A fully automated method for content-based color image retrieval is developed to extract color and shape content of an image. A color segmentation algorithm based on the k-mean clustering algorithm is used and a saturated distance is proposed to discriminate between two color points in the HSV color space. The feature set describing an image includes main object shape, which is extracted using the morphological operations. The computed image features are tagged within the image and a graphical user interface is presented for retrieving images based on the color and shape of the objects. The experimental results using natural color images demonstrate effectiveness of the proposed method.
Content-Based Color Image Retrieval

by

Audrey Varanguien de Villepin

A Thesis Submitted
to
Oregon State University

In Partial Fulfillment of
the requirements for the
degree of

Master of Science

Presented September 24, 1999
Commencement June 2000
Master of Science thesis of Audrey Varanguien de Villepin presented on September 24, 1999

Approved:

[Signature]

Redacted for Privacy

10-29-99

Major Professor, representing Electrical and Computer Engineering

[Signature]

Redacted for Privacy

Chair of Department of Electrical and Computer Engineering

[Signature]

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

[Signature]

Redacted for Privacy

Audrey Varanguien de Villepin, Author
ACKNOWLEDGMENTS

Thanks to Doctor Wojtek Kolodziej for advice and support all along this work. His assistance was key to the successful completion of this effort.

Thanks to Olivier Colle for his insight and suggestions for improving my work, as well as his guidance and motivation. Without him, this work would not have been possible.

I appreciate the people of Oregon State University for making my education possible, especially the faculty and staff of the Electrical and Computer Engineering department and the fellow students of the Modern Communication Center laboratory.

I thank my parents for supporting me and making it possible for me to study at Oregon State University.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER I : INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>I.1 Problem statement</td>
<td>1</td>
</tr>
<tr>
<td>I.2 Approach</td>
<td>2</td>
</tr>
<tr>
<td>I.3 Objectives</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER II : LITERATURE REVIEW</td>
<td>4</td>
</tr>
<tr>
<td>II.1 Previous work in color image segmentation</td>
<td>4</td>
</tr>
<tr>
<td>II.2 Previous work on shape feature</td>
<td>12</td>
</tr>
<tr>
<td>II.3 Previous work in color image retrieval systems</td>
<td>13</td>
</tr>
<tr>
<td>CHAPTER III : METHODS</td>
<td>16</td>
</tr>
<tr>
<td>III.1 Color reduction</td>
<td>17</td>
</tr>
<tr>
<td>III.2 Shape representation</td>
<td>41</td>
</tr>
<tr>
<td>CHAPTER IV : RESULTS</td>
<td>49</td>
</tr>
<tr>
<td>IV.1 Results</td>
<td>49</td>
</tr>
<tr>
<td>IV.2 User interface</td>
<td>52</td>
</tr>
<tr>
<td>CHAPTER V : CONCLUSIONS</td>
<td>59</td>
</tr>
<tr>
<td>V.1 Conclusions</td>
<td>59</td>
</tr>
<tr>
<td>V.2 Future work</td>
<td>60</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>62</td>
</tr>
<tr>
<td>APPENDICES</td>
<td>65</td>
</tr>
<tr>
<td>APPENDIX A : SATURATED DISTANCE</td>
<td>66</td>
</tr>
<tr>
<td>APPENDIX B : MULTIDIMENSIONAL GAUSSIAN LAW</td>
<td>69</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2-dimensional image space and 3-dimensional color space</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>Non perceptual uniformity of the RGB color space</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>HSV cone</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>Skeleton of the k-mean algorithm</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>Maximization of the minimum distance</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>Saturated distance</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>Saturated distance concept</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>Process to find the segmentation threshold</td>
<td>27</td>
</tr>
<tr>
<td>9</td>
<td>C1, curve of the percentage of misclassified color points</td>
<td>28</td>
</tr>
<tr>
<td>10</td>
<td>C2, curve of the percentage of color points switching class</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>Color segmentation flow chart</td>
<td>31</td>
</tr>
<tr>
<td>12</td>
<td>Algorithm to find the 2 dominant color points of a class</td>
<td>34</td>
</tr>
<tr>
<td>13</td>
<td>Class encapsulation in ellipsoids</td>
<td>36</td>
</tr>
<tr>
<td>14</td>
<td>Example of a class segmentation using the 2 dominant color points</td>
<td>37</td>
</tr>
<tr>
<td>15</td>
<td>Gaussian distribution</td>
<td>38</td>
</tr>
<tr>
<td>16</td>
<td>Eigenvectors coordinate system in 2 dimensions</td>
<td>39</td>
</tr>
<tr>
<td>17</td>
<td>Position of the two dominant color points</td>
<td>41</td>
</tr>
<tr>
<td>18</td>
<td>Shape feature determining algorithm</td>
<td>43</td>
</tr>
<tr>
<td>19</td>
<td>Image test</td>
<td>45</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES, CONTINUED

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Graphical user interface for the image tagging</td>
<td>54</td>
</tr>
<tr>
<td>21</td>
<td>Graphical user interface for the basic image retrieval</td>
<td>56</td>
</tr>
<tr>
<td>22</td>
<td>Graphical user interface for the advanced image retrieval</td>
<td>57</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>46</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>51</td>
</tr>
</tbody>
</table>

1. Computing time of a segmentation iteration
2. Percentage of color points switching class threshold
3. Class selection to represent the image main object
4. Color reduction results
5. Image main object representation results
To Olivier and my parents
CHAPTER I: INTRODUCTION

1.1 Problem statement

The goal is to design an automated tool capable of tagging color images with their dominant color and shape content. The general motivation is content-based image retrieval, which aims to classify a set of images depending on their content. The users of such a system search for images with specific features such as color and shape. Instead of manual classification (e.g. description with key words), an automated tool computes the characteristic image features that will be used to retrieve automatically images with a high degree of accuracy.

Downloading large images over a computer network requires a considerable bandwidth. Thus, the motivation behind the development of content-based image search systems is to find an image meeting a specific criterion without having to manually browse through a large set of images. Images need to be found by an automated tool according to criteria specified by the user, similarly to the text search engines over the Internet, which are already available.

An image search system is based on the images content, as the text search systems are based on the words of the text. Errors are likely to occur if a manual description of an image is performed. The human perception is subjective and various users can describe the same image differently. Thus the retrieval performance depends on the person who described the image. Moreover words, in general, describe the images poorly. For example, the image colors cannot be described by simple words like red, blue or yellow.
Many hues of these colors exist and describing them by words is harder than assigning a numerical value to a color.

1.2 Approach

Basically, all the existing content-based image retrieval systems are based on the same principles:

- Preprocess each image of the database by applying algorithms to identify the image features such as color, shape, and texture.
- Translate the user inputs into a feature vector, which is to be compared to the features computed in the previous step.
- Compute the distances between the user's query feature vector and the preprocessed image feature vectors.
- Sort and display the images by distance in increasing order.

The first step is usually the most time consuming. However, it can be done offline during the pre-processing of the image set.

The difficult issues in an automated image retrieval system are computing image features and finding metrics, which are coherent with the human perception.

While designing a content-based image retrieval system, the first step is to choose the features to describe the images. In this thesis, color and shape allow the user to search for a specific image. Some of the existing content-based retrieval tools also include the texture feature.
1.3 Objectives

The objective is to design an efficient, content-based image retrieval system based on the color and shape features of an image. The first step is to design and implement algorithms to compute these features. The second step is to design a graphical user interface that allows specifying criteria for image retrieval. The final step consists of metrics design to compare the user query with the image features.

The thesis organization is as follows. Chapter 2 gives a literature review on the state-of-the-art content-based image retrieval systems which use color image segmentation, shape feature representation and retrieval techniques. Chapter 3 presents the methods used here to compute the color and shape features and to find the metrics to match the user query with the image features. Chapter 4 presents the results and describes the graphical user interface. Finally, in chapter 5 conclusions and recommendations for future work are presented.
CHAPTER II: LITERATURE REVIEW

This section reviews some methods used in color image segmentation and shape detection. To illustrate the current state of research on color image retrieval, some retrieval systems are presented.

II.1 Previous work in color image segmentation

Color image segmentation requires several design choices need to be made: the color space in which the segmentation will be computed, the distance used to compute the difference between two colors and finally the segmentation algorithm.

II.1.1 Color spaces

The color of an image pixel is usually represented by three coordinates. A color space is defined by mapping the pixels into points in a 3-dimensional space, each axis corresponding to a color component. Figure 1 shows the connection between the 2-dimensional image space and the 3-dimensional RGB color space. The device coordinates of an image data are often red (R), green (G) and blue (B) but the RGB space has the major disadvantage of not being perceptually uniform. Two different colors in one part of the color space do not exhibit the same degree of perceptual difference as two colors in another part of the color space, even though they are the same distance apart (Figure 2). Therefore, the first step is to find a transformation that maps the input image data from device coordinates into a proper color space.
Different color spaces are used depending on the application and the segmentation method employed. This choice is critical to the color classification method as the arrangement and shape of clusters depend on the color coordinate system. Some color spaces standards of the Comission Internationale de l'Eclairage (CIE) such as CIE
L*a*b* and CIE L*u*v* offer improved perceptual uniformity. They represent with equal emphasis the three variants that characterize color: hue, lightness and saturation and are used in applications that mimic human perception, [1, 2]. However, these color spaces are often inconvenient to use due to the non-linearity and computational complexity of transformation to and from the RGB space.

In [3, 6, 12], a preferred non-linear but easily invertible transformation is used: the transformation from RGB to the hue (H), saturation (S) and value (V) color space. The use of this color space improves the perceptual uniformity. In particular [3] uses “preprocessed” HSV values where the saturation and value are used to determine which regions of the image are achromatic. The colors with Value lower than 25 are classified as black and the colors with Saturation higher than 20 and Value lower than 30 are classified as white. The remaining color points fall in the chromatic region of the HSV cone and their Hue components are segmented.

A different approach is used in [4] based on a normalization model of the human visual system that incorporates color perception. After conversion to opponent-colors space, each of the resulting three components is subjected to a perceptual decomposition. The principal components of the opponent-colors space are black-white, red-green and blue-yellow. This color space matches the psychological experiment results, which suggest that the human visual representation of simple colored patterns is pattern-color separable.

The following color spaces: RGB, HSV, Lab, XYZ and YIQ are compared in [7]. They conclude that no color space is good for segmenting many different images. In their experiments, the HSV color space produces the worst results because of the singularity
problem inherent to this space. When the RGB values of a color point are identical, the hue (H) of this point is undefined and it is assigned the value of zero. Thus the algorithm cannot distinguish between two color clusters having equal R, G and B values, for example between a dark gray cluster (R=10, G=10, B=10) and a lighter one (R=50, G=50, B=50).

II.1.2 Distance metrics

The distance measure is a key point in clustering. [2] suggests that the measure should be consistent with the visual resolution and uniform in all directions in the entire color space. Thus, the Euclidean distance is used to compute the distance between two colors in the CIE 1976 uniform color space (L* u* v*). In non-uniform color spaces following this concept is challenging, however many application specific metrics exist.

In [3] different vector distance measures in the RGB color space are investigated. In the following equations the notation $d(i, j)$ refers to the distance between two color points respectively $x_i$ and $x_j$.

1. The generalized Minkowski metric:

$$d_M(i, j) = \left( \frac{1}{M} \sum_{k=1}^{p} \left| x_i^k - x_j^k \right|^M \right)^{\frac{1}{M}}, \quad (\text{Eq II-1})$$

Where $p$ is the dimension of the vector $\bar{x}_i$, and $x_i^k$ is the $k^{th}$ element of $\bar{x}_i$.

Three special cases of the LM metric are of particular interest, namely, $M=1, 2, \infty$. 
2. The Canberra distance:

\[ d_c(i, j) = \sum_{i}^{p} \frac{|x_i^k - x_j^k|}{x_i^k + x_j^k} \]  

(Eq 11-2)

Where \( p \) is the dimension of the vector \( \bar{x}_i \), and \( x_i^k \) is the \( k^{th} \) element of \( \bar{x}_i \).

The Canberra metric applies only to non-negative data, for example color vectors described in the RGB coordinates.

3. The Czekanovski coefficient:

\[ d_z(i, j) = 1 - \frac{2 \sum_{k=1}^{p} \min(x_{ik}, x_{jk})}{\sum_{k=1}^{p} (x_{ik} + x_{jk})} \]  

(Eq 11-3)

As the Canberra distance, this distance is only applicable to vectors with non-negative components.

4. The angular distance:

\[ \theta = 1 - \frac{2}{\pi} \cos^{-1}\left(\frac{\bar{x}_i \cdot \bar{x}_j}{|\bar{x}_i||\bar{x}_j|}\right) \]  

(Eq 11-4)

and \( |\bar{x}_i| = |\bar{x}_i \cdot \bar{x}_j|^{1/2} \) where \( \bar{x}_i \cdot \bar{x}_j \) denotes the inner product of \( \bar{x}_i \) and \( \bar{x}_j \).

Similar colors have almost parallel orientations, different colors will point in different directions in the RGB color space. Thus, the angular distance is a meaningful measure.
5. A new distance measure which combines the angle between two vectors and their magnitude difference:

$$d_N(i, j) = 1 - \frac{2}{\pi} \cos^{-1} \left( \frac{\bar{x}_i \cdot \bar{x}_j}{\| \bar{x}_i \| \| \bar{x}_j \|} \right) \left[ 1 - \frac{\bar{x}_i - \bar{x}_j}{\sqrt{3.255^2}} \right].$$  \hspace{1cm} (Eq II-5)

In [5], a computationally simple approach is used. An algebraic composite of coordinate difference is replaced by a color coordinate maximum metric (maxRGB). The distance between two color points is given by their maximum (normalized) coordinate difference.

II.1.3 Segmenting methods

Image segmentation techniques can be roughly classified into four categories, the histogram-based segmentation, the neighborhood-based segmentation, the physically-based segmentation and the color clustering based segmentation methods.

For color classification, [1] extracts the color clusters by an iterative analysis of one-dimensional histograms. Color clusters detection depends on the spatial arrangement of the clusters in the color space. If the clusters do not overlap, a one-dimensional histogram in a space such as RGB or HSV can distinguish them. Otherwise, the clusters cannot be distinguished in these spaces and image needs to be transformed into a space where the clusters will not overlap. Thus the principal component method id used. The principal component coordinates are computed using the eigenvalues and eigenvectors of the covariance matrix of the color points. The colors are transformed into the principal component coordinate system and one-dimensional histograms are computed for the first
and second components. Then the histogram peaks in the histogram determine the main colors of the cluster.

[2] uses the RGB histogram to find color peaks and discards small (low number of color points) peaks. Each color point is assigned to the closest peak using the Euclidean distance in the L*u*v* color space. In [10], the 3 one-dimensional color histograms are computed in the RGB color space and stored for the future retrieval.

The second category of segmentation technique is the neighborhood-based approach [7]. Their multiresolution color image segmentation algorithm uses Markov random fields. The most important feature of the Markov random fields is that the probability of a particular site to assume a certain value depends only on its neighbors, and not on the whole image.

Another type of segmentation techniques is based on a clustering algorithm like the k-means clustering algorithm [8, 12, 13], also called the generalized Lloyd algorithm. Its purpose is to find the most significant colors of an image or a region. The outline of the algorithm is as follow:

1. Choose $k$ initial centers in the color space. The choice can be random or calculated from the image.
2. Cluster each color point as belonging to the nearest neighbor center.
3. Compute the new center of each cluster as the cluster mean.
4. Repeat 2 and 3 until the number of color points switching cluster is zero or is less than a fixed threshold.

After this segmentation, the image is divided into $k$ clusters or color point sets of various sizes. Small classes are considered as noise [12] and a minimum cluster size
threshold is set to filter them out. The points belonging to these regions are merged with the largest neighboring region.

This algorithm can be applied in different spaces. In [12], the clustering is performed in a 3 dimensional color space whereas, in [13], it is performed in the 4 dimensional color-texture sub-space to add the texture information.

The K-means algorithm is used in our work and modified to yield a better segmentation performance.

II.1.4 Other color features

In order to include spatial information with color, a new color feature called the color correlogram is proposed in [9] for image indexing and retrieval. The highlights of this feature are:

- Spatial correlation of colors.
- Global distribution of local spatial correlation of colors.
- Easy computation and fairly small feature size.

Informally color correlogram of an image is a table indexed by color pairs, where the $k^{th}$ entry for $i$ and $j$ specifies the probability of finding a point of color $j$ at a distance $k$ from a point of color $i$ in the image. Such image feature tolerates large changes in appearance of the same scene caused by changes in viewing positions, changes in the background scene (e.g. partial occlusions, camera zoom).
11.2 Previous work on shape feature

The second, after color, feature which is mainly used in content-based image retrieval systems is the shape of the objects contained in the image. Some of the methods used to describe this feature are presented in this section.

The shape representations proposed in the literature are not generally invariant to large variation of image size, position and orientation. In order to incorporate invariance to rigid motions (rotation and translation) these methods need to be applied for all possible motions, therefore reducing the image retrieval speed. When considering large image databases, the speed decrease may become significant. Consequently, some of the methods try to identify shape features which are either invariant to rotations and translations or which can be efficiently computed for a number of possible rigid motions.

Accordingly, [10] describes the shape information contained in an image on the basis of its significant edges. Edge direction histogram generated using the Canny edge operator represents the shape attribute. Shape-based retrievals are performed using a histogram intersection technique. This method has the advantage to be invariant to translation in the image and it identifies the general shape information. However it has several disadvantages:

- Edge directions are neither scale nor rotation invariant.
- Edge directions are affected by the nature of the edge, for example, black on white or white on black. These two types of edges differ in their directions by 180°.
• The matching results depend on the number of bins used to represent the shape histogram of the edge directions. Commonly 36 bins are used each spanning 10°.

Also, in order to increase retrieval accuracy, the results obtained from query based on individual features are integrated. They represent similarity values between the query and the retrieved image. The results of the shape-based retrieval and the color-based retrieval are integrated using a weighted average.

Different “eigenrepresentation” is used in [17] where each shape is characterized by the solution of an eigenvalue problem related to its intimate structure. These “eigenrepresentations” are used as a basis for correspondence and recognition. To obtain a representation of an object from its description by a set of points, they use its deformation modes, computed by a finite-element method approach. The idea is to characterize an object by considering it as a set of mass points, connected by elastic relations. A prealignment of the shapes from the query image and from the database images is computed by a fast-moment of inertia method, which provides various degrees of invariance with respect to rotation, scale and size of the objects to compare.

II.3 Previous work in color image retrieval systems

Several systems based on color, shape and texture attributes have been recently developed to search through image databases. This includes QBIC [18], Photobook [15], Virage, WebSeek [16], Candid [14], Netra [13]. This section is not meant to be an exhaustive survey, but rather a review of selected systems to illustrate the current state of research.
The QBIC (Query by Image Content) system [18] allows queries on large image and video databases based on:

- Example images,
- User-constructed sketches and drawings,
- Selected color and texture patterns,
- Camera and object motion, and
- Other graphical information.

The QBIC system has two main components: database population and database query. During the population process, image and video features such as color, texture, shape and motion are extracted and stored in the database. The processes of population and retrieval are not done over the Internet.

The WebSeek system [16] collects image and videos on the World Wide Web with an automated agent, processes them in both text and visual feature domains, catalogues and indexes them for fast search and retrieval. The search engine utilizes both text-based navigation and content-based technology for searching visually through the catalogued images and videos. The complete system possesses several functionalities:

- Searching using content-based techniques,
- Query modification using content-based relevance feedback,
- Automated collection of visual information,
- Compact representation of images and videos for displaying query results,
- Image and video subject search and navigation,
- Text-based searching,
• Search results lists manipulations such as intersection, subtraction and concatenation.

In the CANDID (Comparison Algorithm for Navigating Digital Image Databases) system [14], a global signature describing the texture, shape or color content for every image stored in the database is computed. A normalized distance between probability density functions of feature vectors is used to match signatures. This method is used to retrieve images that are similar to an example target image.

The Photobook system [15] is a set of interactive tools for browsing and searching images and image sequences. Direct search on image content is made possible by use of semantics-preserving image compression, which reduces images to a small set of perceptually-significant coefficients. Among the Photobook descriptions are:

• Appearance,
• 2-dimensional shape,
• Textural properties.
CHAPTER III: METHODS

This chapter describes the methods used to design an image retrieval system based on color and shape content. A content-based retrieval system efficiency is measured by successful matches between the image features and user specifications. This requires a definition of suitable image features, a feature extraction method, and a feature distance measure that can be computed in the real time. The following properties are required:

- Image features agree with the human visual perception in the sense that retrieved images correspond to the user expectations.
- The distance measure is computationally simple and suitable for fast online matching.
- Compact representation of the features reduces memory and disk space requirement.
- Features have suitable separation properties, which are robust to noise and allow positive discrimination.

With these properties in mind, this section presents the methods selected to extract the features and to retrieve matching images. The color feature is extracted using a clustering algorithm, namely the k-mean or generalized Lloyd algorithm. The shape of the image main object is computed after performing morphological operations on each color cluster and is characterized by perimeter and area.
III.1 Color reduction

The first feature extracted from an image are the dominant color classes. The developed method computes the image main color classes in the HSV color space using the k-mean algorithm.

III.1.1 Color space

The image formats used in this research are 24-bits RGB bitmaps and JPEGs. The RGB space has the major disadvantage of not being perceptually uniform. This means that two different colors in one part of the color space will not exhibit the same degree of perceptual difference as two colors in another part of the color space, even though they are the same "distance" apart. Therefore, the first step is to find the transformation to map the input image data into the proper color space. As mentioned in the literature review, different color spaces are being used depending on the application and the segmentation method employed. Since the composition and shape of color clusters depend on the color coordinate system, the choice of the color space is critical.

Color spaces such as CIE L*a*b* and CIE L*u*v* offer improved perceptual uniformity, as compared to RGB, they represent with equal emphasis the three variables that characterize color: hue, lightness and saturation. However, these color spaces are inconvenient due to the complexity and non-linearity in forward and reverse transformations from the RGB color space. The transformation from RGB to HSV color space is non-linear, however easily invertible. The HSV color space is selected here since it is more perceptually uniform than the RGB space. The images are converted from RGB to HSV using the Mathworks Matlab function “rgb2hsv”.
The model of HSV color space is cylindrical, however usually it is represented as a cone or a hexagonal cone, as displayed in Figure 3. The vertical axis represents the Value (V) ranging from 0 corresponding to dark colors to 1 corresponding to bright colors. The Saturation (S) ranges from 0 at the V-axis to 1 and contains the color richness. The angle Hue (H) with the red color at 0° represents the colors.

![Figure 3: HSV cone](image)

The transformation function from RGB to HSV returns three components ranging from 0 to 1. For further computation, they are scaled by a factor equal to 255. Thus each pixel of an image is represented by three color components H, S and V in a 3-dimensional cube. The HSV cube represents color only and does not contain any spatial information, as shown in the literature review section, Figure 1.
III.1.2 Color segmentation algorithm

Figure 4 presents the steps to compute the k-mean clustering algorithm in \( k \) classes. The number of classes \( k \) has to be determined prior to performing the algorithm. This critical choice is discussed in III.1.3.

Some changes made to the algorithm skeleton will be explained in this chapter. The following details each step of the algorithm.

III.1.1.2 Initial centers computation

The initial centers need to be chosen to obtain a suitable initial color separation. Thus, the initial \( k \) centers are computed to be the \( k \) furthest apart color points of the image. In this work, the initial center computation using the following algorithm is performed once at the beginning of the segmentation with only two centers. However, to generalize the algorithm, the process to compute more than 2 initial centers is explained:

1. Select the initial reference center as being the center of the 3-dimensional HSV cube.

2. Compute \( g_1 \), the furthest color point from the initial reference center.

3. Choose \( g_2 \), the furthest color point from \( g_1 \).

4. Choose successively \( g_i \), where \((i=3, 4, \ldots, k)\), the furthest color point from the previous \( g_j \), where \((j=1, 2, \ldots, i-1)\). When more than two points are considered, a definition of "furthest" needs to be set. A color point \( p_c \) is defined as being the furthest point from all \( g_j \) if the minimum of the distances \( d(p_c, g_j) \) is the maximum among all the minimum distances of \( d(p, g_j) \), \( p \) being any image color point.
Step 1:
Compute the set of initial classes centers $g_i$, where $i=1, 2, \ldots, k$.

Step 2:
Assign the current color point to the closest class center $g_i$.

Step 3:
Update the corresponding class center $g_i$ as being the class mean.

Last color point?

Yes

The % of color points switching class is above a threshold?

Yes

No

End

Color point counter

Figure 4: Skeleton of the k-mean algorithm
Thus, the point $g_i$ are calculated using equation:

$$g_i = \arg \max_{p \in I} \left\{ \min_{j \in [1..i-1]} d(p, g_j) \right\},$$  \hspace{1cm} (Eq III-1)

Where $I$ is the set of the image color points and $d$ the distance between two color points. The next sub-section describes the measure used to compute the distance $d$ in the 3-dimensional HSV color space.

Figure 5 presents an example where three color points $p_1$, $p_2$ and $p_3$ compete to be the furthest point from $g_1$ and $g_2$.  

$$\begin{align*}
\min(d(p_1,g_1), d(p_1,g_2)) &= d(p_1,g_1), \\
\min(d(p_2,g_1), d(p_2,g_2)) &= d(p_2,g_2), \\
\min(d(p_3,g_1), d(p_3,g_2)) &= d(p_3,g_2), \\
\max(d(p_1,g_1), d(p_2,g_2), d(p_3,g_2)) &= d(p_2,g_2).
\end{align*}$$

Figure 5: Maximization of the minimum distance

III.2.1.2. Classification of the image color points

For each color point, the distances to previous centers are calculated. The color point is assigned to the class with the closest center. The measure used to calculate the
distance between two color points is called the “saturated distance” (see Figure 6). The term “distance” is used here informally, since it does not satisfy the distance requirements as shown in Appendix B.

\[ d_{Sat} = f(d) \]

\[ d_{Sat} \] Saturated distance of one coordinate (H, S, or V) between two points.

\[ d \]: H, S or V coordinate difference.

\[ d_{Max} \]: Maximum Value.

\[ D \]: Dead Point.

\[ Sat \]: Saturation Point.

**Figure 6: Saturated distance**

For S and V components, \( d \) represents the absolute value of the coordinate difference. The Hue component is an angle between 0 and 255, thus \( d \) is the smallest positive angle value for H component. The three values of \( d, d_{SatH}, d_{SatS}, d_{SatV} \), are added to compute the final distance between two color points:

\[ d_{Sat} = d_{SatH} + d_{SatS} + d_{SatV} \]  \hspace{1cm} (Eq III-2)
The value $d_{Max}$ is the maximum saturated distance between the coordinate H, S or V of two color points. The saturated distance function has three zones:

1. The dead zone (from 0 to $D$) groups the close points by setting the distance between them to zero.

2. The linear zone (from $D$ to $Sat$) computes the distance using the following equation:

\[
    d_{Sat} = \frac{d}{(S-D)} \times d - \frac{d_{Max} \times D}{(S-D)} .
\]

(Eq III-3)

3. The saturation zone saturates the distance by setting the distance equals to the limit value $d_{Max}$.

Each time a color point is assigned to a class, the class center is recalculated to be the gravity center of the augmented by this new color point. Using the Euclidean distance makes the center position sensitive to the color “outliers”. The saturated distance is chosen similarly to standard statistical robust measures. Figure 7 illustrates the basic properties of the saturated distance.
The center has moved noticeably: it does not represent well the set of points $p_i$ (with $i = 1, 2, 3, 4$).

The center does not move away from the class when a far point is assigned. It represents better the color of the class.

**Figure 7: Saturated distance concept**

Computing the saturated distance between two color points requires suitable choice of $d_{Max}$, $D$ and $Sat$ defined separately for each component H, S and V. A main advantage of this approach is that different importance can be given to each component. The purpose of this separation of HSV components is to determine the main image colors, which match a user request. When a user specifies a color, usually the Hue is selected first and then the Saturation and Value are chosen. Therefore the most important part of the request is the Hue component and Hue should be given more importance in the process of segmentation. The values chosen for $d_{Max}$ are 10 for the Hue component, 1 for the Saturation and the Value components.
The color point coordinate difference ranges from 0 to 255 for S and V and from 0 to 128 for H. The dead point is fixed at 0 for the three components. The saturation point is fixed at 100 for S and V and 50 for H.

**III.3.1.2. Segmentation threshold**

The standard k-mean algorithm [12] stops when no color point switches class. This may require a significant computing time due to the number of iterations needed. Usually, after a few iterations, most image color points have an accurate classification, with only a few color points left missclassified. Table III.1 shows that the computing time depends on the image size by $\theta(n)$, $n$ being the number of pixels. When the size of the image is doubled, computing time experimentally quadruples. While working with large images, the computing time becomes significant, thus a threshold needs to be set to stop the iterations when the image segmentation is satisfactory.

<table>
<thead>
<tr>
<th>Image Size</th>
<th>Computing Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1 = 48 \times 36$</td>
<td>$t_1 = 1 \text{ s}$</td>
</tr>
<tr>
<td>$N_2 = 96 \times 72$</td>
<td>$t_2 = 4 \text{ s}$</td>
</tr>
<tr>
<td>$N_2 = 4 \times N_1$</td>
<td>$t_2 = 4 \times t_1$</td>
</tr>
<tr>
<td>$N_3 = 192 \times 144$</td>
<td>$t_3 = 17 \text{ s}$</td>
</tr>
<tr>
<td>$N_3 = 4 \times N_2$</td>
<td>$t_3 \sim 4 \times t_2$</td>
</tr>
<tr>
<td>$N_4 = 384 \times 288$</td>
<td>$t_4 = 70 \text{ s}$</td>
</tr>
<tr>
<td>$N_4 = 4 \times N_3$</td>
<td>$t_4 \sim 4 \times t_3$</td>
</tr>
</tbody>
</table>

*Table 1: Computing time of a segmentation iteration*
Thus the number of iteration needs to be decreased. While segmenting, for each iteration, the percentage of color points switching class can be calculated, whereas the percentage of missclassified color points remains unknown. If the latter was known, there would be of course no need for finding a threshold. Thus, a relationship needs to be determined between these two percentages to set a threshold depending only on the percentage of color points switching class. The flow-chart in Figure 8 presents the process of plotting curves that will be used for finding a suitable threshold.

Examples of the curves $C_1$ and $C_2$ are displayed in Figure 9 and Figure 10. Although the curves are different for each image, the same patterns are found for all images.

The percentage of missclassified color points decreases quickly until it reaches a value under 1% (which is a tolerable difference) and then it converges slowly to 0. The limit point $L$ of curve $C_1$ is the curve point where the decreasing speed changes. The corresponding number of iterations is equal to 3. Now, consider curve $C_2$ representing the percentage of color points switching class.

During the third iteration, 1.6% of color points have switched class. After that, the percentage decreases continuously to reach zero after eleven iterations. According to this example, the threshold should be set at 1.6% of color points switching class and the segmentation would stop after three iterations instead of eleven.
Segment the original image until no color point switch class → "reference image".

Start the segmentation on the original image.

Perform a segmentation iteration.

Calculate and store for further analysis the percentage of color points missclassified regarding to the reference image.

Calculate and store for further analysis the percentage of color points switching class during this iteration.

One color point switched class?

Yes

No

Plot the curve $C_1$ of the percentage of color points missclassified depending on the number of iterations.

Plot the curve $C_2$ of the percentage of color points switching class depending on the number of iterations.

Figure 8: Process to find the segmentation threshold
Percentage of missclassified color points = f (Number of iterations)

Figure 9: C1, curve of the percentage of missclassified color points

Percentage of color points switching class = f (Number of iterations)

Figure 10: C2, curve of the percentage of color points switching class
Table 2 gives the results of 10 experiments. Images used have different sizes and content. Each image is segmented in two classes, the process described by the flow chart of Figure 8 is performed to plot curves, which are studied as above. The table contains:

- The number of iterations corresponding to the limit point $L$.
- The percentage of color points switching class corresponding to this number of iterations.

The percentage minimum of color points switching class is 0.8%. By setting the threshold to 0.7%, we ensure in most cases that less than 1% of color points are missclassified.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Corresponding percentage of color points changing class</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.6 %</td>
</tr>
<tr>
<td>5</td>
<td>0.9 %</td>
</tr>
<tr>
<td>4</td>
<td>1.9 %</td>
</tr>
<tr>
<td>4</td>
<td>1.1 %</td>
</tr>
<tr>
<td>4</td>
<td>1.3 %</td>
</tr>
<tr>
<td>5</td>
<td><strong>0.8 %</strong></td>
</tr>
<tr>
<td>3</td>
<td>1.9 %</td>
</tr>
<tr>
<td>5</td>
<td>0.9 %</td>
</tr>
<tr>
<td>4</td>
<td>1.5 %</td>
</tr>
<tr>
<td>4</td>
<td>1.2 %</td>
</tr>
</tbody>
</table>

**Table 2: Percentage of color points switching class threshold**
III.1.3 Selecting the number of classes

A critical choice in applying the k-mean segmentation algorithm is the number of colors. Since image color contents vary, choosing a-priori fixed number of classes is not suitable. Thus, an adaptive method is used here. The image is first segmented into two classes. Each class is then studied separately. If the class color standard deviation is under a threshold, the class is “compact” and it is not segmented any further. Otherwise, the class is classified as “non-compact” and it is divided into two separate classes. The initial class centers are computed to be the two furthest apart image colors points. Whereas, when dividing each class, the two initial centers are computed using additional statistics. The recursion applies to the subsequent class division. The segmentation stops when the number of classes reaches the maximum value or when all the classes are identified as compact. Since the purpose of the segmentation is to extract the image main colors, the maximum number of classes is set here to 8. The flow chart of Figure 11 shows the entire algorithm and the specific steps of determining if a class is compact and finding the new centers.
Segment the image in 2 classes.

Add the 2 classes to the class counter.

Class Counter

Compute the class standard deviation in the 3-dimensional space HSV using the saturated distance.

Compact class?

Total number of classes = 8?

No

Compute the 2 dominant class color points \((g_1, g_2)\).

Segment the class in 2 classes with the initial centers being \(g_1\) and \(g_2\).

Add the 2 classes to the class counter.

Last class in counter?

Yes

End of Segmentation.

No

Figure 11: Color segmentation flow chart
III.1.1.3. Compactness threshold selection

Defining the class “compactness” is done by computing the color standard deviation in the HSV space using the saturated distance as a measure.

\[
\sigma = \left[ \frac{\sum_{p \in I} \left( d_{Sat}(p - g) \right)^2}{n} \right]^{1/2}, \quad \text{(Eq III-4)}
\]

Where \( I \) is the set of the image color points \( p \), \( n \) the total number of color points, and \( g \) the class center. A small standard deviation may indicate that the class color points are close to the class center and form a compact class. A large standard deviation may indicate that the color points are spread out around the class center and many different colors are present in this class. Defining the small and large standard deviation is done by setting a compactness threshold. This means finding a threshold to distinguish two colors in the HSV color space. The colors with distance less than the compactness threshold are considered identical.

Although being more perceptually uniform than the RGB color space, the HSV color space is not completely uniform. Two different colors in one part of the color space will not exhibit the same degree of perceptual difference as two colors in another part of the color space, even though they are the same "distance" apart. Thus finding a fixed threshold for the visual differentiation of two colors is not possible. Selecting too large threshold may not differentiate visually distinct colors, in some part of the HSV space. Selecting too small threshold may lead to the overextensive segmentation. The latter may
cause reaching the maximum class count when some color classes are segmented too fine
and others are not segmented properly.

By definition, the saturated distance between two color points is the sum of each
coordinate distance. The maximum distance for each coordinate equals to its weight
which is 10 for Hue, 1 for Saturation and 1 for Value. Thus, using the Saturated Distance
definition, maximum distance between two color points in HSV space is equal to 12. The
threshold is set experimentally to be the fourth of the maximum distance between two
color points, that is 3.

**III.2.1.3. Selection of initial centers for subsequent segmentation**

In contrast to the first step of image segmentation where the initial centers are
computed to be the furthest image color points, the subsequent segmentations use
different initial centers. These initial centers are chosen to give a more efficient class
segmentation. If a class is not compact, the algorithm segments it in two classes. Two
dominant colors exist in this class and the algorithm described in Figure 12 identifies
them.
Compute the 3x3 color covariance matrix $COV$.

Compute the eigenvectors $V_i$ where ($i = 1, 2, 3$) and the eigenvalues $\lambda_i$ of $COV$. To obtain the class principal axis.

Compute the class barycenter $c$ using the Euclidean distance.

Transform the class points into the coordinates defined by $V_i$ with center $c$.

Compute the covariance matrix $COVTRANS$ in the new coordinate system.

Find the largest eigenvalue $\lambda_M$.

Take the corresponding eigenvector: $V_M$.

From $COVTRANS$, take the variance $\sigma^2_M$ corresponding to $V_M$.

Compute the 2 dominant color points ($g_1, g_2$) being at $1.2\sigma_M$ from $c$ on $V_M$.

Figure 12: Algorithm to find the 2 dominant color points of a class
The following details the steps of calculating the two dominant color points of the class which will be used as initial centers for the class subsequent segmentation. The dominant points are determined using some statistics.

Compute the covariance matrix of a class in the HSV color space:

\[
COV = \begin{bmatrix}
\sigma_H^2 & K_{SH} & K_{VH} \\
K_{SH} & \sigma_S^2 & K_{SV} \\
K_{VH} & K_{SV} & \sigma_V^2
\end{bmatrix},
\] (Eq III-5)

With \(H, S\) and \(V\), the sets containing the Hue, Saturation and Value coordinate of all the class color points.

With \(\sigma_H^2\) the variance of the Hue coordinates,

\(\sigma_S^2\) the variance of the Saturation coordinates,

\(\sigma_V^2\) the variance of the Value coordinates.

With \(K_{SH}\) the covariance between the \(S\) and \(H\) coordinates,

\(K_{VH}\) the covariance between the \(V\) and \(H\) coordinates,

\(K_{SV}\) the covariance between the \(S\) and \(V\) coordinates.

The class points can be encapsulated in a 3-dimensional ellipsoid as shown in Figure 13. The points of a 3-dimensional ellipsoid are described by this matrix:

\[
(p - p_0)^T COV^{-1} (p - p_0) = 1,
\] (Eq III-6)

With \(p \in HSV\) and \(p_0\) the ellipsoid center.

The ellipsoid axes directions and lengths are determined using the eigenvalues and eigenvectors of the class covariance matrix. The axis directions are the eigenvectors
with the principal axis being the eigenvector corresponding to the largest eigenvalue. The two dominant color points are chosen to be on this principal axis.

The image has been segmented into 2 classes and the classes are encapsulated in 3-dimensional ellipsoids.

**Figure 13: Class encapsulation in ellipsoids**

Figure 14 shows an example of finding the 2 dominant color points of a class and corresponding results.
The view is the one of the two principal axis of the 3-dimensional ellipsoid representing the class.

**Figure 14: Example of a class segmentation using the 2 dominant color points**

Determining the position of the dominant points on the principal axis is done by calculating their distance from the class center, so that this distance equals to a fixed number. Since most of the color points are encapsulated in the 3-dimensional ellipsoid without a few being far away from the ellipsoid, we assume that the points have a similar distribution than the Gaussian distribution. Thus, this distance value is determined using the multidimensional Gaussian law.

In one dimension, the probability that a sample from a Gaussian distribution with mean $\mu$ and standard deviation $\sigma$ will fall within two standard deviation range from $\mu$ is:

$$P(\mu - 2\sigma \leq x \leq \mu + 2\sigma) = 0.9544$$  \hspace{1cm} (Eq III-7)

Hence over 95% of the samples will fall within $2\sigma$ and the mean $\mu$ as shown in Figure 15.
Appendix B shows that for the 3-dimensional Gaussian law, similar approximation can be done for each dimension:

\[ P(\mu_i - 2.4\sigma_i \leq X_i \leq \mu_i + 2.4\sigma_i) = 0.95, \quad (\text{Eq III-8}) \]

where \( X_i \) is the \( i \)-th component of \( X \in \mathbb{R}^3 \).

A class color point is between the class center \( c \) and \( 2.4 \sigma_i \) with a probability of 95% on each axis. Thus, the distance between each of the main dominant color points \( g_1 \) and \( g_2 \) and the class center \( c \) is selected to be half of this value, that is \( 1.2 \sigma_M \).

In the HSV space, the variances available are for the Hue, Saturation and Value components. Here the variance \( \sigma_M \) corresponding to the principal axis needs to be determined. Thus the class points are transformed into the space formed by the covariance matrix eigenvectors with center being the class center, as illustrated in Figure 16 for a 2-dimensional case.
The transformation from the HSV space centered at O into the eigenvectors space centered at $c$ is done by:

$$T \times p_{HSV} - c = p_{V_1V_2V_3}$$

(Eq III-9)

with $T = \text{inv} \left[ \begin{array}{ccc} V_{1H} & V_{2H} & V_{3H} \\ V_{1S} & V_{2S} & V_{3S} \\ V_{1V} & V_{2V} & V_{3V} \end{array} \right]$ and $c = \left[ \begin{array}{c} c_H \\ c_S \\ c_V \end{array} \right]$

Once the class points are transformed, the covariance matrix is calculated to obtain the variance $\sigma_M$:

$$\text{COVTRANS} = \begin{bmatrix} \sigma_{V_1}^2 & K_{V_1V_2} & K_{V_1V_3} \\ K_{V_1V_2} & \sigma_{V_2}^2 & K_{V_2V_3} \\ K_{V_1V_3} & K_{V_2V_3} & \sigma_{V_3}^2 \end{bmatrix}$$

(Eq III-10)
With $V_1$, $V_2$ and $V_3$, the sets containing the coordinates in the eigenvector coordinate system of all the class color points.

Finally, the centers $g_1$ and $g_2$ for the next segmentation are set on each side of the current class center $c$ along the eigenvector $V_M$ corresponding to the maximum eigenvalue, using the equations:

\[
\begin{bmatrix}
  g_{1H} \\
  g_{1S} \\
  g_{1V}
\end{bmatrix} = \begin{bmatrix}
  e_H \\
  e_S \\
  e_V
\end{bmatrix} + 1.2\sigma_M \times \begin{bmatrix}
  V_{MH} \\
  V_{MS} \\
  V_{MV}
\end{bmatrix} \quad \text{(Eq III-11)}
\]

and

\[
\begin{bmatrix}
  g_{1H} \\
  g_{1S} \\
  g_{1V}
\end{bmatrix} = \begin{bmatrix}
  e_H \\
  e_S \\
  e_V
\end{bmatrix} - 1.2\sigma_M \times \begin{bmatrix}
  V_{MH} \\
  V_{MS} \\
  V_{MV}
\end{bmatrix} \quad \text{(Eq III-12)}
\]

Figure 17 shows a 2-dimensional representation of the placement of the two dominant color points.
Once the two dominant points are calculated, the class is segmented and the compactness of each class is checked to decide about further segmentation.

At the end of the color segmentation, the image is segmented in $k$ classes, with $k$ ranging from 2 to 8. Each image color point belongs to a class and its color is changed to be the color of the class center reducing the original 16 million colors to a maximum of 8 colors.

When the image is color segmented, the next step is to study the image main object shape.

**III.2 Shape representation**

The shape feature describes the image main object. The images used for this work are assumed to have one large object with a dominant color. Thus, the main object is likely to be represented by one of the color classes that have been computed during the color segmentation.
The first step is to determine the best class that would represent the image main object and then extract information about the color and the shape of this object. Here, we use the object area and perimeter to represent its shape. Finding the main object as well as calculating the shape data is described in Figure 18.

The algorithms used by the color segmentation are performed on the image color points, that is the color coordinates H, S and V, whereas the algorithms used for the shape recognition use the spatial position of the image pixels. For example, the standard deviation is the spatial standard deviation, computed using the spatial position of the pixels.

III.2.1 Image main object

One of the color classes computed in the color segmentation is likely to represent the image main object. The main object is assumed to be large therefore the size of the corresponding color class must be large. To separate the background, which is usually a large class, the spatial standard deviation is calculated. The pixels that form an object are close to each other in the space whereas the background pixels are spread wider. Thus the spatial standard deviation of the class that represents the object must be small.

A weight $W_i$ is computed for each class $i$ to take in consideration these two factors: the class size and its compactness. The class, which has the largest weight, is the one corresponding to the main object.
Compute image size $A_S$ and maximum standard deviation $\sigma_X$. Then, work on each image class. $W_M \leftarrow 0$

Create a binary image representing the current class.

Calculate the class area $A_i$ and the class spatial standard deviation $\sigma_i$.

Compute the class weight to be:

$$W_i = \left(\frac{A_i}{A_S}\right) + \left(1 - \frac{\sigma_i}{\sigma_S}\right).$$

$W_i > W_M$?

Yes

The class representing the main object is this one. $W_M \leftarrow W_i$

No

Last class?

Yes

Perform a morphological closure on the class representing the main object.

Calculate the class perimeter.

Main image object information: area, perimeter and color.

Figure 18: Shape feature determining algorithm
Thus for each class, the following is calculated:

\[ W_i = \left( \frac{A_i}{A_S} \right) + \left( 1 - \frac{\sigma_i}{\sigma_S} \right), \]  
\[(\text{Eq III-13})\]

With \( A_i \): the class area = the number of pixels,

\( A_S \): the image size = height \times width = total number of pixels in the image,

\( \sigma_i \): the class spatial standard deviation.

\( \sigma_S \): the maximum image standard deviation.

The maximum image standard deviation is the standard deviation of a class composed by two points located at two opposite image corners. The image maximum standard deviation is:

\[ \sigma_S = \sqrt{\left( \frac{\text{Width}}{2} \right)^2 + \left( \frac{\text{Height}}{2} \right)^2}, \]  
\[(\text{Eq III-14})\]

With Width, the image width and Height, the image height.

Moreover:

- if \( \sigma_i \) is small: \( 1 - \frac{\sigma_i}{\sigma_S} \rightarrow 1, \)
- if \( \sigma_i \) is large: \( 1 - \frac{\sigma_i}{\sigma_S} \rightarrow 0, \)
Also:

\[
\text{if } A_i \text{ is large: } \left( \frac{A_i}{A_S} \right) \rightarrow 1, \\
\text{if } A_i \text{ is small: } \left( \frac{A_i}{A_S} \right) \rightarrow 0.
\]

Thus the class weight takes value between 0 and 2 and the class with the largest weight is the most suitable to represent the main object. Next an example of finding the image main object is presented. The image test is presented in Figure 19.

**Figure 19: Image test**

The result of the segmentation gives 7 classes:

Each class is represented as a binary image and shown in Table 3 together with the corresponding weight.
Table 3: Class selection to represent the image main object

Since, it has the largest weight, the 6th class is selected by the algorithm to represent the object. In the original image, this represents the flower, which is the image main object.
III.2.2 Main object shape representation

After identifying the class, which represents the image main object, the information about the object shape is extracted. The object area has already been calculated in the previous step. To be able to match more efficiently a user query, the object area is converted to the percentage of the object area to the total image size. The user input consists of drawing an object (see IV.2.2 Image search) which gives an indication of the object size ratio. Thus the significant area information is not the pixel count but the ratio.

The second data used to describe the object is its perimeter. To calculate the object perimeter a morphological closure operation is performed on the class. This smoothes object contours and corresponds better to the user input. Users usually don't draw detailed contours to form a query. The function used to perform the closure is the Matlab function “bwmorph.m”.

A closure is the sequence of two main morphological operations: dilation and erosion. The state of any given pixel in the output image is determined by applying a rule to the neighborhood of the corresponding pixel in the input image. The rule used defines the operation as a dilation or an erosion:

- For dilation, if any pixel in the input pixels’ neighborhood is 1, the output pixel is 1. Otherwise, the output pixel is 0.
- For erosion, if every pixel in the input pixel’s neighborhood is 1, the output pixel is 1. Otherwise, the output pixel is 0.

The neighborhood is represented by a structuring element, which is here selected as a 3 by 3 square.
The result of the closure operation on the class is: . The object contour is smoother and the object is almost completely filled. For this object, the perimeter is determined using the Matlab function “bwperim.m”. The function works as follows: a pixel is part of the perimeter if its value is 1 and there is at least one zero-valued pixel in its neighborhood. The algorithm uses a 4-neighborhood: a pixel is said to be in the 4-neighborhood of another one if they are connected along the horizontal or vertical direction.

The resulting image gives a suitable contour. Finally the number of pixels which form the perimeter is computed.

Thus, the object feature is composed of the ratio of its area to the image size, the perimeter and color. Now a suitable metric should be defined to calculate distance between user query and the object features. Selection of this metric is beyond scope of this work and it is suggested for further research.
CHAPTER IV: RESULTS

This section presents the experimental results of color reduction algorithm and main object representation. The algorithm was implemented using Matlab 5.3 and Mideva 4.5. The latter is compatible with the Matlab syntax and is compiled which accelerates the execution time. Once the image features are computed, they are tagged within the image. The graphical user interface for the image tagging and retrieval part has been developed under Microsoft Visual J++ 6.0. However the link has not been implemented between the Matlab functions and the user interface. Thus the graphical user interface is presented in this section to show what the user input would be like but no retrieval is performed.

IV.1 Results

IV.1.1 Color segmentation tests

Table 4 presents the results of the color reduction algorithm. The two first tests are performed on computer synthesized images. The first test is performed on a simple image composed of 4 colors easily distinguishable by the human eye and the color segmentation correctly identifies 4 classes. The second test is performed on a random color image. The image has been created using the random function for each image color point. On a random image, the algorithm should not be able to find compact color classes. The algorithm finds 8 colors since it is the maximum number of colors allowed, however the classes are not compact. The other tests are performed on specific images. Images were taken with an analog camera, scanned with a resolution of 100 pixels/inch, stored in
TIFF uncompressed file format and transformed to bitmap file format for processing.

Independent of the file format, the images are 24 bit true color images.

<table>
<thead>
<tr>
<th>Input Image</th>
<th>Color Reduced Image</th>
<th>Color Centers in RGB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 4 color image</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\begin{bmatrix} 255 &amp; 252 &amp; 2 &amp; 255 \ 0 &amp; 255 &amp; 252 &amp; 0 \ 252 &amp; 0 &amp; 255 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>2: Random image</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\begin{bmatrix} 209 &amp; 93 &amp; 202 &amp; 29 &amp; 222 &amp; 229 &amp; 93 &amp; 133 \ 254 &amp; 152 &amp; 76 &amp; 66 &amp; 175 &amp; 117 &amp; 111 &amp; 3 \ 234 &amp; 17 &amp; 164 &amp; 69 &amp; 162 &amp; 243 &amp; 205 &amp; 142 \end{bmatrix}$</td>
</tr>
<tr>
<td>3: Flower 1</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\begin{bmatrix} 193 &amp; 207 &amp; 174 &amp; 186 &amp; 241 &amp; 178 &amp; 105 \ 74 &amp; 234 &amp; 203 &amp; 198 &amp; 197 &amp; 44 &amp; 153 \ 85 &amp; 233 &amp; 167 &amp; 155 &amp; 129 &amp; 34 &amp; 118 \end{bmatrix}$</td>
</tr>
<tr>
<td>4: Flower 2</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\begin{bmatrix} 229 &amp; 52 &amp; 88 \ 230 &amp; 73 &amp; 119 \ 218 &amp; 64 &amp; 77 \end{bmatrix}$</td>
</tr>
<tr>
<td>5: Flower 3</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\begin{bmatrix} 189 &amp; 79 &amp; 188 &amp; 56 \ 199 &amp; 122 &amp; 172 &amp; 74 \ 180 &amp; 92 &amp; 85 &amp; 74 \end{bmatrix}$</td>
</tr>
<tr>
<td>6: Flower 4</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\begin{bmatrix} 199 &amp; 70 &amp; 111 &amp; 157 &amp; 178 &amp; 198 \ 202 &amp; 104 &amp; 144 &amp; 143 &amp; 182 &amp; 164 \ 231 &amp; 94 &amp; 101 &amp; 167 &amp; 122 &amp; 91 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

Table 4: Color reduction results
### IV.1.2 Tests of the shape recognition

<table>
<thead>
<tr>
<th></th>
<th>Input Image</th>
<th>Color Reduced Image</th>
<th>Classes</th>
<th>Object Class - Perimeter</th>
<th>Object Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="reducedimage1.png" alt="Reduced Image 1" /></td>
<td><img src="classes1.png" alt="Classes 1" /></td>
<td><img src="area31.4perimeter72.png" alt="Area 31.4 72" /></td>
<td><img src="color25522.png" alt="Color 255 2 2" /></td>
</tr>
<tr>
<td>2:</td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="reducedimage2.png" alt="Reduced Image 2" /></td>
<td><img src="classes2.png" alt="Classes 2" /></td>
<td><img src="area28.8perimeter101.png" alt="Area 28.8 101" /></td>
<td><img src="color1784434.png" alt="Color 178 44 34" /></td>
</tr>
<tr>
<td>3:</td>
<td><img src="image3.png" alt="Image 3" /></td>
<td><img src="reducedimage3.png" alt="Reduced Image 3" /></td>
<td><img src="classes3.png" alt="Classes 3" /></td>
<td><img src="area29.2perimeter178.png" alt="Area 29.2 178" /></td>
<td><img src="color229230218.png" alt="Color 229 230 218" /></td>
</tr>
<tr>
<td>4:</td>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="reducedimage4.png" alt="Reduced Image 4" /></td>
<td><img src="classes4.png" alt="Classes 4" /></td>
<td><img src="area65.9perimeter278.png" alt="Area 65.9 278" /></td>
<td><img src="color18817285.png" alt="Color 188 172 85" /></td>
</tr>
<tr>
<td>5:</td>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="reducedimage5.png" alt="Reduced Image 5" /></td>
<td><img src="classes5.png" alt="Classes 5" /></td>
<td><img src="area76.3perimeter232.png" alt="Area 76.3 232" /></td>
<td><img src="color7010494.png" alt="Color 70 104 94" /></td>
</tr>
</tbody>
</table>

**Table 5: Image main object representation results**
Tests performed on images 1 through 3 gives satisfactory results. A main object and its color are extracted and they truly correspond to the input image main object.

The image of the 4th test has the same Hues in the flower main object and the dominant part of the background. Since the Hue coordinate has a large weight in the color segmentation, the algorithm cannot distinguish the flower from the background. Thus an image must have a main object Hue distinguishable from the background in order to perform a successful segmentation.

The image of the 5th test has a small object with a distinct color. However the object is too small in comparison to the blue background.

In conclusion, the algorithm finds a main object and its color in an image under the following, neither obvious, conditions:

- The object is large enough.
- The object color is distinguishable from the background.

**IV.2 User interface**

The user interface is currently running on the Internet at http://lyon.ece.orst.edu. It consists of tagging images with their relevant content and retrieving images based on a color, shape or both.

**IV.2.1 Image tagging**

The Java applet developed under Microsoft Visual J++ 6.0 can currently only extract from an image its main colors. A link needs to be made between the image features recognition algorithms described in this work and the Java applet.
The data extracted are tagged to the image. Only JPEG compressed images can be tagged. The JPEG image header has a comment space available for external data. The applet reads the image header, finds the comment space and writes the data. Thus the data can be read along with the image when retrieving images and then compared to the user query. Many retrieval systems store the image features separately from the image in a feature database. The main advantage in tagging features within the images is that if the image is deleted or altered, no attempt will be made to retrieve it. A disadvantage of this approach is that the image header needs to be read over the network to compare the image features with the user query, and that is time consuming. However, the system does not read the whole image but only a small part containing the header, reducing the downloading time considerably.

The graphical user interface for the image tagging process is shown in Figure 20.
The following web page gives the user instructions to tag an image:

**Content-based image tagging**

If this is your first time tagging an image using our tag engine, go here.

To choose the image you want to tag, use the menu that appears on your screen:

1. Click on the button named "Choose a file to tag".
2. Choose the file on your local hard drive and click Ok.
3. Just wait the message "End of tagging". Note that it can take more than one minute to prepare the image depending on its size.

**WARNING** In the program, make sure that applets cannot access the operating on the file we found to be able to read and write all images on the local drive. 

You need to change the security parameters of the internet explorer 4.0 program:

1. Choose from the menu, "View => Internet options", and then the "Security" folder.
2. Choose "Custom" and then click on "Settings".
3. Look at the "Java permissions" and choose "Custom".
4. Then click on "Java custom settings".
5. Choose the folder "Edit permissions" and then Enable everything.

The following menu appears to let the user select the image he wants to tag:

**Figure 20: Graphical user interface for the image tagging**
IV.2.2 Image search

The user can perform a basic or a more advanced search.

IV.2.2. Basic search

The graphical user interface of a basic search is shown in Figure 21. The user can choose a color using the color wheel and draw an object shape dragging the mouse to draw straight lines. When pressing the Search button, the system reads these data and starts the retrieval process. The metrics to perform the shape retrieval process has not been implemented in this work. The user can press the Advanced Search button to go to a more advanced user interface.

IV.2.2. Advanced search

The graphical user interface for the advanced search is shown in Figure 22. The difference with the basic search is that the user can enter the color by HSB or RGB coordinates instead of choosing the color on the color wheel. The color is then displayed on the color wheel to allow the user to verify that he entered the correct coordinates. Since the programming language Microsoft Visual J++ has the HSB color space implemented, the HSV space is replaced by the HSB space. However they represent the same color information, the Brightness coordinate is used instead of the Value coordinate, the difference resides in the transformation calculations from the RGB color space.
Figure 21: Graphical user interface for the basic image retrieval
Figure 22: Graphical user interface for the advanced image retrieval
The graphical user interface is designed to allow the user to do three different requests:

- Choose a color, using the basic or advanced search, and request images that have this color as one of their main colors.

- Draw a shape and request to find images, which have a main object similar to the shape drawn.

- Choose a color, draw a shape and request to find images, which have a main object similar to the shape drawn and color close to the one chosen.
CHAPTER V: CONCLUSIONS

V.1 Conclusions

A fully automated method for content-based color image summarization is developed to extract color and shape content of an image without human input.

A color segmentation algorithm is used to segment image dominant colors. The input image coded in 16 million colors is color reduced to a maximum of 8 colors. The image is mapped into the HSV color space where pixels are analyzed for detecting color clusters. In the process of color classification, color clusters are extracted using an algorithm based on the k-mean clustering algorithm. A saturated distance is proposed to discriminate between two color points in the HSV color space.

To complete the feature set describing an image, the segmented image is studied. The shape of the image main object is extracted using the morphological operations. The whole process is automated and there is no need for human interaction while summarizing the image features. The experimental results using natural color images demonstrate feasibility of this method by segmenting the original images to summarize image colors and main object shape information.

Once the image color and shape features are extracted, they are tagged within the image. A graphical user interface is presented to allow the user to tag JPEG file format images and to retrieve images over the Internet. Requests are based on a sketch drawn by the user and colors chosen on a color wheel.
V.2 Future work

There are number of areas for continuing research on content-based color image summarization:

- Study of the clustering algorithm performance dependence on the selection of the color space and color distance measure.
- Shape features summarization as there are no general methods to extract a complex shape of an object.
- Use of additional visual features, such as texture to further enhance the content-based summarization.

Also, as a direct continuation of this project, the following can be studied:

- To make the retrieval process feasible, metrics need to be developed to measure the distance between image features. The color reduction algorithm uses a color distance threshold to define equivalent colors. This threshold can be used to calculate the distance between color features. Evaluating the distance between two object shapes has not been studied in this project and is an open area for future work.
- The algorithms code to extract image features need to be changed so they can be used within the Java applet for retrieval purpose. These algorithms use specific Mathworks Matlab functions, thus the code cannot be translated into pure C or Java code. Some tools like MathTools Matcom can integrate m and c files under Microsoft Visual C++ projects, such tools need to be evaluated for the compatibility with Java applets.
• Currently, only JPEG file format can be tagged, but other image file formats should be studied for the tag placement.
BIBLIOGRAPHY


APPENDICES
APPENDIX A: SATURATED DISTANCE

A function $d$ needs to fulfill the following requirements to be a distance function:

- If $a = b$, then $d(a, b) = 0$.
- If $a$ not equal to $b$ not equal to $c$, then:
  1. $d(a, b) > 0$.
  2. $d(a, b) = d(b, a)$.
  3. $d(a, b) + d(b, c) \geq d(a, c)$.

The saturated distance is represented in the following figure:

$d_{Sat}$: Saturated distance of one coordinate (H, S, or V) between two points.

$d$: H, S or V coordinate difference.

$d_{Max}$: Maximum Value.

$D$: Dead Point.

$Sat$: Saturation Point.

Saturated distance
For S and V components, $d$ represents the absolute value of the coordinate difference. The Hue component is an angle between 0 and 255, thus $d$ is the smallest positive angle value for H component. The three values of $d$, $d_{\text{SatH}}$, $d_{\text{SatS}}$, $d_{\text{SatV}}$ are added to compute the final distance between two color points:

$$d_{\text{Sat}} = d_{\text{SatH}} + d_{\text{SatS}} + d_{\text{SatV}}.$$ (Eq 1)

The value $d_{\text{Max}}$ is the maximum distance between the coordinate H, S or V of two color points. The saturated distance function has three zones:

1. The dead zone (from 0 to $D$) groups the close points by setting the distance between them to zero.
2. The linear zone (from $D$ to $Sat$) computes the distance using the following equation:

$$d_{\text{Sat}} = \frac{d_{\text{Max}}}{(S - D)} \times d - \frac{d_{\text{Max}} \times D}{(S - D)}.$$ (Eq 2)

3. The saturation zone saturates the distance by setting the distance equals to the limit value $d_{\text{Max}}$.

To demonstrate that this function does not satisfy the distance requirements, we just need to show with an example that one of the requirements is not satisfied.

Consider the following values of $d_{\text{Max}}$, $D$ and $S$ for each of the H, S and V components:
<table>
<thead>
<tr>
<th></th>
<th>H</th>
<th>S</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{\text{Max}}$</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$D$</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>$S$</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

**Example values**

We want to demonstrate that when $a$ not equal to $b$ not equal to $c$, the third requirement is not satisfied:

Take $a=(10, 100, 100)$, $b=(30, 120, 120)$, $c=(40, 130, 130)$.

We calculate $d_{\text{Sat}}(a, b)$, $d_{\text{Sat}}(b, c)$, $d_{\text{Sat}}(a, c)$:

\[
d_{\text{Sat}}(a, b) = d_{\text{Sat}}(a, b) + d_{\text{Sat}}(a, b) + d_{\text{Sat}}(a, b)
\]
\[
= 0 + 0 + 0
\]
\[
= 0
\]

\[
d_{\text{Sat}}(b, c) = d_{\text{Sat}}(b, c) + d_{\text{Sat}}(a, b) + d_{\text{Sat}}(b, c)
\]
\[
= 0 + 0 + 0
\]
\[
= 0
\]

\[
d_{\text{Sat}}(a, c) = d_{\text{Sat}}(a, c) + d_{\text{Sat}}(a, c) + d_{\text{Sat}}(a, c)
\]
\[
= 10 + 1 + 1
\]
\[
= 12
\]

Thus: $d_{\text{Sat}}(a, c) > d_{\text{Sat}}(a, b) + d_{\text{Sat}}(b, c)$.

The saturated distance does not satisfy the fourth requirement for a function to be a distance.
APPENDIX B: MULTIDIMENSIONAL GAUSSIAN LAW

In this appendix, the 3-dimensional Gaussian law is studied to determine for each dimension, the range from the mean where the probability that the sample falls into that range is 95%. The range depends on the variance.

If \( X \) is a scalar Gaussian random variable with mean \( \mu \) and variance \( \sigma^2 \), its probability distribution function (pdf) is:

\[
f_X(x) = \frac{1}{\sqrt{2\pi \sigma}} \exp \left( -\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2 \right) \tag{Eq 1}
\]

The probability that a sample of such a distribution will fall between \( a \) and \( b \) is:

\[
P(a \leq X \leq b) = \frac{1}{\sqrt{2\pi \sigma^2}} \int_a^b e^{-\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2} \, dx \tag{Eq 2}
\]

With \( \beta = (x - \mu) / \sigma \), \( d\beta = (1/\sigma) \, dx \), \( b' = (b - \mu) / \sigma \), \( a' = (a - \mu) / \sigma \), we obtain:

\[
P(a \leq X \leq b) = \frac{1}{\sqrt{2\pi}} \int_{a'}^{b'} e^{-\frac{1}{2} x^2} \, dx,
\]

\[
= \frac{1}{\sqrt{2\pi}} \int_{0}^{b'} e^{-\frac{1}{2} x^2} \, dx - \frac{1}{\sqrt{2\pi}} \int_{0}^{a'} e^{-\frac{1}{2} x^2} \, dx
\]

The function \( \text{erf}(x) \) is defined as:
\[
\text{erf}(x) = \frac{1}{\sqrt{2\pi}} \int_{0}^{x} e^{-\frac{t^2}{2}} \, dt
\]  
(Eq 3)

Hence:

\[P(a \leq X \leq b) = \text{erf}\left(\frac{b - \mu}{\sigma}\right) - \text{erf}\left(\frac{a - \mu}{\sigma}\right)\]  
(Eq 4)

With \(b=\mu+t\) and \(a=\mu-t\):

\[P(\mu - t \leq X \leq \mu + t) = \text{erf}\left(\frac{t}{\sigma}\right) - \text{erf}\left(\frac{-t}{\sigma}\right)\]

And \(\text{erf}(t/\sigma) = - \text{erf}(-t/\sigma)\), thus:

\[P(\mu - t \leq X \leq \mu + t) = 2 \times \text{erf}\left(\frac{t}{\sigma}\right)\]  
(Eq 5)

For \(t = 2\sigma\):

\[P(\mu - t \leq X \leq \mu + t) = 2 \times \text{erf}\left(\frac{2\sigma}{\sigma}\right) = 2 \times \text{erf}(2) = 2 \times 0.47724 = 0.95448\]

Hence over 95% of the samples will fall within \(2\sigma\) and the mean \(\mu\):

\[P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) = 0.95\]
1-dimensional Gaussian distribution

Now, consider the multidimensional Gaussian law. The pdf of the random vector \( X = (X_1, \ldots, X_n)^T \) with independent components \( X_i, i=1, \ldots, n \) is the product of the individual pdf's of \( X_1, \ldots, X_n \) that is:

\[
 f_X(x_1, \ldots, x_n) = \prod_{i=1}^{n} f_{X_i}(x_i) \\
= \frac{1}{(2\pi)^{n/2} \sigma_1 \ldots \sigma_n} \exp \left[ -\frac{1}{2} \sum_{i=1}^{n} \left( \frac{x_i - \mu_i}{\sigma_i} \right)^2 \right].
\] (Eq 6)

Where \( \mu_i, \sigma_i^2 \) are the mean and variance, respectively, of \( X_i, i=1, \ldots, n \).

The objective of this study is to determine an approximated range, thus the independent variables case is studied.

From Equation 5, we can derive the 3-dimensional case for independent variables:
\[ P(\mu - t \leq X \leq \mu + t) = 2 \times \text{erf} \left( \frac{t_1}{\sigma_1} \right) \times 2 \times \text{erf} \left( \frac{t_2}{\sigma_2} \right) \times 2 \times \text{erf} \left( \frac{t_3}{\sigma_3} \right) \]  

(Eq 7)

With \( \mu = (\mu_1, \mu_2, \mu_3), t = (t_1, t_2, t_3) \) and \( \sigma = (\sigma_1, \sigma_2, \sigma_3) \).

Choosing \( t_1 = \alpha \sigma_1, t_2 = \alpha \sigma_2 \) and \( t_3 = \alpha \sigma_3 \) gives:

\[ P(\mu - t \leq X \leq \mu + t) = 8 \times \text{erf} (\alpha)^3 \]  

(Eq 8)

We determine \( \alpha \) such that the probability equals to 0.95:

\[ P(\mu - t \leq X \leq \mu + t) = 8 \times \text{erf} (\alpha)^3 \]

\[ \text{erf}(\alpha) = \left( \frac{0.95}{8} \right)^{\frac{1}{3}} \Leftrightarrow \alpha = 2.4 \]

Hence 95\% of the samples will fall within \( 2.4 \sigma_i \) and the mean \( \mu_i \) on each dimension.