ACKNOWLEDGMENTS

We wish to express our greatest gratitude towards Dr. Tran Minh Triet, our adviser in this thesis for his instructions, direction specification, our motivation, and especially the truthfulness in doing scientific research. He teaches us not only background of computer science but also necessary soft skills in thinking and action. His devotion always inspires us a lot to go on. Without him, we could not overcome obstacles of the very first steps.

To our mentor Le Do Hoang Nam in Machine Learning group, we are grateful for his technical supports, including helpful comments on our progress. His lectures always create our motivations about new knowledge, which has brief explanation.

We also give special thanks to our young sister and our kind friends, Tran Thi Khanh Linh and Nguyen Huynh Duy Hung, for providing powerful resource and external device of computer to conduct important experiments, and warmly thanks to our aunt for a great place during our work.

Our appreciation is extended to all of our lecturers, who provide us valuable technical skills, critical thinking during our study in Advance Program in Computer Science of Faculty of Information Technology, University of Science.

Furthermore, we would like to thank to all members in our research group such as our friends, Nguyen Ngoc Chau Sang and Nguyen Hoang Minh Tri for code assistant.

Last but not least, our parents are the ones we are deeply indebted. Without their financial and mental support, not only during our work on this dissertation but also throughout our time in University, we could not have full concentration to complete the final project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Human-Computer Interaction is a research field that attempts to improve the way people and computers interact by creating smarter human-computer interfaces replacing traditional interfaces. Augmented Reality is a popular trend of Human-Computer Interaction field, which integrates the real physical world with augmented information corresponding to entities and external context in reality. Augmented Reality aims to blur the boundary between real and virtual world to create a more natural and exciting human-computer interface for users. Due to many kinds of augmented information, Augmented Reality has an enormous number of applications in various social sectors. However, conventional Augmented Reality systems focus only on enabling users to interact with virtual augmented information in the rendering phase of Augmented Reality. In contrast, we would like to propose and develop a smart environment of Augmented Reality in which users are allowed to interact with printed predefined templates to be inputted to the detection phase by creating sketches themselves as alternating inputs. Our proposed system can recognize users' sketches and then overlay the corresponding helpful information for users. For the sketch recognition module in our system, we prefer category-level object recognition approaches to instant recognition approaches because of the wide variance of users' sketches. For this reason, we evaluate some advanced image classification approaches, which are recently applied in Machine Learning, on a standard well-known sketch dataset to figure out the most suitable approach among them for our sketch recognition module. The chosen approach is a approach which computes a explicit feature map to transform image features from a non-linear kernel to a linear one, which can combine the high accuracy of a non-linear classifier and the high speed of a linear classifier. The efficient of this approach makes our

proposed Augmented Reality system able to run in real-time with high accuracy. In addition, we design the AR system so that the sketch recognition module is completely independent of the system. This independence makes the sketch recognition module very flexible and any image classification approach can be plugged in the module without changing the architecture of the system. Besides testing the system with the standard sketch dataset, we also demonstrate a demo of our system with a sketch dataset of planet signs, which is created by ourselves, as a example of numerous practical applications of our proposed system.

Chapter 1

Introduction

1.1 Overview

At the dawn of computing, computers were only utilized as machines for fast computation. As computers were getting much smaller and easier to use, people began to use computers more for their personal tasks with the help of mouse and keyboard, the only means of computer interaction at that time. As computers were becoming an integral part of people's lives, the clear need for more flexible ways of interaction between people and computers arose. Human-Computer Interaction (HCI) is a research field that aims to improve the approaches that people (users) and computers interact to make computers more friendly, usable, and receptive to user's needs.

Augmented Reality (AR) is one of the most popular trends of comtemporary HCI field that aims at developing new human computer interfaces. Instead of showing information on isolated environment generated by computers, AR integrates physical world with augmented information corresponding to entities and external context in reality. By this way, AR can blur the boundary between real and virtual worlds to create a more natural and exciting human-computer interface for users. Although the concept of AR originated in 1960's, studies and applications of AR have flourished only since late 1990's [28]. Augmented information in AR can be in various kinds, e.g. texts, clips, images, graphs, or 3D models. Therefore, AR has an enormous number of applications in a variety

of sectors such as education [24], health care [25], geology [39], and art [29]. Google Glass[1], a wearable AR system with an optical head-mounted display, is planned to be officially released in late 2013, which is a remarkable milestone in the development of AR and allows AR to be widely applied to most of aspects of daily life in near future.

A typical AR system consists of 2 main phases: detection and rendering. The detection phase tries to detect if a learned template exists in a query image captured from the real world. The rendering phase then augments useful virtual objects corresponding to the found templates. In the early AR systems, these 2 phases were quite static in the sense that no user's interaction is involved in those processes. As AR was developed more rapidly in 1990's, people started to wish to be able to interact with AR systems. Many researches have been undertaken to integrate the feature of user interaction into modern AR systems. However, recent researches have focused only on adding user interaction experience to the second phase of AR, i.e. the rendering phase. As a result, dozens of AR applications nowadays allows users to interact with augmented virtual objects, e.g. manipulate the 3D objects or select the overlaid information by hand, in real-time so that the systems know user's feedback from which they can give subsequent appropriate responses.

On the other hand, the first phase of AR, i.e. the detection phase, has been mostly improved on forms of templates, not the way users interact with those templates. Some kinds of templates has been used in conventional AR systems. The first kind of template is marker. A marker is a special and predefined pattern, e.g. Barcode, QR code, or BCH-ID code, associated with a virtual object to be augmented. A risen issue is that those artificial markers are easy for systems to detect but not friendly to users. A user typically cannot know exactly which virtual object corresponds to a given marker. Another kind of template, known as natural image, is proposed to be used in AR, which minimizes the user feeling of unnaturalness caused by markers. A natural image used as a template

[1]Source: Google, 7 2013. [Online]. Available: http://www.google.com/glass/start/.

for detection can be a panel, a magazine's page, a product's cover, as well as a magic card. Due to the natural template, users can directly apply AR on various regular things to see more augmented information about them without the requirement of a temporary meaningless marker. Furthermore, the replacement of markers with natural images allows researchers to extend the original problem of instance recognition which usually uses feature matching techniques to the problem of class recognition in which an AR system can recognize an image and predict to which class the image belongs, thanks to image classification methods. For example, AR systems can know which books users are reading or which brand a product belongs to based on logo classification.

Nevertheless, both markers and natural images are printed templates and fixed in their contents. The contents of these kinds of templates are created offline, not in real-time. That means as being used as an input for an AR system to detect, these templates' contents cannot be changed by users to make new templates corresponding to different virtual augmented information. In this thesis, we suggest integrating the user interaction into the detection phase of AR so that the way of creating AR templates for detection becomes more flexible and brings more exciting experiences to users. To this end, our proposed system allows users to sketch themselves pictures of certain things (i.e. objects, animals, plants) instead of using printed predefined templates. The system then recognize those images in real-time and immediately display the corresponding virtual information for users. In other words, we would like to replace the printed predefined template as a traditional input of AR for detection with a new means of input which is users' sketches.

Sketch recognition is not a new issue. This problem has been studied by many researchers [53, 46, 16]. Sketch recognition can be viewed as instance recognition problem or class recognition one. The first can be dealt by feature matching and geometric alignment strategies commonly used in Computer Vision. The latter, which is also known as category-level or generic object recognition, can be resolved by image classification approaches from Machine Learning. We prefer to

approach sketch recognition in AR as a class recognition problem.

There are 2 reasons why sketch recognition in AR should be viewed as a problem of class recognition instead of instance recognition. The first reason is because the preferred question is to which class a sketch belongs, not which instance a sketch is. User's sketches vary enormously but they portray a finite set of classes. The second reason is because sketches in a class can have different spatial arrangements of features. Instance recognition relies on the processes of matching two images' features and then aligning the two sets of found matching features. That is why two images cannot match if the spatial arrangements of their matching features are inconsistent. Class recognition methods in Machine Learning can overcome this drawback of instance recognition strategies because these methods pay more attention on global image's descriptor representing image's contents than spatial arrangement of image's features.

Class recognition has been deeply studied for a long time. Powerful recognition methods in the field of Machine Learning have been developed to classify various kinds of patterns such as handwritten texts, gestures, speeches, objects and scene images with high accuracy. Applying recent advanced classification methods in Machine Learning, we develop a smart environment with AR in which users' sketches are used as the input to be recognized.

1.2 Motivation

There are 4 reasons why we want to develop our proposed system:

First, AR is one of the hottest trends in the field of HCI and promises to be widely applied in daily life. Google Glass, the most innovative and well-known product of AR, is attracting a lot of interest from both end-users and researchers.

Second, an AR system not only needs to provide more virtual information for users but also needs to involve users' interactions to enhance users' exciting ex-

periences.

Third, the traditional input of printed predefined template in an AR system should be replaced with users' sketches so that users can easily customize the input in real-time.

Fourth, recent powerful methods of image classification in the field of Machine Learning should be applied to our proposed system to deal with the problem of sketch recognition.

1.3 Objectives

This thesis has two main objectives:

- Study and compare some recent approaches of image classification in Machine Learning in order to select the most efficient approach among them for building a sketch recognition module.
- Propose and develop a smart environment of Augmented Reality in which users are provided with extra multimedia and social-media information corresponding to sketches created by themselves, based on the built sketch recognition module.

To fulfill this objective, we have to take the following tasks:

- Study terms of HCI, history of development to know about HCI.
- Study terms of AR, history of development, properties of an AR system as well as detection methods in AR to have a background on AR.
- Study the problem of sketch recognition and its known approaches.
- Study, implement, and conduct some experiments with image classification methods in Machine Learning to find out which method suits the problem of sketch recognition.
- Develop an AR system with users' sketches as the input.

1.4 Outline

- Chapter 2. We briefly introduce HCI, AR as well as their various applications.
- Chapter 3. We give an overview of sketch recognition problem and image classification methods in Machine Learning.
- Chapter 4. Our conducted experiments are presented and the evaluation results are shown.
- Chapter 5. The details of our proposed AR system are described.
- Chapter 6. We conclude the thesis and plan some further research.

Chapter 2

Backgrounds and Related works

In the chapter, we provide the following features.

- An overview Human-Computer Interaction (HCI) including definitions, terminologies and current systems. In addition, the related knowledge is for HCI system design consist of unimodal and multimodal.
- An interesting approach using sketch as an input for HCI is briefly introduced. Additionally, we also introduce to Augmented Reality, which provide interface as well as smart environments of HCI.

2.1 Human-Computer Interaction (HCI)

Procedures designed for interaction between human beings and computer are enhanced in recent decades [1]. The usual interaction based on external device like keyboard, mouse, etc. However, recent studies have focused on multimodal than unimodal HCI and effective communications are simple orders [1]. Unimodal and multimodal HCI are presented in section 2.1.3 and 2.1.4 respectively. Thank to achievements of technology, the distance between human beings and computer is more and more narrowed. Many new interactive forms are introduced and many productions are released.

2.1.1 Techniques of Human-Computer Interaction

Nowadays, techniques can design HCI system based on Human's sense. There are three typical types consist of visual perception, Hearing, and Haptic perception.

The most of devices are used for input is visual perception, and these devices are binary or pointer forms. The binary form devices usually use buttons on keyboard. The pointer form devices such as mouse, stick controller, or touch pen are the most popular. The output devices can be pictures or printers.

The input devices of Hearing require module of Speech Recognition [49], which can help the sound input and output devices such as speaker, warning system, GPS, etc become easier. However, the system for speech recognition is quiet hard to implement.

The devices with expensive price such as haptic interact with skin as well as muscle by through touching, force. The devices of haptic is often used for virtual reality or disability.

Recent technologies wish to develop combining many interaction ways and utilize available modules such as internet and cartoon. These technologies can be divided into three main kinds: mobile devices, wireless device, and virtual device. For example, Global Positioning System (GPS) use for supported devices in military such as infrared projector, Personal digital assistant (PDA). Application for virtual tour is used to watch flats without presentation.

2.1.2 Advanced techniques of Human-Computer Interaction

There are many ways for communication human beings and computer. Compute can recognize handwritten letters of human to convert into the corresponding letters. Further more, human beings can interact with computer by voice, move-

ments, etc and even brain. Interface is used for interaction includes input and

Figure 2.1: Dr. Peter Brunner presented simulation interactive system computer can understand what a human being thinks [1].

output data procedure. In addition, interaction is conducted by different specific channels, communication. There are many channels for connection user want. These connection channels is called as a modality [27].

In the next section, we want to introduce two typical modality HCI of unimodal and multimodal.

2.1.3 Unimodal Human-Computer Interaction

A procedure based only on single channel is called unimodal. Based on modalities for connection, there are three main kinds:

- Visual - based.
- Audio - based.
- Sensor - based.

HCI based on visual senses is concentrated to study []. Some interesting fields for development are as follow.

[1]Image source: http://computer.howstuffworks.com/brain-computer-interface.htm/printable

- Face analysis.
- Tracking body movements.
- gestures recognition.
- Eye movements recognition

To application of face analysis as well as expression recognition [12, 19, 45], the application recognize expression of human to make suitable actions. In tracking body movements as well gesture recognition, the application help user order for requirement. The applications of eye movements recognition is used for disabilities. Fig. 2.2 presents the disability to use computer based on eye moments recognition.

Figure 2.2: Eye moments recognition system for disability.

HCI based on hearing is also a significant field The system can be divided into the following groups:

- Speech recognition.
- Expression analysis of listener.
- Music communication.

Studies about speech recognition [7] has been focused for a long time. Recently, research is about human expression in smart environment is also focused on [7]. An interesting field based on music communication is applied for industry of art including sound and visual perception [37].

2.1.4 Multimodal Human-Computer Interaction

Multimodal HCI is system use combining many connection channels. One of the most popular system is the system combining gestures and speech [43]. The important thing of multimodal HCI is the interactive support of channels for recognition. For instance, based on lip moments (visual-based) can support speech recognition (audio-based). Speech recognition (audio-based) can support action recognition (visual-based).

Figure 2.3: Project of Natal Xbox 360 [2].

There are many application of multimodal HCI system. The applications support disability to communicate with computer by actions, head, eyes, and speech. Based on current expression human, system can turn on different types of music like pop or rock music. Computer also recognize expression based on voice with specific change of amplitudes. In healthy, Neuro-Surgical Robotics with components including hand,, feedback vision sensor, controller. Sensor handle signal

[2]Image source: http://www.stuffwelike.com/stuffwelike/2009/06/14/project-natal-ismicrosofts-next-console/

for surgeon continuously to instruct for communication of human and computer. In entertainment, thank to action recognition, the application of Natal project on Xbox 360 allow user to control without including devices.

In this work, Human being can interact with computer through sketches as a visual-based modality HCI. User can sketch what they imagine about objects. To be able to visualize information from sketches input, interface as well as smart environment is also important. We propose sketches as input for Augmented Reality (AR) smart environment. The next section is about description of 2D sketch as an input for HCI with Augmented Reality smart environment.

2.2 2D sketch as an input for HCI

User can interact with computer through sketches as an input. Augmented Reality provide interface as well as smart environment for HCI. Based on sketch of user, Augmented Reality will augment virtual information (e.g. multimedia or social media) for contexts in real world. In this section, an overview as well as frameworks and application of Augmented Reality is presented as follow.

2.2.1 Augmented Reality

Introduction of Augmented Reality

From handling with reality information, system provides augmented information which relates to current real contexts. This is not only to help user get information of reality but also to allow them interact with virtual objects which is augmented in visualization. Fig. 2.4 presents Reality and Virtual Reality at many different levels. Users only see virtual objects in Virtual Reality. Augmented Reality is applied to combine Virtual information for Reality [2].

Augmented Reality is to augment virtual information in reality make user be interested by visualizing lively virtual objects [2]. A suitable Augmented Reality visualizes virtual information on reality contexts or real space which users are

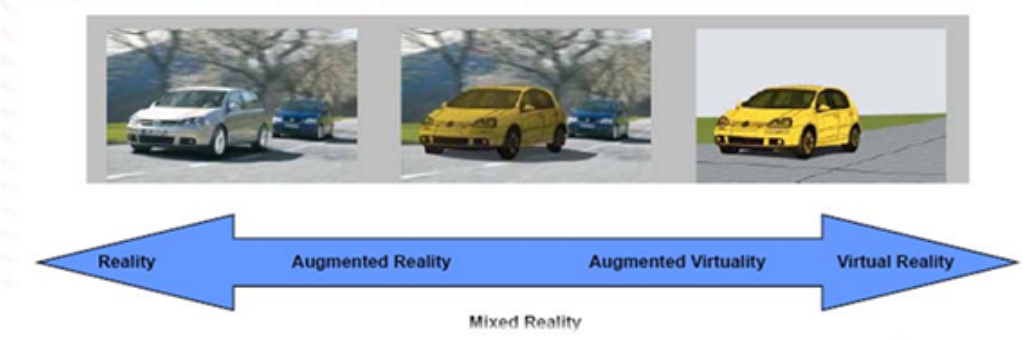


Figure 2.4: Relation between Reality and Virtual Reality [28]

able to observe reasonably. Thus, users can not see any distinction between virtual objects and real contexts. This is also main purpose of Augmented Reality to remove distinction between Virtual objects and reality and improve awareness and interaction between human beings and reality [24].

Figure 2.5: Temple 3D model in Greece augmented to natural sense of real one [3]

.

Fig. 2.5 is shown to augment 3D model object to real context of Greece temple. Virtual Information is augmented on contexts has many different kinds. However, there is two main kinds including multimedia (texts, picture, video, 3D models, sound, etc) and social media (phone number, web-page, comments in internet community, etc). In addition, Augmented Reality can be improved by integrating interaction between human beings and virtual objects.

[3]Image source: http://www.theposthole.org/read/issue/6

The properties of Augmented Reality [2] is as follow.

- Mixing between reality and virtual objects reasonably.
- Recording user's action to update information in real time and make distinction between Augmented Reality and Computer graphics[52].
- Visualization is the closest to the most of natural view.

Brief history of Augmented Reality

The idea of Augmented Reality appeared firstly in the 1930's when computer had been introduced and was popularly used in the 1990's [28]. Until the 1960's and 1970's, computer companies began working on visual environments of Augmented Reality. In the late 1950's, a cinematographer named Morton Helig designed a simulator device with name of Sensorama [4]. This device is used to simulate sounds, vibration, and taste. In 1966, head mounted display was introduced at the first time and was designed by Ivan Sutherland. This device allowed user interact with around environments directly.

Brief history of Augmented Reality can be presented on overview of Table 2.1 based on Ig-Jea Kim's presentation at SIGGRAPH Asia conference in 2010 [28].

Practical application of Augmented Reality

Augmented Reality system provides many different kinds of information such as texts, pictures, video, etc. These information can be used for many different fields including education [24], healthy [25], geology [39], art [29].

Augmented Reality also provides basic information such as heart beat, blood pressure, and status of parts for surgeons. This can help doctors diagnose status of patents quickly. Say for example, Augmented Reality can be used for X-Rays imaging based on real image from echocardiography [41] as well as observing

[4]Online available: http://www.mortonheilig.com/InventorVR.html.

Year	Event
1968	Ivan Sutherland designed Augmented Reality system of head mounted display at the first time.
1992	Tom Caudell and David Mizell introduced the term “Augmented Reality”
1996	Jun Rekimoto introduced maker with 2D matrix. These markers allow camera to determine the six arbitrary points region.
1997	Stevie Feinner introduced Touring Machine, which is the first Mobile Augmented Reality System (MARS).
1999	H.Kato and M.Billinghurst introduced ARToolkit.
2000	B.Thomas introduced AR-Quake, which is the extension editor of game on the famous Quake computer.
2003	R. Raskar introduced iLamp, which derived the origin for augmenting information with camera system.
2006	Nokia introduced MARA, mobile device can presented Augmented Reality with many sensors.
2009	SixthSense Project of Massachusetts Institute of Technology (MIT) used devices based on Augmented Reality.

Table 2.1: Benchmarks of Augmented Reality

status of baby in body of pregnant woman [5]is shown in Fig. 2.6.

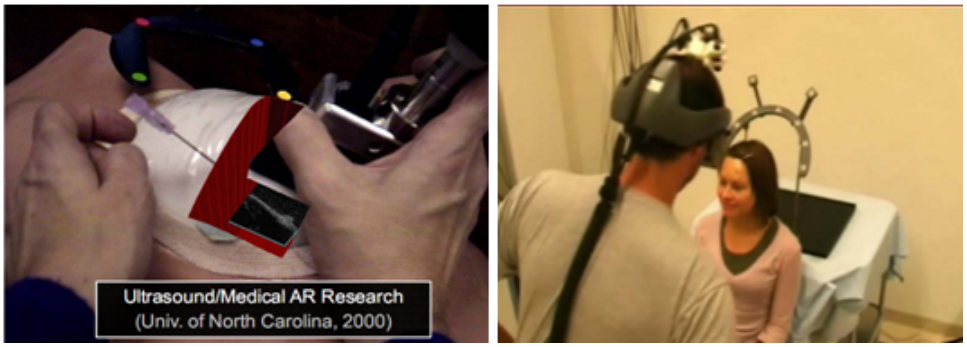


Figure 2.6: Combining Echocardiography and Augmented Reality. [28]

[5]Online available: http://www.cs.unc.edu/Research/us/.

Recently, Augmented reality also has been applied in film-making industry. To get creature of film, contribution of Augmented Reality in field of entertainments is very significant. Especially, creatures about science fiction, mixing real characters and cartoon characters in film named "Who Framed Roger Rabbit", or film about super heroes, etc.

Figure 2.7: The film "Who Framed Roger Rabbit" mixing real and cartoon characters [28].

An interactive game Play Station Move, Wonderbook of Sony brought the lively magical world for users from normal books only. PS Move is a magic wand help users enter the enchanted world of dragons, fairies and spell caster. Expo is also the significant field for Augmented Reality. Instead of coming far away place of North pole to discovery natural life of animals, using Augmented Reality technology with 3D models, we can observe life of species in North pole. Additionally, Augmented Reality system also provides information about species extinction which did not exist in the earth and interacts with these species. This still has many disadvantages in current time of Expo.

In education, Augmented Reality helps learning and teaching become more intuitive and attractive. AR application "Learngears" help learners study about the planets of solar system or geology by interacting with 3D models of the Earth which is visualized from a book.

A project in our lab called "Smart Interactive Book" is able to display augmented content of an arbitrary book contains markers which are figures. The

augmented content is the field of astronomers help learners be able to visual the activities of planets in solar system easily. In addition, "Smart Interactive Book" also allow learners to interact with planets as well as virtual objects by using hand touching.

Figure 2.8: Wonderbook [6].

(a) BBC Frozen Planet AR.

(b) Natural Geographic Channel Live AR.

Figure 2.9: Augmented Reality (AR) at Expo[7].

[6]Image Source: http://www.engadget.com/2012/06/04/sony-wonderbook-ps-move/.
[7]Image Source: http://vimeo.com.
[8]Available online video: http://www.youtube.com/watch?v=iT2ek8N0VlY.
[9]Available online video: http://www.youtube.com/watch?v=3K6GXECbTxM.

Figure 2.10: Augmented Reality Leanergears [8].

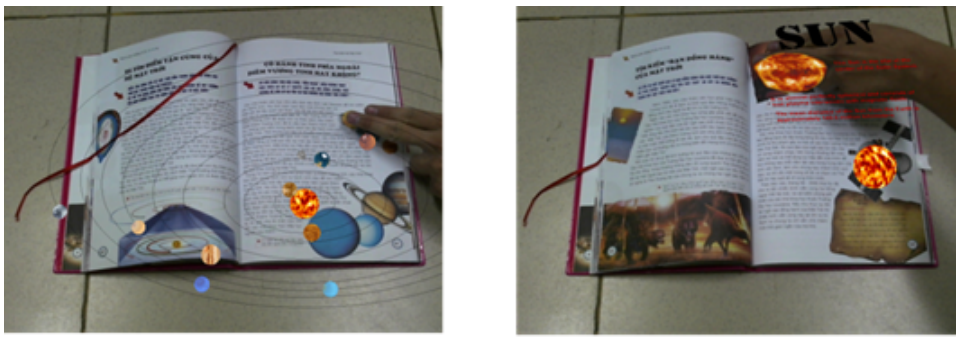


Figure 2.11: Smart Interactive Book [9].

Researchers of IBM Labs study about the way to help shoppers be able to reach to production and transactions through shopping application of Augmented Reality. This application can send coupons, greetings, comments, and detail information of production for customers. Similarly, A study in our Labs "Smart Shopping Assistant" [42] allows users to use smart phone to search necessary information about production.

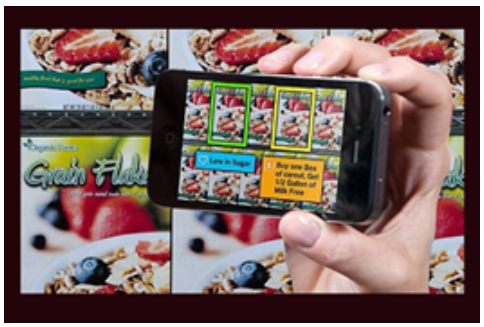


Figure 2.12: Smart Shopping Assistant.

2.2.2 Frameworks using Augmented Reality

Object tracking is a significant component in applications of Augmented Reality. Tracking in application of Augmented Reality usually require the higher complexity and handle in real time. There are two popular tracking frameworks: localization and visual information [28].

Framework using localization

Tracking based localization is widely used in Global Positioning System (GPS). Using this technology in tracking help users use Augmented Reality even in natural context. However, because of global system, the accuracy is quiet low. In addition, some frameworks tracking objects based on position such as Cell-ID, Ultra Wide Band, Wifi Positioning System [28].

Framework using image processing

In case of using stable context, users do not be allowed to move flexibly. Framework using camera is a possible solution, because region for tracking is smaller than region in framework using localization. There are two important things for frameworks using camera, which are camera configuration and tracking algorithms. With enhancements of camera as well as optimization method for tracking algorithms, results achieve accuracy closes to the state-of-the-art benchmarks. There are two main kinds of frameworks using camera including Marker tracking and natural feature of objects. *Marker pattern*

This framework is used to predict camera pose with respect to object based on marker [29], specific image [32], or bokode [40]. Marker is special pattern such as barcode, QR-code, BCH-ID marker. Markers connect to target object. Frame from real world is recorded, after that, region contains markers is extracted. regions are compared with suitable patterns. Time for recognizing a marker is linear with respect to the number of markers. Therefore, the efficient of framework is very high in case the great number of marker in image.

ARToolKit [39], ARToolKitPlus [61], and ARTag [20] are famous tools used widely in practical of Augmented Reality. ARToolKitPlus supported two kinds marker for ID-Marker: Simple ID-Marker and BCH ID-Marker. Markers is created by 6×6 matrix. Simple ID-Marker encode a number of 9 bits for 6×6 matrix, the number of supported patterns is 512 including ID (from 0 to 511). BCH ID-Marker use Cyclic Redudancy Check algorithm, which encoded a number of 12 bits for 6×6 matrix. BCH ID-Marker also supports 4096 patterns and ID is from 0 to 4095. The number of supported markers increase follow size of matrix.

Using markers usually achieve the high speed handling and accuracy. In addition, these markers can be included for arbitrary targeted objects. However, the newspapers or magazines usually do not have space enough to include markers. Including markers to pages make reader not feel comfortable.

(a) Barcode. (b) QR-code. (c) BCH ID-Marker.

Figure 2.13: Patterns used in Augmented Reality (AR).

Natural pattern image

For example, images can be cover of newspapers or magazines. Natural image based matching is the typical framework use sub-pattern in a large image (e.g., frames captured from camera). There are two mature approaches of template matching: local feature-based template matching and template-based template matching.

Approach of template matching uses color information is majority factor to determine the different levels within images. This difference could be between a large image and region parts from original image. The approach uses some quantity to distinct pattern such as Sum of Squared Differences (SSD), Sum of Absolute Differences (SAD) [34]. Frameworks of template matching uses simple regions. Further more, this approach can handle with simple as well as complicated image. However, the disadvantage is variant orientation of image such as scale, rotation, and pitch. Thus, using this framework is impossible in our context.

The approaches use features such as edges [55], angle [8], or blob [36, 4]. There are two steps for handling of local feature including detecting and extracting keypoints of images. With this approach, we can apply in recognition object because the advantage is variant orientation. However, the disadvantage is very large complexity. There are some optimization methods like Randomized Tree and Ferns [44] but the algorithm require the great amount of time in training process. Combining recognition with tracking is complex problem because of the high complexity and only using for the tiny of objects.

Image classification

Problems of frameworks mentioned in the above sections are to detect whether frame image recorded contains patterns (Marker or natural patterns) and its position. Thus, pattern in image frame can be lacked of an arbitrary part with respect to original pattern. However, relation about space between these components are still reserved about presentation.

In image classification algorithms, a image can be predicted to a label of class if image contains similar features with with original images in that class. Eventually, predicted images do not reserve relative space with original images.

Image classification can be used to recognize patterns appeared in a large im-

age whether pattern belong to which class, i.e. figuring out topic of patterns. Therefore, the main question of frameworks in the above sections is "What pattern does frame image contain ?". In frameworks using image classification, the question is "What topic is image ?".

For instance, to make know which pages in enchanted book "Snow White and the Seven Dwarfs" for readers, this framework recognizes natural patterns which are images from book. In image classification of patterns in book, features are extracted for classifying. Thus, reading an arbitrary page of book can conclude for reader that they read what topics of book without the specific page.

With achievements of AR frameworks, AR is applicable of smart environment for HCI. In this work, specially, we propose 2D sketch as an input for Augmented Reality in HCI system. This make user obtain interesting experiences with normal 2D sketches while virtual objects is lively visualized at the same time.

2.3 Summary

Human-Computer Interaction with Augmented Reality smart environment makes the distance between human beings and computer vanish. With Multimodal HCI system, people can combine many information such as images, audio, video, 3D models animation, etc to establish a hierarchical framework for augmenting in real world. In addition, Augmented Reality also provide user interesting experiences to enjoy valuable features. In this thesis, we develop Multimodal Human-Computer Interaction using input 2D sketch images for Augmented Reality smart environment combine with many connection channels such as audio, video, texts and 3D models. The system allow user to be able to interact with computer by relative sketch objects, and recognize objects to augment necessary virtual information. In the next chapter, we will introduce about sketch recognition module which is the main factor in the whole framework.

Chapter 3

Sketch Recognition

3.1 Sketch Recognition Approaches

Sketch data is one of visual figures users can manipulate with computer flexibly. Sketch recognition is one of interesting parts in pattern recognition as well as computer vision. Many approaches achieves promising results and can be applicable to practical frameworks. In a research of Akshay [5], method based on values of entropy as a parameter to classify sketches which can be texts or charts. Another method of Dean Rubine [50] defined 13 distinct features for sketch representation. The disadvantage of these methods is the significant affection of versions in sketching to accuracy in recognizing because sketches can be lacked of an arbitrary part by hiding.

Some methods followed the idea of retrieval and synthesis for sketches. A image can be considered to content-based retrieval [15]. However, this can not achieve semantic understanding because of ignoring learning from sample of sketches. Synthesis systems users can sketch to generate the huge amount of data to offset the problem of geometric between sketches and image content [17] or allow users include texts, labels, contexts around, etc for sketches [10].

Several approaches of selecting geometry features or static image features to classify categories of images. An approach of Lee et al. [33] used fast nearest neighbor matching to figure out similarity of geometry in edge of objects. Simi-

larly, a proposed method of Heeyoul Choi [11] based on multiplication of Isomap to compute the difference within sketches.

Combining ideas of synthesis system with geometry feature [13] get higher accuracy than conventional methods because representation of images is more detail and can be considered in a larger space of features.

In series of researches [53, 46], understanding human beings try to sketch input at higher level is quite significant. System identity automatically small patches of strokes types such as lines, circles, and arcs from noisy user input achieved good performance in real time.

A new baseline of sketch recognition is that of trying to discover how human beings sketch objects and how well human beings and computer can recognize sketches [16]. Based on common benchmarks, they try to define taxonomy of objects. On LabelMe dataset [51], selecting 1000 labels appear with the most frequent. After that, removing labels of objects which can be duplicated such as context around, position change, or combine with another object. Similar to Princeton Shape benchmark [56] and Caltech-256 [23], selecting typical categories for sketching. By asking to request members to make filtering categories of 250 objects, this is baseline for generating new dataset of human sketch. In addition, by using algorithm about feature representation of images combine supervised learning like Support Vector Machine for classification, they achieved promising results and a framework in real time.

To summarize, the problem of sketch recognition can be viewed as either instance recognition problem or class recognition one. The first can be dealt by feature matching and geometric alignment strategies commonly used in Computer Vision. The latter, which is also known as category-level or generic object recognition, can be resolved by image classification approaches from Machine Learning. In this thesis, we prefer to approach sketch recognition in AR as a class recognition problem.

There are 2 reasons why sketch recognition in AR should be viewed as a problem of class recognition instead of instance recognition. The first reason is because the preferred question is to which class a sketch belongs, not which instance a sketch is. User's sketches vary enormously but they portray a finite set of classes. The second reason is because sketches in a class can have different spatial arrangements of features. Instance recognition relies on the processes of matching two images' features and then aligning the two sets of found matching features. That is why two images cannot match if the spatial arrangements of their matching features are inconsistent. Class recognition methods in Machine Learning can overcome this drawback of instance recognition strategies because these methods pay more attention on global image's descriptor representing image's contents than spatial arrangement of image's features.

Class recognition has been deeply studied for a long time. Powerful recognition methods in the field of Machine Learning as well as Neuroscience have been developed to classify various kinds of patterns such as handwritten texts, gestures, speeches, objects and scene images with high accuracy. To deal with the problem of sketch recognition, we focus only on the problem of image classification simply because sketches are essentially images.

3.2 Sketch Recognition with Machine Learning

Sketch recognition can be done by using image classification approaches in Machine Learning. In image classification, an image is classified according to its visual content. Specifically for the problem of sketch recognition, each sketch of users is regarded as an image, which is then classified in different ways corresponding to different approaches. To choose the appropriate approach for the sketch recognition module in our proposed system, we try five image classification approaches in Machine Learning and evaluate their performances. We divide these approaches into two groups: one using global descriptors to encode

images, and the other one using local descriptors to encode images. Although used techniques and schemes may vary slightly from approach to approach, all of these five approaches follow two main processes: training process and testing process.

The training process is briefly illustrated in Figure 3.1. In this process, we have a training set of m sketch images $S = \{s^{(1)}, ..., s^{(m)}\}$ and a list of labels $Y = \{y^{(1)}, ..., y^{(m)}\}$ where $y^{(i)} \in \{1, ..., n_{class}\}$ is the class label of the sketch image $s^{(i)}$ and n_{class} denotes the number of classes. For example, $y^{(i)} = k$ indicates the sketch image $s^{(i)}$ belongs to the class k.

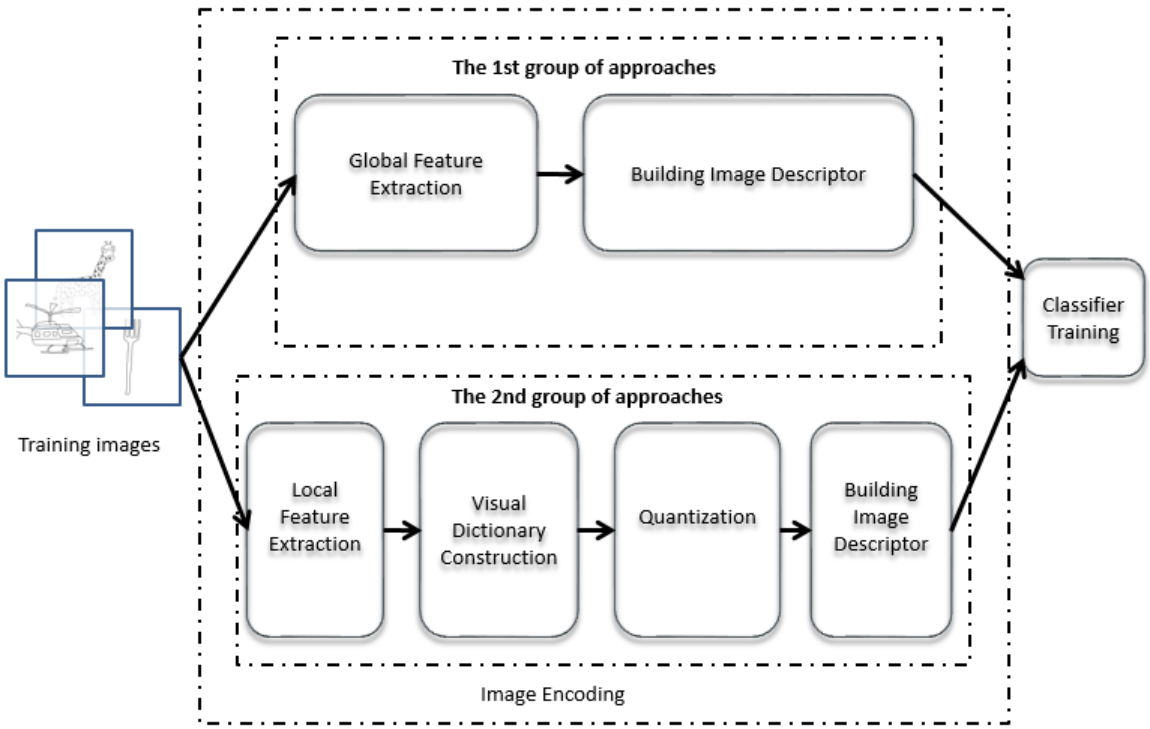


Figure 3.1: The training process for image classification.

The training process consists of two key stages as follows:

- **Image Encoding**: The goal of this stage is to encode the information of the image $s^{(i)}$ into an M-dimensional image descriptor $x^{(i)}$ which represents

the entire image. After this stage, we have a list of training examples $\{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$ where $x^{(i)} \in \mathbb{R}^M$ is the descriptor of the image $s^{(i)}$ and $y^{(i)}$ is the class label of the image $s^{(i)}$, ready for training a classifier in the next stage. Based on the scheme of encoding, we divide the five approaches into two main groups.

- The first group of approaches, including Approach 1 and Approach 2, treat an image as a whole. An image is regarded as a grid of pixels in which each pixel is represented by a floating point number indicating the grayscale intensity at that location. The encoding stage includes two sequential steps:
  - *Global Feature Extraction*: In this step, this group use the entire image, which includes all pixels as the input, as the global feature. As a result of extraction, we obtain the grid of all pixels per image.
  - *Building Image Descriptor*: The grid of pixels for each image are then encoded into a representing vector, i.e. image descriptor, by simply being unrolled (Approach 1) or also going through a Sparse Autoencoder (Approach 2). All image descriptors are used for training a classifier in the classifier training stage.
- The second group of approaches, including Approach 3 - 4 - 5, considers an image a collection of image patches and encodes the image by describing the image in terms of a set of common visual patches. More specifically, the encoding stage is divided into four sequential smaller steps:
  - *Local Feature Extraction*: This step aims to extract local patches that characterize the image and to describe those patches. There are two strategies of extracting local patches that are commonly used in the problem of image classification. The first one is extracting the local patches around the interest points of an image, e.g. Harris corners or SIFT interest points, which are called

sparse features. The second strategy is extracting the corresponding patches of densely sampled keypoints over the entire image, which are called dense features. We prefer the second strategy for recognizing sketches because the first one based on interest points is only suitable for texture or scene images, which have many salient points, not for sketch images. The number of interest points per sketch image is too small to represent the whole sketch. Therefore, we use only dense features, specifically dense SIFT, for all approaches of this group. Next, the patches around the features are described by vectors of identical length, e.g. 128 for SIFT feature. These vectors are called feature descriptors. After this step, we obtain a set of feature descriptors for each image.

* *Visual Dictionary Construction*: The number of all training images' features is very large. However, they fall into a much smaller number of groups of similar visual features in the feature space of training images. Each group is represented by a visual word, i.e. a visual feature that describes the common characteristics of the group. The goal of this step is to cluster the visual features of all training images into a number of clusters, each of which corresponds to a visual word. The most popular clustering algorithm is K-Means. Therefore, K-Means algorithm is used for all approaches. This step results in a visual dictionary formed by all visual words.

* *Quantization*: As we have a dictionary of visual words, an image should be regarded as a collection of visual words, i.e. Bag-of-Words model, instead of discrete features. For this reason, each feature of an image must be quantized to the set of visual words. There are two kinds of quantization. The first one is hard quantization in which a feature can be assigned to only one visual word,

as Vector Quantization used in Approach 3. The second kind is soft quantization in which a feature can be assigned to multiple visual words, as Locality-constrained Linear Coding used in Approach 4 and Kernel Codebook Encoding used in Approach 5. After this step, each feature is represented in terms of the visual words.

* *Building Image Descriptor*: The number of features vary greatly from image to image. Therefore, to equalize the lengths of image descriptors representing different images so that we can compare two images in the classifier training stage, we have to pool the features of each image by an average pooling function on a single layout of the entire image (Approach 3 and Approach 5) or a max pooling function on a spatial pyramid layout of image (Approach 4) to build the final image descriptor for the image. This step finalizes the image encoding stage and every training images is represented by the corresponding image descriptor of a fixed length, ready for training a classifier in the next stage.

- **Classifier Training**: In this stage, the resulting image descriptors from the image encoding stage are inputted to a classifier so that the classifier can learn one or some classification models depending on the approach. The most popular classifier is Support Vector Machine (SVM). Thus, we also use SVM for all approaches except Approach 1 in which a Neural Network is used as a classifier. Among the approaches with SVM, we only use a non-linear SVM with the Gaussian kernel for Approach 5 and use linear SVMs for the others. After this stage, the training process finishes and results in one or some trained classification models, ready for the testing process.

In the testing process, a testing image is also processed through the feature encoding stage as in the training process without the step of visual dictionary construction. In addition, in the stage of classifier training, we do not need to

build classification models again but apply these ones to the testing image in order to predict which class the image belongs to.

A summary of five classification approaches that we evaluate in this thesis is shown in Table 3.1. The detail of each approach is described in the following sections.

Appr.	**Image Encoding**				**Classifier**
	Global Feature			*Building Image Descriptor*	
No. 1	Grid of all pixels from image			Unrolling	Neural network
No. 2	Grid of all pixels from image			Unrolling + Sparse Autoencoder	Linear SVM
	Local Feature	*Clustering*	*Quantization*	*Building Image Descriptor*	
No. 3	Dense SIFT	K-means	Vector Quantization	Average pooling on spatial pyramid	Explicit feature map + Linear SVM
No. 4	Dense SIFT	K-Means	Locality-constrained Linear Coding	Max pooling on spatial pyramid	Linear SVM
No. 5	Dense SIFT	K-Means	Kernel Codebook Encoding	Average pooling on single layout of entire image	Non-linear SVM with Gaussian kernel

Table 3.1: Approaches of image classification for evaluating.

3.3 Approach 1 - Using Neural Network

This approach falls into the first group of approaches, which treat an image as a whole. The idea of this approach is very obvious. Providing we do not any special kind of feature to extract for our images, we simply retain all of the image information by getting the grid of all pixels per image. Since the obtained pixels are raw data, we use a neural network to classify images based on their pixels. A neural network is able to classify raw data because the information processing paradigms of neural networks are inspired by the way biological neural systems process data.

Specifically, in this approach, the grid of pixels for each image is first unrolled into a representing vector. The vector is then used as an input to a neural network for learning a classification model. The flow of the approach is shown in Figure 3.2.

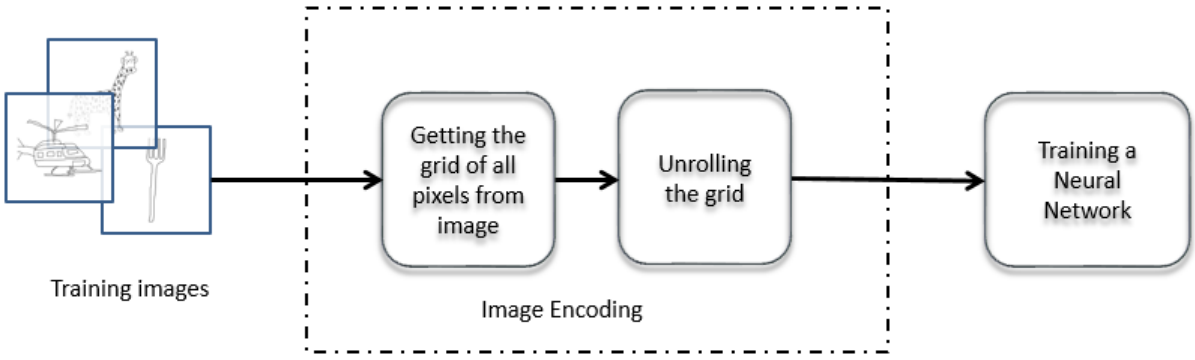


Figure 3.2: The flow of Approach using Neural Network.

3.3.1 Global Feature Extraction

Let n_{pixel} denote the number of pixels in an image. The number of pixels per image is typically large. If they are inputted to a neural network, the training time is very slow. Therefore, all images should be resized to a small size (e.g. 32×32 or 64×64).

3.3.2 Building Image Descriptor

The image descriptor is the unrolling of all pixels, so it has the length equal to the number of pixels, i.e. $x^{(i)} = [x_1, ..., x_M], i = 1, ..., m; M = n_{pixel}$ where $x^{(i)}$ indicates the i-th training example.

3.3.3 Classifier Training

Neural networks are used for learning classification model in this approach. We first briefly describe the neural network model constructed for our problem of sketch recognition and then show how we train a neural network from the training examples and how a trained neural network is used to predict the class label for a testing example.

Neural Network Model

A neural network has 3 layers: an input layer, several hidden layers, and an output layer. For simplicity, we use a neural network with only one hidden layer in our problem. Let L_1, L_2, L_3 denote the input layer, the hidden layer, and the output layer respectively. We also use $u_i^{(l)}$to denote the i-th unit of layer L_l.

For our problem, the input layer L_1 consists of n_{in} units and unit $u_i^{(1)}$ takes element x_i of image descriptor x as input, which is also the input of the neural network. The output layer L_3 consists of n_{out} units and unit $u_i^{(3)}$ outputs the probability that the training example x belongs to the class labeled as i, which is also the output of the neural network. The hidden layer L_2 consists of n_{hid} units and each unit $u_i^{(2)}$ plays the role of a neuron which receives its input computed from the outputs of all units in the input layer and uses its output to compute the inputs of every units in the output layer.

Our neural network has parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, where $W_{ij}^{(l)}$ denotes the parameter (or weight) associated with the connection between unit j in layer L_l, and unit j in layer L_{l+1}. Thus, we have $W^{(1)} \in \mathbb{R}^{n_{hid} \times n_{in}}$ and $W^{(2)} \in \mathbb{R}^{n_{out} \times n_{hid}}$. Furthermore, we add one bias unit to the input layer and the

hidden layer as an intercept term. The bias units, which is not counted in the number of units in a layer, don't receive inputs and always output the value +1. Let $b_i^{(l)}$ denote the bias associated with unit i in layer $l+1$, so $b^{(1)} \in \mathbb{R}^{n_{hid} \times 1}$ and $b^{(2)} \in \mathbb{R}^{n_{out} \times n_{hid}}$.

We use $z_{i(l)}$, $a_{i(l)}$ to denote the input and output value of unit $u_{i(l)}$ respectively. Output value $a_{i(l)}$ is computed from input value $z_{i(l)}$ as $a_{i(l)} = f(z_{i(l)})$ where $f : \mathbb{R} \to \mathbb{R}$ is called the activation function. In our problem, we choose f to be a sigmoid function:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{3.1}$$

The visualization of a sigmoid function is presented in Figure 3.3.

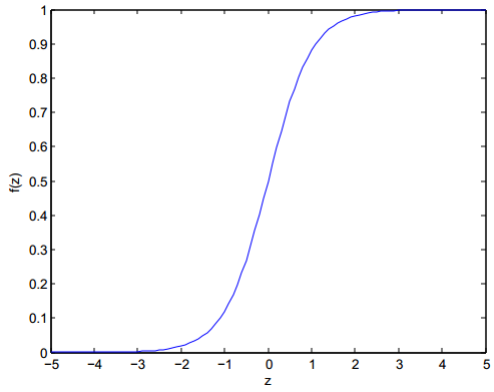


Figure 3.3: Visualization of a sigmoid function whose range is $[0, 1]$.

Formalizing above descriptions, the computation that this neural network represents is given by:

$$\begin{aligned} z_i^{(2)} &= \sum_{k=1}^{n_{in}} W_{ik}^{(1)} a_k^{(1)} + b_i^{(1)} \\ &= W_i^{(1)} a^{(1)} + b_i^{(1)} \end{aligned} \tag{3.2}$$

$$a_i^{(2)} = f(z_i^{(2)}) \tag{3.3}$$

$$
\begin{aligned}
z_i^{(3)} &= \sum_{k=1}^{n_{hid}} W_{ik}^{(2)} a_k^{(2)} + b_i^{(2)} \\
&= W_i^{(2)} a^{(2)} + b_i^{(2)}
\end{aligned}
\tag{3.4}
$$

$$
a_i^{(3)} = f(z_i^{(3)}) \tag{3.5}
$$

Unit $u_i^{(1)}$ in the input layer L_1 do not have input value $z_i^{(1)}$, so the activation of the input layer takes value from the input training example:

$$
a^{(1)} = x \tag{3.6}
$$

$$
n_{in} = M = n_{pixel} \tag{3.7}
$$

The hypotheses $h_{W,b}(x)$, with parameters W, b that we can fit to our data is determined as:

$$
h_{W,b}(x) = a^{(3)} \tag{3.8}
$$

The neural network for our problem is illustrated in Figure 3.4.

Training

Suppose we have a fixed training set $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$ of m training examples, where $x^{(i)}$ and $y^{(i)}$ are the image descriptor and the corresponding label of image $s^{(i)}$ respectively. Since $a^{(1)}$ is set to $x^{(i}$, image descriptor $x^{(i}$ must be normalized to the range $[0, 1]$ before being inputted to the neural network.

Also, recall that $h_{W,b}(x)_k$ is set to $a_k^{(3)}$ whose value is in the range $[0, 1]$. Although a class label takes value from $1, 2, ..., n_{class}$, for the purpose of training a neural network, we need to record the training labels as vectors containing only

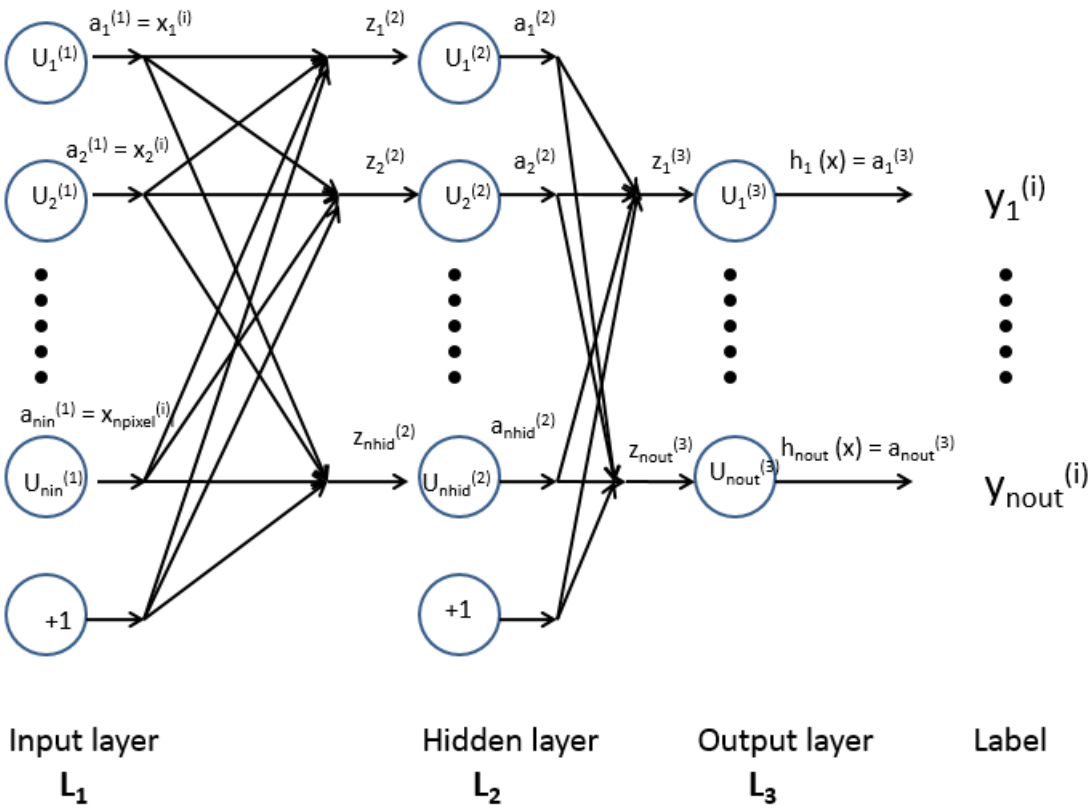


Figure 3.4: The Neural network model for our problem.

values 0 or 1, so that:

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, ...or \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{3.9}$$

Specifically, if sketch image $s^{(i)}$, whose image descriptor is $x^{(i)}$, belongs to class k, then the corresponding $y^{(i)}$ should be a n_{class}-dimensional vector with $y_k = 1$, and the other elements equal to 0.

After normalizing all training examples, we can train our neural network using batch gradient descent. In detail, for a single training example, the cost function with respect to that single example is defined as follows:

$$J(W,b;x,y) = \frac{1}{2}\|h_{W,b}(x) - y\|^2 \tag{3.10}$$

Let us define the overall cost function for a training set of m examples to be:

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x,y)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2 \\ &= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}\left(x^{(i)}\right) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2 \end{aligned} \tag{3.11}$$

The first term in Equation 3.11 is an average sum-of-squares error term. The second term is a regularization term which helps reduce the magnitude of the weights and prevent overfitting.

The main purpose of training a neural network is to minimize cost function $J(W,b)$ with respect to W and b. This task can be done in two steps as follows:

- Initialize each parameter $W_{ij}^{(l)}$ and each $b_i^{(l)}$ to a small random value near zero.
- Use an optimization algorithm such as batch gradient descent to update the parameters W, b. For example, applying batch gradient descent to our problem, we have to repeat one iteration of gradient descent to update the parameters W, b as follows:

$$W_{ij}^{(l)} := W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b) \tag{3.12}$$

$$b_i^{(l)} := b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b) \tag{3.13}$$

  where α is the learning rate. The algorithm stops when no significant change of the parameters W, b is updated.

The pivotal step of batch gradient descent is computing the partial derivatives $\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)$ and $\frac{\partial}{\partial b_i^{(l)}} J(W,b)$ in Equation 3.12 and 3.13 respectively. Referring to Equation 3.11, these partial derivatives can be computed as:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J\left(W,b;x^{(i)},y^{(i)}\right)\right] + \lambda W_{ij}^{(l)} \tag{3.14}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W,b) = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial b_i^{(l)}} J\left(W,b;x^{(i)},y^{(i)}\right) \tag{3.15}$$

The direct computation of these derivative is very complicated. An efficient way to estimate these derivatives is using the backpropagation algorithm [65]. This algorithm includes the following steps.

- Perform a feedforward pass, which means computing the activations for the input layer L_1, then the hidden layer L_2, and finally the output layer L_3.
- For each unit i in the output layer L_3, assign

$$\delta_i^{(3)} = \frac{\partial}{\partial z_i^{(3)}} \frac{1}{2}\|y - h_{W,b}(x)\|^2 = -\left(y_i - a_i^{(3)}\right) f'\left(z_i^{(3)}\right) \tag{3.16}$$

- For each unit i in layer L_2, set

$$\delta_i^{(2)} = \left(\sum_{j=1}^{n_{out}} W_{ji}^{(2)} \delta_j^{(3)}\right) f'\left(z_i^{(2)}\right) \tag{3.17}$$

  For each unit i in layer L_1, set

$$\delta_i^{(1)} = \left(\sum_{j=1}^{n_{hid}} W_{ji}^{(1)} \delta_j^{(2)}\right) f'\left(z_i^{(1)}\right) \tag{3.18}$$

- Compute the desired partial derivatives as follows:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J\left(W, b; x, y\right) = a_j^{(l)} \delta_i^{(l+1)} \tag{3.19}$$

$$\frac{\partial}{\partial b_i^{(l)}} J\left(W, b; x, y\right) = \delta_i^{(l+1)} \tag{3.20}$$

After training the neural network, we have a set of optimized parameters $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$, which represents our trained neural network.

Prediction

Given a testing example $x^{(t)}$, we have to perform a feedforward pass with the optimized parameters (W, b) to compute hypotheses $h_\theta(x^{(t)})$. The predicted class label of the testing example from the neural network is the label that has the largest output $(h_\theta(x^{(t)}))_k$. In other words:

$$pred(x^{(t)}) = \underset{k=1,\ldots,n_{class}}{\arg\max} \; (h_\theta(x^{(t)}))_k \tag{3.21}$$

3.3.4 Summary

Approach using Neural Network is able to classify images based on their descriptors consisting all pixels. However, the input data to the neural network is very raw, which not only increases the training time but also decreases the classification accuracy of the neural network.

3.4 Approach 2 - Using Sparse Autoencoder

This approach also belongs to the first group of approaches, which regards an image as a whole. The idea of the approach is to exploit unsupervised learning for encoding images. While supervised learning is very powerful, its restriction is that it requires researchers to have knowledge in target domains to figure

out a good feature representation so that a supervised learning algorithm can perform well. For the problem of image classification, finding a good feature representation means finding an good image encoding scheme, which is difficult for researchers who are not familiar with terms of image. For example, in Approach using Neural Network, we suppose we do not know much about image information and image processing, so we put all raw data of each image directly to a neural network. Fortunately, the knowledge in the domain of digital image can be ignored by using an autoencoder, which is a powerful unsupervised learning method that can automatically learn good features from unlabeled images.

Specifically, this approach uses a sparse autoencoder as a tool for encoding images. The image descriptors resulted from the autoencoder are then used to train a SVM, which is a supervised learning algorithm. The flow of the approach is shown in Figure 3.5.

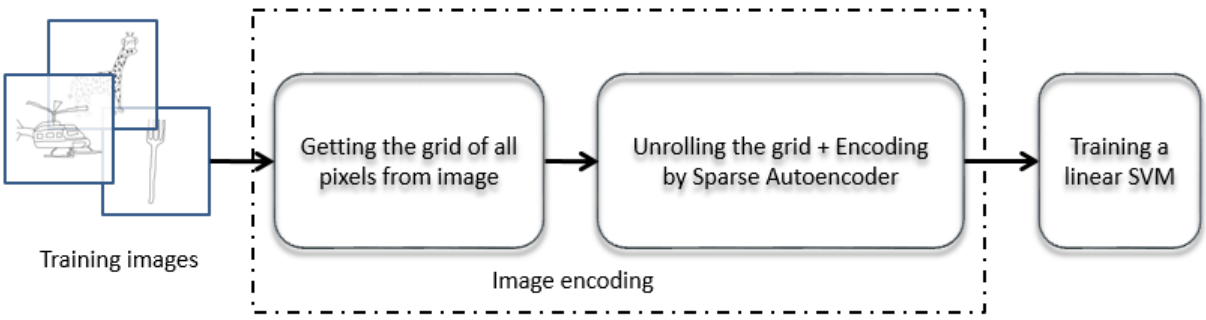


Figure 3.5: The flow of Approach using Sparse Autoencoder.

3.4.1 Global Feature Extraction

Like Approach using Neural Network, to avoid the large number of pixels, all images are resized to a small size (e.g. 32×32 or 64×64). Next, we extract the grid containing all pixels of each image. Each grid is the global information of entire image.

3.4.2 Building Image Descriptor

We also unroll all pixels to form a vector $x \in \mathbb{R}^{n_{pixel}}$. This vector, however, is not the final image descriptor of the image. The vector x is considered a temporary representation, which is used as the input for an autoencoder to encode into a final image descriptor $\ddot{x} \in \mathbb{R}^M$.

We treat our labeled training examples $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$, where $x^{(i)}$ is the temporary representation of sketch image $s^{(i)}$ created above, as unlabeled training examples $\{x^{(1)}, ..., x^{(m)}\}$. An autoencoder [26] is a special neural network that applies propagation, setting the target values to be equal to the inputs.

We use the same neural network model and notations applied as a classifier in Approach using Neural Network (see 3.3) for our autoencoder with additionally setting $y^{(i)} = x^{(i)}$ and $n_{hid} = M$ where M is the number of dimensions of the final image descriptor $\ddot{x}^{(i)}$ as mentioned above. The autoencoder used in this approach is illustrated in Figure 3.6.

The autoencoder tries to learn a hypotheses $h_{W,b}(x) \approx x$. In other words, given an input x, the autoencoder tries to approximate x by learning a function such that through that function the autoencoder can output $\hat{x}$ similar to x.

As $n_{hid} < n_{in}$, i.e. $M < n_{pixel}$ for our autoencoder, the autoencoder is able to learn a compressed representation of the input data, which is shown as the activation of the hidden layer. That means the autoencoder tries to reconstruct the n_{pixel}-dimensional input x in order to encode it into only a M-dimensional image descriptor.

As $n_{hid} > n_{in}$, i.e. $M > n_{pixel}$ for our autoencoder, the autoencoder is also able to learn a very good representation of the input data if we impose a sparsity constraint on the hidden units. Intuitively, a neuron is considered being active if its activation is close to 1, or being inactive if its activation is close to 0.

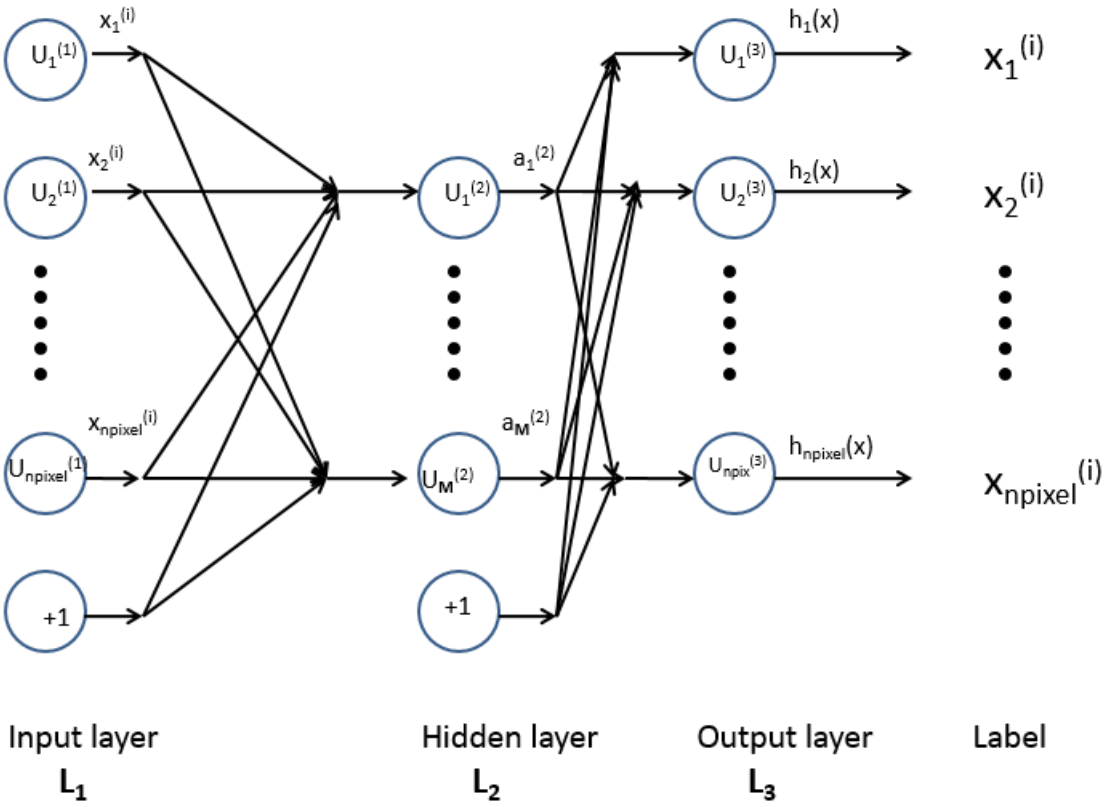


Figure 3.6: The autoencoder is used in Approach 2 for encoding image.

We wish the neurons to be inactive most of the time. Let $\hat{\rho}_j$ denote the average activation of hidden unit $a_j^{(2)}$ over the training set. $\hat{\rho}_j$ is computed as:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} \left[a_j^{(2)}(x^{(i)}) \right] \tag{3.22}$$

where $a_j^{(2)}(x^{(i)})$ denotes the activation of hidden unit $u_j^{(2)}$ if the neural network is given a specific input x. We want a sparsity constraint to be approximately enforced:

$$\hat{\rho}_j = \rho \tag{3.23}$$

where ρ is a sparsity parameter, which is typically a small value close to zero. Due to this constraint, the hidden unit's activations is mostly near 0, i.e. inactive. Imposing the sparsity on the autoencoder causes the cost function to be

updated to a new cost function:

$$J_{sparse}(W,b) = J(W,b) + \beta \sum_{j=1}^{n_{hid}} KL(\rho \,||\, \widehat{\rho}_j) \tag{3.24}$$

where $\sum_{j=1}^{n_{hid}} KL(\rho||\hat{\rho}_j)$ is a sparsity penalty term, which is added to cause $\hat{\rho}_j$ to be close to ρ, and β is used to control the weight of the sparsity penalty term. $KL\,(\rho \,||\, \widehat{\rho}_j)$ is the Kullback-Leibler (KL) divergence [3] which is defined as:

$$KL\,(\rho \,||\, \widehat{\rho}_j) = \rho \log \frac{\rho}{\widehat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\widehat{\rho}_j} \tag{3.25}$$

The KL-divergence has the property that it obtains the minimum of 0 at $\hat{\rho}_j = \rho$, and otherwise it blows up as $\hat{\rho}_j$ diverges from ρ. For this reason, minimizing this penalty term keeps $\hat{\rho}_j$ not to deviate significantly from ρ.

After training our sparse autoencoder in the same way used for the neural network in Approach 1, we also obtain optimized parameters $(W,b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$. We then use the pair $(W^{(1)}, b^{(1)})$ to compute activations of hidden units $a_k^{(2)}(x^{(i)})$ for training example $x^{(i)}$. The final image descriptor $\ddot{x}^{(i)} = [\ddot{x}_1^{(i)}, ..., \ddot{x}_M^{(i)}]$ of sketch image $s^{(i)}$, which corresponds to temporary representation $x^{(i)}$, is exactly those activations:

$$\ddot{x}_k^{(i)} = a_k^{(2)}(x^{(i)}), k = 1, ..., M \tag{3.26}$$

The image descriptors of training images are used to train a SVM in the stage of classifier training.

3.4.3 Classifier Training

This approach uses a linear Support Vector Machine (SVM) as a classifier. SVM is one of the best supervised learning algorithms. The idea of this classifier is to build a classification model for each class, which is an optimal separating

hyperplane, also called the decision boundary, to divide a set of data points into two separate region each of which corresponds to being either of that class or not of that class. We first briefly describe a linear SVM model, then demonstrate how to train the SVM from the training examples, which is the image descriptors encoded from our sparse autoencoder in the encoding stage, and finally show how to predict the class label for a testing example.

Linear SVM Model

A binary SVM is used for a binary classification problem in which an example, i.e. feature vector, x has its class label $y \in \{1, -1\}$. We define our binary classifier with parameters w, b as:

$$h_{w,b}(x) = g(\omega^T x + b) \tag{3.27}$$

where $g(z) = 1$ if $z \geq 0$, and $g(z) = -1$ otherwise. Intuitively, equation $\omega^T x + b = 0$ plays the role of a separating hyperplane which separates the space into two sides. $\omega^T x + b$ can be considered a function. One of the sides corresponds to class label 1, which contains data points, each of which is positive as plugged into the function. The other side corresponds to class label -1.

Given an example $x^{(i)}, y^{(i)}$, the functional margin of (w, b) with respect to the example is defined as follows:

$$\hat{\gamma}^{(i)} = y^{(i)} \left(w^T x^{(i)} + b \right) \tag{3.28}$$

Given a set of examples $U = \{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$, $\hat{\gamma}$ denotes the functional margin of (w, b) with respect to U. This can be determined as the smallest of the functional margins of the individual examples:

$$\hat{\gamma} = \min_{i=1,...,m} \hat{\gamma}^{(i)} \tag{3.29}$$

The functional margin, however, depends on the magnitude of $w^T x + b$. To make the functional margin invariant to rescaling of the parameters (w, b), we define the geometric margin of (w, b) with respect to an example $x^{(i)}, y^{(i)}$ to be:

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|} \right) \tag{3.30}$$

The geometric margin of a classifier with respect to an example is visualized in Figure 3.7. In Figure 3.7, the line that separates the side containing circle signs and the side containing cross signs is the separate hyperplane, i.e. the decision boundary, $\omega^T x + b = 0$. The vector w is orthogonal to the separating hyperplane. Point A represents the input $x^{(i)}$ of some an example with class label $y^{(i)} = 1$. The line segment AB, which is the distance from point A to the separating hyperplane, represents the geometric margin of (w, b) with respect to example $x^{(i)}$.

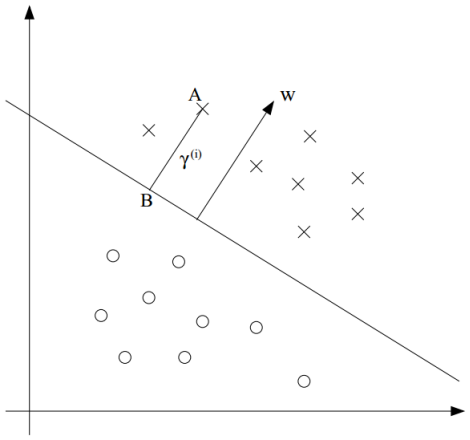


Figure 3.7: The geometric margin of a classifier with respect to an example. [1]

With the set of example $U = \{(x^{(i)}, y^{(i)}); i = 1, ..., m\}$, we also define the geometric margin of (w, b) with respect to U as γ:

$$\gamma = \min_{i=1,...,m} \gamma^{(i)} \tag{3.31}$$

[1]Image source: https://class.coursera.org/ml/

Intuitively, enlarging the geometric margin increases the possibility of correct classification. Thus, in the problem of binary classification, the main goal is to find a decision boundary that maximizes the geometric margin γ of the classifier with respect to a given set of examples. This is equivalent to solving the following optimization problem:

$$\begin{aligned} &\max_{w,b,\gamma} \ \gamma \\ &s.t. \ y^{(i)} \left(\left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|} \right) \geq \gamma, i = 1, \cdots, m \\ &\quad \|w\| = 1 \end{aligned} \tag{3.32}$$

Due to the relation $\gamma = \frac{\hat{\gamma}}{\|w\|}$, we can rewrite the optimization problem in terms of $\hat{\gamma}$ to remove the constraint $\|w\| = 1$ which is non-convex one as follows:

$$\begin{aligned} &\max_{w,b,\gamma} \ \frac{\widehat{\gamma}}{\|w\|_2} \\ &s.t. \quad y^{(i)} \left(w^T x^{(i)} + b \right) \geq \widehat{\gamma}, i = 1, ..., m \end{aligned} \tag{3.33}$$

The objective function $\frac{\hat{\gamma}}{\|w\|}$ is non-convex again. Since the geometric margin is invariant to rescaling of the parameters as we mention above, we can enforce the scaling constraint that the functional margin of w, b with respect to the training set must be 1:

$$\hat{\gamma} = 1 \tag{3.34}$$

Rescaling the functional margin $\hat{\gamma}^{(i)} = y^{(i)} \left(w^T x^{(i)} + b \right)$ can be done by rescaling w, b, so this scaling constraint do not change the meaning of our optimization problem. Substituting this into our problem, we have the following equivalent problem:

$$\begin{aligned} &\min_{w,b,\gamma} \ \frac{1}{2} \|w\|^2 \\ &s.t. \quad y^{(i)} \left(w^T x^{(i)} + b \right) \geq 1, i = 1, ..., m \end{aligned} \tag{3.35}$$

Here, maximizing $\frac{\hat{\gamma}}{\|w\|} = \frac{1}{\|w\|}$ is equivalent to minimizing $\|w\|^2$. To solve the opti-

mization problem in 3.51, we can use either commercial quadratic programming [54] or Lagrange duality [38]. Solving this optimization problem results in the optimal margin classifier, which is the parameters w, b with the largest possible geometric margin with respect to the training set. The points with smallest margins, which are closest to the decision boundary, lie on two lines, which are denoted by dashed lines in Figure 3.8, parallel to the decision boundary. These points are called the support vectors.

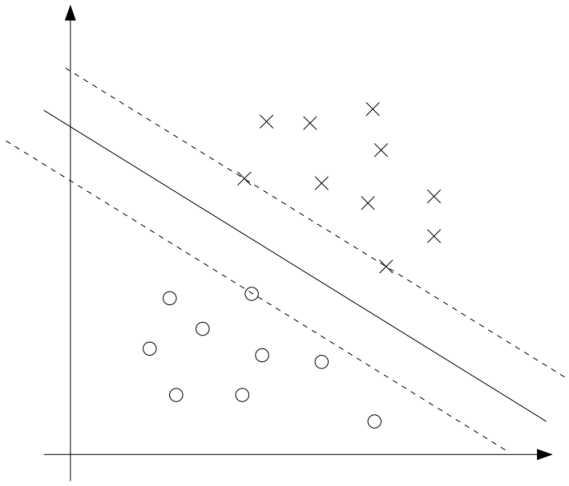


Figure 3.8: An optimal margin classifier with its support vectors. [2]

In some cases, set of examples are non-linearly separable, which means it is impossible to separate the positive and negative examples using some decision boundary. For this reason, to allow examples to have functional margin less than 1, the constraint of the margin in Equation 3.51 is changed to a looser one as follows:

$$\begin{aligned} &\min_{w,b,\gamma} \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ &s.t.\ y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \xi_i, i = 1, ..., m \\ &\xi_i \geq 0, i = 1, ..., m \end{aligned} \tag{3.36}$$

where $\sum_{i=1}^{m}\xi_i$ is a penalty term, which is added to ensure that most examples have

[2] Image source: https://class.coursera.org/ml/

functional margin at least 1, and C is used to control the weight of the penalty term.

Training

We have a set of training examples $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$, where $x^{(i)}$ is actually image descriptor $\ddot{x}^{(i)}$ resulted from the image encoding stage, and $y^{(i)}$ is the class label of corresponding image $s^{(i)}$.

Since our problem has more than two class, we have to use a multiclass SVM instead of a binary SVM. A multiclass SVM can be constructed by many ways such as one-versus-all strategy, one-versus-one strategy, using directed acyclic graph, or using error-correcting output codes. We, however, choose one-versus-all strategy for our multiclass SVM because this strategy is very popular and commonly used.

In one-versus-all classification, we have to train multiple binary classifiers, one for each class. For our problem, we have n_{class} classes of sketches, so we train a number of n_{class} binary SVM, one of which is for each of the n_{class} classes in our training set of sketches. When training the binary SVM for class k, we have to set label $y^{(i)}$ of training example $x^{(i)}$ to 1 if $y^{(i)} = k$. Otherwise we set label $y^{(i)}$ to -1 if $y^{(i)} \neq k$ to indicate this training example does not belong to class k. We train each binary SVM independently using Equation 3.36.

AFter the training step, we attain a set of n_{class} binary SVM $\{(W^{(1)}, b^{(1)}), ..., (W^{(n_{class})}, b^{(n_{cla}$ where $(W^{(k)}, b^{(k)})$ are the parameters of the binary SVM for class k.

Prediction

Given a testing example $x^{(t)}$, we plug the example into the trained SVM of each class k to compute the geometric margin $\gamma^{(t)} = 1\left((w^{(k)})^T x^{(t)} + b^{(k)}\right)$, which can be considered the probability that the example belongs to class k. We then pick the class for which the corresponding SVM outputs the highest probability and

return the class label as the prediction for the testing example:

$$pred(x^{(t)}) = \underset{k=1,\ldots,n_{class}}{\arg\max} \left(\left(w^{(k)}\right)^T x^{(t)} + b^{(k)} \right) \tag{3.37}$$

3.4.4 Summary

Approach using Sparse Autoencoder uses a SVM to classify images based on the feature representation extracted by a sparse autoencoder. The sparse autoencoder can be also used to find useful features for a wide range of problems (e.g. ones in text, audio, etc). However, even though Approach using Sparse Autocencoder enhance the performance of Approach using Neural Network by obtain a better encoding than the raw representation, the feature resulted from this simple version of sparse autoencoder is not by itself competitive with the best hand-engineered features in the field of image classification.

3.5 Approach 3 - Using Explicit Feature Map

This approach is suggested in one example of image classification in VLFEAT [58]. The approach falls into the second group of approaches, which regards an image as a collection of image patches. The approach builds the descriptor for each image by constructing a spatial pyramid of histograms on the image using Vector Quantization coding. The matching kernel that is commonly used for evaluating the similarity between two histograms is either intersection kernel or Chi-square kernel. Both these kernels are non-linear. While non-linear kernels tend to give better classification accuracy, linear kernels are more efficient for training. Non-linear kernels are conventionally dealt by non-linear SVM with the kernel trick. However, in this approach, a new approach is suggested to compute an explicit feature map [60] to transform the image descriptors to a linear space so that they can be trained in a linear SVM with a linear kernel.

Specifically, in this approach, to encode sketch images, each image is first extracted a dense set of SIFT descriptors. The space of the SIFT features extracted from a subset of training images are then clustered using K-Means algorithm

to make a dictionary of visual words each of which corresponds to a cluster. Next, the descriptors of each image are quantized into visual words using the conventional method of vector quantization. The quantized descriptors are then used to build a histogram of visual words using sum pooling. Afterwards, the histograms of all levels of image resolutions are concatenated to form a spatial pyramid of histograms for each sketch image. The resulting spatial histogram is considered the image descriptor.

For the classifier training stage, an explicit feature map is computed to transform the image descriptor of each image. The transformed image descriptors are then used to train a linear multiclass SVM. The flow of this approach is shown in Figure 3.9.

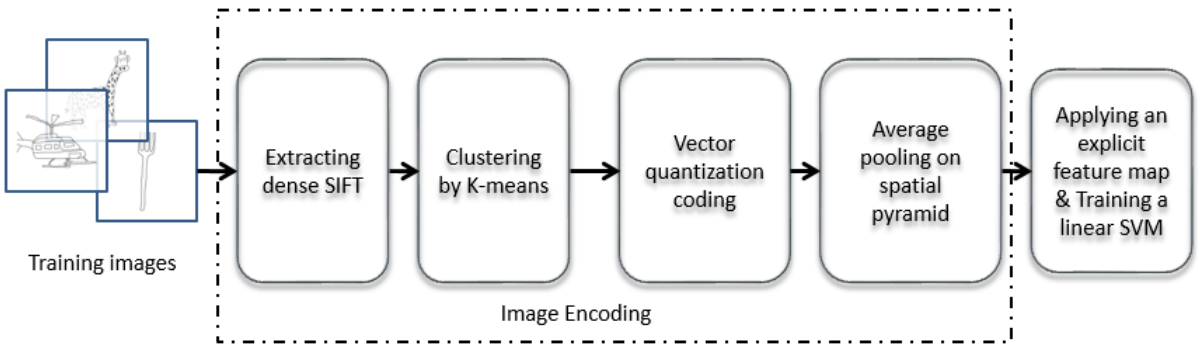


Figure 3.9: The flow of Approach 3.

3.5.1 Feature Extraction

This step extracts dense multi-scale SIFT descriptors for each sketch image $s^{(i)}$ in our training set. We prefer the second strategy for recognizing sketches because the first one based on interest points is only suitable for texture or scene images, which have many salient points, not for sketch images. The number of interest points per sketch image is too small to represent the whole sketch. Therefore, we use only dense features, specifically dense SIFT, for all approaches of this group. Next, the patches around the features are described by vectors of the same

length, e.g. 128 for SIFT feature. These vectors are called feature descriptors. After this step, we obtain a set of feature descriptors for each image.

SIFT Feature

The scale invariant feature transform (SIFT) descriptor were introduced by Lowe [36]. It is used to describe the information of the image region around interest points found by the SIFT detector. The intensity image is filtered with a difference of Gaussian (DoG) kernel at increasingly coarse scales. The resulting images are then stacked to a volume. Scale-space extrema are detected within the volume, and DoG interest points are then localized. Next, each detected keypoint is assigned a particular scale and one or more orientations obtained from local gradient directions. This kind of SIFT feature is also referred as sparse SIFT feature because it is extracted based on interest points.

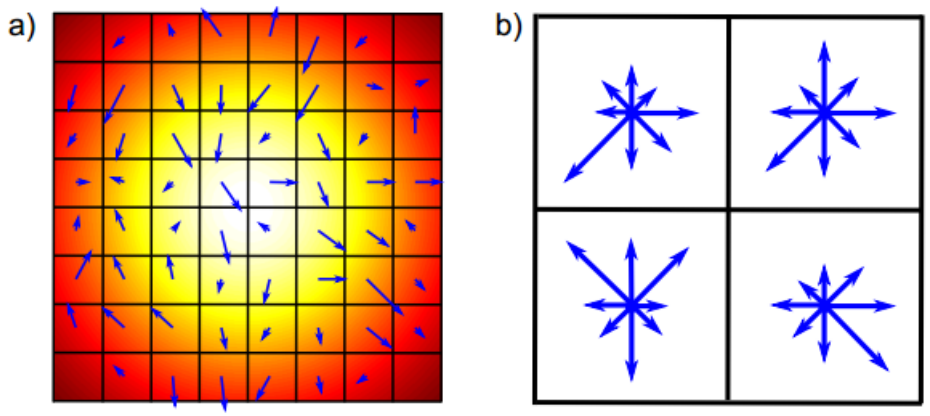


Figure 3.10: SIFT descriptor a) Gradient map for the local patch around a SIFT interest point. b) Orientation histogram for each cell of the patch.[3]

The square local patch around the interest point are described by the SIFT descriptor. First, the gradient orientation and amplitude map of a 16 x 16 pixel patch around the interest point are computed. The 16 x 16 patch is then subdivided into non-overlapping 4 x 4 cells. In each cell, the image orientation previously computed is quantized into 8 bins depicting 8 different gradient direc-

[3]Image source: http://www.computervisionmodels.com/

tions over the range $0 - 360^o$, resulting in a histogram of 8 orientations weighted by the pixel's gradient amplitude and by the pixel's distance from the interest point. Finally, histograms of 4 x 4 = 16 cells are concatenated to make a final descriptor of the region around the interest point which is a single vector of 128 dimensions (16 cells x 8 dimensions). The descriptor is normalized afterwards. An illustration of SIFT descriptor is shown in Figure 3.10.

As reflected in the name, SIFT descriptor is invariant to translation, scaling, and rotation. Due to using gradients and being normalized, the SIFT descriptor is invariant to constant intensity changes and contrast as well. In addition, the descriptor is not affected much by small deformation because of orientation pooling within each cell.

Dense SIFT Feature

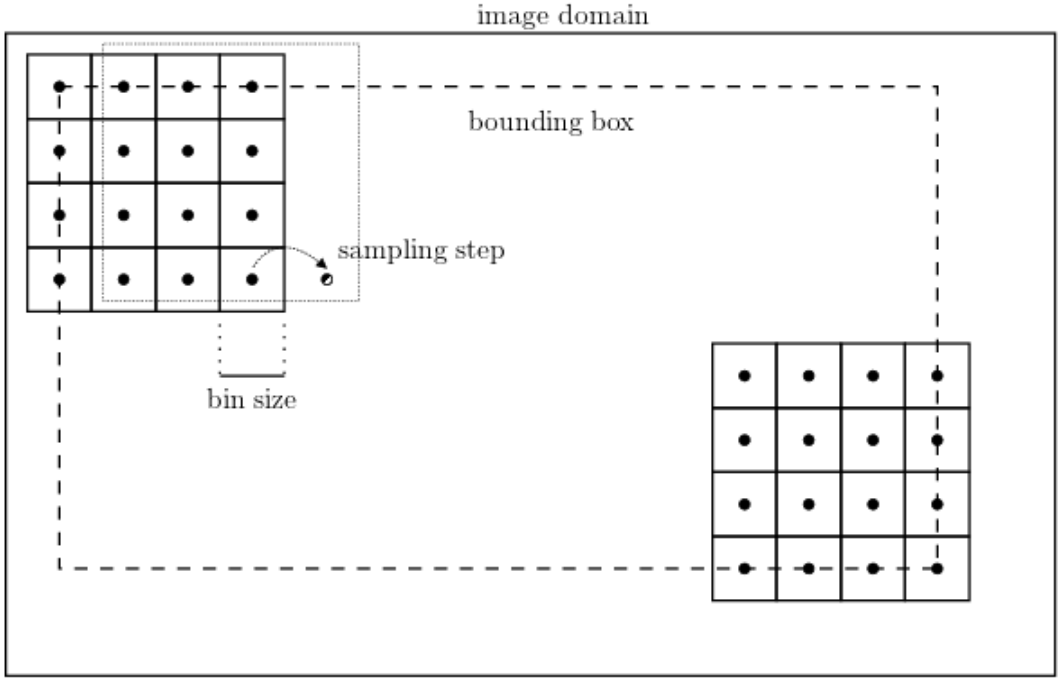


Figure 3.11: Dense SIFT descriptor geometry.[4]

However, sparse SIFT features are not suitable for our sketch recognition problem. Since the number of sparse SIFT features extracted from an sketch

[4]Image source: http://www.vlfeat.org/api/dsift.html

image is usually small, those features cannot represent the whole image. We extract dense SIFT features from sketch images instead. Dense SIFT [6] is a dense version of SIFT that computes a large number of descriptors of densely sampled features instead of interest points determined by the SIFT detector. Specifically, an image is divided into a regular grid with sampling step of L pixels. N is chosen to be 5, 10 and 15 as well. SIFT descriptor is then computed on every points on the grid. Circular support patches with radii $R = 4$, 8, 12 and 16 pixels are used when computing SIFT descriptor at each point on the grid. An illustration of dense grid of SIFT can be seen in Figure 3.11.

This process results in k 128-dimensional SIFT descriptors (where k is the number of circular support patches) that represents the visual information of each grid point. As $k > 1$, multiple descriptors are calculated to support multiscale variation. There is a overlap between the patches with radii 8, 12, and 16. The parameters L and k are significant because they control the overlap degree. The resulting descriptors are invariant to rotation due to the rotation invariance of SIFT.

After the step of feature extraction, we have the same number of SIFT descriptor, denoted by N for each sketch image $s^{(i)}$. Let the set of the SIFT descriptors for sketch image $s^{(i)}$ be denoted by $V^{(i)} = \{v_1^{(i)}, ..., v_N^{(i)}\}, i = 1, ..., m$ where m is the number of sketch images in our training set and v is a D-dimensional feature vector. Here, $D = 128$ for SIFT descriptor. We also use V to denote the set of the SIFT features extracted from all training sketch images. So $V = \{V^{(1)}, ..., V^{(m)}\}$.

3.5.2 Visual Dictionary Construction

This step follows the bag-of-words model [14] which consider an image a collection of visual words instead of discrete visual descriptors. The total number of the visual descriptors is actually very large. However, the similar descriptors concentrate in some regions that are approximately disjoint in the feature space. Hence, in this step, we use K-Means algorithm to partition the visual descriptors

into K clusters to make a dictionary of visual words, each of which corresponds to a cluster.

K-Means Clustering Algorithm

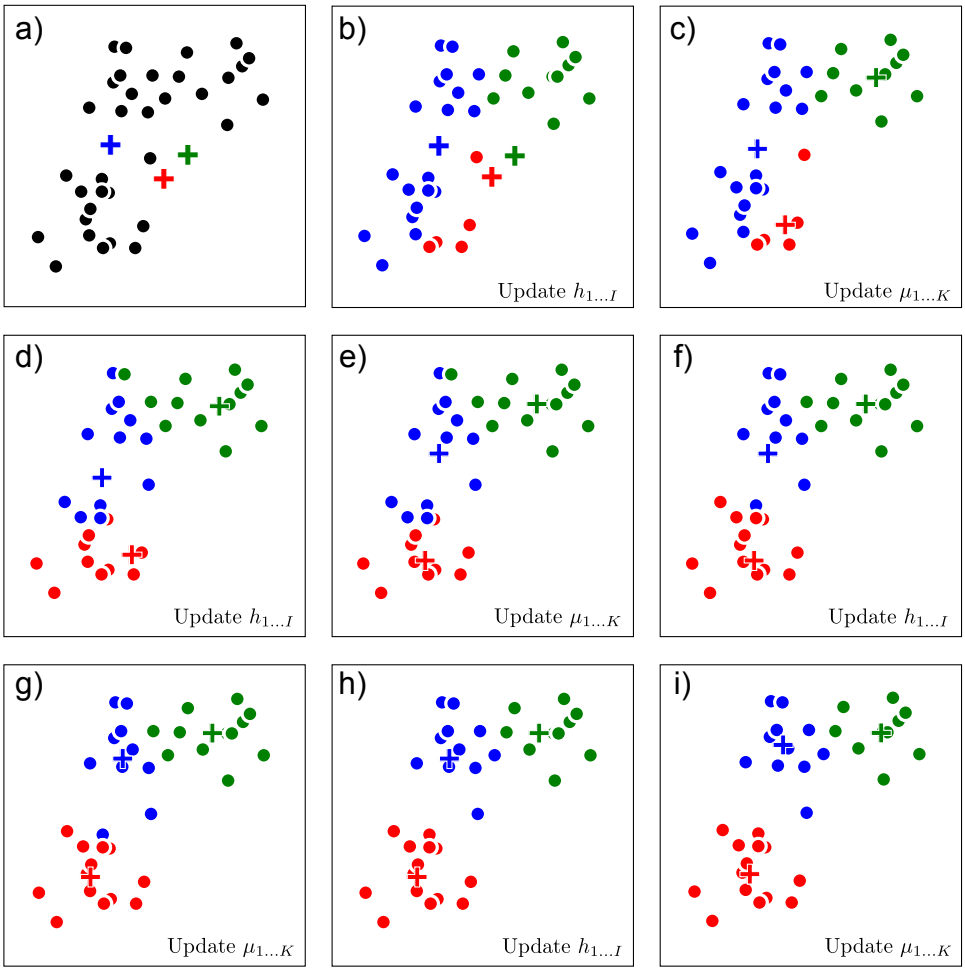


Figure 3.12: K-means algorithm for $K = 3$ clusters. a) Initialize 3 cluster centers (drawn as crosses) to random positions in the descriptor space. b) Assign each descriptor to the nearest cluster center. c) Update the cluster centers to be the mean of the cluster's descriptors. d-i) Alternate steps b and c iteratively until there is no change of centers.[5]

K-means is probably the most popular clustering algorithm. It aims to partition the space of visual descriptors into informative regions whose internal

[5]Image source: http://www.computervisionmodels.com/

structure can be ignored. In terms of image classification, these informative regions are referred as visual words and the descriptors space is called the visual vocabulary.

We take from the set V a subset of I visual descriptors $u_1, ..., u_I$ where $u_i \in V = \{v_1, ..., v_N\}$. K-means algorithm tries to find K cluster centers $\mu_1, \mu_2, ..., \mu_K \in \mathbb{R}^D$ and I indexes $h_1, h_2, ..., h_I \in \{1, ..., K\}$ that assign descriptor u_i to the corresponding center μ_{h_i}. The descriptors hereby are approximated as $x_i \approx \mu_{h_i}$. To figure out the assignment indexes and the cluster centers, we optimize the following problem:

$$\mu_{1..K}, h_{1..I} = \arg\min_{\mu,h} \sum_{i=1}^{I} \|u_i - \mu_{h_i}\|^2 \tag{3.38}$$

This cost function can be minimized using either Lloyd's algorithm [35] or Approximate Nearest Neighbour (ANN) algorithm. We use the former which is more common. Lloyd's algorithm is an alternating strategy in which we first seek the Euclidean nearest cluster center for each descriptor:

$$h_i = \arg\min_{h_i} \|u_i - \mu_{h_i}\|^2 \tag{3.39}$$

Next, we update the center for each cluster as follows:

$$\begin{aligned} \mu_k &= \arg\min_{\mu_k} \sum_{i=1}^{I} \|u_i - \mu_{h_i}\|^2 \\ &= \frac{\sum_{i=1}^{I} u_i \delta(h_i - k)}{\sum_{i=1}^{I} \delta(h_i - k)} \end{aligned} \tag{3.40}$$

where $\delta(t)$ returns 1 as t equals 0 and 0 otherwise. By this way, the updated center for each cluster μ_k is chosen to be the mean of the descriptors which belongs to that cluster. The process of K-means clustering algorithm is illustrated in Figure 3.12.

The step of clustering results us a set B of K clusters $\mu_1, \mu_2, ..., \mu_K$ in D-dimensional space. μ_i is called the codeword, i.e. the visual word. B is called the codebook, i.e. the visual dictionary.

3.5.3 Quantization

Based on the bag-of-words model, this step tries to quantize the large set of the visual descriptors to a smaller set of the visual words, i.e. consider an image in terms of visual words rather than visual descriptors. Specifically, each D-dimensional descriptor is encoded to a K-dimensional vector which belongs to the space of the codebook. In this step, we use the vector quantization coding for quantizing the descriptors.

Vector Quantization

Vector quantization (VQ) coding plays a role of baseline encoding upon which many later coding methods improve. Recall that $V^{(t)} = [v_1^{(t)}, ..., v_N^{(t)}] \in \mathbb{R}^{D \times N}$ is the set of descriptors for sketch image $s^{(t)}$ in a D-dimensional space. The codebook $B = [\mu_1, ..., \mu_K] \in \mathbb{R}^{D \times K}$ where each codeword μ_i corresponds to a visual word is given from the step of visual dictionary construction. VQ coding tries to solve the following constrained least square fitting problem:

$$\begin{aligned} &\arg\min_{C^{(t)}} \sum_{i=1}^{N} \left\| v_i^{(t)} - B c_i^{(t)} \right\|^2 \\ &s.t. \left\| c_i^{(t)} \right\|_{l^0} = 1, \left\| c_i^{(t)} \right\|_{l^1} = 1, c_i^{(t)} \geq 0, \forall i \end{aligned} \tag{3.41}$$

where $C^{(t)} = [c_1^{(t)}, ..., c_N^{(t)}] \in \mathbb{R}^{K \times N}$ contains the corresponding codes for elements of $V^{(t)}$. The constraint $\left\| c_i^{(t)} \right\|_{l^0} = 1$ is given to make sure only one non-zero element is allowed in each code $c_i^{(t)}$, corresponding the the quantization id of $v_i^{(t)}$. The constraint $\left\| c_i^{(t)} \right\|_{l^1} = 1, c_i \geq 0$ makes sure the coding weight for $v_i^{(t)}$ must be 1. Two above constraints leads to the fact that each descriptor is assigned (quantized) to a unique codeword, i.e. hard assignment, and the coding weight is always 1.

This step of quantization results in set of codes $C^{(t)} = [c_1^{(t)}, ..., c_N^{(t)}]$ corresponding to set of descriptors $V^{(t)} = [v_1^{(t)}, ..., v_N^{(t)}]$ for sketch image $s^{(t)}$.

3.5.4 Building Image Descriptor

The conventional bag-of-words encoding completely disregard the spatial layout of the features in the image, and hence cannot know the geometrical regularities in image composition and the features' spatial arrangement, which can make the classification task more accurate. To incorporate weak geometry into a bag-of-words encoding, the spatial pyramid layout is used to build the image descriptor. This spatial pyramid layout can be extended to any encoding by computing one encoding for each spatial region and then concatenating to build the result descriptor that represents the image.

In this step, we pool the descriptors of each image on a spatial pyramid constructed for the image. The pooled descriptors from every spatial regions are then concatenated to form a spatial pyramid representation of the image, which is the final image descriptor. We choose the average function as the pooling function for each spatial region, which corresponds to a resulting histogram of visual words. We first introduce the pyramid matching framework as well as the spatial pyramid scheme applied for image histograms. Then we show how to apply this scheme to our quantized features so as to build the image descriptor.

Pyramid Matching Framework

Pyramid matching [22] is a scheme used to estimate the approximate similarity between two sets of feature vectors, i.e. the distance between these two set in the same feature space. The idea of this scheme is to combine the representations of smaller set partitions in the feature space at multiple resolution and compare resulting multi-resolution representations of sets. More specifically, this scheme makes a sequence of increasingly coarser grids in the feature space and takes a weighted sum of the number of matches for each resolution. Two points are

regarded as a match if they belongs to the same grid cell at the same resolution. Matches found at finer resolutions are weighted more because those matches are more highly expected than matches at coarser resolutions.

Let us formalize the scheme's idea. Two sets of vectors X and Y in a d-dimensional feature space are given. For each resolution $l \in 0, 1, 2, ..., L$, the grid is divided in to 2^l cells along each dimension, resulting in $D = 2^{dl}$ cells in total. Let H_X^l and H_Y^l be the histograms of X and Y at the resolution l, so that $H_X^l(i)$ and $H_Y^l(i)$ count the number of points from X and Y that belong to the i-th cell of the grid. The number of matches between two sets X and Y at resolution l is computed by the histogram intersection function:

$$I(H_X^l, H_Y^l) = \sum_{i=1}^{D} \min(H_X^l(i), H_Y^l(i)) \tag{3.42}$$

For the rest, the notation I^l is used to refer $I(H_X^l, H_Y^l)$. In addition, the word "level" refers to the resolution. It should be noticed that the matches found at finer level include all the matches at coarser level. For this reason, the number of new matches at level l is computed as $I^l - T^{l+1}$. The weight factor $\frac{1}{2^{L-l}}$ is also added to penalize matches found in finer level. Finally we obtain a pyramid match kernel:

$$\begin{aligned} \kappa^L(X, Y) &= I^L + \sum_{l=0}^{L-1} \frac{1}{2^{L-l}} (I^l - I^{l+1}) \\ &= \frac{1}{2^L} I^0 + \sum_{l=1}^{L} \frac{1}{2^{L-l+1}} I^l \end{aligned} \tag{3.43}$$

Spatial Pyramid Scheme

A pyramid match kernel supports multi-resolution matching of two sets of feature vectors. It, however, get rid of all spatial information of the set. Moreover, the pyramid match kernel is not a good kernel for matching high-dimensional features, e.g. SIFT descriptors, as the matching quality of the pyramid kernel

decreases linearly with the number of dimensions [30]. Spatial pyramid scheme [31] is used to overcome these drawbacks.

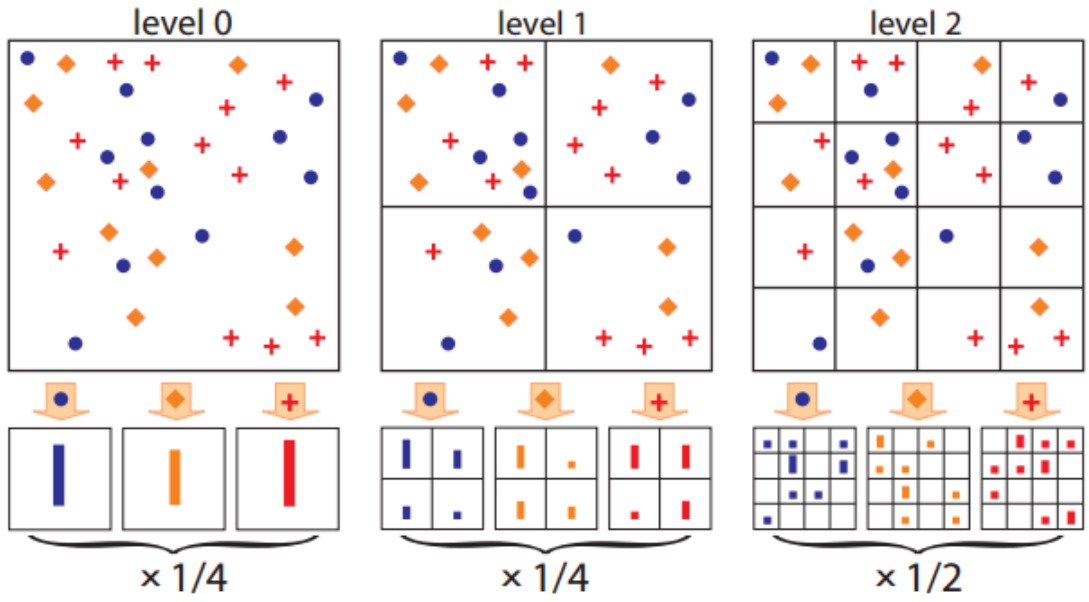


Figure 3.13: Pyramid constructioon for $L = 2$. The image contains there kinds of features, denoted by diamonds, crosses, and circles. The image is divided in three different levels. The features are then counted for each kind of feature and each level. Each spatial histogram is weighted as calculated in Equation 3.43 [6]

This approach applies pyramid matching in the two-dimensional image space, and use conventional vector quantization techniques to quantize all descriptors in the feature space into P discrete codewords, i.e. visual words. More specifically, to match two images X and Y, this approach first figures out two sets of features of type p for each chanel p of two images, denoted by X_p and Y_p respectively. All channel kernels sum up the final spatial pyramid matching kernel:

$$K^L(X, Y) = \sum_{p=1}^{P} \kappa^L(X_p, Y_p) \tag{3.44}$$

As L = 0, this becomes the standard bag of words method. Furthermore, instead of computing the kernel for each seperate level, this approach can concatenate

[6]Image source: http://www.cs.illinois.edu/homes/slazebni/

the appropriately weighted histograms of all channels at all resolutions as can be seen in Figure 3.13. The histogram intersection function can be then applied to the resulting histogram whose number of dimensions is:

$$P\sum_{l}^{L} 4^l = P\frac{1}{3}(4^{L+1} - 1) \tag{3.45}$$

Build Image Descriptor

As a result from the quantization step, we achieve the set of codes $C^{(t)} = [c_1^{(t)}, ..., c_N^{(t)}]$ corresponding to set of descriptors $V^{(t)} = [v_1^{(t)}, ..., v_N^{(t)}]$ for sketch image $s^{(t)}$. The histogram $z_i^{(t)}$ of each spatial region of the spatial pyramid constructed for image $s^{(t)}$ is calculated by average pooling on the VQ codes as follows:

$$z_j^{(t)} = \frac{1}{Q}\sum_{i=1}^{Q} c_{ij}^{(t)}, j = 1, ..., K$$

where Q is the number of descriptors in that region of the image and K is the number of dimensions of VQ code $c_i^{(t)}$. Recall that in our problem $x^{(t)} \in \mathbb{R}^M$ denotes the image descriptor of sketch image $s^{(t)}$. Thus, the image descriptor $x^{(t)}$, which is formed by concatenating the histograms of all spatial regions, has the number of dimensions as:

$$M = K\sum_{l}^{L} 4^l = K\frac{1}{3}(4^{L+1} - 1) \tag{3.47}$$

where L is the number of levels of the spatial pyramid. The image descriptors of all training images are used for training a classifier in the next stage.

3.5.5 Classifier Training

Instead of using a non-linear SVM with kernel trick for training images, this approach compute explicitly a feature map to transform the image descriptors

obtained from the image encoding stage to a linear space in which a linear SVM can be used for training. A linear SVM takes only the training time of $O(n)$ instead of $O(n^3)$ that a non-linear SVM takes. More importantly, a linear SVM helps reduce the testing time from $O(n)$ to $O(1)$, which is more appropriate for our real-time AR system than a non-linear SVM. First, we give an overview of kernel as well as kernel trick. Next, we introduce a non-linear SVM with a non-linear kernel. Afterwards, we demonstrate how to apply an explicit feature map into our problem.

Kernel

Given a feature map ϕ mapping vector representations from the original input space to the feature space, we can define the corresponding kernel to be

$$K(x, z) = \phi(x)^T \phi(z) \tag{3.48}$$

where x, z are two vectors which belong the original input space. For example, we have a learning algorithm. We want the algorithm to learn feature $\phi(x)$ instead of x. However, the cost of computing $\phi(x)$ is often high since it is typically an extremely high dimensional vector. Fortunately, $K(x, z)$ may be very inexpensive to calculate without finding directly $\phi(x)$ and $\phi(z)$ and then taking their inner product.

Hence, the kernel trick used in this kind of problem is that we try to represent our learning algorithm only in terms of inner product $\langle x, z \rangle$ so that when the algorithm learns in high dimensional feature space given by ϕ, all inner products $\langle x, z \rangle$ can be replaced with $\langle \phi(x), \phi(z) \rangle = K(x, z)$. Then, we can compute kernel $K(x, z)$ by using an efficient way without having to explicitly find ϕ and represent vector $\phi(x)$.

To find an efficient way to compute $K(x, z)$, we consider it from the view of similarity. In Euclidean space, the similarity between two vector x, z is computed by the inner product:

$$\langle x, z \rangle = x^T z \tag{3.49}$$

If in some learning problem, we want to measure how similar x and z are in a different way, we define a reasonable function $K(x, z)$ for measurement. For example, it can be a Gaussian function:

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \tag{3.50}$$

A raised issue is whether we can use a specified function $K(x, z)$ for similarity measurement as a kernel for a learning algorithm. In other words, given some function K, how do we know if it is a valid kernel which means there is some feature map ϕ so that $K(x, z) = \phi(x)^T \phi(z)$ for all x, z.

Given a finite set of m points $x^{(1)}, ..., x^{(m)}$, we can overload the same notation K to denote the corresponding kernel matrix of size $m \times m$ in which $K_{ij} = K(x^{(i)}, x^{(j)})$. By Mercer's theorem, for function K is a valid (Mercer) kernel, i.e. if it corresponds to some feature map ϕ, it is necessary and sufficient that the corresponding kernel matrix is symmetric positive semi-definite. Then, there is some Hilbert space H and some feature map ϕ so that $K\left(x^{(i)}, x^{(j)}\right) = \left\langle \phi(x^{(i)}, \phi(x^{(j)} \right\rangle_H$. For this reason, our kernelized learning algorithm can work in the original input space with the kernel function K as if we map the input data by the feature map ϕ to space H and taking their inner products.

Non-linear SVM

A non-linear SVM uses a non-linear kernel for learning. To exploit the kernel trick, we have to represent a SVM in terms of only inner products. Recall that in 3.4.3 we obtain the following primal optimization problem for a linear SVM:

$$\begin{aligned} &\min_{w,b,\gamma} \ \tfrac{1}{2}\|w\|^2 \\ &s.t. \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, i = 1, ..., m \end{aligned} \tag{3.51}$$

We can construct the Lagrangian for our problem as follows:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left[y^{(i)}(w^T x^{(i)} + b) - 1 \right] \tag{3.52}$$

where α_i are the Langrange multipliers. Minimizing this Lagrangian gives us the following dual optimization problem:

$$\begin{aligned} &\max_\alpha W(\alpha) = \sum_{i=1}^{m} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle x^{(i)}, x^{(j)} \right\rangle \\ &s.t. \alpha_i \geq 0, i = 1, ..., m \\ &\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned} \tag{3.53}$$

Here, $w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}$. For a linear SVM, the decision function is $f(x) = w^T x + b$. We can rewrite this decision function as follows:

$$\begin{aligned} f(x) = w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \left\langle x^{(i)}, x \right\rangle + b \end{aligned} \tag{3.54}$$

This means for a testing example x, we have to calculate a quantity that depends only on the inner product between x and the points in the training set. To utilize a kernel function for measuring the similarity between two examples in SVM, by the kernel trick we must replace inner product $\left\langle x^{(i)}, x \right\rangle$ in Equation 3.54 with the kernel $K(x^{(i)}, x)$.

In this approach, we represents each image as a spatial pyramid of histograms. To match, i.e. measure the similarity of, two histograms, we can use either intersection kernel or Chi-Square kernel. We prefer the Chi-Square kernel:

$$K(x^{(i)}, x^{(j)}) = 2\frac{x^{(i)}x^{(j)}}{x^{(i)} + x^{(j)}} \tag{3.55}$$

Explicit Feature Map

Although a non-linear SVM tends to yield better classification accuracy than a linear SVM [63], it has some significant drawbacks:

- Training using a non-linear kernel is much slower $O(n^3)$ than training a linear SVM $O(n)$.
- Testing is quite slow: the kernel must be computed between each testing example and all support vectors. For this reason, the testing time is linear $O(n)$ in the number of support vectors. Meanwhile the testing time of a linear SVM is only a constant.
- A non-linear SVM takes a lot of memory to store the Lagrange multipliers α and all the support vectors.

Linear SVMs are very fast to train [18] but limited to use inner products to compare descriptors, which is not suitable for several encodings. In this approach, we use a explicit feature map [60] in order to take advantages of the accuracy of a non-linear SVM and the speed of a linear SVM. To this end, a approximated feature map ϕ corresponding to kernel K is computed explicitly to transform input data x, z to a linear space in which we can exploit a linear SVM to train the transformed data with linear kernel $\langle \phi(x), \phi(z) \rangle$.

An explicit feature map can be used for a class of kernels, called the additive homogeneous kernels [60] $K(f, g) = \sum_{i=1}^{D} k(f_i, g_i)$ where k is itself a kernel defined on the non-negative reals. Our chosen kernel, Chi-Square kernel, is a additive homogeneous kernel. Hence, we can compute a approximated feature map for this kernel and then use it to transform image descriptors $\{x^{(1)}, ..., x^{(m)}\}$

resulted from the encoding stage to a linear space. Finally, we can use a linear SVM to efficiently train the transformed image descriptors as described in 3.4.3.

3.5.6 Summary

The spatial histograms form the image encoding stage cannot be efficiently trained in a linear SVM because it is limited to use an inner product to compare descriptors. Approach using Explicit Feature Map aims to transforming the image descriptors through a explicit feature map, which emulates a non-linear Chi-square kernel as a linear one. By this way, the approach can exploit both the high speed of a linear SVM and the high accuracy of a non-linear SVM.

3.6 Approach 4 - Using Locality-constrained Linear Coding

This approach belongs to the second group of approaches, which regards an image as a collection of image patches. The idea of the approach is to replace the conventional Vector Quantization coding, which is a hard quantization, with the Locality-constrained Linear Coding (LLC) [62], which is a soft quantization, in the quantization step. The LLC accompanied by a max pooling function makes the final image descriptors efficient to be trained in a linear SVM.
Specifically, this approach extracts dense SIFT descriptors for each image and encodes them by LLC based on a visual dictionary created by K-Means clustering algorithm. In each region of a spatial pyramid layout of the image, the features are max pooled. The pooled features of every regions are then concatenated to form the image descriptor. The descriptor is finally inputted to a linear SVM for training. The flow of this approach is shown in Figure 3.14.

3.6.1 Feature Extraction

In this approach, we extract a dense grid of SIFT descriptors is extracted from all training images as in the feature extraction step of Approach 3 (see 3.5.1).

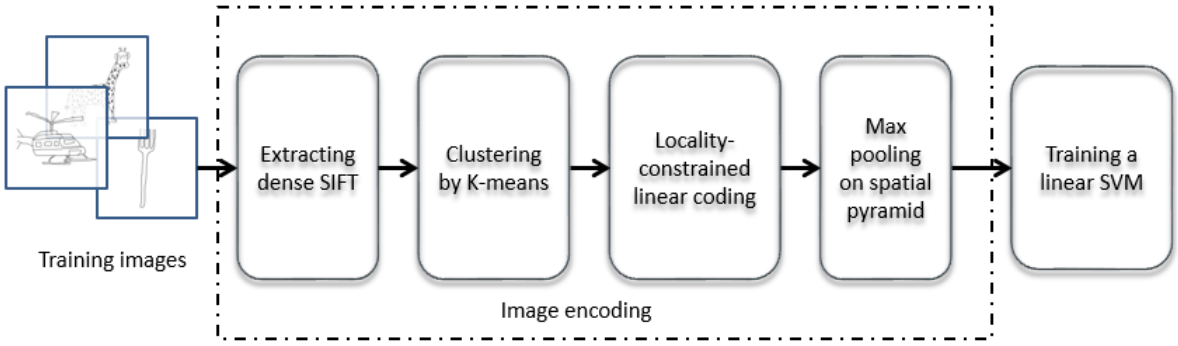


Figure 3.14: The flow of Approach 4.

After this step, we also attain a set of SIFT descriptors for each image $V^{(i)} = \{v_1^{(i)}, ..., v_N^{(i)}\}, i = 1, ..., m$ where m is the number of sketch images in our training set and i denotes the i-th image.

3.6.2 Visual Dictionary Construction

The space of SIFT descriptors is clustered using K-Means, which is already mentioned in 3.5.2, to construct a visual dictionary B of K visual words $\mu_1, \mu_2, ..., \mu_K$ in D-dimensional space.

3.6.3 Quantization

This step replaces the hard quantization of VQ coding, in which a descriptor is represented by only one visual word, by the soft quantization of locality-constrained linear coding (LLC) [62], in which a descriptor is represented by more than one visual word. Let $C^{(t)} = [c_1^{(t)}, ..., c_N^{(t)}]$ denote the corresponding codes for the set of descriptors of image $s^{(i)}$ which is $V^{(t)} = [v_1^{(t)}, ..., v_N^{(t)}]$ where N is the number of SIFT descriptors extracted for each sketch image. To reduce the loss that might be caused by VQ code, LLC code relaxes the restrictive constraint of VQ problem $\left\|c_i^{(t)}\right\|_{l^0} = 1$ in Equation 3.41 by adding a locality regularization term $\left\|d_i^{(t)} \odot c_i^{(t)}\right\|^2$, which makes the chosen visual words tend to be near the descriptor. Then the VQ problem becomes the following one:

$$\min_{C} \sum_{i=1}^{N} \left\| v_i^{(t)} - B c_i^{(t)} \right\|^2 + \lambda \left\| d_i^{(t)} \odot c_i^{(t)} \right\|^2 \\ s.t. 1^T c_i^{(t)} = 1, \forall i \tag{3.56}$$

where $\odot$ denotes the element-wise multiplication, λ is used to control the weight of the locality regularization term, and $d_i^{(t)} \in \mathbb{R}^K$is the locality controller giving freedom for each basis vector proportional to its similarity to descriptor $v_i^{(t)}$.

$$d_i^{(t)} = \exp(\frac{dist(v_i^{(t)}, B)}{\sigma}) \tag{3.57}$$

where $dist(x_i^{(t)}, B) = \left[dist(x_i^{(t)}, \mu_1), ..., dist(v_i^{(t)}, \mu_K)\right]^T$, and $dist(v_i^{(t)}, \mu_j)$ is the Euclidean distance between $v_i^{(t)}$ and μ_j. The constraint $1^T c_i^{(t)} = 1$ ensures the shift-variance of the LLC code.

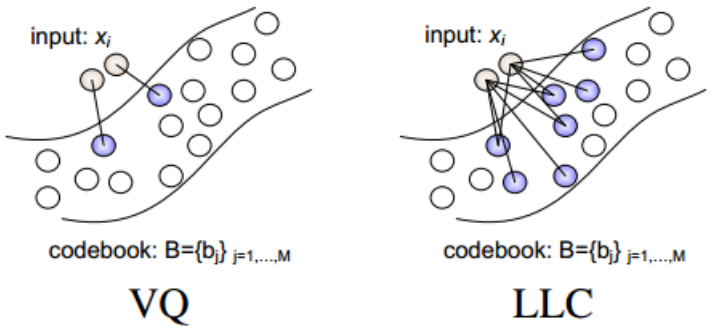


Figure 3.15: A comparison between Vector Quantization (VQ) and Locality-constrained Linear Coding (LLC). The chosen bases for the descriptor are highlighted in blue.

[7]

Unlike the VQ code that ignores the relationships between bases, the LLC code represents each descriptor more accurately by multiple bases, which gives less quantization error than the VQ code. Furthermore, the LLC code allows the learned representation to capture salient patterns of descriptors. An illustration showing the comparison between these 2 coding schemes can be seen in 3.6.3

[7]Image source: http://www.ifp.illinois.edu/ jyang29/LLC.htm

3.6.4 Building Image Descriptor

In this step, for each spatial region of a spatial pyramid, which is already mentioned in 3.5.4, the codes of the descriptors are pooled together to get the corresponding pooled feature. These pooled features from each spatial region are concatenated to build the final image descriptor for each image. Max pooling is chosen to be the pooling method in this approach.

In this approach, the representation $z_j^{(t)}$ of each spatial region for image $s^{(t)}$ is computed by max pooling on the absolute LLC codes as follows:

$$z_j^{(t)} = \max(\left|c_{1j}^{(t)}\right|, ..., \left|c_{Qj}^{(t)}\right|), j = 1, ..., K \tag{3.58}$$

where Q is the number of descriptors in that region of the image and K is the number of dimensions of LLC code $c_i^{(t)}$. Recall that in our problem $x^{(t)} \in \mathbb{R}^M$ denotes the image descriptor of sketch image $s^{(t)}$. Thus, the image descriptor $x^{(t)}$, which is formed by concatenating the representations of all spatial regions, has the number of dimensions as:

$$M = K \sum_{l}^{L} 4^l = K \frac{1}{3}(4^{L+1} - 1) \tag{3.59}$$

where L is the number of levels of the spatial pyramid. The image descriptors of all training images are used for training a classifier in the next stage.

3.6.5 Classifier Training

In this stage, the set of training example $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$, where $x^{(i)} \in \mathbb{R}^M$ and $y^{(i)}$ are the image descriptor and the class label of corresponding image $s^{(i)}$ respectively, are used to train a linear SVM as we describe in Approach using Sparse Autoencoder (see 3.4.3). We are able to use a linear SVM instead of a non-linear SVM with a non-linear matching kernel due to two reasons. The first one is that the LLC code of each descriptor has much less quantization errors

than VQ code. The second reason is that the representations of spatial regions, which is computed by max pooling, are salient and robust to local translation.

3.6.6 Summary

Approach using Locality-constrained Linear Coding uses LLC for the spatial pyramid of image instead of VQ coding in the traditional spatial pyramid so that a linear SVM can be effective used for training images.

3.7 Approach 5 - Using Kernel Codebook Encoding

This approach is suggested by Mathias Eitz et al in his SIGGRAPH paper [16] for their purpose of evaluating computational sketch recognition versus human sketch recognition. We evaluate this approach because it is one of state-of-the-art approaches in sketch recognition. This approach belongs to the second group of approaches, which considers an image collection of image patches. The authors uses Kernel Codebook Encoding [21, 48] for the quantization step and uses a non-linear SVM with a Gaussian kernel for evaluating the similarity of two training samples.

Specifically, in this approach, they first extract a large number of a kind of local feature similar to SIFT. A subset of these features are then used to constructed a visual dictionary using K-Means clustering algorithm. Afterwards, all features are quantized against the visual dictionary using kernel codebook coding. The codes of each image are then pooled by using average pooling to form a frequency histogram of visual words. The resulting histogram is considered the image descriptor, which is used for training a non-linear SVM with a Gaussian kernel. The flow of this approach is shown in Figure 3.16.

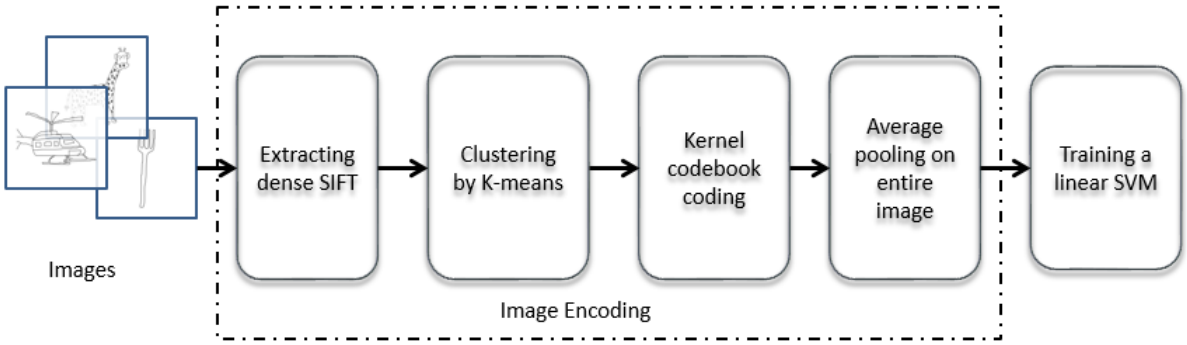


Figure 3.16: The flow of Approach 5.

3.7.1 Feature Extraction

The authors extract a dense grid of a kind of local descriptor which is closely related to the SIFT descriptor but stores orientations only. Each of the extracted descriptor is only a 64-dimensional vector. This step results in a set of descriptors for each image, $V^{(i)} = \{v_1^{(i)}, ..., v_N^{(i)}\}, i = 1, ..., m$ where m is the number of sketch images in our training set and i denotes the i-th image.

3.7.2 Visual Dictionary Construction

In this step, a subset of the descriptors are clustered into K clusters using K-means clustering algorithm as we demonstrate in 3.5.2. As a result, we obtain a dictionary to construct a dictionary B of K visual words $\mu_1, \mu_2, ..., \mu_K$.

3.7.3 Quantization

The authors uses kernel codebook coding [21, 48] as a soft quantization method in which each descriptor is assigned to multiple visual words. Let $C^{(t)} = [c_1^{(t)}, ..., c_N^{(t)}]$ denote the kernel codebook codes for set of descriptors $V^{(t)} = [v_1^{(t)}, ..., v_N^{(t)}]$ corresponding to i-th image, where N is the number of descriptors extracted for each sketch image. The kernel codebook code is computed as follows:

$$c_{ij}^{(t)} = \frac{\mathrm{K}(v_i^{(t)}, \mu_j)}{\sum_{k=1}^{K} \mathrm{K}(v_i^{(t)}, \mu_k)} \tag{3.60}$$

where i denotes the i-th image, j denotes the j-th element, μ_k denotes the k-th cluster, and $\mathrm{K}(v, \mu)$ is the distance between descriptor v and cluster μ which is measured by a Gaussian kernel:

$$\mathrm{K}(v, \mu) = \exp(-\frac{\gamma}{2}\|v - \mu\|^2) \tag{3.61}$$

The resulting kernel codebook code $c_i^{(t)}$ is an encoding of size K.

3.7.4 Building Image Descriptor

The histogram $z_i^{(t)}$ for image $s^{(t)}$ is computed by average pooling on the kernel codebook codes as follows:

$$z_j^{(t)} = \frac{1}{N} \sum_{i=1}^{N} c_{ij}^{(t)}, j = 1, ..., K$$

where N is the number of descriptors extracted for each image and K is the number of dimensions of VQ code $c_i^{(t)}$. Recall that in our problem $x^{(t)} \in \mathbb{R}^M$ denotes the image descriptor of sketch image $s^{(t)}$. Thus, the image descriptor $x^{(t)}$ is set to the histogram of visual words $z^{(t)}$:

$$x_i^{(t)} = z_i^{(t)}, i = 1, ..., M \tag{3.63}$$

Here, $M = K$. The resulting image descriptors are used as inputs for the classifier training stage.

3.7.5 Classifier Training

In this stage, the set of training example $\{(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})\}$, where $x^{(i)} \in \mathbb{R}^M$ and $y^{(i)}$ are the image descriptor and the class label of corresponding image $s^{(i)}$ respectively, are used to train a non-linear SVM with a Gaussian kernel:

$$\mathrm{K}(x^{(i)}, x^{(j)}) = \exp(-\frac{\gamma}{2}\left\|x^{(i)} - x^{(j)}\right\|^2) \tag{3.64}$$

3.7.6 Summary

Approach using Kernel Codebook Encoding achieves good classification accuracy because it utilize a non-linear SVM with the Gaussian kernel, which is suitable for the kernel codebook encoding scheme. However, the non-linear SVM has a long training time, which makes the approach impractical for real applications.

3.8 Conclusions

We make a brief summary of the five approaches as follows:

- **Approach using Neural Network**: This approach uses images' pixels for training a neural network. Since the images' pixels are raw data and the number of zero pixels in a sketch image is typically considerable, so the neural network cannot classify images with high accuracy.

- **Approach using Sparse Autoencoder**: This approach enhances Approach using Neural Network by encoding the raw data of pixels using a basic version of sparse autoencoder. The resulting representations from the sparse autoencoder are used as the image descriptors to train a linear SVM. Although this approach can scale well to new problems, it is hard for this approach to be competitive with or superior to the best hand-engineered representations. Perhaps more sophisticated versions of sparse autoencoder are able to do this.

- **Approach using Locality-constrained Linear Coding**: This approach replaces the Vector Quantization coding in traditional spatial pyramid matching scheme by a much more effective coding called Lacality-constrained Linear Coding. The combination of this coding scheme with a max pooling function enables the final image descriptor to be efficiently trained in a linear SVM. This efficiency allows this approach to be applied in real applications.

- **Approach using Kernel Codebook Encoding**: This approach trains images by using a non-linear SVM with a Gaussian kernel appropriate to the kernel codebook encoding that is also based on a Gaussian kernel, which yields high classification accuracy. However, using non-linear kernel makes the approach impractical for real applications.

- **Approach using Explicit Feature Map**: This approach uses the spatial pyramid of histograms to build the image descriptors. The image descriptors are compared by Chi-square kernel. However, to avoid using a non-linear SVM which increases the training and testing time, the approach computes a approximated feature map to transform the image descriptors to a linear space so that the distance between two descriptors calculated by inner product in that space is equivalent to the distance calculated by Chi-square kernel in original space. By using a explicit feature map, the approach can exploit both the high speed of a linear SVM and the high accuracy of a non-linear SVM. For this reason, we choose this approach for the sketch recognition module in our propose AR system.

Chapter 4

Experimental Results

At the first stage, we want to verify the efficient from computing accuracy as well as performance. From that, the evaluation is to choose the effective frameworks for Augmented reality system.

We choose sketch human dataset in research of Eitz at el. [16] as a practical data for experiments. Human sketch dataset includes 20000 images corresponding to 250 categories. For each category, there are 80 images scaled at size of 1111 $\times$ 1111 with format of ".png". Sketches in a category are the different versions of object and the most of strokes are similarly. Implementing a effective method for human sketch dataset is huge challenge. The previous results show that the accuracy human beings recognize objects achieved 73.1%. The best result in previous work computer recognition achieved 56% at 80 images per category training size and 20 test images per category and achieved 51% at 50 images training size and 30 test images per category. This can be considered to the significant work for dataset. Based on these properties of dataset, the implemented methods here can provide knowledge for computer.

Human sketch dataset has a huge sample for providing many data and knowledge for computer in learning as well as classification. The main purpose of implemented algorithms is to extract the efficient features or representation, which can be capable of and work well with classifying images for each category. There are two typical kinds of algorithm for experiments including algorithms inspired

by neuroscience such as neural network, or sparse autoencoder; and algorithm about type of encoding in spatial pyramid matching.

In all experiments, we use the powerful tools of Support Vector Machine LIBSVM [9] and LIBLINEAR [18] for classifying images. In addition, since the size of training image is too large for feature extraction (1111×1111). Specifically, we select 50 training and 30 testing images randomly per category. All experiments run on a computer with configuration of 8.00GB RAM, 64-bit Operating System of Intel(R) core(TM) i7 and CPU@ 2.20Hz.

4.1 Experiments setup

4.1.1 Approach 1-Using Neural Network

Algorithms inspired by neuroscience get the high of complexity as well as require the huge amount of time. The input image in neural networks is intensity pixel reshaped into column vectors as a input layer. Thus, computation can be very complicated if the number of size input is very big. Meanwhile size input of image is 480×480 which can cause many disadvantage for computation. To reduce the cost for computing, images is re-scaled into 32×32 which human beings can still recognize; and normalized pixels values to $[0, 1]$ (gray scale). In neural networks, the number of hidden size for hidden layer is chosen is 400 or 600. The number of iterations for back-propagation algorithm is 400.

#hidden neurons	accuracy (%)
400	20.61%
600	21.61%

Table 4.1: Neural network results for human sketch recognition

From results from Table 4.1, the accuracy is very low. The main reason here is

reducing the number dimension about size of input images, which causes lacking of the great number of information for representation of image to classifying in neural network. In the next experiment, the results of method inherited from neural network with higher level feature inputted to linear SVM.

4.1.2 Approach 2-Using Sparse Autoencoder

The number of hidden neurons for the sparse autoencoder is 400, i.e. the dimension of higher level features of each images is 400 ($f \in \mathbb{R}^{400}$). These features are extracted from the number of iterations in back-propagation algorithm of 100 and 400. After that, SVM model is to train and test features to predict label for test samples.

#iterations	accuracy (%)
100	28.41%
400	30.17%

Table 4.2: Sparse autoencoder results for human sketch recognition

The results in sparse autoencoder achieved show that this approach does not be efficient with low accuracy with base line [16]. However, averaged accuracy is higher than neural network about 8.18%. Next, different type of approach about spatial pyramid scheme can achieve better results.

4.1.3 Approach 3-Using Explicit Feature Map

The experiment is to verify the efficient of our proposed system mentioned in chapter 3. A spatial pyramid of visual words (PHOW) at levels of [1 2 4] and [2 4] is set up to implement encoding. Specially, the final feature vectors are also entries in spatial pyramid histogram with the number of visual words for vocabulary is 600,i.e. $K = 600$. By concatenating histogram at levels, The

number of dimensions of feature are 12600 and 12000 for levels [1 2 4] and [2 4] respectively. SVM is also used to classify with input of features histogram.

Round	Accuracy (%)
1	60.27%
2	60.61%
3	60.47%
4	59.80%
5	59.19%
6	59.87%
7	60.56%
8	59.68%
9	61.17%
10	60.55%
Average	**60.21**% ± **0.58** %

Table 4.3: Results of our approach at pyramid of [2 4] for human sketch recognition within 10 rounds.

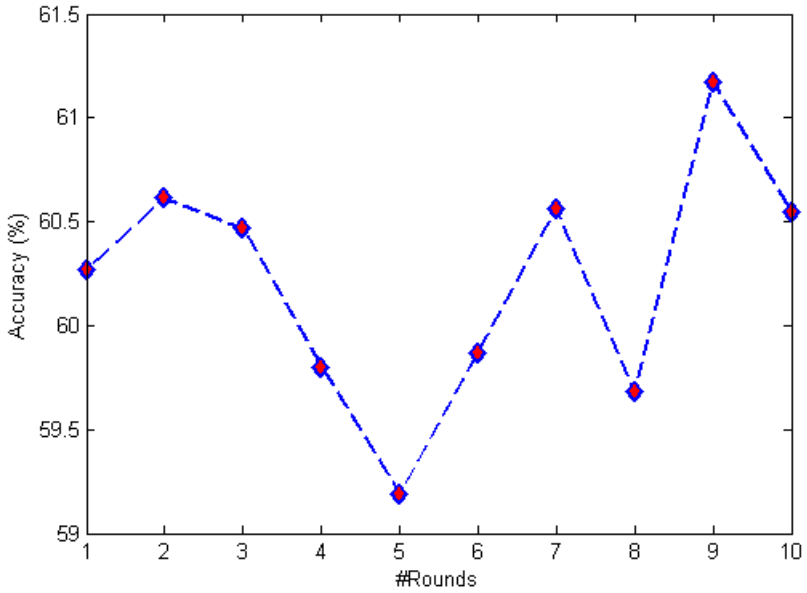


Figure 4.1: Results of our approach at pyramid of [2 4] visualization.

Table 4.3 and Figure 4.1 show the efficient of algorithm at pyramid of [2 4]. The averaged result achieved accuracy of 60.21%, which is higher than accuracy of base line in approach 4.1.5. From these promising results, approach at pyramid of [1 2 4] also is implemented to enhance classification of SVM to obtain the higher result.

Round	Accuracy (%)
1	62.79%
2	62.25%
3	62.50%
4	62.61%
5	61.65%
6	61.01%
7	61.96%
8	61.91%
9	61.79%
10	62.21%
Average	**62.07**% ± **0.52** %

Table 4.4: Results of our approach at pyramid of [1 2 4] for human sketch recognition within 10 rounds.

From Table 4.4 and Figure 4.2, accuracy is enhanced about 1.86%. With result of base line in approach 4.1.5, the result of accuracy is increasing about 11.97%. To make sense about these results, we also provide confusion matrix which represent the number of right prediction for each category at level pyramids of [2 4] and [1 2 4].

Figure 4.3 and 4.4 present confusion matrix of the highest accuracy within 10 rounds with the best results of 61.17% and 62.79% at pyramid [2 4] and [1 2 4] respectively. An effective method retrieve confusion such that entries on the diagonal are visualized because the value at that entry is the highest within 250 entries (in the same row for each row of confusion matrix).

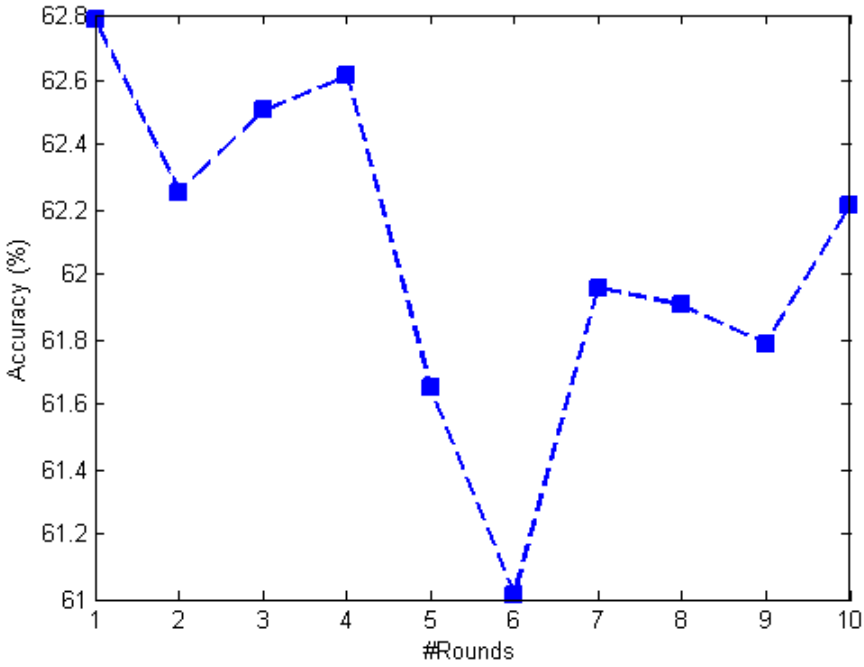


Figure 4.2: Results of our approach at pyramid of [1 2 4] visualization.

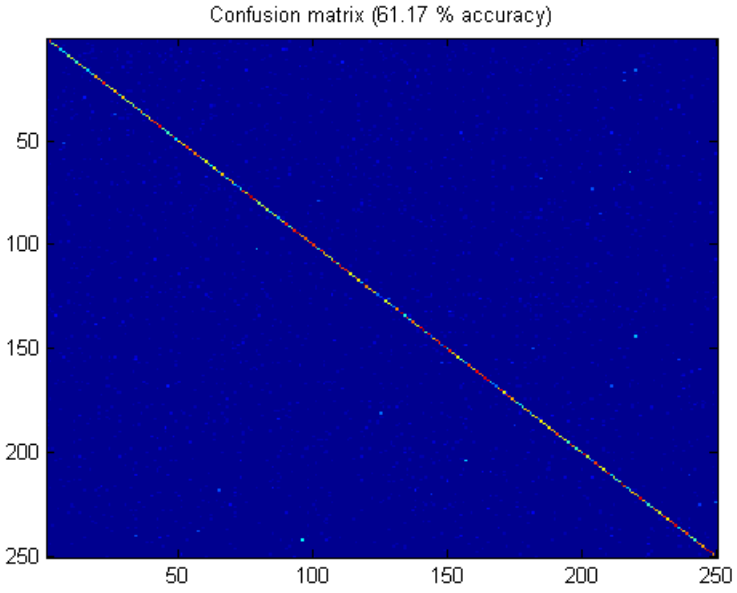


Figure 4.3: Confusion matrix at pyramid of [2 4] visualization

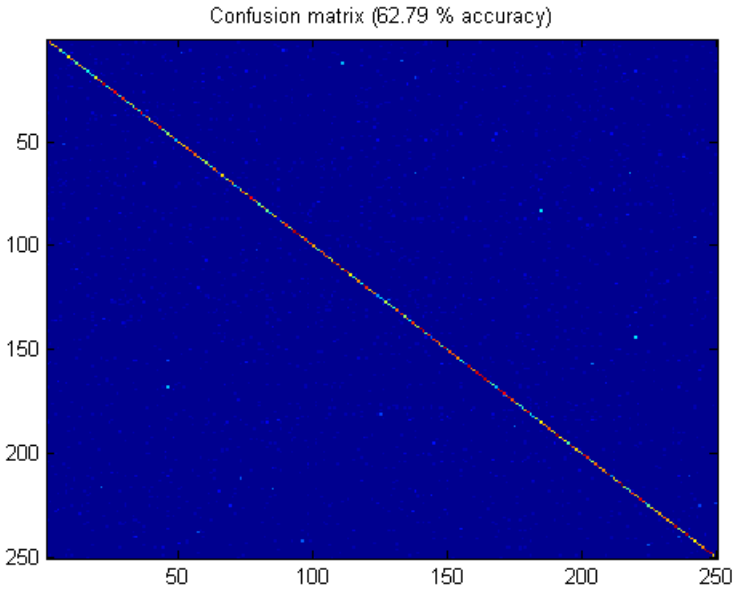


Figure 4.4: Confusion matrix at pyramid of [1 2 4] visualization

4.1.4 Approach 4-Using Locality-constrained Linear Coding

In locality linear coding [62], the dimension of feature vector of pyramid histogram [1 2 4] is given by.

$$1K + 4K + 16K = 21K. \tag{4.1}$$

where K is the number of visual words in codebook. In this case, $K = 1024$. Thus, the dimension of feature vectors is 21504 entries. Then, these feature vectors are inputted to LIBLINEAR SVM [18] for training and testing.

Because locality linear coding require the less cost of time [62], the algorithm runs for 10 rounds with randomizing 50-30 train-test sample per category. From Figure 4.5, the accuracy changes based on randomized data sample in dataset. These results are higher than results in neural network and sparse autoencoder, because locality linear coding use features like SIFT can reserve properties of original image better than scaling and normalizing images.

Round	Accuracy (%)
1	48.37%
2	48.15%
3	47.97%
4	48.58%
5	48.11%
6	48.01%
7	48.21%
8	47.75%
9	47.93%
10	48.18%
Average	**48.13**% ± **0.0023** %

Table 4.5: Results Locality linear coding in pyramid of histogram approach for human sketch recognition within 10 rounds (i.e. the number of times for running time.)

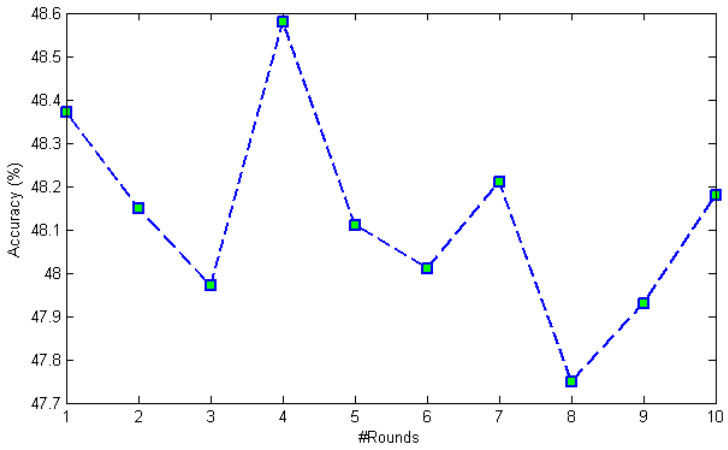


Figure 4.5: Locality linear coding result visualization.

4.1.5 Approach 5-Using Kernel Codebook Encoding

The experiment simulate method of Eitz at el. [16] used features similar to dense SIFT as local feature combine with Kernel codebook [21, 48] of soft quantization. The final feature is a dimension histogram of 500. Follow to Eitz at el. [16], we

classify data set by using Gaussian Kernel SVM with the best parameter $\gamma = 17.8$ and cost function $C = 3.2$. Similar to experiment 4.1.4, the experiment run within 10 rounds with randomizing samples data to compute averaged accuracy.

Round	Accuracy (%)
1	47.29%
2	48.19%
3	47.59%
4	46.73%
5	46.81%
6	46.39%
7	47.36%
8	47.21%
9	47.00%
10	47.33%
Average	**47.19**% ± **0.50** %

Table 4.6: Results base line approach [16] for human sketch recognition using classification of Gaussian kernel SVM within 10 rounds.

Figure 4.6 presents the change of accuracy with respect to randomized sample data. This approach achieved a promising result and well performance for application system.

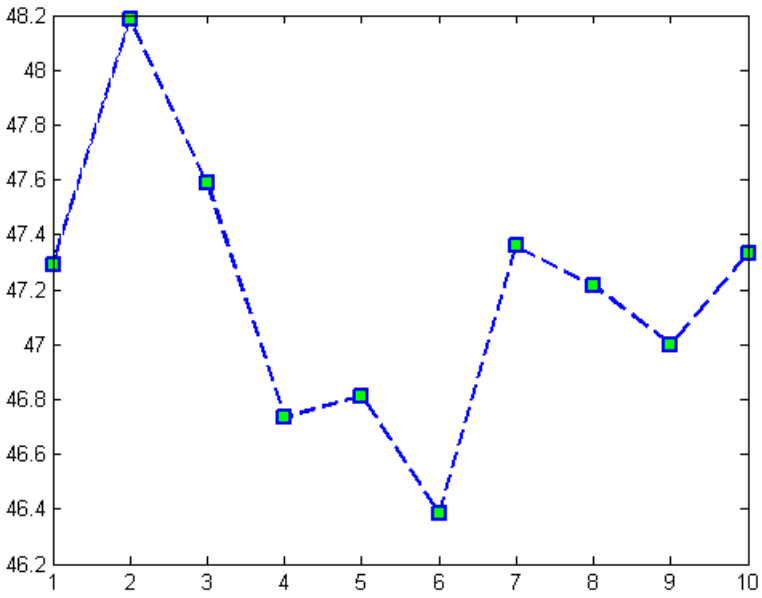


Figure 4.6: Based line result visualization.

4.2 Evaluation

To evaluate the disadvantage and advantage of approaches, the best results of each approach is presented in the following figure. The approaches use raw pixel

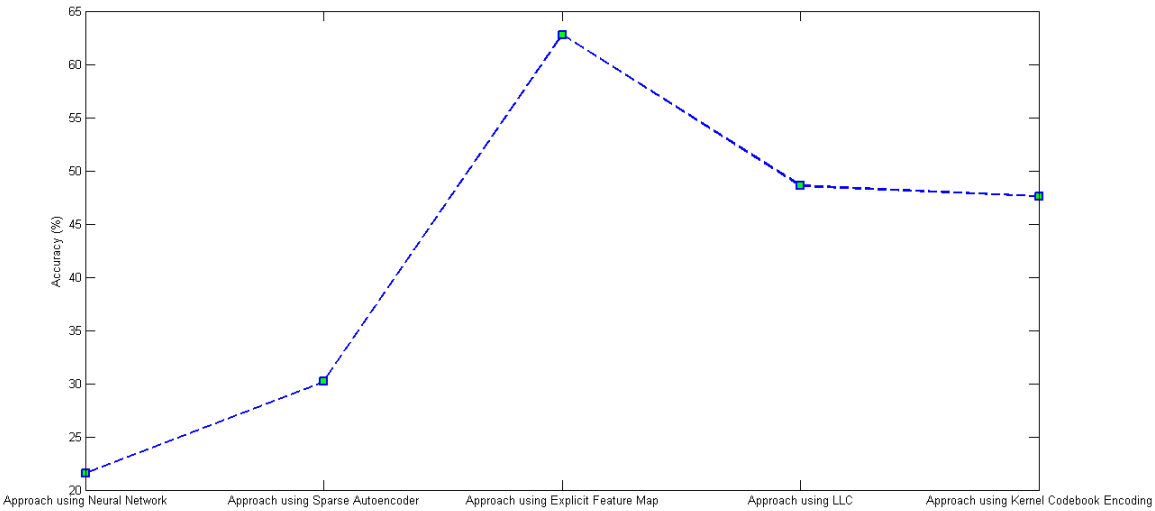


Figure 4.7: The best results of each approach.

is not suitable for sketch recognition because there are three following reasons.

- Re-scaling and normalizing input image cause lacking of the great number of information, which lead to the bad representation of image.
- The number of categories is very large (250 objects).
- The complexity of computation for hidden neuron is very big. Increasing the dimension of feature is very hard and require the great amount of time.

However, image representation in neural network and sparse autoencoder is simple. Local features is only input of intensity pixels and network learning features automatically without specific knowledge of experts. This can be considered to higher level features.

Using dense SIFT as local feature, the approaches achieve the promising result and is capable of the application of AR system. The approach 4.1.3 achieves the best results of 62.79%. In addition, the processing of testing or recognizing one image require the time of **0.6002** seconds including extracting final image descriptor and prediction based on linear SVM testing phase. This can be capable of AR system in real-time.

4.3 Summary

For each approaches, there still has the disadvantages and advantages in computation and implementation. However, From experiments, the approach 4.1.3 is the most suitable method within 5 approaches for AR system in future. In our work, we propose sketch recognition module for AR system. Thus, the accuracy and speed for computation in real time is very important. In general, a efficient sketch recognition approach will help users interact with AR system more flexibly and effective to sketch what they want.

Chapter 5

Proposed Augmented Reality System

In this chapter, we describe our proposed AR system which recognizes users' sketches and augment corresponding useful data, either multimedia or social-media information.

Specifically, types of augmented resources are used for information in AR system including image, text, audio, video, and 3D models. For each entity, we obtain the corresponding attributes described in AR information description, which can be external file module of XML file. With this module, user can select and configure plugins as well as related attributes of entities in AR system.

Thanks to select an effective sketch recognition algorithm tested in chapter 4, specific processes are established in Augmented Reality system. This system provides user a smart environment to interact with computer through sketching objects with query images captured from camera in real time. In addition, programmer can change the attributes and functions of plugins which are type of augmented resource through external handling file of XML in further.

In section 5.5, some demonstrations in real time are presented to describe interaction between human beings and computer through sketching objects. The objects here belong to the categories in sketch dataset mentioned in research of

Eitz at el. [16] consist of apple, bowl, bell, wheel, parrot, etc, which are type resources of image, text, audio, and video.

Additionally, we also try to demonstrate the another categories objects with 3D models combine corresponding videos information. The objects of nine planets in solar system consist of Sun, Mercury, Earth, Mars, Jupiter, Neptune, Uranus, Saturn, and Venus are presented form 3D models with video describes necessary information. This application can be applicable of education.

5.1 Overview

We propose an AR system with sketch recognition including 4 following processes:

- Visualization sub-system: This sub-system has 2 main functions. The first function is capturing frames from the real word and sending the query images to the processing block for recognizing. The second function is receiving the AR information corresponding to the query images and augmenting the information to the original frames.
- Processing sub-system: This sub-system processes the query image to recognize its category. This sub-system includes 2 processes: training process and testing process. These 2 phases are described in Section 5.2 and 5.3 respectively.
- AR information management sub-system: This sub-system manages AR information which is provided for the visualization sub-system to augment.

The overall AR system is shown in Fig. 5.1.

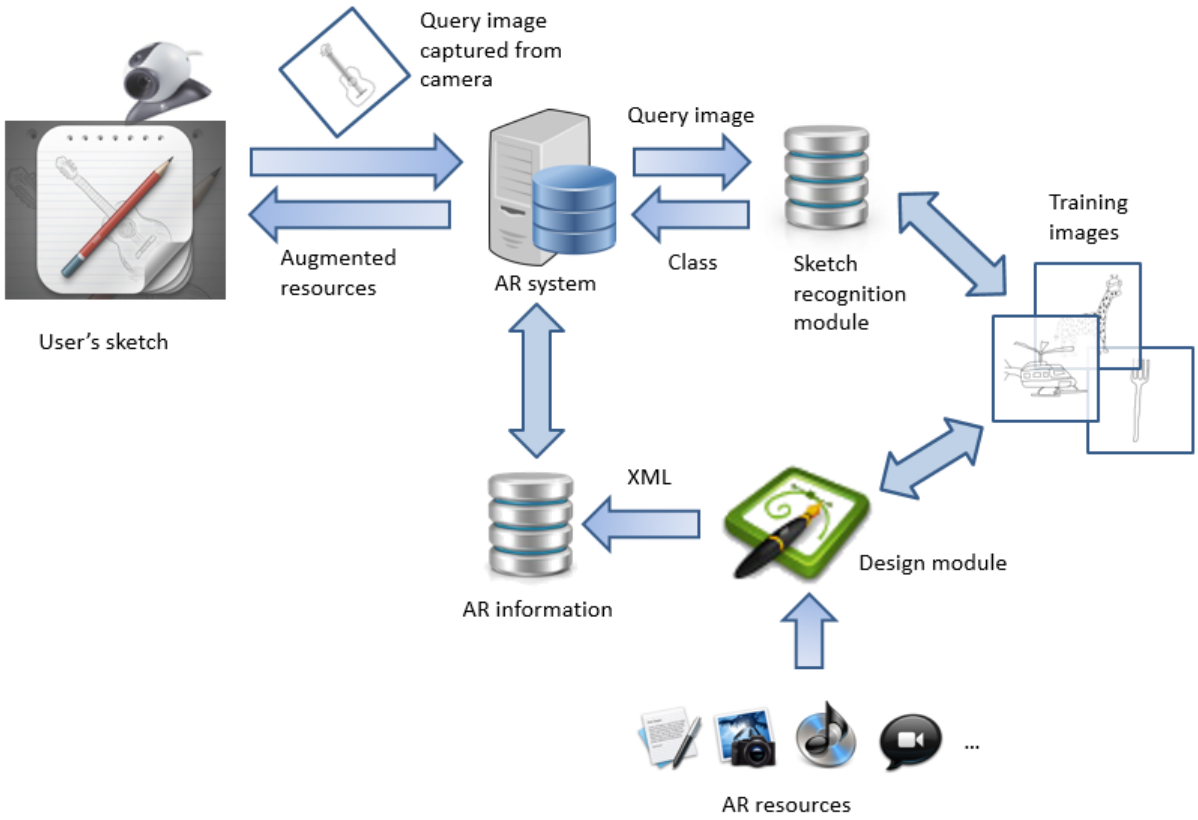


Figure 5.1: The overall AR system with sketch recognition.

5.2 Training process

The goal of the training process is to build classification models for the training set of sketches. To this end, the training set is first separated into different categories. The sketches of each category is then inputted to the sketch recognition module of our system to train classification models. Different image classification approaches can be used for this process. Currently in our system we choose Approach 5 (see Section 3.7). After finishing the training process, all trained models are stored in the sketch recognition module for later use in the testing process. The training process is briefly illustrated in Figure 5.2.

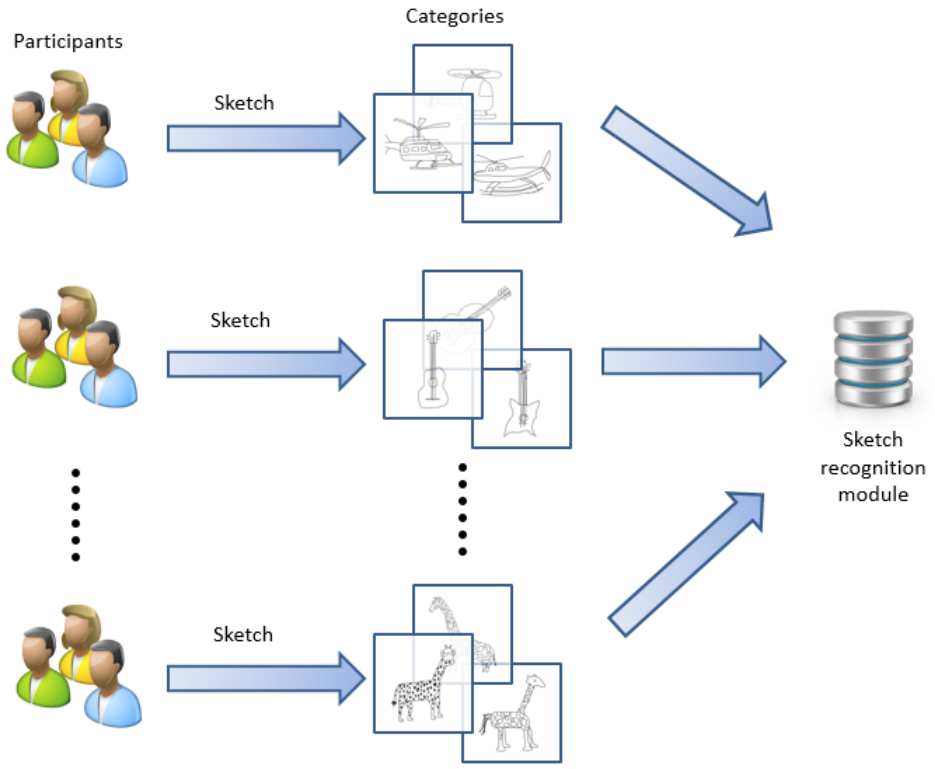


Figure 5.2: The training process of the system.

5.3 Testing process

In this process, a user's sketch is captured from the current frame by a camera. The sketch image is then passed to the AR module. Next, the AR module sends the query image to the sketch recognition module. Finally, the recognition module uses the classification models, which are already trained in the training process, to predict the category of the query image. The output of the testing process if the category to which the query sketch image belongs. The testing process is briefly illustrated in Figure 5.3.

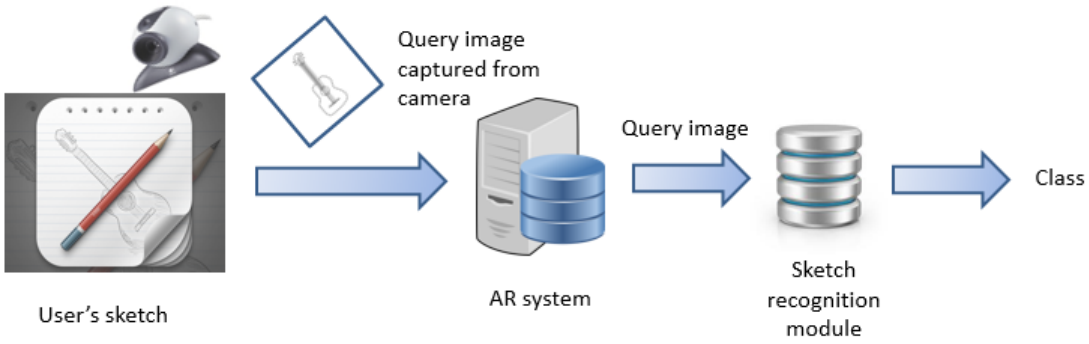


Figure 5.3: The testing process of the system.

5.4 Design Module

This sub-system enables users to define AR information corresponding to a specified category of sketch. Since AR information can be in many types such as multimedia information (images, audio, videos, etc.) or social media (webpages, texts, etc.), we apply a plugin architecture for the design module as illustrated in Figure 5.4.

The AR information description processor processes AR information description files (in XML format) to classify them into categories. For each category, the AR information plugin manager choose a suitable plugin to be responsible for that category. The goal of a plugin is to enable users to select resources and configure attributes of information related to AR resources through XML files that describes AR information description (see Figure 5.5).

Training images are inputted to User interface for training in sketch recognition module. The categories of entities are created in Augmented Reality system through training process. With Augmented reality information plugin manager, user can select necessary augmented resource to add entities for AR system. There is a constrain in design module. In case of change plugins, user want to declare new plugin or new entity of the categories, the manipulation in XML file have to contain that entity with corresponding plugins. In general, user make

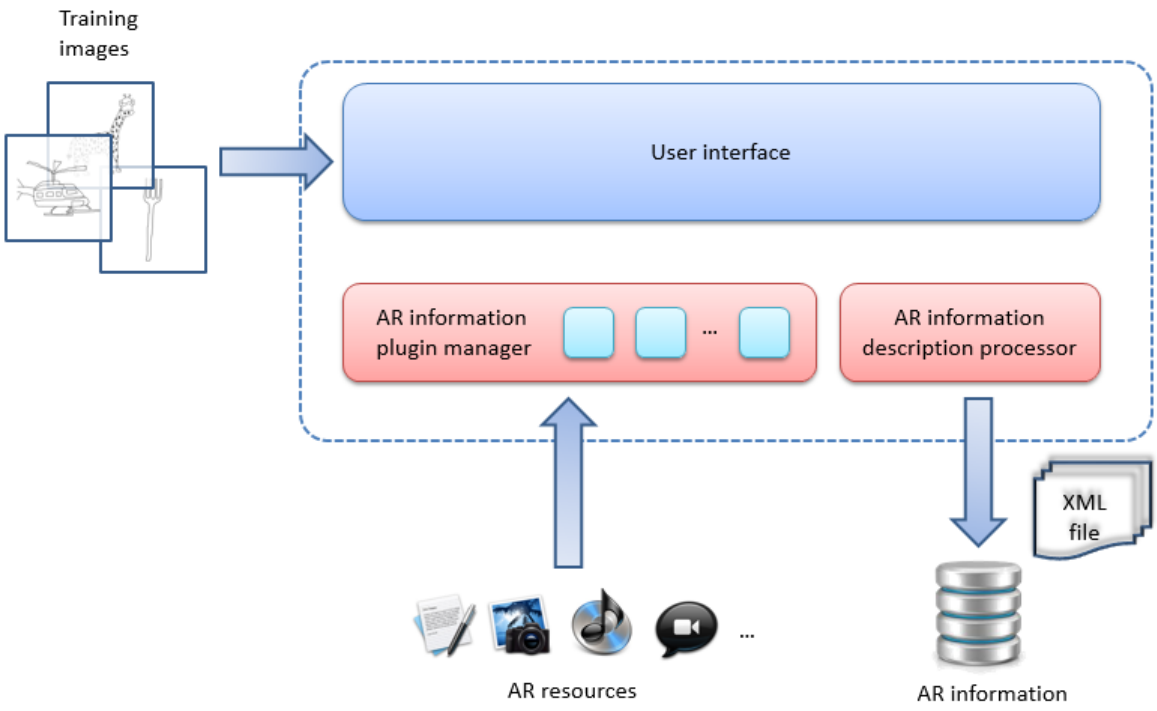


Figure 5.4: Desgin module.

declaration through external XML file module. This helps user approach the Augmented Reality system naturally and easier.

For each type of augmented information, there is a plugin component in subsystem design module. The main functions of these plugins are as follow.

- The plugins help user be able to select augmented resources and configure its attributes, which relate augmented information. In addition, the system extract descriptions of resource through external XML file.
- Programmer can add new plugin to the system by declaring new entity class in system and manipulate in XML file with necessary augmented resource.

In the section 5.5, we will provide some demonstration of Augmented Reality system, which help users obtain interesting experiences with lively and attractively information of objects of dataset research of Eitz at el. [16] as well as planets in solar system applied for education.

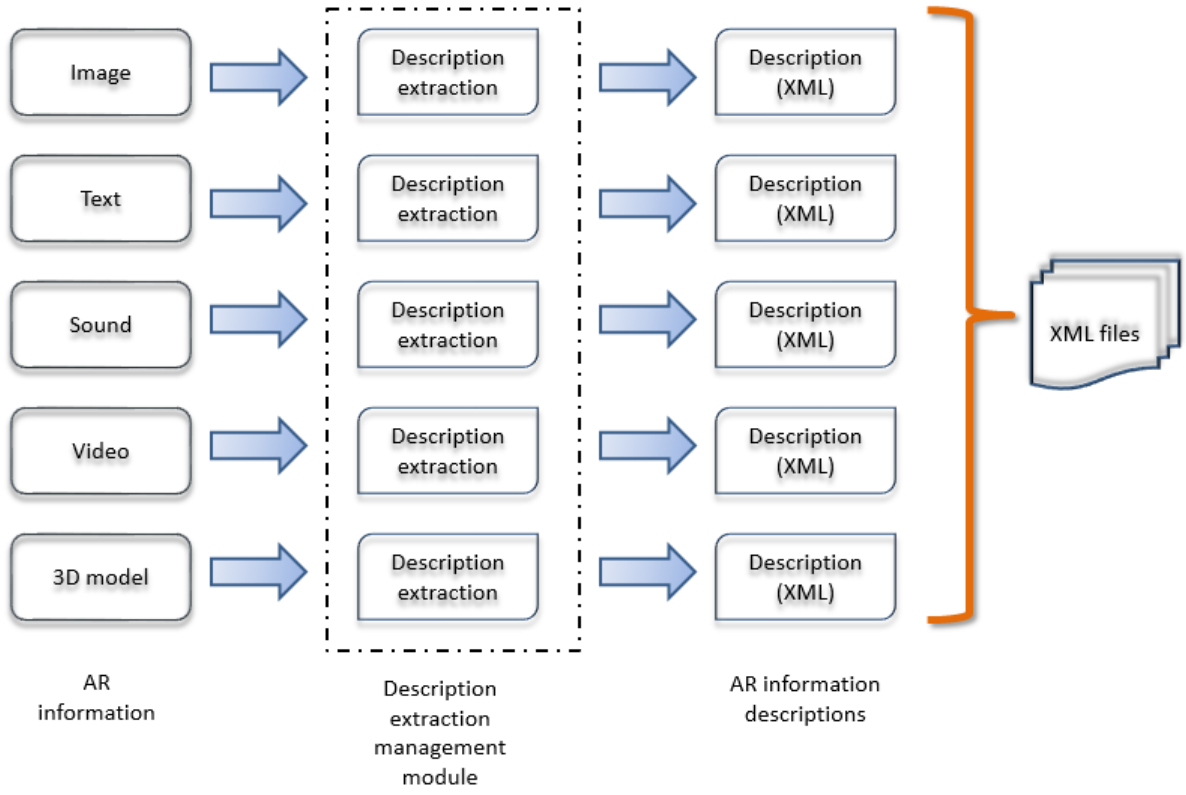
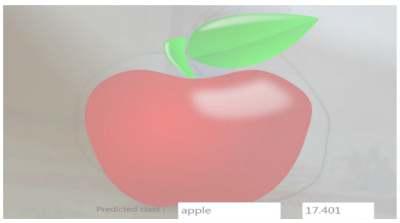


Figure 5.5: AR information description.

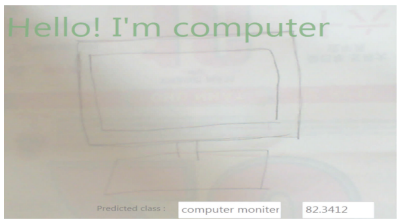
5.5 Demonstrations

In our system, we help users interact with nine entities which divided into two areas. The first area is objects belong categories in dataset mentioned in research of Eitz at el. [16] consist of apple, bowl, bell, computer monitor, candle, cloud, wheel, pizza, and parrot. Specifically, the image resource plugin is augmented to apple. The text plugin is augmented to bowl and computer monitor. The audio plugin is augmented to bell. And the video plugin is augmented to candle, cloud, pizza and parrot.

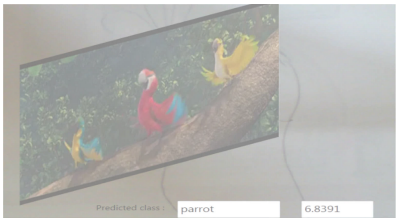
The second area of demonstration, the objects belong categories in solar system consist of Sun, Mercury, Earth, Mars, Venus, Jupiter, Uranus, Saturn, and Neptune. The augmented resources are combine between 3D models and videos



(a) Apple is image plugin.



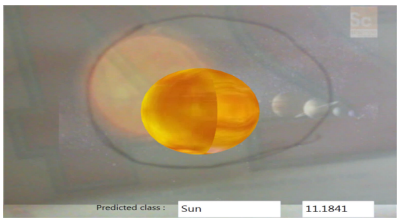
(b) Computer moniter is text plugin.



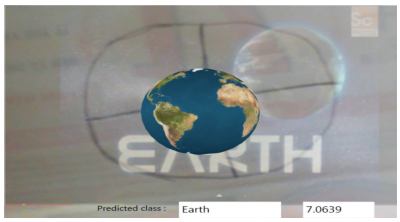
(c) Parrot is video plugin.

Figure 5.6: Demonstration for dataset mentioned in research of Eitz at el. [16]

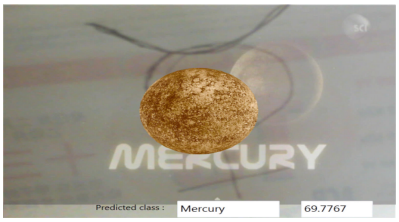
describe for the corresponding planets in real frame.



(a) The Sun.



(b) The Earth.



(c) The Mercury.

Figure 5.7: Demonstration for planets in solar system with 3D models and dissertation video.

The inputted frames for demonstration are images of planet signs in astrology,

which help distinguish shape of planets. User needs to remember these signs to describe for planets in solar system.

5.6 Summary

The proposed Augmented Reality system is architecture of modules combination with components of systems, which are Visualization sub-system, Processing sub-system, and Augmented Reality management sub-system. In addition, some external modules such as sketch recognition module and XML file module are combined to the system.

In the Augmented Reality system, user can create new training images for sketch recognition module. For instance, dataset of planet signs in solar system mentioned in section 5.5 with interesting augmented resources. This can be applicable of education in school for children to help approach the science of astrology as well as useful knowledge about life naturally and lively.

Chapter 6

Conclusions and Future work

6.1 Conclusions

In the thesis, we study techniques and smart systems of HCI. During the study, we specially have a major concern for AR, which is a very hot trend of HCI and promises to be widely applied in daily lives. We study applications, properties of an AR system as well as frameworks used in AR. We notice that most of conventional AR systems focus only on enhancing users' experiences in the rendering phase by enables them to interact with virtual augmented information. In contrast, the detection phases of AR in those systems mainly utilize a traditional method that uses inputs of printed predefined templates for detection. We desire to replace these inputs with a new means of input, which is specifically users' sketches in the thesis, so that the way of creating AR inputs becomes more flexible and bring more exciting experiences to users. For this reason, we propose and develop a smart environment of AR in which the system is able to recognize sketches from users and provide helpful multimedia and social-media information related to their sketches. The proposed system has much practical value and can be applied in different sectors, especially education due to the flexible and lively inputs of sketches.

To deal with sketch recognition in the system, we study the problem of sketch recognition and approaches that have been used to recognize sketches. Among the approaches, we focus on image classification approaches in Machine Learn-

ing. The reason is that sketch recognition should be considered a problem of category-level object recognition instead of instance recognition due to a wide variance of users' sketches. For image classification, we evaluate five different approaches, which follow recent trends in the image classification field of Machine Learning, to figure out the most suitable approach among them for applying in the sketch recognition module of our proposed AR system.

For the purpose of evaluation, we use a standard sketch dataset published by Eitz et al in his SIGGRAPH 2012 paper [16]. After evaluation the approaches, we realize that Approach using Neural Network has too low classification accuracy (21.61%) since this approach is only based on raw data of all pixels. Approach using Sparse Autoencoder is also based on pixels but extract a more effective representation through a sparse autoencoder. Although the accuracy is raised a little bit (30.17%), the approach is still not efficient enough to be used. Approach using Kernel Codebook Encoding, which is suggested by Eitz in the mentioned paper, yields a quite high accuracy (we obtain 47.19% while they achieves 51% as described in the paper, maybe due to his optimization). Our other evaluated approaches are competitive with or superior to their approach. Specifically, Approach using Locality-constrained Linear Coding is nearly as efficient as their approach (48.13%). In addition, due to the properties of locality-constrained linear coding, this approach exploits a linear kernel instead of a non-linear kernel of their approach, which increases both training and testing time. Remarkably, Approach using Explicit Feature Map is superior to their approach in both accuracy (62.07%) and speed. This approach computes a explicit feature map to transform image features from a non-linear space to a linear one so that it can exploit both the high accuracy of a non-linear classifier and the very high speed of a linear classifier. We decide to choose Approach using Explicit Feature Map for the sketch recognition module due to not only its high classification accuracy but due to its short testing time, which enables our AR system to run in real-time.

We test our AR system in a real-time environment and actually the system

has good stability, high accuracy, and fast response as expected. In addition, to demonstrate an example application of geography education, we create ourselves an extra sketch dataset of planet signs with corresponding augmented multimedia information. Moreover, the sketch recognition module is designed to be totally independent of the proposed AR system. Therefore, if any approach yields a better classification performance than Approach using Explicit Feature Map, we will replace the current approach with the new one for our sketch recognition module without changing the architecture of the AR system. A better recognition module will definitely raise the performance of the overall AR system.

6.2 Future work

For the sketch recognition approach, we plan to improve as follows:

- Use a multiple kernel classifier [57], for example a multiple kernel [59] which is a combination of dense SIFT, self-similarity and geometric blur features.
- Try state-of-the-art encodings such as Fisher encoding [47] and Super Vector Encoding [64].

For the AR system, we plan to improve as follows:

- Enable the system to recognize multiple sketches simultaneously.
- Apply some techniques to help the system stable in various contexts such as light, wind.
- Create many more practical datasets so that the AR system can be applied widely.

REFERENCES

[1] C. M. Adams, "Constructing symmetric ciphers using the cast design procedure," *Des. Codes Cryptography*, vol. 12, no. 3, pp. 283–316, 1997.

[2] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, Aug. 1997.

[3] J. A. Bagnell and D. M. Bradley, "Differentiable sparse coding," in *NIPS*, 2008, pp. 113–120.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[5] A. Bhat and T. Hammond, "Using entropy to distinguish shape versus text in hand-drawn diagrams," in *Proceedings of the 21st international jont conference on Artifical intelligence*, ser. IJCAI'09. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 1395–1400. [Online]. Available: http://dl.acm.org/citation.cfm?id=1661445.1661669

[6] A. Bosch, A. Zisserman, and X. Muñoz, "Scene classification via plsa," in *ECCV (4)*, 2006, pp. 517–530.

[7] J. P. Campbell and Jr., "Speaker recognition: A tutorial," 1997.

[8] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.1986.4767851

[9] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[10] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, "Sketch2photo: internet image montage," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 124:1–124:10, Dec. 2009. [Online]. Available: http://doi.acm.org/10.1145/1618452.1618470

[11] H. Choi and T. Hammond, "Sketch recognition based on manifold learning," in *AAAI*, 2008, pp. 1786–1787.

[12] I. Cohen, N. Sebe, L. Chen, A. Garg, and T. S. Huang, "Facial expression recognition from video sequences: Temporal and static modelling," in *Computer Vision and Image Understanding*, 2003, pp. 160–187.

[13] P. Corey and T. Hammond, "Gladder: Combining gesture and geometric sketch recognition," in *AAAI*, 2008, pp. 1788–1789.

[14] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[15] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Comput. Surv.*, vol. 40, no. 2, pp. 5:1–5:60, May 2008. [Online]. Available: http://doi.acm.org/10.1145/1348246.1348248

[16] M. Eitz, J. Hays, and M. Alexa, "How do humans sketch objects?" *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 31, no. 4, pp. 44:1–44:10, 2012.

[17] M. Eitz, R. Richter, K. Hildebrand, T. Boubekeur, and M. Alexa, "Photosketcher: interactive sketch-based image synthesis," *IEEE Computer Graphics and Applications*, 2011.

[18] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[19] B. Fasel and J. Luettin, "Automatic facial expression analysis: A survey," *PATTERN RECOGNITION*, vol. 36, no. 1, pp. 259–275, 1999.

[20] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *CVPR (2)*, 2005, pp. 590–596.

[21] J. C. Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *Proceedings of the 10th European Conference on Computer Vision: Part III*, ser. ECCV '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 696–709. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88690-7_52

[22] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *ICCV*, 2005, pp. 1458–1465.

[23] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007. [Online]. Available: http://authors.library.caltech.edu/7694

[24] S. Hai-Jew, *Virtual Immersive and 3d Learning Spaces: Emerging Technologies and Trends*, ser. Premier reference source. Information Science Reference, 2011. [Online]. Available: http://books.google.com.vn/books?id=B5zUPXWwOZ8C

[25] M. Haller, M. Billinghurst, and B. H. Thomas, *Emerging technologies of augmented reality - interfaces and design*. Idea Group Publishing, 2007.

[26] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[27] A. Jaimes and N. Sebe, "Multimodal human-computer interaction: A survey," *Comput. Vis. Image Underst.*, vol. 108, no. 1-2, pp. 116–134, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2006.10.019

[28] I.-J. Kim, "Introduction to augmented reality and its applications: (copyright restrictions prevent acm from providing the full text for this article)," in *ACM SIGGRAPH ASIA 2010 Courses*, ser. SA '10. New York, NY, USA: ACM, 2010, pp. 1:1–1:1. [Online]. Available: http://doi.acm.org/10.1145/1900520.1900521

[29] M. Knecht, C. Traxler, O. Mattausch, W. Purgathofer, and M. Wimmer, "Differential instant radiosity for mixed reality," in *Proceedings of the 2010 IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2010)*, Oct. 2010, pp. 99–107, best Paper Award! [Online]. Available: http://www.cg.tuwien.ac.at/research/publications/2010/knecht_martin_2010_DIR/

[30] T. D. Kristen Grauman, "The pyramid match kernel: Efficient learning with sets of features," *Journal of Machine Learning Research*, vol. 8, pp. 725–760, 2007.

[31] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR (2)*, 2006, pp. 2169–2178.

[32] W. Lee, Y. Park, V. Lepetit, and W. Woo, "Point-and-shoot for ubiquitous tagging on mobile phones," in *ISMAR*, 2010, pp. 57–64.

[33] Y. J. Lee, C. L. Zitnick, and M. F. Cohen, "Shadowdraw: real-time user guidance for freehand drawing," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 27:1–27:10, Jul. 2011. [Online]. Available: http://doi.acm.org/10.1145/2010324.1964922

[34] J. P. Lewis, "Fast normalized cross-correlation," 1995.

[35] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–136, 1982.

[36] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[37] M. J. Lyons and et al., "Designing, playing, and performing with a vision-based mouth interface," in *CONF. ON NEW INTERFACES FOR MUSICAL EXPRESSION*, 2003, pp. 116–121.

[38] O. L. Mangasarian and D. R. Musicant, "Lagrangian support vector machines," 2000.

[39] S. Martedi, H. Uchiyama, G. Enriquez, H. Saito, T. Miyashita, and T. Hara, "Foldable augmented maps," *IEICE Transactions*, vol. 95-D, no. 1, pp. 256–266, 2012.

[40] A. Mohan, G. Woo, S. Hiura, Q. Smithwick, and R. Raskar, "Bokode: imperceptible visual tags for camera based interaction from a distance," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 98:1–98:8, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1531326.1531404

[41] P. Mountney, S. Giannarou, D. Elson, and G.-Z. Yang, "Optical biopsy mapping for minimally invasive cancer screening," in *Proceedings of the 12th International Conference on Medical Image Computing and Computer-Assisted Intervention: Part I*, ser. MICCAI '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 483–490. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04268-3_60

[42] V.-T. Nguyen, T.-N. Le, Q.-M. Bui, M.-T. Tran, and A. D. Duong, "Smart shopping assistant: A multimedia and social media augmented system with mobile devices to enhance customers' experience and interaction," in *PACIS*, 2012, p. 95.

[43] S. Oviatt, "The human-computer interaction handbook," J. A. Jacko and A. Sears, Eds. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 2003, ch. Multimodal interfaces, pp. 286–304. [Online]. Available: http://dl.acm.org/citation.cfm?id=772072.772093

[44] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. Pattern Anal. Mach.*

Intell., vol. 32, no. 3, pp. 448–461, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2009.23

[45] M. Pantic, S. Member, and L. J. M. Rothkrantz, "Automatic analysis of facial expressions: The state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1424–1445, 2000.

[46] B. Paulson and T. Hammond, "Paleosketch: accurate primitive sketch recognition and beautification," in *Proceedings of the 13th international conference on Intelligent user interfaces*, ser. IUI '08. New York, NY, USA: ACM, 2008, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/1378773.1378775

[47] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV (4)*, 2010, pp. 143–156.

[48] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *CVPR*, 2008.

[49] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[50] D. Rubine, "Specifying gestures by example," *SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 329–337, Jul. 1991. [Online]. Available: http://doi.acm.org/10.1145/127719.122753

[51] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: A database and web-based tool for image annotation," *International Journal of Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[52] D. Schmalstieg and D. Wagner, "Experiences with handheld augmented reality," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ser. ISMAR '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–13. [Online]. Available: http://dx.doi.org/10.1109/ISMAR.2007.4538819

[53] T. M. Sezgin, "Sketch based interfaces: Early processing for sketch understanding," in *Proceedings of PUI-2001. NY.* ACM Press, 2001.

[54] F. Sha, Y. Lin, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," in *Advances in Neural and Information Processing Systems.* MIT Press, pp. 897–904.

[55] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, 1994, pp. 593 – 600.

[56] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *In Shape Modeling International*, 2004, pp. 167–178.

[57] M. Varma and D. Ray, "Learning the discriminative power-invariance trade-off," in *ICCV*, 2007, pp. 1–8.

[58] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[59] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *ICCV*, 2009, pp. 606–613.

[60] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, 2012.

[61] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," 2007.

[62] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *CVPR*, 2010, pp. 3360–3367.

[63] J. Zhang, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: a comprehensive study," *International Journal of Computer Vision*, vol. 73, p. 2007, 2007.

[64] X. Zhou, K. Yu, T. Zhang, and T. S. Huang, "Image classification using super-vector coding of local image descriptors."

[65] Y. H. Zweiri, "Optimization of a three-term backpropagation algorithm used for neural network learning," *International Journal of Computational Intelligence*, vol. 3, p. 2006.

APPENDIX

We provide some samples of categories in Human Sketch Recognition dataset.

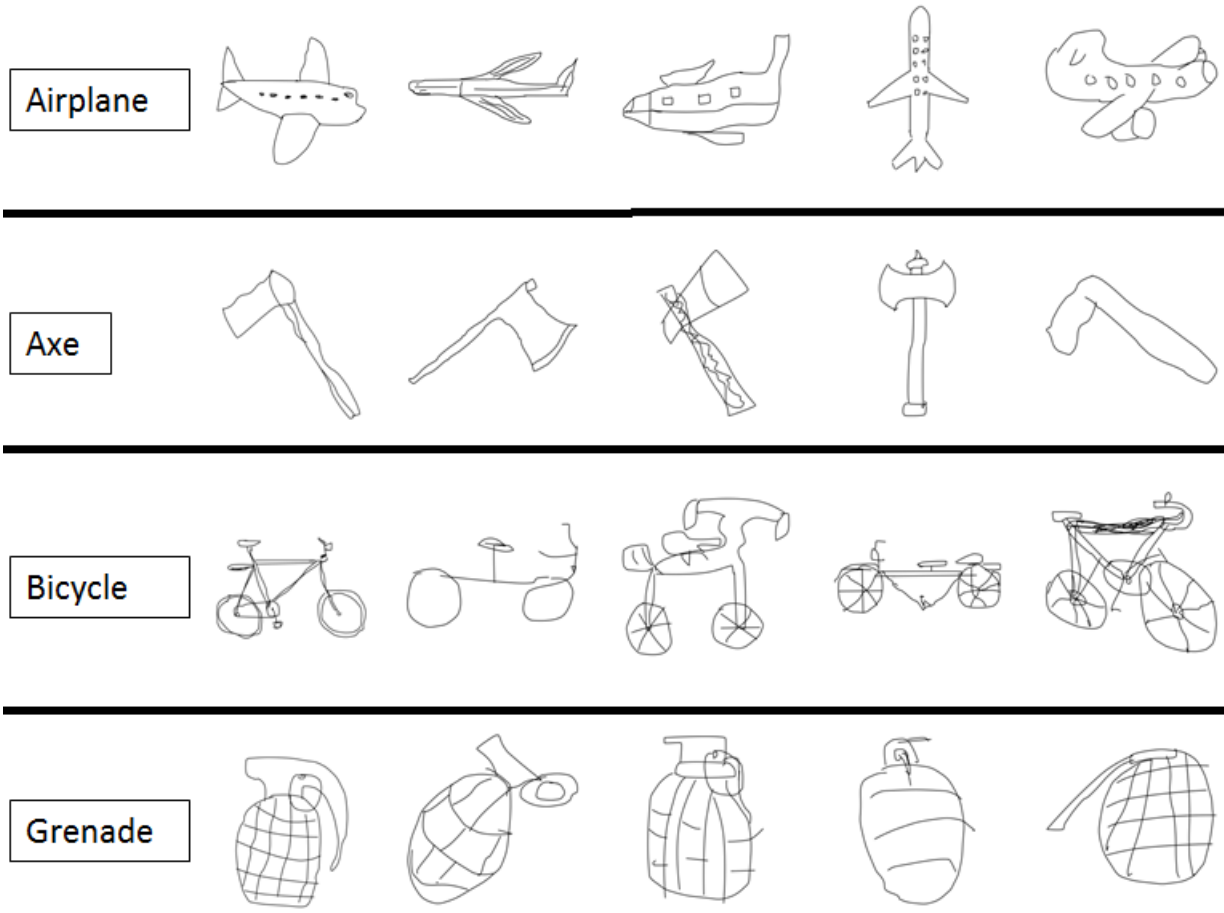


Figure 6.1: Visualization of 5 samples for 4 categories of Airplane, Axe, Bicycle, and Grenade