Intermediary System Using Image Classification for Online Shopping

Yunan Liu
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd
Part of the Electrical and Computer Engineering Commons

BYU ScholarsArchive Citation
Liu, Yunan, "Intermediary System Using Image Classification for Online Shopping" (2016). All Theses and Dissertations. 6015.
https://scholarsarchive.byu.edu/etd/6015

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Intermediary System Using Image Classification for Online Shopping

Yunan Liu

A thesis submitted to the faculty of
Brigham Young University
In partial fulfillment of the requirements for the degree of
Master of Science

Dah-Jye Lee, Chair
James Archibald
Doran Wilde

Department of Electrical and Computer Engineering
Brigham Young University
June 2016

Copyright © 2016 Yunan Liu
All Rights Reserved
ABSTRACT

Intermediary System Using Image Classification for Online Shopping

Yunan Liu
Department of Electrical and Computer Engineering, BYU
Master of Science

Online shopping is becoming a popular option for consumers. Currently, the most common product searching method that online shopping websites provide is keyword search. Most shoppers have to carefully select relevant keywords to search for their favorite products. Finding desired products using a query image for online shopping is currently not available. Image has been used for searching similar images in the database but they are usually not well annotated. Research effort has been devoted to developing reliable image-based retrieval systems for applications such as medical image retrieval and trademark search. None of these developments focuses on improving online shopping experiences for consumers.

This thesis reports the development of an image retrieval system to provide better online shopping experience for consumers. The system searches products with similar appearance such as shape and textures to the query images the user provides. Turn angle is a contour based shape descriptor. It has many unique properties that make it a perfect shape matching method for image retrieval. The best matching image has the shortest shape distance to the query shape. Turn angle, however, could fail with slightly stretched shapes. Dynamic programming is used to help turn angle match slightly deformed shapes. Another technique called centroid distance is also included as a restriction for shape matching in order to avoid retrieving irrelevant or disparate shapes. With a well-built database, the enhanced turn angle descriptor that includes dynamic programming and centroid distance is able to reach a high accuracy rate.

Shape matching alone is usually not sufficient for a powerful retrieval system. Products with similar shape but very different textures will not be distinguished based solely on shape matching. Edge histogram is a robust shape descriptor for texture matching. It can be implemented to construct either global or local histogram for this purpose. Global edge histogram uses only 5 bins, which is simple but ignores detail texture information. Local and semi-global edge histograms are more complex but retains detail texture information. A hierarchical matching system is built to combine the shape and texture descriptors for better retrieval accuracy.

Easy access to the shopping system is desired. An Android Application is developed to provide consumers a convenient and friendly tool to use the system. Grab cut is applied to the captured image to segment the object from the background. The segmentation provides the retrieval system the required contour information for shape matching. The Android Application submits the captured image along with the segmented contour to the server. After the retrieval process is completed, the server sends retrieved images of similar products back to the Android App for the user to consider. Using the retrieval system via a handheld device provides a user-friendly online shopping experience.

Keywords: CBIR, Turn Angle, Edge Histogram Descriptor, Android Application
ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Dah-Jye Lee, for his guidance, help and patience. He has always been patient and willing to spend time to discuss my research even after school hours. I really appreciate the personal time he spent to help me. His spirit motivates me every time I ran into a problem or felt discouraged. He is not only an advisor or my research but also a mentor for my personal development in life.

I would also like to thank Professor Doran Wilde for the thoughts of algorithm design. Theories included in this thesis could not be implemented efficiently without that knowledge. I also want to express my sincere thanks to another member of my committee Professor James Archibald, for serving on my committee and the support throughout my graduate study here.

My study at BYU is going to end. Remembering these two years I have spent here, I really enjoyed meeting nice people who share the same interest and passion in technologies with me and appreciate the time they spent with me working on projects overnight. I would like to thank them all epically my friends in the Robotic Vision Lab at BYU. I hope our friendships will last forever.

Finally, I want to give my most sincere thanks to my parents for their support and love. This work is dedicated to them. They gave the best without getting anything in return. I can always feel their meticulous care and warmest love even if I am far away from home. Thank you for always being there for me.
# TABLE OF CONTENTS

LIST OF TABLES ........................................................................................................................................... vi
LIST OF FIGURES .......................................................................................................................................... vii

1  Introduction .................................................................................................................................................. 1
   1.1  Background ........................................................................................................................................... 1
   1.2  Approach Selection ............................................................................................................................... 2
   1.3  Image Features and Corresponding Challenges ....................................................................................... 4
      1.3.1  Shape Features ................................................................................................................................ 5
      1.3.2  Texture Features ............................................................................................................................... 7
   1.4  Other Challenges .................................................................................................................................... 9
   1.5  Thesis Organization ............................................................................................................................... 9

2  State of the Art ............................................................................................................................................ 11
   2.1  Shape Features ..................................................................................................................................... 11
      2.1.1  Simple Shape Features ................................................................................................................ 12
      2.1.2  Shape Context ................................................................................................................................ 12
   2.2  Texture Features ................................................................................................................................... 15
      2.2.1  Gabor Filter ................................................................................................................................... 17

3  System Design ............................................................................................................................................. 19
   3.1  The User App ....................................................................................................................................... 20
      3.1.1  User Interface Logic ....................................................................................................................... 22
      3.1.2  Grab Cut Layout ............................................................................................................................. 22
      3.1.3  Connection with the Server .......................................................................................................... 25
   3.2  The Server ........................................................................................................................................... 26
      3.2.1  Data Acquisition ............................................................................................................................ 26
      3.2.2  Preprocessing .................................................................................................................................. 26
      3.2.3  Efficient Database Generation ..................................................................................................... 30
      3.2.4  Database Update ............................................................................................................................ 32

4  Algorithms ................................................................................................................................................ 33
   4.1  Turn Angle ......................................................................................................................................... 33
      4.1.1  Invariance and Robustness ........................................................................................................... 36
      4.1.2  Dynamic Programming and Centroid Distance .......................................................................... 37


LIST OF TABLES

Table 3-1: Cost Matrix .................................................................................................................. 31
Table 5-1: Binary Table that Shows the Relationship of Possible Retrieval Outcomes .......... 48
Table 5-2: Ranked List Used to Calculate Precision and Recall .................................................. 49
Table 5-3: Retrieval Results .......................................................................................................... 55
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Retrieval Results of the Three Image Search Engine</td>
<td>4</td>
</tr>
<tr>
<td>1-2</td>
<td>Shapes of Cup, Bike and Elephant</td>
<td>5</td>
</tr>
<tr>
<td>1-3</td>
<td>Original Image (Left), Binary Image (Middle) and Contour (Right)</td>
<td>6</td>
</tr>
<tr>
<td>2-1</td>
<td>Shape Context Descriptor</td>
<td>13</td>
</tr>
<tr>
<td>3-1</td>
<td>A User-Friendly Online Shopping System</td>
<td>20</td>
</tr>
<tr>
<td>3-2</td>
<td>Structure of the System</td>
<td>20</td>
</tr>
<tr>
<td>3-3</td>
<td>Main Layout of the App</td>
<td>21</td>
</tr>
<tr>
<td>3-4</td>
<td>Graph Built upon an Image</td>
<td>22</td>
</tr>
<tr>
<td>3-5</td>
<td>Graph with Sink and Source Node</td>
<td>23</td>
</tr>
<tr>
<td>3-6</td>
<td>Grab Cut Interface</td>
<td>25</td>
</tr>
<tr>
<td>3-7</td>
<td>Data Reduction Results</td>
<td>29</td>
</tr>
<tr>
<td>3-8</td>
<td>Flow Chart of Database Generation</td>
<td>31</td>
</tr>
<tr>
<td>4-1</td>
<td>Turn Angle</td>
<td>33</td>
</tr>
<tr>
<td>4-2</td>
<td>A Special Case While Calculating Distance</td>
<td>35</td>
</tr>
<tr>
<td>4-3</td>
<td>Finding Out Rotation Angle Between Shapes</td>
<td>37</td>
</tr>
<tr>
<td>4-4</td>
<td>Turn Angle and Centroid Distance</td>
<td>40</td>
</tr>
<tr>
<td>4-5</td>
<td>Edge Detection Kernels</td>
<td>41</td>
</tr>
<tr>
<td>4-6</td>
<td>Edge Detection Kernels</td>
<td>42</td>
</tr>
<tr>
<td>4-7</td>
<td>Masked Image</td>
<td>43</td>
</tr>
<tr>
<td>4-8</td>
<td>EHD Feature Extraction Result</td>
<td>44</td>
</tr>
<tr>
<td>5-1</td>
<td>Four Possible Retrieval Outcomes</td>
<td>48</td>
</tr>
<tr>
<td>5-2</td>
<td>Plotting P-R Curve Using Raw Data</td>
<td>50</td>
</tr>
</tbody>
</table>
Figure 5-3: P-R Curves for Six Classes ................................................................. 52
Figure 5-4: Deformed, Noisy Shapes, and Corresponding Prototype .................. 53
Figure 5-5: Results Retrieved by Shape............................................................... 54
Figure 6-1: Similar Shapes of a Sports Shoe and a Casual Shoe ......................... 58
1 Introduction

This thesis uses shape and texture based image matching methods to build a product retrieval system and evaluates its performance with a simple database of shoe images. This hierarchical image retrieval system comprises three major components: shape and texture matching, automatic database generation, and a user-friendly hand-held device App.

1.1 Background

Nowadays, online shopping is attracting an increasing number of people all over the world due to its conveniences. It provides a 24 hour service. Consumers can shop online any time as long as they have the access to the Internet, whereas shopping at a conventional retail store requires travel and must take place during regular business hours. Online shopping provides online price comparison services for quick search for the best deals. It usually has generous return policies to compensate for the advantages of physically seeing products in the stores. Before using all these convenient services that online shopping provides, the desired product has to be found. Currently, this kind of tasks are commonly performed by keyword search.

After over 20 years’ development, keyword search engines such as Google, Yandex, and Baidu deliver fast and accurate results when the correct keywords are submitted. Many search engines take the Big Data approach to provide customized results based on the user’s activity history to enhance the user’s experience. Although keyword search engine is a mature and powerful tool for information retrieval, it could fail under certain circumstances.

Searching for products by keywords could be impossible when the consumer finds a product of great interest but without detail product information. As an example, keyword searching does not work with a product image which is randomly captured on a street and not well annotated. No useful keywords can be extracted from that image. However, the content of
the image contains a wealth of information about the product. The information can be used for retrieving similar products. However, such a system using the content of an example image to retrieve the desired products is currently not available.

This problem motivated us to build a product retrieval system to enhance the online shopping experience for consumers. Content-based Image Retrieval (CBIR) system is the application of computer vision techniques to perform the image retrieval task.

Before CBIR was invented, people could only search images by metadata such as keywords, tags or descriptions associated with images in the database. The result of metadata search relies entirely on the quality and completeness of annotations. However, a large proportion of images are not well annotated or even missing annotations which makes searching for similar images extremely challenging. Having humans manually annotate images by entering keywords or descriptions in a large database is the only way to solve this problem. It is time consuming and may not capture the desired keywords to describe an image. A CBIR system is able to deal with all these challenges for image-based search.

Nowadays, many technologies have been developed to build CBIR systems. They are divided into two categories: feature based or machine learning technologies [1][2]. To build an efficient product retrieval system using CBIR, the proper technologies need to be selected.

1.2 Approach Selection

Feature based and machine learning technologies have already been deployed in image search engines such as TinyEye, Yandex, and Google Image. To analyze the performance of these two kinds of technologies, TinyEye, Yandex, and Google Image are tested using an image shown in Figure 1-1 (c).
Feature based technologies are used in TinyEye. These technologies recognize objects by extracting image features, generating feature vectors, and measuring the similarity between two images based on their feature vectors. A challenge of using this kind of technologies is selecting the right features which are suitable for the retrieval task. TinyEye selects fingerprint as the image feature to retrieve the same or partially same images and obtains accurate retrieval results as shown in Figure 1-1(a).

Machine learning technologies are used in Yandex and Google Image. These technologies use machine learning algorithms such as Convolutional Neural Network (CNN) [3][4] to perform the retrieval task. CNNs are composed of multiple functional layers. A huge amount of images is needed to train the weights of the connections between each layer. Based on the enormous image databases that Yandex and Google Image have, these two image search engines have trained CNNs which are capable of retrieving similar images accurately. Examples of retrieved images of Yandex and Google Image are shown in Figure 1-1 (b) and Figure 1-1 (d).

Although machine learning technologies are able to retrieve more desirable results, they are not as efficient as feature based technologies in the development of the product retrieval system. The CNN network has to be retrained after the structure of the network changes. At the same time, the product retrieval system has to sync its database with the retailers’ databases which means changes including adding new products or deleting out-of-stock products are made to our database every day. These changes may change the structure of the CNN network and then cause retraining of the network which is time consuming and not efficient. Thus feature based technologies are used to build the product retrieval system in this work. As mentioned before, proper features are needed to retrieve desirable images using feature based technologies.
1.3 Image Features and Corresponding Challenges

Retrieving the same or partly same images is always easier than retrieving visually similar images. Local features like SIFT and SURF have been used to recognize the same or partly same images \[5]\[6]. This kind of features performs well even if images are rotated and translated. However, it is hard to evaluate how similar two images are using these local features. For example, two high heel shoes that are made of different materials will be recognized as different objects because SIFT and SURF focus on local pixel values and different material will present differently on pixel levels. For this reason, higher level features are needed to accurately describe an object. Shape, texture, and color features are commonly used to describe image content.

Color and texture features can retrieve similar images but the contents of those images are not necessarily related. Shape features focus on the shapes of the objects which are segmented from images and can distinguish different categories of objects effectively. Shape,
color and texture are often combined to produce better retrieval results [7][8]. Considering that a product may have different colors for consumers to choose, only shape and texture features are used to retrieve products in our system.

Figure 1-2: Shapes of Cup, Bike and Elephant

1.3.1 Shape Features

Shape is a powerful and intuitive feature for classifying objects. It is defined as the geometric appearance of an object [9]. Most objects have their own distinguishing shapes such as cups, bikes, and elephants as shown in Figure 1-2. Humans can recognize different kinds of shapes easily according to their outlines. Thus shape can be an efficient feature for describing and distinguishing different classes of objects. There are many different kinds of shape features which extract information from the region or contour of an object. Shape features can be divided into two main classes: region-based and contour-based shape features. To extract different kinds of shape features, a shape is represented in two different ways.

Region-based shape features are generated from binary images as shown in Figure 1-2. They consider binary images as distributions and extract information based on image moments. Binary images can be described using a binary function as shown in Equation (1.1).

\[
f(x, y) = \begin{cases} 
1, & \text{if } f(x, y) \in D \\
0, & \text{otherwise},
\end{cases}
\]
where \(x\) and \(y\) are the coordinates of pixels in the image. \(D\) is the region of the object. Statistical features can be easily calculated from this kind of representation.

Contour-based shape features are based on Curvature Scale Space representation of the contour [10]. The way of representing shape used for contour-based feature extraction is shown in Equation (2.2).

\[
p(n) = (x(n), y(n)), \quad n \in [0, N - 1],
\]

where \(N\) represent the number of contour points, \(p(N) = p(0)\). \(x(n)\) and \(y(n)\) are the coordinates of the \(n^{th}\) contour point. This representation makes it easier to extract contour-based shape features. Examples of Curvature Scale Space representations are shown in Figure 1-3.

![Figure 1-3: Original Image (Left), Binary Image (Middle) and Contour (Right)](image)

Besides the different methods of shape representation, region-based shape features and contour-based shape features have different properties [10][11]. Region-based shape features can describe complex shapes which consist of multiple disconnected regions, and simple shapes with or without holes. This property makes region-based shape feature perform well even the shape is separated into several disconnected regions. Region-based shape feature is also robust to
segmentation noise. However, shapes which have similar region-based shape features may still have different contour properties.

Contour-based shape features emulates the human visual system very well and have good shape generalization properties. This kind of shape features are robust to perspective transformation and to significant non-rigid deformations such as the outline of a running person or walking beetle. It is easy to compress the contour-based shape feature vector into a compact vector since the size of the feature vector can be adjusted by the complexity of the contour. Region-based and contour-based shape features can both accomplish certain shape retrieval tasks, however, there are still challenges when applying these two shape features to shape matching.

Challenges such as the change of viewing angle, noises on the contour, the alignment of two shapes and the inconsistent number of contour points, etc. make shape matching a complex task. The change of viewing angle can make the contour of the object become totally different. A simple example would be a cylinder. The side of a cylinder appears as a rectangle while the top and bottom are circles. They are totally different shapes. Thus recognizing a simple cylindrical object imaged at different viewing angle can be difficult. In shape matching, correctly mapping contour points of a shape to contour points of another shape is really important. A noise point, inconsistent number of contour points, and misaligned shapes can easily break this mapping and cause problem.

1.3.2 Texture Features

Texture is another important image characteristic defined as the visual and tactile quality of a surface. It represents some statistical properties of an image [12][13]. There are different ways to analyze texture including structural, statistical, and model-based approach [12].
The structural approach uses texels and placement rules to represent textures. Texels or texture elements are fundamental units of texture space. They compose the texture by repeating over and over again on the surface of an object. Texels can be found using some local images features. The placement rule expresses the spatial relationship between the texels. This kind of algorithm can be applied to recognize some artificial textures which have certain kinds of texels and placement rules. The problem of this algorithm is that it is really challenging to detect texels and the corresponding placement rules of natural materials like grasses and leaves.

Statistical approaches are more efficient dealing with natural materials according to researches which show that statistic features are used by human visual systems for texture discrimination [14][15]. To describe textures, statistics including first-order statistics, second-order statistics, and higher-order statistics are calculated based on selected features. Many invariant texture methods based statistics have been developed such as polarograms, texture edge statistics, harmonic expansion, multiple correlation, and moment invariants. These statistics are mostly computed from grayscale images.

The other way to perform texture recognition is using model-based methods. Researchers model the texture image as a probability model or a linear combination of a set of basis functions, and use coefficients of these models to describe the texture. Different models such as simultaneous autoregressive model (SAR), Markov model, and Wold-like model, etc. are used to perform the texture-based retrieval task. Estimation of the coefficients and choosing the correct model which is suitable for the selected texture are major tasks of model-based methods. These three methods are suitable for recognizing different textures but sharing some common challenges when they are applied to build the product retrieval system.
Texture feature is used to refine the retrieval results given by shape matching in our system. There are many mature texture features which can distinguish different texture accurately [16]. However, objects can be made of multiple materials, and the objects’ surfaces may have different patterns. The texture feature needs to be able to distinguish materials and patterns simultaneously. The other challenge is recognizing textures under different rotation, scaling, and illumination. Thus invariance to rotation, scaling, and illumination of the image is also desired.

1.4 Other Challenges

Generating a well-classified database with thousands of images could be a challenge. A simple approach is generating the database manually. Manually creating a large image database is only feasible for big online companies because of their abundant resources. For this application, an efficient way to generate and maintain the database is necessary in order to keep the database up to date with the products. Details on database generation and maintenance are discussed in Section 3.1.3.

Another challenge of this work is to build a user-friendly system. Users need a convenient tool to capture images, perform object segmentation and display retrieval results. A handheld device app is developed. Details of this app are described in Section 3.2.

1.5 Thesis Organization

The state of the art of CBIR and technologies behind it are reviewed in Chapter 2. The structure of the product retrieval system including two different parts: the server and the user app are introduced in Chapter 3. Chapter 3 also introduces an efficient approach to generating a well-classified database automatically, the mechanism to maintain the database, and the functions of
the user app. The algorithms used for shape and texture matching are introduced in details in Chapter 4. Evaluation and actual retrieval results are presented in Chapter 5. Chapter 6 includes summary, future work, and contributions.
2 State of the Art

Shape and texture features are often used in CBIR systems. This chapter introduces some practical CBIR systems, and the shape and texture feature they used. Some other shape and texture features that we have learned and tested are also discussed in this chapter.

2.1 Shape Features

Different shape features have been applied to practical CBIR systems such as spine X-ray image retrieval system, fish species recognition system, and human detection system [9][17][18].

The spine X-ray image retrieval system is built for retrieving spine X-ray images which focus on the Anterior Osteophyte pathology. Multiple Open Triangle and Corner-Guided Dynamic Programing algorithm are used to provide a high retrieval recall percentage and retrieval speed. After all relevant spine images are retrieved, the precision is still above 85% [9].

Fish species are recognized using different shape features: turn angle distribution, bend-angle with Fourier descriptors, and turn-angle with tangent space searching. Turn angle distribution is the most suitable shape feature for fish recognition. The system reaches an accuracy of 73% with 6 fish species. Bend-angle with Fourier descriptor has a lower accuracy rate. Turn-angle with tangent space searching is accurate but time consuming because a 1-D search is needed to compensate for rotation and shape indexing shift. The human detection system also uses multiple shape features: turn-angle with tangent space searching, bend-angle with Fourier descriptors, and turn functions with power cepstrum. Using tangent space searching has the best retrieval results among all these three matching methods. These image matching methods can also be applied to detect other objects such as animals, vehicles, and shoes.
Tangent space searching with turn-angle can be applied to shoe images retrieval directly but the accuracy is low when matching stretched or shortened shapes. It needs to be modified to achieve higher accuracy. Fourier descriptor with bend-angle is not suitable for retrieving shoe images because bend-angle cannot capture sufficient information from the shoe contours. In order to build an efficient product retrieval system, we have learned and tested other simple shape features and shape context.

2.1.1 Simple Shape Features

There are many simple shape descriptors like eccentricity, circularity, ellipse variance and convexity. They are used to perform basic shape matching tasks [19].

Eccentricity measures shape aspect ratio. It is the ratio of the length of the major axis to the length of the minor axis. Circularity measures how similar a shape is to a circle. It is the ratio of the area of the shape to the area of the circle which has the same perimeter as the shape. Ellipse variance maps error of a shape to fit an ellipse with the same covariance matrix as the shape. Convexity is defined as the ratio of the perimeter of the convex hull to the perimeter of original contour. These features are simple but capable of accomplishing basic tasks of recognizing distinct shapes.

2.1.2 Shape Context

Shape context was developed by Serge Belongie and Jitendra Malik in 2000 and quickly became a popular shape matching algorithm [20][21][22]. The purpose of shape context is measuring shape similarity and recovering of point correspondence. To make shape context efficient, the contour points must be reduced to N points. These N points do not need to be key points or uniformly distributed in space since shape context has strong tolerance for the quality
of the contour points. Vectors originating from a reference contour point to all the other points are then generated. After that, vectors are compressed into a compact descriptor for the reference point by computing a coarse histogram of relative coordinates (relative or absolute angle and logarithm of distance). As shown in Figure 2-2, a circle centered at a contour point is divided into 60 fan-shaped sub-regions. Each sub-region represents a bin of the histogram and the content of the bin is the number of the contour points in the sub-region.

![Figure 2-1: Shape Context Descriptor](image)

Shape context builds a histogram for each point on the contour. These histograms compose a very rich descriptor of the shape. As an efficient shape descriptor, shape context also has to be invariant to translation, scaling and rotation. Shape context is invariant to translation intrinsically because everything is measured relative to contour points. Normalization is required to make shape context invariant to scaling because it uses distances between contour points as part of the descriptor. Normalizing the distances is done by dividing all radial distances by the median distance. A simple way to make shape context invariant to rotation is measuring all angles relative to the shape’s axis of least inertia. Another way to make it invariant to rotation is measuring all angles relative to the line segment between the reference point and the point next
to the reference point. After applying these methodologies, the similarity between two shapes can be measured by shape context.

When performing shape matching, there are two criteria to determine the correspondence of two sets of contour points: the similarity of these two shapes and the uniqueness of the correspondence. The shape distance is comprised of two terms: distance between shape contexts and local appearance. The distance is defined as:

\[
C_s = \frac{1}{2} \sum_{k=1}^{K} \left| \frac{g(k) - h(k)}{g(k) + h(k)} \right|^2,
\]  

(2.1)

where \(g(k)\) and \(h(k)\) are two normalized \(K\)-bin histograms. \(C_s\) is the \(\chi^2\) distance between them. The local appearance term \(C_A\) is defined as:

\[
C_A = \frac{1}{2} || \left( \frac{\cos(\theta_1)}{\sin(\theta_1)} \right) - \left( \frac{\cos(\theta_2)}{\sin(\theta_2)} \right) ||,
\]  

(2.2)

where \(C_A\) is half the length of the chord in unit circle between the unit vectors with angle \(\theta_1\) and \(\theta_2\). The combined matching cost is the weighted sum of \(C_A\) and \(C_s\) as described in Equation (2.3).

\[
C = (1 - \beta) C_s + \beta C_A.
\]  

(2.3)

The similarity between two shapes can be measured by Equation (2.3). The unique correspondence of two set of contour points can then be found by finding a one-to-one match of the two sets of contour points and minimizing their total cost. Minimizing the total cost is a square assignment problem and can be solved by the Hungarian algorithm [23][24]. Because of the time complexity, the time for finding the minimum cost is \(O(N^3)\) which is not very efficient.
Some improvements have been made by Serge Belongie, Jitendra Malik and Jan Puzicha in 2005 [22]. They introduced thin plate spline (TPS) to shape context and redefined the shape distance. After finding out the correspondence between two sets of contour points, they build a TPS model based on the correspondence to describe the transformation between two shapes. TPS interpolation is then used to transform one shape into the other and match the transformed shapes. The transformation and matching are repeated several times before a minimum cost is found. The new distance is defined as the sum of shape context distance, local appearance and bending energy which measures the energy cost of the shape transformation.

A thought that is very inspiring and ingenious presented in [22] is the method of generating the prototype which can be used to represent a whole class of shapes. This method is modified to generate our database, train the prototypes and classify images. Its concept is pretty similar to K-means [25][26]. First, K shapes are randomly chosen as preliminary prototypes. All the other images are then labeled according to these preliminary prototypes. After labeling all the other images, new prototypes are selected by minimizing the average distance of the prototypes to all the other shapes in the class. Repeatedly labeling the images and selecting prototypes until the database reaches a desired accurate rate. The modified method for automatically generating our database is described in detail in Chapter 3.1.3.

2.2 Texture Features

Besides shape, texture is another important and powerful image feature in retrieving certain types of images. Many different texture descriptors including Texture Browsing Descriptor (TBD), Homogeneous Texture Descriptor (HTD), and Edge Histogram Descriptor (EHD) are used in CBIR systems [16].
TBD is a compact descriptor which uses regularity, directionality, and coarseness as texture characteristics. Regularity is used to determine if a periodic pattern with well-defined directionality and coarseness values exists. Directionality is used to describe the dominant directions of the texture. Coarseness is related to image scale or resolution. This descriptor is useful for browsing textures. It can be used to retrieve textures with similar perceptual properties and then use HTD to refine the retrieval results.

HTD is a statistical texture descriptor. It is computed by first filtering the image with a bank of orientation and scale sensitive filters, and then calculating the mean and deviation of the filtered image in frequency domain. This descriptor is robust, effective and easy to compute [27][28].

EHD captures the spatial distribution of edges. To compute EHD, the image is divided into four by four blocks and 16 edge histograms are extracted from these blocks. Each histogram has five bins representing five types of edges. The edge distribution is a good texture signature that is useful for image to image matching even when the underlying texture is not homogeneous.

The texture descriptor chosen for building the product retrieval system has to be precise to achieve good retrieval results. Textures retrieved using TBD have similar perceptual properties. These textures are not necessarily similar or belong to the same category. Thus TBD as a candidate is not eligible. HTD and EHD are capable of retrieving similar textures precisely. They are learned and tested before applied to our system. HTD is tested using Gabor filters.
2.2.1 Gabor Filter

Gabor filter is a group of wavelets, each wavelet captures the energy at a specific frequency and direction. These wavelets can be generated from dilation and rotation of the mother wavelet which is described in Equation (2.4).

\[
\varphi(x, y) = \frac{1}{2\pi \sigma_x \sigma_y} \exp\left[-\frac{1}{2} \left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \exp(j2\pi W x), \tag{2.4}
\]

where \(\sigma_x\) and \(\sigma_y\) are the space constants of the Gaussian envelope along the x-axis and the y-axis respectively. \(W\) is the frequency of the sinusoidal wave. The letters \(m\) and \(n\) represent different scaling and orientation of the wavelets generated from the mother wavelet. Let \(m\) and \(n\) denote the scaling and orientation, \(n = 0, 1, 2… N-1, m = 0, 1, 2… M-1\), the generated wavelets are described in Equation (2.5).

\[
\varphi_{m,n}(x, y) = a^{-m} \frac{1}{2\pi \sigma_{x,m,n} \sigma_{y,m,n}} \exp\left[-\frac{1}{2} \left(\frac{\tilde{x}^2}{\sigma_{x,m,n}^2} + \frac{\tilde{y}^2}{\sigma_{y,m,n}^2}\right)\right] \exp(j2\pi a^m U_l x) \tag{2.5}
\]

\[
\tilde{x} = a^{-m}(xcos\frac{\pi n}{N} + ysin\frac{\pi n}{N}) \tag{2.6}
\]

\[
\tilde{y} = a^{-m}(-xsin\frac{\pi n}{N} + ycos\frac{\pi n}{N}) \tag{2.7}
\]

\[
\delta_{x,m,n} = \frac{(a+1)\sqrt{2\ln2}}{2\pi a^m(a-1)U_l} \tag{2.8}
\]

\[
\sigma_{y,m,n} = \frac{1}{2\pi \tan(\frac{\pi}{2N}) U_l^2 (\frac{1}{2\pi a^m(a-1)U_l})^2} \tag{2.9}
\]

\[
a = \left(\frac{U_l}{U_l}\right)^{\frac{1}{M-1}}. \tag{2.10}
\]

The wavelets are used as kernels applied to the image to extract texture features. Let \(G_{m,n}(x, y)\) denotes the transformed image’s pixel value calculated by the convolution at a
specific scaling (n) and orientation (m). The mean $\mu_{m,n}$ and standard deviation $\sigma_{m,n}$ of $G_{m,n}(x, y)$ can be used to represent the texture.

The features generated by Gabor filters can be modified to be rotation invariant. For each scaling (n), a rotation of the Gabor filter causes a translation of the feature vector along the orientation axis. Thus a search on the orientation axis can be applied to determine the rotation angle and compensate for it.

Gabor filters perform well for matching textures but could not retrieve desired surface patterns during the test. For this reason, EHD is employed and described in detail in Chapter 4.
3 System Design

Our system is designed as an intermediary between consumers and retailers as shown in Figure 3-1. The retrieval system allows a customer to upload a product image. After the image is received, it retrieves the image of the most similar products along with the links which direct the customer to the retailers’ websites. The customer can choose the favorite products from the retrieved product images and follow the links to purchase them. The system also needs to communicate with the retailers’ servers to sync with their image databases daily, since retailers add new products and delete out-of-stock products every day as product lines change. Sale statistics is sent to the retailers to evaluate the product retrieval system.

The product retrieval system is composed of two important parts. Considering the fact that many people shop online using mobile apps and the convenience for the user to capture and submit a query image, we design the system to include a handheld device app and a server. An Android app running on the handheld device interacts with the users and is able to capture the image of a desired product and perform necessary pre-processing of the image. The processed image is then sent to the server via Internet. Once the query images is received, the server starts matching and retrieving the most similar images from the product image database. The final step is sending the retrieved images back to the handheld device and showing them on the screen. The server and Android app parts of the system are introduced in detail in this chapter. Figure 3-2 shows the structure of the system.

The rest of this chapter is used to describe the details of the handheld device app and server including the user interface, the algorithm used to perform object segmentation, and the new efficient method to automatically generate the database, etc.
3.1 The User App

The user app is responsible for capturing and segmenting the image and communicating with the server. The app will activate the Android camera app to use the camera, capture an
image and then save the image in the local disk. It will then perform Grab cut on the saved image, segment the object with human interaction and save the result in the local disk. The segmented and original images will be uploaded to the server for further processing. After the server completes its tasks, the retrieved images will be sent back to the user app and shown on the screen of the handheld device.

Figure 3-3: Main Layout of the App
### 3.1.1 User Interface Logic

The app has four layouts: the main layout, the Android camera layout, the Grab cut layout, and the result layout. As shown in Figure 3-3, the main layout shows the captured image and segmented result. It includes “upload image” buttons. The original and segmented images are uploaded to the server once these buttons are touched. The Android camera layout captures an image and saves it. The Grab cut layout handles the segmentation. Due to the complexity of the background of the captured image, Grab cut may not always generate a perfect segmentation result. User’s input may be required to improve the segmentation result. The result layout displays the retrieved images to the user.

### 3.1.2 Grab Cut Layout

Grab cut layout uses Grab cut to perform the object segmentation task. Grab cut is an advanced version of Graph cut. Graph cut takes each pixel in the image as a vertex in a graph. It completes the graph by connecting each pair of adjacent pixels and assigns them a weight. Figure 3-4 shows a graph that is built upon an image.

![Graph Built upon an Image](image)

**Figure 3-4: Graph Built upon an Image**
The capital letters in the graph indicate the intensity values of the pixels, and numbers are weights between pixels. The weight is determined by the similarity between the pair of pixels. The more similar the two pixels are, the heavier the weight will be assigned to the connection. This means it is harder to divide this pair of pixels than other pairs.

In graph cut, a source (object) node and a sink (background) node are also created based on the object and background area pixels that the user selects. Weights are set based on how strong a pixel is related to the source or the sink node. The more it is related to the node, the heavier weight it will receive. Figure 3-5 shows the finished graph with a source and a sink node. To simplify the graph, only three pixels are shown in Figure 3-5.

![Graph with Sink and Source Node](image)

**Figure 3-5: Graph with Sink and Source Node**

In Figure 3-5, the red lines are the connections to the source while the blue lines are the connections to the sink. A thicker line means a stronger connection. To separate the object and the background, a cut through connections with the minimum effort or lowest cost must be found. In Figure 3-5, the yellow line shows the minimum cut in that graph. After the cut, nodes
A and C are still connected to the source node, since they belong to the object. At the same time, node B is connected to the sink node because the Graph cut classifies this pixel as the background. To make Graph cut an efficient method for segmentation, many algorithms have been developed to determine the minimum cut in polynomial time [29].

Grab cut is slightly different from Graph cut [30]. Instead of choosing specific pixels as foreground and background, Grab cut chooses a region of interest in the object and uses all pixels in the region as the foreground pixels. Based on that, Graph cut is applied to the image once to generate a region which contains the object and very little background. Using this new generated region as the foreground pixels, a more accurate segmentation can be found by applying Graph cut again. After several iterations, the desired segmentation can be found. The region of interest can also be changed using user input in case of the desired result cannot be found automatically.

As shown in Figure 3-6 (b), the region of interest is set by touching the upper left corner and lower right corner of the object on the screen. Once the Region of Interest (ROI) is set, Grab cut can be applied to the image by touching the “do Grab cut” button. If the user is not satisfied with the result as shown in Figure 3-6 (c), the user can touch the “choose fg” or “choose bg” buttons to manually select foreground or background pixels and perform Grab cut again to redefine the segmentation. A cluster of green pixels shown in Figure 3-6 (d) are the pixels that the user manually labels as the background. Foreground pixels are enclosed by red points in the image. The “redo Grab cut” button can reset the ROI in case the user does not set the ROI correctly. After the user is satisfied with the result and touches the “finish” button, the original and segmented images will be saved in the local disk of the handheld device. Figure 3-6 (e) shows the final segmentation result using Grab cut.
3.1.3 Connection with the Server

Once the original and segmented images are saved in the local disk, they can be uploaded to the server for further processing. This part is accomplished using WampServer on a Windows laptop. WampServer is a Windows web development environment includes apache web server.
and PHP. Using WampServer, the images can be encoded into strings and then sent to the server. After the server receives the images, it will perform image matching and retrieve the top similar images and save them. At the same time, the PHP program will be checking the status of the retrieved images. Once all retrieved images become available, it will send those images back to the user app. The user app will then display the retrieval results on the handheld device’s screen.

3.2 The Server

The server is the core of the system. It is responsible for receiving query images sent from the app and retrieving top similar images automatically. Finally, it needs to send those top images back to the app. The server stores the database of all processed and classified images.

3.2.1 Data Acquisition

The images need to contain only one object and have a simple or uniform background so that the algorithm can easily perform preprocessing tasks. Several image databases such as Google images, ImageNet and Shutterstock are available online. They provide a huge variety of images for research. Unfortunately, these images are not suitable for our application because the majority of the images they provide contain multiple objects and with very complex backgrounds. It would be really challenging for our algorithm to perform the preprocessing tasks on those images. Instead, online shopping websites are used as our image sources for our application because the images they provide include single objects and very simple backgrounds.

3.2.2 Preprocessing

Several preprocessing tasks must be performed before moving onto the training, matching and refining phases. The first task is contour extraction. This process must be done automatically or as efficiently as possible to make it user-friendly. Due to this consideration,
Grab cut as described in Section 3.1.2 is selected to perform this task. In some cases, Grab cut will segment out more than one foreground regions. When this happens, all connected components need to be detected and ranked by their size. The region which has the largest area is considered as the object because most other foreground regions are relatively small and are treated as noises. The object contour can be easily extracted and saved in a text file once the connected component has been detected.

The number of points on each contour could be different depending on the size of the object and the resolution of the image. It would be impossible to try to determine corresponding points and matching turn angles if the raw contour data was used directly. Besides that, the number of points can be very big even for a small image, which will slow down the system. For these reasons, the number of contour points must be reduced. One method to reduce the contour points is evaluating the curve by comparing all vertices’ relevance measure iteratively and eliminating one vertex with the lowest relevance measure each time until it reaches a predetermined number [17]. The relevance measure measures how much the vertex contributes to the forming of the whole shape. The relevance measure is calculated as:

\[
K(s_1, s_2) = \frac{|\beta(s_1, s_2) - 180| l(s_1) l(s_2)}{l(s_1) + l(s_2)},
\]

where \(s_1\) and \(s_2\) are two adjacent line segments, \(\beta\) is the bend angle between two line segments, and \(l(s_1), l(s_2)\) are the normalized length from the vertex to the two adjacent vertices.

Another way to perform data reduction is sampling contour points at a fixed interval. The procedure is simpler. Let \(N\) denote the final number of contour points, \(L\) is the total length of the contour. The length of fixed interval should be \(L/N\). Then start from one point, calculate the Euclidean distance from this point to the clockwise adjacent one. If the distance is shorter than
L/N, continue to the next contour point. If the distance of a contour point equals to L/N, the point is then selected. If the distance is longer than L/N, then a new point on this contour needs to be determined to make sure the final number of points will equal to N. The method used to select a new point is to simply find one point on the last line segment which makes the distance from itself to the starting point of the segment equals to L/N. For example, if the fixed interval L/N = 5, the start point \( P_1 (0,0) \) and three following points are \( P_2 (2,0) \), \( P_3 (4,0) \) and \( P_4 (6,0) \). The distance between \( P_1 \) and \( P_3 \) is 4 and is shorter than 5. However, the distance between \( P_1 \) and \( P_4 \) is 6 which is longer than 5. In this case, a new point \( P \) locates on the x-axis and 1 units away from the point \( P_3 \) need to be found. Let \( x_3 \), \( y_3 \) and \( x_4 \), \( y_4 \) denote the coordinates of \( P_3 \) and \( P_4 \), and then the coordinates of the desired point \( P \) can be interpolated as:

\[
x = x_3 + \frac{(x_4-x_3)t}{\sqrt{(x_4-x_3)^2+(y_4-y_3)^2}} \tag{3.2}
\]
\[
y = y_3 + \frac{(y_4-y_3)t}{\sqrt{(x_4-x_3)^2+(y_4-y_3)^2}}. \tag{3.3}
\]

In this case, \( x = 5, y = 0 \).

The relevance measure method can describe shape very efficiently because points sampled by relevance measure are mostly distributed on the line where the line segments made more contribution to the forming of the whole shape. There are some drawbacks about this method. The alignment of two sets of contour points can be more difficult because the reduced points are not evenly distributed. It is also difficult to calculate the distance between two shapes. On the other hand, two sets of contour points sampled with fixed intervals can be easily aligned using tangent space search. It is also easier to calculate the cost using these point sets. The downside of using evenly distributed points is that some fine details and significant shape information may be lost. Fortunately, this problem can be alleviated by increasing the number of sampled points.
The results of these two data reduction methods are shown in Figure 3-7. The contour for the image on the left in Figure 3-7 is the original contour of a shoe from segmentation. Most of contour points are obviously redundant. The middle one shows the reduced contour (50 points in total) using relevance measure method. Most points which are more important to the whole shape are distributed in the blue bounding boxes. The right one shows points sampled with fixed interval. It shows that it also can describe the shape in fine details if we have enough sampled points. The fixed interval method is selected due to the conveniences it brings to the contour points alignment and shape cost calculation.

![Figure 3-7: Data Reduction Results](image)

Other preprocessing tasks are needed to speed up the system and avoid repeated calculation. The shape and texture features are pre-calculated and saved into text files along with the images. Finally, there are five files for each product image: the original image, the mask image and three files that contain contour points, turn angles and edge histogram. The mask image is a binary image generated by Grab cut in which the foreground is white and the background is black. The contour point file contains the coordinates of all contour points. Each row of the file includes the x and y coordinates of one data point. Turn angles and centroid distances are calculated and saved in the clockwise order from the right most point after contour
point reduction. The edge histograms are extracted from an area in the original image corresponding to the white area in the mask image.

The next step is to create a well-classified database. This database includes about 2000 images of shoes that are classified into 11 categories. Although they do not cover all categories of shoes for online shopping, they are sufficient to prove the feasibility of using matching for this application.

3.2.3 Efficient Database Generation

Generating the database manually is a time-consuming process and requires a lot of manual work. We introduce a new way to automatically generate the database which can reduce the amount of manual work. First we manually classify a small number of images collected as a preliminary database. Using the nearest-neighbor algorithm, the preliminary database is built by finding the nearest class in the preliminary database for each sample, calculating the averages of all samples on the class, and using this updated average as a new representation of the classe. In our case, prototypes are trained to present each class using the modified nearest-neighbor algorithm. Since the samples are already classified manually in the preliminary database, the prototype which represents a class can be found directly using a cost matrix that contains all costs between any two samples in the class. The sample with the smallest sum of costs is selected as the prototype to represent the class, since it has the most similar shape to all the other samples in the class. The approach calculating the cost will be introduced in detail in Chapter 4. Table 3.1 shows a sample of the cost matrix.

Once the preliminary database is built, the whole database can be built easily with little manual work. The program can classify other images automatically based on the initial prototypes with slightly low accuracy. Each time after the program perform classification, the
classification mistakes that the program makes will be corrected manually. The prototypes will then be trained again with higher accuracy. After repeating this procedure several times, the system will gain a desired high accuracy. The flow chart is shown in Figure 3-8.

Table 3-1: Cost Matrix

<table>
<thead>
<tr>
<th>cost</th>
<th>Sample1</th>
<th>Sample2</th>
<th>Sample3</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sum</td>
<td>sum</td>
<td>sum</td>
<td>sum</td>
</tr>
<tr>
<td></td>
<td>A+B</td>
<td>A+C</td>
<td>B+C</td>
<td>-</td>
</tr>
<tr>
<td>Sample1</td>
<td>0</td>
<td>A</td>
<td>B</td>
<td>-</td>
</tr>
<tr>
<td>Sample2</td>
<td>A</td>
<td>0</td>
<td>C</td>
<td>-</td>
</tr>
<tr>
<td>Sample3</td>
<td>B</td>
<td>C</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3-8: Flow Chart of Database Generation
3.2.4 Database Update

After the database is generated, maintaining the database is also an important task. The database must be kept up to date because online shopping websites are constantly adding new products and removing out-of-stock products. A self-updating mechanism is developed to sync the database with the rapidly changing product items on the retailers’ websites. When a retailer’s website uploads a new product belongs to a new class that is not included in the database, the system will add a new class to the database. The new uploaded image is then used as the prototype of the new added class. After more images of the same class are uploaded and labeled, the new prototype can be trained. When an old product is no longer selling, the corresponding files will be deleted from the database. Retraining the prototype is also needed after a few products are deleted. If the class becomes empty after the corresponding files are deleted, the whole class will be removed from the database. Occasionally, the database also needs to be inspected manually to keep the database accurate.
4 Algorithms

4.1 Turn Angle

In geometry, the word turn is defined as a unit of plane angle measurement equal to $2\pi$ radians. Several different methods can be used to calculate the counterclockwise or clockwise turn angle of a line. In this thesis, turn angle is calculated using atan2 function. The distribution of turn angle calculated by atan2 function is shown in Figure 4-1 (a).

To calculate the turn angles of a shape, the contour of the object needs to be extracted from the binary image and saved as coordinates of contour points in a text file. Lines can be found by connecting each pair of adjacent contour points. Turn angles is then calculated as the angles between the line segment and the x-axis. Figure 4-1 (b) shows a turn angle of a line segment on an object’s contour calculated by atan2.

![Figure 4-1: Turn Angle](image)

To create the feature vector, turn angles for all contour points are measured and saved into a feature vector. A feature vector will have N turn angles for N contour points. The length of the feature vector can be changed by controlling the number of contour points using the fix-
interval data reduction method discussed in Section 3.2.1. The longer the feature vector is, the more detail of the shape it can describe. However, matching long feature vectors will require more processing time. Sixty points is sufficient to describe the shape of most objects. After the feature vector is created, the shape distance between two shapes is then calculated using Equation (4.1).

\[ C = \sum_{i=1}^{n}(V_1(i) - V_2(i))^2, \]  

(4.1)

where \( V_1 \) and \( V_2 \) are feature vectors of two shapes. It is the \( L_2 \) norm of these two vectors.

However, matching two shapes using Equation (4.1) is not invariant to rotation and the shift of the start point. For this reason, several additional processing steps have to be performed before matching the feature vectors.

A good shape descriptor needs to be translation, scaling and rotation invariant. Turn angle is intrinsically translation invariant because for an arbitrary line, the absolute angle between it and the x-axis will not change no matter how the line translates. Scaling invariance can be achieved by reducing the number of contour points to a number which is set in advance. The feature is then normalized according to the total length of the contour. If the feature vector is plotted as a curve, rotation of a shape makes the curve move vertically and shift of the start point where turn angle is measured will make the curve move horizontally. In other words, a transformation between two shapes needs to be found to align them and minimize the cost. This step can be accomplished by applying a tangent space search on the feature vector. The cost is measured as shown in Equation (4.2).

\[ C = \min_{t \in [0, n-1], \theta \in \mathbb{R}} \sum_{i=1}^{n}(V_1(i + t) - V_2(i) + \theta)^2, \]  

(4.2)
where $t$ is the shift of the starting point and $\theta$ is the rotation angle. In this way, the shift of the starting point and the rotation angle can be found and compensated to make turn angle invariant to rotation and the shift of start point. When matching the feature vectors, a special case as shown in Figure 4-2 needs to be considered.

![Figure 4-2: A Special Case While Calculating Distance](image)

The turn angle of line segment 1 will be approximately 180 degrees while the turn angle of line segment 2 is about -180 degrees. Two line segments are really close to each other and yet their turn angles are quite different. Due to this issue, Equation (4.2) is modified as Equation (4.3).

$$C = \min_{t \in [-3,3]} \sum_{i=1}^{n} (\text{sgn}(V_1(i + t) - V_2(i)) \times \min(\text{abs}(V_1(i + t) - V_2(i)), 360 - \text{abs}(V_1(i + t) - V_2(i))) + \theta)^2,$$  \hspace{1cm} (4.3)

where $\text{sgn}(x)$ is the sign function, $t$ is the shift, $\theta$ is the rotation angle. However, this matching method is time consuming depending on the rotation angle and the number of contour points. To
make the shape matching more efficient, a more efficient method of matching shapes is
introduced in Section 4.1.1.

### 4.1.1 Invariance and Robustness

One-to-one match of contour points between two shapes needs to be found before their
feature vectors can be compared. The one-to-one match of contour points can be found by
aligning the start points of the shapes to be compared. All other contour points will follow
because they are stored in the clock-wise order. One way to align two start points is to select one
start point of one shape and use all contour points from the other shape one at a time as the start
point. The contour point from the other shape that gives the minimum cost is chosen as the start
point. Assuming there are n contour points for each shape, the time complexity of this matching
method is $O(n^2)$, which is not efficient. After the preprocessing step, the front of all shoes is
facing right in the image. Using this property, the right most point in the image is chosen as the
start point. For shapes in the same class, the start point is 6 points off at most. Then, by going
through at most 6 neighboring points, rotation invariance can be achieved and time complexity
can be reduced to $O(n)$.

As shown in Equation (4.3), $t$ can be found using the method above. Each time when
matching two shapes from a selected start point, the rotation angle needs to be calculated by
going through all turn angles of both shapes and calculating the difference between them.
Ideally, when match two identical shapes, the differences will equal to the rotation angle. For
similar shapes, an approximation of the rotation angle can be found by the average of the
differences. Figure 4-3 shows the turn angles for two boots, the brown one is rotated about 30
degrees. As shown in the figure, this rotation causes the plot to shift up by 30 degrees. This can
be compensated by moving down the plot according to the calculated rotation angle to finally make turn angle function for shape matching invariant to rotation.

![Figure 4-3: Finding Out Rotation Angle Between Shapes](image)

4.1.2 Dynamic Programming and Centroid Distance

Turn angle function can be invariant to translation, scaling, and rotation by performing the processing steps introduced in Section 4.1.1. However, translation, scaling, and rotation invariance in not enough for turn angle to match all kinds of shapes. It fails when matching stretched or shortened shapes. Take the shape of a boot as an example. The upper part of a boot may be longer or shorter than the average boot. They are still classified as a boot but the shape distance between these boots increases due to their stretched or shortened upper parts. If the original matching algorithm is used for these cases, then a lot of false negatives will occur during the matching process. Dynamic programming is used to solve this problem. When matching two
shapes, parts of one shape can be stretched or shortened to match the other shape. This problem is transferred into a finding-the-shortest-path problem by filling out two $n$ by $n$ cost matrices, where $n$ is the number of contour points. Let $V_1, V_2$ denote the turn angles of two shapes, CM and TM denotes the cost matrices. Then the CM matrix is represented as

$$CM(i,j)_{i \in [0,n-1], j \in [0,n-1]} = \text{dis}(V_1(i), V_2(j)),$$

where $\text{dis}(x, y)$ is the Euclidean distance between turn angles $x$ and $y$. The CM matrix contains the distances between each pair of turn angles. The TM matrix is generated according to the distances in CM. Each element of TM contains a path of the minimum cost starting from the start points and ending at the current position. The first row and the first column of TM are a copy of CM. Starting from the second row, each element is filled by the sum of the value of the element itself and the minimum value of the three elements which are on the top or left side of it. The equation for this calculation is:

$$TM(i,j)_{i \in [1,n-1], j \in [1,n-1]} = CM(i,j) + \min(CM(i-1,j), CM(i-1,j-1), CM(i,j-1)).$$

TM matrix tracks the minimum path until it reaches the end so the last element of the TM matrix saves the cost of the shortest path of the two turn angle functions.

Dynamic programming can solve the shape stretching and shortening problem. For example, it allows a shape to be recognized as a boot regardless the length of its upper part. However, the false positive rate increases due to the high tolerance of shape stretch brought by dynamic programming. Obviously, turn angle function itself cannot handle this problem. Another feature is needed. Centroid distance is added to the feature set. Centroid is the center of
gravity of a shape. Assuming a shape has a uniform mass distribution its centroid can be calculated as:

\[
    c_x = \frac{\sum_{x \text{ in shape}} x}{A},
\]

\[
    c_y = \frac{\sum_{y \text{ in shape}} y}{A},
\]

(4.6) (4.7)

where \(c_x\) and \(c_y\) are the coordinates of the centroid and \(A\) is the area of the shape. A centroid distance is the distance between the centroid and a contour point. Centroid distance is intrinsically invariant to translation and rotation but is sensitive to scaling change. For this reason, centroid distance must be normalized by the total length of the shape contour before matching.

By adding this restriction to turn angle and dynamic programming, the system is able to recognize similar shapes with fewer false negatives and distinguish very different shapes. Figure 4-4 shows the centroid distances of two different boots. As shown in Figure 4-4 (b), the two boots have similar turn angles (matched using dynamic programming) but should be classified as two different boots. In Figure 4-4 (c), the plotted centroid distances of the two boots are more distinguishable than turn angles. From Point 10 to Point 30, the red line (the boot on the left) is constantly lower than the blue one (the boot on the right) except for the peek that represents the longer upper part. From Points 30 to 49, the centroid distances show the difference between two different styles of the bottoms of the boots.

Retrieving similar shapes using turn angle has achieved desired results. To further distinguish products with similar shapes, Edge Histogram Descriptor is added to the system to match the materials and surface patterns of these products.
4.2 Edge Histogram Descriptor

Edge is considered to be an important feature which is used by human to recognize an object visually. Histogram is a most commonly used method in the field of compositing global feature of an image such as color histogram. EHD is a very efficient and useful feature for texture matching. The histogram has five bins representing five different types of edges: vertical, horizontal, non-directional, and two diagonals. Five kernels are used to detect these five types of edges. Kernels are small matrices as shown in Figure 4-5.
The five types of edges detected by EHD kernels are shown in Figure 4-6. Five new images are generated by applying these five kernels to the original image. Each new generated image represents one type of edge. Pixel values of these new images can be used to determine to which type of edge the corresponding pixel in the query image belong. These pixel values are calculated as shown in Equation (4.8).

\[ R_{x,y} = \sum_{i \in [0,n], j \in [0,n]} K_{i,j} I_{x+i,y+j}, \]  

(4.8)

where \( n \) is the size of the kernel, \( x \) and \( y \) are the coordinates of the pixel. Figure 4-6 (c) shows the horizontal edge kernel applied to a horizontal edge. The absolute value of the convolution is high because the top pixels and bottom pixels have very different pixel values. The absolute value of the convolution is low for the other four types of edge kernels according to Equation (4.8).

A threshold value is needed as a standard to determine the edge type. Any pixel whose absolute value is above the threshold is considered to be an edge pixel. If the current pixel is an edge pixel of a certain type of edge, the corresponding bin of the histogram is then increased by 1. The histogram is then used as a texture descriptor after the image has been traversed.

Edge histogram is a statistical feature. It is invariant to translation intrinsically but not scaling, and rotation invariant. Before matching edge histograms, several process steps are needed to eliminate the noises from the background, and make EHD scaling and rotation invariant.
4.2.1 Invariance and Robustness

EHD is a straightforward and efficient texture descriptor but there are several problems to be solved before it can be deployed. One problem is the effect of the image background. The only information needed for texture feature is the edge histogram of the object region. The image background region will only add noises to the histogram. To remove the background, the mask image generated during the segmentation process is applied to the original image by bitwise ANDing these two images. All pixels in the background are then set to zero. The mask image is a binary image where pixels in the object region have a value of 255, and other pixels are zero. To avoid capturing edges on the boundary of the object, the mask image is eroded using the erosion morphology operation before performing the bitwise AND. Figure 4-7 shows an example of the masked image.

Depending on the image resolution, the raw edge histogram values are higher for a large image size. The histogram must be normalized by dividing the histogram values by the total
number of pixels in the object region. OpenCV function countNonZero can count how many non-zero pixels in a binary image. The total number of pixels in the object region can be calculated by applying this function to the mask image.

![Figure 4-7: Masked Image](image)

Rotation is another problem for texture analysis. The EHD kernels are not rotation invariant which means matching images with rotation directly will not return accurate result. In Section 4.1.1, turn angle has the same problem and it is solved by using the tangent space search, which determines the rotation angle and compensates for it by adding the rotation angle to the feature vector. The impact of rotation on EHD can also be eliminated by rotating the image by the rotation angle. Once the rotation angle is calculated, the two objects can be aligned by rotating one of the images by the rotation angle.

After these process steps are performed, the texture similarity between two images can be calculated as the $L_2$ norm of the two edge histograms. Section 4.2.2 shows visualizations and examples of EHD.
4.2.2 Visualization and Example

To visualize the edge distribution, we use different colors to present different types of edges. Let red represent the vertical edges, blue represent the horizontal edges, green for the two
diagonal edges and purple for non-directional. An example of the edge distribution extracted by EHD is shown in Figure 4-8.

Figure 4-8 (a) shows the original image (the image on the left) and the result image (the image on the right) generated using EHD. Figure 4-8 (b) shows the edge histogram extracted from the original image shown in Figure 4-8 (a). The edge histogram has 5 bins in total. Each bin represents a type of edge and contains the corresponding normalized value.
5 System Evaluation

There are plenty of image retrieval systems and several methods have been developed to evaluate and compare their performances. The most commonly used method is Precision-Recall curve [31][32]. A high precision means the system is able to retrieve more relevant images than irrelevant images. Recall evaluates the ability of retrieving all relevant images. Precision and recall values are calculated according to the four types of retrieval results as described in the next section.

5.1 Possible Retrieval Results

The four possible outcomes from an image retrieval system are true positive, false positive, true negative, and false negative. True positives are images which are retrieved and are truly relevant to the query image. False positives are images that are retrieved but not relevant. True negatives are images that are not retrieved and not relevant, while false negatives are relevant images that are not retrieved. Their relationships are shown in Figure 5-1 and Table 5-1.

False positives are inevitable for real image retrieval systems. The more relevant images are retrieved, the more likely the false positives will be returned. In other words, the increase of recall causes the decrease of precision. Precision will reach its minimum value when recall goes up to one or all relevant images are retrieved. For a good retrieval system, the precision and recall are both expected to be very high. If we use recall as the x-axis, precision as the y-axis and plot the P-R curve, a high performance system should have a higher precision as the recall increases.
Figure 5-1: Four Possible Retrieval Outcomes

Table 5-1: Binary Table that Shows the Relationship of Possible Retrieval Outcomes

<table>
<thead>
<tr>
<th>Relevant</th>
<th>True positive</th>
<th>False negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not relevant</td>
<td>False positive</td>
<td>True negative</td>
</tr>
</tbody>
</table>

Retrieved  Not retrieved

The precision and recall values are calculated according to definitions.

\[
\text{precision} = \frac{\text{true positive}}{\text{retrieved}} \quad (5.1)
\]
\[
\text{recall} = \frac{\text{true positive}}{\text{relevant}}. \quad (5.2)
\]

5.2 Precision and Recall Calculations

The first step to plotting a P-R curve is to prepare reference datasets and test images. The datasets for the experiments have already been established using the method described in Section
3.1.3. A test set of 284 images has been collected by randomly selecting 20 percent of the images in six classes, which have the most images and distinct shapes among all classes. After excluding the test images, prototypes are retrained for these six classes. To calculate the P-R curve, all test images are compared with the prototype of each class using turn angle and the ranked lists of retrievals are generated according to the matching results. Adjusting a threshold on each ranked list gives us different sets of retrieved images. Then the retrieved images are labeled manually as true positives or false positives according to which precision and recall values are measured.

Table 5-2 shows an example of a ranked list:

<table>
<thead>
<tr>
<th>List#</th>
<th>Image#</th>
<th>Shape distance/threshold</th>
<th>relevant</th>
<th>precision</th>
<th>recall</th>
<th>Total image</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>0.34</td>
<td>yes</td>
<td>1</td>
<td>0.02</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>39</td>
<td>0.37</td>
<td>no</td>
<td>0.5</td>
<td>0.02</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>0.39</td>
<td>yes</td>
<td>0.667</td>
<td>0.04</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>56</td>
<td>0.42</td>
<td>yes</td>
<td>0.75</td>
<td>0.06</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>0.43</td>
<td>no</td>
<td>0.6</td>
<td>0.06</td>
<td>50</td>
</tr>
</tbody>
</table>

For example, if the threshold is 0.34, the first image will be retrieved and it is a true positive. In this case, \( \text{precision} = \frac{\text{true positive retrieved}}{\text{retrieved}} = \frac{1}{1} = 1 \) and \( \text{recall} = \frac{\text{true positive relevant}}{\text{relevant}} = \frac{1}{50} = 0.02 \).

The recall will not change but precision will decrease to 0.5 when the threshold is adjusted to 0.37 because two images will be retrieved but only the first one is true positive. The whole set of precision and recall values can then be calculated by repeatedly adjusting the threshold until all...
relevant images are retrieved to reach 100% recall. After all precision and recall values are calculated, the P-R curve can be plotted according to the values.

5.3 Precision-Recall Curve

A P-R curve can be given by plotting the set of precision and recall values directly. However, the curve will have a saw-tooth shape like shown in Figure 5-2:

Assuming the nth retrieved image is a false positive, recall remains the same because the number of true positive does not change while precision decreases because the number of retrieved images increases. Then the curve drops vertically or straight down because one recall has two precision values. On the other hand, if the nth retrieved image is a true positive, the curve jags up to the right because recall and precision both rise. This is why the P-R curve has a saw-tooth shape if raw precision and recall values are used. It is always useful to eliminate the
saw-tooth shape of P-R curve. The standard way to do this is to use interpolation to assign a precision value for each recall level as demonstrated in Equation (5.3).

\[ p_{interp} = \max_{r' \geq r} p(r'), \quad \text{(5.3)} \]

where \( p_{interp} \) is the interpolated precision. It is assigned with the max precision value found for any recall \( r' \geq r \).

The P-R curve can also be very informative due to the large number of images. Down sampling can make the curve easier to read. A way to do this is breaking down the recall region into 10 levels of 0.0, 0.1, 0.2… 1.0. Precision values are interpolated for each recall levels. Let \( r_i \) denote the \( i^{th} \) recall level. Then the interpolated precision value is:

\[ p_{interp} = \max_{r_i \leq r \leq r_{i+1}} p(r). \quad \text{(5.4)} \]

Finer recall resolution makes the plotted curve clearer and more readable. P-R curves plotted using this interpolation method and retrieval results are shown in the next section to evaluate the system.

5.4 **Evaluation Results**

The test was done on a set of 284 images from six classes as described in Section 5.2. The P-R curves of these six classes are down sampled and plotted as shown in Figure 5-3. The curves show that the system performed well retrieving relevant images. The precision values of these classes remain above 80% even when their recall values reaches 1.0. Class two and class 10 have the lowest precision values because some shapes in these two classes are deformed and include noises as shown in Figure 5-4. Figure 5-4 (a) shows deformed shapes and Figure 5-4 (b) shows
shapes with noisy contours. Shapes which are deformed or noisy can increase the shape distance and cause the precision to drop.

Figure 5-3: P-R Curves for Six Classes
The hierarchical matching method uses shape to classify the query image. After the query image is classified, top 20 images with similar shape are retrieved from the class by shape. Since different products may have similar shapes, these 20 images may contain different types of products. For example, dress shoe and sport shoe have similar shapes as shown in Figure 5-5 (a). The top 20 images with shape similar to the sport shoe are shown in Figure 5-5 (b).

The top left image has the highest rank and the lower right image has the lowest rank. As shown, a lot of dress shoe images are retrieved and ranked higher than sport shoes. The retrieval results using shape need to be distinguished and re-ranked using EHD texture descriptor. The re-ranked list of top 20 images is shown in Figure 5-5 (c). Sport shoe images moved up to the top of the list while dress shoe images moved down to the bottom. The top five images circled in Figure 5-5 (c) are then selected as the final retrieval results.

Shape is proved to be an efficient feature to classify products like shoes. However, shape alone cannot further distinguish products with similar shapes. To distinguish products with similar shapes, EHD texture descriptor and a hierarchical matching method are used.
Evaluating texture matching performance using shoe images is a complex task. It is hard to define texture classes for shoes because materials and surface patterns are both considered. Since humans are the ultimate judges of texture similarity, six images were chosen from 3 different classes as the test images. The top 5 most similar images are retrieved for each image and then evaluated by human. The retrieval results are shown in Table 5-3.
### Table 5-3: Retrieval Results

<table>
<thead>
<tr>
<th>Query image</th>
<th>Retrieval results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td><img src="image1.png" alt="Query Image" /> <img src="image2.png" alt="Retrieval Image" /> <img src="image3.png" alt="Retrieval Image" /> <img src="image4.png" alt="Retrieval Image" /> <img src="image5.png" alt="Retrieval Image" /></td>
</tr>
<tr>
<td>(b)</td>
<td><img src="image1.png" alt="Query Image" /> <img src="image2.png" alt="Retrieval Image" /> <img src="image3.png" alt="Retrieval Image" /> <img src="image4.png" alt="Retrieval Image" /> <img src="image5.png" alt="Retrieval Image" /></td>
</tr>
<tr>
<td>(c)</td>
<td><img src="image1.png" alt="Query Image" /> <img src="image2.png" alt="Retrieval Image" /> <img src="image3.png" alt="Retrieval Image" /> <img src="image4.png" alt="Retrieval Image" /> <img src="image5.png" alt="Retrieval Image" /></td>
</tr>
<tr>
<td>(d)</td>
<td><img src="image1.png" alt="Query Image" /> <img src="image2.png" alt="Retrieval Image" /> <img src="image3.png" alt="Retrieval Image" /> <img src="image4.png" alt="Retrieval Image" /> <img src="image5.png" alt="Retrieval Image" /></td>
</tr>
<tr>
<td>(e)</td>
<td><img src="image1.png" alt="Query Image" /> <img src="image2.png" alt="Retrieval Image" /> <img src="image3.png" alt="Retrieval Image" /> <img src="image4.png" alt="Retrieval Image" /> <img src="image5.png" alt="Retrieval Image" /></td>
</tr>
<tr>
<td>(f)</td>
<td><img src="image1.png" alt="Query Image" /> <img src="image2.png" alt="Retrieval Image" /> <img src="image3.png" alt="Retrieval Image" /> <img src="image4.png" alt="Retrieval Image" /> <img src="image5.png" alt="Retrieval Image" /></td>
</tr>
</tbody>
</table>
As shown in Table 5-3 (a), (c), (e), and (f), EHD texture descriptor is capable of distinguishing different shoes with similar shapes and retrieving visually similar images. Table 5-3 (b) and (d) shows some undesirable results. As shown in Table 5-3 (b), an image of a boot with shoelace is used as the query. The system retrieves several boots without shoelace but with more complex surface patterns. This is because EHD used in the system only extracts the global edge histogram and uses it as texture feature vector while local edge distribution is ignored. As shown in Table 5-3 (d), a rain boot image is used as the query image and leather boots are retrieved. Rain boots are generally made of rubber instead of leather. However, rain boots may have different surface patterns which cause the system recognize rain boots as leather boots.

The system overall achieved desirable results. Several future works such as expanding the database, and adding local texture features could be done to improve the retrieval results. Chapter 6 will summarize the contribution of this work, and discuss future works in detail.
6 Conclusions

6.1 Summary

CBIR has been employed in many areas due to the rapidly developing computer vision technologies. There is not a system designed specifically for product retrieval even with market prospects. Image is a very popular form of media which is spreading and being shared via the Internet every day. Images founded on the Internet may be labeled with wrong or nonsense keywords which makes it difficult for consumers to find desired products on online shopping websites. An efficient product retrieval system would greatly improve online shopping experience. In this paper, the retrieval system is for shoes but it can be easily modified to suit other products.

Shape and texture features are often used in CBIR systems. Shape is an efficient feature to distinguish different type or style of shoes. A modified turn angle shape descriptor has been proposed. Turn angle is a contour based shape feature. Several preprocessing steps need to be done to make it less time consuming and invariant to rotation. Before extracting the shape feature vector, a data reduction algorithm is applied to the raw contour points to shorten the feature vector. A tangent space search is used to determine the rotation angle. To match stretched shapes, a dynamic programming algorithm is used and centroid distance is added as a restriction. P-R curves are plotted for six different classes of shoes which show that the new shape descriptor is capable of retrieving relevant images with high accuracy.

Shape alone is not sufficient to retrieve desired products. It cannot distinguish different kind of shoes with similar shapes like sport shoes and dress shoes as shown in Figure 6-1. To perform a refined search, EHD, a statistical texture descriptor, is employed and a hierarchical matching method combined shape and texture is proposed. EHD is not invariant to scale and
rotation so compensations must be made before matching. Shape and texture are used on different level during the hierarchical matching. Shape is used to classify the query image and retrieve 20 most similar images in the class. Texture is then used to distinguish similar shaped shoes and retrieve the top 5 most similar images among those 20 images. Some real results are shown in Section 5.4 and prove that shoes made of different materials but have similar shape can be distinguished by texture.

Figure 6-1: Similar Shapes of a Sports Shoe and a Casual Shoe

The quality of the retrieval results relies heavily the quality of the database. Manually building a database with thousands of images is a time consuming and expensive process. Instead of manually building the database, a new method using the concept of K-Nearest Neighbors and Prototype Selection in [22] is used. The only manual work is building a preliminary database and correcting the machine classification mistakes. Several functions are added to make changes to the database so it can be synced with rapidly changing retailer’s
database. Each time it syncs with retailer’s database, the ground truth is built using Prototype Selection.

To make the system accessible, a user app is developed to capture and process images. It invokes the Android camera app to capture an image and save it on the local disk. Grab cut is then used to segment the object. Human interactions can be involved to improve the segmentation result. Once the segmentation is completed, the app uploads the original and segmented images to the server for further processing and waits for the final retrieval results. The server sends the retrieval results back to the user app so that the retrieved images can be shown on the screen.

With all the features mentioned above, the system still has room for improvement. Several improvements can be carried out to make the retrieval result more desirable.

6.2 Future Work

Currently, the system is only evaluated by six classes of images collected from the Internet. The test can only demonstrate the overall system performance. More complex test sets are needed to give an objective evaluation of the performance. Research into new evaluation methods is also needed to perform more thorough evaluations.

Some improvements are needed to improve shape and texture matching result. As shown in Figure 5-3, precision drops when more images are retrieved. One reason for the precision drop is that some contours are fuzzy and include noise which increase the shape distance. There are existing algorithms that can eliminate signal noises. Finding and modifying an algorithm to eliminate contour noises should improve the shape matching result. Currently, the ratio of turn angle and centroid distance is one to one. Every time the ratio changes, the matching result
changes. Finding out the optimal weights for turn angle and centroid distance by experiments could also improve the accuracy.

EHD gives us decent results distinguishing different materials. In this work, only the global edge histogram is used. To achieve a better result, local and semi-global edge histograms could be used. An image is divided into 4 by 4 sub-images. Each sub-image has a local edge histogram. For semi-global edge histogram, four connected sub-images are clustered and there are 13 different clusters as described in the reference [33].

The development of the system is still in a preliminary stage. It retrieves similar images without other information like the name, the brand or the web link. The database has only shoe images and limited number of classes. Additional information and types of products could be added to the system to make it a more powerful online shopping tool.

6.3 Contributions

To the best of our knowledge, there is no product retrieval system for enhancing online shopping system. The first contribution of this thesis is building an intermediary system using image classification. The system includes a server and a user app. The server efficiently retrieves the top most similar images to the query image and syncs the database daily to keep it up to date. The user app provides the consumers an easy way to use the system. It performs the object segmentation task and upload the original and segmented images to the server. The user interfaces are user-friendly.

The second contribution of this thesis is inventing a new method to generate the database automatically. Building a database is very time consuming and a labor intensive task. The new method uses the concept of K-means and Prototype Selection [21]. It reduces the amount of labor and time needed to generate the database by automatically labeling images and training
prototypes based on a preliminary database. Manual work is still needed to correct machine classification mistakes but is significantly reduced.

The third contribution is improving turn angle by using dynamic programming and adding centroid distance as a restriction. Turn angle does not recognize stretched or shortened shapes well. Using dynamic programming enables turn angle to recognize stretched and shortened shapes. Centroid distance is added as a restriction to help turn angle distinguish different shapes (stretched or shortened). Figure 5-3 shows that the improved turn angle performs better than the original version.

The fourth contribution is developing and implementing a hierarchical matching system which uses shape and texture features to retrieve the most similar images. Shape is used to classify the image and retrieve the top 20 images in the class. Some images which belong to other classes may be retrieved because of the similar shape. Texture is then used to distinguish those images and retrieve the top five images among those 20 images. Table 5-3 shows that texture is able to distinguish different kinds of objects even though they have similar shapes.
REFERENCES


[30] C. Rother, V. Kolmogorov, and A. Blake, “GrabCut — Interactive Foreground Extraction using Iterated Graph Cuts.”

