Master Thesis
Software Engineering
Thesis no: MSE-2006:09
July 2006

Classification of objects in images based on various object representations

Radoslaw Cichocki

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Radoslaw Cichocki
E-mail: radoslaw.cichocki (at) gmail.com

External advisor:
prof. Halina Kwasnicka
Address:
Wroclaw University of Technology
Institute of Applied Informatics
Wybrzeze Wyspianskiego 27
(Skwer Idaszweskiego 1, budynek D2)
50-370 Wroclaw
Phone:
+48 77 320 23 97

University advisor:
dr. Ludwik Kuzniarz
School of Engineering

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : +46 457 271 25

ABSTRACT

Object recognition is a hugely researched domain that employs methods derived from mathematics, physics and biology. This thesis combines the approaches for object classification that base on two features – color and shape. Color is represented by color histograms and shape by skeletal graphs. Four hybrids are proposed which combine those approaches in different manners and the hybrids are then tested to find out which of them gives best results.

Keywords: multiple cue object recognition, color histogram, skeletal graph

CONTENTS

1 INTRODUCTION

Humans perceive the surrounding world primarily by the sense of vision which has developed for thousands of years. The ease of recognizing faces which characterizes even infants is a result of long evolution not only of the eyes but also the brain, in particular its part called the *primary visual cortex*. Infants are capable of recognizing faces, bees can locate flowers in a meadow but computers still struggle to equal the nature in accomplishing alike tasks. This seemingly ordinary and common ability is one of the most desired natural organisms' features to be transferred to computers. Machine vision, also referred to as computer vision, in its current shape has already many applications but it also creates new possibilities that were earlier out of reach. At present, among many more uses, it is employed for: analyzing medical images; finding defective products in production lines by analyzing input from cameras; finding wanted terrorists in crowds by examining faces; recognizing handwritten text; opening gates at parking lots for cars with given license plate numbers; locating specified objects in satellite images; and many more [3]. This thesis aims to mimic a natural vision system by combining two different methods of machine vision. It is realized by implementing an *object recognition* system and using the implementation for an experiment conducted on a set of images.

Aims and objectives of this thesis

The task of object recognition (OR) concerns, in the simplest case, stating whether an object of a specific class is present in an image or not, e.g. flames or smoke could be searched in satellite photographs of forests. In general, however, OR comprises more complicated tasks including stating what objects or categories of objects are present in an image and indicating the position of particular objects.

The aforementioned tasks are accomplished best when taking multiple *features* (refer to appendix A) of objects into consideration, e.g. shape, color, size, relative position of their components. Analyzing various object features requires the researcher to design some representations of these features, i.e. the color will be represented in a different way than the shape; the location could be expressed as a pair (x, y) of coordinates within the image whereas the size could be a ratio of the object height or width to the same characteristics of the whole image and therefore be expressed as just one number. Having clarified the problem, the general aim and objectives of the thesis, research questions and expected outcomes can be stated.

Aim:
Examine what ways of combining object recognition methods give best results.

Objectives:
1. Implement at least two object recognition methods.
2. Combine the methods in at least two ways.
3. Evaluate the combinations from different perspectives.

Research questions:
1. What object representations should be chosen?
2. How to combine the object recognition methods?
3. How are one ways of combining the methods better than others?

Expected outcomes:
1. A nature-inspired object recognition system.
2. The relationship between the combinations of object recognition methods and the effectiveness of the recognition process.

The research methodology for fulfilling the aim of the thesis will be experiment-based, i.e. the research question (3) will be answered quantitatively by conducting an experiment while research questions (1) and (2) will be answered qualitatively by reasoning and argumentation.

Structure of the thesis

Chapter 1 provides a gentle introduction to the field of computer vision with its biological roots. It states what the aims and objectives are for this thesis and poses the research questions which are answered in later chapters. A list of expected outcomes is also specified.

In chapter 2 the current state of the art is described based on literature review. Section 2.1 introduces the currently widely used overall approach to the object recognition task and gives the basic definitions of the terms used throughout the thesis. Then it explains what object features are selected for the performed research. Section 2.2 describes how color is currently represented to enable computers to use it in the object recognition process. Section 2.3 gives a similar explanation and overview of the methods that are presently used for 2D and 3D object recognition with the use of shape.

Chapter 3 explains the approach proposed by the author. It describes feature representations, algorithms and proposed ways of combining the object recognition methods. In section 3.1 the applied histogram representation is briefly presented, while section 3.2 constitutes more complex depiction of the graph representation and relevant algorithms employed. Section 3.3 presents the modelbase along with the division of the objects into classes. It is also the section where the ways of combining the object recognition methods are proposed and the application implemented for the experiment is depicted.

Chapter 4 defines the experiments that were conducted and gathers the results obtained. For every experiment a discussion of the results follows. Section 4.1 lists the parameters of the system built for the thesis and describes the experiment that was conducted. In section 4.2 the results of the parameter tests are gathered and discussed.

Finally, chapter 5 gives a summary of the thesis, highlights the achievements, proposes improvements to be done and sets directions for further research. Applications of the work are also described.

2 OBJECT RECOGNITION – CURRENT STATE OF THE ART

This chapter presents current approaches for the object recognition process. It gives definitions of the basic terms used throughout the thesis and describes the object representations which are applied for representing object features as well as algorithms for computing and matching them.

2.1 The overall approach

A generic approach for object recognition was proposed in [14]. Virtually every OR system can be described in terms of the model presented in the mentioned work. The model is depicted in Figure 1.

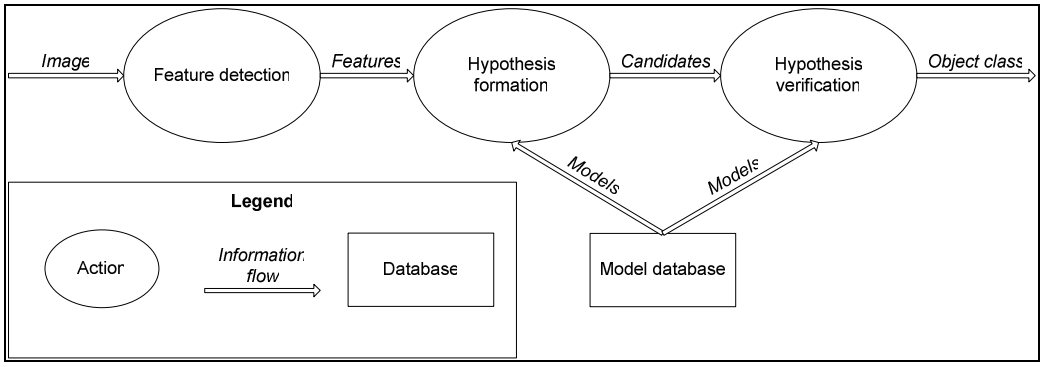


Figure 1: Model of an OR system (adapted from [14], page 1 of Chapter 8)

The model describes the steps of the OR process. However, before explaining the process it is necessary to define the terms which will be used throughout the thesis.

Image – a two dimensional array of pixels. The pixels within one image are encoded using the same color model.

Color model – a way of representing color. Color can be described by a set of numbers, e.g. the RGB (Red, Green, Blue) model represents color as a combination of three basic colors; the HSV model describes color by its hue, saturation and value [3][8][14].

Class – a category which describes a group of objects that have a set of common features. The term 'class' and 'category' are used interchangeably throughout this thesis.

Object – an element which can be assigned to a class. Every object is characterized by a set of features.

Feature – a characteristic of an object which can be used to distinguish an object from other objects [14]. Similarities in features allow for assigning objects to classes.

Having defined these terms an OR system can now be described. A system of this kind always takes an image as input. The image is then processed by feature extraction procedures, as depicted in Figure 1. As a result, features which characterize the object present in the image are extracted. Some examples of features which can be useful for OR are color, shape, size, position in relation to other objects, etc. The features have to be expressed in a form appropriate for further processing, i.e. form which allows for

making comparisons within the same feature among different objects. The extracted features are then used to indicate the most suitable category for the object. This step is called hypothesis formation. The system matches the features up with the features of objects which are known, i.e. which are present in the model database. The model database (also called modelbase) contains the objects known for the OR system, that are called models. Hypothesis formation finishes when the known objects are marked with likelihoods of matching the object in the image, based on the formerly extracted features – this limits the solution space if a threshold of likelihood is set which qualifies only a group of classes to be further matched. Next the hypothesis is verified in the hypothesis verification step. This involves applying more sophisticated and resource-demanding methods. Whichever methods are employed, they also use the model database. The probabilities assigned to the candidates are refined this way and the system selects the class with the greatest likelihood to pass it to the output. The process description was adapted from [14].

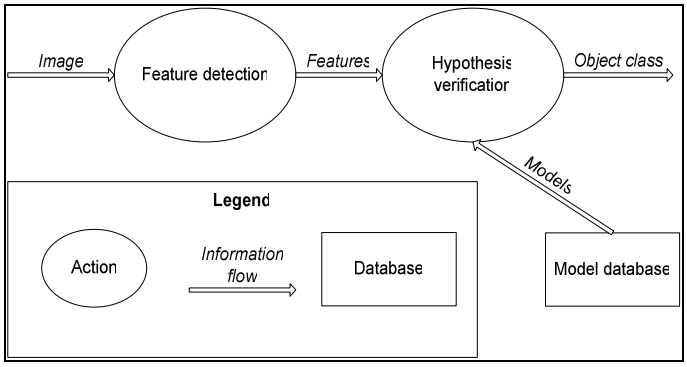


Figure 2: An OR system without hypothesis formation

The step of hypothesis formation is optional and can be omitted, as shown in Figure 2. In such case the features are passed directly to the hypothesis verifier that performs a full match on every object in the modelbase using the given features. This negatively influences the performance of an OR system but is acceptable when the modelbase is relatively small, that is when it is affordable to perform a full match procedure on every object in the modelbase.

Every step in the model is generic, meaning it can be realized in various ways. Hypothesis verification, for example, is pure classification. Therefore any classifier, either a heuristic or an algorithm, can be used in this step to assign a class to the input object (represented by features). The choice of features and their representation is one of the key factors which have to be discussed in order to proceed.

Choice of features

As promptly mentioned before, the features which visually describe objects in reality are: color, shape, scale, position in relation to other objects, etc. The most valuable feature for object recognition is its *shape*, which is defined for 2D space in [19] as 'the bounded interior of a simple closed (Jordan) curve'. This definition does not let a shape to have holes in it and the term 'shape' will be understood in compliance with this definition in the thesis. Humans can recognize items in gray-scale images, where the color is removed. Even placing an object on a uniform background with no other objects in the image does not prevent an individual from giving the right answer, although such an image contains no information about the scale of the object or any other relation to other objects. It is therefore clear that the outline and topology, which constitute the shape, are of paramount importance for proper object recognition.

The second most important feature is color which is a perception of the wavelength of light reflected from the surface of an object. Color is frequently the first

characteristic used when looking for an object in the surrounding. Let us consider a situation where an individual is asked to find a lemon at a grocery store. The first action they take is not to look for an oval shape but to focus their sight on objects that are yellow. Only then are they trying to determine whether the object is oval or not. Hence, despite that from OR perspective the shape carries the most valuable and unique information, the color is also a high-value feature. The above conclusions cause that in this thesis the features which will be used for the OR process are shape and color. The rest of the chapter is devoted to the representations of these features which are currently used in the field of computer vision.

2.2 Representing and matching objects by color

Colors are represented in certain color models, however in order to describe the whole object (rather than just one pixel) by its color a more sophisticated approach has to be used. Swain and Ballard [20] proposed a technique called color indexing. Color indexing has its roots in mathematical statistics and uses histograms of colors to represent objects. A histogram is a way of representing the distribution of a feature. It can be understood as an array of frequencies of samples in a given population, where the array is indexed with the samples and the value at a specified index is the number of occurrences of this particular sample (Figure 3). This means that for a given image of size $n \times m$ the sum of the values in the histogram will also be equal to $n \times m$, that is the number of pixels (samples).

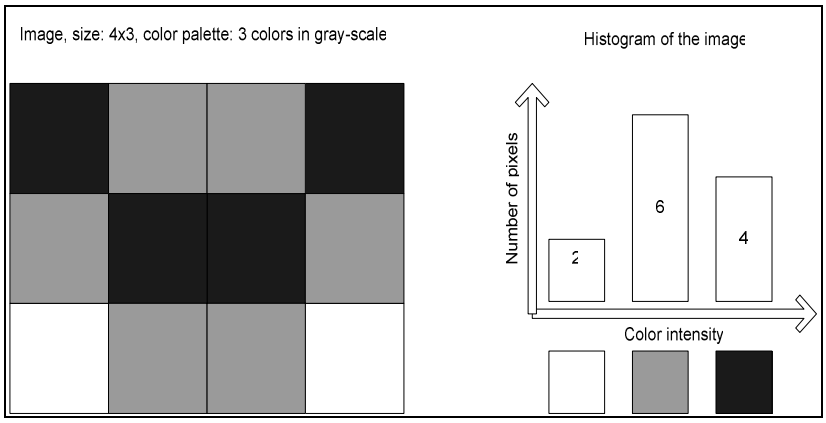


Figure 3: A sample image and its histogram

Simply counting particular pixel occurrences entirely neglects any shape information. This fact has its positive and negative implications. In Figure 4 images (1) through (4) depict the same object which is rotated by 90 degrees in every image. The histogram of the object is the same for every angle: 0, 90, 180 and 270. This is a very desirable feature (called rotation invariance) because it allows for minimizing the modelbase size – it is only required to store the histogram of the object acquired at one position and this will suffice for recognizing the object regardless of its rotation angle in the input image. The disadvantage, on the other hand, is ironically caused by the same feature. When the shape changes, as in image (5) in the same figure, the histogram does not change, although the shape was modified and the image now depicts a different object. In practice, the larger the size of the image, the less likely it is that there will occur a very similar histogram but of a different object. By enlarging the size of the image the number of samples (pixels) is increased causing the histogram to be more representative, that is – more probable to be unique.

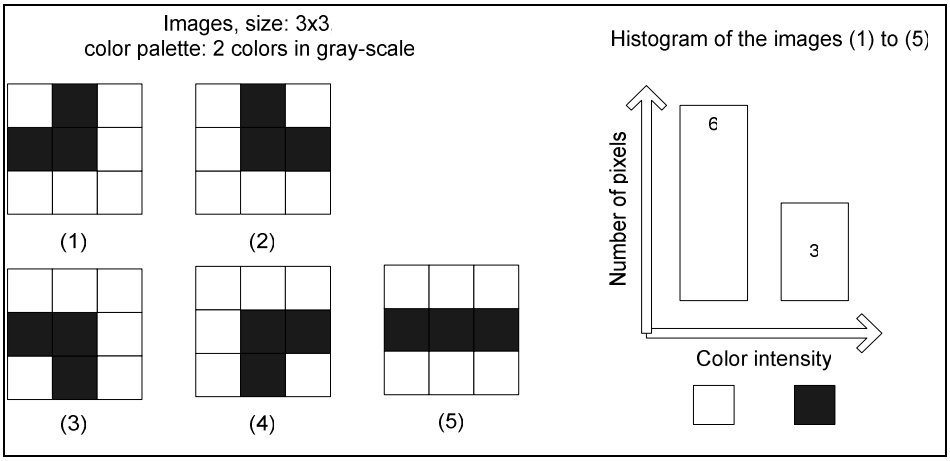


Figure 4: Rotation invariance and its negative implications

The histograms in Figure 4 were calculated for gray-scale images purely for explanatory purposes. Histograms are calculated to represent information about color which is always encoded in a color model. Let us consider an RGB histogram. Every pixel is described by three integers from the range of 0 – 255. This means that either three histograms will be built for one image (a separate histogram for every basic color) or just one three dimensional histogram. The first approach leads to significant information loss and therefore will not be used in this thesis. In particular, separation of the basic colors into three independent histograms renders the representation ambiguous. This situation is depicted in Figure 5.

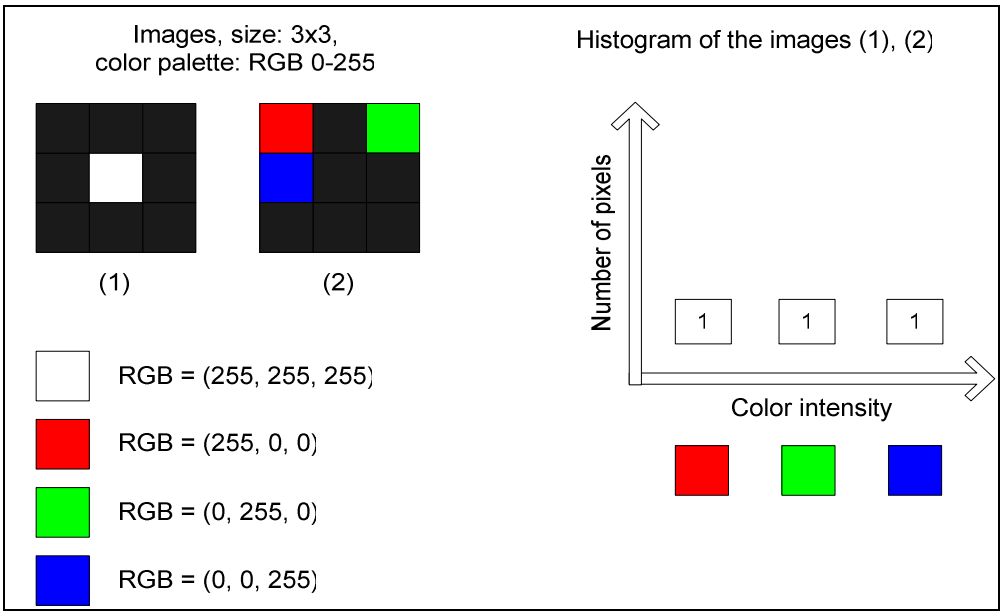


Figure 5: Histogram ambiguity

The histograms in Figure 5 were intentionally drawn as one, not three histograms, for brevity and to convey the idea in a clear manner. Both images (1) and (2) have the same histograms when their basic colors are separated. This stems from the fact that in order to obtain the largest intensity of the white color, maximum values have to be assigned to all the component colors: red, green and blue, as in image (1). Image (2) contains separate pixels with maximum values set just for one component color and minimum values for the two remaining colors. This produces the same three histograms as for the image containing one white pixel. The ambiguity can be evaded by using a histogram of three joined values, i.e. a three dimensional histogram which is indexed by a value in the form of (R, G, B). Furthermore, to obtain a more compact

histogram, the small bins are gathered into ranges which represent similar colors. A pixel in a gray-scale has 256 possible shades of grey, what normally requires a histogram to have 256 bins to store the number of occurrences for every shade. Such approach results in a large number of insignificant zeroes (no occurrences) and therefore the 256 values are often split into ranges [7][20], e.g. 16 bins representing shades of values 0..15 (bin 1), 16..32 (bin 2), …, 239..255 (bin 16). Additionally it is advised to use the HSV color model instead of RGB when computing histograms however Swain and Ballard [20] have worked out their own color space which they use for producing histograms. In this thesis the RGB model will be used, as it gives satisfactory results for simple object recognition tasks.

Another weakness of color indexing is its sensitivity to changes in the lighting of the scene [7]. When the lighting of the object changes it modifies the color histogram of the image and therefore can lead to misclassification of the object from the model stored in the modelbase, as the model was computed under other lighting conditions. One simplistic solution to this problem is not to use the pixel values themselves as histogram indices, but make use of color differences between neighboring pixels instead, since the neighboring pixels have most likely changed by the same factor when the light became stronger or weaker. More sophisticated methods for handling illumination sensitivity are known to date, such as 'color constant color indexing' [9]. For more techniques see [7].

Finally when the histograms are computed they have to be compared in order to find the best matching model for a query histogram. A variety of measures for histogram similarity were proposed [9][10][13]. A few are mentioned below, assuming that h and g denote two color histograms.

1. Histogram Euclidean distance:

$$d^2(h,g) = \sum_A \sum_B \sum_C (h(a,b,c) - g(a,b,c))^2 .$$

2. Histogram intersection distance:

$$d(h,g) = \frac{\sum_A \sum_B \sum_C \min(h(a,b,c), g(a,b,c))}{\min(|h|,|g|)},$$

where $|h|$ and $|g|$ are the numbers of samples in histograms h and g, respectively. This measure has the advantage of not letting the colors absent from the query histogram contribute to the intersection distance what has a positive effect of ignoring the background.

3. Histogram quadratic distance

$$d(h,g) = (h-g)^t A (h-g),$$

where A is a similarity matrix, which *(i, j)*th element is given by:

$$a_{ij} = \frac{1 - d_{ij}}{\max(d_{ij})},$$

for RGB space and d_{ij} is the distance between colors i and j [10].

The measures of histogram similarity mentioned above constitute metrics which can be easily used for finding the model in the modelbase that best matches the query histogram provided as the input for the OR system.

2.3 Representing and matching objects by shape

Shape of an object is more complex than its color characteristics, which results in less straightforward representations. The issue becomes even more complicated when 3D shapes are considered.

To begin with 2D shapes, a variety of representations were proposed and are widely used. One approach is to represent the shape as the outline of the object, i.e. a closed curve which cuts the space in two regions [18], where every point in the space belongs either to the inside or outside of the object. The curves can be matched by employing appropriate metrics for the purpose, e.g. Frenet distance metrics [1]. Another approaches make use of other features of the shape, like the histogram of distances from the center of weight of the shape [11]. Perhaps the most versatile representation of a 2D shape is a graph. Graphs are built in a number of ways, one place their vertices in characteristic points of the outline [14], others are constructed to represent the topology of parts of the object. For 3D shapes more sophisticated methods are applied, among which aspect graphs [17] are worth mentioning. An aspect graph represents a 3D object as a set of 2D views of the object from different viewing angles. Every view constitutes a vertex and edges connect the vertices that differ the least in terms of 2D shape similarity. An exemplary aspect graph is shown in Figure 6.

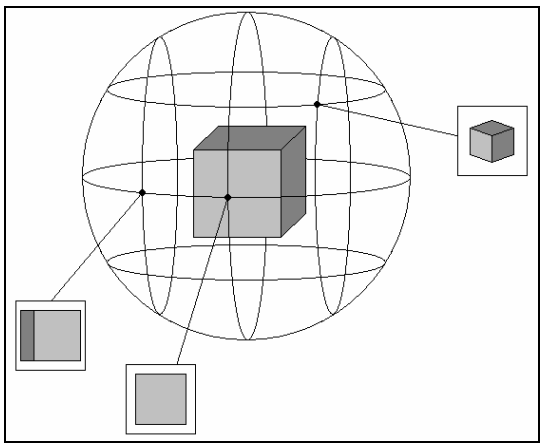


Figure 6: An aspect graph of a cube

As the thesis regards only 2D shapes, the rest of the section focuses on a particular 2D shape representation called a ‘shock graph’ [15][16][19] which was modified to be the representation of shape in this thesis. Before the definition of a shock graph is given, the term *skeleton* has to be explained first since it forms a basis for the shock graph.

A skeleton of a shape was first described by Blum in [2]. Blum defines the skeleton as the set of centers of the largest circles inscribed in the shape. These circles are at most cases tangent to the shape at two points. An example of a shape and its skeleton is shown in Figure 7.

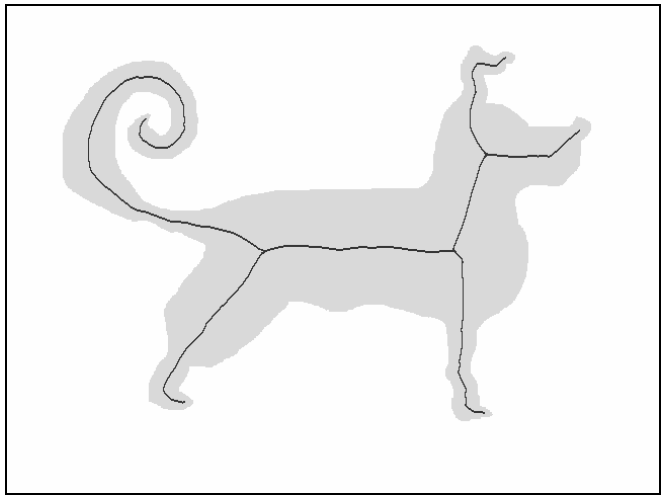


Figure 7: A shape and its skeleton

Another way to define the skeleton, according to Blum, is to imagine that the shape is made from uniform, dry grass. When all point on the boundary of the shape are simultaneously set on fire, they start burning causing the shape to shrink. When the fire fronts from opposing sides meet, they extinguish at the points which form the skeleton. This way some points are formed earlier than others – in the case of the dog in Figure 7 the skeletal points at the ear of the dog would be formed relatively fast and the points at the chest, late. The time of formation corresponds to the radii of the circles centered at the skeletal points. The grater the time of formation, the grater the radius of the inscribed circle which simply stands for the distance from the skeletal point to the closest point on the shape's outline.

Skeletons represent the relations of parts of the object well, however they are somewhat sensitive to even slight changes of the shape. This has significant meaning in real-life applications, where noise is a frequent issue. The approach that currently gives the best results for computing stable skeletons was proposed by Dimitrov *et al.* in [4]. The algorithm is explained below.

Dimitrov's *et al.* algorithm for calculating the skeleton of a shape:

The algorithm for obtaining the skeleton takes as input an image which contains the shape. An example of an input image is presented in Figure 8.

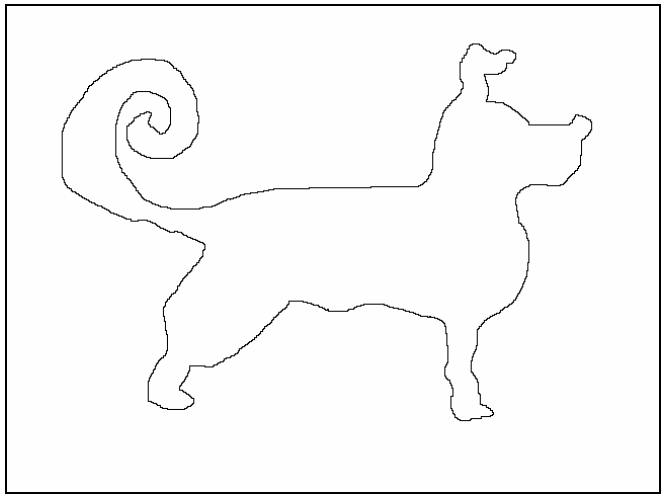


Figure 8: An input image for Dimitrov's *et al.* algorithm

Next a signed distance transform of the object has to be computed. This transform computes for every point of the image which lies outside the shape, the distance of that point to the nearest point of the shape. The points which form the shape are assigned with zeroes. For the points located in the interior of the shape, signed distance transform assigns the negative distance from the closest point of the shape.

Apart from signed distance transforms, unsigned distance transforms exist which differ from their signed counterparts in that they assign only the positive distance from the shape points, either for the point located outside or inside the shape. The original Dimitrov's *et al.* algorithm uses a signed distance transform, however it was modified for the purpose of this thesis to use an unsigned distance transform (see section 3.2), in particular the Euclidean distance transform (EDT) described in [6]. Therefore an example of an unsigned distance transform (EDT), is given in Figure 9. The brightness of the pixels changes with their distance from the shape – the brighter the pixel, the farther it is located from the shape. For a signed distance transform the pixels inside the shape would get darker as their distance from the shape would increase. The pseudocode for EDT can be found in [6].

Figure 9: Euclidean distance transform of a shape

The above image can be treated as a function of two variables $f(x,y)=d$ where (x,y) is the position of a pixel in the image and d is the distance at (x,y) assigned by the transform. Next Dimitrov *et al.* mark the directions of the greatest change of the function f. This is simply done by calculating gradients (or reversed gradients) of this function. The image after this operation would look like in Figure 10. Below is the pseudocode for calculating the gradients.

Compute reversed gradients of a distance transform *DT*:

For *y := 2* to *DT.height-1* do
    For *x :=* 2 to *DT.width-1* do
        Create vector *v* at point *(x,y)*:
        v[x-1][y-1] := (DT[x-1][y] - DT[x+1][y], DT[x][y-1] - DT[x][y+1])
        and normalize it to obtain a unit vector
    End *{ For }*
End *{ For }*

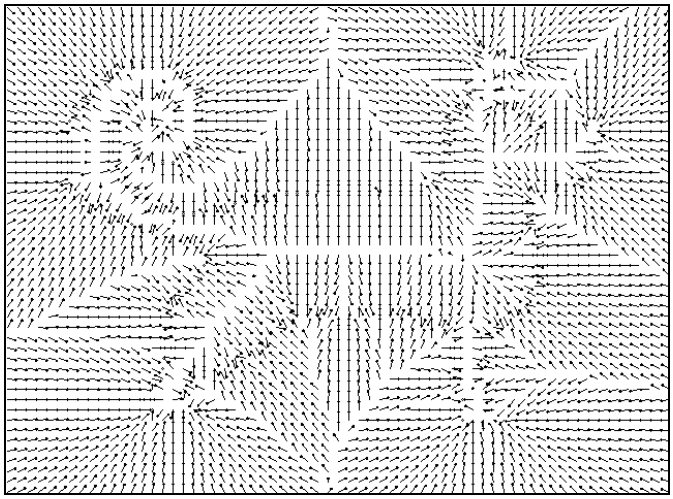


Figure 10: Reversed gradients of the EDT

It is now easily seen that the directions of gradients can be used to determine where the skeleton lies. The most interesting points are the ones where the vectors spread apart from each other. To remind, the image above was obtained from an unsigned distance transform (EDT). In the case of signed distance transform the gradients outside the shape would be pointing just as in Figure 10 but the ones inside the shape would have exactly the opposite directions. This would cause them to be pointing into each other inside the shape and apart from each other on the outside. The purpose of computing signed distance transform is now clear – to enable straightforward skeleton elicitation. It can, however, be obtained with unsigned distance transform as well, as described later in section 3.2.

When the gradient field is computed, the next step is to transform it into scalar values which will facilitate acquiring the skeleton. The property which can be used for this task is called the divergence of a vector field – a term that has its roots in physics. The interpretation of divergence is that it determines the extent to which a point in a vector field behaves like a source or a sink (of energy or a fluid substance). Every point where the vectors are directed apart from each other in Figure 10 behaves like a source, where the field is generated, however if the distance transform would be signed, the points inside the object which now are sources, would be the sinks. Locating sinks or sources (depending on the directions of vectors) is equivalent to finding the skeleton. The divergence at point P of the image is computed as [4]:

$$div(P) = \sum_{i=1}^{8} N_i \circ \nabla D(P_i),$$

where P_i is the ith 8-neighbor[1] of P and N_i denotes a unit outward normal vector at P_i of a unit circle centered at P. $\nabla D(P_i)$ is the gradient vector of function f at point P_i. The above formula is therefore a sum of eight cross products of vector pairs. The normal vectors are depicted in Figure 11 for better understanding.

[1] A pixel P_i is an 8-neighbor of P when the two pixels are adjacent (8-adjacency). P_i is a 4-neighbor of P when the pixels are adjacent either vertically or horizontally but not at a slant (4-adjacency).

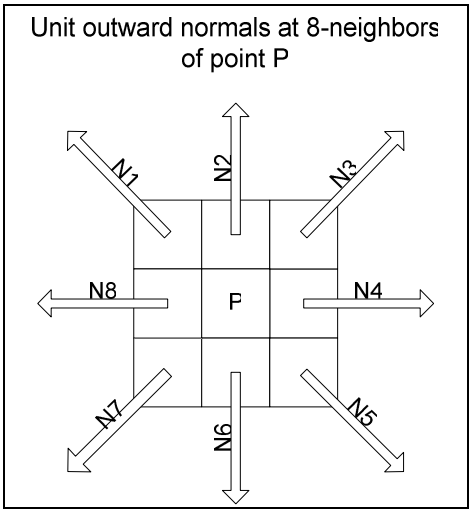


Figure 11: Normal vectors

The divergence is then computed by iterating through the surrounding pixels of *P* and summing the cross products of the current normal and the gradient. The sum is then assigned to the point *P* and represents the value of the divergence at this point. The resulting image is shown in Figure 12. The pseudocode for computing the divergence now follows [4].

Compute the divergence *div* of the gradient vector field *grad*

For *y := 2* to *grad.height-1* do
    For *x :=* 2 to *grad.width-1* do

$$div[x\text{-}1][y\text{-}1] := \sum_{i=1}^{8} N_i \circ grad((x,y)_i)$$

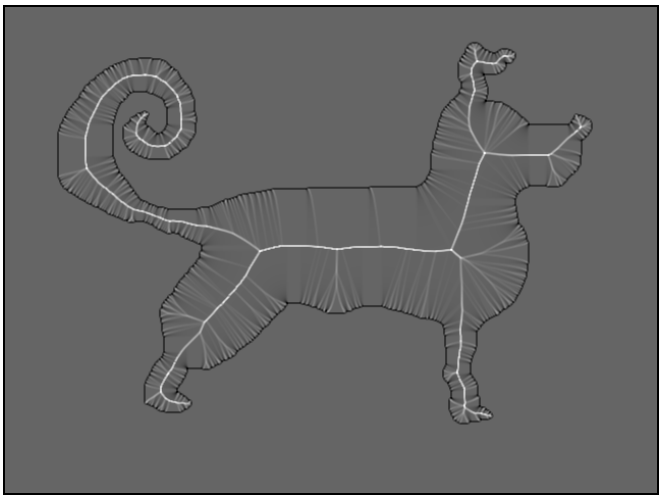


Figure 12: The divergence of the gradient vector field

Note that the divergence outside the object is not taken into account and the brighter the pixels, the grater the value of the divergence. The skeleton can be now seen, but there are still come computations to be done to actually obtain it. In order to acquire the skeletal points, Dimitrov *et al.* propose a divergence-based thinning algorithm. It consists in removing the points from the outline of the shape which have the minimum divergence value. Additionally, the points on the contour can be removed if their removal neither produces a hole in the object nor separates it into two

objects. The algorithm uses a heap to store the divergence values and extract relevant points in logarithmic time.

Divergence-based thinning [4]:

```
For each border point P
    If( P is removable )
        Heap.insert( P ) { with div( P ) as the sorting key for insertion }
End { For }
While( Heap.size > 0 )
    P := Heap.extractMin( )
    If( P is removable )
        If( ( P is not an end point ) or ( div( P ) < Threshold ) )
            Remove P
                For all neighbors Q of P
                    If( Q is removable )
                        Heap.insert( Q )
        Else mark P as skeletal (end) point
        End { If }
    End { If }
End { While }
```

The threshold in the algorithm has to be set empirically and causes that the unwanted branches are removed from the skeleton. The higher the threshold, the less points will be marked as skeletal points. For the details of the algorithm, such as the conditions for when a point is removable or when it is an end point, see [4]. When the algorithm is applied to the divergence transform obtained in the previous step, it results in the skeleton, similar to the one depicted in the Figure 7, page 9. Having described the notion of the shape's skeleton and the algorithm for its acquisition, it is now possible to define the aforementioned shock graph.

Shock graphs [15][16][19]:

A shock graph *SG(O)* of a 2D shape *O*, according to the definition in [19], is a labeled graph $G = (V,E,\gamma)$, where:

- vertices $V = \{1,2,...,n\}$
- edges $(i, j) \in E \subseteq V \times V$ directed from vertex i to vertex j if and only if $i \neq j, t_i \geq t_j$ (times of formation), and $i \cup j$ is connected in the plane
- labels $\gamma : V \rightarrow l$, with $l \in \{\tilde{1},2,\tilde{3},4,\#,\Phi\}$, with $\#$ representing the root of the graph and Φ its leaf nodes
- topology such that, $\forall j \in V$ with $\gamma(j) \neq \#$, $\exists i \in V$ with $(i, j) \in E$

Abstracting from the formalities above, a shock graph can be described as a directed, rooted, acyclic, labeled graph. What is distinctive for shock graphs is the labeling of nodes. There are 4 categories of nodes which are assigned to the nodes depending on the change of the radius of an inscribed circle, center of which moves along the skeleton of the shape (Figure 13).

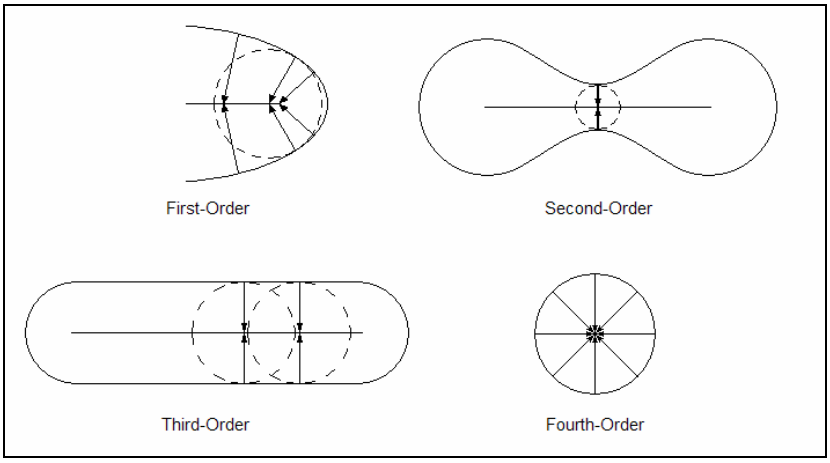


Figure 13: The types of shocks (figure adapted from [16], page 15)

A node of a shock graph is called a shock and is labeled as a first-order shock when it represents a protrusion in a object, i.e. the radius of a circle changes monotonically – either increases or decreases. The second-order shocks represent necks in the object, that is, the points where the function of change of the radius reaches a local minimum – it implies that the second-order shocks are surrounded by first-order shocks. The third order shocks represent the parts of the shape which maintain equal thickness along their length, in other words, the radius of the inscribed circle does not change as the circle travels down a branch of the skeleton. The fourth-order shocks represent a local maximum of the radius change function, e.g. as in the skeleton of an ellipse. Constructing and matching shock graphs are to date best described in [16], please refer to it for further details.

3 PROPOSED APPROACH

The approach proposed for this thesis is based on what was described in the previous chapter. Two feature representations were chosen, namely histograms and graphs. The histogram approach is the color indexing itself and the graph approach is based on shock graphs.

3.1 Algorithms used for histograms

The first approach used for the needs of the OR system built for this thesis was to use a standard backpropagation neural network [5] as a classifier for the histogram data. Histograms were treated as three separate entities, one for each color channel. Surprisingly, the ambiguity of such representation (described in section 2.2, Figure 5, page 6) did not hinder correct object recognition. The network was, however, trained to recognize only one specific object, i.e. the output of the network consisted in only one neuron which indicated whether the histogram fed to the input of the network represented the object which the network was trained to recognize or not. This approach is depicted in Figure 14. Note that the input of the network was in practice a vector which was a concatenation of the three histograms representing R, G and B color channels.

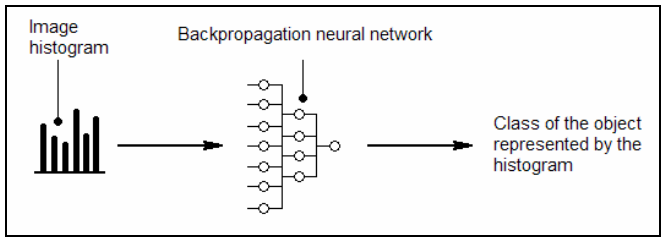


Figure 14: One approach to histogram classification

The network was trained to look for a specific mug in a fixed scene. The scene was a table with objects standing on its surface. The output of the network indicated whether the image contains the objects of the class 'mug' of not. Promising results were obtained for the conditions described. The mug was found even when it was rotated, regardless of the direction of the rotation. Furthermore, the mug was found in the image even when approximately 70% of its surface were occluded by objects that were unknown to the network. The training patterns for the network were pictures of the background, which was the table, and only a few pictures showed fragments of the mug. The results were hence impressive. Unfortunately the approach was abandoned due to problems with training the network to recognize more classes. For the purpose, a network with the number of output neurons equal to the number of classes was constructed. Every neuron represented one class and was intended to produce the output of value 1 if the class of the object fed to the input was this neuron's class and -1 otherwise. Achieving such result turned out to be challenging – the author conducted some rough experiments with changing the number of histogram bins which affected the length of the input vector for the neural network and should affect, and in a favorable situation facilitate, the training process. However, no satisfactory results were attained and therefore two alternative solutions were considered to be implemented. One was to train separate neural networks for every class, i.e. every neural network would only recognize objects of the class it was trained to respond to. This approach would be resource-consuming and the inconvenient training process would not be omitted. Hence eventually the standard approach was selected. The histogram was converted to the unambiguous, three-dimensional form and the

histogram intersection distance metric was used as a classifier (see section 2.2 for details) instead of the backpropagation neural network.

3.2 Algorithms used for graphs

The approach to shape recognition used in this thesis was inspired by the shock graphs approach introduced in section 2.3. It will be explained by presenting the subsequent steps which lead to construction of the graph designed for the OR process. Similarly to the shock graph approach, first a skeleton of the shape has to be acquired. This is done by using Dimitrov's *et al.* algorithm [4] (section 2.3). The algorithm used in this thesis is, however a slightly modified version. Originally a signed distance transform of the image was used – to recall: this means that for every point located outside the shape the positive distance is computed from the nearest point lying on the outline of the shape, while the negative distance is assigned to each point located inside the shape. It was, however, desired to use an available implementation of an unsigned Euclidean distance transform algorithm proposed by Felzenszwalb and Huttenlocher [6] (downloadable from Felzenshwalb's website). To enable Dimitrov's *et al.* algorithm to use an unsigned distance transform instead of the signed one a modification of the input image was necessary. Recall that the input of the original algorithm is a monochromatic image where the background is marked with one color and the shape with the second one, as in Figure 8, page 9. Note that after applying the distance transform, either signed or unsigned, the points which constitute the shape are assigned with zero-distance, as the nearest points of the shape to them are themselves. This fact was used for adapting the skeleton construction algorithm to use an unsigned distance transform. The goal is to neglect the divergence values outside the shape. The divergence is uniform if the vectors which it was computed from, point the same direction. The vectors in this approach are the gradients of the distance transform. It is now easy to notice that when all the points outside the shape are marked with the same color as the shape itself, they will be treated the same as the shape, that is, they will be assigned with zero-distance when a distance transform is applied. On the other hand, the points inside the shape will be marked with the regular distance values. As a result the desired uniform divergence outside the shape, and diverse in the inside are obtained. An example of the input image for the modified algorithm as well as its unsigned Euclidean distance and the gradient field are shown in the figures below. The divergence obtained from such approach is depicted in an earlier section in Figure 12, page 12.

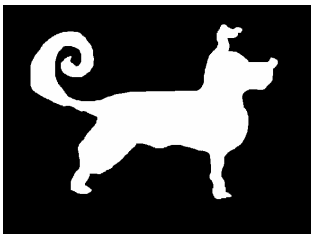


Figure 15: Input image for the modified algorithm

Figure 16: Euclidean distance transform

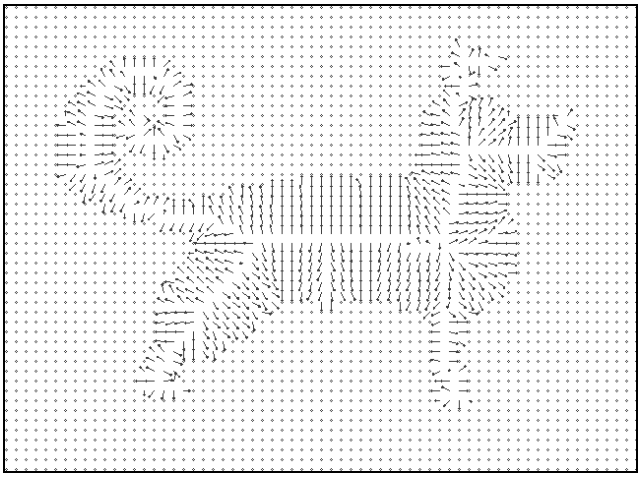


Figure 17: Gradients of the distance transform

Dimitrov *et al.* do not describe how they acquire the shape from the input image. The shape has to be organized in some data structure, as the divergence-based thinning algorithm (section 2.3) starts with storing every 'border point' (i.e. point which constitutes the shape) in a heap (recall that the divergence value is the sorting key for insertion). This problem is similar to finding the convex hull of a set of points, however in this particular case the points are adjacent. The author decided to accomplish this task with a simple algorithm presented below.

Shape acquisition algorithm

Find starting point *S { by horizontal line scan, top-down and left-to-right }*
Set the starting point *S* as the current point *C { C := S }*
Set the point (*C.x – 1,C.y*) as the previous point *P { P :=* (*C.x – 1,C.y*) *}*
Do
    Set the next point *N* as *N :=* NextBorderPoint(*C, P*)
    P := C *{ current is now previous }*
    C := N *{ next is now current }*
    Add point *C* to the data structure representing the shape
While($C \neq S$) *{ while current is not where the algorithm has started }*

The NextBorderPoint( ) routine will be now described in detail. Given a point *C*, which is the current point being processed, and a point *P*, which was the previously processed point, it is easy to find the next point *N* of a closed curve. The task is accomplished by analyzing the 8-neighors of the point *C* starting from the point which is next after the point *P* in a specific order. In theory, the order is clockwise, as shown in Figure 18, however in practice the image coordinate system is constructed in such manner that the values on the vertical axis grow down the axis, hence the order has to be reversed and is in practice counterclockwise.

2	3	4
1	**C**	*N*
P		

Figure 18: Finding the next point N of a shape given the current point C and the previous P

The numbers indicate the sequence of visiting the points to test whether their color represents the background or the shape. In this illustration the next point *N* was found in the fifth iteration of the NextBorderPoint( ) routine. The pixels marked blank lie within the interior of the shape. The whole algorithm ends when the current point is equal to the point where the algorithm has started and then the divergence-based thinning follows.

When the thinning finishes it results in two products. The first is the image which contains the skeleton and the second is a collection of *terminal points*. These are the points of the skeleton which are adjacent to only one point, in other words, a terminal point is one of the two end points of a skeleton branch. One more collection of points is needed for constructing a graph that could be used for object recognition, in particular the collection of the *junction points* which are the points that have 3 or more neighbors, in other words, junction points are the points where 3 or more skeleton branches meet. Again, Dimitrov *et al.* do not specify an algorithm for obtaining the junction points and the branches but the algorithm is quite straightforward. It makes use of the same NextBorderPoint( ) routine used in the algorithm for obtaining the shape from an image. The starting points are the terminal points of the branches and the stop criteria is reaching a point that has 3 or more neighbors. Traversing every branch gives the information about the first and last point of each branch. This information is then used to form edges between nodes of the graph. Next, graph and its construction will be explained.

Constructing the graph

The graph *G(O),* designed for the purpose of this thesis, of a 2D shape *O* is defined as $G = (V, E)$, where:

- vertices $V = \{1,2,...,n\}$
- edges $(i, j) \in E \subseteq V \times V$ directed from vertex i to vertex j if and only if $i \neq j, t_i \geq t_j$ (times of formation), and $i \cup j$ is connected in the plane
- topology such that, $\forall j \in V$ with $\gamma(j) \neq \#$, $\exists i \in V$ with $(i, j) \in E$

The graph is defined on the basis of a shock graph and is its simplified version. Compared to a shock graph the graph defined above lacks the labels of nodes, however a root node is clearly distinguished. In addition to this the nodes contain different data than the original nodes in a shock graph and are positioned at different points of the image. Nodes of the original shock graph contain the points of the skeleton for which the values of radius change function (also called the time formation change function) changes in one of the ways depicted in Figure 13, page 14 (types of shocks). This implies that the shock graph nodes are created along the skeleton branches. In addition to the collection of the points, a shock graph node also contains the minimum and maximum time of formation of the points within the node. The nodes of the graph defined in this section, on the other hand, contain slightly different information. Nodes of this graph are constructed only at terminal and junction points and contain the Euclidean coordinates of the point of construction and the time of formation (radius of the inscribed circle), which is directly attained from the Euclidean distance transform. When the nodes are constructed, they are connected with edges which are directed from the vertices with grater time of formation towards those where the value is smaller. Graphically this means that a vertex constructed from the center of a circle which radius is bigger will be the parent and the smaller radius node will be its child. When the nodes are connected with the edges, a root node is added and connected with the vertices which in-degree is equal to zero, i.e. the vertices with no parents. This

facilitates the graph traversal because if there would be more than one zero in-degree nodes, starting the graph traversal at one such a node would prevent visiting the remaining zero in-degree nodes. It has even more important implications when comparing graphs, as polynomial-time algorithms for graph comparison exist for rooted, directed, acyclic graphs [15], whereas no such algorithms are known to date for graphs that have no root. An example of the graph is shown in Figure 19. For a summary of all graph construction steps see Figure 21 at page 24.

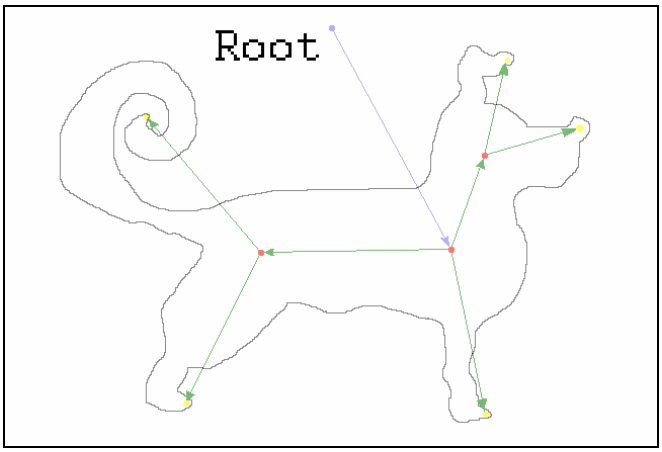


Figure 19: Graph built from the skeleton of the shape

Matching graphs

As in the case of histograms, graphs can be compared by using a metric for graph similarity. Graph similarity approaches are divided into two categories: the first compare the graphs statically by analyzing the two graphs and producing a similarity value; the second accomplish this task dynamically, i.e. they define a space of graph transformations (adding, removing, altering nodes and edges) then put the two graphs to be compared in that space and search for the shortest path of transformations which transform the first graph into the second one. This is called a 'tree-edit-distance algorithm' [12]. Costs are assigned to particular types of transformations and the length of the path is the sum of the costs along the path. This algorithm can be stated otherwise to find a graph which both graphs can be transformed to by the lowest cost.

In this thesis, a structural approach was used, based on [19][15]. It takes two factors into account when comparing graphs, those are the topology of the graph and the distance between vertices (their similarity). The distance between two vertices *u* and *v* of the earlier described graph is defined as:

$$C(u,v) = |u.t - v.t| + \|u.tsv - v.tsv\| + |u.d - v.d|,$$

where *t* is the time of creation of a node, *d* is the Euclidean distance of a vertex from its parent (recall that every vertex is created from a point in an image and hence is located in a 2D space; distance from the root is set to zero) and *tsv* is the *topological signature vector* (TSV) [15]. TSVs are the elements responsible for representing graph topology at a node. Construction of TSV at a node *u* requires building an adjacency matrix representing a subgraph rooted at *u*. An adjacency matrix *A* is a square matrix with its rows and columns indexed with the vertices of the graph which it is supposed to represent. The *(i,j)*-th entry of *A*, denoted as $A_{i,j}$, equals 1 if $(i, j) \in E$, and 0 otherwise [19]. TSV at a node *u* is then computed as [15]:

$$tsv(u) = [\lambda_u, \lambda_1, \ldots, \lambda_k].$$

Given that u has k children, λ_u denotes the sum of k largest eigenvalues of matrix A_u built for the graph rooted at u. Similarly, λ_i where $i \in [1;k]$, denotes the sum of k largest eigenvalues of matrix A_i built for the graph rooted at the i-th child of u. Additionally $\lambda_i > \lambda_{i+1}$ and $\forall i.\lambda_u > \lambda_i$. The latter predicate is a property of the eigenvalues but the first inequality has to be realized by sorting. In practice the graphs in the modelbase differ in the maximal number of their node's children, therefore a maximal size of TSV is empirically set. If k is smaller than this value, the vector is padded with zeroes.

To compare the graphs, a matrix $\Pi(G,H)$ has to be formed, where G and H denote the graphs to be compared [19]. The rows of the matrix are indexed with the vertices of graph G and the columns with the vertices of graph H. The *(i,j)*-th entry of $\Pi(G,H)$, denoted $\Pi(G,H)_{i,j}$, is $C(G_i,H_j)$, the distance between the i-th vertex of graph G and the j-th vertex of graph H. In the original algorithm for comparing graphs [19] a bipartite edge weighted graph is next formed with the edge weights from $\Pi(G,H)$. The graph is then processed to obtain the best set of matching nodes. The version of this algorithm designed for this thesis, however, differs at this point. For each vertex G_i, the minimum distance function value is found to indicate the best matching vertex H_j. This is done by iterating through every column (H_j) of $\Pi(G,H)$ to find for each row (G_i) the minimum $C(G_i,H_j)$ value. It is not sufficient however, to find the best matching for every row since many rows can fit best to the same column. In other words, one vertex H_j can have many vertices from G that are assigned to it as its best matches. Due to this fact, a second step is needed which consists in finding the best match among the many nodes from G assigned to H_j. When the best match is found the pair of H_j and its best match are not allowed to be chosen again and the remaining candidates for matching H_j are assigned to the remaining vertices of H in the manner described earlier. The algorithm keeps rearranging the choices until there are no vertices of H which have more than one vertex from G assigned. This implies that when forming the matrix $\Pi(G,H)$ this algorithm requires the graph H not to have less nodes than the graph G, i.e the matrix must have more columns than rows, otherwise the algorithm would not stop. Pseudocode for the graph matching algorithm will now follow.

The algorithm for measuring the similarity of two graphs *G* and *H*

{ build $\Pi(G,H)$ *matrix }*
For each G_i in G
    For each H_j in H
        $\Pi(G,H)_{i,j} := C(G_i,H_j)$

{ find the minimums for every row }
For each $\Pi(G,H)_i$ *{ for each row }*
    Find the the index *jMin* of lowest value of $\Pi(G,H)_{i,j}$
    Add i to the *candidatesAt[jMin] { an array of lists }*
End *{ For }*

{ rearrange the choices of best matches so that the matches are unambiguous }
While($\exists i.candidatesAt[i] > 1$)
    Find the index *cMin* of the best candidate in the list *candidatesAt[i]*
    Remove *cMin* from the list of *freeMininumums { indices of vertices of H }*
    Remove row indices *indices* not equal to *cMin* from *candidatesAt[i]*
    For each *row* in *indices*
        For each *column* in *freeMinimums*
            Find the index *jMin* of the lowest value of $\Pi(G,H)_{row,column}$
        Add *row* to *candidatesAt[jMin]*
    End *{ For }*
End *{ While }*

{ compute the similarity of graphs G and H }
result := 0
For *i = 1* to *candidatesAt.Length* do
    If(*candidatesAt[i].isNotEmpty*)
        Set *j* as the only vertex stored at *candidatesAt[i]*
        result := result + $\Pi(G,H)_{i,j}$
    End *{ If }*
End *{ For }*

$similarity := \frac{(G.n + H.n) \cdot result}{2 \cdot G.n \cdot H.n}$ *{* [15] *G.n and H.n denote the numbers of vertices }*

The algorithm returns the similarity of the graphs *G* and *H*. The measure, however is not bounded to the range *[0;1]*, as it is in the original algorithm [15] and therefore it needs to be divided by an empirically chosen value to be scaled to a similar range.

3.3 Modelbase, hybrids and the application

The modelbase used for the OR system built for this thesis consisted of 26 objects which depicted numerous tools. These tools varied in shape and color to make use of both OR methods that were intended to be combined. Furthermore, every model was assigned by the author to a class with a given name, based on the real-life use and/or looks of the tool which the model represented. The division of the objects into classes is depicted in Table 1 and Table 2. The headers of the tables contain the names of the classes, while the cells are filled with the objects assigned to these classes.

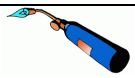
Blowtorch	Bolt	Chainsaw	Drill	Hammer
 *Blowtorch*	 *Bolt*	 *Chainsaw*	 *Green drill*	 *Hammer*
 *Red blowtorch*	 *Nail*		 *Yellow drill*	 *Mallet*
 *Rotated relabeled blowtorch*	 *Screw*		 *Soldering iron*	 *Pickaxe*

Table 1: Modelbase, part 1

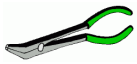
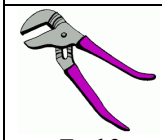
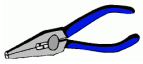
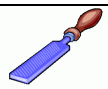
Pliers	Fork	Screwdriver
 *Tool 1*	 *Pitchfork*	 *Brown chisel*
 *Tool 2*	 *Rake*	 *Yellow chisel*
 *Tool 3*		 *File*
 *Tool 4*		 *Green screwdriver*
 *Tool 5*		 *Orange screwdriver*
 *Tool 6*		

Table 2: Modelbase, part 2

Some objects depicted in the images contain holes in their contours (e.g. Chainsaw, Soldering iron, Tool 5). Such contours can not be directly transformed into shapes, as the definition of the shape does not allow for holes in a shape. An algorithm for transforming such an image into its counterpart representing the shape has to find the contour of the object (using the algorithm described in section 3.2) and then using an interior filling algorithm fill the inside of the closed curve which constitutes the

shape. This functionality was, however, not implemented in the application and the pictures were manually transformed into their black-and-white shape representations.

Hybrids

The aim of the work is to combine the two OR methods described in section 3.1 and section 3.2 and check which combination yields best results. Four approaches for combining the methods are now proposed:

1. Grater confidence selection (GC) – independently apply the histogram and the graph methods (denoted by HM and GM respectively) to the input image and choose the result of the method which gives the similarity measure closer to zero. In other words, choose the result of the method which is more confident about its answer.
2. Histogram method with a threshold and the final decision made on the basis of the grater confidence selection (HTGC) – apply the histogram method, if the similarity value is below the threshold, set the result as final. Otherwise apply the graph method and eventually, as in the case of the grater confidence selection, choose the result of the method with grater confidence.
3. Histogram method with a threshold and the final decision made on the basis of the threshold (HT) – apply the histogram method, if the similarity value is below the threshold, set the result as final. Otherwise apply the graph method and set its result as final without checking the confidence of both methods.
4. Graph method with a threshold and the final decision made on the basis on the threshold (GT) – this hybrid is analogous to the previous one, except that it treats GM as the primary method.

Application

The application implemented for this thesis consists of three main functional modules and the graphical user interface (GUI) (Figure 20).

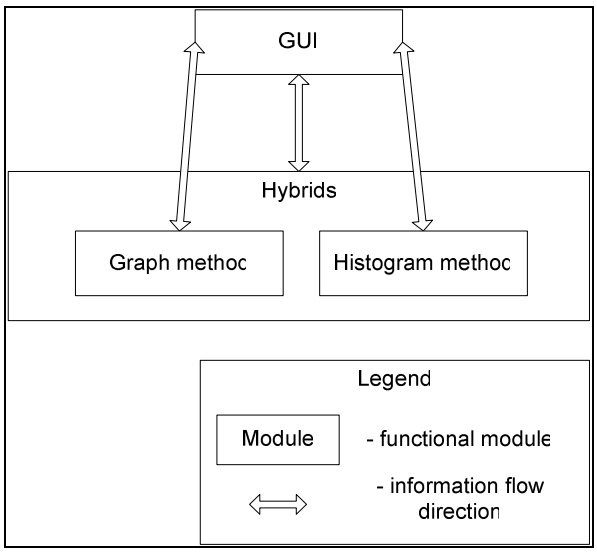


Figure 20: Overall view of the functional modules

The *Graph method* module realizes the graph object recognition approach explained in section 3.2. The *Histogram method* functional module implements the color indexing method described in section 2.2. *Hybrids* implements the four hybrids described earlier in this section. The Graph method and the Histogram method can be

invoked independently from the GUI or work together by being invoked from the *Hybrids* module which is also run from the GUI. The Graph method, shown in detail in Figure 21, takes an image as input. The image has to be monochromatic with the object marked white on black background. The image is passed to the *Euclidean distance transform* module where it is processed in the way explained in section 3.2. The result is passed on to the *Gradients computation* module which calculates the gradient vector field on the basis of EDT. The *Divergence computation* module takes the vector field as input to calculate its divergence. Note that the input image is also passed to the *Contour acquisition* module. Its product, the shape of the object, and the formerly mentioned divergence are passed to the *Thinning algorithm* which produces the skeleton of the shape. The skeleton is then used for creating the graph representation of the object depicted in the input image, this step is realized by the *Graph builder*. In order to determine the class of the object, the best matching graph in the modelbase has to be found. Therefore the graph produced by the *Graph builder* is passed to the *Graph matching* module which performs the comparison and returns the class of the object from the input image as the output of the Graph method module.

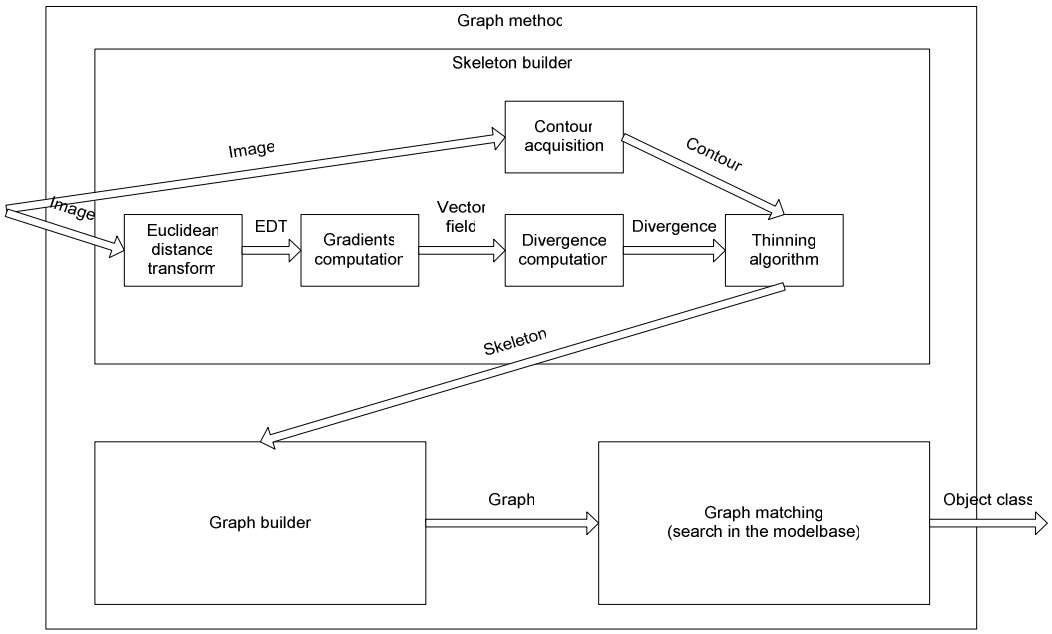


Figure 21: Graph method functional module

Compared to the Graphical method module, the Histogram method module is very simple (Figure 22). The input image, in this case a color image, is passed to the *Histogram builder* which computes the three-dimensional color histogram (see section 2.2). The histogram is then set as the input of the *Histogram matching* module which searches for the best matching object in the modelbase and returns the class of this objects as the output of the whole Histogram method module.

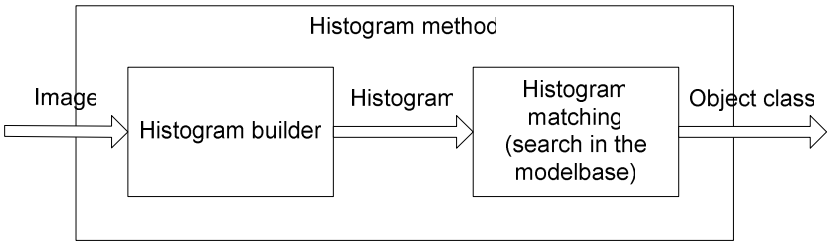


Figure 22: Histogram method functional module

4 RESULTS AND DISCUSSION

This chapter presents the results obtained from the application described in chapter 3. First the parameters of the system are described, then tests of the parameters follow along with discussions of the outcomes.

4.1 System parameters and experiment description

The behavior of the system can be adjusted by changing the values the following parameters:

1. The *number of histogram bins* is the number of ranges that each dimension of the color histogram is divided into (section 2.2). If the histogram is not divided into ranges, but instead every single value has its own bin, it results in an overly 'precise' histogram which represents every object extremely accurately. In consequence problems with making generalizations arise, i.e. assigning similar objects to the same class. Additionally such histogram would be gigantic[2] and hence comparing these histograms would be very time-consuming. Taking it to the other extreme, if each dimension of the histogram was divided into one big bin which would represent all the values then every object would be represented by the same histogram and this would also be an undesired situation.
2. The *divergence threshold* used in the thinning algorithm. The value of this parameter controls how complex the skeleton will be. Increasing the threshold results in removing more points in the thinning process so that the skeleton shrinks and the unwanted, insignificant branches are removed. Setting an optimal value for this parameter is important because graphs formed on the basis of an overly complex skeleton which contains many insignificant branches will tend to represent the object in too much detail to be compared to other objects of the same class with a decent similarity value. On the other hand, graphs constructed from a very simplified skeleton will all be very similar to each other which will also hinder the matching algorithm in finding the correct class.
3. *TSV vector size* is the number of dimensions of a TSV vector. The size of the vector is determined by the largest number of children a vertex in any graph in the modelbase can have. This stems from the way how TSVs are formed (section 3.2).
4. The *graph metrics scale* is the value that the result of the graph matching algorithm has to be divided by in order to scale it to the range of *[0;1]*. Changing this value uniformly alters the similarity values given by the graph method and can therefore be used for changing the results obtained from the hybrids.
5. The *histogram method threshold* is the value which performs the role of a switch that initiates the graph method when the histogram method is not sure enough about its result. It alters the behavior of HTGC and HT.
6. The *graph method threshold* is analogous to the previous parameter but influences GT.

[2] Given that one color channel consists of 256 values, a 3D histogram of three channels would consist of $256^3 = 16777216$ values. A value of a histogram is an unsigned integer and takes up 4 bytes. This results in a 16777216*4 [Bytes] = 64 [MB] big histogram with most of its bins holding zero values.

The *number of histogram bins* was set to a fixed number of 16, based on [7]. *TSV vector size* was arbitrarily set to 20, what will suffice for graphs even larger than the ones built from this modelbase. The results of testing the remaining parameters are presented in the next section.

Experiment description

To find out which methods give best results for the object classification task, an experiment was planned and conducted. The core of the experiment can be described in terms of pseudocode.

Core experiment

```
For each M          { M is a method: histograms, graphs or one of the hybrids }
    For each O      { O is an object from the modelbase }
        Remove O from the modelbase
        Using method M find the best match B of O in the modelbase
        Store (M, O, B)
        Put O back into the modelbase
    End { For }
End { For }
```

As a result a set of *(M, O, B)* triplets is produced, where each triplet describes what object *B* best matched object *O* according to method *M*. The parameters of the method *M* can be set before the core experiment is initiated. When examining a range of values of a parameter the core experiment can be surrounded by another for-loop which iterates though the values of the parameter. The idea is to invoke every method for the same object and find the best match when the object is removed from the modelbase. The objects are explicitly assigned to classes and therefore it can be easily told if the method correctly classified the object ($class\_of(O) = class\_of(B)$) or not ($class\_of(O) \neq class\_of(B)$).

4.2 Parameter tests

In this section the system parameters described in the previous section are tested to determine their influence on the effectiveness of OR, and hence the object classification task.

Divergence threshold

Testing the parameter consisted in changing the *divergence threshold* values within the range of [2.5; 6] with step 0.1. The results are gathered in the table below. The columns are named with parameter or method names. The histogram method (color indexing) is denoted by HM, while the graph method is denoted by GM. Each value in a column described by a method name represents the number of objects correctly classified by the method for a certain divergence threshold value. Values of the remaining parameters for this experiment were set as follows:

Graph metrics scale	= 7.0
Histogram method threshold	= 0.065
Graph method threshold	= 0.2

Divergence threshold	HM	GM	GC	HTGC	HT	GT
2.5	6	9	5	6	9	9
2.6	6	9	8	8	9	9
2.7	6	10	6	6	10	10
2.8	6	9	6	6	10	9
2.9	6	8	6	6	10	8
3.0	6	8	6	6	10	8
3.1	6	8	7	8	10	8
3.2	6	8	7	8	10	8
3.3	6	8	8	8	10	8
3.4	6	9	7	8	11	9
3.5	6	9	8	8	11	9
3.6	6	10	8	8	11	10
3.7	6	10	8	8	10	10
3.8	6	10	8	8	10	10
3.9	6	9	6	6	10	9
4.0	6	9	6	6	10	9
4.1	6	8	6	7	10	8
4.2	6	8	6	7	10	8
4.3	6	8	6	7	10	8
4.4	6	10	6	7	10	10
4.5	6	10	6	7	10	10
4.6	6	10	6	6	10	10
4.7	6	10	8	8	10	10
4.8	6	10	5	6	10	10
4.9	6	10	5	6	10	10
5.0	6	11	7	8	10	11
5.1	6	10	7	8	10	10
5.2	6	11	6	7	10	11
5.3	6	8	6	7	9	8
5.4	6	7	5	6	9	7
5.5	6	6	5	6	8	6
5.6	6	8	7	8	10	8
5.7	6	9	7	8	11	9
5.8	6	10	9	10	11	10
5.9	6	11	10	11	10	11
6.0	6	9	8	10	10	9
Sum:	**216**	**327**	**241**	**264**	**359**	**327**

Table 3: The divergence threshold test

Recall that the divergence threshold influences how the skeleton will be formed and, in consequence, how the graph of a shape will look like. Therefore by changing the divergence threshold it is possible to fine-tune the graph method by analyzing the number of correctly classified objects as the threshold changes. The best results for the graph method were obtained for three threshold values: 5.0, 5.2 and 5.9. All the three values resulted in 11 positive classifications (42% of correct answers, since 26 objects were in the modelbase). It is worth noticing that the largest numbers of correct classifications for the hybrids usually occur at different threshold values than they do for GM. The peak values of GM and the hybrids usually occur at different threshold values. Although apparently 11 is the largest number of proper answers that the

hybrids cannot exceed (as is the case of GM), the hybrids are still more accurate for the vast majority of the threshold values than HM or GM used alone. This stems from the fact that a set P_{Histo} of objects that are classified correctly by HM and the set P_{Graph} of objects classified correctly by GM are not identical in most cases (including the case of this modelbase). The idea of a hybrid uses this fact to find a compromise by approximating the set $P_{Hybrid} = P_{Histo} \cup P_{Graph}$. The last row of Table 3 holds the sums of all values in each column. These sums can be treated as a measure of accuracy of a hybrid under varying conditions (changes of the divergence threshold). Using this measure it is stated that HT achieved best results. The last hybrid which defines its result on the basis of the graph threshold (GT) turned out to work exactly the same as GM alone because of the value of the graph threshold set. The least effective hybrid was the grater confidence selection (GC), however it was still better than the histogram method used alone.

Graph metrics scale

The *graph metrics scale* was tested within the range of [10;20] with step 0.5 for two selected *divergence threshold* values. These divergence threshold values were chosen because of their influence on the change of the number of positively classified objects by three hybrids. The results are gathered in the table below. The values of the parameters that were fixed for this experiment:

Histogram method threshold = 0.065
Graph method threshold = 0.072

After careful analysis of Table 4 it can be stated that with the growth of the graph metrics scale factor the number of positively classified objects tends to slightly increase for the methods that use the grater confidence rule to compute their results. For the divergence threshold value of 3.3 the number of positive classifications changes as the graph metrics scale changes from 11.5 to 12, whereas for the threshold of 3.4 the changes occur for two pairs of values: 11 – 11.5 and 12.5 – 13. The reason of those changes is that the graph metrics scale changes the similarity value of a pair objects calculated by GM while the similarity of the objects calculated by HM remains the same. This causes the grater confidence methods (GC and HTGC) to give slightly different classification results for different graph metrics scale values, as they are the only methods that are trying to make their classifications dependent on the relation of the similarity values computed by HM and GM. The last method which could benefit from increasing the graph metrics scale is GT because the graph method threshold was fixed during this experiment thus lowering the similarity values for GM (increasing graph metrics scale) would eventually result in obtaining the threshold and producing similarity values lower than the threshold, what would lead to selecting GM results as final. However the graph method threshold was so small that most probably no similarity values computed by GM were lower than the threshold and therefore no switching to GM occurred, what caused GT to act as HM neglecting all classifications of GM.

To gain a better understanding of how the classifications are carried out, a table was produced (Table 5) which contains the rows labeled with the names of the objects from the modelbase and the columns labeled with the names of the classes. The gray fields represent the proper class for an object and the method name abbreviations were put in the cells representing the real answers of the methods, i.e. their classification of a given object. When an abbreviation is located in a gray cell, it means that the method which it stands for classified the object correctly, on the contrary the abbreviation located in a white cell denotes a wrong classification. The table was obtained for the divergence threshold of 3.3 and the graph metrics scale of 11.5. The rest of the parameters remain the same as for the graph metrics scale experiment.

Divergence threshold	Graph metrics scale	HM	GM	GC	HTGC	HT	GT
3,3	10	6	8	7	9	10	6
	10.5	6	8	7	9	10	6
	11	6	8	7	9	10	6
	11.5	6	8	7	9	10	6
	12	6	8	8	10	10	6
	12.5	6	8	8	10	10	6
	13	6	8	8	10	10	6
	13.5	6	8	8	10	10	6
	14	6	8	8	10	10	6
	14.5	6	8	8	10	10	6
	15	6	8	8	10	10	6
	15.5	6	8	8	10	10	6
	16	6	8	8	10	10	6
	16.5	6	8	8	10	10	6
	17	6	8	8	10	10	6
	17.5	6	8	8	10	10	6
	18	6	8	8	10	10	6
	18.5	6	8	8	10	10	6
	19	6	8	8	10	10	6
	19.5	6	8	8	10	10	6
	20	6	8	8	10	10	6
3,4	10	6	9	7	9	11	6
	10.5	6	9	7	9	11	6
	11	6	9	7	9	11	6
	11.5	6	9	7	9	11	6
	12	6	9	8	10	11	6
	12.5	6	9	8	10	11	6
	13	6	9	9	11	11	6
	13.5	6	9	9	11	11	6
	14	6	9	9	11	11	6
	14.5	6	9	9	11	11	6
	15	6	9	9	11	11	6
	15.5	6	9	9	11	11	6
	16	6	9	9	11	11	6
	16.5	6	9	9	11	11	6
	17	6	9	9	11	11	6
	17.5	6	9	9	11	11	6
	18	6	9	9	11	11	6
	18.5	6	9	9	11	11	6
	19	6	9	9	11	11	6
	19.5	6	9	9	11	11	6
	20	6	9	9	11	11	6

Table 4: Graph metrics scale test

	Blowtorch	Bolt	Chainsaw	Drill	Hammer	Pliers	Fork	Screwdriver
Blowtorch	HM GM GC HTGC HT GT							
Red Blowtorch	GM GC HTGC HT						HM GT	
Rotated relabeled blowtorch	HM GM GC HTGC HT GT							
Bolt		HM GT		GM GC HTGC HT				
Nail				GM			HM GC HTGC HT GT	
Screw		HM HTGC HT GT		GM GC				
Chainsaw		HM GM GC HTGC HT GT						
Green drill		HM GM GC HTGC HT GT						
Yellow drill		GM GC HTGC HT					HM GT	

	Blowtorch	Bolt	Chainsaw	Drill	Hammer	Pliers	Fork	Screwdriver
Soldering iron				GM GC HTGC HT			HM GT	
Hammer		GM					HM GC HTGC HT GT	
Mallet			GM				HM GC HTGC HT GT	
Pickaxe		GM					HM GC HTGC HT GT	
Tool 1						HM GM GC HTGC HT GT		
Tool 2						HM HTGC HT GT		GM GC
Tool 3	GM GC HTGC HT						HM GT	
Tool 4						GM GC HTGC HT	HM GT	
Tool 5						GM GC HTGC HT	HM GT	

	Blowtorch	Bolt	Chainsaw	Drill	Hammer	Pliers	Fork	Screwdriver
Tool 6						GM GC HTGC HT	HM GT	
Pitchfork				GM	HM GC HTGC HT GT			
Rake		HM GC HTGC HT GT						GM
Brown chisel					GM HT		HM GC HTGC GT	
Yellow Chisel							HM GM GC HTGC HT GT	
File						GM	HM GC HTGC HT GT	
Green screwdriver		GM					HM GC HTGC HT GT	
Orange screwdriver		GM					HM GC HTGC HT GT	

Table 5: Classification results

From Table 5 a conclusion can be drawn about the ease of classification. It seems that the best classification results were obtained for the 'Blowtorch' and 'Pliers' class. For the 'Pliers' especially GM was reliable since the shape of the objects of this class was very similar. HM, on the other hand performed well for the 'Bolt' class as the objects of the class were similarly colored. The 'Fork' class has a strange property of 'attracting' and misleading the methods what causes many incorrect answers (to fix this the algorithms used for the basic methods would have to be further modified to gain on correct answers). The hybrids either made use of the information given by both

HM and GM and classified an object correctly or not, but what needs to be emphasized is that a hybrid cannot give a correct answer when both of the basic methods (HM, GM) answered incorrectly because the hybrids are simply more or less complex 'switches' that choose the answer of one of the basic methods. They do not perform any averaging of the answers of HM and GM to choose something 'in between'. This can be seen in the above table – when a hybrid abbreviation is located in a gray cell, either one or both of the basic methods are located in the cell as well.

Divergence threshold	Histogram metod threshold	HM	GM	GC	HTGC	HT	GT
3,3	0.01	6	8	8	8	8	6
	0.015	6	8	8	8	8	6
	0.02	6	8	8	8	8	6
	0.025	6	8	8	8	8	6
	0.03	6	8	8	8	8	6
	0.035	6	8	8	8	8	6
	0.04	6	8	8	8	8	6
	0.045	6	8	8	8	8	6
	0.05	6	8	8	8	8	6
	0.055	6	8	8	8	8	6
	0.06	6	8	8	9	9	6
	0.065	6	8	8	10	10	6
	0.07	6	8	8	10	10	6
	0.075	6	8	8	9	9	6
	0.08	6	8	8	5	5	6
	0.085	6	8	8	5	5	6
	0.09	6	8	8	5	5	6
	0.095	6	8	8	5	5	6
	0.1	6	8	8	5	5	6
3,4	0.01	6	9	9	9	9	6
	0.015	6	9	9	9	9	6
	0.02	6	9	9	9	9	6
	0.025	6	9	9	9	9	6
	0.03	6	9	9	9	9	6
	0.035	6	9	9	9	9	6
	0.04	6	9	9	9	9	6
	0.045	6	9	9	9	9	6
	0.05	6	9	9	9	9	6
	0.055	6	9	9	9	9	6
	0.06	6	9	9	10	10	6
	0.065	6	9	9	11	11	6
	0.07	6	9	9	11	11	6
	0.075	6	9	9	10	10	6
	0.08	6	9	9	5	5	6
	0.085	6	9	9	5	5	6
	0.09	6	9	9	5	5	6
	0.095	6	9	9	5	5	6
	0.1	6	9	9	5	5	6

Table 6: Histogram method threshold test

Histogram method threshold

For the purpose of determining the significance of this parameter the values of the remaining parameters were set in the following manner:

Graph metrics scale = 14
Graph method threshold = 0.072

The values of the histogram method threshold were tested in the range from 0.01 to 0.1 with the step equal to 0.005. The results are presented in Table 6. The histogram method threshold, as in the name, influences only HT and HTGC methods. The remaining methods were left in the table for illustrative purposes only. Notice that the dependency of the number of positive classifications on the value of the threshold is not monotonic. As the threshold increases, the number of positive classifications tends to remain at a certain level, then it shortly rises to reach a maximum and lowers again. This happens because when the threshold is small, then no objects are classified with the similarity value lower than the threshold and hence the answer of GM is selected as final – it can be easily seen when GM and HTGC (or HT) columns are compared for the threshold values in the range of [0.01; 0.055] because for those thresholds HTGC and HT behave the same as GM. From 0.06 to 0.075 the threshold has the value which lets the switching of the basic methods yield best results. It means that some answers for these values are taken from HM which adds its contribution to the 9 objects positively classified by GM. Yet still the most interesting values are the ones from the range of [0.08; 0.1] because they cause the hybrids to perform even worse than the worst basic method (HTGC and HT correctly classify 5 objects while HM classifies 6). This occurs because the hybrids are starting to behave as HM alone but clearly one object is assigned grater similarity value than the maximum threshold used in this experiment. This causes that for this particular object, the answer provided by GM is taken as final, but unfortunately graphs fail to classify this object correctly what results in the performance of the hybrid worse than HM alone. The object was identified to be the ‘Bolt’ which was properly classified by HM through assigning the similarity value of 0.103 to the ‘Screw’ object which belongs to the same class. The threshold used in this experiment, however, did not exceed 0.1 and therefore the ‘Bolt’ was assigned to the class ‘Drill’ because that was the answer that GM provided, but was wrong – fortunately such anomalies occur very rarely and for the most part the hybrids perform equally good or better than the best basic method. It is now easily seen that setting the histogram method threshold over 0.103 would cause HTGC and HT to act as HM.

Graph method threshold

The parameters fixed for this experiment were:

Graph metrics scale = 14
Histogram method threshold = 0.07

Testing the graph method threshold was conducted for the range [0.005; 0.1] with step equal to 0.005 for the same values of divergence threshold as for the previous experiments, that is 3.3 and 3.4. The results are shown in Table 7. As in the case of the histogram method threshold, the graph method threshold influences only its relevant hybrid, GT. For small threshold values GT behaves similarly to HM, even worse for the divergence threshold equal to 3.4 and the graph method threshold set to 0.03 (the explanation of this fact is analogous to the one given for a similar case in the previous experiment). As the threshold raises GT starts to resemble GM to eventually become GM itself neglecting all HM classifications, which is an undesired feature for a hybrid. Setting the graph method threshold higher than 0.1 would not give better results, as 0.1

already causes GM classifications to be treated as final without even considering HM results. Although seemingly GT should perform very well because it was intended to favor the best basic method (GM), this experiment showed that favoring a basic method does not implicate treating it as the primary method, i.e. setting a threshold for it. The results presented in the earlier experiments indicate that the opposite fact is true – favoring the weaker method first and betting on the better method in case the weaker one is not sure enough of its answer seems to be an overwhelmingly better approach.

Divergence threshold	Graph method threshold	HM	GM	GC	HTGC	HT	GT
3,3	0.005	6	8	8	10	10	7
	0.01	6	8	8	10	10	7
	0.015	6	8	8	10	10	7
	0.02	6	8	8	10	10	7
	0.025	6	8	8	10	10	7
	0.03	6	8	8	10	10	6
	0.035	6	8	8	10	10	7
	0.04	6	8	8	10	10	7
	0.045	6	8	8	10	10	6
	0.05	6	8	8	10	10	6
	0.055	6	8	8	10	10	6
	0.06	6	8	8	10	10	7
	0.065	6	8	8	10	10	7
	0.07	6	8	8	10	10	8
	0.075	6	8	8	10	10	8
	0.08	6	8	8	10	10	8
	0.085	6	8	8	10	10	8
	0.09	6	8	8	10	10	8
	0.095	6	8	8	10	10	8
	0.1	6	8	8	10	10	8
3,4	0.005	6	9	9	11	11	7
	0.01	6	9	9	11	11	7
	0.015	6	9	9	11	11	7
	0.02	6	9	9	11	11	7
	0.025	6	9	9	11	11	6
	0.03	6	9	9	11	11	5
	0.035	6	9	9	11	11	7
	0.04	6	9	9	11	11	7
	0.045	6	9	9	11	11	6
	0.05	6	9	9	11	11	6
	0.055	6	9	9	11	11	7
	0.06	6	9	9	11	11	7
	0.065	6	9	9	11	11	8
	0.07	6	9	9	11	11	9
	0.075	6	9	9	11	11	9
	0.08	6	9	9	11	11	9
	0.085	6	9	9	11	11	9
	0.09	6	9	9	11	11	9
	0.095	6	9	9	11	11	9
	0.1	6	9	9	11	11	9

Table 7: Graph method threshold test

5 SUMMARY

The aim of this thesis was to examine what ways of combining object recognition methods give best results. To perform this task two OR methods were implemented and combined in four ways. Next the system parameters were tested within specified ranges in order to find their influence on the outcomes and assess the value of each hybrid.

From all the proposed hybrids HT turned out to be the best in terms of the largest number of correct answers. The divergence threshold and next the graph metrics scale tests indicated that HT performs better than any other hybrid. The key seems to lie in the usage of the weakest basic method as the basis for making a decision about which answer to choose. This can be well illustrated through a 'fools and geniuses' metaphor. HM with its 6 correct answers is almost half worse than GM with its 11 correct answers at maximum. Therefore HM could be seen as a fool and GM as a genius. With this metaphor in mind what HT truly conveys is 'if the fools are not sure enough, let us blindly rely on the geniuses'. This works as the geniuses are mostly right. On the other hand what GT tries to exploit is 'if the geniuses are not sure enough, let us blindly rely on the fools'. This simply has to lead to a disaster. Recall the example of looking for a lemon at a grocery store (given by the author in the introduction). The first feature used for this task was also color – shape was taken into consideration later. This fact is reflected in the results of the thesis.

A surprising fact is that the grater confidence rule, i.e. choosing the answer of the method which is the most confident about its result, gives poor outcomes. The explanation is quite straightforward – it cannot perform well since the similarities of the objects assigned by HM and GM are different for the same pair of objects, and obviously different for other objects. To give a clarifying example, let us imagine three pictures. Picture one shows a sausage, picture two a yellow banana and picture three shows a lemon. Note that picture one and two show objects which are very similar in shape while objects in pictures two and three do not differ much in color. Now, HM would be very sure that the banana is a lemon, while GM would be convinced that the banana is a sausage. Suppose now an additional picture four is introduced, which depicts a green banana. Now GM should indicate that the yellow banana is the most similar to the green banana, while HM should not change its rating, that is pictures two and three are still the most similar according to HM. The point is that simply comparing the values of similarity given by GM and HM is irrelevant not only because they measure different features but also because the methods use different measures (HM uses histogram intersection while GM uses its graph similarity algorithm). This causes that although the values of the similarities are narrowed to the same range of [0; 1], the distribution of the values in the range differs what hinders making fair comparisons (HM could be still more sure that the yellow banana is a lemon than GM would be that it is a green banana). Introducing the experiment with changing the graph metrics scale was an attempt to fix it but still it did not change the distribution of the values, it simply scaled the whole range with the hope of adjusting it to the HM-given similarity values range. GC hybrid was the worst not only because of the number of correct answers given but also from the perspective of the computational power required. It was the only hybrid which required executing both basic methods before giving an answer – this was needed to compare the similarities used by each basic method. The remaining hybrids executed the second basic method only if the first did not reach the threshold.

To sum up, the work succeeded in achieving its goals. All research questions stated in Chapter 1 were answered – question (1) in sections 2.2, 2.3 and the answers were specified in sections 3.1 and 3.2, question (2) in section 3.3, question (3) partially in section 4.2 and partially in the current chapter. Although the goals were achieved,

the obtained number of correct classifications does not suffice for the system to be usable in practice. Not only this, but also the specified format of the input images prevents the system from being anything but an academic example (the input images have to contain an object on white background). Despite the system was fully sufficient for performing the research for this thesis, these limitations left the author not fully satisfied and gave rise to many ideas for future work.

Future work

Numerous aspects of the OR system proposed in this thesis can be further improved and examined. The most significant directions for improvements and research are discussed below.

Improvement of the efficiency of the basic methods' algorithms

Although the experiments showed what the best way to build a hybrid is, the results are still poor – 11 out of 26 objects gives the percentage of 42% correct answers. Although 100% accuracy was impossible due to the intentional assigning of only one object to the 'Chainsaw' class, 42% is not satisfactory for most applications. A good OR system cannot be built out of weak bricks which are the basic methods. A proposed direction for improving HM is to experiment with HSV histograms, other histogram similarity metrics and varying numbers of histogram bins. As for GM, a great number of modifications are possible, however the best hint would be to further expand the graph representation to resemble the shock graph and use the algorithms for shock graph matching which are already proven to be highly accurate [15][16][19].

Scale invariance

However the basic methods used are rotation invariant, it would be good to make them scale invariant as well. For histograms this would simply mean to normalize the histogram, that is to divide the values in every bin either by the number of all samples used for computing the histogram or by the largest value stored in the histogram, whichever gives better results. This modification could influence the accuracy of the histogram similarity metrics and therefore requires more research. The graphs could be made scale invariant by performing a similar normalization of the radii of the circles stored in the vertices along with the normalization of the distances from parent vertices. This would involve modifying the graph construction algorithm but should probably not cause any significant changes to the graph matching algorithm, however to be sure more research needs to be conducted after the modifications are applied.

Segmentation

The uniform background for the images is a serious limitation which disables the OR system form taking real-life photographs as input. One proposed solution for overcoming this weakness is to use image segmentation [3]. In short, this technique finds regions of interest by dividing the plane into the background and the objects. Note that HM is ready to be used with real life photos (such tests were even conducted in the first approach with the neural network as the classifier but they were not documented). The segmentation has to be applied because of GM which requires the shape to be clearly separated from the background.

Other approaches for combining the basic methods

The hybrids built in this thesis used the same overall rule for combining the methods – they treated the methods as separate entities which performed the same task. Another approach would be to make them cooperate in a non competitive manner. One example of such an approach would be to use HM at the hypothesis

formation stage while GM would verify the hypotheses. In these terms, at present they are both hypothesis verifiers. Other basic methods could also be tried instead of or in addition to the ones implemented for this thesis.

Applications of the improved system

In the end, two ideas are given for application of the system which classifies objects. One idea is to embed it in a web search engine to allow for searching for images which actually depict what is being sought. Currently when a user types in a word, e.g. 'cat', for a web search and marks that he would like to find the images which depict the given object, the web search engine searches for image files which contain the given phrase either in the file name or in the tags attached to the file, but not in the actual image. This results in many files which do not actually contain the object the user wanted, but just happen to be named the same as the object. An intelligent approach would employ the object recognition techniques alike to the ones described in this thesis to provide reliable searches. The search could be performed on demand, but also the system could be used to tag image files in the background, so the search engine could use the ordinary tags assigned (but assigned by the OR system) to perform the searches faster.

Another idea is to use the system in robots that would take actions based on what they see. They could provide help for the disabled, e.g. by looking for objects in the surrounding and fetching them. Robots that can react to changes in the environment are highly desired in the industry and object recognition is of primary importance for detecting those changes and taking proper actions.

6 BIBLIOGRAPHY

[1] M. Bakircioglu, U. Gerander, N. Khaneja, M. I. Miller, *Curve matching on brain surfaces using induced Frenet distance metrics*, Human Brain Mapping, 6(5):329-331, 1998

[2] H. Blum, *Biological shape and visual science,* Journal of Theoretical Biology, 38:205-287, 1973

[3] E. R. Davies, *Machine Vision. Theory, Algorithms, Practicalities. 3rd edition*, Elsevier, 2005

[4] P. Dimitrov, C. Philips, K. Siddiqi, *Robust and efficient skeletal graphs,* IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head, SC, 2000

[5] L. Fausett, *Fundamentals of Neural Networks. Architectures, Algorithms, and Applications*, Prentice Hall, 1994

[6] P. F. Felzenszwalb, D. P. Huttenlocher, *Distance Transforms of Sampled Functions*, Cornell Computing and Information Science, TR2004-1963

[7] G. D. Finlayson, *Colour Object Recognition*, Master Thesis, Simon Fraser University, 1992

[8] D. A. Forsyth, J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, 2002

[9] B. V. Funt, G. D. Finlayson, *Color Constant Color Indexing*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 5, 1995

[10] S. Jeong, *Histogram-Based Color Image Retrieval*, Psych221/EE362 Project Report, 2001

[11] M. M. Kazhdan, *Shape Representations and Algorithms for 3D Model Retrieval*, PhD. Dissertation, Princeton University, 2004

[12] P. Klein, S. Tirthapura, D. Sharvit, B. Kimia, *A tree-edit-distance algorithm for comparing simple, closed shapes*, Symposium on Discrete Algorithms, 696-704, 2000

[13] S. M. Lee, J. H. Xin, S. Westland, *Evaluation of Image Similarity by Histogram Intersection*, COLOR research and application, Vol. 30, No. 4, 2005

[14] L. Chi Mai, *Introduction to Computer Vision and Image Processing*, Department of Pattern Recognition and Knowledge Engineering, Institute of Information Technology, Hanoi, Vietnam

[15] D. Marcini, A. Shokoufandeh, S. Dickinson, K. Siddiqi, S. Zucker, *View-Based 3-D Object Recognition using Shock Graphs*, IEEE 1051-4651/02, 2002

[16] D. Marcini, *Indexing and Matching for View-Based 3-D Object Recognition Using Shock Graphs*, Master Thesis, University of Toronto, 2003

[17] R. D. Schiffenbauer, *A Survey of Aspect Graphs*, Polytechnic University, Technical Report, TR-CIS-2001-01, 2001

[18] T. B. Sebastian, J. J. Crisco, P. N. Klein, B. B. Kimia, *Constructing 2D Curve Atlases*, 2000

[19] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, S. W. Zucker, *Shock Graphs and Shape Matching,* ICCV, 222-229, 1998

[20] M. J. Swain, D. H. Ballard, *Color indexing.* International Journal of Computer Vision, 7:11-32, 1991

APPENDIX A – THE DICTIONARY

This appendix is the dictionary of words and phrases which were used throughout the thesis. It is provided purely for convenience, as the same definitions can be found in the text of the thesis.

Class – a category which describes a group of objects that have a set of common features. The term 'class' and 'category' are used interchangeably throughout this thesis.

Color model – a way of representing color. A color can be described by a set of numbers, e.g. the RGB (Red, Green, Blue) model represents a color as a combination of three basic colors; the HSV model describes a color by its hue, saturation and value [3][8][14].

Feature – a characteristic of an object which can be used to distinguish an object from other objects of the same class [14] and also allows for assigning an object to a class.

Image – a two dimensional array of pixels. The pixels within one image are encoded using the same color model.

Junction points – the points of the skeleton which have 3 or more neighbors.

Object – an element which can be assigned to a class. Every object is characterized by a set of features.

Shape – for 2D space 'the bounded interior of a simple closed (Jordan) curve' [19]

Skeleton of a shape – the set of centers of the largest circles inscribed in the shape, as described by Blum [2]

Terminal points - the points of the skeleton which are adjacent to only one point.

Topological Signature Vector (TSV) – a vector representing the topology of a graph at a specific node [15]. The way of calculating TSV is described in section 3.2.

APPENDIX B – THE GALLERY

In this appendix some interesting images which were obtained as the half-products of the pre-processing for graphs are gathered. The purpose of the appendix is simply to constitute a proof that true beauty comes with computer vision, apart from the mathematical formulas.