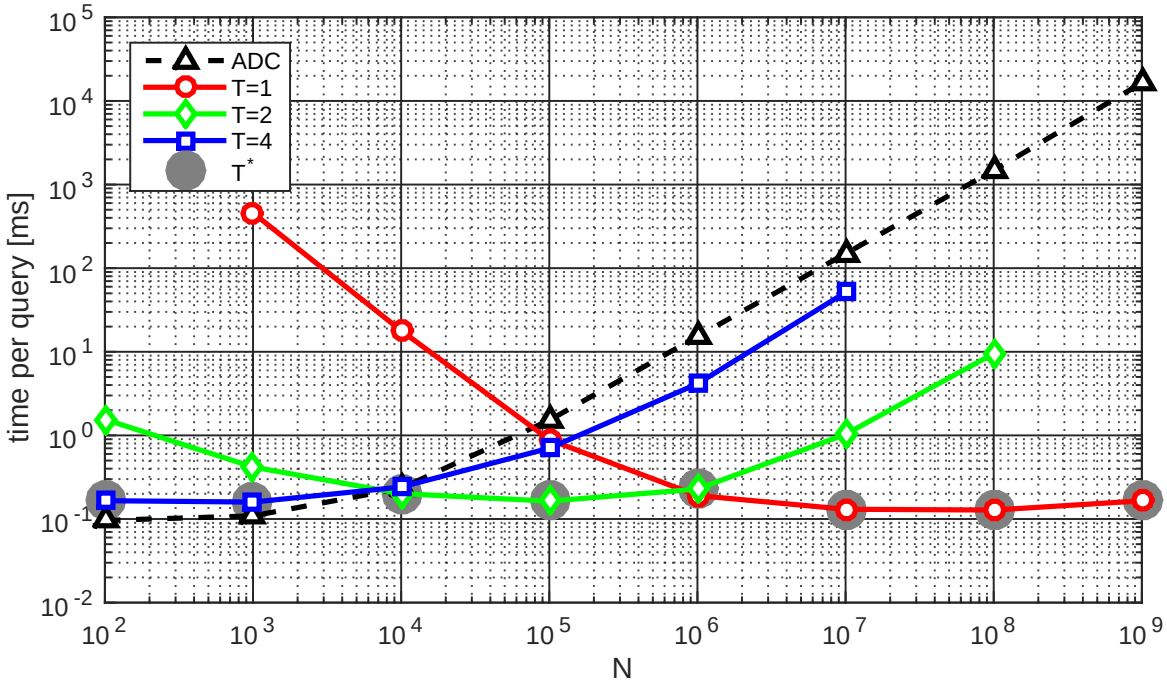
code length b. To construct the proposed PQTable, we need to decide one parameter value T, which is the number of dividing tables. If $b = 32$, for example, we need to select the data structure as being either a single 32-bit table, two 16-bit tables, or four 8-bit tables, corresponding to $T = 1, 2$, and 4, respectively. We first show that the statistics of the assigned vectors are strongly influenced by the distribution of the database vectors, and present an indicative value, $b/\log_2 N$, as proposed by previous work on multitable hashing [39, 83], that estimates the optimal T.

Because the proposed data structure is based on a hash table, there is a strong relationship between N, the number of entry 2^b, and the computational time. If N is small, almost all of the entries in the table will not be associated with identifiers, and generating candidates will take time. If N is appropriate and the entries are well filled, the search is fast. If N is too large compared with the size of the entry, all entries are filled and the number of identifiers associated with each entry is large, which can again cause slow fetching. We show the relationship between N and the computational time for 32-bit codes and $T = 1, 2$, and 4 in Figure 4.5a, and for 64-bit codes and $T = 2, 4$, and 8 in Figure 4.5b. Note that these graphs are log–log plots. For example, the table with $T = 1$ computes 10^5 times faster than a linear ADC scan when $N = 10^9$ in Figure 4.5a. From these figures, each table has a "hot spot." In Figure 4.5a, for $N = 10^2$ to 10^3, 10^4 to 10^5, and 10^6 to 10^9, $T = 4, 2$, and 1 are the fastest, respectively. Given N and b, we need to decide the optimal T without actually constructing tables.

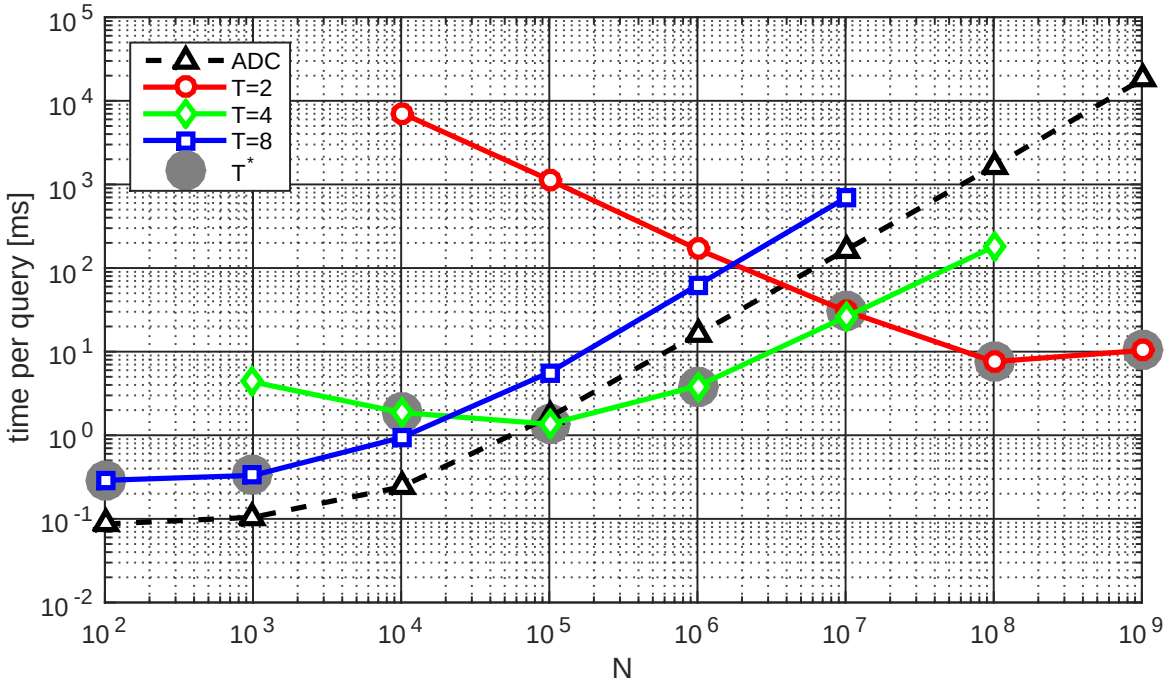
We can show that the statistics in the table are strongly influenced by the distribution of the input vectors. Consider the average number of candidates generated s in finding the nearest neighbor. For the case of a single table, this represents the number of iterations of the **while** loop in Algorithm 2. If all vectors are equally distributed among the entries, we can estimate the average number $\bar{s}$ by a fill factor $p \equiv N/2^b$, which shows how well the entries are filled. Because the probability of finding the identifiers for the first time in the s-th step is $(1-p)^{s-1}p$ when $p < 1$, the expected value of s for finding the item for the first time is

$$\bar{s} = \begin{cases} \sum_{s=1}^{\infty}(1-p)^{s-1}ps = \frac{1}{p} & (\text{if } \ p < 1) \\ 1 & (\text{if } \ p \geq 1). \end{cases} \tag{4.5}$$

This estimated number and the actual number required to find the identifiers, evaluated for the SIFT1B data, are shown in Table 4.1. There is a very large gap between the estimated and actual number (roughly 10 times) for all N. In addition, the table gives the number of assigned items for the entry when the nearest items are found, which corresponds to $|\{u'\}|$ in Algorithm 2 when hashing returns identifiers for



(a) 32-bit PQ codes from the SIFT1B data.



(b) 64-bit PQ codes from the SIFT1B data.

Figure 4.5: Runtime per query of each table.

Table 4.1: Estimated and actual number of steps to find an item for the first time, and actual number of retrieved items for the first entry, for 32-bit codes from the SIFT1B data.

	N	10^4	10^5	10^6	10^7	10^8	10^9
# steps	Actual	2.8×10^4	2.5×10^3	1.7×10^2	25	6.0	1.1
	Estimate($\bar{s}$)	4.3×10^5	4.3×10^4	4.3×10^3	4.3×10^2	43	4.3
# items	Actual	1.1	2.1	12	1.2×10^2	1.2×10^3	1.1×10^4

the first time. This number is surprisingly large when N is very large. For example, for the case of $N = 10^9$, the first candidate usually has associated identifiers (s is actually 1.1), with 1.1×10^4 identifiers being assigned to that entry. This is surprising because, if the identifiers were distributed equally, the estimated step s would be 4.3, and the number of associated identifiers would be less than 1 for each entry. This heavily biased behavior, caused by the distribution of the input vectors, prevented us from theoretically analyzing the relationship between the various parameters. Therefore, we chose to employ an empirical estimation procedure from the previous literature, which was both simple and practical.

The literature on multitable hashing [83, 39] suggests that an indicative number, $b/\log_2 N$, can be used to divide the table. Our method is different from those in previous studies because we are using PQ codes. However, this indicative number can provide a good empirical estimate of the optimal T in our case. Because T is a power of two in the proposed table, we quantize the indicative number into a power of two, and the final optimal T^* is given as

$$T^* = 2^{Q\left(\log_2(b/\log_2 N)\right)}, \tag{4.6}$$

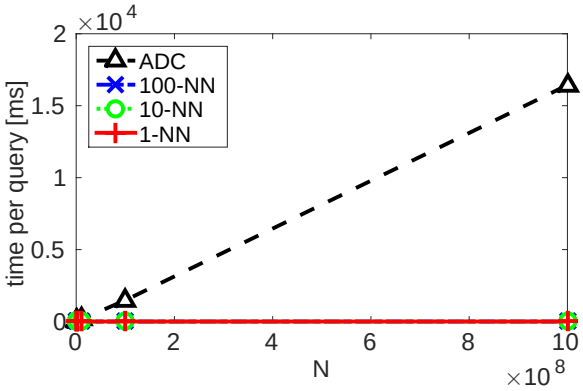
where $Q(\cdot)$ is the rounding operation. A comparison with the actual number is shown in Table 4.2. In many cases, the estimated T^* is a good estimation of the actual number, and the error was small even if the estimation failed, such as in the case of $b = 32$ and $N = 10^6$ (see Figure 4.5a). Selected T^* values are plotted as gray dots in Figure 4.5a and Figure 4.5b.

4.5 Experimental Results

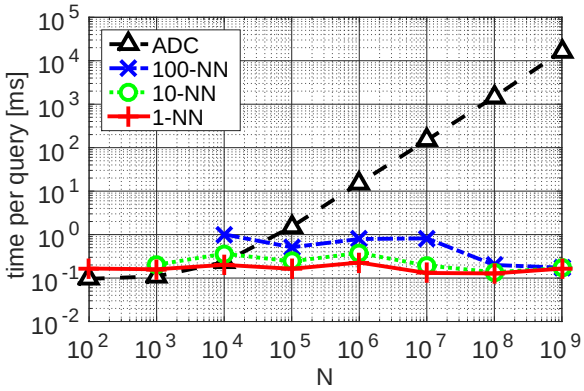
We present our experimental results using the proposed method. The evaluation was performed using the SIFT1B data from the BIGANN dataset [58], which contain one billion 128-D SIFT features and provide learning and querying vectors, from

Table 4.2: Estimated and actual T for the SIFT1B data.

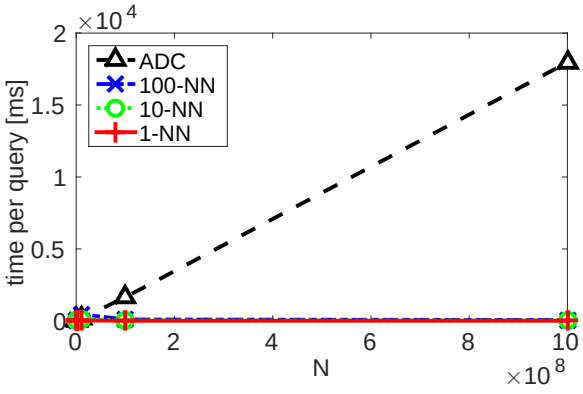
	N	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9
$b = 32$	Actual	4	4	2	2	1	1	1	1
	T^*	4	4	2	2	2	1	1	1
$b = 64$	Actual	8	8	8	4	4	4	2	2
	T^*	8	8	4	4	4	2	2	2



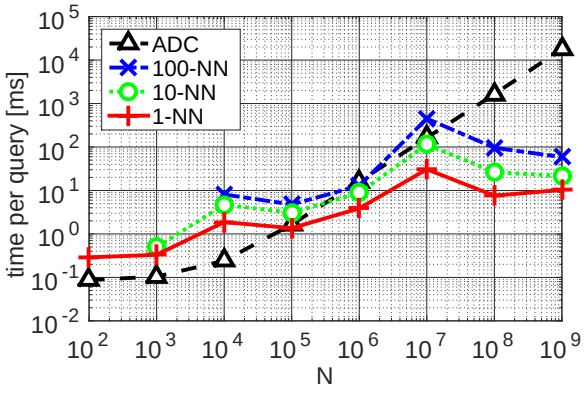
(a) Plot of 32-bit PQ codes.



(b) Log–log plot of 32-bit PQ codes.



(c) Plot of 64-bit PQ codes.



(d) Log–log plot of 64-bit PQ codes.

Figure 4.6: Runtimes per query for the proposed PQTable with 1-, 10-, and 100-NN, and a linear ADC scan (SIFT1B data).

Table 4.3: Speed-up factors for $N = 10^9$.

	Speed-up factors for k-NN vs. ADC			
b	1-NN	10-NN	100-NN	ADC
32	1.0×10^5	9.7×10^4	9.5×10^4	16.4 s
64	1.7×10^3	8.4×10^2	3.1×10^2	18.0 s

which we used 10 million for learning and 1,000 for querying. The centers of the PQ codes were trained using the learning data (this learning process is required for all PQ-based methods), and all database vectors were converted to 32- or 64-bit PQ codes. Note that we selected T using Equation (4.6). All experiments were performed on a notebook PC with a 2.8 GHz Intel Core i7 CPU and 32 GB RAM. Results using another data, GIST1M, are also presented as additional results, which shows the same tendency as for SIFT1B.

Speed: Figure 4.6 shows the runtimes per query for the proposed PQTable and a linear ADC scan. The results for 32-bit codes are presented in Figure 4.6a and Figure 4.6b. The ADC runtime depends linearly on N and required 16.4 s to scan $N = 10^9$ vectors, but the runtime for the PQTable was less than 1 ms for all cases. When the database size N was small, the ADC was faster. Note that the time to encode the query vector was more significant than the retrieval itself for $N \leq 10^2$. The proposed method was faster for the case where $N > 10^5$, and was approximately 10^5 times faster than the ADC for $N = 10^9$, even for the 1-, 10-, and 100-NN cases. Figure 4.6c and Figure 4.6d show the results for 64-bit PQ codes. The speedup was less dramatic than for the 32-bit cases, but was still 10^2 to 10^3 times faster than the ADC, particularly for $N = 10^8$ to $N = 10^9$, where the runtime was 10.3 ms for $N = 10^9$ in the 1-NN case. We report the speedup factors with respect to the linear ADC in Table 4.3.

Accuracy: Accuracy results for the proposed method are: Recall@1=0.001, @10=0.019, and @100=0.071, for 32-bit codes, with Recall@1=0.067, @10=0.249, and @100=0.569, for 64-bit codes, for $N = 10^9$. Note that all values are the same as for the linear ADC scan case.

Memory usage: We show the estimated and actual memory usage of the PQTable in Figure 4.7. For the case of a single table, the theoretical memory usage involves the identifiers (4 bytes for each) in the table and the centroids of the PQ codes, a total of $4N + 4KD$ bytes. For multitable cases, each table needs to hold the identifiers, and the PQ codes are also required for computing d_{AD}. Here, the final memory usage is $(4T + b/8)N + 4KD$ bytes. These are theoretical lower-bound estimates. As a

Table 4.4: A performance comparison for different methods using the SIFT1B data with 64-bit codes. The bracketed values are from [8]. We show recall values involving similar computational times to those for PQTable.

System	#Cell	List len (L)	R@1	R@10	R@100	Time [ms]	Memory [Gb]	Required params	Required training step
PQTable	-	-	0.067	0.249	0.569	10.3	19.8	None	None
IVFADC [56]	2^{13}	8 million	0.104 (0.112)	0.379 (0.343)	0.758 (0.728)	111.6 (155)	(12)	#Cell, L	Coarse quantizer
OMulti-D-OADC-Local [8]	$2^{14} \times 2^{14}$	10000	(0.268)	(0.644)	(0.776)	(6)	(15)	table-order, k, L	Coarse quantizer, local codebook

reference, we also show the cases when PQ codes are lineary stored for the linear ADC scan ($bN/8 + 4KD$ bytes) in Figure 4.7.

For the $N = 10^9$ with $b = 64$ case, the theoretial memory usage is 16 GB, and the actual cost is 19.8 GB. This difference comes from an overhead for the data structure of hash tables. For example, for 32-bit codes in a single table, directly holding 2^{32} entries in an array requires 32 GB of memory, even if all of the elements are empty. This is because a NULL pointer requires 8 bytes with a 64-bit machine. To achieve more efficient data representation, we employed a sparse direct-address table [83] as the data structure, which enabled the storage of 10^9 data points with a small overhead and provided a worst-case runtime of $O(1)$.

When PQ codes are linearly stored, it requires only 8 GB for $N = 10^9$ with $b = 64$. Therefore, we can say there is a trade-off among the proposed PQTable and the linear ADC scan in terms of runtime and memory footprint (18.0 s with 8GB v.s. 10.3 ms with 19.8 GB).

Additional Results using GIST1M data: We show the performance evaluation using GIST1M data from the BIGANN dataset [56]. The GIST1M data contain one million 960-D GIST features in the database and provide vectors for learning and querying, from which we used 500,000 for learning and 1,000 for querying.

Figure 4.8 shows the runtimes per query for the proposed method and the ADC for the GIST1M dataset. We observed the same tendencies as for the SIFT1B case. Because the size of the GIST1M database vectors was only one million, and our method is most efficient for much larger N, the speedup factors were not as high as for the SIFT1B dataset, but they were still always faster than the ADC for all parameter settings with $N = 10^6$.

Note that accuracy of the proposed method is the same as for ADC: Recall@1=0.031, Recall@10=0.065, and Recall@100=0.173, for 32-bit codes, and Recall@1=0.056, Recall@10=0.125, and Recall@100=0.338, for 64-bit codes.

Distribution of each component of the vectors: We investigated how the distribution of vector components affects the search performance, particularly for the

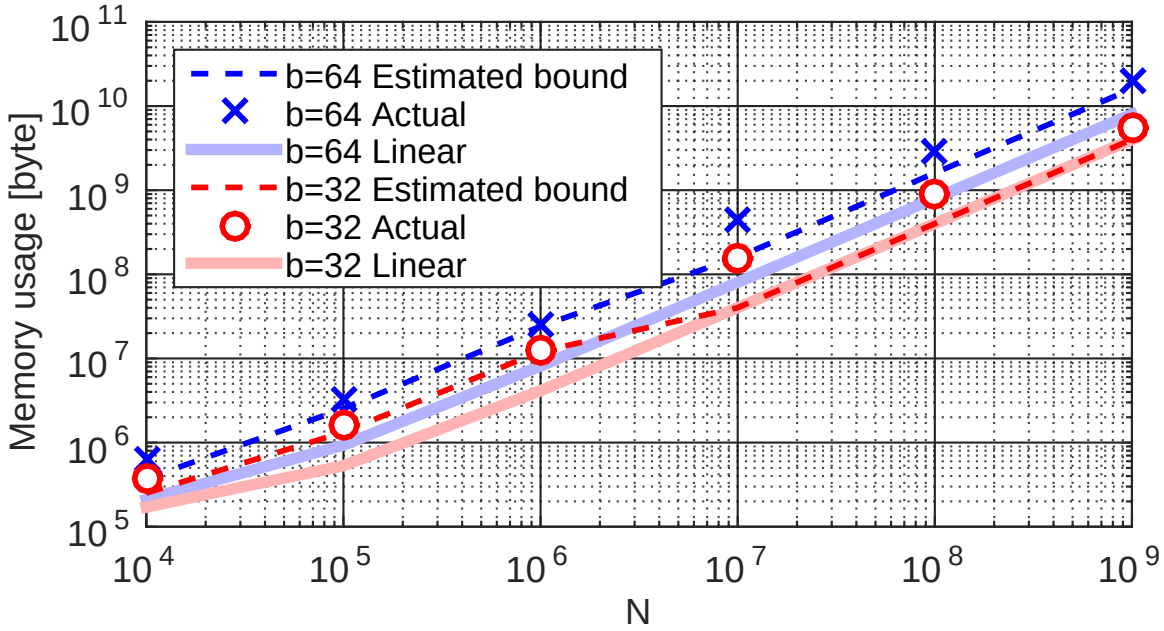
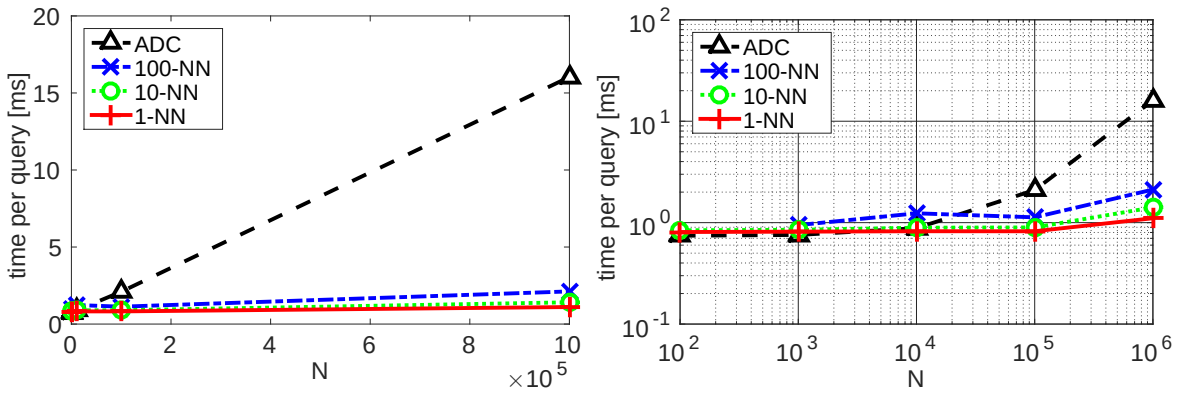


Figure 4.7: Memory usage for the tables using the SIFT1B data. The dashed lines represent the theoretically estimated lower bounds. The circles and boxes represent the actual memory consumption for 32- and 64-bit tables. We also show the linearly stored case for the ADC scan.

multitable case. We prepared the SIFT1M data, using principal-component analysis (PCA), to have the same dimensionality (128). Then both the original SIFT data and the PCA-aligned SIFT data were zero-centered and normalized to enable a fair comparison. Finally, the average and standard deviation for all data were plotted, as shown in Figure 4.9. In both cases, a PQTable with $T = 4$ was constructed and dimensions associated with each table were plotted using the same color. Note that the sum of the standard deviation for each table is shown in the legends.

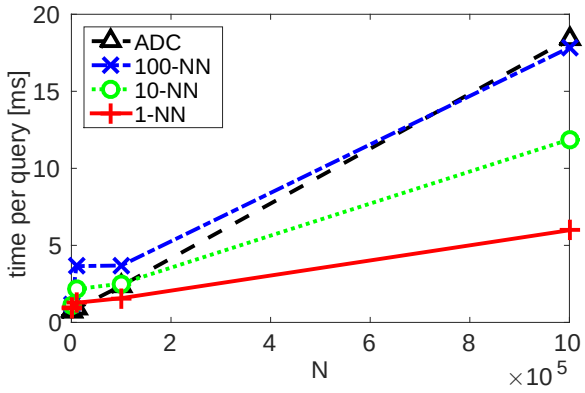
As shown in Figure 4.9a, the values for each dimension are distributed almost equally, which is the best case for our PQTable. On the other hand, Figure 4.9b shows a heavily biased distribution, which is not desirable because elements in Tables 2, 3, and 4 have almost no meaning. In such a situation, however, the search is only two times slower than for the original SIFT. From this, we can say the PQTable remains robust for heavily biased element distributions. Note that the recall value is lower for the PCA-aligned case because PQ is less effective for biased data [56].

Comparative study: A comparison with existing systems for the SIFT1B data with 64-bit codes is shown in Table 4.4. We compared the proposed PQTable with two short-code-based inverted indexing systems, IVFADC [56] and OMulti-D-OADC-Local [7, 8, 62]. IVFADC is the simplest system, and can be regarded as the baseline,

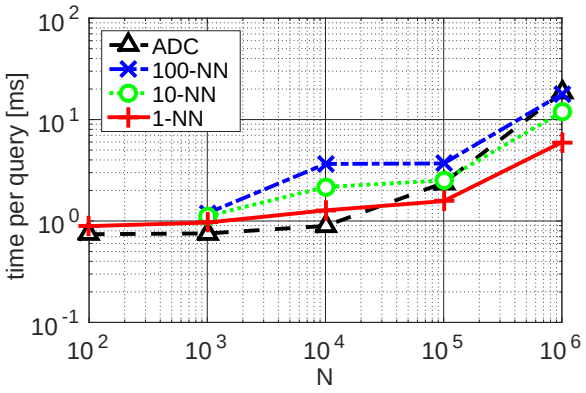


(a) Plot of 32-bit PQ codes.

(b) Log–log plot of 32-bit PQ codes.

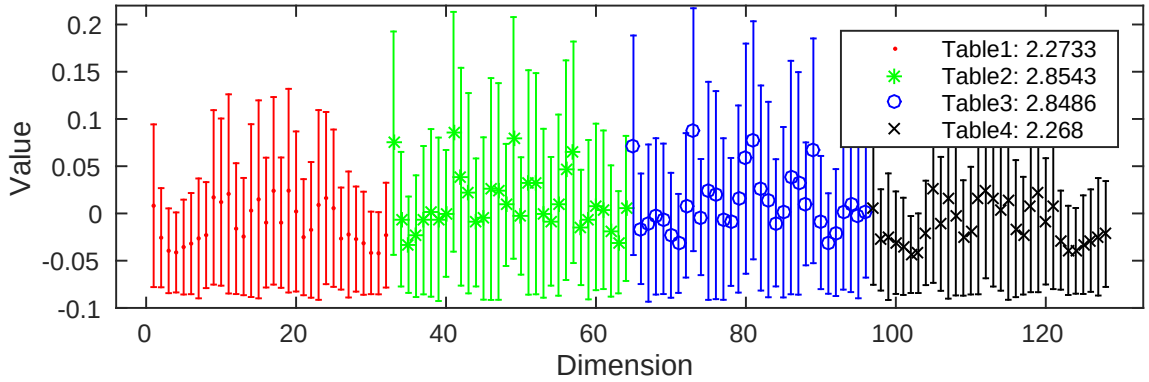


(c) Plot of 64-bit PQ codes.

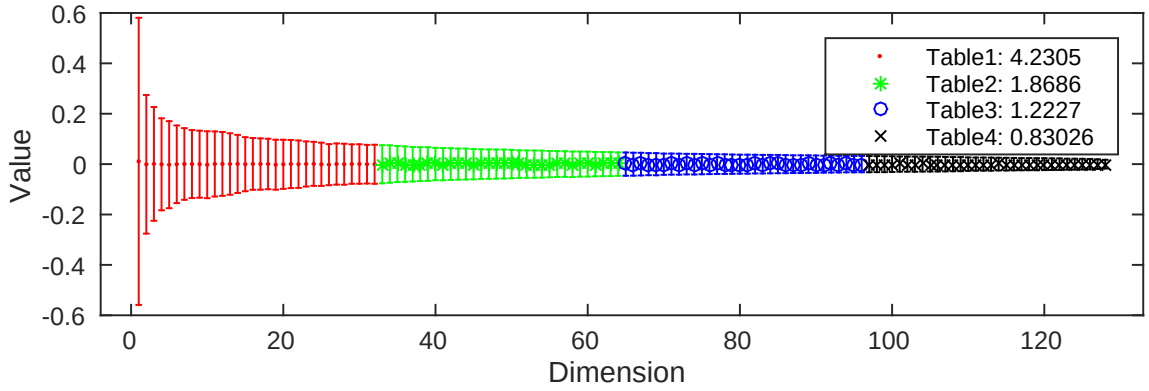


(d) Log–log plot of 64-bit PQ codes.

Figure 4.8: Runtimes per query for the proposed PQTable with 1-, 10-, and 100-NN, and a linear ADC scan (GIST1M dataset).



(a) Original SIFT data. Runtime per query: 6.45 ms. Recall@1=0.25



(b) PCA-aligned SIFT data. Runtime per query: 12.94 ms. Recall@1=0.17

Figure 4.9: The average and standard deviation for the original SIFT vectors and the PCA-aligned vectors.

where the coarse quantizer uses a k-means method. OMulti-D-OADC-Local is a current state-of-the-art system, where the coarse quantizer involves PQ [5], the space for both the coarse quantizer and the short code is optimized [37], and the quantizers are per-cell learned [7, 62]. Note that the T for the PQTable is automatically set to two by using Equation (4.6).

From the table, IVFADC has slightly better accuracy, but is usually slower than PQTable. IVFADC requires two parameters to be tuned, namely the number of space partitions #Cell and the length of the returned list L. The best-performing system, OMulti-D-OADC-Local, can produce superior results. It can achieve better accuracy and memory usage than PQTable. Regarding memory, as discussed in the previous sebsection, the memory overhead of PQTabe is not so small. As for computational cost, both are similar, but OMulti-D-OADC-Local is better with the best parameters. However, it has three parameters to be tuned, namely table-order, k, and L. All have to be tuned manually. In addition, several training steps are required, namely learning the coarse quantizer and constructing local codebooks, both of which are usually time-consuming.

From the comparative study, we can say there are advantages and disadvantages for both the inverted indexing systems and our proposed PQTable. **Static vs. dynamic database:** For a large static database where users have enough time and computational resources for tuning parameters and training quantizers, the previous inverted indexing systems should be used. On the other hand, if the database changes dynamically, the distribution of vectors might vary over time and parameters might need to be updated often. In such cases, the proposed PQTable would be the best choice. **Ease of use:** The inverted indexing systems produce good results but are difficult to handle by novices because they require several tuning and training steps. The proposed PQTable is conceptually simple and much easier to use, because users do not need to decide on any parameters. This would be useful if users were using an ANN method simply as a tool for solving problems in some other domain, such as fast SIFT matching using ANN for 3D reconstruction [23].

Limitations: The PQTable is no longer efficient for $\geq$ 128-bit codes with the SIFT1B data ($T = 4$, $N = 10^9$), taking 3.4 s per query, which is still ten times faster than for the ADC, but slower than for the state-of-the-art system [8]. This is caused by the inefficiency of the "checking" process for longer bit codes. Handling these longer codes is proposed for future work. Note that $\leq$ 128-bit codes are practical for many applications, such as the use of 80-bit codes for state-of-the-art image-retrieval systems [102].

Another limitation is the heavy memory usage of PQTable. Constructing a memory efficient data structure for Hash tables remains a future work.

Chapter 5

Data-driven Application: Drawing Assitance

We proposed the sketch-based manga search framework in Chapter 3, and the theoretical improvement for the core search part in Chapter 4. The next step is to propose a data-driven application using the proposed search schemes. In this chapter, we focus on a drawing support for novices, and propose a data-driven drawing assistance tool.

5.1 Introduction

Imitation and emulation of high-quality artistic work represent essential training components in art education [65]. By observing how established artists draw different strokes, students can learn stroke-based interpretations of real scenes, which provides an entry point to establishing a unique interpretation. Consequently, many art teachers encourage students to emulate existing work and stroke patterns, a process facilitated by the wide availability of both professional and hobbyist artistic content on the Internet.

The aim of this section is to push the traditional imitation and emulation paradigm a step further. In this context, we present *DrawFromDrawings*, an interactive drawing system that provides users with visual feedback for assistance in 2D drawing using a database of sketch images. Whenever the user feels unsatisfied with a sketch, an unsatisfactory region can be marked alongside an associated region of interest (ROI) in a reference sketch from the database (Figures 5.1(a) and (b)). Using a marker position in a specially designed assistance palette, the user can explore new strokes to

(a) (b) (c) (d) (e)

Figure 5.1: Given stroke correspondences between (a) reference and (b) user sketches, (c) DrawFromDrawings interpolates the user strokes based on the original user strokes, the reference strokes, and strokes created by warping the reference strokes based on the shapes of the ROI, according to a marker on the assistance palette. Interpolated strokes are displayed interactively as either deformation or (d) suggestive feedback to stimulate the user's imagination in producing (e) their own drawing.

blend with their original user strokes and the selected reference strokes (Figure 5.1(c)). The user can integrate interpolated strokes as either deformation feedback, whereby user strokes are directly replaced by interpolated strokes to a user-specified degree, or suggestive feedback, which involves overlaying interpolated strokes as a suggestive template.[1] Using a set of simple user interactions, the feeling of the drawing can be retained even after iterated periods of assistance (Figure 5.1(d)), with the sketch evolving naturally towards some satisfactory convergence (Figure 5.1(e)).

The major technical challenge is providing efficient and intuitive interpolation for visual feedback. First, we need to handle any significant structural differences between user strokes and specified reference strokes, such as differences in length, curvature, and stroke count, as illustrated in Figures 5.1(a) and 5.1(b). Second, user interactions should be simple and intuitive for casual drawers. We need to avoid complicated user interactions such as specifying or refining point/stroke correspondences one-by-one, which might be acceptable for professional artists [11, 120]. Third, we want to be able to refer to rasterized sketch images where much of the stroke information is lost. A system with this capability will benefit significantly from the vast quantity of sketches available on the Internet. Last but not least, all features must be executable at an interactive speed to enable interactive feedback with the user.

Our technical contribution lies in solving these issues by automatically detecting correspondences of as-long-as-possible (ALAP) stroke segments from strokes within

[1] In this section, we refer to computing stroke shape changes as *interpolation*, and refer to replacing user strokes with computed stroke changes in the GUI as *deformation*.

the ROI perimeters. Sketch images in the database are preprocessed to represent rasterized strokes as a collection of single curves without branches. ALAP segments and their many-to-many correspondences are then detected by consideration of the geodesic distance along strokes. We also parameterize the ALAP segment interpolation based on the original user strokes, the reference strokes, and the novel strokes created by warping the reference strokes based on the shapes of the user and reference ROIs. These algorithms yield new, visually reasonable strokes at a rate that enables users to seek possible new strokes easily and interactively. The user study on utility of the interpolation indicated that participants could fully utilize the interpolation to handle sketches varying significantly in shape and complexity.

Another key issue we want to investigate in this section is whether or not the deformation of user strokes helps with drawing. Unlike recent work on corrective feedback [77, 72, 131, 123], which applies small, possibly unnoticeable, deformations to user strokes for beautification purposes, it is possible that aggressive deformations will overwhelm a user's own strokes at the expense of creativity. On the other hand, recent work on suggestive interfaces [30, 70, 48] argues that suggestions may nonetheless provide the user with the opportunity to draw a sketch effectively with their own strokes. Consequently, in addition to the deformation feedback mode, we implemented a suggestive feedback mode in which the user draws and refines strokes on their own with interpolated strokes providing guidance. Our user study on the efficacy and usage of these feedback modes indicated that suggestive feedback enabled experienced drawers to develop and render their ideas with their own style and deformation feedback enabled inexperienced drawers to finish sketch compositions quickly.

5.2 Prior Work

In this section, we describe prior work on drawing assistance interfaces and deformation techniques for 2D shapes/lines.

5.2.1 Drawing Assistance Interfaces

Various drawing assistance systems have been explored in the past, many of which can be categorized as *stroke correction* or *stroke suggestion*. The stroke correction approaches are based on interpreting user strokes to better match the strokes to the users' intent. The systems by Igarashi and colleagues [50] and by Arvo and

Novins [4] automatically interpret and replace freehand strokes with geometric primitives. Barla and colleagues [10] presented a method to simplify input sketches while maintaining the morphological structure of the original sketches by clustering strokes and replacing each cluster with simpler strokes. HelpingHand [77] utilizes a stroke database constructed by trained artists to improve input stroke trajectories, in addition to applying styles from the database to the input strokes. Limpaecher and colleagues [72] introduced a correction vector field learned via a crowdsourcing game to automatically fit user strokes to target face images. The system developed by Zitnick [131] improves the latest strokes by averaging them with similar ones in the database. PortraitSketch [123] automatically adjusts the outline strokes, based on gradients and landmarks. EZ-sketching [103] automatically adjusts a drawing by considering not only local structures but also a global relationship between strokes. Whereas the goal of the stroke correction approaches is to beautify freehand drawings, our interface interpolates user and reference strokes that may vary significantly to help users explore novel strokes for better drawing.

A suggestive interface provides feedback to the users, with the choice of referring to the suggestions or ignoring them being up to the users. This approach has been adopted for various graphic applications such as 3D modeling [49, 111], physically based animation [68], and recently 2D drawing [30, 70, 48]. iCanDraw [30] analyzes relationships among facial features such as eyes and suggests corrective feedback to users based on the extracted features and current user strokes. ShadowDraw [70] overlays edges detected from multiple real photos under the current user strokes. Drawing Assistant [48] uses various computer vision techniques to extract features such as its rough shape and a skeleton from a target real reference photo. These features are suggested to users so they can learn which features they should focus on when drawing. Whereas previous work reveals that corrective feedback can improve users' drawings significantly, we want to investigate, via a user study, when suggestive feedback is to be preferred over deformation feedback and vice versa.

5.2.2 2D Shape Interpolation Techniques

"Inbetweening" is a process for creating intermediate animations between two 2D keyframes. Unlike 3D interpolation, which can handle occlusion problems easily, inbetweening needs to parse 2D drawings to handle such 3D-related aspects in the resulting 2D animations. Among extensive studies on inbetweening, Baxter and colleagues [12] introduced an approach to detect features based on stroke sharpness for shape matching. Liu and colleagues [73] detected correspondences between

distinctive points, and obtain stroke correspondences through manifold learning. BetweenIt, from Whited and colleagues [120], combines stroke motion computed from a logarithmic spiral and stroke deformation from curvature averaging and twisting warps. Stroke correspondences are detected by traversing a graph in which an edge is a stroke and a node is a junction of strokes. Assuming that correspondences are given, as-rigid-as-possible (ARAP) shape interpolation, originating from Alexa and colleagues [1], has been noted as creating animations that look physically reasonable [51, 96, 107]. These inbetweening and ARAP interpolation approaches can achieve natural 2D animations in both space and time. In contrast, however, the DrawFromDrawings system is intended to be an assistive interface for single 2D sketches. Therefore, we develop correspondence detection and interpolation algorithms where simple user interaction can produce a novel variety of strokes from user and reference strokes that differ significantly in their structure, rather than smooth, natural animations produced from structurally similar strokes. It should also be noted that these existing works assume that strokes are vectorized, whereas our method can interactively handle rasterized strokes.

More relevant to our work is the latent doodle space [11]. Given multiple sketch images, this approach detects stroke correspondences similarly to K-means clustering and builds a latent space based on dimension reduction techniques. Navigation in the latent space yields a novel sketch image. Whereas this method assumes that the user draws multiple similar sketches for input, we aim at an assistive interface for novice and hobbyist drawers and do not therefore assume that their drawings are necessarily similar to database sketches.

5.3 GUI and Overall Workflow

When using the DrawFromDrawings system, the user repeats three steps, namely *drawing and retrieval*, *exploration*, and *refinement*, until satisfactory convergence occurs.

Drawing and Retrieval: The user starts by drawing in the *user canvas* (Figure 5.2(a)). The user can draw and erase strokes freely, or clear the user canvas if necessary. As the user draws strokes, several similar sketch images are automatically retrieved, sorted, and displayed in the *search canvas* (Figure 5.2(d)) based on the stroke similarity described in Section 5.4. If the user observes a favorite sketch in the search canvas, the user can select it for display in the *reference canvas* (Figure 5.2(b)). The reference sketch can be rotated and/or flipped to adjust the orientation of the reference sketch to the user sketch if necessary (Figure 5.2(e)).

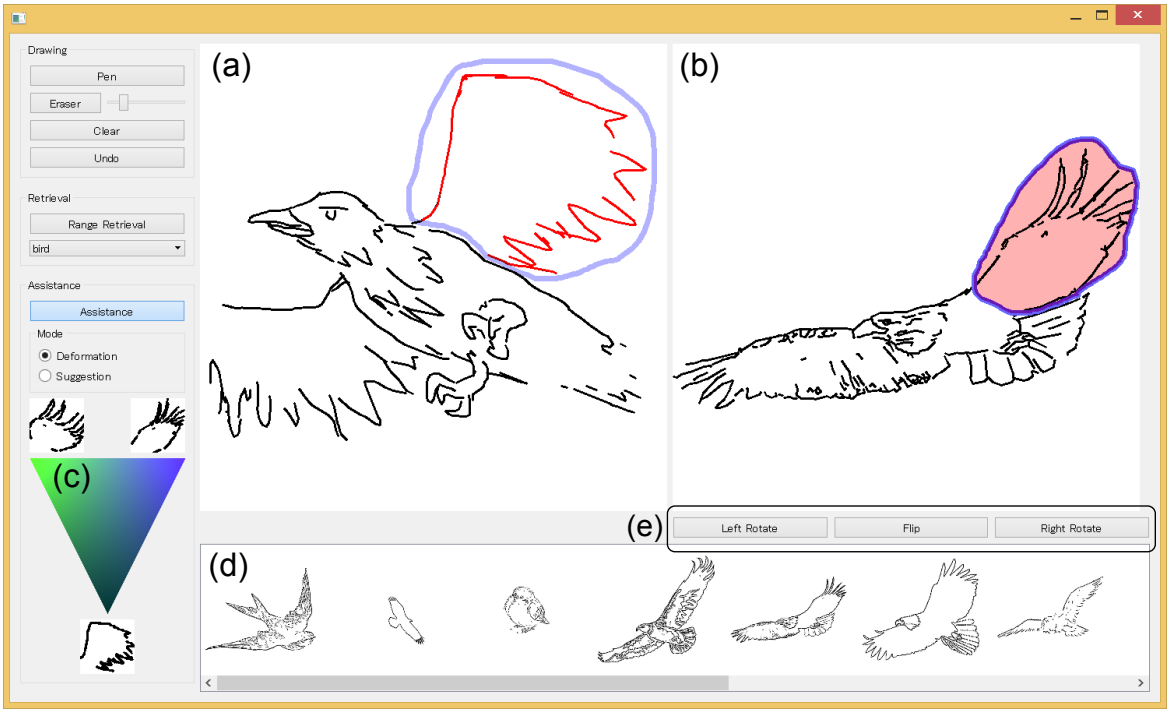


Figure 5.2: The DrawFromDrawings interface: (a) User canvas, (b) reference canvas, (c) assistance palette, (d) search canvas, and (e) buttons to rotate and flip the reference sketch.

Exploration: Whenever the user becomes dissatisfied with their drawings, the user can invoke the assistance function with a reference sketch. This starts by clicking the *Assistance* button and drawing an ROI perimeter that outlines approximately the objects/components of interest in each of the user and reference sketches (the blue curves in Figures 5.2(a) and (b)). The pink blob inside the perimeter is set as the ROI, and all strokes inside the ROI are considered for assistance. The color of the selected user strokes is changed to red.

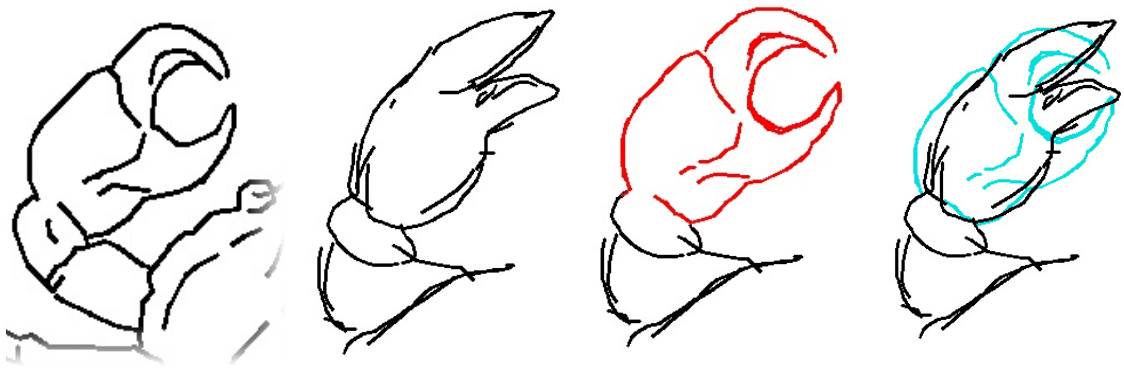
Having identified the ROIs in the user and reference sketches, the assistance process starts by dragging a red marker in the *assistance palette* (Figure 5.2(c)) to explore novel strokes, created with the algorithm described in Section 5.5. The DrawFromDrawings system interactively interpolates the user and reference strokes according to a marker position, where the vertical position of the marker selects the originality of the user strokes such as stroke style and shape, and the horizontal position selects the proportion of the outline shape of the user strokes to be considered. Interpolated strokes are rendered around the assistance palette so that the users can observe the approximate relationship between the marker position and the shape of interpolated strokes.

Refinement: For stroke refinement, the user can utilize interpolated strokes via either *deformation* or *suggestion* feedback. The only difference between these two modes is that deformation feedback actually alters the selected user strokes (Figure 5.3(c)), whereas suggestive feedback overlays the interpolated strokes on the user strokes as guidance for the user (Figure 5.3(d)). If the user decides to follow the guidance, the user deletes original strokes and draw based on the guidance. If not, the user can ignore the guidance and continue drawing.

The choice between deformation or suggestion feedback depends upon the user's preference. Notice that the interpolation algorithm is the same in both feedback modes. If the deformed/suggested strokes do not align well with existing user strokes, the position of the interpolated strokes can be adjusted manually.

5.4 Sketch Database

We begin by collecting sketch images from image search engines on the Internet using general category names (e.g., bird, dog, car) together with "sketch" or "line drawing." The sketch images are stored along with the specified names as meta-information in the database. We preprocess the raw sketch images by resizing them so that their longer sides become 480 pixels and binarizing them via adaptive



(a) (b) (c) (d)

Figure 5.3: Feedback visualization: (a) Reference strokes, (b) user strokes, (c) new strokes (red) created via deformation feedback, and (d) suggestive strokes (blue) overlaying the user strokes.

thresholding. Because sketch images in our current database are rasterized, we apply a line-thinning operation [127] to obtain stroke maps. Strokes shorter than 10 pixels are removed. As the user draws a stroke, a database search runs with the current drawing as the query. For this, we employ the sketch-based image retrieval approach presented by Eitz and colleagues [33], which is based on histogram of oriented gradients. This step efficiently retrieves sketches roughly similar to user input from the database. We then apply the fast directional chamfer matching (FDCM) method [74] to re-rank the top 10 retrieval results. FDCM calculates point-to-line distance, and is therefore suitable for accurate re-ranking with shape-similarity order. Note that FDCM measures the relation between two strokes in a pointwise manner, while Eitz's method compares histograms of sampled edge orientations. FDCM is a more accurate distance measure than Eitz's one, but slower. Therefore, we first search candidates by Eitz's method, then re-rank them using FDCM. The user can obtain retrieval results either from the entire database or from a subset of the database by specifying a category name from the meta-information list.

Optionally, the user can use region-based retrieval. Given a region the user is interested in for retrieval (e.g., a wing of a bird), image features are computed solely from strokes in the selected region and used as the query. The FDCM-based ranking will then consider only strokes from inside the region.

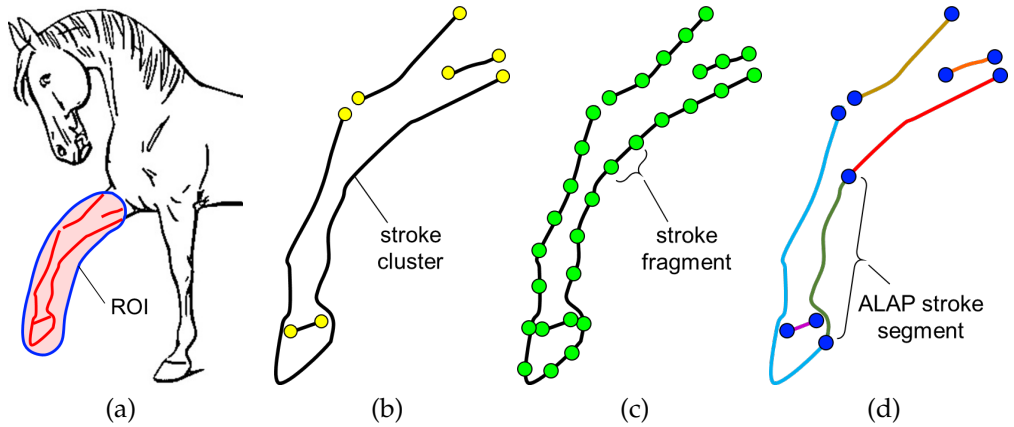


Figure 5.4: Our stroke representations: (a) ROI in a reference sketch (blue) and a set of selected strokes (red), (b) stroke clusters, (c) stroke fragments, and (d) ALAP stroke segments.

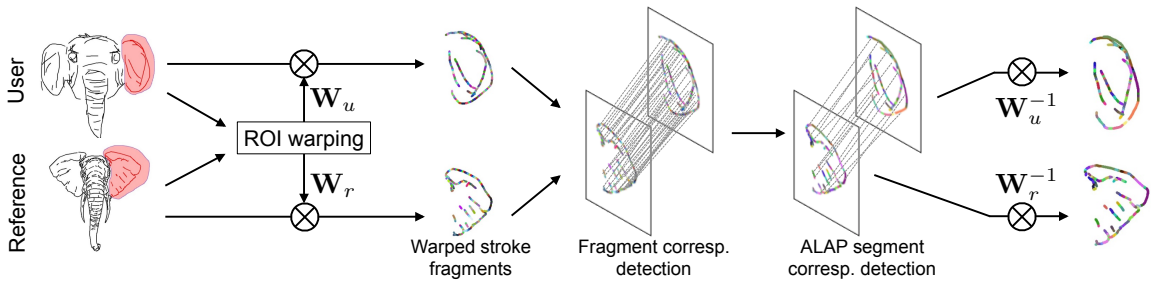


Figure 5.5: User and reference ROIs are aligned with TPS, and selected strokes are warped accordingly. Correspondences between stroke fragments are detected by formulating the detection as a linear assignment problem. ALAP stroke segments and their correspondences are obtained by considering the fragment correspondences and stroke continuity in each stroke cluster.

5.5 Interpolating User Strokes

Our interpolation algorithm involves the detection of correspondences between selected user and reference strokes and the interpolation of the corresponding stroke segments. The correspondence detection and interpolation are quite challenging in this case because strokes in the user and reference ROIs often differ significantly from each other with respect to position, orientation, length, shape, and stroke count. Therefore, we consider the perimeters of the user and reference ROIs as a cue for stroke correspondence detection and interpolation.

5.5.1 Detecting Stroke Correspondence

Given that an ROI is specified to outline the target strokes approximately (Figure 5.4a), we can utilize the perimeter of the ROI to guide the detection of stroke correspondence. The goal here is to divide reference and user strokes into *ALAP stroke segments* (Figure 5.4d) and to detect the correspondence between strokes in the user and reference sketches. Figure 5.5 gives an overview of the detection of ALAP stroke segments.

Stroke Representation and Preprocessing: From selected strokes in a rasterized reference image, we first detect line segments using line fitting with RANSAC [36] in a sequential manner. A line segment is then considered as connected to another if their endpoints are sufficiently close (< 3 pixels), and they are merged into a *stroke cluster* (Figure 5.4b). Note that each stroke cluster is a single curve, which therefore enables the simple linear interpolation described later. The stroke clusters are resampled at constant geodesic intervals (5 pixels in our implementation), which yields strokes between the sampling points as stroke fragments (Figure 5.4c). Using the ALAP segment detection described below, each stroke cluster is resegmented into ALAP stroke segments (Figure 5.4d).

For the user sketch, we consider a single user stroke as a stroke cluster, and apply the same stroke fragmentation and ALAP segment detection.

Aligning Stroke Fragments via ROI Warping: The user and reference ROIs may differ significantly and the ROIs therefore need to be aligned for initialization of the following correspondence detection. Figure 5.6 illustrates the alignment of the ROIs.

We first compute a convex hull for each ROI and the center of mass (COM) of the convex hulls. We project the COM vertically to the upper direction to obtain the point that crosses the convex hull. We resample each convex hull from the crossing point in the clockwise direction with the same point count, and list the

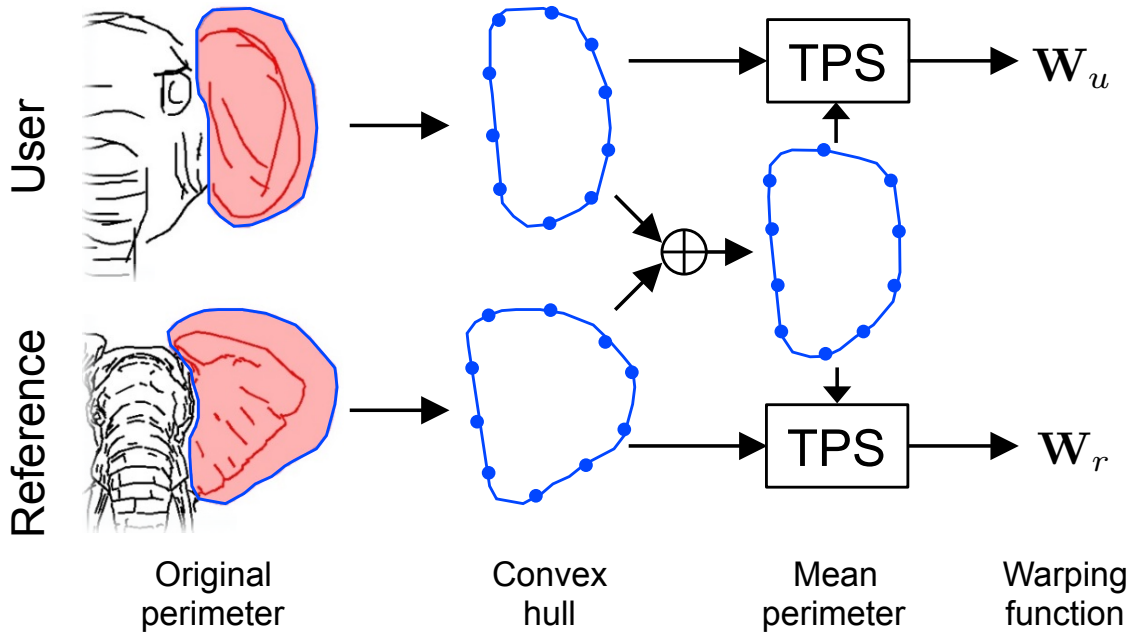


Figure 5.6: Given the convex hulls of user and reference ROIs, the mean perimeter is obtained, and warping functions from the user/reference convex hulls to the mean perimeter are computed using TPS to align user and reference strokes.

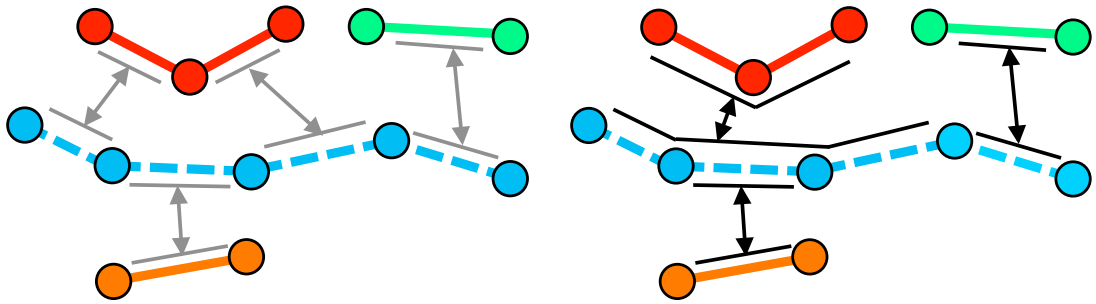
correspondences between each resampling point on the user and reference convex hulls. These correspondences enable computation of a *mean perimeter* by simply taking the mean position of each corresponding point pair. Despite recent work on correspondence detection for such simple perimeters (e.g., [13]), we found that such methods often fail, particularly for simple perimeters such as circular shapes.

The set of corresponding point pairs enables the estimation of warping functions $\mathbf{W}_u$ and $\mathbf{W}_r$ that will align the user and reference convex hulls, respectively, to the mean perimeter. We use the thin-plate splines (TPS) technique [15] to compute each warping function by minimizing the following energy function:

$$\iint \left(\left(\frac{\partial^2 \mathbf{W}}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 \mathbf{W}}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 \mathbf{W}}{\partial y^2} \right)^2 \right) dx dy, \tag{5.1}$$

where the point pairs serve as constraints on the minimization. After $\mathbf{W}_u$ and $\mathbf{W}_r$ are computed, we warp all stroke fragments in the user and reference ROIs.

Detecting Fragment Correspondences: We solve the fragment correspondence detection problem as a linear assignment problem for a bipartite graph. This can be considered as a globally optimal neighbor search for all stroke fragments. Defining the sets of warped stroke fragments in the user and reference ROIs as $\mathcal{F}^u$ and $\mathcal{F}^r$,



(a) (b)

Figure 5.7: (a) Fragment correspondences and (b) ALAP segment correspondences. Dashed and solid lines indicate user and reference strokes, respectively. Gray lines in (a) and black lines in (b) represent fragment and ALAP segment correspondences, respectively.

respectively, a bipartite graph is represented as $\mathcal{G} = \{\mathcal{F}^u, \mathcal{F}^r, \mathcal{E}\}$, where $\mathcal{E}$ represents a set of graph edges connecting user and reference fragments. We compute a weight d for an edge connecting user and reference fragments, $f^u \in \mathcal{F}^u$ and $f^r \in \mathcal{F}^r$, as

$$d(f^u, f^r) = ||\mathbf{m}^u - \mathbf{m}^r||^2, \tag{5.2}$$

where $\mathbf{m}$ represents the COM position of f in the warped space. We solve this linear assignment problem using the Jonker–Volgenant algorithm [61]. For the case where $|\mathcal{F}^u| \neq |\mathcal{F}^r|$, we add dummy nodes to equalize the node counts. The edge weight for dummy nodes is set to a large constant and fragments that do not find globally optimal neighbors are assigned to the dummy nodes [13]. Such fragments are then reassigned to the nearest nodes in the warped space using Equation (5.2).

Detecting ALAP Segment Correspondences: The interpolation of strokes should appear smooth, but fragment-wise interpolation can often appear cluttered. By considering that each stroke cluster contains connections of associated fragments, we can collect cluster pairs for which at least one fragment correspondence is detected. Note that cluster pairs can involve many-to-many correspondences, as illustrated in Figure 5.7a where the user stroke cluster (light-blue dashed line) corresponds to three reference clusters (red, orange, and green solid lines).

Having obtained cluster pairs based on fragment correspondences, we detect a pair of ALAP stroke segments for each stroke cluster pair (i.e., one segment per cluster). Let (C^u, C^r) be a stroke cluster pair associated with fragment correspondences $\{(f^{u\prime}, f^{r\prime})\}$. We now focus on only the user strokes for simplicity, collecting all possible pairs of user fragments from the user fragment set $\{f^{u\prime}\}$ and finding the pair that has

the maximum geodesic distance along C^u:

$$\left(\hat{f}_i^{u\prime}, \hat{f}_j^{u\prime}\right) = \underset{f_i^{u\prime}, f_j^{u\prime}}{\arg\max}\, d_g\left(f_i^{u\prime}, f_j^{u\prime}\right), \tag{5.3}$$

where $d_g(\cdot)$ computes the geodesic distance along a cluster of target fragments. We then connect all fragments between $\hat{f}_i^{u\prime}$ and $\hat{f}_j^{u\prime}$ along C^u and use this as a user ALAP segment $\mathcal{S}^u$. Similarly, we detect a reference ALAP segment $\mathcal{S}^r$ from $\{f^{r\prime}\}$ in C^r, and associate $\mathcal{S}^r$ with $\mathcal{S}^u$ as an ALAP segment correspondence. We apply this detection of ALAP segments and their correspondences to all pairs of stroke clusters (Figure 5.7b).

The ALAP segment correspondences can yield a better interpolation than can fragment correspondences and cluster correspondences. Interpolation using fragment correspondences often causes separation of long strokes and results in discontinuous stroke shape changes (Figure 5.8). Interpolation using cluster correspondences can cause shrinking and stretching, and any difference in the number of clusters for user and reference strokes is difficult to handle (Figure 5.9). The many-to-many correspondences of ALAP segments can overcome these issues by producing interpolations that are reasonable for the exploration of various stroke configurations.

5.5.2 Interpolating User Strokes with Reference Strokes

With the ALAP stroke segments and their correspondences having been detected, the final step is to interpolate strokes between the user and reference sketches. To create a variety of strokes that can stimulate the user's imagination, we consider the following two types of target strokes, namely the selected reference strokes and strokes created by warping the reference strokes based on the ROI perimeters of the user and reference strokes. These choices are based on a "WYSIWYG" strategy, where we choose the first type of target stroke because the users will clearly be interested in it, and choose the second type because reference strokes whose perimeter resembles the user ROI perimeter might be intuitively appealing.

For the first type of target stroke, a similarity transformation $\mathbf{T}_r^u$ is needed, which adjusts the position and scale of the reference ROI to the user ROI. This similarity transformation can be computed simply by aligning the COM positions of both ROIs and calculating the scale based on the size of the ROIs (Figure 5.10d). For the second type of target stroke, we compute a warping function $\mathbf{W}_r^u$ by aligning the convex hull of the reference ROI to that of the user ROI using TPS (Figure 5.10c).

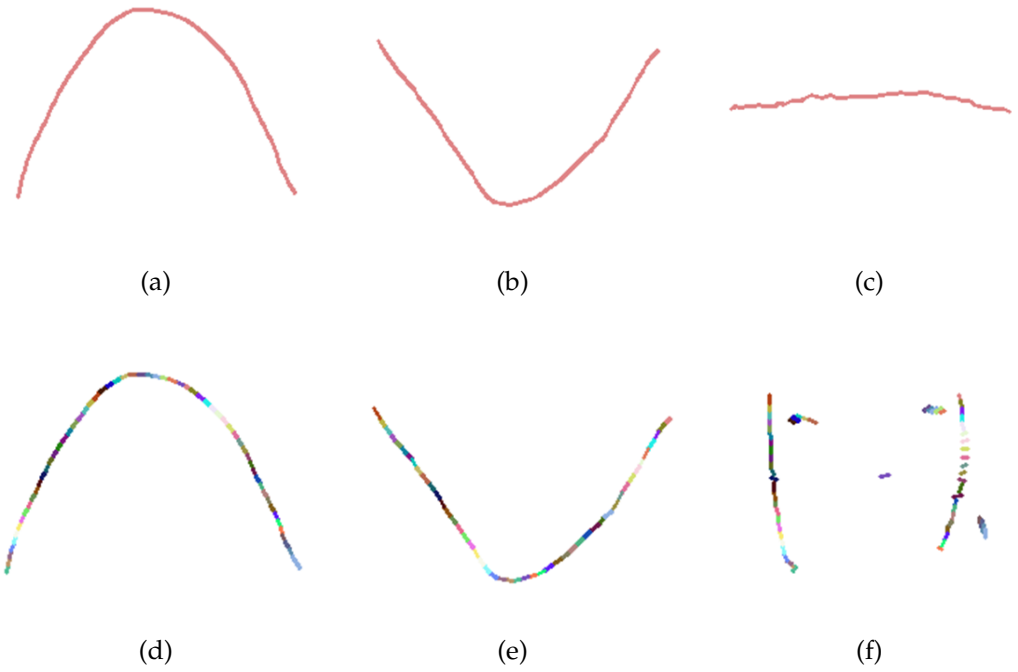


Figure 5.8: Given (a, b) two stroke clusters, the ALAP segment correspondence enables (c) reasonable stroke shapes with the proposed interpolation ($\alpha = 0.5, \beta = 0.5$). In contrast, if the clusters contain (d, e) many short fragments (represented by colors), interpolating correspondence fragments with the same algorithm results in (f) discontinuous stroke shapes.

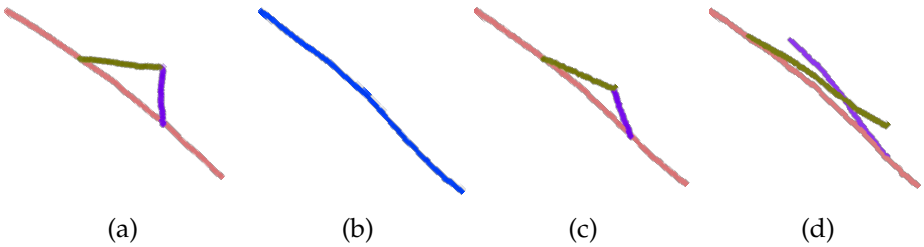


Figure 5.9: Given (a, b) two stroke clusters, the ALAP segment correspondences retain (c) the triangle shape with the proposed interpolation ($\alpha = 0.5, \beta = 0.5$), whereas the cluster correspondences result in (d) strokes that intersect each other.

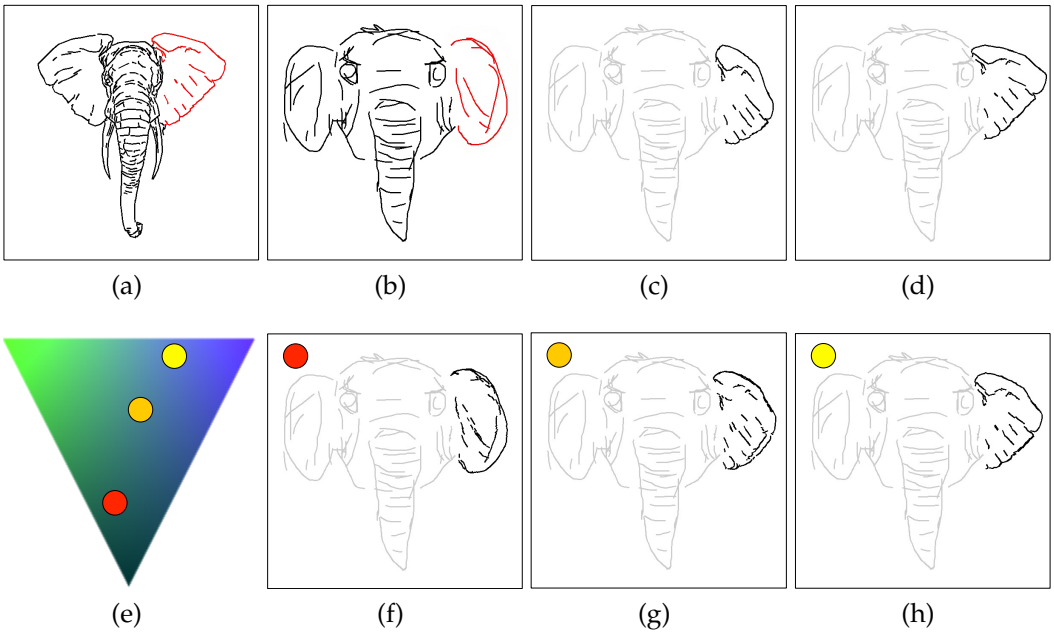


Figure 5.10: Given convex hulls and selected strokes (red) in (a) a reference sketch and (b) a user sketch, (c) reference strokes whose outline shape are matched to that of the user strokes and (d) reference strokes with an adjusted position and scale are computed. The user moves a marker on (e) the assistance palette to create (f, g, and h) interpolated strokes.

Interpolation of strokes is a linear blending of the above two types of target stroke and the user strokes. We resample the corresponding user and reference ALAP segments and compute an interpolated position for each resampling point $\mathbf{q}$ as:

$$\mathbf{q} = (1-\beta)\,\mathbf{p}^u + \beta\left((1-\alpha)\,\mathbf{W}_r^u(\mathbf{p}^r) + \alpha\mathbf{T}_r^u(\mathbf{p}^r)\right), \tag{5.4}$$

where $\mathbf{p}^u$ and $\mathbf{p}^r$ are corresponding resampling points in the user and reference ALAP segments, respectively. α is a parameter indicating the degree to which the outline shape of resulting strokes resembles that of the user strokes and β is a parameter indicating the degree of preservation of the originality in the user strokes.

As the user moves a marker on the assistance palette (Figure 5.10e), α and β are varied, with all $\mathbf{q}$ on the ALAP segments in the user sketch being updated accordingly (Figures 5.10(f)–(h)). Horizontal and vertical movement of this marker changes α and β, respectively. Specifically, any position on the left edge of the assistance palette sets $\alpha = 0$ and any position on the right edge sets $\alpha = 1$. The assistance palette shape is triangular, because α does not change strokes so much when β is small.

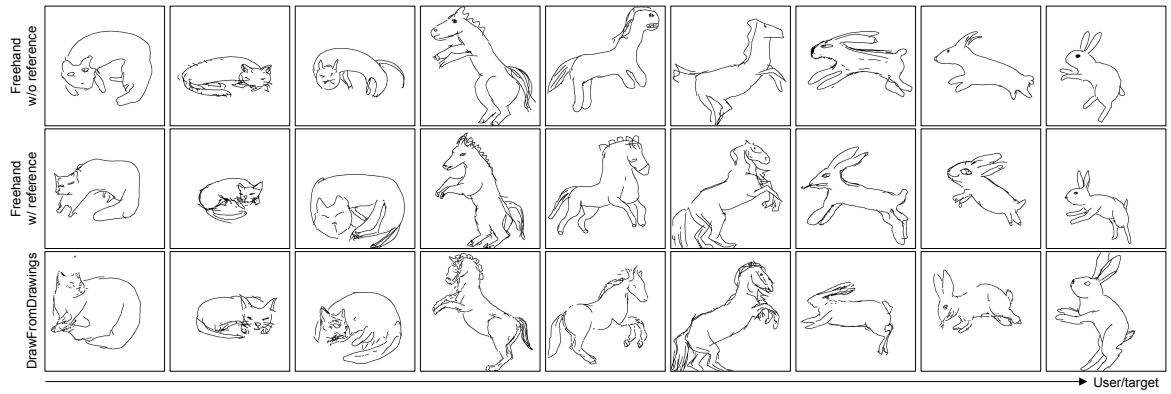


Figure 5.11: Drawings from the user study for task object drawing. Leftmost three columns: a curled cat, middle three columns: a standing horse, and rightmost three columns: a jumping rabbit. All three sketches in each column were done by the same participant. The IDs of participants are 2, 4, 10, 1, 11, 4, 5, 6, and 12, from left to right, respectively.

5.6 Experimental Results

We referred to the databases used in Eitz and colleagues' sketch retrieval system [32] for the categories and used several image search engines on the Internet (i.e., Google Image Search and Bing Image Search) to collect the sketch images. All the sketch images were rasterized. There were 98 category names and 3922 images. We implemented DrawFromDrawings on a PC with a 2.8 GHz Intel Core i7 CPU and 32 GB RAM and used a 21-inch pen display (WACOM DTZ-2100) for drawing.

We performed three user studies, investigating drawing improvements with DrawFromDrawings, comparing between the deformation and suggestive feedback modes, and analyzing utility of the stroke interpolation. In all studies, we first gave instructions on all features (drawing, sketch database retrieval, and the deformation and suggestion feedback modes) of the DrawFromDrawings interface. Next, we asked participants to draw an animal with the DrawFromDrawings system for practice and then started the user study. After tests with the system, participants were asked to complete a survey.

5.6.1 Drawing User Study

The first user study was to draw three specified task objects: a curled cat, a horse standing on two rear legs, and a jumping rabbit. We excluded any sketches that closely matched these task objects from the cat, horse, and rabbit categories to prevent

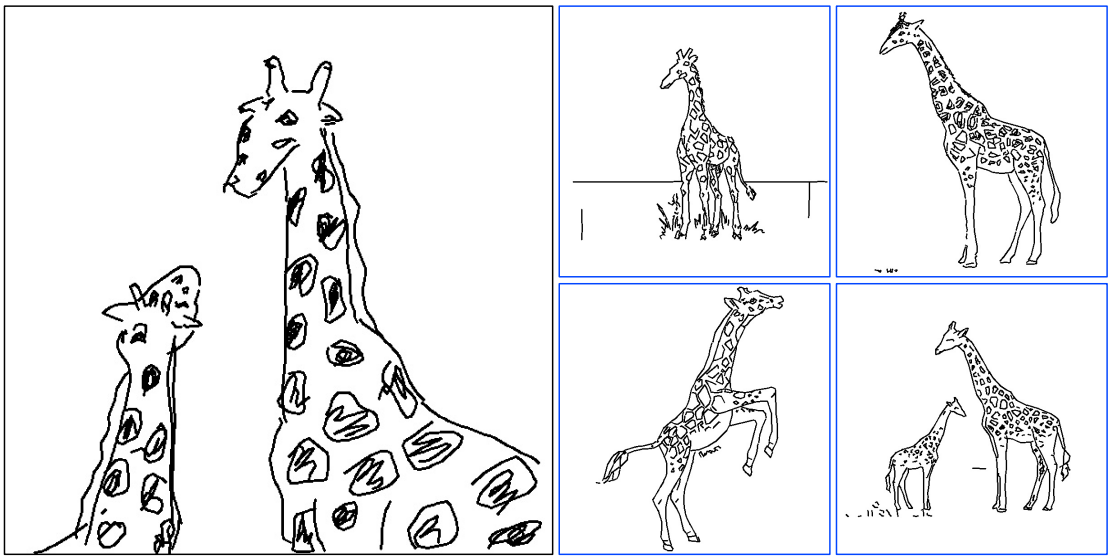


Figure 5.12: Freeform drawing by Participant 3 (left, with black frame) and four sketch images he referred to (right, with blue frames). This participant used suggestive feedback in completing this sketch.

participants from simply reproducing those sketches. We recruited 12 participants. The participants' self-reported drawing skill ranged from 1 (poor) to 5 (good), with a mean of 2.83. We asked them to draw each task object with three versions of the system: freehand drawing without any reference, freehand drawing with reference sketches that they selected themselves from the databases, and drawing with our interface. The order of drawing the task objects and of using the drawing systems was randomized. When drawing with our interface, the participants could choose to use deformation or suggestive feedback modes whenever they wanted. We recorded the modes the participants used (i.e., freehand, deformation feedback, or suggestive feedback). In the survey for the drawing user study, the participants ranked each drawing system with a score from 1 (least/worst) to 5 (most/best) for "fun," "ease of drawing," and "satisfaction with drawings," and wrote down freeform comments.

Drawing Results: Figure 5.11 shows several examples of task object drawings using the three drawing systems. Sketches in the top, middle, and bottom rows were drawn using freehand drawing without any reference, freehand drawing with reference sketches, and drawing with our interface, respectively. Whereas freehand drawing with references was of higher quality than drawing without references, our system improved the quality of the drawings even further.

After the task object drawing, we asked these participants to draw an object from their imagination (e.g, a favorite animal or creature). Figure 5.12 shows the freeform

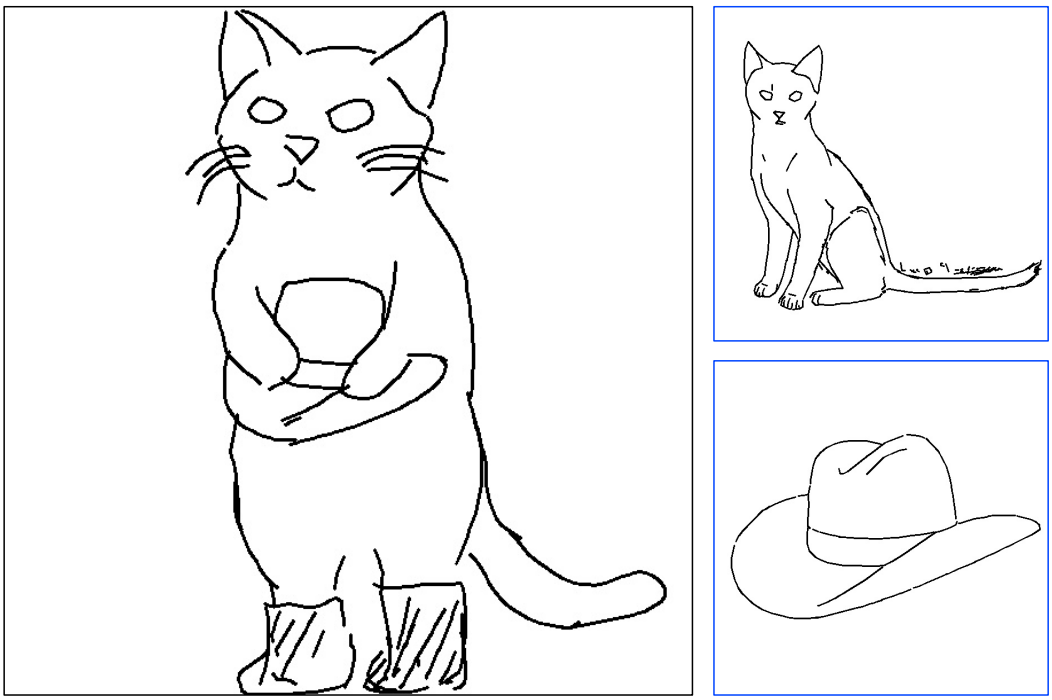


Figure 5.13: Freeform drawing by Participant 12 (left, with black frame) and reference sketch images of a cat and a hat (with blue frames). This participant used suggestive feedback for the hat.

drawing (left, with black frame) produced by Participant 3 with a self-reported drawing skill level of 5. The other four images (right, with blue frames) are sketches that this participant referred to. Although the body shapes and texture were drawn freehand, suggestive feedback was used to its full extent in refining the heads of the parent and child giraffes, thereby preserving the user's stroke style.

Figure 5.13 shows the freeform drawing (left with black frame) produced by Participant 12, whose drawing skill level was 3. The other two images (right, with blue frames) are the sketches referred to. First, the cat was drawn, with a cat sketch (top right) in the reference canvas. The hat was then drawn using suggestive feedback (bottom right), with the strokes representing the forelegs being refined to represent appropriate occlusions. The sketch composition was polished using various features of our interface, and completed as a drawing of *Le Chat Botté* (Puss in Boots).

Figure 5.14 exhibits other results of the freeform drawing study. All participants combined the deformation and suggestive feedback modes in addition to drawing with side-by-side reference images. This would have been difficult with systems based on the imitation and emulation paradigm [30, 70, 48], because such systems simply overlay the current drawing with database sketches or corrective feedback.

Note that the average timings of the task object drawing and freeform drawing were 3.16 min, and 7.32 min, respectively.

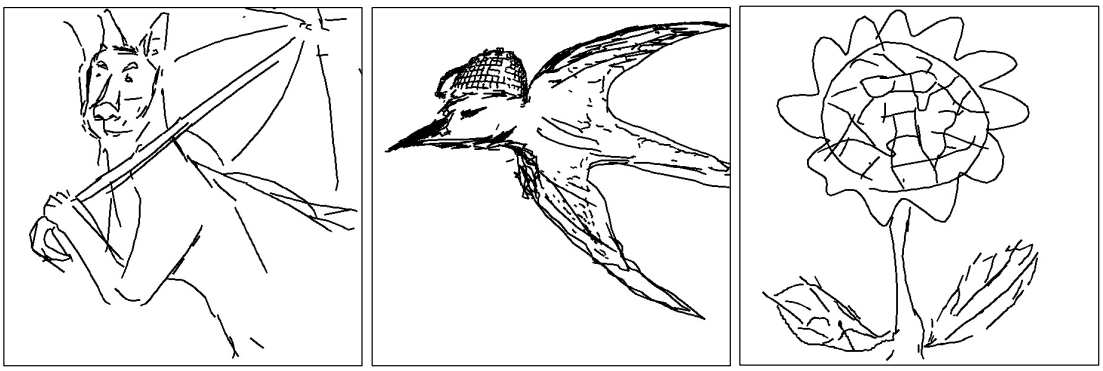
Recorded Data and Survey Results: Figure 5.15-left summarizes the scores for each system. The scores indicate that a significant majority of the users found DrawFromDrawings enjoyable and easy to use, and were satisfied with the quality of sketches using the DrawFromDrawings assistance functions.

The recorded data indicated a strong preference by skillful drawers for drawing strokes with suggestive feedback. Participants 3 and 4, with drawing skill levels of 5 and 3, respectively, used only the suggestive mode of feedback. Participant 4 commented: *A good thing about suggestive feedback is that I can modify my drawings through a reference image with keeping feelings of drawing by myself. In contrast, the drawings with deformation feedback did not match my original drawings.*

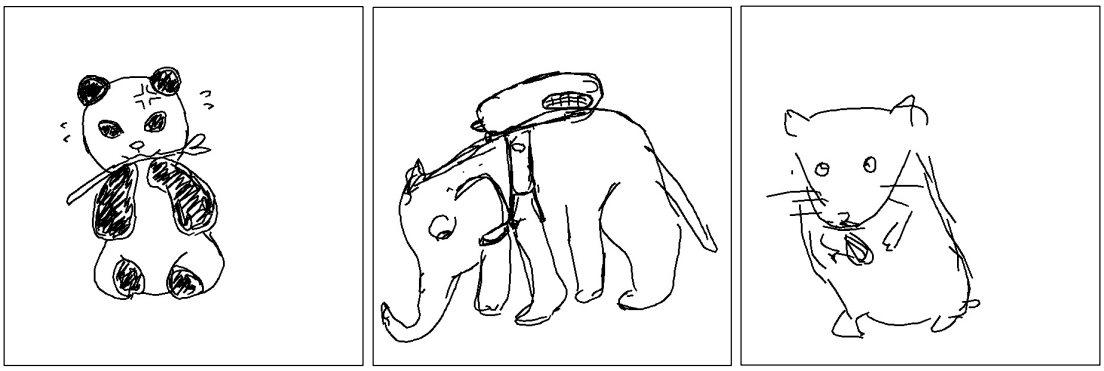
On the other hand, participants with poor self-reported drawing skill levels tended to use deformation feedback rather than suggestive feedback. Participants 6 and 10 (drawing skill levels of 2 and 1) mainly used deformation feedback. According to Participant 10: *Even when I used suggestive feedback, I found I just traced a suggested drawing. Therefore, I finally used only deformation feedback.* In addition, Participant 7 said that deformation feedback was useful for drawing complex objects such as textures and fur.

Overall, each of the participants enjoyed drawing with our system, and was satisfied with their drawings. Participant 3 said: *Even if I started drawing with a vague impression of what to draw, this system motivated me to draw details of the object.* Participant 7 noted that *The benefit of this system is, I can not only specify the parts of drawings for assistance, but I can also control to what extent I want to refer to a reference image.* Participant 12 reported that all operations were very intuitive and easy to understand.

Qualitative Evaluation: We recruited 10 additional participants to evaluate the results of the task object drawing described above. We made a set of triples of three sketches drawn by freehand without reference, freehand with reference, and our system for each task object by all previous participants (36 triples in total), and asked the new participants to pick the best from each triple. We randomized the order of the triples, and the placement of drawings in each triple. As shown in Figure 5.15-right, the participants favored the drawings using our system in 51% of the examples, whereas they favored freehand with reference and freehand without reference in 29% and 20% of cases, respectively.



(a) Participant 7. Drawing level: 3 (b) Participant 5. Drawing level: 3 (c) Participant 2. Drawing level: 2



(d) Participant 8. Drawing level: 3 (e) Participant 6. Drawing level: 2 (f) Participant 4. Drawing level: 3

Figure 5.14: Other results from the freeform drawing session.

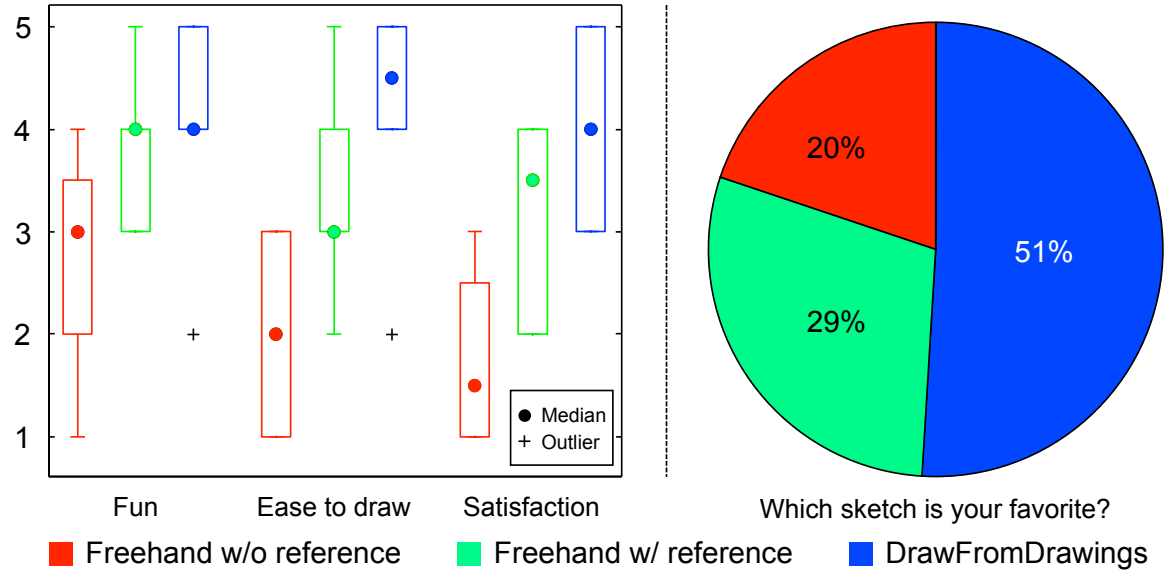


Figure 5.15: Summary of scores. Left: subjective user evaluation of freehand drawing without any reference (red), freehand drawing with reference (green), and drawing with our system (blue). The vertical axis represents the score (1: least/worst; 5: most/best). Right: evaluation of task object drawings by another set of participants.

5.6.2 Deformation vs. Suggestion

We conducted a second user study to evaluate the efficacy and usage of the deformation and suggestive feedback modes. We recruited two participants who were hobbyist drawers. After instruction and practice for an hour, we asked them to draw objects of interest. They were instructed to draw two objects; one using only deformation feedback and the other using only suggestive feedback. After the study, they were asked to complete a freeform survey on the pros and cons of each feedback mode.

The resultant drawings are displayed in Figure 5.16. The two participants referred to three and five reference sketches, respectively, in completing their drawings. The freeform comments from these participants contained similar observations about each feedback mode as those from the task object drawing study. Suggestive feedback enabled them to clarify exactly what they wanted to draw, while keeping their own stroke style. In contrast, deformation feedback allowed them to finish complicated objects quickly, but often resulted in unsatisfactory drawings with an inconsistent stroke style.

Figure 5.16: Drawings from the deformation vs. suggestion study. The top two involved deformation feedback, and the bottom two involved suggestive feedback. The two sketches in each column are by the same participant.

5.6.3 Utility of Interpolation

The third study was conducted to investigate how the stroke interpolation was utilized. In order to eliminate the effect of drawing skill differences, we prepared two reference sketches, *squirrel* and *fish*, from our database, and asked a hobbyist drawer to create six target sketches from the references with different complexity and shape (Figure 5.17). We recruited five participants, and asked them to interpolate the reference sketches to match them to each target sketch by adjusting α and β through the assistance palette.

Figure 5.18 shows the resulting sets of α and β. These interpolation parameters were distributed in the range of $\alpha \in [0.2, 0.8]$ and $\beta \in [0.4, 0.9]$. Surprisingly, α varied independently of the shape differences, despite the role of α in Eq. (5.4). On the other hand, the more complex the target sketches were, the lower β was on average. This observation indicates that β plays an important role, mainly for sketches with complicated texture such as fur. Overall, participants found the combination of α and β useful to achieve satisfactory stroke interpolation.

5.7 Discussion

In both user studies, we observed the same tendency that the participants, particularly those with good drawing skills, preferred suggestive feedback to deformation feedback. According to the resulting drawings and comments by the participants, they preferred to draw characteristic features such as faces and the bend of front legs for the standing horse task on their own, and therefore used suggestive feedback. This may be because experienced participants have already established their own stroke style and wanted to draw such important features in the target objects for themselves. In the first user study, one user (Participant 4) commented that the results of using deformation feedback were not consistent with the user's own original strokes, which the user did not like. One participant in the second user study mentioned that, thanks to the suggested strokes, ideas could be developed gradually during the course of the drawing. According to these comments and the resulting drawings, we believe that the suggestive feedback available in our interface would help users discover their own stroke style and develop their ideas, which is the motivation and goal of our assistive interface and was discussed in the past work [70, 72, 123].

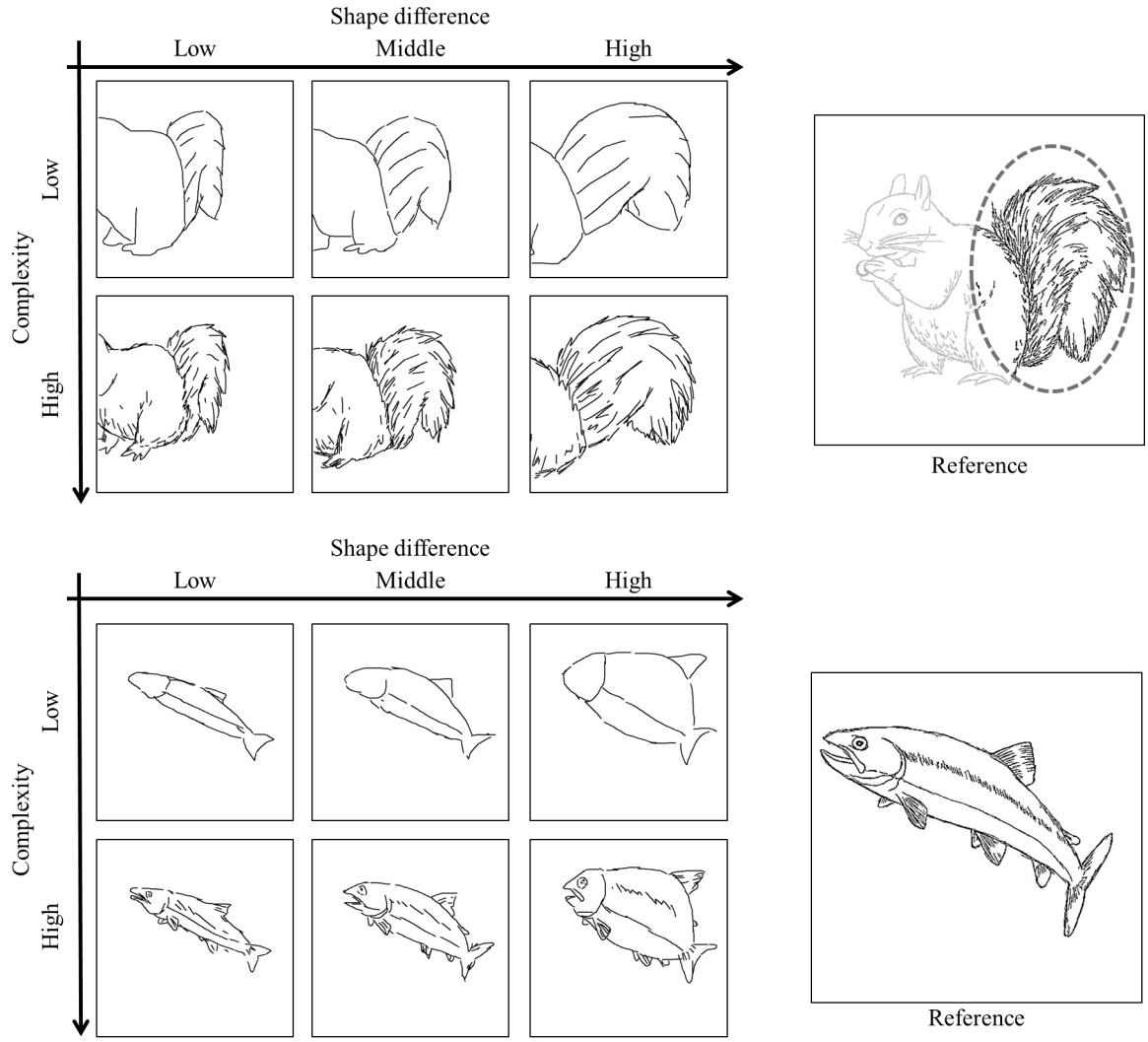


Figure 5.17: The reference and target sketches. Top: *squirrel*, bottom: *fish*.

Participant 7 thought that deformation feedback was useful, particularly for complicated textures such as fur. Although fur is a critical element when drawing animals, it is not as representative as faces. Therefore, deformation feedback might be useful for experienced drawers in reducing the effort in drawing less important but necessary features. This would allow them to focus their creative effort on the more important features.

On the other hand, we observed that some inexperienced participants preferred deformation feedback. This might be because they had not yet established their own stroke style and did not care so much about consistency of stroke style. The way they created these drawings was quite similar to example-based clipart composition [14]

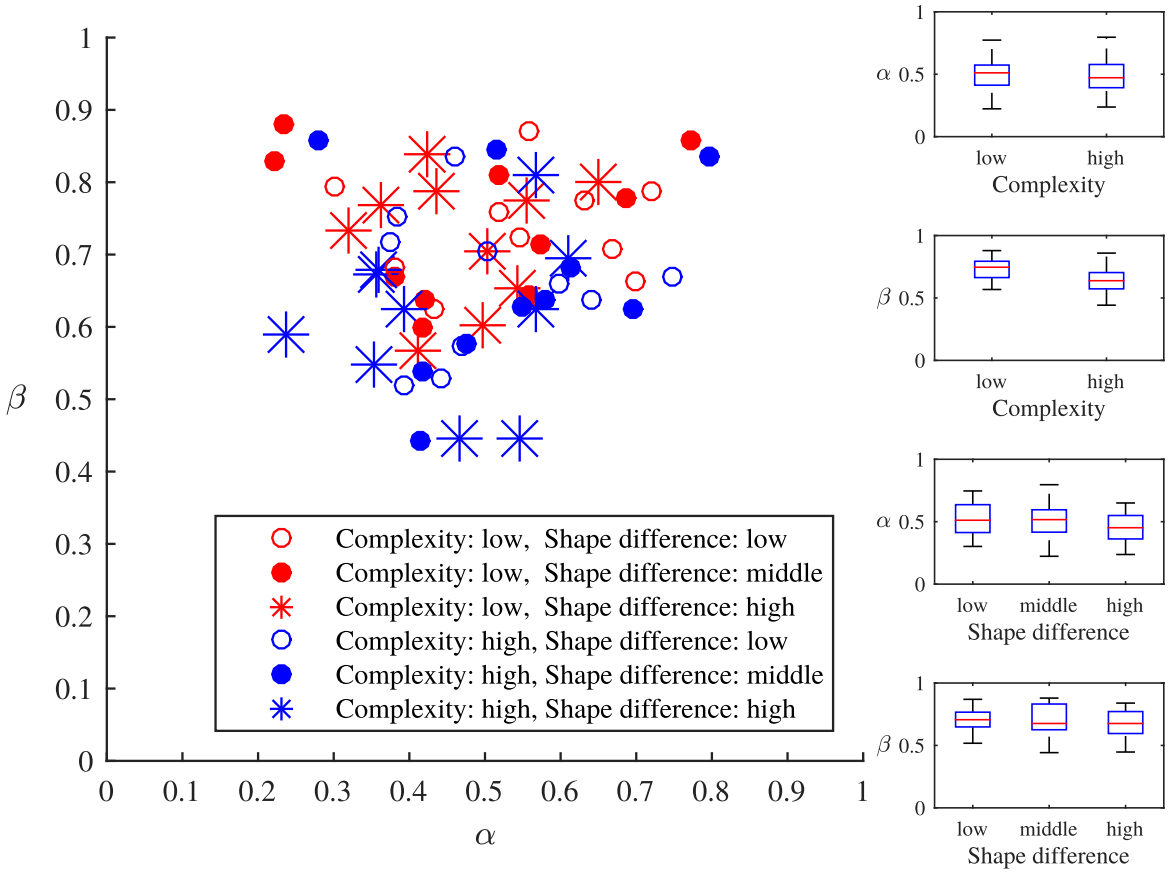


Figure 5.18: Left: Plots of (α, β) in the third user study. Right: Plots of α and β w.r.t. complexity and shape difference.

and assembly-based 3D modeling approaches [21]. However, during the freeform drawing, Participants 2, 6, and 10, all with self-reported poor drawing skills, were able to develop and express their imagination (e.g., Figure 5.14(c)).

Our stroke correspondence detection and interpolation has a limitation. Because our method relies on the convex hulls of ROIs, an association failure might occur, such as when the user-specified ROIs are thin and concave with significantly different orientations. An example of such a failure is shown in Figure 5.19, where the interpolated strokes do not reflect either reference or user strokes. However, we observed that the participants in the user studies always selected strokes with similar outlines from the user and reference sketches, and an ROI specification similar to this example would not be natural for users in general.

Another limitation of the proposed system might be the complexity of its workflow. The drawer needs to select ROIs and exploit the assistance palette, which looks a little more complicated than other suggestive interfaces [30, 70, 48]. However, a typical use of our system would be to first sketch some parts or the whole of a target object and then to improve portions of the drawing with our system, rather than by using the system features for every stroke. We observed that the participants quickly got used to the system features during the user study, and could create their own drawings, more than half of which were judged to be better than drawings with the other two systems by another set of participants (see Figure 5.15). Another limitation, which is shared among all of reference-based drawing assistance systems [70, 72], is that the possible assistance is limited to the sketches in the database.

Our interpolation algorithm considers the three strokes described in Section 5.5.2. Although no participants complained about variations in interpolated strokes, it might be interesting to explore other target strokes, such as user strokes warped to the reference ROI. Having more target strokes would raise another usability issue for the design of the assistance palette.

Another interesting future direction would be a suggestive and/or corrective interface for hatching, which is another important feature in general drawing. Hatching requires different stroke skills, and strategies for analyzing, interpolating, and suggesting hatching strokes might also be different from those for line-drawing strokes. Investigations in this direction might be worthwhile for future computer-assisted drawing systems.

(a) (b) (c)

Figure 5.19: Failure of stroke interpolation for an ant leg. If (a) the reference ROI and (b) the user ROI are thin and concave with significantly different orientations, (c) the interpolated strokes could be meaningless.

Chapter 6

Conclusion

To handle large scale manga data and create novel values, we proposed **the fundamental component**, **the theoretical improvement**, and **the data-driven application**.

For the fundamental component, we build a sketch-based manga retrieval system and novel query schemes. The retrieval system consists of three steps: labeling of margin areas, EOH feature description, and approximate nearest-neighbor search using product quantization. The query schemes include relevance feedback and query retouch, both of which are new schemes that interactively rerank the retrieved results. We built a new dataset of manga images, Manga109, which consists of 21,142 manga images drawn by professional manga artists. To the best of our knowledge, Manga109 is currently the biggest manga image dataset. It is available to the research community. We conducted a comparative study, localization evaluation, and a large-scale qualitative study. The experiments verified that: (1) the retrieval accuracy of the proposed method is higher than those of existing methods; (2) the proposed method can localize an object instance with reasonable runtime and accuracy; and (3) sketch querying is useful for manga search.

For the theoretical improvement, we developed the PQTable, a nonexhaustive search method for finding the nearest PQ codes without need of parameter tuning. The PQTable is based on a multiindex hash table, and includes candidate code generation and merging of multiple tables. From an empirical analysis, we showed that the required parameter value T can be estimated in advance. An experimental evaluation using the SIFT1B data showed that, the PQTable could compute results 10^2 to 10^5 times faster than the ADC-scan method. The disadvantage of the PQTable is its heavier memory overhead compared to the state-of-the-art systems.

For the data-driven application, we have created an interactive assistive interface for 2D drawing. Given selection scribbles and ROIs in the user and reference sketches,

our interface interactively visualizes novel strokes that account for the original user strokes, reference strokes, and potential new strokes. The new strokes are created by warping between shapes in the user ROI and the reference ROI using ALAP stroke segment correspondences as the primitives. Our algorithm considers structurally varying strokes and rasterized sketches from simple user inputs in an interactive manner. We implemented both of the suggestive and deformation feedback modes, and tested them with several participants. The participants enjoyed the experience and were able to create unique sketches using our interface. An evaluation by another set of participants indicated that our system could assist the participants in drawing better sketches than by freehand drawing.

Limitation & Future work

Bibliography

[1] Alexa, M., Cohen-Or, D., and Levin, D. (2000). As-rigid-as-possible shape interpolation. In *Proc. SIGGRAPH*, pages 157–164. ACM.

[2] Alexe, B., Deselaers, T., and Ferrari, V. (2012). Measuring the objectness of image windows. *IEEE TPAMI*, 34(11):2189–2202.

[3] Aramaki, Y., Matsui, Y., Yamasaki, T., and Aizawa, K. (2014). Interactive segmentation for manga. In *Proc. SIGGRAPH Poster*. ACM.

[4] Arvo, J. and Novins, K. (2000). Fluid sketches: Continuous recognition and morphing of simple hand-drawn shapes. In *Proc. UIST*. ACM.

[5] Babenko, A. and Lempitsky, V. (2012). The inverted multi-index. In *Proc. CVPR*. IEEE.

[6] Babenko, A. and Lempitsky, V. (2014a). Additive quantization for extreme vector compression. In *Proc. CVPR*. IEEE.

[7] Babenko, A. and Lempitsky, V. (2014b). Improving bilayer product quantization for billion-scale approximate nearest neighbors in high dimensions. *CoRR*, abs/1404.1831.

[8] Babenko, A. and Lempitsky, V. (2014c). The inverted multi-index. *IEEE TPAMI*.

[9] Babenko, A. and Lempitsky, V. (2015). Tree quantization for large-scale similarity search and classification. In *Proc. CVPR*. IEEE.

[10] Barla, P., Thollot, J., and Sillion, F. (2005). Geometric clustering for line drawing simplification. In *Proc. EGSR*.

[11] Baxter, W. and Anjyo, K. (2006). Latent doodle space. In *Proc. Eurographics*.

[12] Baxter, W., Barla, P., and Anjyo, K. (2009). Compatible embedding for 2d shape animation. *IEEE TVCG*, 15(5):867–879.

[13] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE TPAMI*, 24(4):509–522.

[14] Benjamin, W., Chandrasegaran, S., Ramanujan, D., Elmqvist, N., Vishwanathan, S., and Ramani, K. (2014). Juxtapoze: Supporting serendipity and creative expression in clipart compositions. In *Proc. CHI*, pages 341–350. ACM.

[15] Bookstein, F. L. (1989). Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE TPAMI*, 11(6):567–585.

[16] Buttcher, S., Clarke, C. L. A., and Cormack, G. V. (2010). *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press.

[17] Cao, Y., Chan, A. B., and Lau, R. W. H. (2012). Automatic stylistic manga layout. In *Proc. SIGGRAPH Asia*. ACM.

[18] Cao, Y., Lau, R. W. H., and Chan, A. B. (2014). Look over here: Attention-directing composition of manga elements. In *Proc. SIGGRAPH*. ACM.

[19] Cao, Y., Wang, C., Li, Z., Zhang, L., and Zhang, L. (2010). Spatial-bag-of-features. In *Proc. CVPR*. IEEE.

[20] Cao, Y., Wang, C., Zhang, L., and Zhang, L. (2011). Edgel index for large-scale sketch-based image search. In *Proc. CVPR*. IEEE.

[21] Chaudhuri, S., Kalogerakis, E., Guibas, L., and Koltun, V. (2011). Probabilistic reasoning for assembly-based 3D modeling.

[22] Chen, T., Cheng, M.-M., Tan, P., Shamir, A., and Hu, S.-M. (2009). Sketch2photo: Internet image montage. In *Proc. SIGGRAPH Asia*. ACM.

[23] Cheng, J., Leng, C., Wu, J., Cui, H., and Lu, H. (2014a). Fast and accurate image matching with cascade hashing for 3d reconstruction. In *Proc. CVPR*. IEEE.

[24] Cheng, M.-M., Zhang, Z., Lin, W.-Y., and Torr, P. (2014b). Bing: Binarized normed gradients for objectness estimation at 300fps. In *Proc. CVPR*. IEEE.

[25] Chu, W.-T. and Chao, Y.-C. (2014). Line-based drawing style description for manga classification. In *Proc. MM*. ACM.

[26] Chu, W.-T., Yu, C.-H., and Wang, H.-H. (2015). Optimized comics-based storytelling for temporal image sequences. *IEEE TMM*, 17(2):201–205.

[27] Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. (2007). Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. ICCV*. IEEE.

[28] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proc. CVPR*. IEEE.

[29] Delalandre, M., Iwata, M., and Kise, K. (2014). Fast and optimal binary template matching application to manga copyright protection. In *Proc. DAS*.

[30] Dixon, D., Prasad, M., and Hammond, T. (2010). icandraw: Using sketch recognition and corrective feedback to assist a user in drawing human faces. In *Proc. CHI*. ACM.

[31] Duan, L.-Y., Lin, J., Wang, Z., Huang, T., and Gao, W. (2015). Weighted component hashing of binary aggregated descriptors for fast visual search. *IEEE TMM*, 17(6):828–842.

[32] Eitz, M., Hays, J., and Alexa, M. (2012). How do humans sketch objects? In *Proc. SIGGRAPH*. ACM.

[33] Eitz, M., Hildebrand, K., Boubekeur, T., and Alexa, M. (2011). Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE TVCG*, 17(11):1624–1636.

[34] Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338.

[35] Fang, H., Gupta, S., Iandola, F., Srivastava, R. K., Deng, L., Dollar, P., Gao, J., He, X., Mitchell, M., Platt, J. C., Zitnick, C. L., and Zweig, G. (2015). From captions to visual concepts and back. In *Proc. CVPR*. IEEE.

[36] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 24(6):381–395.

[37] Ge, T., He, K., Ke, Q., and Sun, J. (2014). Optimized product quantization. *IEEE TPAMI*, 36(4):744–755.

[38] Gong, Y., Lazebnik, S., Gordo, A., and Perronnin, F. (2013). Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI*, 35(12):2916–2929.

[39] Greene, D., Parnas, M., and Yao, F. (1994). Multi-index hashing for information retrieval. In *Proc. FOCS*. IEEE.

[40] Guérin, C., Rigaud, C., Mercier, A., Ammar-Boudjelal, F., Betet, K., Bouju, A., Burie, J.-C., Louis, G., Ogier, J.-M., and Revel, A. (2013). eBDtheque: A Representative Database of Comics. In *Proc. ICDAR*. IEEE.

[41] He, K., Wen, F., and Sun, J. (2013). K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In *Proc. CVPR*. IEEE.

[42] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*. IEEE.

[43] Heo, J.-P., Lin, Z., and Yoon, S.-E. (2014). Distance encoded product quantization. In *Proc. CVPR*. IEEE.

[44] Herranz, L., Ćalić, J., Martínez, J. M., and Mrak, M. (2012). Scalable comic-like video summaries and layout disturbance. *IEEE TMM*, 14(4):1290–1297.

[45] Hoashi, K., Ono, C., Ishii, D., and Watanabe, H. (2011). Automatic preview generation of comic episodes for digitized comic search. In *Proc. MM*. ACM.

[46] Hu, R., Barnard, M., and Collomosse, J. (2010). Gradient field descriptor for sketch based retrieval and localization. In *Proc. ICIP*. IEEE.

[47] Hu, R., Wang, T., and Collomosse, J. (2011). A bag-of-regions approach to sketch-based image retrieval. In *Proc. ICIP*. IEEE.

[48] Iarussi, E., Bousseau, A., and Tsandilas, T. (2013). The drawing assistant: Automated drawing guidance and feedback from photographs. In *Proc. UIST*, pages 183–192. ACM.

[49] Igarashi, T. and Huges, J. F. (2001). A suggestive interface for 3d drawing. In *Proc. UIST*. ACM.

[50] Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H. (1997). Interactive beautification: A technique for rapid geometric design. In *Proc. UIST*. ACM.

[51] Igarashi, T., Moscovich, T., and Hughes, J. F. (2005). As-rigid-as-possible shape manipulation. In *Proc. SIGGRAPH*. ACM.

[52] Ito, K., Matsui, Y., Yamasaki, T., and Aizawa, K. (2015). Separation of line drawings and screentones of manga. In *Proc. Eurographics*.

[53] Iwamura, M., Sato, T., and Kise, K. (2013). What is the most efficient way to select nearest neighbor candidates for fast approximate nearest neighbor search? In *Proc. ICCV*. IEEE.

[54] Jègou, H., Douze, M., and Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *Proc. ECCV*.

[55] Jègou, H., Douze, M., and Schmid, C. (2009). Packing bag-of-features. In *Proc. ICCV*. IEEE.

[56] Jègou, H., Douze, M., and Schmid, C. (2011a). Product quantization for nearest neighbor search. *IEEE TPAMI*, 33(1):117–128.

[57] Jègou, H., Perronnin, F., Douze, M., Sànchez, J., Pèrez, P., and Schmid, C. (2012). Aggregating local image descriptors into compact codes. *IEEE TPAMI*, 34(9):1704–1716.

[58] Jègou, H., Tavenard, R., Douze, M., and Amsaleg, L. (2011b). Searching in one billion vectors: Re-rank with souce coding. In *Proc. ICASSP*. IEEE.

[59] Jiang, Y.-G., Wang, J., Xue, X., and Chang, S.-F. (2013). Query-adaptive image search with hash codes. *IEEE TMM*, 15(2):442–453.

[60] Jing, G., Hu, Y., Guo, Y., Yu, Y., and Wang, W. (2015). Content-aware video2comics with manga-style layout. *IEEE TMM*.

[61] Jonker, R. and Volgenant, A. T. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

[62] Kalantidis, Y. and Avrithis, Y. (2014). Locally optimized product quantization for approximate nearest neighbor search. In *Proc. CVPR*. IEEE.

[63] Kazi, R. H., Igarashi, T., Zhao, S., and Davis, R. C. (2012). Vignette: Interactive texture design and manipulation with freeform gestures for pen-and-ink illustration. In *Proc. CHI*. ACM.

[64] King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758.

[65] Kleiner, F. S. (2013). *Gardner's Art through the Ages: The Western Perspective*. Cengage Learning.

[66] Kopf, J. and Lischinski, D. (2012). Digital reconstruction of halftoned color comics. In *Proc. SIGGRAPH Asia*. ACM.

[67] Lab., A. Y. (2015). Manga109 dataset. http://www.manga109.org.

[68] Laszlo, J., Neff, M., and Singh, K. (2005). Predictive feedback for interactive control of physics-based characters. In *Proc. Eurographics*.

[69] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*. IEEE.

[70] Lee, Y. J., Zitnick, C. L., and Cohen, M. F. (2011). Shadowdraw: Real-time user guidance for freehand drawing. In *Proc. SIGGRAPH*. ACM.

[71] Levi, K. and Weiss, Y. (2004). Learning object detection from a small number of examples: the importance of good features. In *Proc. CVPR*. IEEE.

[72] Limpaecher, A., Feltman, N., Treuille, A., and Cohen, M. (2013). Real-time drawing assistance through crowdsourcing. In *Proc. SIGGRAPH*.

[73] Liu, D., Chen, Q., Yu, J., Gu, H., Tao, D., and Seah, H. S. (2011). Stroke correspondence construction using manifold learning. *Computer Graphics Forum*, 30(8):2194–2207.

[74] Liu, M.-Y., Tuzel, O., Veeraraghavan, A., and Chellappa, R. (2010). Fast directional chamfer matching. In *Proc. CVPR*. IEEE.

[75] Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE TIT*, 28(2):129–137.

[76] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110.

[77] Lu, J., Yu, F., Finkelstein, A., and DiVerdi, S. (2012). Helpinghand: Example-based stroke stylization. In *Proc. SIGGRAPH*. ACM.

[78] Marchand-Mailet, S. and Sharaiha, Y. M. (1999). *Binary Digital Image Processing: A Discrete Approach*. Academic Press.

[79] Matsui, Y., Aizawa, K., and Jing, Y. (2014). Sketch2manga: Sketch-based manga retrieval. In *Proc. ICIP*. IEEE.

[80] Matsui, Y., Yamasaki, T., and Aizawa, K. (2011). Interactive manga retargeting. In *Proc. SIGGRAPH Poster*. ACM.

[81] Mei, T., Rui, Y., Li, S., and Tian, Q. (2014). Multimedia search reranking: A literature survey. *ACM Comp. Surv.*, 46(3):38:1–38:38.

[82] Norouzi, M. and Fleet, D. J. (2013). Cartesian k-means. In *Proc. CVPR*. IEEE.

[83] Norouzi, M., Punjani, A., and Fleet, D. J. (2014). Fast exact search in hamming space with multi-index hashing. *IEEE TPAMI*, 36(6):1107–1119.

[84] Olsen, L., Samavati, F. F., Sousa, M. C., and Jorge, J. A. (2009). Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103.

[85] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*. IEEE.

[86] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. CVPR*. IEEE.

[87] Qu, Y., Pang, W.-M., Wong, T.-T., and Heng, P.-A. (2008). Richness-preserving manga screening. In *Proc. SIGGRAPH Asia*. ACM.

[88] Qu, Y., Wong, T.-T., and Heng, P.-A. (2006). Manga colorization. In *Proc. SIGGRAPH*. ACM.

[89] Ren, S., Jin, C., Sun, C., and Zhang, Y. (2014). Sketch-based image retrieval via adaptive weightinge. In *Proc. ICMR*. ACM.

[90] Rigaud, C., Burie, J.-C., Ogier, J.-M., Karatzas, D., and de Weijer, J. V. (2013). An active contour model for speech balloon detection in comics. In *Proc. ICDAR*. IEEE.

[91] Rigaud, C., Guérin, C., Karatzas, D., Burie, J.-C., and Ogier, J.-M. (2015). Knowledge-driven understanding of images in comic books. *IJDAR*, 18(3):199–221.

[92] Rother, C., Kolmogorov, V., and Blake, A. (2004). "grabcut": Interactive foreground extraction using iterated graph cuts. In *Proc. SIGGRAPH*, pages 309–314. ACM.

[93] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). Imagenet large scale visual recognition challeng. *IJCV*, pages 1–42.

[94] Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. J. (2013). Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245.

[95] Sato, K., Matsui, Y., Yamasaki, T., and Aizawa, K. (2014). Reference-based manga colorization by graph correspondence using quadratic programming. In *Proc. SIGGRAPH Asia Technical Brief*. ACM.

[96] Schaefer, S., McPhail, T., and Warren, J. (2006). Image deformation using moving least squares. In *Proc. SIGGRAPH*. ACM.

[97] Schneider, R. G. and Tuytelaars, T. (2014). Sketch classification and classification-driven analysis using fisher vectors. In *Proc. SIGGRAPH Asia*. ACM.

[98] Serra, J. (1983). *Image Analysis and Mathematical Morphology*. Academic Press.

[99] Shotton, J., Blake, A., and Cipolla, R. (2008). Multi-scale categorical object recognition using contour fragments. *IEEE TPAMI*, 30(7):1270–1281.

[100] Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *Proc. ICCV*. IEEE.

[101] Smeulders, A. W., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE TPAMI*, 22(12):1349–1380.

[102] Spyromitros-Xioufis, E., Papadopoulos, S., Kompatsiaris, I. Y., Tsoumakas, G., and Vlahavas, I. (2014). A comprehensive study over vlad and product quantization in large-scale image retrieval. *IEEE TMM*, 16(6):1713–1728.

[103] Su, Q., Li, W. H. A., Wang, J., and Fu, H. (2014). Ez-sketching: Three-level optimization for error-tolerant image tracing. In *Proc. SIGGRAPH*.

[104] Sun, W., Burie, J.-C., Ogier, J.-M., and Kise, K. (2013a). Specific comic character detection using local feature matching. In *Proc. ICDAR*. IEEE.

[105] Sun, W. and Kise, K. (2013). Detection of exact and similar partial copies for copyright protection of manga. *IJDAR*, 16(4):331–349.

[106] Sun, X., Wang, C., Xu, C., and Zhang, L. (2013b). Indexing billions of images for sketch-based retrieval. In *Proc. MM*. ACM.

[107] Sýkora, D., Dingliana, J., and Collins, S. (2009). As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proc. NPAR*.

[108] Sýkora, D., Dingliana, J., and Collins, S. (2009). Lazybrush: Flexible painting tool for hand-drawn cartoons. In *Proc. Eurographics*.

[109] Tanaka, T., Shoji, K., Toyama, F., and Miyamichi, J. (2007). Layout analysis of tree-structured scene frames in comic images. In *Proc. IJCAI*.

[110] Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2015). The new data and new challenges in multimedia research. *CoRR*, abs/1503.01817.

[111] Tsang, S., Balakrishnan, R., Singh, K., and Ranjan, A. (2004). A suggestive interface for image guided 3d sketching. In *Proc. CHI*. ACM.

[112] Tseng, K.-Y., Lin, Y.-L., Chen, Y.-H., and Hsu, W. H. (2012). Sketch-based image retrieval on mobile devices using compact hash bits. In *Proc. MM*. ACM.

[113] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective search for object recognition. *IJCV*, 104(2):154–171.

[114] Vedaldi, A. and Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *Proc. MM*.

[115] Wang, J., Shen, H. T., Song, J., and Ji, J. (2014a). Hashing for similarity search: A survey. *CoRR*, abs/1408.2927.

[116] Wang, J., Wang, J., Song, J., Xu, X.-S., Shen, H. T., and Li, S. (2014b). Optimized cartesian k-means. *IEEE TKDE*.

[117] Wang, S., Zhang, J., Han, T. X., and Miao, Z. (2015). Sketch-based image retrieval through hypothesis-driven object boundary selection with hlr descriptor. *IEEE TMM*, 17(7):1045–1057.

[118] Wang, X., Bai, X., Ma, T., Liu, W., and Latecki, L. J. (2012). Fan shape model for object detection. In *Proc. CVPR*. IEEE.

[119] Weiss, Y., Torralba, A., and Fergus, R. (2008). Spectral hashing. In *Proc. NIPS*.

[120] Whited, B., Noris, G., Simmons, M., Sumner, R. W., Gross, M., and Rossignac, J. (2010). Betweenit: An interactive tool for tight inbetweening. In *Proc. Eurographics*.

[121] Wu, Z., Ke, Q., Isard, M., and Sun, J. (2009). Bundling features for large scale partial-duplicate web image search. In *Proc. CVPR*. IEEE.

[122] Xia, Y., He, K., Wen, F., and Sun, J. (2013). Joint inverted indexing. In *Proc. ICCV*. IEEE.

[123] Xie, J., Hertzmann, A., Li, W., and Winnemöller, H. (2014). PortraitSketch: Face sketching assistance for novices. In *Proc. UIST*. ACM.

[124] Yu, Q., Yang, Y., Song, Y.-Z., Xiang, T., and Hospedales, T. M. (2015). Sketch-a-net that beats humans. In *Proc. BMVC*.

[125] Zhang, T., Du, C., and Wang, J. (2014). Composite quantization for approximate nearest neighbor search. In *Proc. ICML*.

[126] Zhang, T., Qi, G.-J., Tang, J., and Wang, J. (2015). Sparse composite quantization. In *Proc. CVPR*. IEEE.

[127] Zhang, T. Y. and Suen, C. Y. (1984). A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239.

[128] Zhang, W. and Ngo, C.-W. (2015). Topological spatial verification for instance search. *IEEE TMM*, 17(8):1236–1247.

[129] Zhang, Y., Jia, Z., and Chen, T. (2011). Image retrieval with geometry-preserving visual phrases. In *Proc. CVPR*. IEEE.

[130] Zhou, R., Chen, L., and Zhang, L. (2012). Sketch-based image retrieval on a large scake database. In *Proc. MM*. ACM.

[131] Zitnick, C. L. (2013). Handwriting beautification using token means. In *Proc. SIGGRAPH*.

Publications

Publications related to the thesis

Journal papers

[1] Yusuke Matsui, Takaaki Shiratori, and Kiyoharu Aizawa. "DrawFromDrawings: 2D Drawing Assistance with a Sketch Database." In *Trans. Vis. Comp. Graph.*, IEEE, 2015. (minor revision)

International conference

[2] Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. "PQTable: Fast Exact Asymmetric Distance Neighbor Search for Product Quantization using Hash Tables." In *Proc. ICCV*, IEEE, 2015.

[3] Yusuke Matsui. "Challenge for Manga Processing: Sketch-based Manga Retrieval." In *Proc. MM*, Doctoral Symposium, ACM, 2015.

[4] Yusuke Matsui, Kiyoharu Aizawa, and Yushi Jing. "Sketch2Manga: Sketch-Based Manga Retrieval." In *Proc. ICIP*, IEEE, 2014.

Domestic conference

[5] 松井勇佑，山崎俊彦，相澤清晴． "PQTable: ハッシュテーブルを用いたプロダクト量子化ベクトルの高速探索"． 画像符号化シンポジウム / 映像メディア処理シンポジウム (PCSJ/IMPS)，2015．

[6] 松井勇佑，相澤清晴，Yushi Jing． "スケッチ入力を用いた漫画画像検索"． HCGシンポジウム，2014．

Publications non-related to the thesis

International conference

[7] Yuji Aramaki, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Interactive Segmentation for Manga using Lossless Thinning and Coarse Labeling." In *Proc. ASC*, APSIPA, 2015.

[8] Masaki Saito and Yusuke Matsui. "Illustration2Vec: A Semantic Vector Representation of Illustrations." In *Proc. SIGGRAPH Asia*, Technical Briefs, ACM, 2015.

[9] Tuan Anh Nguyen, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Selective K-means Tree Search." In *Proc. MM*, Short Paper, ACM, 2015.

[10] Tuan Anh Nguyen, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Searching for Nearest Neighbors with A Dense Space Partitioning." In *Proc. ICIP*, IEEE, 2015.

[11] Kota Ito, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Separation of Manga Line Drawings and Screentones." In *Proc. Eugoraphics*, Short Paper, 2015.

[12] Kazuhiro Sato, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Reference-based Manga Colorization by Graph Correspondence Using Quadratic Programming." In *Proc. SIGGRAPH Asia*, Technical Brief, ACM, 2014.

[13] Saemi Choi, Yusuke Matsui, and Kiyoharu Aizawa. "Diffusion: Change the Ambience of a Space with a Small Amount of Ink." In *Proc. SIGGRAPH Asia*, Poster, ACM, 2014.

[14] Daiki Matsumoto, Yusuke Matsui, Toshihiko Yamasaki, Kiyoharu Aizawa, Takanori Katagiri. "IllustStyleMap: Visualization of Illustrations based on Similarity of Drawing Style of Authors." In *Proc. SIGGRAPH*, Poster, ACM, 2014.

[15] Yuji Aramaki, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. "Interactive Segmentation for Manga" In *Proc. SIGGRAPH*, Poster, ACM, 2014.

[16] Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. Interactive Manga Retargeting. In *Proc. SIGGRAPH*, Poster, ACM, 2011.

Domestic conference

[17] 伊東浩太，松井勇佑，山崎俊彦，相澤清晴． "グラフを用いた漫画画像の領域選択"． HCGシンポジウム，2015．

[18] 荒巻祐治，松井勇佑，山崎俊彦，相澤清晴．“拡張グルーピングと深層特徴を用いた漫画における文字領域検出の改善”．HCGシンポジウム，2015.

[19] 伊東浩太，松井勇佑，山崎俊彦，相澤清晴．“学術漫画データセットのためのアノテーションツール”．映像情報メディア学会年次大会 (ITE)，2015.

[20] 荒巻祐治，松井勇佑，山崎俊彦，相澤清晴．“連結成分に基づいた漫画における文字領域の検出”．映像情報メディア学会年次大会 (ITE)，2015.

[21] 藤本東，松井勇佑，山崎俊彦，相澤清晴．“ラフスケッチの線画化と評価の検討”．画像の認識・理解シンポジウム (MIRU)，2015.

[22] 藤本東，松井勇佑，山崎俊彦，相澤清晴．“フィルタリング処理に基づくラフスケッチの線画化の検討”．マルチメディア・仮想環境基礎研究会 (MVE)，2015.

[23] 伊東浩太，松井勇佑，山崎俊彦，相澤清晴．“漫画のスクリーントーン除去に関する検討”．HCGシンポジウム，2014.

[24] 松本大輝，松井勇佑，山崎俊彦，相澤清晴，片桐孝憲．“著者の画風の類似性に基づくイラスト画像の可視化”．パターン認識・メディア理解研究会 (PRMU)，2014.

[25] 荒巻祐治，松井勇佑，山崎俊彦，相澤清晴．“漫画を対象としたインタラクティブセグメンテーション”．パターン認識・メディア理解研究会 (PRMU)，2014.

[26] 松本大輝，松井勇佑，山崎俊彦，相澤清晴．“フィッシャーベクターと属性を用いたイラスト画像の著者同定とクラスタリング”．電子情報通信学会総合大会 (IEICE)，2014.

[27] 荒巻祐治，松井勇佑，山崎俊彦，相澤清晴．“漫画におけるオブジェクトの選択的抽出”．電子情報通信学会総合大会 (IEICE)，2014.

[28] 佐藤和博，松井勇佑，山崎俊彦，相澤清晴．“マンガ彩色のための2次計画法によるマッチング”．パターン認識・メディア理解研究会 (PRMU)，2014.

[29] 佐藤和博，松井勇佑，相澤清晴．“類似画像を用いたマンガの彩色”．情報科学技術フォーラム (FIT)，2013.

[30] 佐藤和博，松井勇佑，相澤清晴．“類似画像を用いたマンガの彩色”．画像の認識・理解シンポジウム (MIRU)，2013.

[31] 佐藤和博，松井勇佑，相澤清晴．“参照画像を利用したマンガキャラクタの彩色手法の検討”．画像の認識・理解シンポジウム (MIRU)，2012.

[32] 新井俊宏，松井勇佑，相澤清晴． “漫画画像からの顔検出”． 電子情報通信学会総合大会 (IEICE)，2012．

[33] 佐藤和博，松井勇佑，相澤清晴． “参照画像を利用したマンガキャラクタの彩色”． 電子情報通信学会総合大会 (IEICE)，2012．

[34] 松井勇佑，山崎俊彦，相澤清晴． “インタラクティブな漫画のリターゲティング”． 映像情報メディア学会年次大会 (ITE)，2011．

[35] 松井勇佑，山崎俊彦，相澤清晴． “漫画に対するリターゲティング”． 画像の認識・理解シンポジウム (MIRU)，2011．

[36] 松井勇佑，山崎俊彦，相澤清晴． “二値線画である漫画画像のリターゲティング”． 電子情報通信学会総合大会 (IEICE)，2011．

Technical descriptions

[37] Yusuke Matsui, Kota Ito, Yuji Aramaki, Toshihiko Yamasaki, and Kiyoharu Aizawa. “Sketch-based Manga Retrieval using Manga109 Dataset.” In *arXiv*, 2015.

[38] 松井勇佑，山崎俊彦． “ゼロから始めるクラウド・コンピューティング～Amazon Web Service応用編～”． 映像情報メディア学会誌，7月号，vol. 69，pp. 112-116，2015．

[39] 松井勇佑，崔セミ． “SIGGRAPH ASIA 2014 参加報告”． 映像情報メディア学会誌，3月号，vol. 69，pp. 224-227，2015．

[40] 松井勇佑，佐野峻平，古田諒佑． “ICIP 2014 参加報告”． 映像情報メディア学会誌，2月号，vol. 69，pp. 136-140，2015．

[41] 松井勇佑. “マイクロソフトリサーチアジア（北京）見学取材レポート”． 映像情報メディア学会誌，2月号，vol. 57，pp. 155-156，2014．

[42] 松井勇佑. “Kniectを用いた画像処理プログラミング(1)”． 映像情報メディア学会誌，9月号，vol. 40，pp. 766-770，2012．

Awards

[43] 松井勇佑. 学生論文賞． IMPS，2015．

[44] 松井勇佑. 最優秀インタラクティブ発表賞． HCGシンポジウム，2014．

[45] 松井勇佑. 学府長賞． 東京大学大学院学際情報学府，2013.

[46] 松井勇佑. 原島博学術奨励賞． 電気・電子情報学術振興財団，2012.

[47] Yusuke Matsui. ACM Student Research Competition 3rd Place. SIGGRAPH, 2011.

[48] 松井勇佑. 工学部長賞． 東京大学工学部，2011.