WEB-BASED IMAGE SEARCH REFINEMENT VIA FEATURE EXTRACTION AND RANKING

by

Chrystalla Tryfonos

A Thesis Submitted to the Faculty of
the Computer Science Department
in Partial Fulfillment of the Requirements for the
Degree of Master of Science

California State University Channel Islands
Camarillo, California
November 2010

APPROVED FOR THE COMPUTER SCIENCE PROGRAM

_____

Advisor: Dr. William Wolfe Date

_____

Dr. Andrzej Bieszczad Date

_____

Dr. Peter Smith Date

APPROVED FOR THE UNIVERSITY

_____

Dr. Gary A. Berg Date

WEB-BASED IMAGE SEARCH REFINEMENT VIA FEATURE EXTRACTION AND RANKING

by
Chrystalla Tryfonos

ABSTRACT

This thesis is concerned with the development of the Refined Image Search by Example (RISE) system, which carries out web-based image searches and attempts to refine the results by utilizing an image analysis and ranking algorithm. Initially, background research is carried out regarding existing frameworks and systems for web-based image retrieval. RISE is based on the assumption that the combination of a textual and an image query can yield more meaningful search results. Specifically, the image results of a text-based search are retrieved from the web using the Google Image Search JavaScript API. Then, a combination of texture, color, and brightness low-level features is extracted from the resulting images as well as from the query image using C#. These features were selected after investigating a larger set of possible features and feature combinations using Matlab. Lastly, a similarity measure is utilized in order to determine the ranking of each resulting image with respect to the query image. Thus, the RISE image search results are presented in a refined and more relevant order using the user interface that was developed in ASP.NET and C#.

Acknowledgements

I would like to thank my thesis advisor, Dr. William Wolfe, for his valuable guidance and support throughout the research and development process of this project. I would also like to thank my fiancé for his encouragement and emotional support during the past few months.

TABLE OF CONTENTS

List of figures

CHAPTER 1

INTRODUCTION

1.1 Introduction to Web-Based Image Retrieval (WBIR)

The continuous growth of the World Wide Web (WWW) as well as the advances in digital imaging and related technologies, have led to an extremely voluminous distribution of images over the Internet. It is therefore very crucial that there exist efficient and reliable systems for handling web-based image search and retrieval tasks. The development of such systems has been a topic of ongoing research during the last few decades, a fact that shows how current and challenging the problem of retrieving images from the web continues to be.

Indeed, there is a large amount of complexity linked to the development of Web-Based Image Retrieval (WBIR) systems. This complexity is basically associated with the free and unattended use of images in web pages, as well as the lack of a standard that describes the relationships between the text and embedded images in the same web pages. In addition, web images tend to have a broad range of meanings, since they are created by different people and for different purposes, and their qualities vary significantly. As a result, the usage of a traditional database system and the sole use of a visual model for the retrieval process, become impossible.

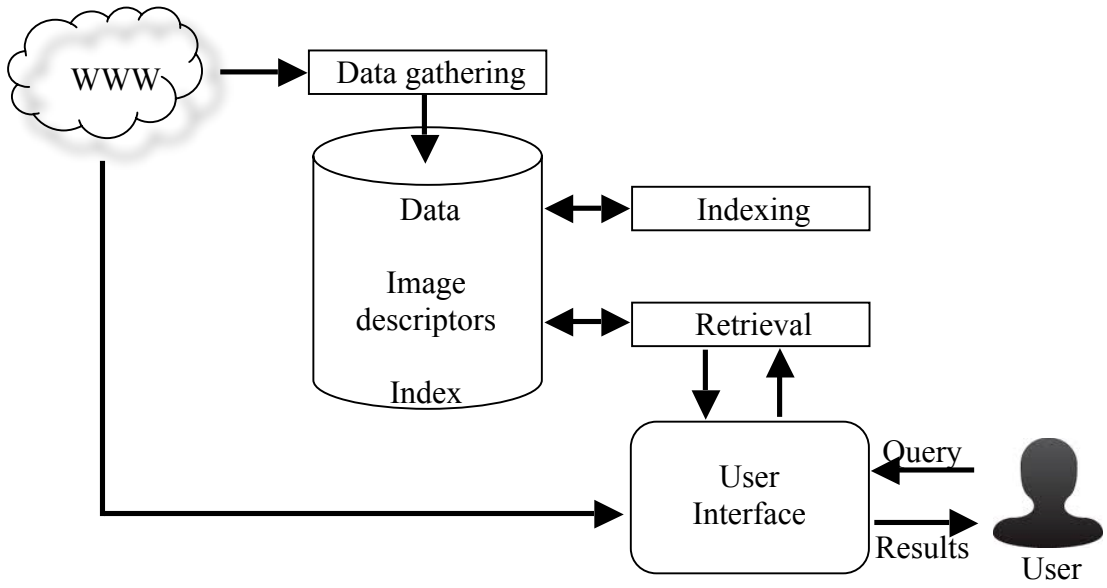

Fig. 1: General structure of a WBIR system.

A number of issues need to be addressed when developing WBIR systems are discussed in [1]. The most crucial of these issues are: (1) finding a means to gather the image data, (2) deciding what image descriptors to use for indexing and retrieval, (3) selecting the similarity and/or matching measures that will be used for retrieval, and (4) deciding how the user will query the system. Figure 1 demonstrates the general structure of a WBIR system, based on the aforementioned issues.

1.2 Frameworks for WBIR

In general, there are two frameworks for WBIR: 'Text-Based Image Retrieval' (TBIR) and 'Content-Based Image Retrieval' (CBIR) [2, 3, and 4]. TBIR, which is also known as 'Semantic-Based Image Retrieval' (SBIR), is the traditional approach of searching images and other types of web content. Web search engines that fall into this category usually collect images that are embedded in different sites on the Internet and index them by utilizing the text information that is shown in their corresponding web pages. For example, such information can be the file names

of the images, surrounding text, image captions, the page title, etc [8, 9]. Provided that a user enters a textual query as input to an image search engine, TBIR algorithms compare this query to the textual information of the indexed images, and results are returned based on the matching rankings that are produced.

Unfortunately, based solely on textual meta-data in order to acquire image results can be a very problematic task. First of all, the textual information that is extracted from a containing web page can be irrelevant to the semantics of an image. Also, lots of 'noise' images exist on the web, such as buttons, banners, or logos, which can render the results useless if their meta-data is included as input to the search algorithm. Furthermore, the content of the text that is meant to accompany an image on the web is highly subjective. That is, different people may write different things to describe the content of an image. This can cause a great level of imprecision between the expected and the actual results of an image search that is based on text information. In addition, one may need to retrieve images having the same look (e.g. texture or color distribution). In this case, CBIR techniques are more suitable for carrying out image searches since text rarely describes the image's features in addition to its content.

CBIR emerged as an alternative to TBIR, in order to alleviate problems that arise in the latter. Instead of utilizing textual information to retrieve images, CBIR systems automatically index images by utilizing their low-level visual features, such as color, texture, shape, and spatial layout [5]. Typically, an example image is provided as input to a CBIR system, something that is also known as 'Query-by-Visual-Example' (QBVE). Other means of QBVE, which are nonetheless sparsely used, include: submitting a sketch, clicking on a texture palette, or selecting a shape of a specific interest. After the query image is provided, its attributes are extracted. The same feature extraction algorithm that is applied on the input image is also applied on the

database images. Then, appropriate measures are used to calculate the similarities between the query image and the database images. More information regarding features and similarity measures that can be used for CBIR will be later investigated in Chapter 2.

Though many sophisticated CBIR algorithms have been developed over the past few years, the users' expectations for attaining image search results that are precise and tailored to their needs, have not yet been completely fulfilled. This is mainly caused by the so-called 'semantic gap' (SG), that is, the lack of coincidence between the information that one can extract from the low-level visual data and the high-level concepts that humans use (keywords, text descriptors), for a given situation [6]. An overview of techniques used to eliminate the SG can be found in [7]. Another reason why a web-based CBIR system can fail is due to the fact that images with similar low level features may have different contents. As a result, numerous disconnected and disparate images may show up in the results, causing frustration to the users.

From what was discussed above, it seems that by using TBIR or CBIR as standalone technologies in a WBIR system does not always yield the anticipated results. What appears more promising, however, is to incorporate both technologies in hybrid systems and attempt to employ techniques that will reduce the SG as much as possible [10]. For example, if a user took a picture near a stream in a park and needs additional pictures of that specific place (e.g. to create a panorama), the park name can be the input to TBIR and the image to CBIR. This thesis introduces a method that combines TBIR and CBIR in order to retrieve images from the WWW. Next, an overview of existing WBIR systems is presented along with the proposed system.

1.3 Overview of existing systems and proposed method

An important number of WBIR systems have been developed since the early '90s. These systems utilize either TBIR or CBIR techniques, or make a combination of the two (hybrid

systems). Below, we take a look at a few of these systems, beginning with earlier ones.

Among the first prototypes that were proposed is WebSeer [10], which exploits image content and associated text to index images. Users have to provide a textual query, coupled with some attributes of the desired image, such as its file size, dimensions, type (photograph or graphic), and prevailing colors. If the search is concentrated on people, the system gives users the opportunity to additionally specify the number of faces and the size of portraits. WebSeer possesses a crawler that traverses the Web, downloading a small number of both HTML and images. The retrieval process entails gathering of every image whose associated text contains any word of the given textual query, sorting of the results based on the weighted sum of these words, and refinement on the basis of image attributes (size, type, dimensions, etc.). After each image is indexed by the weighted keywords, it is processed by a series of tests (most common color test, farthest neighbor test, etc), which aim to classify the image as photograph or drawing, and to detect people's faces.

Early efforts on WBIR also include the DrawSearch [11] and ImageRover [12] systems. DrawSearch is a CBIR system that allows two types of querying: querying by sketch using color and shape distribution, as well as querying by texture content. Shape characteristics extraction is done by the use of a Fourier descriptor approach and texture-based segmentation and texture extraction are achieved by the use of Gaussian Markovian Random Fields (GMRF). The users can refine the results since the system incorporates relevance feedback that uses query-point movement [13].

ImageRover combines textual and visual information. Searches in this system are carried out on a database of 100,000 images collected from the Web. Users begin their search by providing a textual query. After the first results are returned, users can refine their search by providing a new

textual query, by selecting one or more of the returned images to guide a new search based on content, or they can use both. Image content in ImageRover is represented by color histograms, orientation histograms, and texture direction, whereas the HTML content containing the image is represented by a Latent Semantic Indexing (LSI) vector. In addition, Principal Component Analysis (PCA) is utilized to reduce feature dimension and *k-d* trees are used for indexing. A newer system that also combines textual and visual information is AtlasWISE [14]. This system gathers its data by browsing popular index pages, such as Google and Yahoo!, and downloads both text and images. The downloaded images and data are preserved until they are processed (only thumbnails and links are kept), and then they are deleted in order to save storage space. The visual features that are utilized include edge-orientation histograms and color histograms. The textual features, on the other hand, make use of keywords that are relevant to the images. These keywords are estimated from the images' tags and captions, page titles, and the surrounding text. The visual retrieval process is based on a relevance feedback algorithm that allows users to formulate their queries by combining positive and negative image examples. This way, the undesired images or features are eliminated.

Another system that exploits the capabilities of commercial search engines and utilizes both textual and visual features is ReSPEC (Re-Ranking Sets of Pictures by Exploiting Consistency) [15]. After a user enters a textual query, the Yahoo! search engine is used to download the first 500 image results. Then, a graph-based algorithm is used to segment the downloaded images into blobs and to build HSV color histograms for each blob. Subsequently, the blobs are clustered based on a gradient ascent method for finding local density maxima and the cluster that corresponds to the largest number of parent images is found. Finally, the mean of the selected cluster is calculated in the feature space and all Yahoo! results are re-ranked based on the

distance of each blob in every image to the calculated mean. By doing so, the system infers the representation of the object of interest (i.e. image) from text-based to content based, and improves the results of the search.

Rather than deducing the image search from text-based to content based, creators of Olive [16] claim that image results can be improved by introducing additional semantic layers during the retrieval process. The system primarily reformulates the textual query provided by the user. It does so by employing WordNet [17] knowledge base about the given concept. Then, it utilizes Google Images as an external module to collect images. At the same time, a list of categories that are close to the query is generated using the knowledge base. Once these two processes are finished, an answers page is generated. For each image on the answers page, it is possible to search for visually related images using the PIRIA visual search engine [18]. Experimenting with Olive showed that utilization of semantic structures in existing image indexes, like Google, can yield more meaningful results.

While most of the above systems require users to provide textual queries as inputs, none of them allows users to upload their preferred images and use those instead as the base of their search. Even the most popular search engines nowadays (Google, Yahoo!, Bing) are still text-based. Since these engines do not analyze the pixel content of images, they cannot be used to search for image collections that are not annotated. What would change this situation, however, is the development of automatic annotation and tagging techniques for the images, accompanied by a large number of concepts. Unfortunately, this is not a trivial task because creation of competent systems is a tremendously challenging problem.

A system that allows image queries and annotation in real time is ALIPR (Automatic Linguistic Indexing of Pictures – Real Time) [19]. After a user uploads a picture to the system

(or specifies an image URL), a list of automatically derived tags is suggested. Users can make their tag selections from this list and can additionally provide other tags, which they think are relevant to their search. When the image results are presented, users can also choose an image to carry out a related image search, find similar visual images, or just provide a textual query to search images based on tags or titles of the images. In order to achieve the real time computerized suggestions of the image tags, ALIPR combines feature extraction and statistical modeling. Specifically, a fast image segmentation method is used, which is based on wavelets and *k*-means clustering. Additionally, the system incorporates a generalized mixture of model techniques to handle non-vector data, which is developed by using the concept of Hypothetical Local Mapping (HLM).

Another system that allows querying by image examples is TinEye [20]. TinEye is a reverse image search engine for the web that has been given great attention recently. By submitting an image or its URL, a user can discover where this image came from, how it is being used, if modified versions of the image exist, or to find higher resolution versions. Rather than using keywords, watermarks, or inbound links (as Google does) to locate images, TinEye traces images by matching digital fingerprints of the image's pixel array. As a result, it can find exact same images that have been renamed, cropped, or even screen grabbed. TinEye has a tremendous potential to grow[1] and is estimated that it has already indexed more that 1.6 billion images.

There are also a few other systems available on the web that allow direct image use as the query of the search, including Gapoza [21] and Attraseek [22], but less effective than the aforementioned ones. Gapoza is a similar image search engine which allows its users to upload their own image, to provide the URL of an image, to submit a drawing, or to provide a textual query. Similarity is calculated based on color and image features. However, results do not always

[1] http://www.tineye.com/press/

include the object of search and hence there is a need of algorithmic refinement. Finally, Attraseek is a private image search engine and there is only a demo available on the web, from which users can search in an index of a few millions of images. There is not much information available regarding the algorithm used.

The proposed WBIR system combines TBIR and CBIR. Specifically, image search results from TBIR are collected and an input/query image is used to refine these results. This is done by extracting visual features from both the query image and the images that were retrieved by the TBIR. The goal is to utilize low-level features that can be extracted in relatively non-convoluted mathematical operations in order to have a fast WBIR process. The extracted features are primarily color-based and texture-based. Then a process is utilized to produce similarity scores between the query image and the resulting images (that were returned by TBIR). Then, these images are ranked accordingly so that the search results, i.e. the images, are presented to the user in a more relevant order.

1.4 Remaining chapters

The rest of this thesis is organized as follows. Chapter 2 investigates a number of low-level features that can be used for image retrieval and presents similarity measures that can be utilized to compare the query image to the images returned by the WBIR system. Chapter 3 presents RISE, a system that carries out web-based image searches by processing the results acquired by textual queries with the goal of refining the order in which they were initially presented by comparing them against a reference/query image. Chapter 4 illustrates and analyzes example search results of RISE. Finally, Chapter 5 draws conclusions and discusses for future work and possible improvements for RISE.

1.5 Key Terms

- CBIR – Content-Based Image Retrieval
- QBVE – Query-by-Visual-Example
- RISE – Refined Image Search by Example
- SBIR – Semantic-Based Image Retrieval
- SG – Semantic Gap
- TBIR – Text-Based Image Retrieval
- WBIR – Web-Based Image Retrieval
- WWW – World Wide Web

CHAPTER 2

FEATURE EXTRACTION AND SIMILARITY MEASURES

2.1 Feature Extraction

Systems that utilize CBIR technologies (either exclusively or as a part of hybrid applications), depend on the visual properties of images to derive their results. Low-level features, such as color, shape, and texture, are commonly used to capture the visual attributes of both the querying and the candidate images for retrieval. Extraction of features can be done globally (i.e. for the entire image) or locally (i.e. for a specific region of the image). In the case of local feature extraction, segmentation of the images is a common pre-processing step.

2.1.1 Color

Color is one of the most extensively used feature descriptors in CBIR systems. The differences between colors are sought to be measured in ways that are easy to implement. Colors are defined using specific models, like the RGB, HSV, CMY, and HSL color spaces, which are suitable for different applications [22]. Frequently used color features or descriptors are the following: color histogram [23], color moments [24], and color coherence vectors [25]. While the aforementioned features are among the earliest used in CBIR, more recent approaches focus more on the summarization of colors in an image, that is, the creation of signatures out of colors [27, 28]. Other color features used, that were also included in the MPEG-7 standard are: dominant color and scalable color [26].

2.1.1.1 Color histogram

A color histogram denotes the joint probabilities of the intensities of the color channels in a specific color space [23]. Specifically, for the RGB space, the color histogram is defined as

$$h_{R,G,B} = N \cdot \Pr ob\{R = r, G = g, B = b\}, \tag{2.1}$$

where N is the number of pixels in the image and R, G, and B the three color channels.

Computation of the color histogram is achieved by discretizing the colors within the image and by counting the number of pixels for each color. Since the number of pixels is finite, it is sometimes useful to convert the three channel histogram into a single variable histogram. Given a specific transform m, let's say $m = r + N_r g + N_g N_b b$, where N_r, N_g, and N_b are the number of bins for the red, green, and blue colors respectively, the single variable histogram becomes

$$h_m = N \cdot \Pr ob\{M = m\} \tag{2.2}$$

Color histograms are generally efficient and show no sensitivity to small changes in camera viewpoint. However, they do not provide spatial information of the image, which means that two images that are significantly different may have similar histograms. Additionally, color histograms are sensitive to compression and changes in overall image brightness.

2.1.1.2 Color moments

Color moments are measures used to describe the probability distribution of colors in an image and must be calculated for every channel in the color space used. The authors of [24] make use of three central moments for color distribution: Mean (E), Standard Deviation (σ), and Skewness (s). Assuming that p_{ij} is the value of the i^{th} color channel at the j^{th} image pixel and N the number of pixels in an image, the aforementioned central moments are calculated by the following formulas:

$$E_i = \frac{1}{N}\sum_{j=1}^{N} p_{ij}\ , \tag{2.3}$$

$$\sigma_i = \sqrt{\left(\frac{1}{N}\sum_{j=1}^{N}\left(p_{ij} - E_i\right)^2\right)}, \tag{2.4}$$

$$s_i = \sqrt[3]{\left(\frac{1}{N}\sum_{j=1}^{N}\left(p_{ij} - E_i\right)^3\right)}, \tag{2.5}$$

where,

E_i : the average color value in the image

σ_i : the square root of the variance of the distribution

s_i : a measure of the degree of asymmetry in the distribution

A function of the similarity between two image distributions can be defined as the sum of the weighted differences between the moments of the two distributions:

$$d_{mom}(H,I) = \sum_{i=1}^{r} w_{i1}\left|E_i^1 - E_i^2\right| + w_{i2}\left|\sigma_i^1 - \sigma_i^2\right| + w_{i3}\left|s_i^1 - s_i^2\right|, \tag{2.6}$$

where,

(H, I) : the two image distributions being compared

i : the current channel index (e.g. 1=R, 2=G, 3=B)

r : number of channels (e.g. 3)

E_i^1, E_i^2 : the first moments (means) of the two image distributions

σ_i^1, σ_i^2 : the second moments of the two image distributions

s_i^1, s_i^2 : the third moments of the two color distributions

w_i : the user-specified weights for each moment

2.1.1.3 Color coherence vectors

A way of incorporating spatial information into the color histogram is by using the Color Coherence Vectors (CCV) [25]. Every bin of a histogram is divided in two different types: coherent, if it belongs to a large uniformly-colored region, or incoherent, if it does not.

Let a_i be the number of coherent pixels of the i^{th} color bin in an image and b_i the number of incoherent pixels. Assuming that the color histogram of an image is represented by the vector $\langle a_1 + b_1, a_2 + b_2, ..., a_N + b_N \rangle$, then the CCV of the image is formed as:

$$CCV = \langle (a_1, b_1), (a_2, b_2), ..., (a_N, b_N) \rangle. \tag{2.7}$$

By adding spatial information, it was shown that CCV provides better retrieval results than the color histogram, especially for those images which have either mostly uniform color or mostly texture regions [25].

2.1.1.4 MPEG-7 color descriptors

2.1.1.4.1 Dominant Color Descriptor (DCD)

The DCD gives a compact description of the most dominant colors in an image. The dominant colors are calculated for each image rather than being fixed in the space defined by the histogram bins. The DCD layout is

$$DCD = \{(c_i, p_i)\},\ i=1, 2, \ldots, N, \tag{2.8}$$

where,

N : number of dominant colors

c_i : a 3-D color vector (e.g. RGB space)

p_i : percentage of pixels in the image corresponding to color i

The maximum value for N is 8. In order to extract the dominant colors, the colors of the

image are clustered, usually in a perceptually uniform color space such as the LUV space. The dissimilarity, D, between two DCDs

$$DCD1 = \{(c_{1i}, p_{1i})\},\ i=1, 2, \ldots, N_1,$$

$$DCD2 = \{(c_{2i}, p_{2i})\},\ i=1, 2, \ldots, N_2,$$

is

$$D(DCD_1, DCD_2) = \sum_{i=1}^{N_1} p_{1i}^2 + \sum_{j=1}^{N_2} p_{2i}^2 - \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} 2a_{1i,2j} p_{1i} p_{2j}\ , \tag{2.9}$$

where $a_{k,i}$ similarity coefficient between two colors c_k and c_i

$$a_{k,i} = \begin{cases} 1\text{-}d_{k,l}/d_{max} & d_{k,l} \le T_d \\ 0 & \end{cases}$$

with $d_{k.l} = \|c_k - c_l\|$ the Euclidean distance between the colors c_k and c_l. T_d is the maximum distance for two colors still considered to be similar and $d_{\max} = aT_d$, where a is a parameter.

2.1.1.4.2 Scalable Color Descriptor (SCD)

The SCD is defined in the HSV (Hue-Saturation-Value) color space with fixed color space quantization, and uses a Haar transform encoding [29]. The basic unit of the Haar transform consists of a sum and a difference of two adjacent histogram bins: primitive low-pass and high-pass filters. This unit is applied across the 256-bin color histogram of the image. Optional repetition results in lower resolution descriptors of 128, 64, 32 and 16 bits, and hence the descriptor is called scalable.

2.1.2 Texture

Texture provides surface characteristics for the analysis of many types of images including natural scenes, remotely sensed data, and biomedical modalities. Unlike color, texture occurs over a region rather that a point or pixel and it can be measured in terms of smoothness, coarseness, and regularity [30]. Over the past few decades texture has been broadly studied as a means of CBIR and several techniques have been developed, in both the spatial and frequency domain. Early approaches include gray-level co-occurrence matrices [31], Tamura features [32], multi-orientation filter banks [33], and Wold features [39]. In more recent approaches, different transforms were used to extract texture statistics, like the Discrete Cosine Transform (DCT) [34], wavelet transform [35, 36, 38], Gabor transform [37], and Fourier transform [40]. Texture features were also included in the MPEG-7 standard [26].

Extraction of texture features can sometimes be a very time consuming task, especially if a system entails image segmentation. This is a very serious thing to consider when it comes to image retrieval from the web, since large numbers of images are usually collected and processed. Processing and retrieval times of these images are crucial, so we aim for our WBIR system to extract texture features globally in order to speed up the retrieval process.

From the aforementioned texture features, Gabor filter features and MPEG-7 texture descriptors are simple to implement and fast when applied globally on images. Their mathematical descriptions follow.

2.1.2.1 Gabor filter features [41]

Gabor filter features are widely used for texture extraction in CBIR systems. They work optimally in minimizing the joint uncertainty in space and frequency, and are often used as orientation and scale tunable edge and line detectors. A two-dimensional Gabor function $g(x, y)$

is defined as

$$g(x,y)=\frac{1}{2\pi\sigma_x\sigma_y}\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x{}^2}+\frac{y^2}{\sigma_y{}^2}\right)+2\pi j\,Wx\right], \tag{2.10}$$

where, σ_x and σ_y are the standard deviations of the Gaussian envelopes along the x and y direction.

Then, a set of Gabor filters can be obtained by appropriate dilations and rotations of $g(x,y)$:

$$\begin{aligned} g_{mn}(x,y) &= a^{-m}g(x',y'), \\ x' &= a^{-m}(x\cos\theta + y\sin\theta), \\ y' &= a^{-m}(-x\sin\theta + y\cos\theta), \end{aligned} \tag{2.11}$$

where a>1, $\theta = n\pi/\mathrm{K}$, $n = 0,1,...,K-1$, and $m = 0,1,...,S-1$. K and S are the number of orientations and scales. The scale factor a^{-m} is to ensure that energy is independent of m.

Given an image $I(x,y)$ its Gabor transform is defined as

$$W_{mn}(x,y)=\int I(x,y)g^{*}_{mn}(x-x_1,y-y_1)dx_1dy_1, \tag{2.12}$$

where * indicates the complex conjugate. Then the mean μ_{mn} and the standard deviation σ_{mn} of the magnitude of $W_{mn}(x,y)$. Hence, $f=\lfloor\mu_{00},\sigma_{oo},...,\mu_{mn,}\sigma_{mn},\mu_{S-1K-1},\sigma_{S-1K-1}\rfloor$, can be used to represent the texture feature of a homogeneous texture region.

2.1.2.2 MPEG-7 texture descriptors

The MPEG-7 standard [26] included two texture descriptors: the Homogeneous Texture Descriptor (HTD) and the Edge Histogram Descriptor (EHD).

2.1.2.2.1 Homogeneous Texture Descriptor (HTD)

The HDT is a vector that consists of the outputs of a Gabor filter bank. The 2-D frequency

plane is partitioned into 30 channels which are modeled by Gabor filters with different scales and orientations. The HTD layout is:

$$HTD = \left[f_{DC}, f_{SD}, e_1, e_2, ..., e_{30}, d_1, d_2, ..., d_{30} \right], \tag{2.13}$$

where,

f_{DC} : the mean of the whole image

f_{SD} : the standard deviation of the whole image

e_i : the nonlinearly scaled and quantized mean energy of the i^{th} channel

d_i : the energy deviation of the i^{th} channel

As it can be seen from (2,13), the HTD is a 62-dimensional feature vector. For image retrieval, only the distance between two feature vectors, $dis\tan ce\left(HTD_{QUERY}, HTD_{DATABASE}\right)$, needs to be calculated.

2.1.2.2.2 Edge Histogram Descriptor (EHD)

The EHD contains information about the spatial distribution of edges in an image. Each image is divided in 4x4 sub-images and the local-edge histogram for each of these sub-images is stored. The edge histogram edges are categorized in five types: vertical, horizontal 45 degrees diagonal, 135 degrees diagonal, and non-directional. The histogram for each sub-image is obtained by subdividing it into smaller image blocks within which edge detectors are applied. The vector layout of EHD is

$$EHD = \left[h_{0,0}^{90}, h_{0,0}^{0}, h_{0,0}^{45}, h_{0,0}^{135}, h_{0,0}^{nondir}, ..., h_{3,3}^{90}, h_{3,3}^{0}, h_{3,3}^{45}, h_{3,3}^{135}, h_{3,3}^{nondir} \right], \tag{2.14}$$

where $h_{i,j}^{a}$ is the histogram count for pixel (i, j) and direction-bin *a*.

According to [26], the image retrieval performance can be significantly improved if the EHD is combined with other descriptors such as the color histogram descriptor. Also, the EHD and

HTD descriptors complement each other: the EHD performs best on large non-homogeneous regions, while the HTD operates on homogeneous texture regions.

2.1.3 Shape

Shape is a key attribute of segmented image regions and it is not used as widely as color and shape features. Shape representation techniques that have been proposed over the past few years can be broadly divided into contour-based and region-based methods [42]. Contour-based shape descriptors include Fourier descriptors [43], curvature scale space [44], and shape signatures [45]. Region-based shape descriptors include moment descriptors [46] and grid descriptors [47]. Region-based shape approaches seem to be superior to contour-based approaches since they can capture interior information in a shape, rather than exploiting information at boundary points, as contour-based descriptors do. Also, they can be employed to describe disjoint shape and robust shape distortions [42].

Segmentation and extraction of shape features can be a very time consuming and complex process, which can become a drawback when dealing with web-based image retrieval systems. For this reason, we are not going to use any shape features in our system, so we do not further explore mathematical descriptions of such features.

2.2 Similarity measures

After feature extraction is completed, appropriate metrics must be used in order to calculate the visual similarities between a query image and the images in the CBIR system's database. The result of the retrieval is not a single image but a list of images ranked by their similarities with the query image. A lot of similarity measures/distances have been used in image retrieval. Below

we present a few of the most popular ones.

2.2.1 Euclidean distance

Also known as L^2 norm, the Euclidean distance sums up the squared difference between pixels. Between two image vectors **x** and **y**, of length *N*, the Euclidean distance can be expressed as

$$L_2(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N} (\mathbf{x}_i - \mathbf{y}_i)^2 . \tag{2.15}$$

Examples of systems that utilize the Euclidean distance include [48] and [49].

2.2.2 Mahalanobis distance

The Mahalanobis distance calculates the negative sum of the products of the pixels and λ_i, the eigenvalue of a specific dimension [50, 51]. The Mahalanobis distance between the image vectors **x** and **y**, of length N, is

$$Mah(\mathbf{x},\mathbf{y}) = -\sum_{i=1}^{N} \mathbf{x}_i \mathbf{y}_i \frac{1}{\sqrt{\lambda_i}} . \tag{2.16}$$

2.2.3 Kullback-Leibler (K-L) divergence

The K-L divergence is a histogram based metric [35]. Having two image histograms H and H', the K-L divergence is defined by

$$d_{KL}(H,H') = \sum_{m=1}^{M} H_m \log \frac{H_m}{H'_m}, \tag{2.17}$$

where M is the number of bins in the histogram. Basically, the K-L divergence measures how inefficient it would be to code one histogram using the other. A problem with this metric is that it is neither symmetric nor numerically stable.

CHAPTER 3

RISE – REFINED IMAGE SEARCH BY EXAMPLE

3.1. General structure of RISE

The technical work done for this thesis concentrated on the construction of RISE, a system that carries out web-based image searches provided a user defined keyword and an image, and which returns refined image results by comparing the original search results with the given image. Figure 2 illustrates the general structure of RISE.

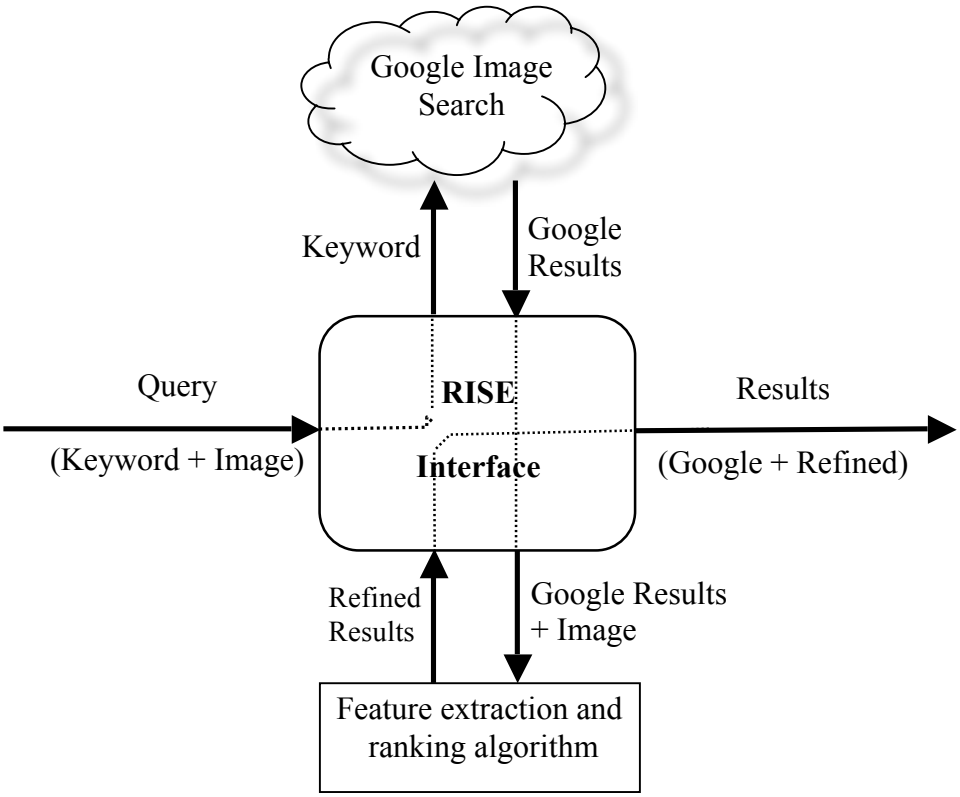

Fig. 2: General structure of RISE.

As it can be seen from Figure 2, RISE basically takes six steps to complete its image search and refinement tasks. Below is a brief description of each of these steps.

- Step 1: A user submits his search query to the system. The query is dual, since it consists of both a keyword and an image.
- Step 2: RISE keeps the query image and utilizes the keyword as a further input to Google Image Search. The functionality of attaining the image data from Google was achieved by exploiting the development options that are available in the respective Google Image Search JavaScript API. Further details for this API can be found in [52]. Since Google poses restrictions on the number of the results that can be returned by utilizing the aforementioned API, the maximum number of images that RISE can get is 64.
- Step 3: RISE receives the URLs of the resulting images and presents the images in the order they were sent from Google.
- Step 4: The image URLs, along with the query image, are subsequently fed to the feature extraction and ranking algorithm of the system. This algorithm extracts low level features from both the resulting images and the given image, and utilizes similarity measures in order to calculate rankings that show how close is the content of the query image to the content of each of the resulting images. Section 3.3 contains a detailed description of the algorithm used.
- Step 5: After the similarity rankings are calculated, the algorithm re-orders the URLs of the image results and sends them back to RISE.
- Step 6: RISE receives the refined results and presents them below the original ones (for comparison purposes). Therefore, the user can observe the differences between the two result sets and evaluate how effective the refinement process is.

3.2. User Interface design

The user interface of RISE was designed in Microsoft Visual Studio 2010 by utilizing the ASP.NET 4 web development model and the web controls that it comes with [53]. On the client side, interactions with the interface are done in JavaScript scripting, whereas interactions on the server side are achieved by C# programming.

In general, the interface of RISE is meant to be simple and easy to use so that it is user-friendly. Figure 3 below shows how the interface looks like before any interactions with the user take place.

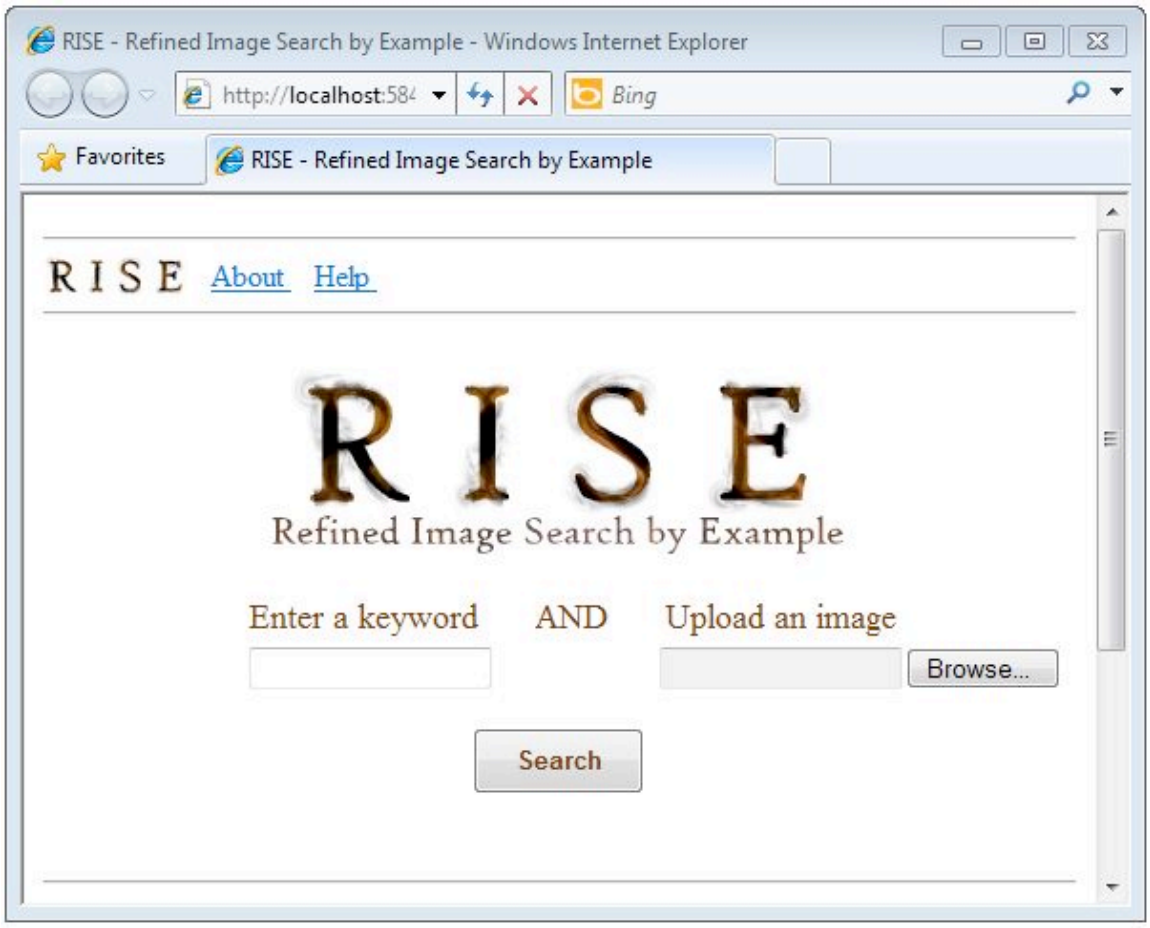

Fig. 3: Initial screen of RISE.

The user-interface components, as they are shown in Figure 3, are now discussed. On the top-left corner of the screen, the logo[2] of RISE was placed as well as two links, 'About' and 'Help', that lead to information regarding RISE itself and its usage. The main part of the screen is composed of a larger version of the logo and the actual search components, accompanied with instructional labels.

The search components are three: a textbox, a file uploader, and a button with the label 'Search'. The textbox serves as the container of the textual query, whereas the file uploader is a means of uploading the query image to the system. By clicking the 'Browse' button of the uploader, users can navigate their local file system and choose the image they wish to give as example for the search. The instructional labels above the textbox and the file uploader, i.e. 'Enter a keyword' and 'Upload an image', prompt users as to where exactly to put their search queries.

The 'Search' button triggers validation of the search. If both the keyword and the example image are provided, then clicking on the 'Search' button means that the search initiates and the interface is waiting for the results to be returned, which are subsequently presented on the screen. However, if a user does not provide both of the queries or he does not upload an image but any other type of file, then clicking the 'Search' button returns appropriate error messages and the search does not initiate. Figures 4, 5, and 6 show examples of failed search initializations.

[2] The logo of RISE was designed in Microsoft Expression Design 4.

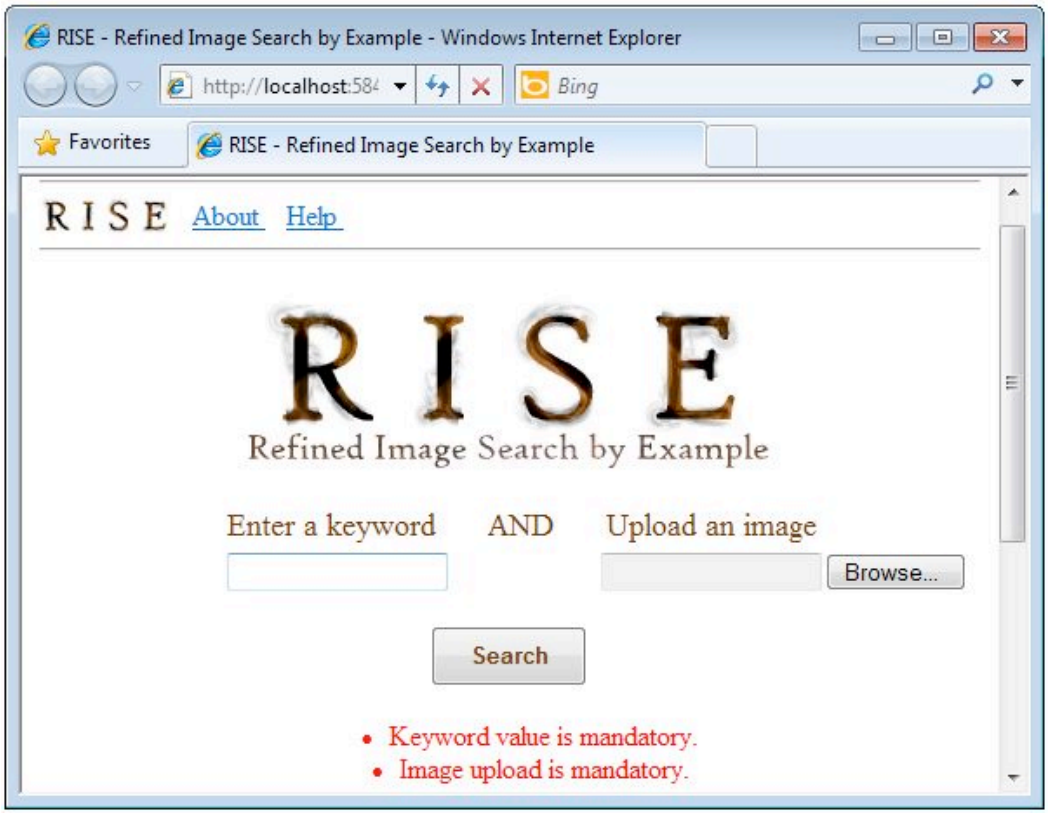

Fig. 4: Search initialization failure due to inexistence of both queries.

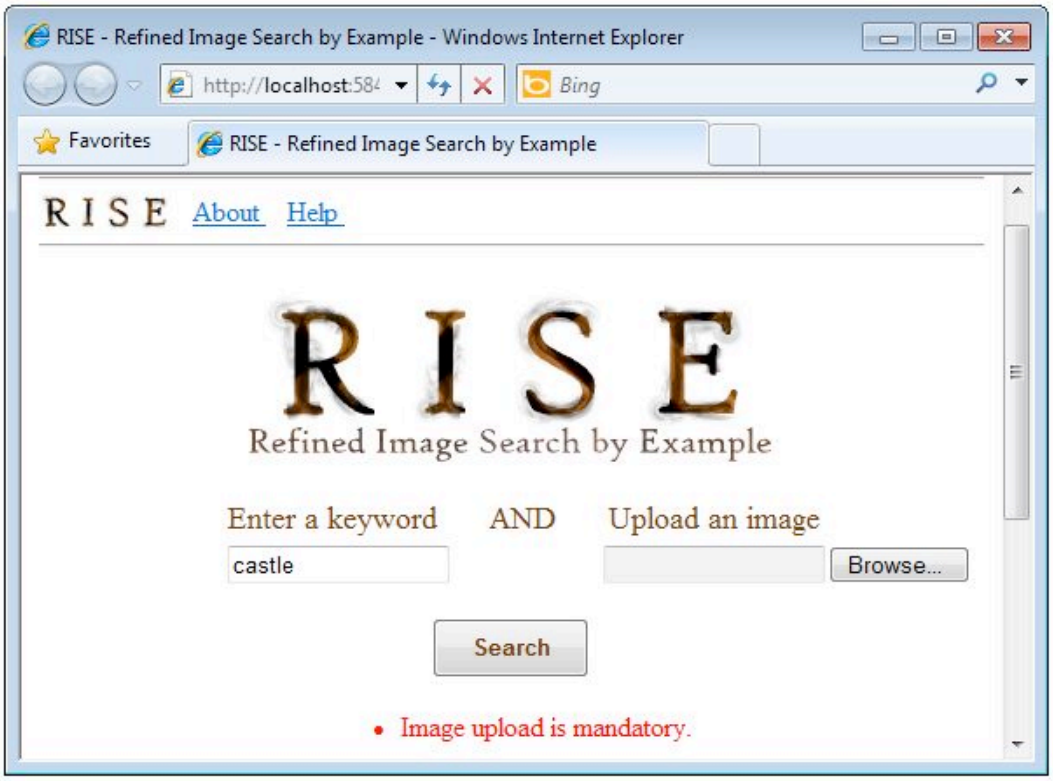

Fig. 5:

Search initialization failure due to inexistence of the image query.

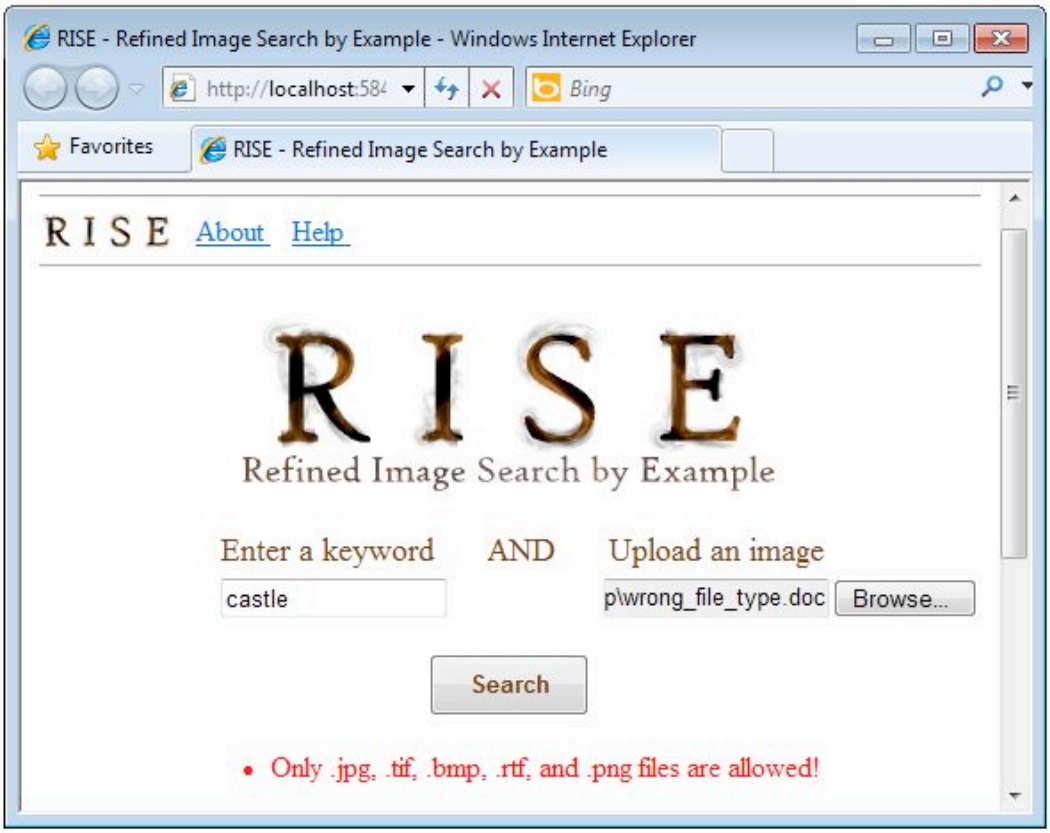

Fig. 6: Search initialization failure due to wrong file type upload.

When the search validates, i.e. both search queries were entered correctly, clicking of the 'Search' button triggers initialization of the search. The keyword entered in the textbox is used as the query of a new Google Image Search and RISE waits for the results to be returned so that it can continue with the search refinement phase.

As soon as the 'Search' button is clicked, an animated 'loading' image appears under the 'Search' button, showing that the system is searching. This loading image is kept on the screen until both sets of results, Google's and refined, are presented on the screen. In addition to the 'loading' image, the query image is also shown on the screen, preceded by the label 'Your image'. Figure 7 shows an example of the aforementioned changes on the screen, when RISE starts carrying out a search using the keyword 'castle' and a related query image.

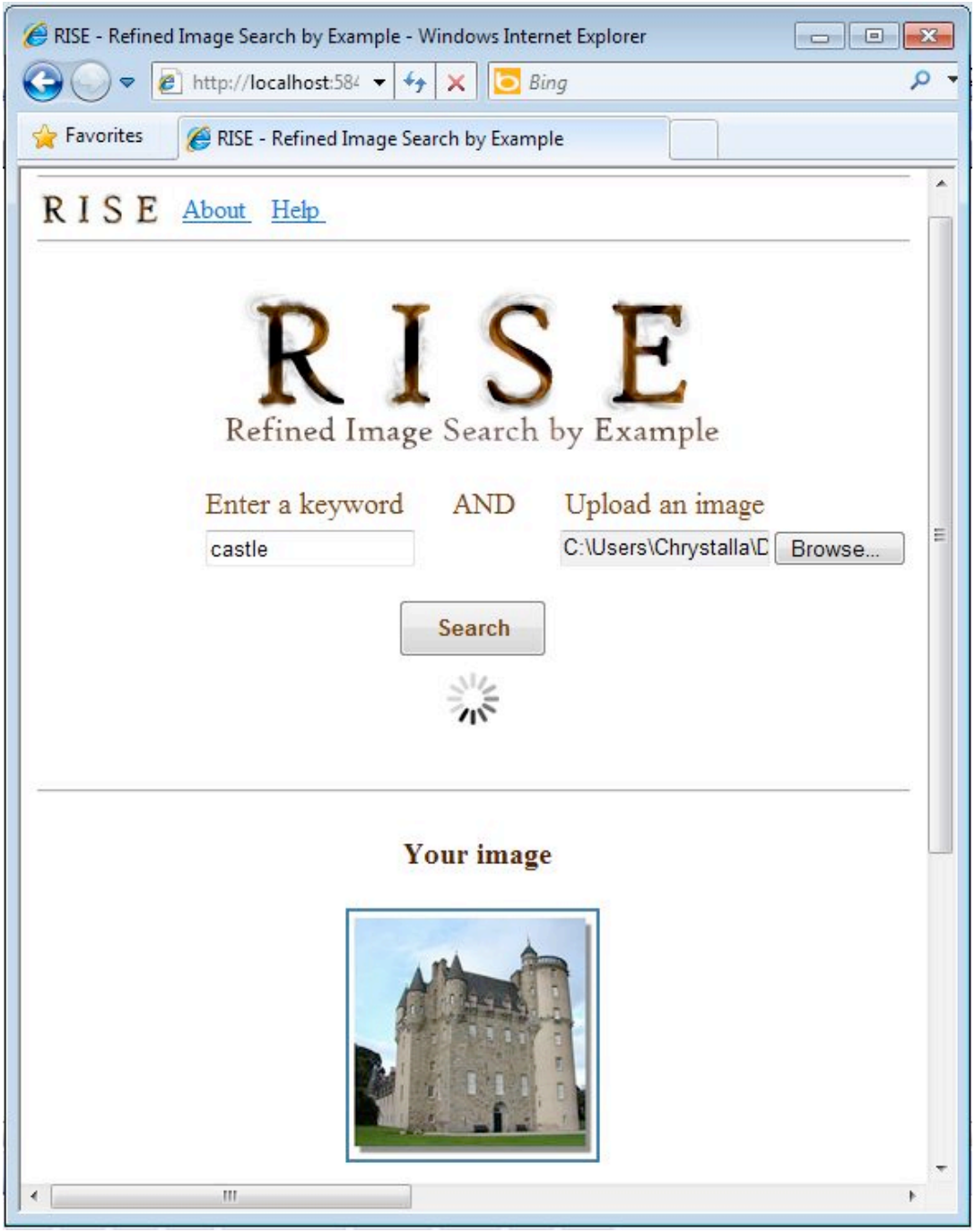

Fig. 7: Search initialization success and immediate changes on the screen.

Below the query image, the results from Google Image Search are presented, and when the refined results are ready they are also presented on the screen. For the presentation of both the Google Image results and the refined results the jCarousel plug-in was used. More information regarding this plug-in can be found in [54]. The results of example searches are discussed in

Chapter 4.

3.3. Feature extraction and ranking algorithm

As it was already mentioned in the previous sections, the feature extraction and ranking algorithm of RISE initiates when the results of Google image search are returned to the system. As input, the algorithm takes the query image and the URLs of the images returned from the search and it implements feature extraction based on both color and texture. The algorithm produces two sets of features for both the query and each of the resulting images. This process is described below.

- Feature set 1: The query image is converted from the RGB (Red-Green-Blue) color space to the HSV (Hue-Saturation-Value) color space and a 4-bin histogram is created for each of the three HSV channels. Each histogram is then normalized for the total number of pixels of the image, since we're dealing with different size images. Each histogram can be calculated using (2.2), but since we then normalize by the number of pixels then the resulting equation becomes:

$$h_m = \Pr ob\{M = m\} \tag{3.1}$$

The reason that the HSV color space was chosen is that, it corresponds more naturally to human perception. Hue corresponds to what color is used, Saturation to how saturated the color is, and Value to brightness, as can be seen in Figure 8. Therefore, the HSV-based features should be more useful for image retrieval purposes than RGB-based features, if features are to be produced independently from each of the three color-space coordinates.

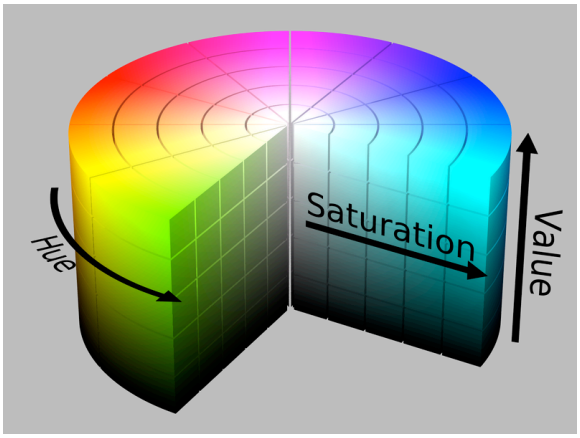

Fig. 8: The HSV color-space.

- Feature set 2: In addition, the mean value of the original RGB image is calculated and used as an approximation to the Luminance of the image. Therefore, a grayscale image is produced. Then, the Histogram of Oriented Gradients (HOG) [55, 56] is calculated for the grayscale image. This method counts occurrences of gradient orientation in localized overlapping regions (cells) of an image. For each cell, both the vertical and the horizontal gradient values are initially calculated by applying the following filter kernels: $[-1,0,1]$ and $[-1,0,1]^T$. In addition, gradient angle/orientation values are calculated for each pixel location using the horizontally and vertically filtered images. Then, the corresponding cell histograms are created. Every pixel within a cell casts a weighted vote, according to the gradient L2-norm, for an orientation-based histogram channel [55]. The cells can have rectangular or radial shape and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is "unsigned" or "signed". In the implementation used for RISE, which was provided by [55], it was assumed that gradient is unsigned and the cells rectangular, and that every image is divided into nine cells where each cells overlaps half of its area in each direction (horizontal or vertical). Furthermore it was assumed that each cell histogram is comprised of two orientation bins.

As a result, the overall gradient histogram for each of the images is composed of eighteen values, so the feature vector has length 18.

To conclude, the two aforementioned feature sets are calculated and the four feature vectors are extracted from each image. Then, for each feature vector of the query image and every corresponding vector of the resulting images, the Euclidean distance between the two feature vectors is calculated using (2.15) and is set as the similarity score for that feature. In order to produce the final similarity score, between the query image and a resulting image, the four feature similarity scores that are associated with the four feature vectors are weighted. This weighting quantifies how effective each of the four features is. For RISE, all such weights were set to 1, so all features have equal importance. Lastly, the four weighted feature similarity scores are summed up to produce the final similarity score.

The feature extraction and generation of similarity scores processes were initially developed in Matlab and then the Matlab Compiler was used to package the Matlab application into a .NET component [57]. Then, the .dll library of the .NET component was incorporated in C# code. As a result, by taking advantage of the fact that Matlab is a high-level language, a large number of features and feature combinations were evaluated first before concluding that the selected subset of 4 features is indeed effective.

CHAPTER 4

SEARCH RESULTS

This chapter discusses example results returned from six different searches using RISE, where the image ranking process is shown to be effective. For each search, the keyword used is listed and the query image is shown. Moreover, the first 8 images that were returned using the text-only search are shown and below them are shown the first 8 images that are returned by RISE, i.e. the refined/ranked results. As was previously mentioned, the jCarousel plug-in for JavaScript allows the user to observe the full results for both sets by clicking on the arrow on the right of the image sequence. Presenting the first 8 results (which correspond to the 8 best matches) is indicative of the performance of the algorithm. It is noted that the original number of images that are returned by the Google API is 64 at the most. The search results are limited to only medium-size images so that the search process is faster.

- Search 1 (keyword: ‘CSUCI Bell Tower’)

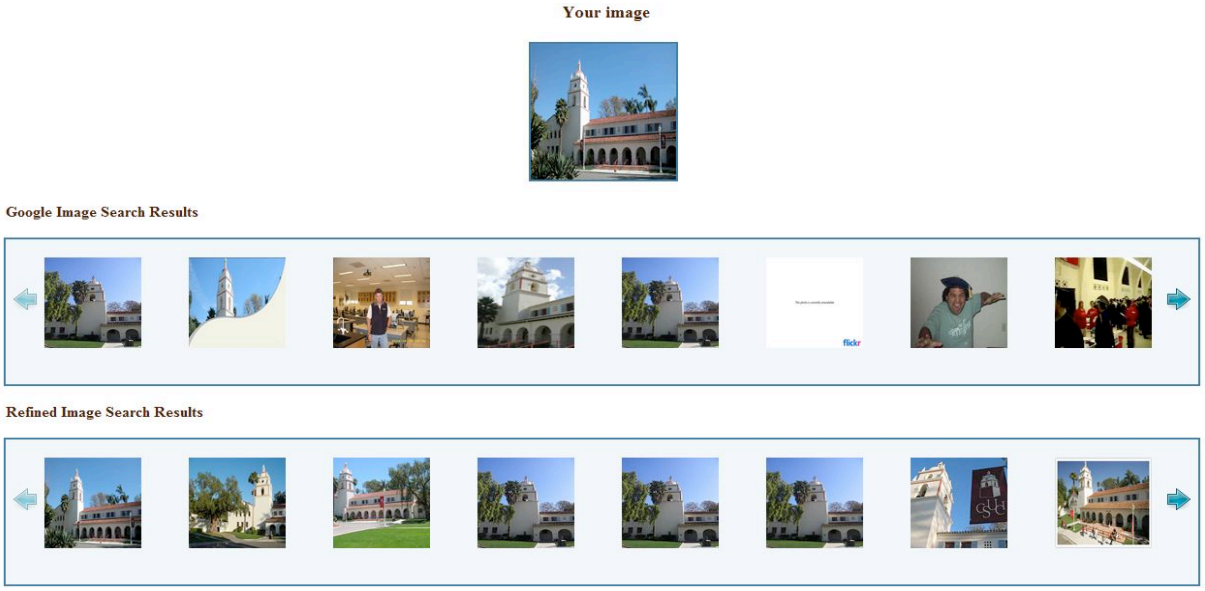

Fig. 9: Results of search with keyword ‘CSUCI Bell Tower’.

The results of this first search provide a good example of how RISE was able to identify the more relevant images to the query image. As it is can be seen, the building that we were looking for (i.e. ‘CSUCI Bell Tower’) appears in the first eight refined results, in contrast to the first eight Google results that contain four unrelated images. Also, it is noted that the first refined result of this search is the same as the query image – the two images vary only in their size.

- Search 2 (keyword: ‘The Alamo’)

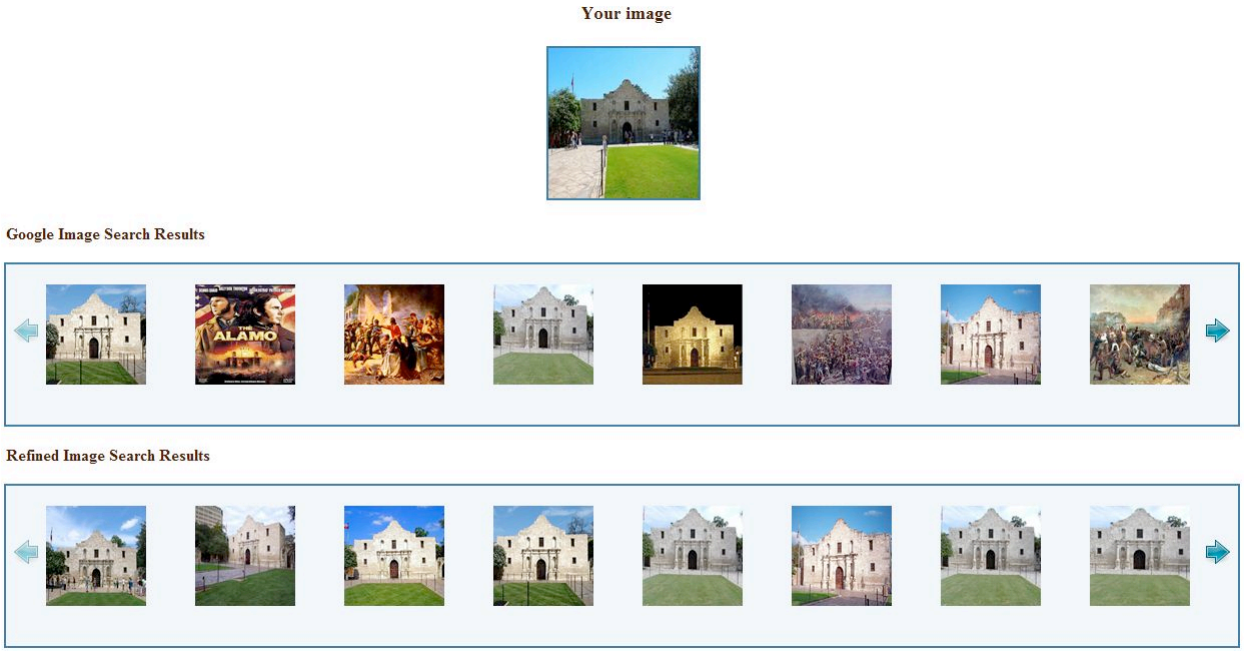

Fig. 10: Results of search with keyword ‘The Alamo’.

This search also provides a nice example of results refinement. The first eight Google results contain four images other than the building depicted in the query image. The first eight refined results, however, all show the building that is being searched, in different views. So, the combination of color and texture features seems to have worked very well in this case. The refined results of this example could be useful in systems that collect images in order to create 3D models of locations by utilizing different views, or in systems that need to create panoramic images.

- Search 3 (keyword: ‘No Reservations’)

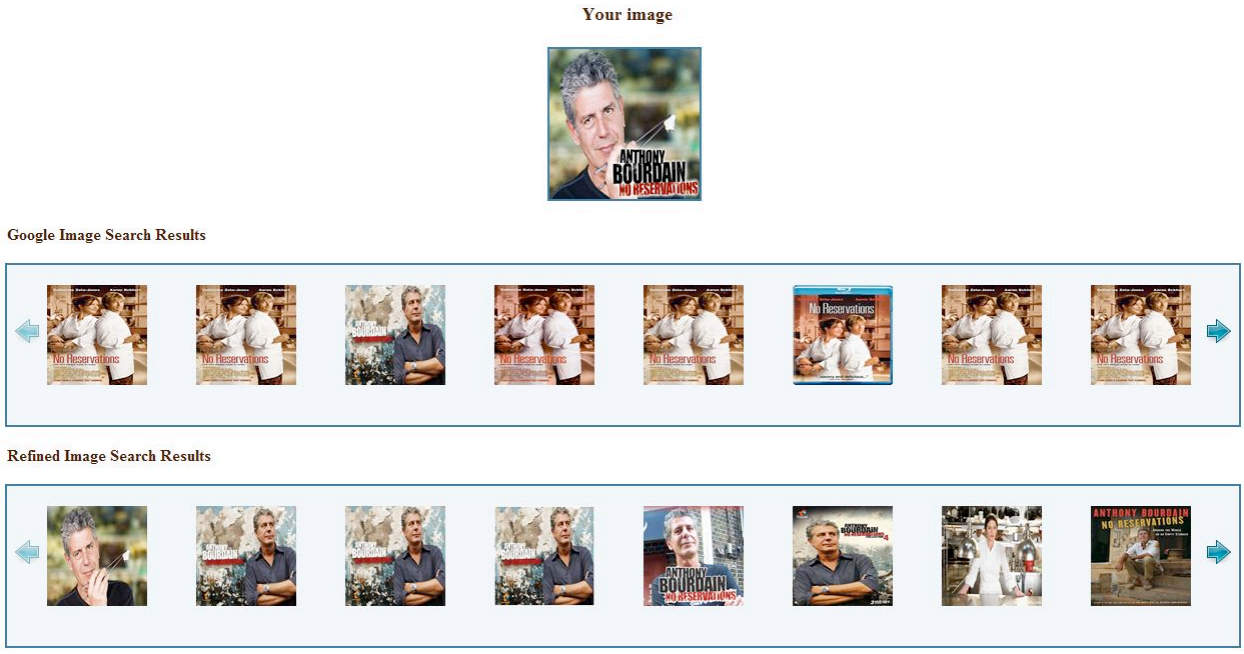

Fig. 11: Results of search with keyword ‘No Reservations’.

The textual query of this search is the name of a TV show and the image query a picture of the show’s host along with some graphics. As it can be seen, seven out of eight of the first refined results correctly depict the object of the search, and only one (the seventh) is irrelevant. On the opposite site, Google shows only one image relevant to the query image and seven irrelevant, since the name of a famous movie is also ‘No Reservations’. What is also shown from the above results, the first image returned in the refined results is actually the query image without the graphics in the bottom-right part of the query image.

- Search 4 (keyword: ‘Andy Warhol paintings’)

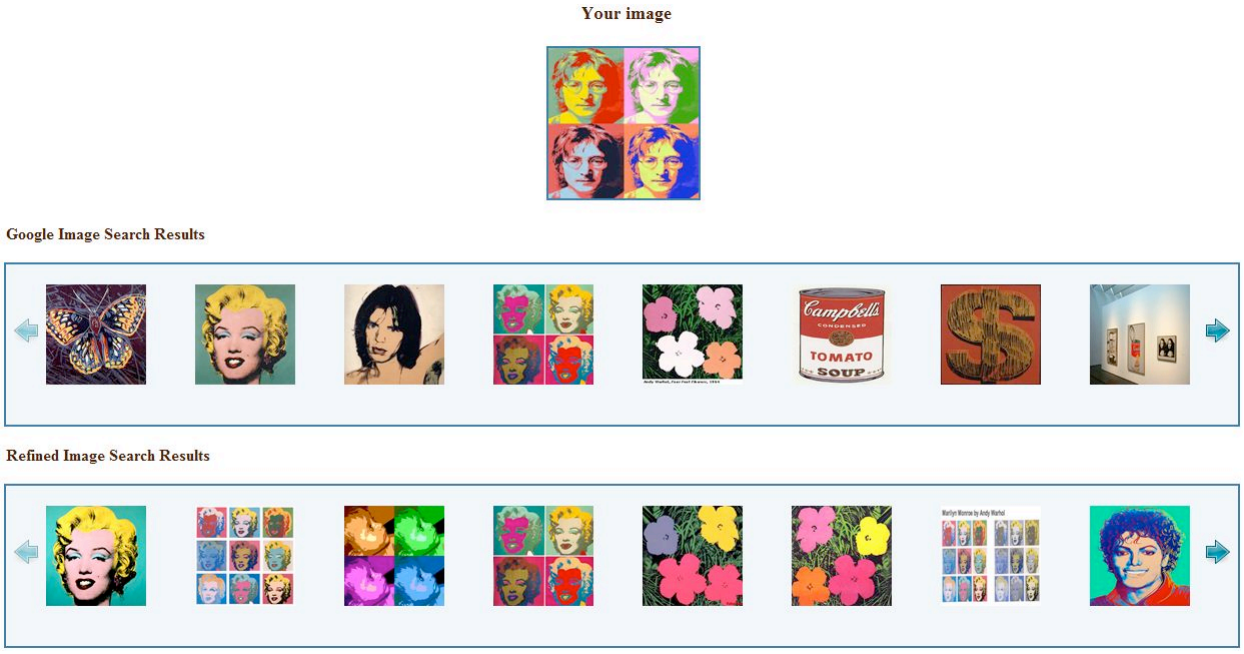

Fig. 12: Results of search with keyword ‘Andy Warhol paintings’.

In this example, the query image is a portrait of John Lennon created by Andy Warhol, so the target of the search was to find similar paintings of to the one shown in the query image, in terms of showing the same image in different colors (this is referred to as the ‘Andy Warhol effect’). Even though only 4 of the 8 images in the refined results show the ‘Andy Warhol effect’, all of them show images having various hues and high saturation, much-like the query image does. For example, even the image that depicts Marilyn Monroe just once is a more saturated version in the refined results set than the one shown in the Google results set. In addition, the texture-based feature could have had a lot to do with retrieving the 4 images having the ‘Andy Warhol effect’, since the gradients are calculated on various regions of the image (locally) and that could have captured the horizontal and vertical edges that are created by repeating the same image multiple times or using crosshair separations. Nevertheless, we need to point out the limitation that since RISE only uses low-level features, it wouldn’t have been possible to specifically search for other

Andy Warhol portraits of the same person as the one depicted in the query image – for example, Andy Warhol made additional portraits of John Lennon which could have been targeted in a search. To do that, we would need higher level features.

- Search 5 (keyword: 'Owl')

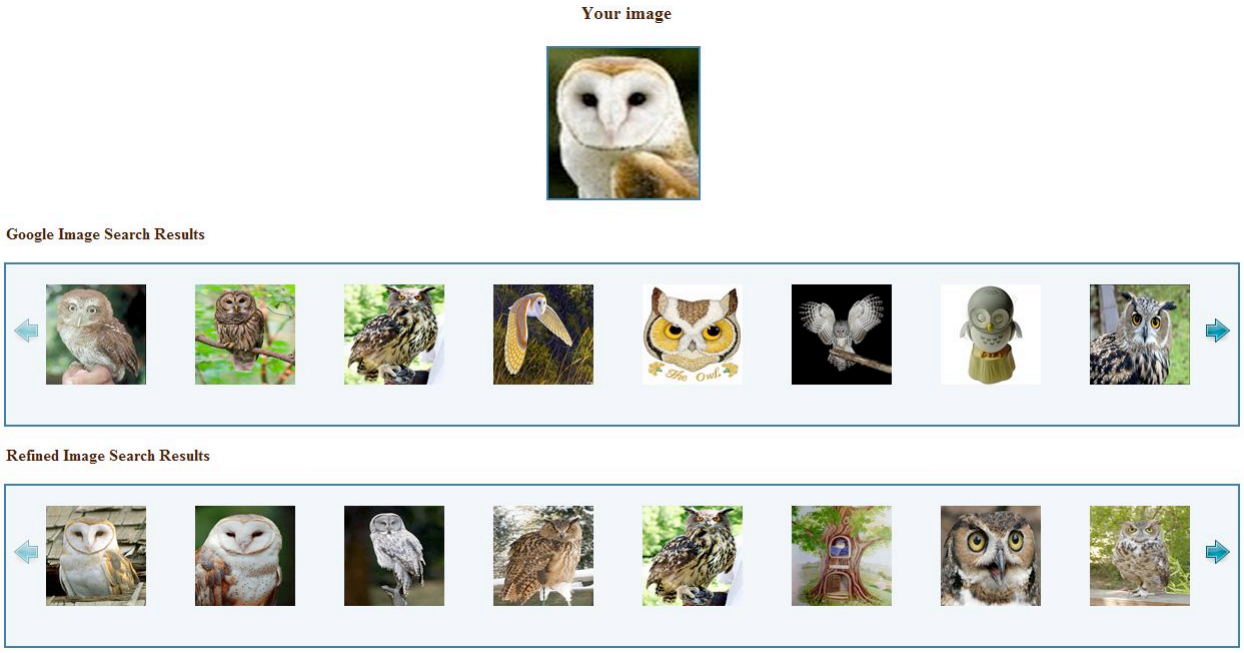

Fig. 13: Results of search with keyword 'Owl'.

Taking a look at this final search example, we can see that the first three refined results depict the same type of owl as the one shown in the query image. Also, the hue and saturation of the refined results is very close to the hue and saturation of the query image. This is not true for the 3rd result which has a noticeably different view, so the high-ranking of this result may be attributed to the brightness-based feature. Lastly, comparing Google's and refined results, we saw that the two graphics-created images (4th and 6th images) were ranked very low by the refined results, perhaps due to the very different texture that graphic images have. Lastly, it is mentioned that in the 55 original images that were returned by Google, the same type of owl was present in 2 additional images that RISE did not identify as one of the top 8. This is due to the

fact that the depicted size of those owls was a lot smaller with respect to the size of the image, so the main statistics that were captured by the low-level features were those of the background.

- Search 6 (keyword: 'Pluto')

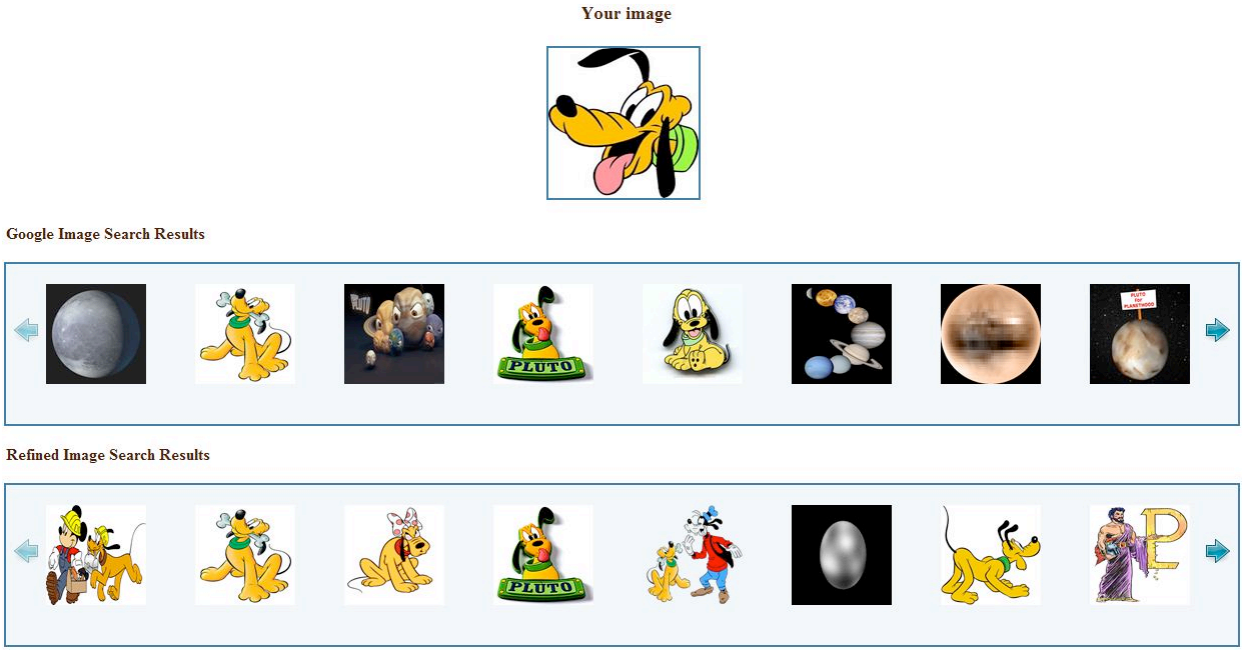

Fig. 14: Results of search with keyword 'Pluto'.

This search utilizes a graphical image of Disney's character 'Pluto'. If one simply uses the keyword 'Pluto' then, for the most part, images of the Disney character and the planet will be returned. So this is a case where a query image can be really useful. As it is shown, Pluto – the Disney character – appears in the 1st-5th and 7th refined results, in contrast to the original results, where Pluto appears in only three results. The 6th and 8th refined results are faulty. For the 6th refined result, specifically, we can say that it most probably appears in this position due to its smooth texture (in the large black region that surrounds the planet); the texture of the input image is also quite smooth since it is a graphical image. Nevertheless, RISE provided good overall results.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

6.1. Conclusions

This thesis concentrated on the development of the Refined Image Search by Example (RISE) system, which carries out web-based image searches and attempts to refine the results by utilizing an image analysis and ranking algorithm. The background research that preceded the development of RISE examined existing frameworks and systems for web-based image retrieval. The idea behind developing of RISE is the fact that search results returned from a search which combines a textual and a visual query can be more effective than a search that uses only one of these queries. Specifically, the image results of a text-based search are retrieved from the web using the Google Image Search JavaScript API. Subsequently, four low level features (histograms of oriented gradients, of hue, of saturation, and of brightness) are extracted from both the resulting images and the query image. Finally, a similarity measure is used to rank the order of the resulting images with respect to the query image. Therefore, the RISE image search results are presented in a refined and more relevant order using the user interface that was developed using ASP.NET and C#.

6.2. Future work

The first thing that can improve RISE is increasing the number of the resulting images that are acquired from the text-based search. Specifically, the third party data provider (Google Image Search API) restricts the number of images that can be used to a maximum of 64, therefore, this eliminates the possibilities of attaining more robust results. The larger the image set is, the

greater the possibility of obtaining more images similar to the reference image becomes. A solution to this problem could be the use of a custom web crawler that has access to more images on the web or the use a different third party data provider having fewer restrictions.

A second thing that can be changed in order to improve the effectiveness of the ranking process, is to utilize high level features. By doing so, semantically meaningful results can be obtained. Examples of such techniques include: relevance feedback for learning the users' 'intention', using machine learning methods to associate low-level features with query concepts, fusing the evidence from HTML text and the visual content of images [7], incorporating pattern recognition methods, etc. Of course, the use of larger sets of low-level features may also lead to better results, particularly if a training process can be implemented so that feature-specific weights can be produce in order to combine their similarity scores into one final score.

Finally, a drawback of RISE is the speed with which it returns the refined results. This can be solved with the use of a Graphical Processing Unit (GPU) or with the use of Cloud Computing resources.

REFERENCES

[1] M. L. Kherfi, D. Ziou, and A. Bernardi, 'Image Retrieval from the World Wide Web: Issues, Techniques, and Systems', *ACM Transactions in Computing Surveys*, vol. 36, no. 1, pp. 35–67, 2004.

[2] Y. Rui, T. S. Huang, and S-F Chang, 'Image Retrieval: Past, Present, and Future', In *Proceedings of the International Symposium on Multimedia Information Processing*, Taipei, Taiwan, 1997.

[3] A. A. Goodrum, 'Image Information Retrieval: An Overview of Current Research', *Informing Science*, vol.3, no.2, pp. 63-67, 2000.

[4] R. Datta, D. Joshi, J. Li, and J. Wang, 'Image Retrieval: Ideas, Influences, and Trends of the New Age', *ACM Transactions in Computing Surveys*, vol.40, no.2, 2008.

[5] Y. Rui, T. S. Huang, and S-F Chang, 'Image Retrieval: Current Techniques, Promising Directions, and Open Issues', *Journal of Visual Communication and Image Representation 10*, pp. 39-62, 1999.

[6] A. W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, 'Content-Based Image Retrieval at the End of the Early Years', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.22, no.12, 2000.

[7] Y. Liu, D. Zhang, G. Lu, and W-Y Ma, 'A survey of content-based image retrieval with high-level semantics', *Pattern Recognition*, vol. 40, no.1, pp. 262-282, 2007.

[8] H. T. Shen, B. C. Ooi, and K. L. Tan, 'Giving meanings to WWW images', *ACM Proceedings in Multimedia,* pp. 39–48, 2000.

[9] H-T Chang, 'Web Image Retrieval Systems with Automatic Web Image Annotating Techniques', *WSEAS Transactions on Information Science & Applications,* Issue 8, vol. 5, pp. 1313-1322, 2008.

[10] C. Frankel, M. J. Swain, and V. Athitsos, 'WebSeer: An Image Search Engine for the World Wide Web', Technical Report TR-96-14, Computer Science Department, University of Chicago, 1996.

[11] E. Di Sciascio, G. Mingolla, and M. Mongiello, 'Content-based Image Retrieval over the Web using Query by Sketch and Relevance Feedback', *Proceedings of the Third International Conference on Visual Information and Information Systems*, vol. 1614, pp. 123-130, Amsterdam, 1999.

[12] S. Sclaroff, M. La Cascia, S. Sethi, and L. Taycher, 'Unifying Textual and Visual Cues for Content-Based Image Retrieval on the World Wide Web', Computer Vision and Image Understanding, vol.75, no. 1-2, pp. 86-98, 1999.

[13] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, 'Relevance feedback: A power tool in interactive content-based image retrieval', *IEEE Transactions on Circuits Systems Video Technology*, vol. 8, pp. 644–655, 1998.

[14] M. L. Kherfi, D. Ziou, and A. Andbernardi, 'Atlas WISE: A Web-based image retrieval engine', *Proceedings of the International Conference on Image and Signal Processing (ICISP)*, pp. 69-77, Morocco, 2003.

[15] N. Ben-Haim, B. Babenko, and S. Belongie, 'Improving web-based image search via content based clustering', *Proceedings of the International Workshop on Semantic Learning Applications in Multimedia (SLAM)*, New York City, 2006.

[16] A. Popescu, G. Grefenstette, 'A Conceptual Approach to Web Image Retrieval', *Proceedings of the 15th international conference on Multimedia*, pp. 297-304, Germany, 2007.

[17] George A. Miller, 'WordNet: A Lexical Database for English', *Communications of the ACM* , vol. 38, no. 11, pp. 39-41, 1995.
http://wordnet.princeton.edu/wordnet/ (accessed July 10, 2010)

[18] M. Joint, P. A. Hède, and P. Adam, 'PIRIA: A general tool for indexing, search and retrieval of multimedia content', *Proceedings of SPIE Image processing: algorithms and systems*, vol. 5298, pp. 116-125, 2004.

[19] J. Li and J. Z. Wang, 'Real-Time Computerized Annotation of Pictures', *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, no. 6, pp. 985-1002, 2008.

[20] Idée Inc., 'TinEye: Reverse Image Search'.
http://www.tineye.com/ (accessed July 10, 2010)

[21] Attrasoft Inc., 'AttraSeek: Search a website with images'.
http://attraseek.com/ (accessed July 10, 2010)

[22] K. N. Plataniotis, A. N. Venetsanopoulos, 'Color Image Processing and Applications', Springer, Berlin, 2000.

[23] J. R. Smith and S-F Chang, 'Tools and techniques for Color Image Retrieval', *Symposium on Electronic Imaging: Science and Technology – Storage and Retrieval for Image and Video Databases IV*, vol. 2670, San Jose, CA, February 1996.

[24] M. Stricker and M. Orengo, 'Similarity of Color Images', *Proceedings on SPIE Storage and Retrieval for Image and Video Databases*, San Jose, CA, 1995.

[25] G. Pass and R. Zabih, ' Histogram Refinement for Content Based Image Retrieval', *IEEE Workshop on Applications of Computer Vision*, pp. 96-102, 1996.

[26] B. S. Manjunath, J-R Ohm, V. V. Vasudeyan, and A. Yamada, 'Color and Texture Descriptors', *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 703-715, June 2001.

[27] C-C Chiang, Y-P Hung, H. Yang, and G. C. Lee, 'Region-based image retrieval using color-size features of watershed regions', *Journal of Visual Communications, Image Representation*, vol. 20, pp. 167-177, Elsevier, 2009.

[28] M. Solli and R. Lenz, 'Color Semantics for Image Indexing', *5th European Conference on Colour in Graphics, Imaging, and Vision*, Finland, June 2010.

[29] R. S. Stankovi and B. J. Falkowski, 'The Haar wavelet transform: its status and achievements', *Computers and Electrical Engineering*, vol. 29, pp. 25-44, Elsevier, 2003.

[30] R. C. González and R. E. Woods, 'Digital Image Processing', pg. 827, Prentice Hall, 2008.

[31] R. M. Haralick, K. Shanmugan, and I. Dinstein, 'Textural features for image classification', *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 610-621, November 1973.

[32] H. Tamura, S. Mori, T. Yamawaki, 'Textural Features Corresponding to Visual Perception', *IEEE Transaction on Systems, Man, and Cybernetcs*, Vol. SMC-8, No. 6, pp. 460–472, June 1978.

[33] J. Malik and P. Perona, 'Preattentive texture discrimination with early vision mechanisms', *Journal of the Optical Society of America*, vol. 7, no. 5, pp. 923-932, 1990.

[34] C-W Ngo, T-C Pong, and R. T. Chin, 'Exploiting image indexing techniques in DCT domain', *Journan of the Pattern Recognition Society*, vol. 34, pp. 1841-1851, Elsevier, 2001.

[35] M. N. Do and M. Vetterli, 'Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback-Leibler Distance', *IEEE Transactions on Image Processing*, vol. 11, no. 2, pp. 146-158, February 2002.

[36] M. H. Pi, C. S. Tong, S. K. Choy, and H. Zhang, 'A Fast and Effective Model for Wavelet Subband Histograms and its Application in Texture Image Retrieval', *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 3078-3088, October 2006.

[37] B. S. Manjunathi and W. Y. Ma, 'Texture Features for Browsing and Retrieval of Image Data', *IEEE Transavtions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837-842, August 1996.

[38] M. Kokare, P. K. Biswas, and B. N. Chaterji, 'Texture image retrieval using rotated wavelet filters', *Pattern Recognition Letters 28*, pp. 1240-1249, Elsevier 2007.

[39] F. Liu and W. R. Picard, 'Periodicity, directionality, and randomness: Wold features for image processing and retrieval, *IEEE Transactions on PAMI.*, vol. 18, no. 7, pp. 722-733, March 1995.

[40] A. Abdesselam, 'Texture Image Retrieval Using Fourier Transform', *International Conference on Communication, Computer and Power (ICCCP),* pp. 343-348, February 2009.

[41] F. Long, H.J. Zhang, and D.D. Feng, 'Fundamentals of content-based image retrieval', *Multimedia Information Retrieval and Management*, Springer, Berlin, 2003.

[42] D. Zhang and G. Lu, 'Review of shape representation and description techniques', *Journal of the Pattern Recognition Society*, no. 34, pp. 1-19, Elsevier 2004.

[43] D. Zhang and G. Lu, 'Generic Fourier Descriptor for Shape-based Image Retrieval', *Proceedings of IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 425-426, Switzerland 2002.

[44] S. Abbasi, F. Mokhtarian, and J. Kittler, 'Curvature scale space image in shape similarity retrieval', *Proceedings of the British Machine Vision Conference,* pp. 53-62, Edinburgh, UK, 1996.

[45] D. Zhang and G. Lu, 'A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures', *Proceedings of the International Conference on Multimedia and Distance Educatiom, pp. 1-9, USA, 2001.*

[46] M. E. Celebi and Y. A. Aslandogan, 'A Comparative Study of Three Moment-Based Shape Descriptors', *Proceeding of the International Conference on Information Technology, Coding and Computing*, pp. 788-793, USA, April 2005.

[47] K. Chakrabarti, M. Ortega, K. Porkaew, P. Zuo, and S. Mehrotra, 'Similar Shape Retrieval in MARS', *Proceedings of IEEE International Conference on Multimedia and Expo*, vol. 2, pp. 709–712, New York, USA, 2000.

[48] Y. Deng, B. S. Manjunath, C. Kenney, M. S. Moore, and H. Shin, 'An Efficient Color Representation for Image Retrieval', *IEEE Transactions on Image Processing*, vol. 10, no.1, pp. 140-147, January 2001.

[49] S. Belongie, C. Carson, H. Greenspan, and J. Malik, 'Color- and Texture-Based Image Segmentation Using EM and its Applications to Content-Based Image Retrieval', *Proceedings of the International Conference in Computer Vision*, pp. 675-682, 1998.

[50] C. Schmid and R. Mohr, 'Local Grayvalue Invariants for Image Retrieval', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530-535, May 1997.

[51] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, 'Learning a Mahalanobis Metric from Equivalence Constraints', *Journal of Machine Learning Research*, no. 6, pp. 937-965, 2005.

[52] 'Google Image Search API',
http://code.google.com/apis/imagesearch/v1/devguide.html
(Accessed: November 11, 2010)

[53] 'ASP.NET 4 Framework documentation',
http://msdn.microsoft.com/en-us/library/ee532866.aspx
(Accessed: November 12, 2010)

[54] 'jCarousel – Riding carousels with jQuery',
http://sorgalla.com/projects/jcarousel/#Dynamic-Content-Loading
(Accessed: November 12, 2010)

[55] O. Ludwig, D. Delgado, V. Goncalves, and U. Nunes, 'Trainable Classifier-Fusion Schemes: An Application To Pedestrian Detection,' in: 12th International IEEE Conference On Intelligent Transportation Systems, vol. 1, pp. 432-437, 2009.

[56] C. Premembida, O. Ludwig, J. Matsuura, and U. Nunes, 'Exploring Sensor Fusion Schemes for Pedestrian Detection in Urban Scenarios', in: Iros 3rd Workshop: Planning, Perception and Navigation for Intelligent Vehicles, pp. 17-22, 2009.

[57] 'Matlab .NET Builder',
http://www.mathworks.com/help/toolbox/dotnetbuilder/
(Accessed: November 14, 2010)