SHAPE RECOGNITION BY USING SCALE INVARIANT FEATURE TRANSFORM FOR CONTOUR

MATHARA ROJANAMONTIEN

A THESIS REPORT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTING IN ENGINEERING SYSTEMS
INTERNATIONAL COLLEGE
KING MONGKUTS INSTITUTE OF TECHNOLOGY LADKRABANG
2017

THESIS TITLE	Shape Recognition by using Scale Invariant Feature Transform for Contour
STUDENT NAME	Mathara Rojanamontien
STUDENT ID	59610040
DEGREE	Master of Engineering
PROGRAME	Computing in Engineering Systems International Program
ADVISOR	Dr. Ukrit Watchareeruetai

Abstract

Contour-based shape feature extraction methods have been researched throughout the years. The reason is that this kind of extraction method can reduce time and computation power needed by computer vision systems by dropping unnecessary information.

Although there are many contour-based shape feature that were proposed before, most of them focus either on global or local features separately, and neglect the another; therefore, missing important information. Also, some of the algorithms that focus on local features may require specific search locations or areas that have to be specified by users or other systems. As a result, additional resources are required.

This thesis proposes a novel shape feature extractor named Contour-SIFT along with a matching method that computes the similarity between two SIFT contour descriptors. It allows a shape to be recognized based on automatically located outstanding local features on its contour, which are extracted globally. The extraction is separated into two parts: feature detection and feature description. Local features are detected from 1-D signal representations of different scales' contour, which are smoothed by using Gaussian kernels. These local feature are described as a list of frequencies from curvature histogram, which is created from curve segment around each detected position. Using the proposed matching algorithm, the descriptors will give high similarity compared

with a model descriptors of a similar shape. Contour-SIFT descriptor also has properties of image scaling-, translation-, and rotation-invariants. Experiment were conducted with different image datasets for verification.

Acknowledgments

With a handful of supports and concerns, the Contour-SIFT algorithm was successfully published in JCSSE 2017. For this reason, I would like to give words of appreciation.

First of all, a great gratitude have to be given to the advisor, Dr. Ukrit Watchareeruetai. He has been extremely supportive through this thesis. Thanks for being so thoughtful, for taking the initiative, and getting it done. Without his books, materials, and a helping hand in implementation, it would be a lot harder to complete the thesis.

Next, to Dr.Chaiwat Nuthong, Dr.Churirat Boonkhun, Miss.Nicha Piemkaroonkong, Mr.Purit Thong-On, people in the CV laboratory, and everyone in CIES classroom. Thanks for sharing your vision, knowledge, and experience. Your comments and suggestions have been helpful.

There is also kindness of International College staffs. Thanks for providing a suitable working space, for the reminding of important notices, and for lending us essential items I had requested for.

Finally, a lot of gratitude have to be given to my families for co-operation, motivations, suggestions, and encouragement, which help me in completion of this thesis. There no limits to their generosity and understanding.

Mathara Rojanamontien

Contents

List of figures

List of tables

Chapter 1

Introduction

In this chapter, the motivation and problem description are described. The chapter also gives accounts about similar issues solved by other researchers and what they had accomplished. After that, it puts forward the proposed algorithm, briefly explains an object recognition system, states the thesis's objectives and scope, then explains the thesis's structure before ending.

1.1 Motivation and problem description

It is important for living creatures to recognize objects. Without the recognizing ability, it is almost impossible to hunt for food, search for safe places, or finding their family members. The task is easy for creatures with visual system because they are able to perceive the difference in appearance between things. For example, a person can recognize a cat on a sofa, a coffee mug next to his laptop, or a sunflower in his garden with ease. Using his eyes and brain, he completes the recognition process that includes perceiving scenes and analyzing information within a fraction of time. Since it is easy to recognize objects by observation, visual processing becomes a primary concerned activity.

A lot of algorithms that mimic human visual processing system have been proposed so far. Such algorithms can be classified as *computer vision based algorithms* as they analyze image data or a sequence of images to extract relevant information.

With technology development, computer vision algorithms have been replaced human vision in many processes. Machines with object recognition ability are introduced to solve various problems such as segmentation, obstacle avoidance, object tracking, or object grasping. As machines are able to work steadily without disposition and tiredness, the replacement reduces human error in work flows and gives higher efficiency for systems. For examples, a security system gains an ability to accurately identify a thief from several surveillance videos and a medical system is able to identify critical patients from unspecified number of x-ray images with a very small failure.

Resemble to human beings, these machines are made with sensors (i.e., camera, etc.) and computation units. They capture images and compare them to predefined images or models in database. The first half can be done without effort by the sensors, but the second half involves computer vision algorithms which have to be chosen carefully. Different algorithms has different complexities and needs different amount of resources. Some algorithms may need longer time but less memory spaces and computation power while others need shorter time but require large storage. Hence, various computer vision algorithms have been introduce as choices for applications.

It is important to note that nowadays digital images hold a lot of information and are stored in large quantity on computers. As a result, the less power, memory, and time consumption, the more preferable the algorithms are. Recall that in order to recognize an object, relevant information has to be extracted and analyzed. In fact, these target information are image features that may or may not contain compulsory data; therefore, the resources should be reduced by extracting and analyzing only necessary features from images.

The examples of features that can be used to recognize objects are intensity levels,

number of pixels, edges, corners, shape and size of objects, or any other high level features. Among them, shape is one of the most important feature when the goal is recognition since it is a very noticeable characteristic of most visible objects. In addition to that, if a proper representation is selected, then it becomes invariant to image translation, rotation, and scaling. In fact, it is commonly known that an image recognition process can be done by observing and comparing the similarities and the differences of shapes. For examples, upon noticing an object, it can firstly be assigned to a category based on its general shape, then after a carefully observation on local parts, its specific information becomes visible. For instance, if a man finds a cat sitting on his laptop, then he first knows that it is a cat, before recognizing its specie, or that it is currently messing his researches up. Therefore, shape has been used by many image processing and computer vision algorithms as a key feature for recognizing and organizing information.

A shape describing method will be classified as *region-based* shape descriptor if it describes shape based on interior points. On the other hand, if the describing method uses boundary points to describe shapes, then it is an *contour-based* shape descriptor. The reason that the name is contour-based, not boundary-based, is because boundary points are commonly sampled stored as a contour. Simply, a contour of a shape is an ordered sequence of Cartesian coordinate 2-D points that are uniformly distributed on its boundary with identical first and last elements. Various contour-based shape recognition algorithms have been extracted features from contours for shape recognition instead of the original shape images to reduce the information needed by the systems.

Two well known contour descriptors are Fourier descriptor (FD) [28] and curvature scale space descriptor (CSSD) [1]. Their performances for image retrieval were compared by Zhang and Lu in 2003 [29]. The authors shown that FD outperforms CSSD in robustness, computation power, and image retrieval. Tan et al. presented a generalized centroid-radii model that can be used to model all forms of shapes along with a system that uses nested R-trees to query similar shapes from a database with the goal of reduc-

ing retrieval time[25]. Later, a method to apply Fourier transform on circle views shape signature of contour for recognizing 2-D object is proposed by Huda and Hussein with a retrieval rate of 83.71% on MPEG-7 database in 2016 [10].

Although there are a lot of contour based shape recognition algorithms proposed so far, most of them focus on global characteristics of shapes. Even though global characteristics are important, if local characteristics are neglected, then important information may not be used. As a clear example, it will be harder when identifying two people while their faces, which contain local characteristics, are covered. On the other hand, focusing on specific parts of a shape without any consideration on global information also has its own disadvantage of missing global viewpoints, which leads to an increment in misidentification. Using the same example of identifying two people, if they happen to have similar faces and are recognized based on the faces alone, then the likelihood of misidentification will increase. For this reason, it is important to pay attention to both global and local shape features.

A combination of apical and basal local features and general global characteristics for plant identification is proposed in 2016 [19]. Nevertheless, the method requires additional input information as target positions and local features are extracted despite the fact that most leaves' species in its target database (Flavia dataset [27]) have similar shapes of apexes and bases. Simultaneously, these leaves may have other unique local parts which can be used to further classify them. Hence, they should be identified based on their outstanding local features which are not fixed manually beforehand. Going back to the same problem, if one man is very muscular, and another man has very long legs, then the first man should be identify based on his muscle, and the second one should be identified based on his legs. To conclude, it is more reasonable to capture the local shape characteristics globally from contours when identifying objects.

Up to this point, contour-based shape recognition that focuses on global local shape features can be expressed in three tasks which are 1) detecting outstanding local features

over the whole shape contour, 2) describing the extracted local features independently, and 3) identifying the shape based on the descriptors.

This thesis proposes a computer vision based-object recognition algorithm named Contour-SIFT, which is a scale-, translation-, and rotation-invariant shape recognition technique motivated by SIFT [15], and a matching algorithm that computes a similarity value between two Contour-SIFT descriptors. As shown in Fig. 1.1, the proposed method is separated into three main parts, each of which completes a task of shape recognition.

1.2 Objectives and scope

After the implementation of the algorithm and the system proposed in this thesis, the following objectives should be completed.

- To develop a computer vision based algorithm for shape recognition
- To develop a computer program that uses the algorithm to recognize objects based on their shapes.
- The proposed algorithm should be invariant to image scaling, rotation, and translation.

Equally important, to reduce the size of the shape recognition problem, this thesis is scoped by the following constrains:

- Each input image must contain only one object
- Each input image must have white background

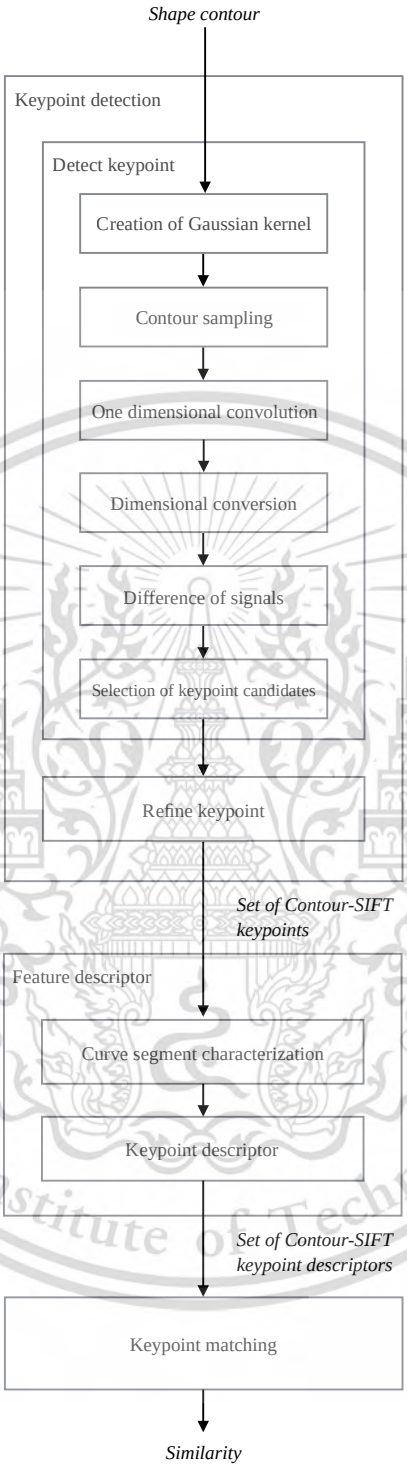


Figure 1.1: The proposed Contour-SIFT algorithm

1.3 Thesis structure

After the introduction, background knowledge and literatures are provided in Chapter 2, methodology of the proposed Contour-SIFT algorithm and Contour-SIFT descriptor matching method is explained in Chapter 3. An object recognition system using the algorithm and some experiments are explained and discussed in Chapter 4. Then, the thesis is concluded in the last chapter, Chapter 5.

Chapter 2

Background knowledge and literature reviews

This chapter provides background knowledge that would assist readers through the rest of the thesis, as well as reviewing existing literatures.

2.1 Background knowledge

2.1.1 Image features

Descriptions or features represent characteristic of a structure [23]. Image features usually represent features of object in an image. Intensity level, weight, height, and area are examples of the simplest image features. Besides, features with higher complexity could be circularity, convexity, curvature, etc. Despite that, not all descriptions are helpful in every applications. Only the features that contribute a better result should be used. Good image features are expected to be invariant of miscellaneous factors (e.g., image scale, object's position, occlusion, and illumination). Usually, the better features, the more complex they are. Also, complex structures are often described by high level

image features.

A feature is produced from a feature function. Particularity, an image feature function is a part of feature extraction which receives an input image and extracts a specific characteristic of the interested part in the image as an image feature. In truth, several features can be combined into a list, called feature vector, which becomes the inputs of classification.

2.1.2 Classification problem

Classification or identification process classifies objects in an image by comparing the feature vectors of training data with that of a testing data [3]. It is one of the most popular processes implemented in computer vision systems. For example, in a fruits collecting system, classification program might be used to classify parts of fruit trees so that a robot can collect the fruits correctly ([[14]][[12]][[22]]). Even basic surveillance systems might used classification to distinguish different types of object (e.g. people vs. obstacles vs. cars. vs. bicycles).

Commonly, a classification problem can be divided into three parts which are *detection*, *extraction*, and *matching*. First of all, detection is done to obtain the location of interested objects. For examples, corner detection, blobs detection, or leaf apex and base detection. Secondly, extraction is done to generate descriptors which hold specific information of the detected objects. For instances, the corners can be described using Harris [9] or SIFT corner descriptor [15] and the leaf's apex and base could be described as a list of angles measured at different positions [19]. Lastly, matching is to match the description of an interest object with one in the existing dataset in order to classify the object's type or category. This process is usually done by a classifier. Significantly, it is where the classification is actually done. All in all, after the three parts of classification problem, an object is identified.

2.1.3 Scale–invariant feature transform (SIFT)

Scale–invariant feature transform or SIFT, proposed by Lowe in 2004 [15], is a keypoint extraction algorithm. It extracts keypoints from images and describes. The algorithm is well-known for its robustness and invariant propoerites. The extracted feature is invariant to scaling and rotation, as well as partly invariant to illumination and viewpoint changes [3]. SIFT can be separated into four main steps including *scale space extrema detection*, *accurate keypoint location*, *keypoint orientation assignment*, and *keypoint descriptors*. The first two steps are mainly done as the detection part while the latter two steps are done as the describing part of classification process. These sections are explained below.

2.1.3.1 Scale space extrema detection

Scale space extrema detection is a step that aims to detect all possible keypoints in an input image. Certainly, to create features that are invariant to scaling, the featured has to be stable across different image scales. In order to achieve this, a continuous scale space function is involved. The scale space of an image $I(x,y)$ is defined as $L_i(x,y,k,\sigma)$ calculated as follows:

$$L_i(x,y,k,\sigma) = G(x,y,k^i\sigma) * I(x,y), \tag{2.1}$$

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \tag{2.2}$$

where $*$ denotes the convolution operation, i is a non-negative integer starts from zero, and σ is the standard deviation of Gaussian function. The convolution process is done for a number of different octaves where σ is double in one after another. To achieve a true scale invariance, a normalization of *Laplacian of Gaussian* (LoG) is needed. LoG with σ^2 can be approximated and replaced by *difference of Gaussian* (DOG) instead

(see Fig. 2.1).

$$D_i(x,y,k,\sigma) = L_{i+1}(x,y,k,\sigma) - L_i(x,y,k,\sigma) \tag{2.3}$$

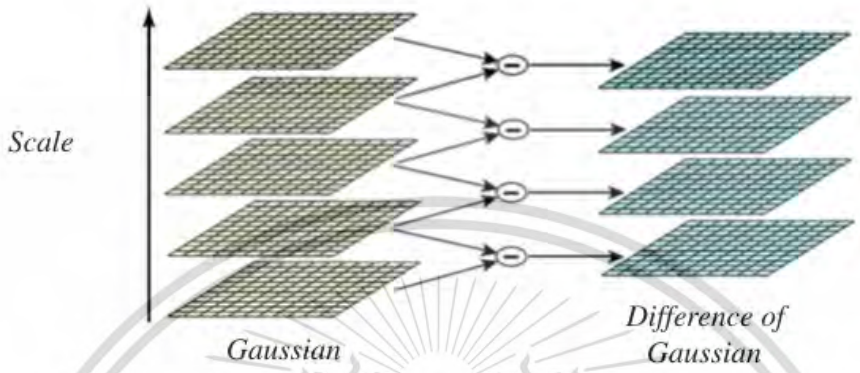


Figure 2.1: Difference of Gaussian at different scales (source [15])

A pixel is considered as a potential keypoint if and only if it is the extrema comparing to its eight neighbors of the same scale, nine neighbors of the previous scale, and nine neighbors of the next scale (see Fig. 2.2). For each scale, blobs of various sizes

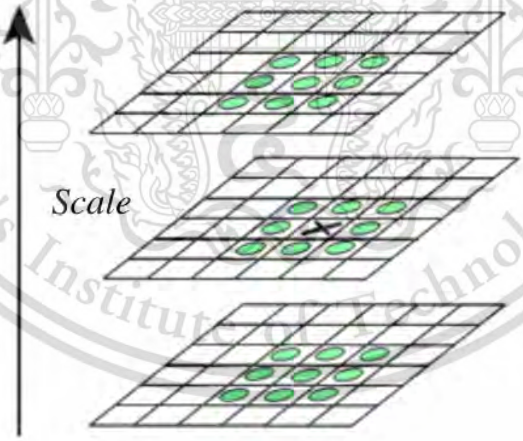


Figure 2.2: $3 \times 3 \times 3$ neighborhood of a pixel (source [15])

are possible to be located by the variation of the σ value. Actually, a Gaussian function with a low value of σ covers smaller local areas and detects small corners while

the function with a higher value of σ covers more and detects larger corners. Hence, extrema (maxmima or minima) points can be detected at different scales with (x, y, σ) as its component. To be precise, a keypoint $p(x, y, \sigma)$ is located at (x, y) position of L at a scale σ. At the end of the process, all potential keypoints become keypoint candidates and are sent to the next section.

2.1.3.2 Accurate keypoint location

After obtaining all keypoint candidates, accurate keypoint location is performed to refine the set of keypoints. Here, actual keypoints are enhanced and the ineffective keypoints that are not sufficiently robust are eliminated. These ineffective points including points with too low contrast (i.e., the intensity value in DoG is lower than a threshold) and points that are on an edge rather than at a distinguishable feature.

First, the local contrast in the area around each keypoint is considered. In this part, a curvature is calculated because it is proportional to the contrast. A small value of curvature indicates a low contrast. If the keypoint's curvature is lower than a threshold, then it is dropped out. Following, the next test checks the localization of the keypoint. If the keypoint is poorly defined, its will have a large principal curvature across the edge but small in the perpendicular direction. The principal curvature can be computed using Hessian matrix, H, at (x, y) and σ of the keypoint as follows:

$$H = \begin{bmatrix} A & B \\ B & C \end{bmatrix} = \begin{bmatrix} \sum_{(a,b)\in W} \frac{\partial^2 D_i(a,b,k,\sigma)}{\partial a^2} & \sum_{(a,b)\in W} \frac{\partial D_i(a,b,k,\sigma)}{\partial a} \frac{\partial D_i(a,b,k,\sigma)}{\partial b} \\ \sum_{(a,b)\in W} \frac{\partial D_i(a,b,k,\sigma)}{\partial a} \frac{\partial D_i(a,b,k,\sigma)}{\partial b} & \sum_{(a,b)\in W} \frac{\partial^2 D_i(a,b,k,\sigma)}{\partial b^2} \end{bmatrix}, \tag{2.4}$$

where W is a window's size applied locally and (a, b) points to elements in the window.

The fact that the differences of neighboring points can be used to estimate derivatives and the fact that eigenvalues of H are proportional to the principle curvatures of D [3] allows the discard of complex eigenvalues computing. Let λ_1 denotes the greater

eigenvalue and λ_2 denotes the smaller one. If $\lambda_1 = r\lambda_2$, then $r = \lambda_1/\lambda_2$. Obviously, the ratio r could be considered instead of the eigenvalues. Moreover, the sum of the eigenvalues and their product can be calculated as a trance of H and a determinant, respectively.

$$trace(H) = \lambda_1 + \lambda_2 = A + C \tag{2.5}$$

$$det(H) = \lambda_1 \lambda_2 = AC + B^2 \tag{2.6}$$

In fact, a division of the trace squared and the determinant can be calculated as shown in Eq. (2.7).

$$\frac{trace(H)^2}{det(H)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r\lambda_2 + \lambda_2)^2}{r\lambda_2^2} = \frac{(r+1)^2}{r} \tag{2.7}$$

The trace squared and the determinant ratio becomes smallest when both λ_1 and λ_2 are equal and increases when r increases. Eventually, from the set of keypoints that survived the contrast elimination, the ones that the ratio $(r+1)^2/r$ is too small are eliminated.

At the end, the points with too low contrast and the points belongs to smooth edges are removed from the set of keypoint candidates before the starting of the next section.

2.1.3.3 Keypoint orientation assignment

The general purpose of keypoint orientation assignment is to remove effects of rotation and scale from the refined keypoints by assigning orientations to keypoints. For each keypoint $p(x, y, \sigma)$, $L_n(x, y, k, \sigma)$ with the closest σ value is selected. After that, gradient magnitudes, $\nabla f(x, y)$, and orientations, $\theta(x, y)$, are computed for points in the local region of the keypoint using the following equations.

$$\Delta x = L_i(x+1, y, k, \sigma) - L_i(x-1, y, k, \sigma) \tag{2.8}$$

$$\Delta y = L_i(x, y+1, k, \sigma) - L_i(x, y-1, k, \sigma) \tag{2.9}$$

$$\nabla f(x, y) = \sqrt{\Delta x^2 + \Delta y^2} \tag{2.10}$$

$$\theta(x,y) = \arctan\frac{\Delta y}{\Delta x} \tag{2.11}$$

An orientation histogram is created from the computed orientations. The histogram has 36 bins, each bin represents different 10 degree angle (i.e., it covers 360 degree in total). Each point in the region is weighted by its gradient magnitude as well as the distance from itself to the keypoint location.

As peaks in the histogram represent the main gradient directions around the keypoint, the highest peaks indicate the significant orientations. Let h_n be the height and θ_n be an average orientation of peak n in the histogram where n denotes bin indices in the range of $[0,36)$ and lower bound l_n and upper bound u_n of θ_n are $36u$ and $36(n+1)-1$, respectively. Eventually, if m is the index of the highest peak, then θ_m becomes the keypoint's orientation. Above that, for each peak n in which $h_n > 80\%h_m$ and $n \neq m$, a new keypoint is created with an orientation of $(l_n+u_n)/2$.

2.1.3.4 Keypoint descriptors

At last, keypoint descriptors are generated. A descriptor describes the local area of a keypoint and is used to compare with other descriptor in the matching process. More important, the orientations in the corresponding area are normalized by image rotation. The θ value of the keypoint is used to rotate the input image. Then, similar to the previous section, the points around the keypoint are collected and separated into subregions. Next, a weighted orientation histogram is created for each subregion. Finally, a feature vector that represents the keypoint is produced as a list of every regions' histogram bins as in Fig. 2.3.

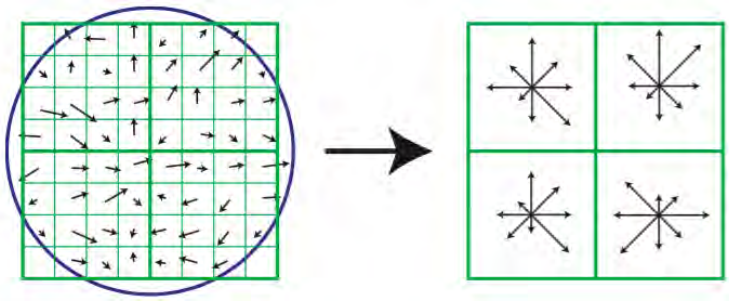


Figure 2.3: A SIFT keypoint descriptor (source [15])

2.2 Literature reviews

2.2.1 Curvature scale space (CSS)

Curvature scale space or CSS is a technique to describe a shape. To be specific, CSS captures inflection points at which the curvature changes its direction in the shape and uses maxima of the collected information to form a representation [1]. The inflection points are identified based on their *curvatures*, the degree of bending, over scales. In fact, if there is a curve and a tangent vector. The vector changes direction along the curve Fig. 2.4. When the curve bends shaper, the vector changes direction faster and when the curve is fairly straight, the direction is not changed much. Curvature measures how quickly the curve changes at a point on the curve as a rate of change of the unit vector's direction. Given that the unit tangent vector $T(t)$ and curvature κ of the curve C can be calculated as follow [24].

$$T(u) = \frac{C'(u)}{|C'(u)|}, \tag{2.12}$$

$$\kappa(u) = \frac{|T'(u)|}{|C'(u)|} = \frac{|C'(u) \times C''(u)|}{|C'(u)|^3}, \tag{2.13}$$

where u is an arc length parameter over C.

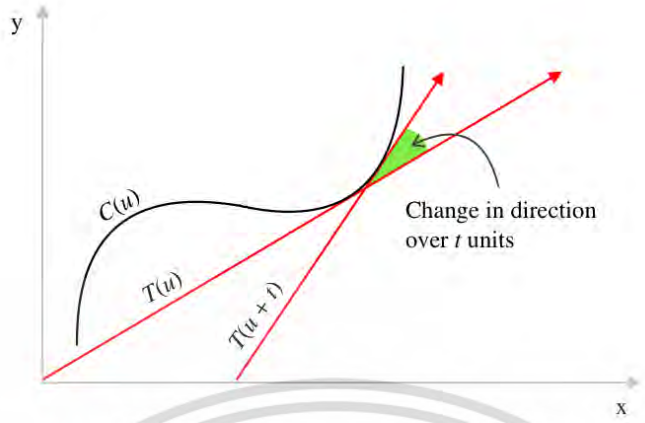


Figure 2.4: Tangent vector changes direction over a curve

If the curve is represent as a parametric vector $C(u) = (x(u), y(u))$, κ can be defined as

$$\kappa(u) = \frac{x'(u)y''(u) - x''(u)y'(u)}{(x'(u)^2 + y'(u)^2)^{\frac{3}{2}}} \tag{2.14}$$

The Gaussian kernel derivations can be used to evolve the curve. The evolved curve x and y components become

$$X_u(u,\sigma) = x(u) * G'(u,\sigma) \tag{2.15}$$

$$Y_u(u,\sigma) = y(u) * G'(u,\sigma) \tag{2.16}$$

$$X_{uu}(u,\sigma) = x(u) * G''(u,\sigma) \tag{2.17}$$

$$Y_{uu}(u,\sigma) = y(u) * G''(u,\sigma) \tag{2.18}$$

where $*$ is a convolution operator, $G(u,\sigma)$ is a 1-D Gaussian function, and $G'(u,\sigma)$ and $G''(u,\sigma)$ are the first and second derivatives of G, respectively.

$$G(u,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{u^2}{2\sigma^2}}, \tag{2.19}$$

$$G'(u,\sigma) = \frac{-u}{\sigma^2} G(u,\sigma), \tag{2.20}$$

$$G''(u,\sigma) = \frac{-\sigma^2 + u^2}{\sigma^4} G'(u,\sigma) \tag{2.21}$$

When using convolution of Gaussian derivations in place of derivatives, curvature of the evolved curve can be calculated as

$$\kappa(u) = \frac{X_u(u,\sigma)Y_{uu}(u,\sigma) - X_{uu}(u,\sigma)Y_u(u,\sigma)}{(X_u(u,\sigma)^2 + Y_u(u,\sigma)^2)^{3/2}} \tag{2.22}$$

From the above equations, a curve is represented based on its boundary points as a sequence of (x,y) coordinates. The curve is smoothed (evolved) by Gaussian functions with different σ. The locations of *curvature zero-crossings*, which are the points where the sign of curvatures change, are detected at different scales. When σ increases, the curve becomes smoother and the number of zero-crossings points decreases as shown in Fig. 2.5.

Figure 2.5: Curves smoothed by $G(u,\sigma)$ of different σ and their zero–crossing points (σ increases from left to right) (source [1])

The CSS zero-crossing points can be represented by a CSS image in which its x–axis denotes the normalized arc length and y-axis denotes the scale. From Fig. 2.6, zero-crossing points come in pairs at the inflection parts of the curve and become one point when the corresponding parts become smooth. Hence, the height of each pair indicates the concavity or convexity of the curve between them, which means the higher peaks, the higher curvature at the part.

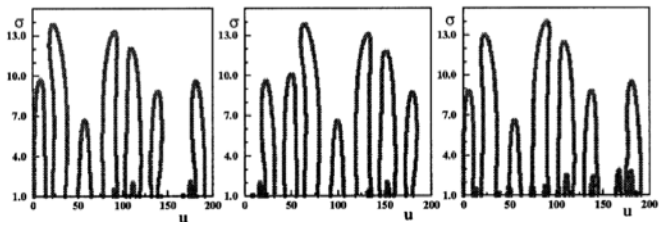


Figure 2.6: CSS images of similar curves (source [1])

From the figure, the circular shift of the hills is caused by rotation and the small hills are the result of noises. By shifting the CSS images properly and by considering only the significant peaks, CSS becomes invariant to image rotation and noise. Also, by normalizing the arc length of the input curves, it becomes invariant to image scale.

Moreover, as the curve can be represented as a pair of peaks in its CSS image, the amount of information needed to store is greatly decreased. Besides, since the points in CSS images indicate concavity or convexity of the curve, more local information could be extracted. For instance, it is possible to detect corners from pairs of zero-crossing points as the middle point between them. However, in order to detected actual corners, when zero-crossing points are selected from high scale CSS images, they have to be tracked back to get the corresponding points in lower scales since their positions change when the curve evolves. As an additional information, sharp corners has higher curvature than the round ones. This has to be considered to not detect false corners from the curve [17].

2.2.2 Fourier descriptor (FD)

Other than CSS, it is also possible to describe a shape contour with Fourier descriptor. Fourier descriptor or FD, is a method to describe two dimensional shape using Fourier transform function. With the function, shape can be converted back and forth, as well as simplified (smoothed) easily. It is also one of the most popular algorithm used to solve computer vision-based recognition problem.

To generate a FD of a shape contour, each point (x,y) coordinate is represented as a complex-valued number with real and imaginary parts, $x+yj$. Then, the following discrete Fourier transform (DFT) function is applied.

$$\begin{aligned} DFT(u) &= \sum_{x=0}^{N-1} C(x) e^{\frac{-j2\pi ux}{N}} \\ &= \sum_{x=0}^{N-1} C(x)[\cos(\frac{2\pi ux}{N}) - j\sin(\frac{2\pi ux}{N})], \end{aligned} \tag{2.23}$$

where u is an indexing parameter, C is the shape contour, and N is the number of elements in C, which is also the number of bases. To reconstruct the shape from its descriptor, the following invert version of the DFT is taken.

$$C(x) = \frac{1}{N} \sum_{u=0}^{N-1} DFT(u) e^{\frac{j2\pi ux}{N}} \tag{2.24}$$

As Fourier function represents the shape with a summation of sinusoidal terms from different frequencies, DFT is often said to be a frequency domain representation of the original input shape contour. By eliminating some terms in the sequence, the shape can be adjusted. It is common to eliminate high frequency terms to reduce noise in images. Notably, the more terms remaining, the more originality remains, and with fewer terms, the shape becomes smoother and rounder.

Also, since shapes can be approximated with a small number of Fourier's parameters, Fourier descriptors are commonly used to classify shapes. Some algorithms consider two shapes as matched if their first T terms are matched.

Other than these facts, properties of the transformation are also inherited by the descriptor. For examples, duality property: minus sign indicates that the shape will be flip, convolution property: convolution in time domain is equivalent to multiplication

in freq domain, time scaling property: expanding signal in time domain, narrowing the signal in frequency domain, translation invariance property: the descriptors remain the same at every location the shape is move to, scaling property: if the shape is scaled by c, then its descriptors are also scaled by c, and rotation property: rotating the shape only affects the phase of the descriptors.

2.2.3 Shock graph

Shock graph is introduced as a 2-D shape representation that is invariant to changes in viewpoint, noises, and occlusions. It is a structural description that is derived from shocks or singularities of a curve evolution process, acting on object contours [21].

2.2.3.1 Shape and shock

For a simple closed curve in 2-D plane, the following evolution equation is used to evolved the curve.

$$\begin{aligned} C_t &= (1+\alpha\kappa)N, \\ C(s,0) &= C_0(s), \end{aligned} \tag{2.25}$$

where $C(s,t)$ is the curve, $N(s,t)$ is the inward normal, s is indexing parameter over C, t is the time of deformation, $\alpha \leq 0$ controls the regularizing effects of curvature κ. When $\alpha = 0$, the equation is hyperbolic and shocks can be formed. The shock is coloring according to the local variation of the radius function at its middle axis (as shown in Fig. 2.7). This colored description later provides a richer foundation for recognition. A 1-shock will be found at a protrusion as the radius function changes monotonically. A 2-shock will be found at a neck (local minimum radius), which is followed by two 1-shocks in opposite directions. A 3-shock radius function is constant along the middle axis, for a curve segment that bend with parallel sides, and a 4-shock is found when the radius function gives a strict local maximum, which happens when the curve evolves into a point. The center of these shocks gives the middle axis. Also, it is noticable

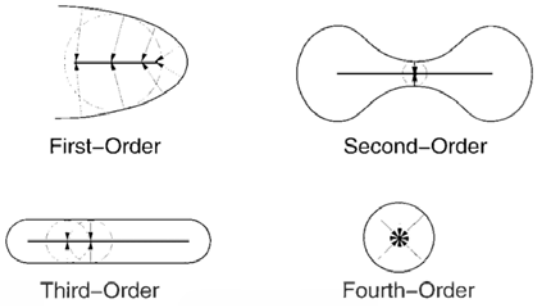


Figure 2.7: Four types of shocks coloring (source [21])

that 2-shocks and 4-shocks are isolated while 1-shocks and 3-shocks are neighbored by other shocks of the same type.

2.2.3.2 Shock graph

Shock graph (**SG**) is an abract representation of shocks derived from a curve evolution process. Shock types label each vertex in the graph and the shock formation times will direct edges to provide an ordering for matching.

To build a **SG**, shocks of the same type that are grouped together when they are connected. A shock group is labeled by $1'$, 2, $3'$, or 4 to indicate its type (i.e., $'$ symbol donotes a curve segment) along with integer i and t_i, which are indexing parameter and time interval, respectively, according to the evaluated radius function. If the group is $1'$ with a branch, then it is breaked apart at the branch point. For more information, t_i will be an interval for shock group with type $1'$, and a scalar number for the rest. In addition, # denotes a start symbol (root) and Φ denotes a terminal (leaf). Hence, **SG** is a connected graph with directed edges, rooted at #, such that V are shock groups, and has no cycle (i.e., it is an acyclic graph).

SG is built with an analogy of growing a shape. By reversing the evolution process, lumps are added onto a point or a seed. Hence, the children of # are shock groups from the last contour evolution and the parents of Φ are shock groups from the first contour

evolution. The closer to the root, the more significant the groups are because they hold the central shape feature.

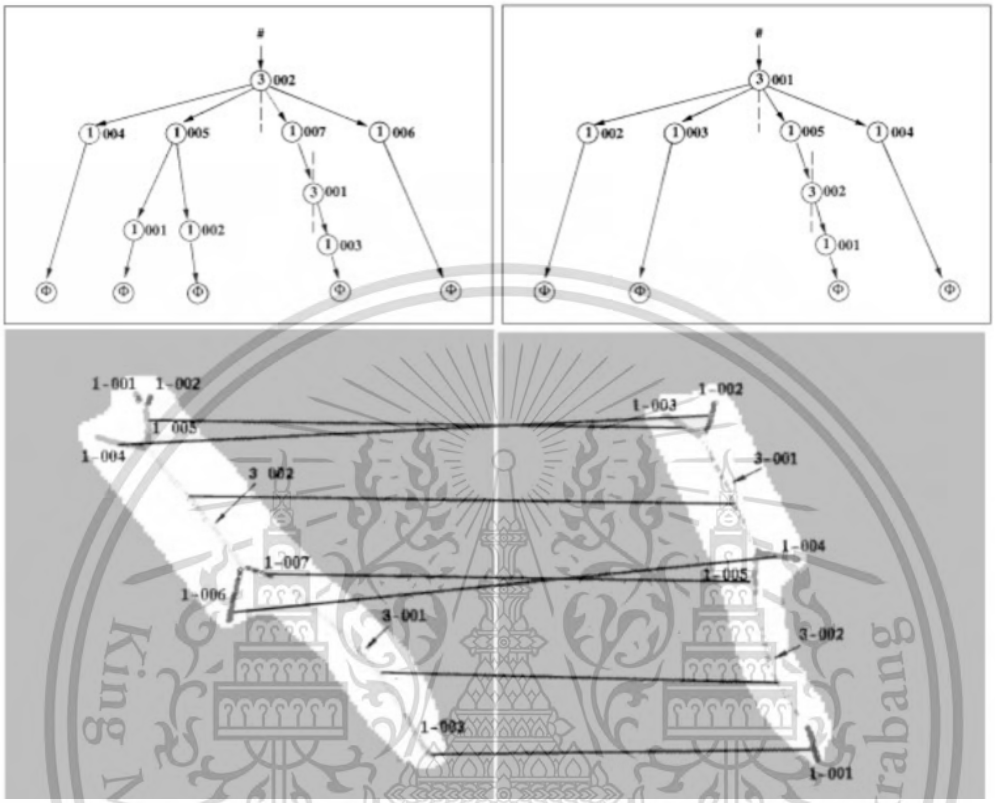


Figure 2.8: Examples of shock graphs and shapes (source [21])

Shock graph grammar There are rules when growing a shape from a seed. The rules are separated into cases of 1) birth, 2) protrusion, 3) union, and 4) death as shown in Fig. 2.9. The groap building operation is started at the # node, then replace the left node of a rule by the right node until no more replacements can be made. From the figure,

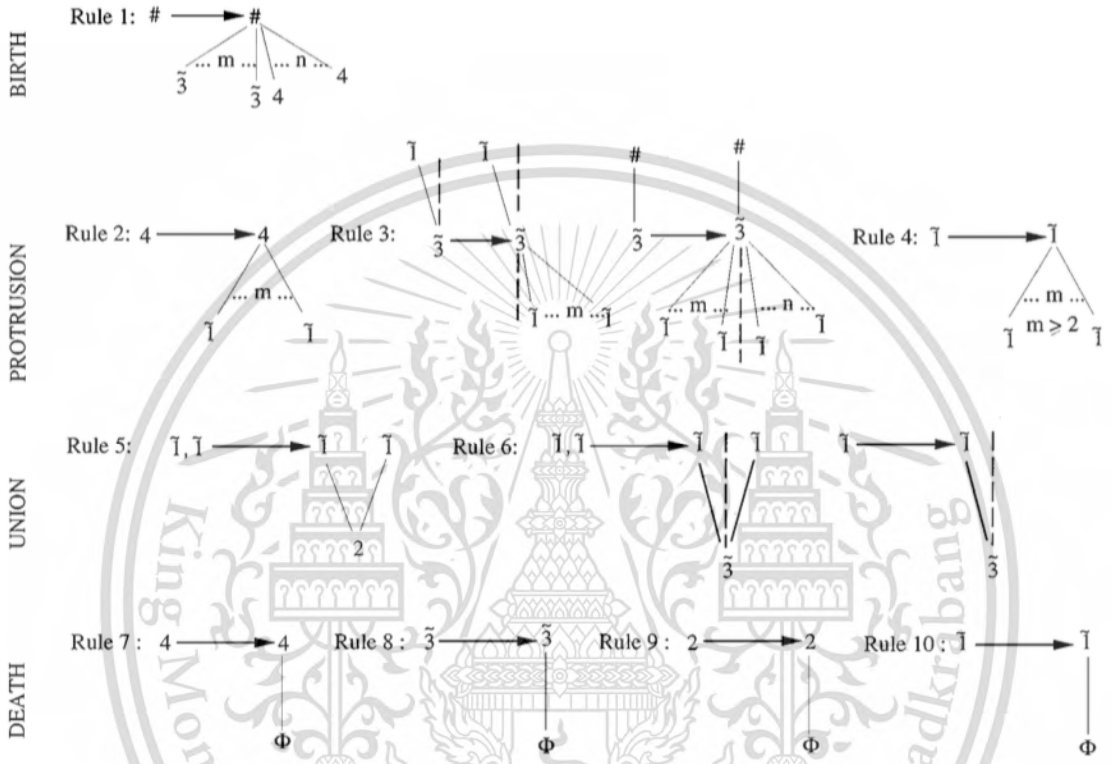


Figure 2.9: The shock graph grammar. Dashed lines of $3'$ parition the group distinct ends. (source [21])

- Only rule 5 creates 2-shocks
- Only rule 1 creates 4-shocks
- Only rule 1 and rule 6 create 3-shocks
- 3-shocks and 4-shocks are the same when they are created from # (rule 1)
- 3-shocks and 2-shocks are the same in the case of union (rule 5 and 6)

- 1-shocks can not be added onto the end of 3-shocks at the same side (rule 3)

2.2.3.3 Shock graph to shock tree

A DAG shock graph can be reduced into a tree, named shock tree.
Let $G = (V,E)$ be the shock graph where $|V| = n$. If a loop L, which is a subgraph of G formed from node a to node b by the intersection of two directed paths, P_1 and P_2, is found in G, then redefine L to be the union of a and disjoint paths P_1' and $_2'$ where $P_1' = P_1 \cup b \cup \Phi$ and $P_2' = P_2 \cup b \cup \Phi$, respectively. After removing loops from, a shock tree of G is created.

A shock tree is represented as an adjacency matrix of $0,1$, with 1 inidates an edge connected a parent to its child. Also, any shock subtree is a submatrix of the adjacency matrix.

2.2.3.4 Shock graph matching

The problem of recognizing shape becomes a problem of graph matching. In fact, to recognize a shape in a scene, we have to match a shock graph in database to the shock graph of the shape which is a subgraph of the scene. Here, the problem of recognizing become a graph search problem, called *largest subgraph isomorphism problem*, which can be solved as a *binary integer programming* (BIP) optimization problem with the following model,

$$\begin{aligned} &Minimize\ Z = -\frac{1}{2}\sum_{u\in G}\sum_{v\in H}\Phi(u,v)||u,v|| \\ &s.t.\ \sum_{u'\in H}\Phi(u,u') \le 1, \forall u \in G \\ &\sum_{v\in G}\Phi(v,v') \le 1, \forall v' \in H \qquad (2.26) \\ &and\ \Phi(x,y) \in 0,1, \forall x \in G, \forall y \in H, \end{aligned}$$

where $||.||$ is a measure of similarity between labels of the corresponsing nodes in two shock graphs, G and H. The goal is to find the coefficient matrix M that minimize the objective function Z.

In term of algorithm, let

- $G = (L_1, E_1)$ be a shock graph represents an object in a scene
- $H = (L_2, E_2)$ be another shock graph represents an object in database
- $|V_1| = n_1, |V_2| = n_2$
- d be the maximum degree of any vertex in G and H (i.e., $d = max(\delta(G), \delta(H))$
- $\chi(v) \in R^{d-1}$ be an eigen-decomposition vector of any vertex v
- $C(u,v)$ be a shock distance between any pair of vertices u and v
- $\Phi(G,H)$ be the set of final node correspondences between G and H, which holds the solution of graph matching problem. Initialize as an empty set

The steps are

1. Create an empty set $\Phi(G,H)$
2. Find d
3. For $u \in V_1$, compute $\chi(u) \in R^{d-1}$
4. For $v \in V_2$, compute $\chi(v) \in R^{d-1}$
5. As long as G and H are not empty
   (a) G_u = rooted subtree of G at u
   (b) H_v = rooted subtree of H at v

(c) Create a $V_{1_u} \times V_{2_v}$ matrix $\prod(G_u, H_v)$ which (u, v) element has the value $C(u,v)||\chi(u) - \chi(v)||$

(d) Create a weight graph $\zeta(V_1, V_2, E_\zeta)$ with edge weight from the matrix $\prod(G, H)$

(e) Find M = max number of element with minimum weight in ζ

(f) Let (u', v') is the pair of vertices with M

(g) Add the pair to *Phi*

(h) Back to 5(a) with (u', v')

(i) Remove the nodes that are added to Φ from G_u and H_v

6. Return the cost of Φ as a result (computed using Z, the cost function)

Chapter 3

Proposed method

As mention before, the objective of this thesis is to develop a shape recognition algorithm that is invariant to image scale, translation, and rotation. Several computer vision based identification methods with those invariance have been reviewed and one of them ignited a motivation. The method is SIFT algorithm which is explained in Section 2.1.3. Originally, SIFT is used to search and describe keypoints in an image which is a 2-D signal. These SIFT keypoints are places where intensity levels significantly change over the image. In combination with the fact that a shapes contour can be converted into a 1-D signal and the rate of change in magnitude of the signal characterizes different shapes, the SIFT contour algorithm is used to detect and describe keypoints in contours. As the original SIFT is invariant to scale, rotation, and translation, the proposed 1-D version of it is also expected to have to same properties.

Similar to the primitive method, SIFT contour is separated into sections of keypoint detection and feature descriptor which complete the first task and the second task of shape recognition. As for the third task, it is handled by a matching algorithm that computes similarity between two SIFT contour descrip- tors. Notably, the proposed algorithm is applied after image preprocessing processes that convert an input shape image into a contour, which is a sequence of Cartesian coordinate system's 2-D points.

These processes are described in the following sections in detail.

3.1 Keypoint detection

A keypoint in a contour is an outstanding point where its signature signal significantly changes. It represents a local feature to be automatically located and described by the algorithm. In order to extract all keypoints, two processes have to be executed in order. The first process is done to detect keypoints and the second process is done to refine keypoints. In the following subsections, these two processes are throughly explained.

3.1.1 Detect keypoint

Adapted from SIFT, the keypoints are detected from different scales of signal for the feature to be invariant to image scale. There are differences in computations due to the fact that the original extracts keypoints from the whole image, but the proposed algorithm extracts keypoints from a contour. Additionally, there is a difference in dimensions of information between the original and the proposed algorithm. In fact, for the algorithm to detect keypoints, the processes include 1) creation of Gaussian kernel, 2) contour sampling, 3) one dimensional convolution, 4) one dimensional conversion, 5) difference of signals, and 6) selection of keypoint candidates. The detail explanations are given below.

3.1.1.1 Creation of Gaussian kernel

Gaussian function is well known as a continuous scale space function. By applying Gaussian of different scale (σ) to the data, the invariant property can be acquired. Hence, 1-D Gaussian kernels are created to scale the input information. The first Gaussian kernel is a result of Gaussian function with σ_0. Later, the σ is increased by k in

each scale. A Gaussian kernel in scale i is denoted as G_i created using the following equation.

$$G_i(s) = G(s, k^i\sigma_0) = \frac{1}{k^i\sigma_0\sqrt{2\pi}} e^{-\frac{s^2}{2(k^i\sigma_0)^2}} \tag{3.1}$$

where s is an integer parameter that iterates over G. Incidentally, as Gaussian kernel has bell shape and the values outside the range of $[-3\sigma, 3\sigma]$ can be ignored as they are too close to zero, the size of G_i is $6k^i\sigma_0$, rounded into an odd number.

3.1.1.2 Contour sampling

A leaf contour is an ordered sequence of 2-D points on a leaf boundary in which the first and the last points are the same (i.e., it forms a circular planar curve). These points in the contour are uniformly distributed along the boundary; therefore, the sequence captures a characteristic of leaf shape. In general, the contour has a part of features of the original leaf pattern which makes extracting features from the contour for classification become sensible.

Local characteristics of shapes can be captured at the finer scales which are blurred by the lower σ Gaussian kernels. In contrast, the global characteristics can be captured at the rougher scales where the contour is blurred with the higher σ Gaussian kernels. Because of these reasons, more points are sampled from contours in finer scales while less points are sampled from the rougher ones for the calculations to cover proper parts of different scales' contours. Thus, the original contour is down-sampled for every octave. To be precise, E scales complete an octave. When octave changes, the contour C_{i-1} is down-sampled by d to create C_i (d is fixed to 2 in the implementation).

3.1.1.3 One dimensional convolution

Convolution is used to blur the contour. A contour C_i is blurred by the 1-D Gaussian kernel G_i created before using 1-D convolution functions below.

$$C_{i,x}(u) = \sum_{s=-\frac{Size_i-1}{2}}^{\frac{Size_i-1}{2}} C_{i,x}(u-s)G_i(s), \tag{3.2}$$

$$C_{i,y}(u) = \sum_{s=-\frac{Size_i-1}{2}}^{\frac{Size_i-1}{2}} C_{i,y}(u-s)G_i(s), \tag{3.3}$$

where u is an arc length parameter of the contour, $Size_i$ is the size of G_i, s is an integer parameter, and x and y are components of point u in C_i. The convolution has to be done in 1-D manner separately for both x and y components; otherwise, if the 2-D convolution function is used, then the result will be a blurry line. The result of this process is an ordered sequence of convoluted contours C in which each contour is evolved by a Gaussian kernel of different σ.

3.1.1.4 Dimensional conversion

The algorithm aims to capture shapes in the form of 1-D signals. As a consequence, the contour has to be converted using signatures. A signature is a 1-D function that converts a 2-D boundary point into a scalar value. With a contour as an input, the function generates a sequence of numbers with the same ordering as the contour points as output. Therefore, the contour can be expressed as a 1-D signal which describes the contour's characteristic over its arc length. $S_i(u)$ is a signature function of $C_i(u)$ $(S_i : \mathbb{R}^2 \rightarrow \mathbb{R})$

There are many signature functions that can be used in the proposed algorithm. For example, curvature function, contour-centroid distance function, and difference of tangent angles function.

Curvature signature is a signature that converts contour into a vector of curvatures measured along the contour. To calculate the signature, for each arc length parameter u in a contour C_i, find $\kappa_i(u)$ using Eq. (2.22).

Contour-centroid distance, CCD, function is a function that sequentially measures Euclidean distances from a contour's centroid to each contour element (see Eq. (3.4) and Fig. 3.1).

$$CCD(u) = ((Centroid_x(u) - x(u))^2 + (Centroid_y(u) - y(u))^2)^{\frac{1}{2}}, \tag{3.4}$$

recall that $x(u)$ and $y(u)$ are element of C_i at position u.

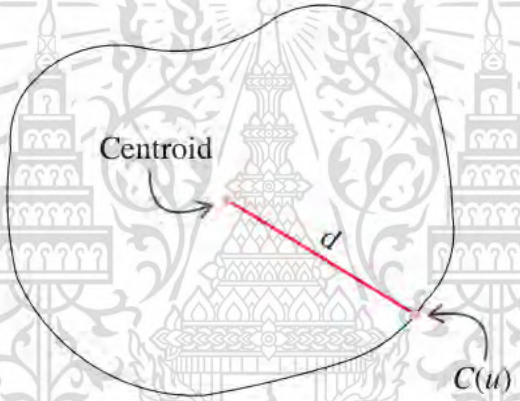


Figure 3.1: d is Euclidean distance between the centroid and a point on contour

The centroid is also known as geometric center. It is the average position of all the points in the contour in all coordinate directions. For two dimensional shape contour, its centroid is calculated as follows:

$$Centroid_x(u) = \frac{1}{N}\sum_{u=0}^{N-1} x(u)$$
$$Centroid_y(u) = \frac{1}{N}\sum_{u=0}^{N-1} y(u)$$

(3.5)

As for the difference of tangent angles function. It is a series of angle differences of tangent vector along a shape contour. Its accuracy depends on the positions' of tangent angles used to calculate the signature, which is controlled by a measurement window. To generate the signature, let $T(u)$ be a tangent vector at $C_i(u)$, $T(u) = \langle x'(u), y'(u) \rangle$ and $\theta(u)$ be its tangent angle, calculated as $\theta(u) = \arctan(\frac{y'(u)}{x'(u)})$. The difference of tangent angle, DTA, at $C_i(u)$ is

$$\begin{aligned} DTA(u) &= \theta(u) - \theta(u-w) \\ &= \arctan(\frac{y(u) - y(u-w)}{x(u) - x(u-w)}), \end{aligned} \tag{3.6}$$

where w is a constant number denotes the size of the measurement window.

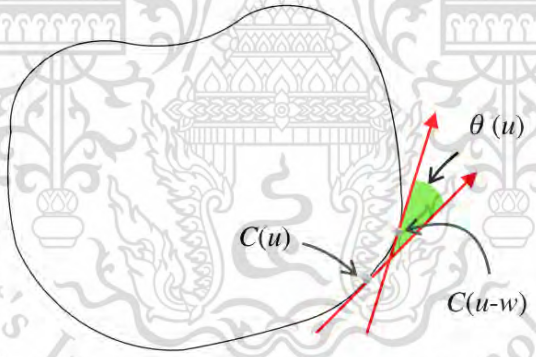


Figure 3.2: θ is the difference in tangent angles between two boundary points

3.1.1.5 Difference of signals

This process is equivalent to the Difference of Gaussian calculation in Lowe's version. The name is changed due to the fact that signals are used instead of intensity

values. The formula is as follows.

$$Dos_i(u) = S_{i+1}(u) - S_i(u) \tag{3.7}$$

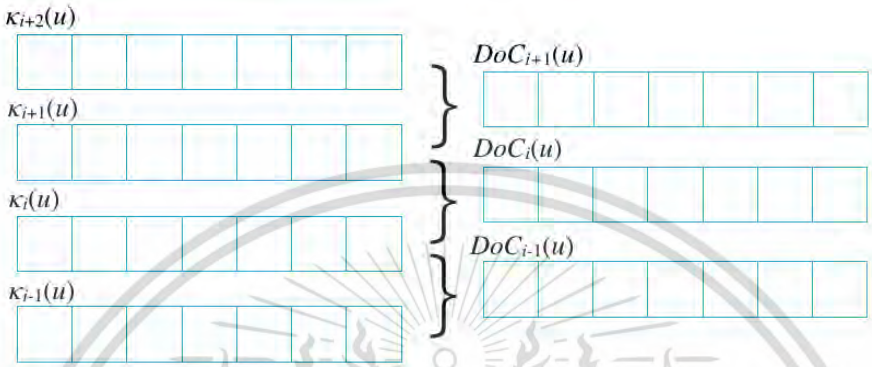


Figure 3.3: Difference of signals at scale i, Dos_i

3.1.1.6 Selection of keypoint candidates

After the *Dos* are calculated for every scale, keypoint candidates are selected. As stated previously that a keypoint is the place where the signal magnitude significantly change, a point becomes a keypoint candidate if and only if it is an extrema comparing with its two neighbors in the same scale, three neighbors in the previous scale, and three neighbors in the next scale (i.e., $Dos_i(u)$ is higher or lower than all points in $\{Dos_i(u-1)$, $Dos_i(u+1)$, $Dos_{i-1}(u-1)$, $Dos_{i-1}(u)$, $Dos_{i-1}(u+1)$, $Dos_{i+1}(u-1)$, $Dos_{i+1}(u)$, $Dos_{i+1}(u+1)\}$). As an example, E in Fig. 3.4 will be a candidate if it is the maximum or the minimum in its neighborhood {A, B, C, D, E, F, G, H, I}.

To summarize, the first part of keypoint detection process, detect keypoint (Section 3.1.1), receives an original contour and generates a set of keypoint candidates as the input of the next part which refines the keypoints.

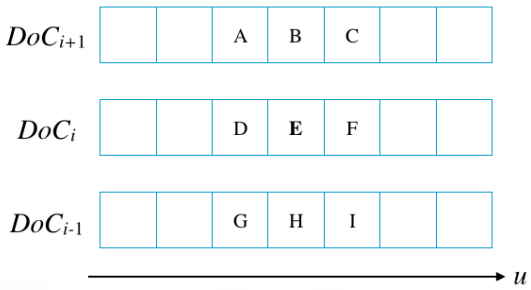


Figure 3.4: 1-D SIFT keypoint selection

3.1.2 Refine keypoint

The set of candidate keypoints extracted from a shape is refined to eliminate noises and keypoints that are too close to each other while having similar values. Keypoints that are considered as noises are those that have lower absolute magnitude but not zero, comparing with a predefined threshold, and a keypoint is considered as too close to another keypoint if at least one of its considerate neighbors are keypoints. The keypoints that are not an extrema in its neighborhood of size r are removed from the set. With the elimination, the jitter noises are filtered out and only keypoints that capture different distinguishing characteristic of a shape are remained. This contributes higher accuracy to the algorithm.

In brief, the result of keypoint detection (Section 3.1) is a set of keypoints extracted from a shape, which is represented by its contour signature. Each keypoint p is a tuple of $\langle i, x, y, u, S_i(u) \rangle$ where (x, y) is a 2-D position of p in contour i, u is an arc length parameter that indicates the position of p in signal i, and $S_i(u)$ is the signal magnitude measured at the point.

3.2 Feature descriptor

The detected keypoints have to be described as feature descriptors. A feature descriptor is an ordered sequence of real numbers, called feature vector. For a keypoint, the feature vector creation process is separated into two parts, all of which are explained below.

3.2.1 Curve segment characterization

In the original work of Lowe where the interested domain is the whole input image, orientations are assigned to image pixels to remove the effect of image rotation. In contrast, for the proposed algorithm, curvature values are used to describe the interested local part of a shape. The orientation assignment is not necessary as a curvature is invariant to rotation by itself. As a result, Lowe's idea was modified.

Instead of an orientation histogram, a normalized curvature histogram is created for a curve segment CS_p which includes 32 adjacent points of p from C_i and the point p. The histogram consists of 26 bins of curvature, starts from $[-\infty, -1.30)$ to $[1.30, \infty)$, with a step size of 0.10. To create the histogram, curvatures are calculated for all points in CS_p and these points are assigned to the histogram's bins according to their curvatures.

Peaks in the histogram contain important local features of the curve segment. Additional keypoints are created for any peaks within 70% of the highest peak with its mean curvature and the same i, x, y, and u as p to emphasize the curve segment in which p belongs to. Hence, when the keypoint detection fails to detect p, these additional points will act as its backups.

3.2.2 Keypoint descriptor

Similar to the previous process, normalized curvature histograms are created. However, for each keypoint p, CS_p is separated into two sections, each of which has its own

histogram. The first section, H_L, describes the left side while the another section, H_R, describes the right side of CS_p, divided by p (see Fig. 3.5). Later, a feature descriptor v of p is created from the two histograms as a vector that contains their frequencies (see Fig. 3.6). After the feature vectors are formed for every keypoint of the input shape image, they can be used in a matching process as a target or a model to recognize the shape.

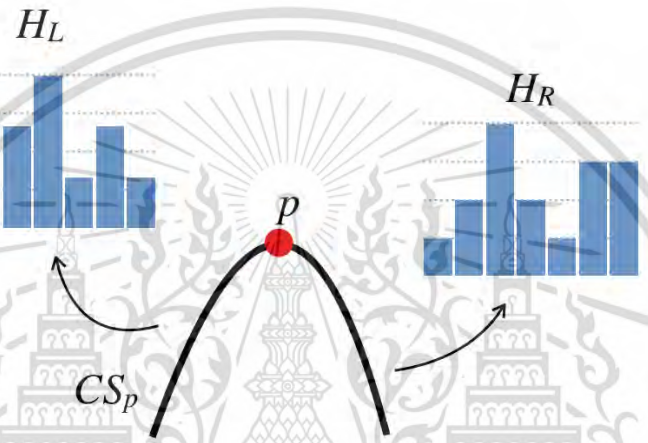


Figure 3.5: H_L and H_R of p

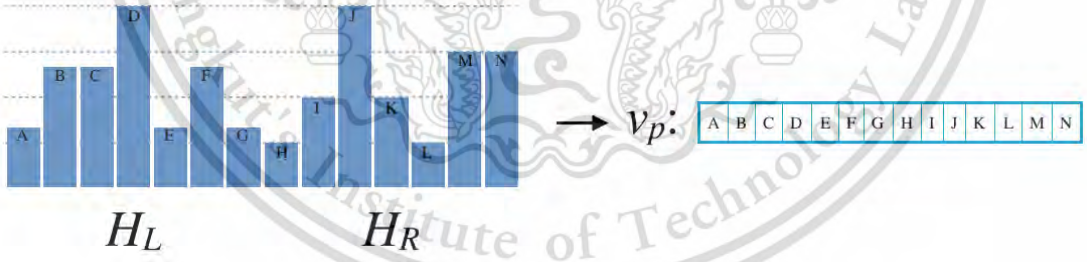


Figure 3.6: A feature descriptor of a keypoint p

3.3 Contour-SIFT keypoint matching

To recognize a shape, its set of keypoint descriptors is compared to sets of model keypoint descriptors in database. The approximate correspondence between these two sets is represented by similarity value. The model with more identical local characteristics to the target, the higher similarity. The proposed keypoint matching process is performed to calculate the similarity between two sets of keypoint descriptors, a target T, and a model M. When matching, the similarity of descriptors are separately computed for every scale i, and the average of these similarities will be returned as a result.

$$Similarity_{av}(T,M) = \frac{1}{N+1}\sum_{i=0}^{N} Similarity(T_i,M_i), \tag{3.8}$$

where N is the highest scale of p found in T and M.
Let T_i be a set of descriptors with scale i in T and M_i be a set of descriptors with the same scale in M. If any of T_i and M_i is empty, then the similarity is zero; otherwise, both sets are sorted by keypoint's magnitude, $S_i(u)$, in ascending order, then the similarity value is computed as follows,

$$Similarity(T_i,M_i) = \frac{matches(T_i,M_i) + matches(M_i,R_i)}{max(|T_i|,|M_i|)}, \tag{3.9}$$

where $matches(A,B)$ is the number of keypoint descriptors in A that are matched to any descirptors in B.

A descriptor d_A of p_A in A is considered as successfully matched to any descriptor d_B of p_B in B if p_A has the same sign of S_i as p_B, the difference between u of p_A and u of p_B is lower than 5% of $|C_i|$, and the closest matched distance from d_A to every d_B is higher than 60% of the second closest or the closest distance is lower than a predefined

threshold, *matched thresh*. The distance is calculated using the following equation.

$$SquareDist(d_A, d_B) = \sum_{id=0}^{D-1} (d_A(id) - d_B(id))^2 \tag{3.10}$$

where id is an indexing parameter over d of size D. Also, if d_B is matched to any d_A, then p_B is removed from B before the matching process of the next p_A begin.

Using the matching algorithm, the similarity between a target and a model of the same shape will be higher comparing with the similarity between itself and models with different shapes. It is important to note that the accuracy of keypoint matching is highly dependent on the choices of signature function and necessary parameters (i.e., σ_0, k, and r) of the keypoint detection process.

Chapter 4

Experiments and discussion

In this chapter, a system that utilizes the proposed algorithm and experimental datasets are explained in the first and the second section, respectively. Then, various experiments that were conducted for verification are explained.

4.1 Shape based-object recognition system using Contour-SIFT algorithm

As shown in Fig. 4.1, the shape recognition system receives a target object image as input and performs three processes to identify the object's identity. The three processes are image preprocessing, feature extraction and description using Contour-SIFT algorithm, and shape matching, which are described below for more detail.

4.1.1 Image preprocessing

The purpose of this process is to create an object contour which is required as an input of the proposed Contour-SIFT algorithm. To achieve that, the input image I is converted into gray-scale using Eq. (4.1) [4]. Next, the gray-scale image is converted

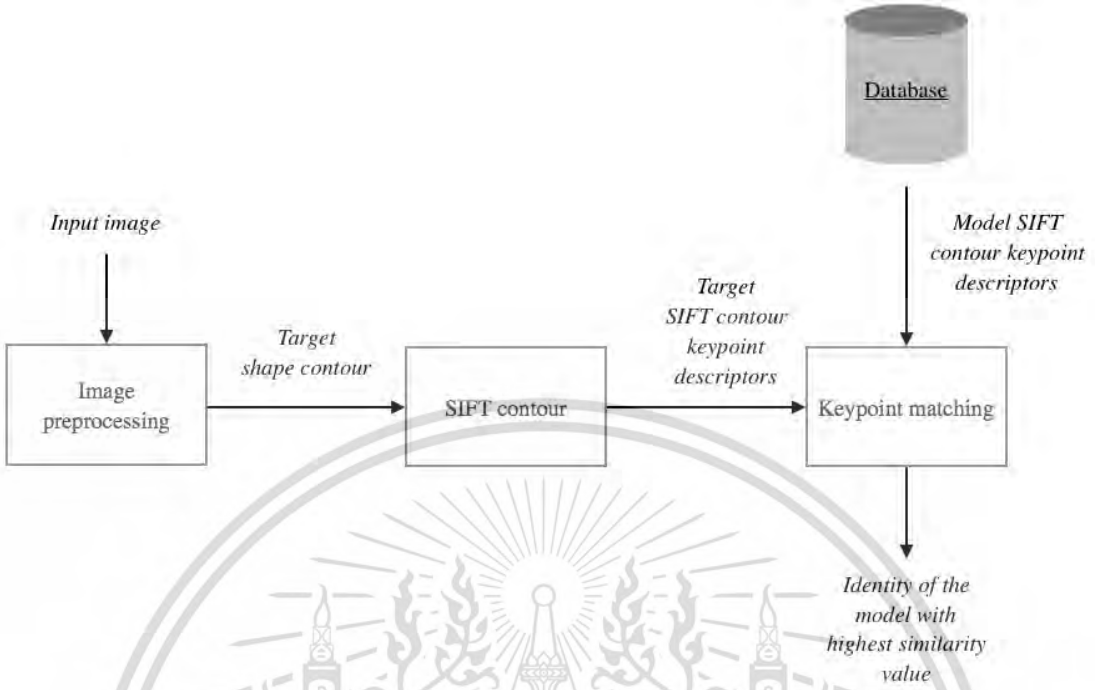


Figure 4.1: Overall process of the proposed object recognition system

into a binary image by thresholding. As an explanation, an intensity value of 255 will be assigned to a pixel in the binary image if the corresponding pixel in the gray-scale image is higher than a threshold; otherwise, an intensity value of 0 will be assigned instead. To be precise, afther the thresholding, the object is painted black and the background is painted white. After that, a morphological operator is applied to the thresholded image in order to fill unwanted holes in the two areas (i.e., black holes in the white area and white holes in the black area). By removing these holes, some noises in the image are eliminated. Finally, the object contour is extracted as a border of the black area. The processes can be explained using Fig. 4.2. With this, the input leaf image is converted into the leaf contour.

$$I(x,y) = 0.299R + 0.587G + 0.114B, \tag{4.1}$$

where R, G, and B are the red component, green component, and blue component of an image pixel $I(x, y)$, respectively.

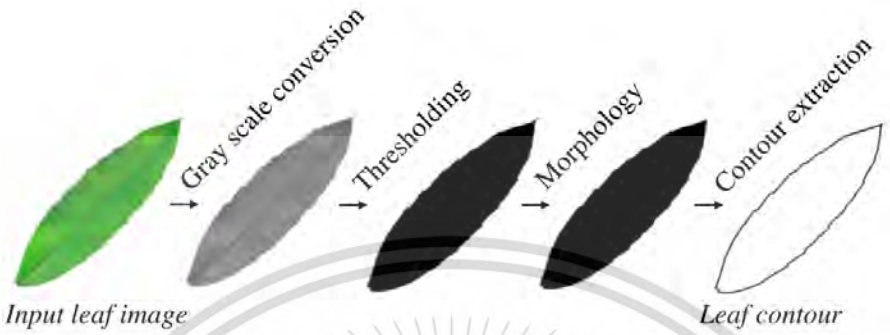


Figure 4.2: Preprocessing process

4.1.2 Feature extraction and feature description

The next step is to extract the interested characteristics from the contour obtained from the preprocessing process. The algorithm explained in Chapter 3 is used to extract the contour descriptor, which is a set of keypoint descriptors.

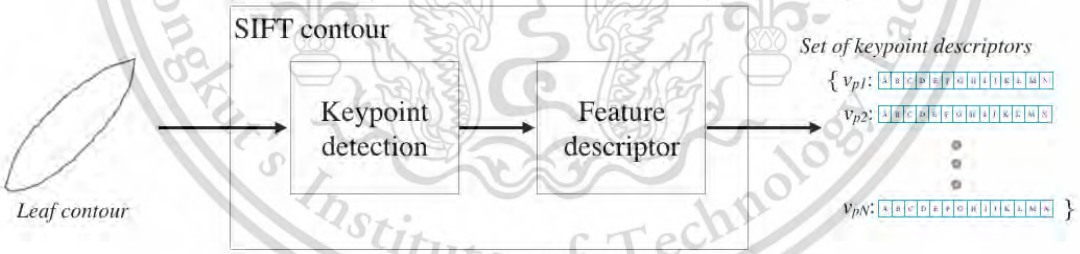


Figure 4.3: N feature vectors are extracted from M leaf contours

4.1.3 Shape matching

Shape matching is the process where shape recognition happens. Before the deployment of the system, sets of keypoint descriptors have to be extracted from training

images and stored in the system's database as models beforehand. As such, when the target shape's descriptor, which is a set of keypoints, reaches this step, it will be matched with every model to get similarity values. The model with the highest similarity value is expected to have the same identity as the target object. Therefore, its identity is given out as the system's output.

All in all, various experiments were conducted with this system to verify the proposed Contour-SIFT and matching algorithm. The experiments and their experimental results are explained in the next sections along with discussion.

4.2 Experimental dataset

There are two experimental datasets that were used as input of the system.

4.2.1 Dataset 1: Flavia200

The first dataset is chosen from Flavia [27]. Flavia dataset is a public dataset for leaf recognition problem. It originally contains images of 32 leaf spicies. Each leaf image in the dataset contains only one leaf, has white background, and a dimension of 1280×960 pixels.

From 32 species of the original Flavia dataset, the experimental dataset only includes 20 of them. From each of these 20 species, 10 images are chosen to be included, resulting in a dataset with 200 images. Fig. 4.4 shows leaves of different species from the dataset.

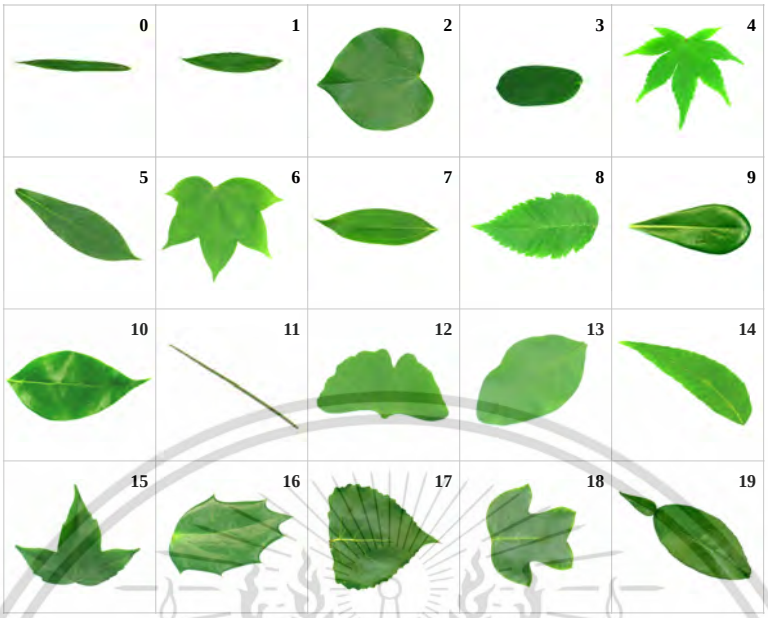


Figure 4.4: Examples of different leaf species from Flavia200 dataset. (source [27])

4.2.2 Dataset 2: Kimia216

Another dataset is Kimia216 [20]. Kimia216 consists of 18 classes, each class consists of 12 images. It contains shapes outlines for birds, bones, brick, camels, car, children, classic cards, elephants, faces, forks, fountains, glasses, hammers, hearts, keys, rays, turtles and a miscellaneous class. A program was implemented with c++ language to extract each object from the original data, Fig. 4.5. After the extraction, the resulting images' dimension is 69×63 pixels. Contours extracted from Kimia216 images are shown in Fig. 4.6.

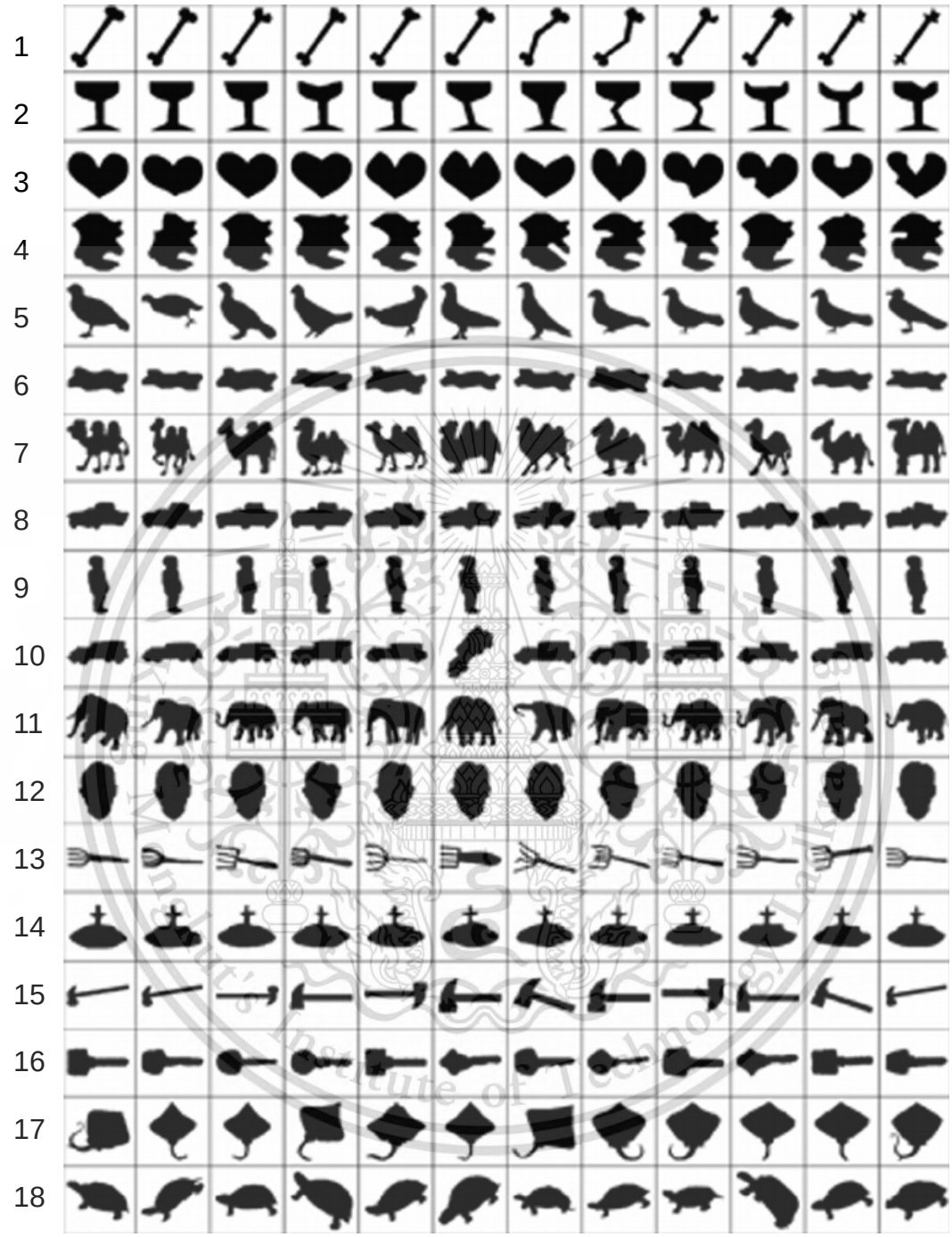


Figure 4.5: Images from Kimia216 dataset. (source [20])

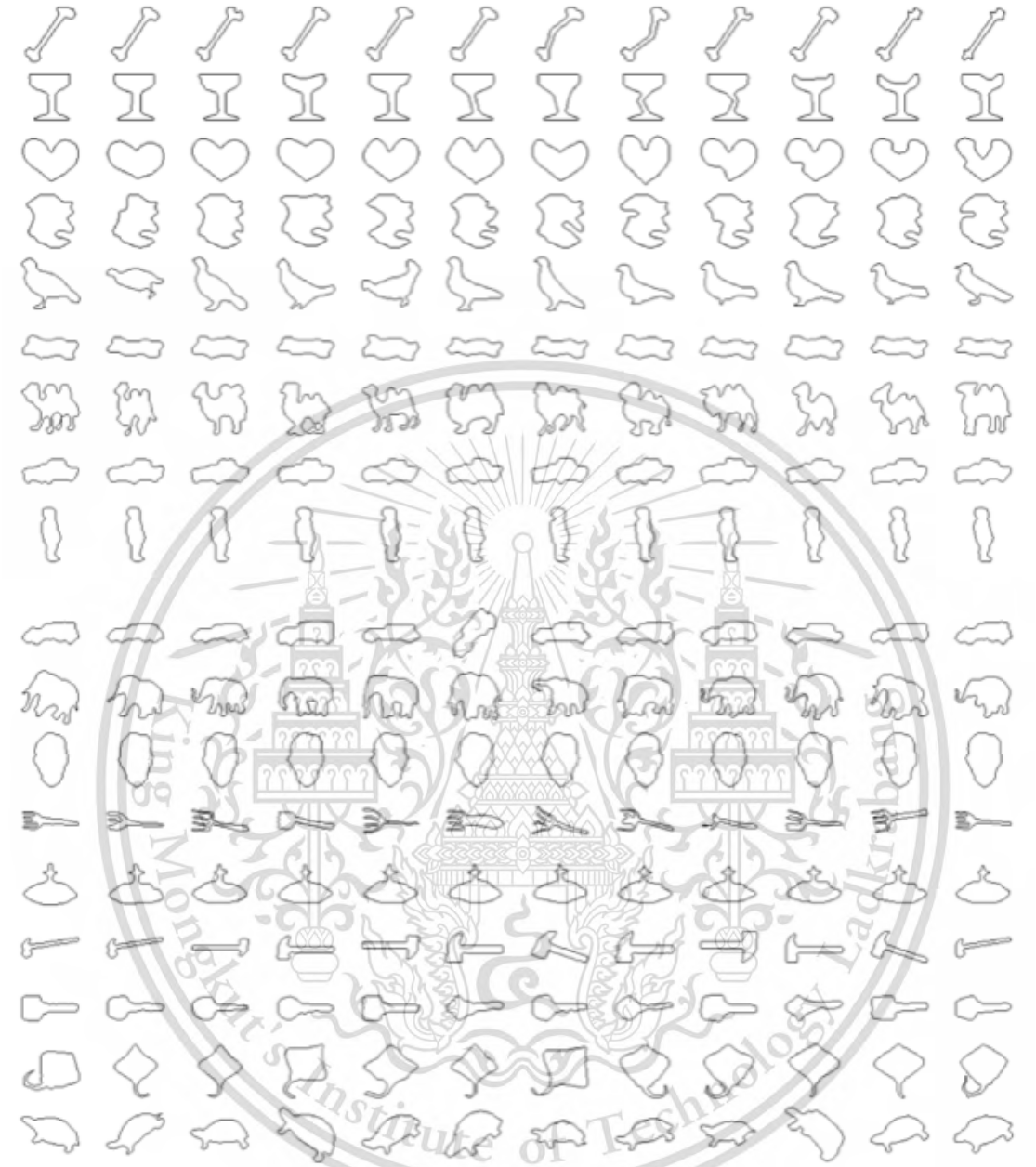


Figure 4.6: Contours from Kimia216 dataset

4.3 Experiment 1: Effect of signature functions

4.3.1 Objective

Performance of the proposed Contour-SIFT depends on its signature function used to convert a 2-D shape contour into a 1-D signal. An experiment was conducted to measure the performance of the algorithm when different signature is used. Three signature functions discussed in Section 3.1.1.4 were tested. They are curvature function, contour-centroid distance function, and difference of tangent angles.

4.3.2 Experiment set-up

This experiment is done with Kimia216. To study the algorithm performance, all of the 216 instances were used as target inputs for the object recognition system explained in Section 4.1. The most similar object is retrieved and its class becomes an answer. A recognition of a target shape will be considered as correct if the model with the highest accuracy has the same class as itself. The Contour-SIFT parameters are set-up according to Table 4.1.

Table 4.1: Signature experiment's parameters for Kimia216.

Parameter	Configuration
No. of octaves	4
No. of scales per octave	3
No. of elements in contour	122
σ_0	0.12
Neighborhood r	5
Matched threshold d	10

4.3.3 Result and discussion

Table 4.2 shows accuracies of the object recognition system. The result of the curvature signature is the highest with an accuracy of 54.63%. Followed by 38.42% and 29.17% of DTA and CCD, respectively. The result indicates that, with the given parameters, curvature function should be used.

The reason for these occurrences is that most of Kimia216 shapes are curvy and have a lot of variations. CCD has the problem with finding centroid of an object. For class number two, four, seven, 11 and 13, centroids location greatly changes from instance to instance. As a result, the keypoints' positions detected by the proposed method are not stable. As for DTA signature, which performs a better job than CCD, it is able to detect keypoints at acute corners, but fails to detect keypoints from curvy parts of contours. Therefore, its performance is lower than that of curvature signature. However, it is important to note that, with the different choices of parameters or datasets, the results can be different.

Table 4.2: Recognition rates of different signatures

Signature function	Accuracy(%)
curvature	54.63
CCD	29.17
DTA	38.42

4.4 Experiment 2: Image rotation invariance verification

4.4.1 Objective

The proposed algorithm should be invariant to image translation, rotation, and scaling. The translation was verified by instances in datasets, which include objects of different places. For the rotaion and image-scaling invariance, the current and the next experiments were conducted to verify.

4.4.2 Experiment set-up

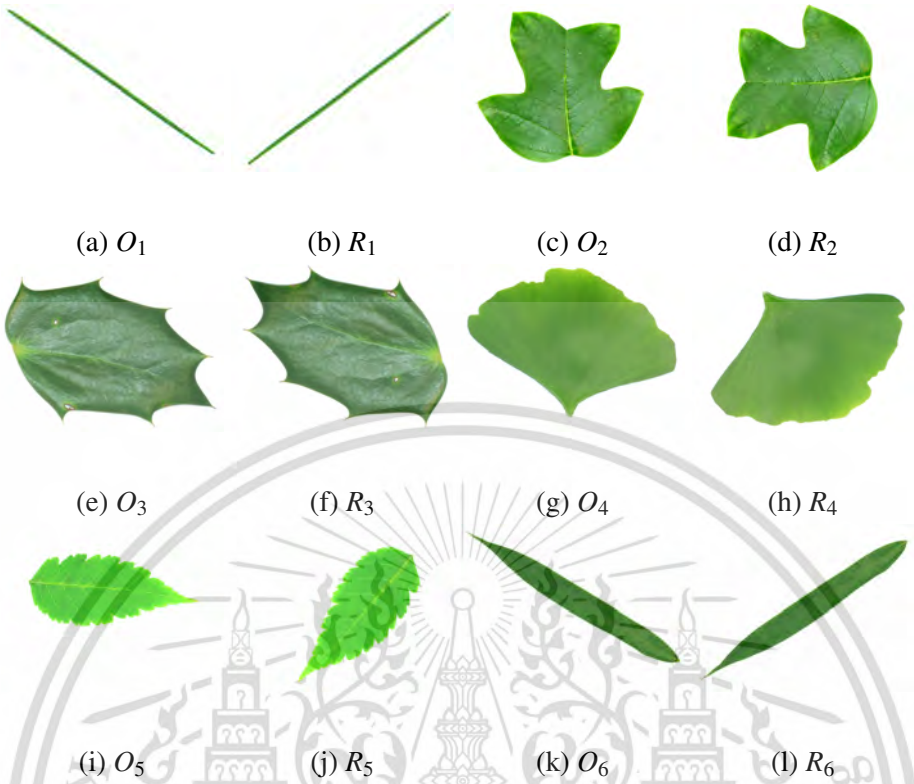
To verify the rotation invariant, six images were selected from Flavia200. The leaf in each image is randomly rotated. Then, the rotated versions are compared with the original version with the matching algorithm. Let R_i be a rotated leaf image of specie i and O_i be its original version. The similarity between R_i and O_i should approach the similarity between O_i and O_i. The correspondance between them can be calculated into percentage with $Similarity(R_i, O_i)/Similarity(O_i, O_i) \times 100$

Fig. 4.7 shows the selected leaf images and their rotated version.

4.4.3 Result and discussion

The experimental result in Section 4.4.3 reports that Contour-SIFT is invariant to image rotation to a good extent. Although the score is lower than 80% for image number five, the other classes have scored above 85%. In Fig. 4.8, Contour-SIFT keypoints detected in the first octave for each input images are shown.

By observation, the similarity score are higher for shapes with less edge teeth. Image number one and image number four has less teeth; hence, they achieve higher scores from the comparison. The reason for this is because when keypoints are matched. The



(a) O_1 (b) R_1 (c) O_2 (d) R_2

(e) O_3 (f) R_3 (g) O_4 (h) R_4

(i) O_5 (j) R_5 (k) O_6 (l) R_6

Figure 4.7: Input dataset for rotation invariant verification.

proposed matching algorithm start the analysis with keypoints that has high signal magnitude, in absolute terms. Therefore, there are higher chances for a teeth to be mismatched to peaks or other teeth instead of the correct ones, which in turn eliminates possible result (that could be correct) for the next keypoint to be considered.

4.5 Experiment 3: Image scaling invariance verification

4.5.1 Objective

Three leaf recognition experiments were done on Flavia200 to verify the scale invariant property of Contour-SIFT algorithm.

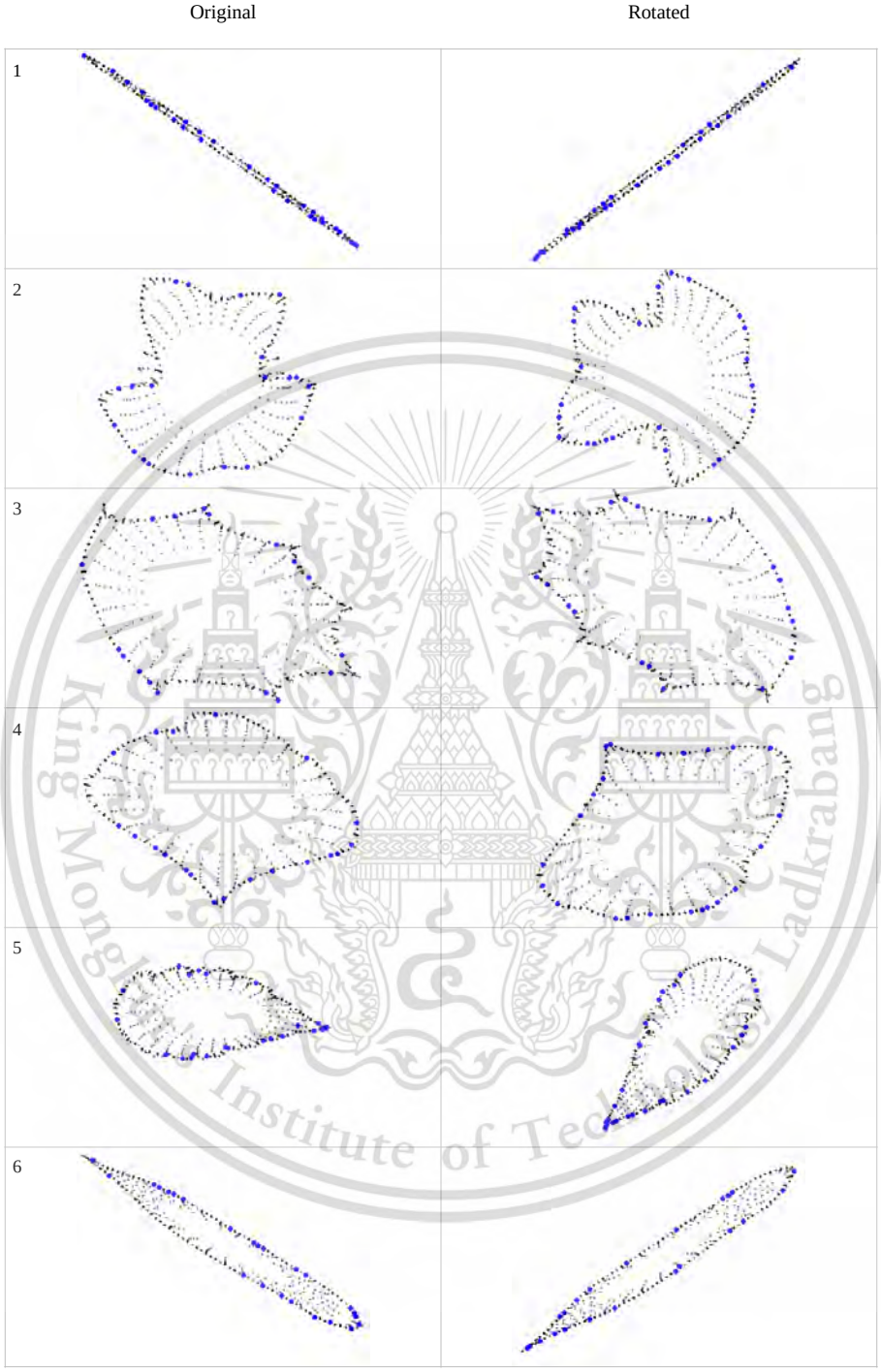


Figure 4.8: Keypoints detected in O_i and R_i of the selected images

Table 4.3: Result of rotation invariant experiment.

Specie	Similarity(%)
1	92.22
2	87.43
3	88.75
4	91.23
5	76.33
6	86.91

4.5.2 Experiment set-up

With the same configuration parameters (see Table 4.4), one experiment was done with the original image size, which is 1280×960, and the other two experiments were done with scaling parameters of 2 and 1.5, resulting in 640×480 and 853×640, respectively.

Table 4.4: Contour-SIFT parameter used in invariance properties verification

Parameter	Configuration
No. of scales per octave	4
No. of octaves	3
No. of elements in contour	1000
σ_0	3.6
Signature function	Curvature
r	20
d	32
Matched threshold	10

4.5.3 Result and discussion

The experimental results are shown in Table 4.5. The accuracies obtained from the three different scales are similar. Thus, image scaling invariant property has been

verified.

Table 4.5: Experimental results of leaf shape recognition with different image scales

Image size	Accuracy (%)
1280 ×960	45
853 × 640	44
640 × 480	44.5

4.6 Experiment 4: Occlusion invariance verification

4.6.1 Objective

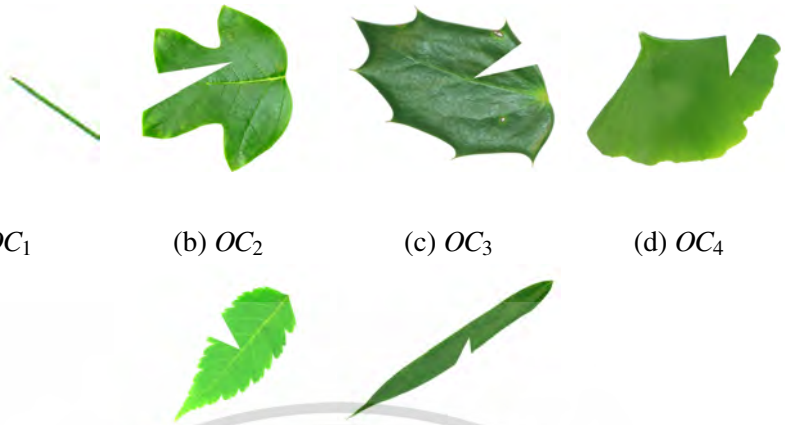
In addition to rotation, translation, and scaling invariance, occlusion invariance is also verified. This experiment was done to check image occlusion invariant property of Contour-SIFT algorithm.

4.6.2 Experiment set-up

The sample images used in the experiment Section 4.4 were tampered again by using PhotoShop, resulting in the following images (Fig. 4.9). These images are considered as containing simple occlusion. The figure shows the altered images that were used as input of the object recognition system. OC_i is R_i with simple occlusion. Each pair of occluded and rotated image is compared using the same method as the previous experiment. The result should approach 100%, to verity the occlusion invariant property.

4.6.3 Result and discussion

None of the similarity in Section 4.6.3 exceed 70% with two of are lower than 50%. This result indicates that the proposed Contour-SIFT descriptor is not steady when occlusion exists in the system. Even a simple occlusion reduces the similarity value by far.



(a) OC_1 (b) OC_2 (c) OC_3 (d) OC_4

(e) OC_5 (f) OC_6

Figure 4.9: Input dataset for occlusion invariant verification.

Table 4.6: Result of occlusion invariant experiment.

Specie	Similarity(%)
1	55.00
2	65.62
3	40.00
4	55.59
5	67.32
6	46.46

Fig. 4.10 shows the causes of the result. A large number of keypoints are detected at the opening parts. They are the reason for the incorrect matches of keypoints. Especially, the keypoints that are located at peaks of those parts. With their high signal magnitude (i.e., in this experiment, it is curvature), they are chosen for matching before the proper ones. In consequence, chances for correct matching of the true keypoints are eliminated.

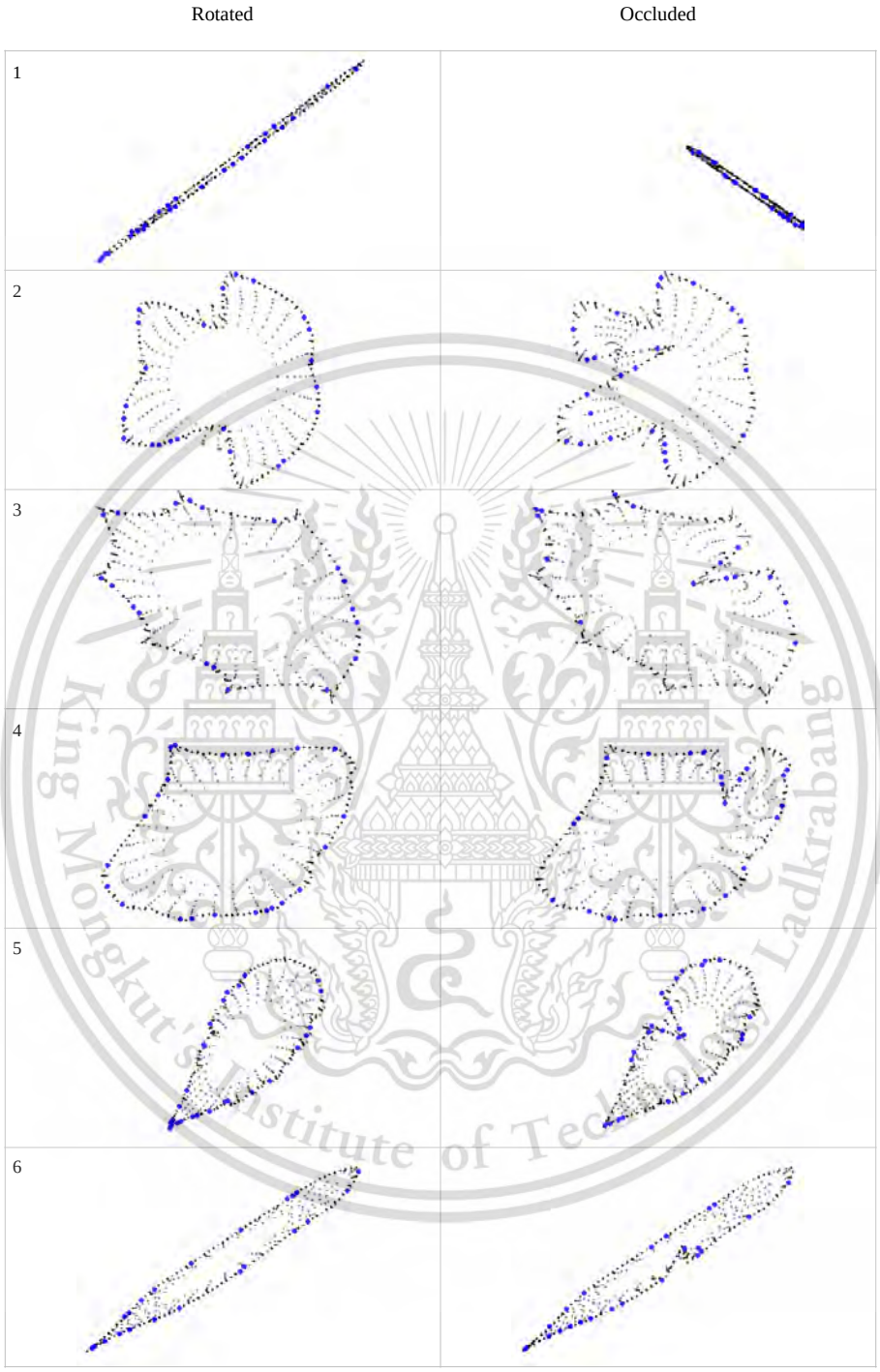


Figure 4.10: Keypoints detected in R_i and OC_i of the selected images

4.7 Experiment 6: Comparing with other algorithms

4.7.1 Objective

To verify the performance of Contour-SIFT algorithm. It should be compared with others methods from literatures.

4.7.2 Experiment setup

The same object recognition system is used with the different choices of shape feature extraction and matching method for image retrievals. Contour-SIFT parameters are set-up according to Table 4.1 and curvature signature is chosen. In this experiment, Kimia216 is chosen as the experimental dataset. The number of retrieved images is 12. It is chosen based on the number of instances in a class. A recognition of a target shape will be considered as correct if one of the retrieved model has the same class as itself. All of the instances in the datasets are used as targets.

The performance of the proposed shape descriptor is compared with other methods in literatures ([18], [13]). The compared methods include Contour Saliences Descriptor (CS), Fourier, Moment Invariants (MI), Shape Saliences (SSD), and Curve Normalization (CN).

4.7.3 Result and discussion

The experimental results are presented below. Table 4.7 shows that the retrieval accuracy of Contour-SIFT is 72%, which is the best, comparing with the other shape features. The retrieval per class is reported in Fig. 4.11. Rate of class *i* belongs to target images of row *i* in Fig. 4.5.

Table 4.7: Results of leaf recognition using different shape features

Shape descriptor	Retrieval rate
CS [13]	0.36
Fourier [13]	0.37
MI [13]	0.40
SSD [13]	0.72
CN [13]	0.64
Contour-SIFT	0.77

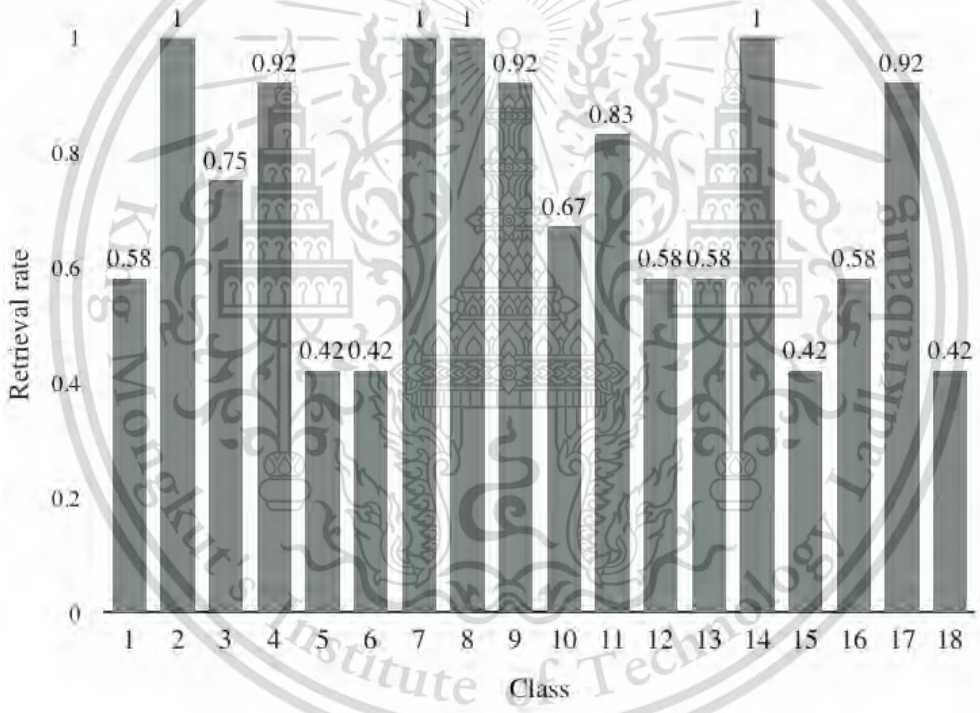


Figure 4.11: Retrieval rate for each class of Kimia216.

The system correctly retrieved images of class 2, 7, 8, and 14 with 100% accuracy, as well as retrieved images of class 4, 9, and 17 with 92% accuracy. In contrast, for class 5, 6, 15, and 18, the rate of retrieval are only 0.42. These indicate that Contour-SIFT

method is good at recognizing unique shapes, but not good with common ones. Also, if there are a lot of variasion in shapes of a class (e.g., class 18 and 5), then the algorithm will not be able to recognize its identity.

Chapter 5

Summary

The goal of this thesis is the contour-based shape descriptor that globally captures local characteristics and has scaling, translation, and rotation invariant properties. For this purpose, Contour-SIFT firstly extracts keypoints from different scales of a shape contour signature. Second, the keypoints are described with frequencies of curvature histograms that capture local feature at each location. Finally, the representations are compared by using a matching algorithm that computes similarity between two Contour-SIFT descriptors. With the matching method, the Contour-SIFT descriptor extracted using the proposed algorithm can be used for shape recognition and image retrieval.

Experiments were done on two public datasets, Flavia and Kimia216. They results indicate that the proposed algorithm is effective. The results of experiment 2 (Section 4.4) and experiment 3 (Section 4.5) also prove the invariance properties of the algorithm. Based on the result in Table 4.7, the performance of Contour-SIFT algorithm is above the other five algorithms including Contour Saliences Descriptor (CS), Fourier, Moment Invariants (MI), Shape Saliences (SSD), and Curve Normalization (CN).

Bibliography

[1] S. Abbasi, F. Mokhtarian, and J. Kittler, "Curvature scale space image in shape similarity retrieval," Multimedia Systems, vol.7, pp.467–476, 1999.

[2] G. Cerutti, L. Tougne, D. Coquin, and A. Vacavant, "Curvature-scale-based contour understanding for leaf margin shape recognition and species identification," Proceedings of 2013 International Conference on Computer Vision Theory and Applications, pp.277–284, 2013.

[3] K. Dowson-Howe, A Practical Introduction to Computer Vision with OpenCV, Wiley, 2014.

[4] J.X. Du, X.F. Wang, and G.J Zhang, Leaf shape based plant species recognition, Applied Mathematics and Computation, vol.185, pp.883–893, 2007.

[5] B. Ellis, D.C. Daly, L.J. Hickey, K.R. Johnson, J.D. Mitchell et al., Manual of Leaf Architecture, Cornell University Press, 2009.

[6] R.B. Fisher, T.P. Breckon, K. Dawson-Howe, A. Fitzgibbon, C. Robertson et al., Dictionary of Computer Vision and Image Processing (2nd Edition), Wiley, 2014.

[7] R.C. Gonzalez and R.E. Woods, Digital Image Processing (3rd Edition), Pearson Education, 2010.

[8] G.L. Grinblat, L.C. Uzal, M.G. Larese, and P.M. Granitto, "Deep learning for plant identification using vein morphological patterns," Computers and Electronics in Agriculture, vol.127, pp.418–424, 2016.

[9] C. Harris and M. Stephen, "A combined corner and edge detector," Proceedings of 1988 the 4th Alvey vision Conference, pp.147–151, 1988.

[10] D. Huda and A.I. Hussein, "Circle views signature: a novel shape representation for shape recognition and retrieval," Canadian Journal of Electrical and Computer Engineering, vol.39, pp.274–282, 2016.

[11] N. Hussin, N. Jamil, S. Nordin, and K. Awang, "Plant species identification by using scale invariant feature transform (SIFT) and grid based colour moment (GBCM)," Proceedings of 2013 Conference on Open Systems, 2013.

[12] A.R. Jimenez, R. Ceres, and J.L. Pons, "A vision system based on a laser range-finder applied to robotic fruit harvesting," Machine Vision and Applications, vol.11, pp.321–329, 2000.

[13] N. Laiche, S. Larabi, F. Ladraa, and A. Khadraoui, "Curve Normalization for Shape Retrieval," Signal Processing-Image Communication, 2014.

[14] P. Li, S. Lee, and H.Y. Hsu, "Review on fruit harvesting method for potential use of automatic fruit harvesting systems," Proceedings of Procedia Engineering, vol. 23, pp.351–366, 2011.

[15] D.G. Lowe, "Distinctive image features from scale–invariant keypoints," Proceedings of 2004 International Journal of Computer Vision, vol.60, pp.91–110, 2004.

[16] A. McAndrew, An Introduction to Digital Image Processing with MATLAB, Course Technology, 2004.

[17] F. Mokhtarian and R. Suomela, "Curvature scale space based image corner detection," Proceedings of 1998 Signal Processing Conference, 1998.

[18] G.V. Pedrosa, M.A. Batista, C.A.Z. Barcelos, "Image feature descriptor based on shape salience points," Neurocomputing, vol.120, pp.156–163, 2013.

[19] M. Rojanamontien, P. Sihanatkathakul, N. Piemkaroonwong, S. Kamales, and U. Watchareeruetai, "Leaf identification using apical and basal features," Proceedings of 2016 the 8th International Conference on Knowledge and Smart Technology (KST), pp.234–238, 2016.

[20] T.B. Sebastian, P.N. Klein, B.B. Kimia, "Recognition of shapes by editing their shock graphs," IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.26, pp.550–571, 2004.

[21] K. Siddiqi et al., "Shock graphs and shape matching," International Journal of Computer Vision. vol.35(1), pp.13-32, 1999.

[22] Y. Song et al, "Automatic fruit recognition and counting from multiple images," Biosystems Engineering, vol.118, pp.203–215, 2014.

[23] M. Sonka, V. Hlavac, and R. Boyle, Image Processing , Analysis, and Machine vision (4th Edition), Cengage, 2015.

[24] J. Stewart, Calculus (6th Edition), Thomson Brooks/Cole, 2009.

[25] K.L. Tan and L.F. Thiang, "Retrieving similar shapes effectively and efficiently," Multimedia Tools and Applications, vol.19, pp.111–134, 2003

[26] B. VijayaLakshmi and V. Mohan, "Kernel–based PSO and FRVM: An automatic plant leaf type detection using texture, shape, and color features," Computers and Electronics in Agriculture, vol.125, pp.99–112, 2016.

[27] S.G. Wu, F.S. Bao, and E.Y. Xu, "A leaf recognition algorithm for plant classification using probabilistic neural network," Proceedings of 2007 IEEE Symposium on Signal Processing and Information Technology (ISSPIT), pp.11–16, 2007.

[28] C.T. Zahn and R.Z. Roskies, "Fourier descriptors for plane closed curves," IEEE Transactions on Computers, vol.C-21, pp.269–281, 1972.

[29] D. Zhang and G. Lu, "A comparative study of curvature scale space and Fourier descriptors for shape-based image retrieval," Journal of Visual Communication and Image Representation, vol.14, pp.39–57, 2003.

[30] L. Zhang, P. Weckler, N. Wang, D. Xiao, and X. Chai, "Individual leaf identification from horticultural crop images based on the leaf skeleton," Computers and Electronics in Agriculture, vol.127, pp.184–196, 2016.

[31] C. Zhao, S.S.F, Chan, W. Cham and L.M. Chu, "Plant identification using leaf shapes-A pattern counting approach," Pattern Recognition, vol.48, pp.3203–3215, 2015.