INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

EagleEye

Large photoset visualization

Carlos Eduardo Henriques de Jesus Fonseca

Dissertação para a obtenção de Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente:	Nome do Presidente
Orientador:	Daniel Gonçalves
Vogais:	Nome do Vogal 1
	Nome do Vogal 2
	Nome do Vogal 3

Mês e Ano

Dedicated to someone special...

Acknowledgments

A few words about the university, financial support, research advisor, dissertation readers, faculty or other professors, lab mates, other friends and family...

Resumo e palavras chave

Inserir o resumo em Português aqui com o máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave...

Palavras-chave: fotografias, visualização de imagens, exploração de imagens,...

Abstract and keywords

Insert your abstract here with a maximum of 250 words, followed by 4 to 6 keywords...

Keywords: photographs, image visualization, image browsing,...

Contents

List of Tables

List of Figures

Acronyms

UI user interface

UX user experience

SOM self organizing map

CBIR content based image retrieval

MP megapixel

FEP feature extraction plugin

WPF Windows Presentation Foundation

EXIF Exchangeable image file format

HDR High Dynamic Range

Chapter 1

Introduction

Since the advent of digital formats, photography has become much more common among people, and photo collections have started to grow more than they used to. But storing digital photographs isn't that different from the past. In fact, people have been storing their photographs on folders on their computers instead of photo albums on the shelf. One could say computers can help people view, explore and find more photos in a shorter period of time, they certainly have the power for that but, in the end, they usually don't do much more than make the user look for the photo album (probably a folder or an "album" on some applications) and then flick through its pages (scrolling) until the photos the user was looking for appear.

This is true for the most common photo management software on the market, like Google's Picasa[1], Apple's iPhoto[2] or Aperture [3], or Adobe's Lightroom [4]. Although they bring some management improvements with them, the analogy above still applies. They all provide a way to select albums or folders and scroll through contents. They allow searching through some existing metadata, the only metadata that some of them generate is only face detection and end up having lots of buttons, toggles and options for their editing capabilities that clutter the interface.

TODO: talk more about photography and panoramas and HDR's and bursts and things! Photography, man!

1.1 Our vision

We want to provide the users a totally different way to interact with their photos by focusing only on them. We want to provide the user a birds-eye view of his photos by showing everything and allowing him to drill down and view any image he wants at a larger size. Eagle Eye is not meant to be an editing platform but a better visualization tool.

With this work we will show that this is a good alternative way to look at the users photo collection and that the user can learn more things from it.

[1] http://picasa.google.com
[2] http://www.apple.com/iphoto
[3] http://www.apple.com/aperture
[4] http://www.adobe.com/lightroom

1.2 Contributions

With our work, we have understood that visualization of large collections at the same time is not only feasible but also interesting to the users.

We observed that images can be really small and still be identifiable by their owners, as long as they are integrated with others of the same event.

TODO: moar stuff here

1.3 Structure of the Document

We will start by going through some previous work related to ours (chapter 2) followed by an identification of the requirements for our work (chapter 3). Next, we will see what was implemented and how it works (chapter 4) and how the users responded to it (chapter 5). To conclude, we will discuss some work that could be done to improve this thesis (chapter 6.1) and final thoughts to conclude (chapter 6).

Chapter 2

Related Work

Interactive image visualization techniques have been explored for some time now, many of them related to image organization or retrieval in large collections. There have been some interesting ideas across the board and we will now take a look at some of them.

2.1 Visual guided navigation for image retrieval

Qiu et al. [47] explore the requirements of a system intended for visualizing large photo collections. They identify as the two most important requirements, the first being an easy to use UI, that gives clean information to the user and helps to create a mental image of the whole collection helping him to navigate on the collection. The second requirement is responsiveness because while image processing can be an heavy task, the user needs to be able to interact with the interface and he won't use the application if it's slow.

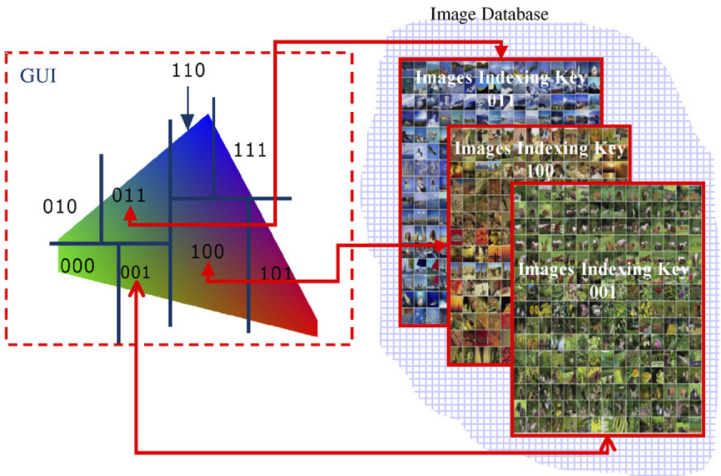


Figure 2.1: The chromacy diagram is split in parts and each image belongs to one of this parts. The diagram is part of the UI and when navigating through the diagram, only the images related to that part are shown.

The system shows all the photos arranged by color, just as many others like it. The difference is

the process in use. Instead of calculating distance vectors based on the histogram of each image, this approach classifies each image with a simple description, like a mean of its colors, and arranges them by that value, on a color map (fig. 2.1). The process is much faster but is also more error prone, specially on photos without a clear main color.

Their tests show they achieved good responsiveness and better results than using a file explorer.

2.2 Does organization by similarity assist image browsing?

Figure 2.2: Three arrangements of 100 images of Kenya, based on visual similarity. On the left is the arrangement with overlap, in the middle a 12x12 grid (which removes the overlap while preserving some of the structure), and on the right a 10x10 grid (which maximises the thumbnail size).

The aim of this work by Rodden and Sinclar [50] was to evaluate how photo organization by similarity (fig. 2.2) could benefit a user looking for images. Some users tested an application that could show the same images both in a random and in an organized by similarity way. This organization by similarity was based on a rough description of the images but it could be other descriptors.

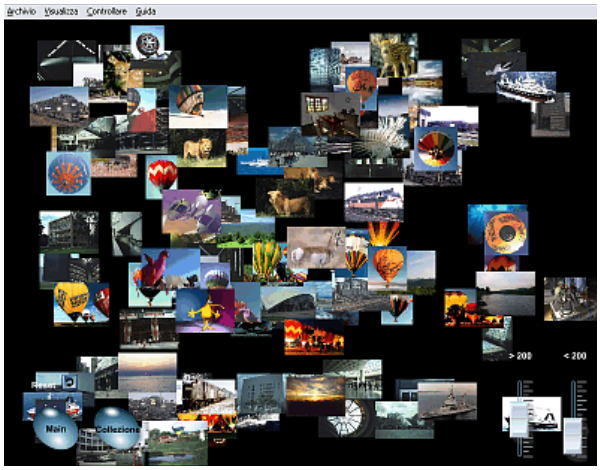
The results differ if the user knows what he's looking for or not. In case he does, being able to filter only the relevant images makes it quick to find the ones that matter. This obviously depends on the quality of the labeling. Users reported that sometimes the similar images appear to merge.

In case the user doesn't know what he's looking for, the random approach might be helpful because the strong images usually contrast to their neighbors and thus appear to stand out.

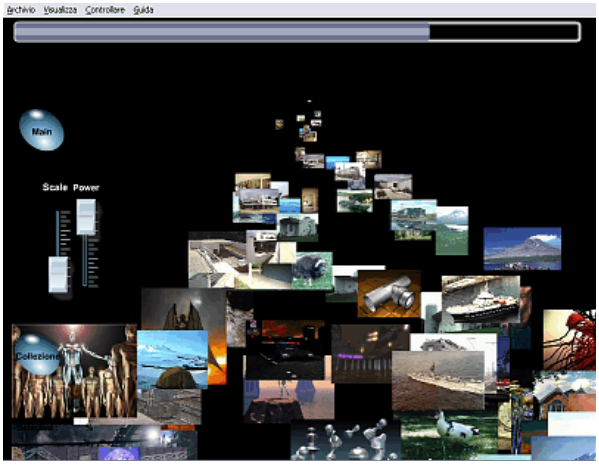
For some people, having access to different arrangements of the same set of images is useful, although the source of the individual differences still needs to be determined.

2.3 Browsing large collections of images through unconventional visualization techniques

Porta [46] describes a few visualization methods for large collections of images and tests them with users. The purpose it to find ways or metaphors that provide a good visualization experience in terms of time spent and quality of the visualization.



2.3.1: Spot display



2.3.2: Shot display

Figure 2.3: The two better visualizations from Porta's work.

Some of the various techniques were the simple image grid view, a grid view with variable and independent height and width (EIB), a view that animates images like they were shot from a distance and get closer to the user called Shot (fig. 2.3.2), a view where images quickly appear on random positions on screen named Spot (fig. 2.3.1), and some other less commons like one that simulates an cylinder created with images (Cylinder), and others less relevant (Rotor, Tornado and Tornado of Planes).

The testing was based on the efficiency of users searching for specific images on a collection of a thousand photos. The Spot view was the best, followed the Shot, Cylinder finally the common grid view. The other views got scores near or below the grid view.

2.4 A comparison of static and moving presentation modes for image collections

This paper by Cooper et al. [15] is not about large libraries but about what kind of interfaces for image showing has greater success of user recognition and preference (fig. 2.4).

With the help of eye-tracking techniques and user preference, the authors determined that static images are better than moving ones because makes them easier to recognize and avoid some user confusion.

2.5 Organizing and browsing photos using different feature vectors and their evaluations

Although it doesn't mention why, Strong [55] focuses on the better experience provided by color organization of a large image collection (fig. 2.5).

A self organizing map (SOM) is used to display the images on the screen featuring zooming, panning

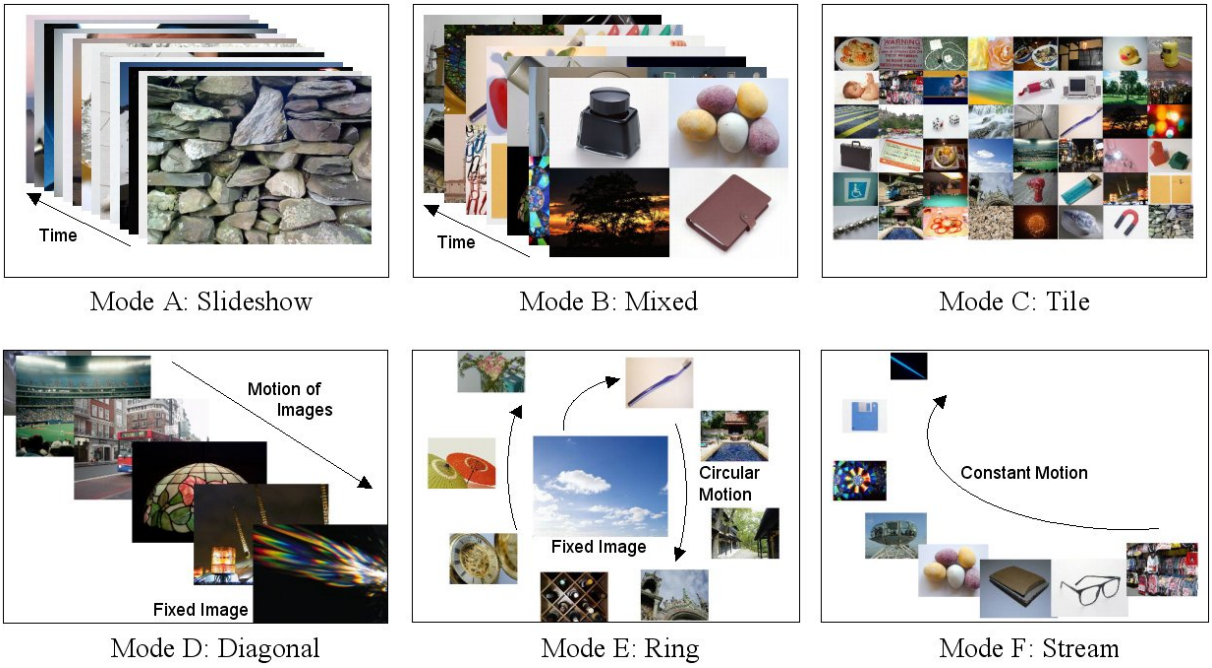


Figure 2.4: The six rapid serial visual presentation modes used in the experiments

and sorting capabilities. The work is then based around the various methods used to determine the images' similarity.

Simpler methods are based on color histograms, which aren't affected by rotations or scales but, by not having spacial information on colors, allows very different images be closer together.

Other methods rely on gradients which contain spatial information and, therefore, are sensitive to image contents but not color. In general, the best methods were found to be the hybrid ones, where both color histograms and gradients are used to classify the images. No user testing is made in this project, neither is the speed of image categorization referred about the methods used.

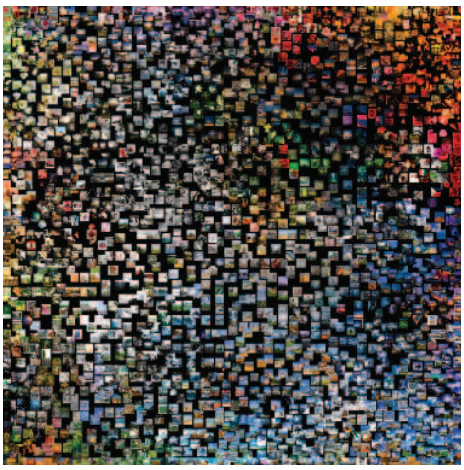


Figure 2.5: The result obtained for organizing 2200+ photos using color autocorrelogram feature vector, using [55]

2.6 An evaluation of color-spatial retrieval techniques for large image databases

Tan et al. [58] present an evaluation of three color-spatial image retrieval techniques.

The signature-based technique creates a signature for each image, based on the most frequent colors, according to a threshold, of each subdivision, or bin, of that image. The comparison between images is made by comparing the main colors present on each bin. It is possible to assign more weight to specific bins according to the user's interest.

The partition-based approach is also based on bins, each having it's own color histogram. The similarity between images is given by the distance of the histograms of the corresponding bins.

The cluster-based method bases on the fact that humans focus on large patches (clusters) of the same color and, therefore, two images will appear similar if both have similar colored clusters on at roughly the same location. This method extracts the larger clusters and their color from each image. The similarity is calculated by the amount of overlap between clusters.

This techniques were tested with a collection of 12,000 images and, besides the color information, the brightness was also analyzed for increased performance. The authors conclude the signature method was generally better on both effectiveness and efficiency.

2.7 Automatic organization for digital photographs with geographic coordinates

In this paper, by Naaman et al. [42], is described a system that organizes digital photographs accordingly to location and date embedded on the metadata.

The objective was trying to mimmic the the way people think about their collections. Photos are usually bursts separated by some time. Based on this and on the different places, events can be created to agglomerate photos from the same bursts. Location naming is done by calculating the most relevant places, like parks or cities, and then mixing the more precise locations with the more important neighbor cities to create a relevant and identifiable name. This was specially important since this work didn't involve showing any maps but only the location names and events.

The authors showed good results and, nowadays, some common applications use similar features although including maps.

2.8 Similarity pyramids for browsing and organization of large image databases

Chen et al. [11] present a method for designing a hierarchical browsing environment called a similarity pyramid. The similarity pyramid groups similar images together while allowing users to view the database at varying levels of resolution. Each level is organized such that similar images are in close proximity

on a two-dimensional grid (fig. 2.6). Images are first organized into a binary tree through agglomerative clustering based on color, edge and texture similarities. The binary tree is transformed into a quadtree, a tree in which each node has four children instead of only two.

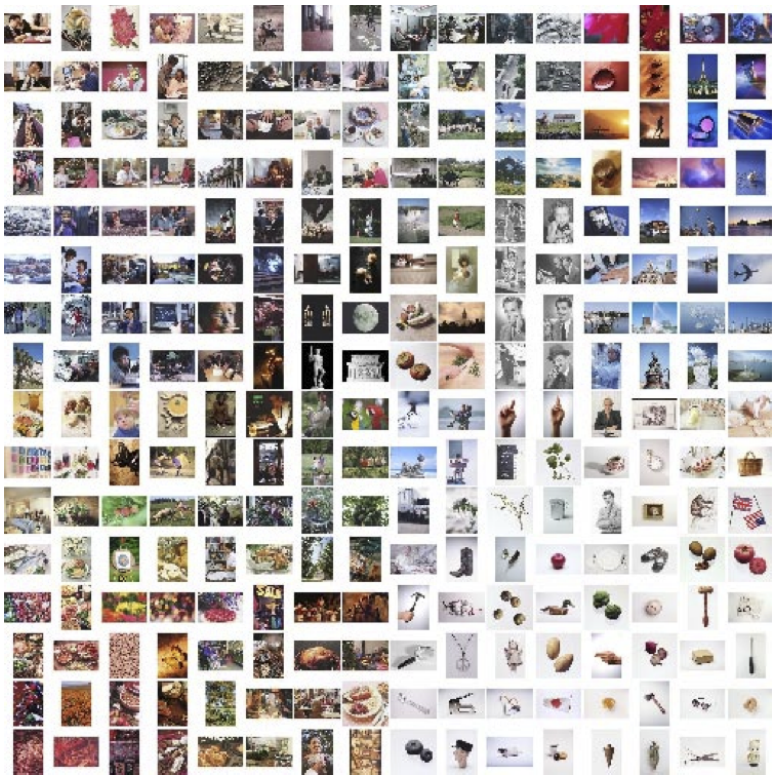


Figure 2.6: Images organized with [11]. Images with similar color and texture are spatially adjacent.

The similarity pyramid is best constructed using agglomerative (bottom-up) clustering methods, and a fast-sparse clustering method is presented which dramatically reduces both memory and computation over conventional methods. This method is based on the flexible agglomerative clustering algorithm, but using only a sparse proximity matrix and exploiting the author's approximate branch and bound search algorithm.

The authors found that the method for mapping the clustering to a pyramid can make a substantial difference in the quality of organization. Finally, a dispersion metric for objectively measuring pyramid organization was proposed, and found that it correlated well with the author's subjective evaluations of pyramid organization.

2.9 NN^k networks for content-based image retrieval

Heesch [29] describes a different interaction technique for content based search in large image collections. Each image is a vertex in a graph and arcs are established between images if there exists at least one combination of features for which one image is retrieved as the nearest neighbor of the other. Each arc is weighted by the proportion of feature combinations for which the nearest neighbor relationship holds. By integrating the retrieval results over several feature combinations, the resulting network helps

expose the semantic richness of images.

The interface reflects the vertexes and respective arcs, allowing to browse between the related images (fig. 2.7) in the network.

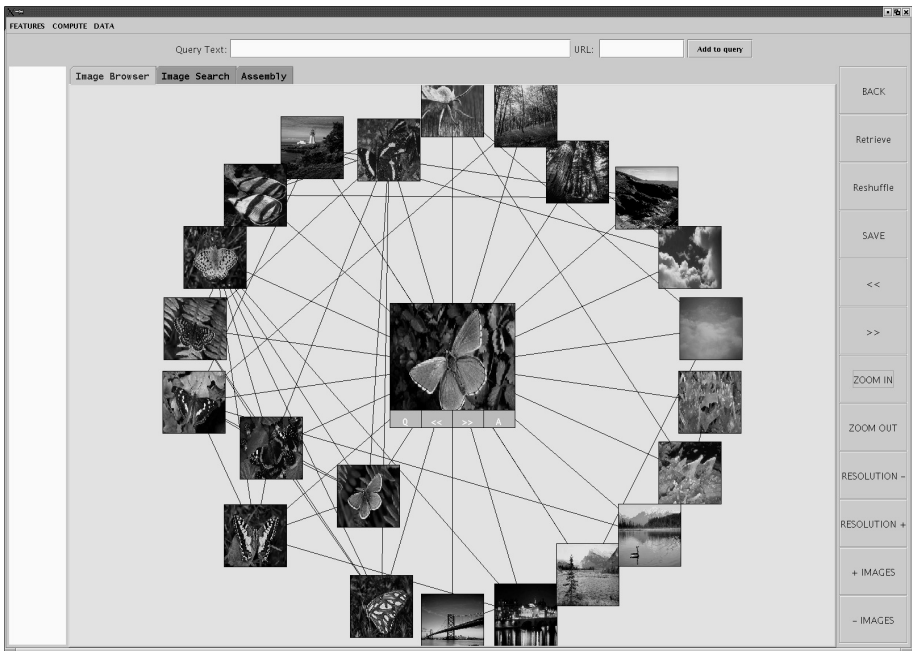


Figure 2.7: Local network around the chosen butterfly image depicted in the middle.

Seven low-level features are used for the classification of the images. HSV Global color Histograms maps the images by color, saturation and brightness; color Structure Descriptor maps the distribution of colors by dividing each image in 64 windows, the color space in 184 bins and associating color bins with image windows; Thumbnail feature compares identical images by saving the grey value of each pixel from a scaled down version of the image; Convolution filters discovers very selective features by reapplying 25 filters three times; Variance feature calculates standard deviations within a sliding window; Uniformity Feature is another statistical feature, calculating the grey level of an image split in 64 parts; Bag of words is the last feature and weights words associated to each image.

Tests showed great results using a mix of search, relevance feedback and browsing, and even only browsing was considerably better than other, more restrictive, approaches.

The technique helps avoid the problem of image polysemy by showing all gathered meanings of the images to the user. The feature network is pre-computed, allowing for quick realtime browsing. The authors claim it took 50 hours to process 32000 images but make no reference to the possibility adding images to the collection, after the computation.

2.10 Phorigami: A Photo browser based on meta-categorization and origami visualization

Hsu et al. [33] try to ease the browsing problem by analyzing the collections and identifying groups of related pictures. Each type of group is visualized in a specific way, inspired by the Origami art.

Groups of similar or related photos were manually classified based on camera movement and subject movement, creating different types of groups static view where both camera and subject are fixed and is presented as a panorama; multi-view where the subject is fixed but the camera is moving and is shown as a presentation; if the subject is moving, the photos are categorized as motion capture and can be shown as an animated photo (fixed camera) or a presentation (moving camera); finally group photos, where different groups of people are photographed, are shown as a folding presentation.

This covers various cases where the photographer takes a few similar photos of the same subject because it's either a panorama, various angles or just to be sure the photo was well captured.

The interface implements the different presentation types as different metaphors, easy for the user to understand, like a folded paper on a wide panorama that can be expanded (fig. 2.8). Although some of them appear to be a little hard to distinguish in its compressed form, it shouldn't be difficult to make it clearer. Other possible problem is the use of different touch interactions for each presentation type that might confuse users on what gesture should they use.

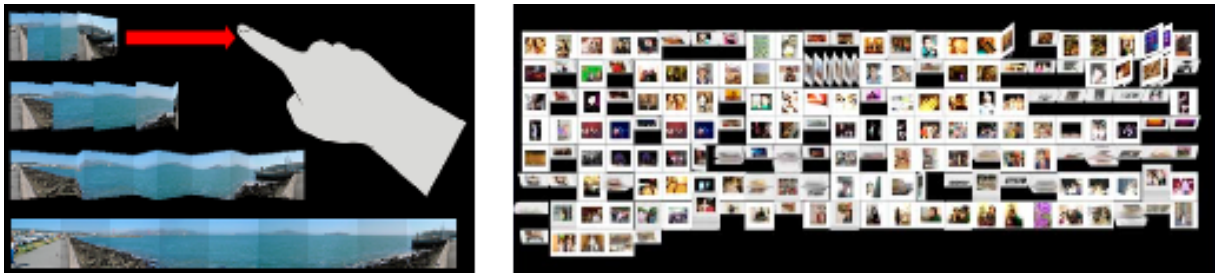


Figure 2.8: On the left, an example of an interaction on a group of photos that makes a panorama. On the right, a visualization on 537 photos with some groups.

2.11 A next generation browsing environment for large image repositories

Schaefer [53] tries to take similarity based organization of images from the 2D space to a 3D sphere, which allows interaction from the users. Rotating the sphere reveals images with different colors while tilting it reveals brighter or darker images.

Large image collections are handled through a hierarchical approach that brings up similar, previously hidden, images when zooming in on an area.

The description of the color is retrieved by calculating the image's median color for it's efficiency over other methods like histograms. This two features are directly mapped onto the sphere's coordinates and

Figure 2.9: Hue sphere of a dataset

the entire structure is pre-calculated so browsing can be performed in real-time. Image overlapping is also avoided (fig. 2.9).

The work was tested on a 4500 image collection with no evaluation as to its performance and a weak and subjective user testing.

2.12 Flexible access to photo libraries via time place, tags, and visual features

MediaGLOW by Girgensohn et al. [26] is the application discussed in this paper. It's a content based image retrieval (CBIR) system with multiple ways to filter and sort the image collection.

The interface allows selecting a range of dates, places and tags at any time to filter the collection and the display will show the photos that match the filters, alongside indications of the existence of photos that match some of the filters. This display can then be arranged by four similarity criteria: temporal (by photo creation time), geographic (distances between places), tags (photos with similar tags are shown closer together) and visual (fig. 2.10).

The time is selected using a timeline on the top of the screen while tags and places are shown on the right side sorted alphabetically, by frequency of, in case of places, as a tree. Multiple selections are allowed to show more photos.

The photo display is graph based, allowing for overlapped images. Various metaphors were developed to ease the navigation of the collection. Zooming is allowed and changes both thumbnail positions and size for better experience, allowing the photos to spread away from each other but also increasing the size so the user can have a better look at them. The authors think that bigger thumbnails and a correct grouping of related photos is more important than spreading them to avoid the overlapping problem.

color coding is used throughout the interface to help the user understand better what is being se-

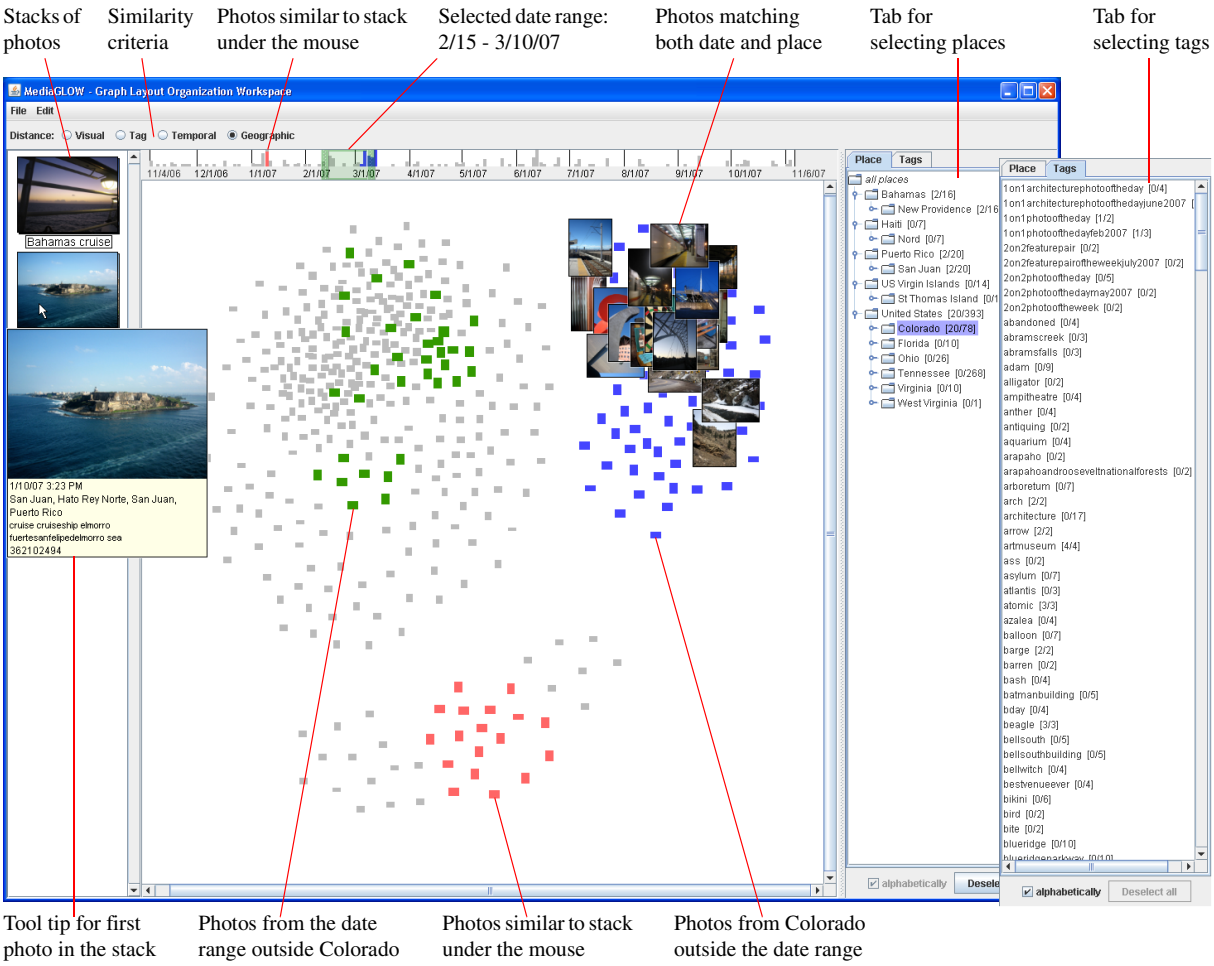


Figure 2.10: Photos Grouped by Geographic Similarity and Filtered by Date and Place.

lected. For instance, on the timeline blue and grey are used to distinguish photos that match or not the selected location/tag. On the photo display, besides the photos that are actually shown are colored blocks: blue for photos that match location only, green for time only and grey for those that didn't match anything.

Detailed user testing was performed and the importance of the multiple ways to organize and search the collections was emphasized since many systems are designed to have a single form of access. Some users also pointed the importance of being able to have a non overlapping view of the photos for part of the task.

Each view was found to have different levels of usage, the geographic being the most used and temporal the least, since it's very similar to the timeline. Visual similarity was less used than expected, even on collections where it was relevant.

2.13 PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps

This work presents PhotoMesa by B. B. Bederson [3], an application that supports browsing sets of images in a zoomable environment. It also supports clustering by metadata, not requiring previous work from the user. Users can choose directories of images and they are all displayed in a space-filling manner fig. 2.11. Images are displayed in groups, from the directories they origin from and users can smoothly zoom in to a group and then to a single image. It generates multiple-resolution thumbnails for maintaing a good performance. Keyboard keys are also used to navigate the canvas. The groups display their name and have different background colors for a better distinction. It also supports drag and drop of images to other applications. Text search is possible as well as selecting a group on a list.

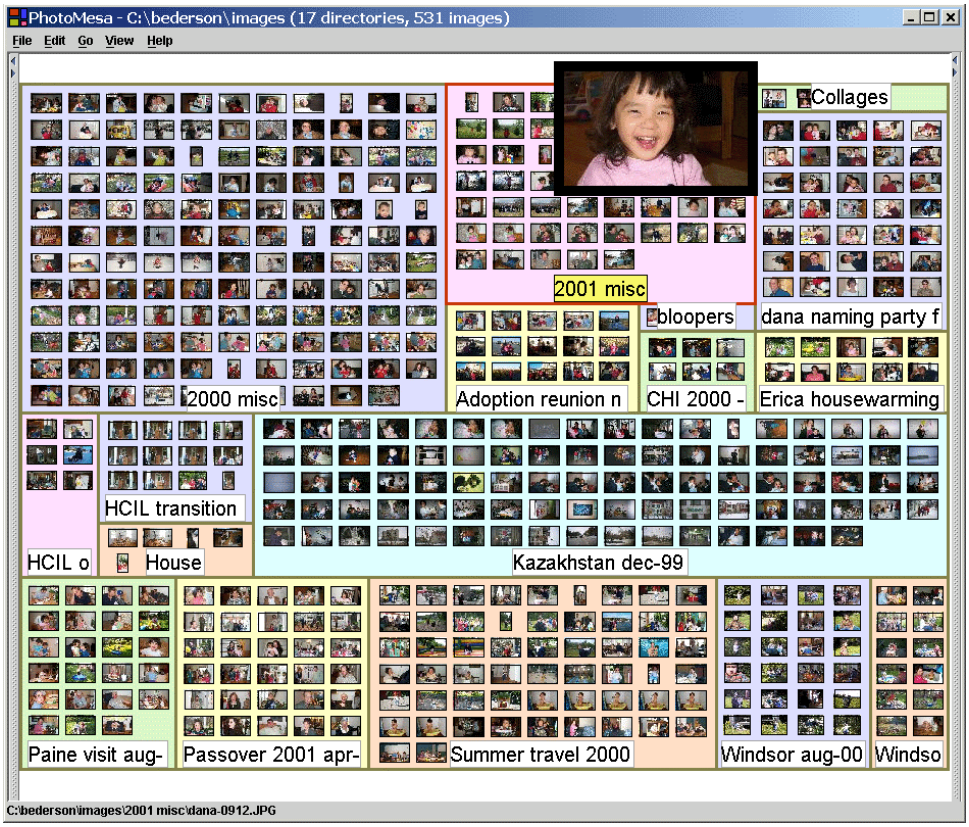


Figure 2.11: PhotoMesa with over 500 images in 17 groups.

PhotoMesa implements two algorithms for space filling, one is Quantum Treemaps by the author, a variation of the Ordered Treemap that is aware of the constraints imposed by the use of images, like grid alignment and common size, and Bubble Maps which is designed to have the least wasted space possible.

This work brings very interesting ideas to the table but it has a somewhat limited reach, with only referring to “over 500 images” on the application and by only taking simple metadata, like the path and modification date of images.

2.14 Discussion

Currently there are a lot of approaches to image organization and each has its own differences as demonstrated on Table 2.1.

Table 2.1: Different browsing methods

Work	Organization				Visualization	Focus	Size
	visual	date	gps	tags			
2.1 Qiu [47]	simple color measures	—	—	—	Grid	Simplicity and Efficiency	60,000
2.2 Rodden [50]	—	—	—	✓	Grid	Similarity usefulness	100
2.3 Porta [46]	—	—	—	—	Spot (and others)	Unconventional visualizations	400
2.4 Cooper [15]	—	—	—	—	Static and animated	Usefulness of animations	—
2.5 Strong [55]	color histograms and gradients	—	—	—	SOM	Evaluation of different features	2,200
2.6 Tan [58]	color histograms of subdivisions	—	—	—	—	Evaluation of different features	12,000
2.7 Naaman [42]	—	✓	✓	—	—	Organization based on events	?
2.8 Chen [11]	colors, edges and textures	—	—	—	—	Efficient fast-sparce clustering	10,000
2.9 Heesch [29]	six different features	—	—	✓	Radial	Complex similarity network	32,000
2.10 Hsu [33]	—	—	—	—	Grid with groups	Interaction on grupped photos	1,333
2.11 Schaefer [53]	color histograms of subdivisions	—	—	—	3D Sphere	3D mapping and interaction	4,500
2.12 Girgensohn [26]	—	✓	✓	✓	Overlapped graph	Having the best way to find photos	450
2.13 Bederson [3]	—	✓	—	✓	Q. Treemap / Bubblemap	Zoomable browser	500

Our survey revealed various methods for extracting the features of each image, from simple to complex.

One of the main problems is obtaining useful information from low level feature extraction of the image contents. Some try to get the most out of each image, with a variety of complex and time consuming procedures. Others try to focus on avoiding the complex computations by only getting simple but somewhat useful information. To contrast with this methods, Girgensohn's work [26] has found that users prefer other ways to filter the collection, like tags, dates and locations. It's still used, but probably isn't worth to spend much time on it with heavy processing.

Date and location are simple similarity measures and can be used to group the collections by events

and locations like Naaman did on this work [42]. Current mainstream software like Apple's iPhoto[1] and Google's Picasa[2] are already doing it in a semi-automatic way.

Textual metadata like tags and descriptions are also widely used both on our survey and possibly on all major mainstream software. The problem with tags and descriptions is that people usually don't assign them to the their photos but that's not a problem we're interested here.

The 3D Sphere [53] is also interesting but doesn't provide a better interface to the collection than expanding the sphere surface grid view to a full screen grid view, keeping the zoom function. The Phorigami work [33] introduces some interesting metaphors for manipulating groups of photos, although some clutter the view and could, therefore, be improved.

An interesting work is the Girgensohn's [26] visualization approach, where images can be organized by various features and can be filtered down, displaying matched photos alongside placeholders for photos that are only match partially. It has some problems like image overlap and capacity for showing large collections.

[1] http://www.apple.com/iphoto
[2] http://picasa.google.com

Chapter 3

Solution Requirements

Our survey was based on various types of previous work from the last ten years. Image browsers and technology have evolved a lot since then but there still isn't a definitive way for a user to look at his larger photo collection and understand its content and evolution.

We have the vision of a system that displays a large set of user's photos at the same time, in various arrangements, revealing patterns, differences and similarities between them.

We will now expose this vision by presenting the main goals we want to achieve with this thesis, as well as some of the guiding implementation requirements that we followed for a better end result.

3.1 Main Goals

The main goal of this thesis is to provide a different approach to the photo collection browsing methods.

More specifically, the work should:

- extract interesting information about the images — we want to be able to organize and classify images. This is explained in detail below, in section 3.3;
- provide an interaction with the full collection — we want to enable the users to easily view and interact with the entire collection at the same time, if they want to, instead of just limiting to a small window of images;
- efficiently display a large number of images in a single screen — this goal goes inline with the previous one: to be able to interact with the full collection, we should be able to display it all, at the same time, using techniques to make better use of available screen space;
- allow the manipulation of the display — we want the users to create different views that enable new perceptions of their collection, based on the extracted information.

With this capabilities, our work should be able to provide the user not only with a better understanding of the collection but also with an easy and interesting way to visually combine and view photographs.

3.2 Implementation Requirements

In addition to the referred main goals, we set our selves some design goals, or requirements, for our implementation. With them, we want our work to get closer to a real application, that real users can use and have some flexibility for it to evolve with time.

Therefore, we set the following requirements:

3.2.1 Ease of Use

Ease of use is one of the most important characteristics of any piece of software and can shape how well it will sell. Much more attention has been given to the user experience (UX) in the last few years. Systems that are easier to use systems, get the job done faster, are more enjoyable to use and allow less experienced users to use it.

Since this is such an important characteristic, we aimed at providing a simple interaction from the beginning to the end, while also providing a powerful system, even though it could be even more refined in a few areas.

3.2.2 Extensibility

The extensibility factor of a system is also important for the added value that can be obtained by quickly adding new features, either by the developers or by third parties. We made some parts of our work with this in mind, by allowing either external plugins or by generalization of code, allowing for future improvements with less trouble.

3.2.3 Performance

One of our main goals is to display a large set of images on the screen, but this brings problems since each image, in full-size, can take a good set of resources of the system. If we multiply this resources for a thousand images, we will not have a performent work.

Therefore, we set this requirement for having a performent work, that can be used with at least a few thousand images without taking down the system while using it.

What else?

3.3 Features of Images and Photographs

As referred before, we want our system to extract interesting information about the images and we will now explain this in detail.

Unlike textual data, images are a type of data where is not trivial to extract information from, in a computational environment. It's easy for us, common people, to understand what certain picture is showing. We can easily distinguish if there are, for instance, animals, people, flowers, buildings but it's

hard for a computer to do the same, and it's even harder to understand if that certain photo of a building and a person was taken because of the building, the person or both.

There have been various developments in the feature extraction front [38, 20, 52] and is currently possible to perform various detections in images with various levels of satisfaction.

Some examples of working solutions for feature detection:

- Face recognition — detection of the presence of faces in the images, their position and relations between them [61, 12, 57, 32];
- Object identification — identify objects in images [60];
- Identification of perceptive colors — what are the main colors that users perceive in certain image [56, 58];
- Image sequence — identification of images of the same scene that were taken in sequence, for grouping purposes [16];
- Image similarity — identification of similar images across the whole collection [61, 59, 64].

We want to use some of this methods to automatically extract information from the contents of images but, since this work is mostly focused on photography, there is another way to obtain really useful information that is EXIF Metadata.

EXIF Metadata is a standard format for metadata included in digitally captured media, like photographs captured by digital cameras. This devices save a lot of information on the image file, like the date and time of the capture, camera settings, camera orientation, location data and even detected faces[1]. This is textual data and, therefore, much easier to retrieve and analyze than the content based methods we discussed above and is a great addition to them.

Mixing all this different data, users should be able to interact, filter, sort and organize their collection in various ways, obtaining new and different perspectives of their images.

[1] the availability of some of this data requires capable camera hardware and software that are getting more common nowadays.

Chapter 4

Eagle Eye

Eagle Eye is the product of this thesis and is our implementation of the requirements previously detailed. It is a visualization tool that enables the display and manipulation of a large image set at once. It is focused at the regular computer user that has a few thousand digital photographs stored on the computer. It allows navigation, through panning and zooming of the canvas where the image collection is disposed, sorting in different ways while also allowing filtering, either textual or visual.

In this chapter, we will go through how does Eagle Eye work, what are its components and how they interact.

4.1 Overview

Our requirements stated the need for features to be extracted from images, so we started by creating a system to enable that. We then changed our focus to the visualization part of the work and redefined the general architecture of the work. The system is now composed of two main elements: the Backend, which is the system that extracts features from the images and prepares everything to be displayed; and the Visualization which performs the display from the images and their extracted features to the user.

This is the overview of the system that we will now detail in the following sections.

4.2 Backend

The backend is one of the two parts that make Eagle Eye. Its purpose is to extract information from images and set everything up for the Visualization.

Currently it is a command line utility that allows the user to enter paths for folders containing image files. The system will then read those images, gather their metadata, process them with the existing plugins to extract visual features and, finally, generate and output the multi-scale imagery (ainda não se falou disto) alongside with control metadata for the visualization.

We will now detail its architecture, and implemented feature extractors.

4.2.1 Architecture

The Backend comprises a library manager to hold the images, feature extractors to process those images and persistence to save all generated data.

Library Manager

The system displays images and, therefore, it needs to know what to show. This is where the library manager comes in. It creates a database which indexes existing JPEG image files stored on the user's computer and makes this information available for other modules to use. It is designed to be used with digital photographs which contain the aforementioned EXIF metadata like time, date, camera information, or location. This information is gathered upon import and is available throughout the Backend.

We explored a few ways to develop the image import process and we rested at the fastest we found. The user refers a folder to be imported and we use a third-party program, ExifTool[1], to crawl through the user-specified folders while identifying all the JPEG images and returning their Exchangeable image file format (EXIF) metadata which is then stored by our system (fig. 4.1).

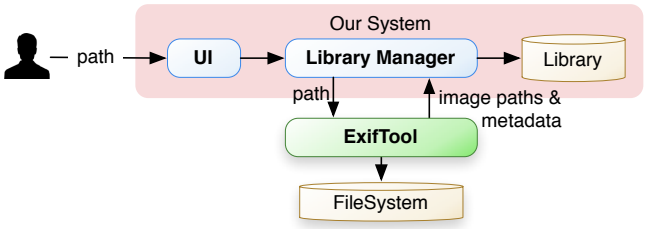


Figure 4.1: Process in use by our system of importing a folder using ExifTool.

Another option for importing metadata was to invoke ExifTool as part of a feature extraction plugin (FEP). Although that could fix a couple of problems with the current implementation, it doesn't make the metadata as ubiquitous as needed. Most FEPs rely on some EXIF plugins to work correctly and the current implementation doesn't easily allow inter-FEP data-sharing.

Feature Extraction

To enable the Visualization to arrange the images on the screen in different ways, they need to be classified. Some information is easy to obtain and compare, like when the photograph was taken. Other information needs to be extracted, like the number of people in the photo or what are the most relevant colors in the image. Fig. 4.2 is a short example of what feature extraction is all about.

We had a few ideas for extracting features from images and, to be possible to add more along the way, we developed a plugin system to ease the creation of other feature extractors in the future.

Each Feature Extraction Plugin is separated from the main system. They need to implement a common interface and are given the ability to access the image data from the library, and save the computed data back. They have freedom to access the image file or its EXIF information. They should, in

[1] ExifTool is a utility that allows easy read and write of file metadata. `http://www.sno.phy.queensu.ca/~phil/exiftool`

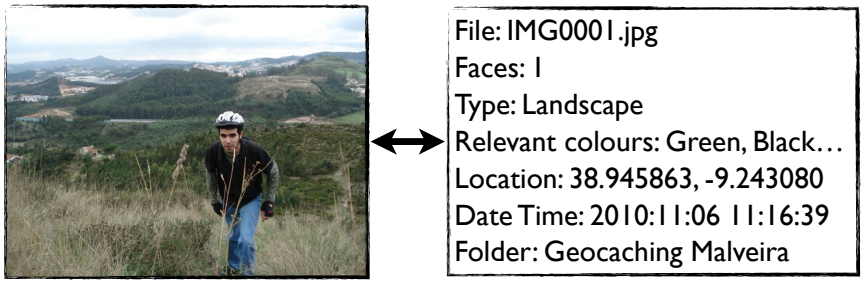


Figure 4.2: Example of information extracted from an image.

the end, store the processed data in a specified way so it gets exported to the visualization. Implemented plugins will be explained in section 4.2.2.

Persistence

Persistence of both library and plugin data are required so the system doesn't lose information that took time to generate. To ease the interaction with a database system, we created a database abstraction layer that hides the complexities of interacting with said system. It also allows us to change to another database system if we see the need for it. We chose Oracle's Berkeley DB[2] for its speed in retrieving data.

With this layer we can hide some optimization complexities like lazy-saving and lazy-loading. We use lazy-saving to save data to disk by chunks instead of doing it on each small update, speeding up the update process. Lazy-loading is not yet implemented but it is essential with libraries with tens of thousands of images, where keeping a complete library in memory is not feasible.

4.2.2 Feature Extraction Plugins

Back in the Solution Requirements chapter (3.3), we detailed some possible features that were interesting to be used in this work. In this section we will detail what extractors have been implemented.

Currently, we have four feature extraction plugins:

- Selection of useful image metadata
- Detection of image's main color
- Face detection
- Generation of multi-scale imagery

We now proceed to the explanation of each one of this plugins.

Selection of useful image metadata

This plugin acts as a filter for all the available EXIF tags. It picks the most relevant ones, codes them in a pre-defined way and appends them to the rest of the information to be exported for the visualization.

[2]Berkeley DB is a high-performace, embeddable, key-value, file-based database available at `http://www.oracle.com/technetwork/database/berkeleydb`

Information when the photo was captured, what device was used, the path where it resides or information about the location where the photo was taken are a good example of the most commonly relevant tags TODO: Ask users what other tags are relevant for them. In the future, this set of extracted tags could be optionally set by the user.

Detection of image's main color

Sometimes people don't recall where or when a photo was taken, or where is it, but they vaguely remember that the photo had some dominant colors, like the red of a parrot in a green background (fig. 4.3). This kind of information can be helpful when searching for a photo in a collection.

Figure 4.3: A colorful photo.

Color extraction from images has been a long standing problem [62, 55, 25, 27, 59, 19, 9]. We wanted to extract the most perceptible colors from images so, in the parrot example (fig. 4.3), the system should associate the image with the colors red, green, white and black and avoid colors that have little relevance, like the small blue of the feathers and also ignore small tonal variances in the reds and greens.

For that we explored two methods that reduce the number of colors in an image to the most essential ones, the first being an adaptive method, selecting a few averaged colors from the image and the second using a palette as reference to select the colors.

A common JPEG photo can contain 16.8 million different colors. Our adaptive method reduces the possible colors to less than ten[3]. The obtained colors are chosen by averaging the colors in the image, so if there's a lot of blue tones, a single, averaged blue will be replacing those tones (see the middle image of fig. 4.4). In areas that have little tonal variation of the color, the resulting color will be very similar to the original (like in the right image of fig. 4.4). But sometimes this method fails to save important colors when they occupy a relatively small area of the image or if the image doesn't have a strong contrasts (shown by the left image in fig. 4.4).

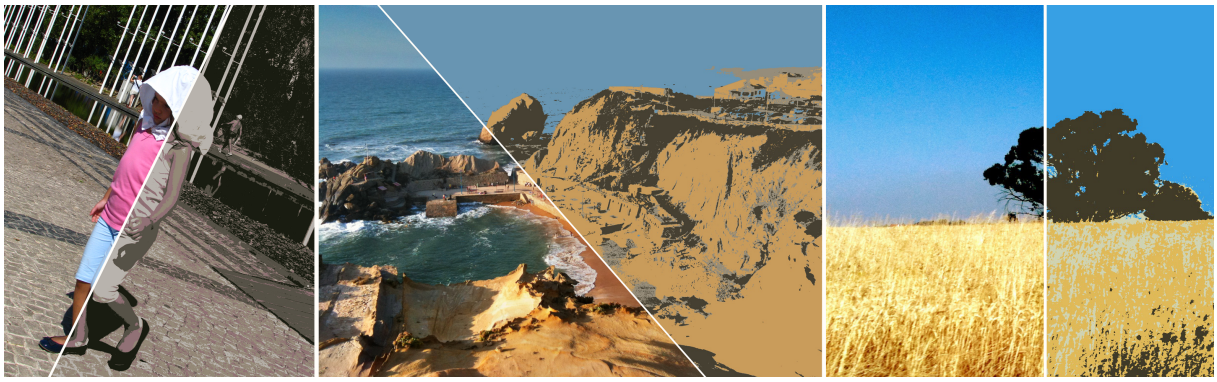


Figure 4.4: Three images where the left half is the original version and the right half is a version with very limited number of colors provided by the adaptive method.

The second method uses a predefined color palette (fig. 4.5) which is an adapted mix between the

[3]We tried with multiple options from one color to ten colors and the results vary from image to image

eleven most recognizable colors (red, yellow, green, blue, purple, brown, orange, pink, black, white and grey)[4] and the 21 ColorAdd colors[5]. This mix has a great range of colors and each one can be easily named with recognizable words[6] which then could be assigned to the images. The images processed with this method get their colors changed to the most similar ones present in our palette. This assures that colors in a relatively small area or images with low contrast still get the deserved attention (fig. 4.6.3), unlike the adaptive method (fig. 4.6.2). This method doesn't reproduce the colors so well as the adaptive but, since we want to obtain generic colors, this one is more reliable.

Figure 4.5: Color palette in use for restricting the possible colors in an image.

4.6.1: Original image

4.6.2: Adaptive method

4.6.3: Mapping method

Figure 4.6: Comparison between the original image, the adaptive color reduction method failing to keep the pink and the blue colors and the color mapping method.

This was the research we made to extract perceptible colors from images. We generated the above demonstration images using ImageMagick[7] and its "colors" feature for the adaptive method and the "remap" for the palette method. Unfortunately, time was not on our side and we had to move to a simpler system.

The current color extractor plugin does a simpler job of calculating the median color of the image and use the hue to index images.

We took some ideas from the work of Qiu et al[47] of exploring a method of image indexing that is fast and simple. On this work, images are sorted into "bins" according the their average color. This bins are divisions of color planes, indexed using binary trees allowing for very fast searches.

We used the open source library AForge.Imaging[8] to obtain histograms for the images and compute

[4] Qiu [47] explains this are the most universal recognizable colors in any language.

[5] ColorAdd is a 21 color catalog with symbols for each color designed specifically for color blind people. We used the colors as a reference, specially the light and dark variations.

[6] We want to avoid names like "Amaranth" or "Munsell" that most people don't know about. A large list of this names can be found on `http://en.wikipedia.org/wiki/List_of_colors`.

[7] ImageMagick is a software suite to create, edit, compose, or convert bitmap images. `http://www.imagemagick.org`

[8] AForge is a framework designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence `http://www.aforgenet.com/framework/`

their average color in RGB[9] and HSL[10] values. Both those values are then stored on the plugin-extracted data for each image. The images will be distributed into bins on the Visualization. We could assign bins at this point but, by not doing so, we are giving freedom to the users to change the number of bins used for display. We will discuss this on the Visualization sections ahead.

Although we didn't implement this plugin the way we desired, it demonstrates that color extraction is feasible and, with more time, a better method that does more than just averaging the colors, including detecting the most relevant ones, could be implemented.

TODO: show of differences between having the image reduced or not. something I never tested :P

Face Detection

The face detection plugin is based on the open-source OpenCV library[11] which processes every image file and detects existing faces (fig. 4.7).

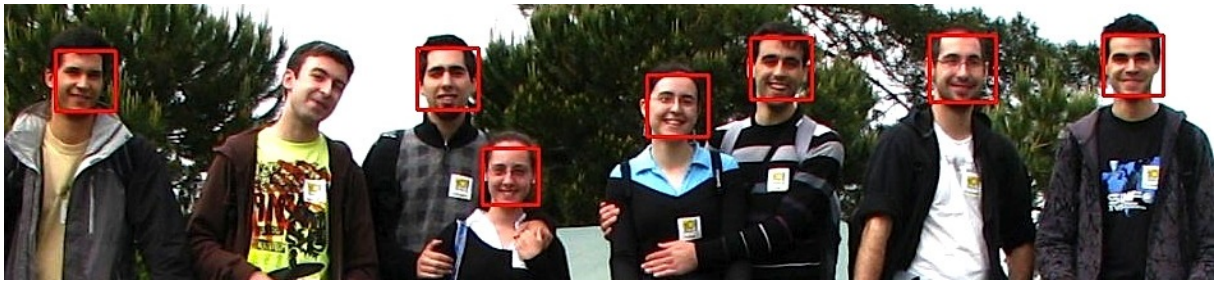


Figure 4.7: Example of the OpenCV library detecting faces on a common photo.

No face recognition software is perfect. Usually if the software can detect every face, it will probably detect some other things in images that aren't faces (false positives). If it's successful in only detecting faces, it will probably miss some other faces that aren't ideally positioned (false negatives). OpenCV is included in the latter, only detecting faces but also missing some that are tilted (like in fig. 4.7) or turned on the side.

This process is quite computationally expensive and therefore we resize all the images down to a more acceptable size, making the process more than five times faster.

We tested 29 images, from six different cameras, ranging from one to ten megapixels, and containing up to thirteen faces. The test consisted in running face detection on each image, in its original size and in various resolutions from 2000 to 200 pixels on its longer side, comparing the number of faces recognised and the time needed to process them. The results can be seen in fig. 4.8.

The purpose of this test is to identify how much can we reduce the images while maintaining a high recognition rate, we are comparing the recognition results of the downscaled versions to the original size and analyze the speedups and failures in recognition. We do include the number of faces actually present in the photos for comparison, corresponding to the 100% value in fig. 4.8.

[9] Color model composed by red, green and blue `http://en.wikipedia.org/wiki/RGB`

[10] Color model composed of hue, saturation and luminance `http://en.wikipedia.org/wiki/HSL_and_HSV`

[11] OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision. Available at `http://opencv.willowgarage.com`

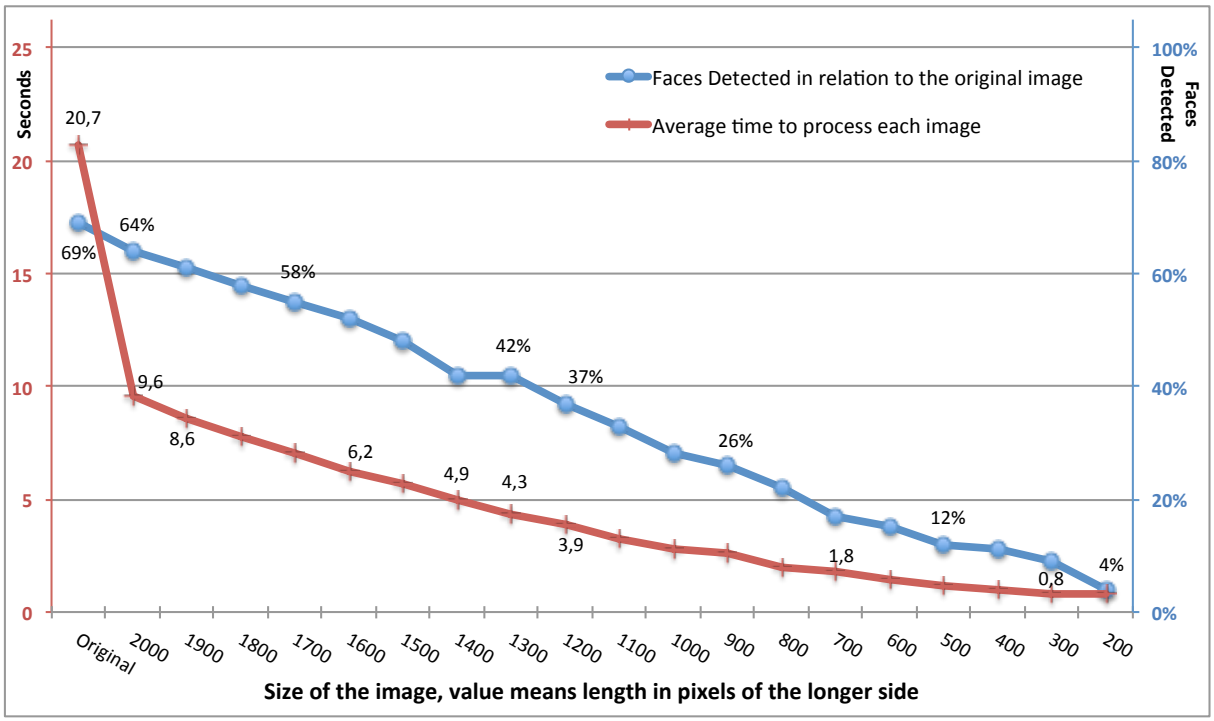


Figure 4.8: Results of the face detection test. 100% of faces is the actual faces present in the test images.

We can see that by only reducing the images to 2000 pixels on the longer side, the processing time fell to less than half (20.7 to 9.6) without much loss in recognition (69% to 64%). The 1300 pixels was the chosen value for being the last with more than 40% recognition rate (60% of the full size image) and being 4.8 times faster[12] than using the original image. In the future, with more tests, we can fine tune the resizing algorithm to get better results.

Generation of multi-scale imagery

This plugin generates all the data files needed for the visualization to work. As referred previously, the visualization relies on the DeepZoom technology and it needs to process the images before they can be displayed. This plugin does exactly that.

Using a library from Microsoft, the plugin generates, for each image, a metadata file and a set of image files representing the original one at multiple scales, from a single pixel to a large, detailed image.

After passing through all images, the collection as a whole is subject of additional computation, this time generating imagery for all images as a single set and a metadata file that agglomerates all image sets used. This metadata file for the collection (called collection.xml) is then altered by the plugin to attach to each image, the data previously generated by the other plugins.

[12] 4.3 seconds per image versus 20.7; one hour and 10 minutes per thousand images versus almost six hours

4.3 Visualization

The greatest challenge of this work was the creation of the visualization part for its requirements. We will now describe this part of Eagle Eye, its architecture, visualization techniques, sorting and filtering capabilities.

4.3.1 Overview

We wanted to make the visualization simple and easy to use, while keeping it flexible and capable enough to allow for an enjoyable and relevant experience.

After the backend has finished the all the processing that is needed, the visualization can be opened and all images that were added to the backend's processing list will appear in what we call "the canvas". After loading the metadata, the user is also presented with a set of options on the top toolbar, which contains all the controls needed to use the system (fig. 4.9).

Figure 4.9: Example of the visualization of 5679 images, organized by capture date, in Eagle Eye.

The Canvas

The canvas is the most relevant part of the visualization as it displays the user's images.

According to A. Torralba [60], an image with 32 pixels per side is the minimum size that allows a user to recognize what it is. His work was about an image library that had nothing to do with the user's photo. On Eagle Eye, we can have even smaller images because our focus is the user's own library. This allows much easier recognition of images since, by looking at the form and colors, the user recalls

what is that image without having to “read” and understand a totally new image. Another very important help is the grouping of events (e.g. by date or path) which allows the user to extrapolate the contents of the whole group from looking at a few of them instead of trying to understand each image individually. This allows much faster recognition of groups with images at really small sizes, even if not all images can be understood.

An example of this can be seen on figure 4.9, which is a screenshot of the application running on a 13.3” screen with a resolution of 1280x800 pixels, where the larger side of each image uses 15 pixels and takes 3 millimeters. On this example, different groups are clearly distinguishable from each other and most of them are easily recognizable by the owner of the photographs. Each group seems to have a specific set of colors, e.g., blue, green, yellow, orange, gray or black, making easy for the owner to distinguish and recognize the photos.

A photo application that only displays all images isn’t very useful, therefore, manipulation of the canvas is allowed by zooming and panning. This means the user can, at any time, use the mouse to drag the canvas around or, by clicking or scrolling, zoom in and out of the canvas. Zooming goes from the default view of thousands of images at the same time, to the full screen view of one of them, and everything in between in a smooth way. This allows a closer look to any group or image, providing an easy and fluid way to navigate the collection.

The toolbar

Figure 4.10: The toolbar on top of the visualization UI.

The user can then use the functions on the toolbar to filter and sort differently. The toolbar is divided in three sections: History, Display and Filtering (fig. 4.10).

The History section contains some basic functions that work similarly to the current web browsers. There are buttons for going back and forward between the history of display states and a save button for bookmarking the current display state, allowing the user to easily get back to it later.

The Display section, in the middle contains two options to change the image display: the sorting options and the display overlays button. The former presents the available sorting options for the current collection, based on the available metadata and on the best ways to display them. The content is presented according to the selected option and changing to another causes the images on display to move around to a new position and form a different display order. This sorting and disposition options will be explained in a later section. The other button in this section of the toolbar is called “Group Names” and enables or disables a layer of information on top of the images. This layer distinguishes the groups of images in display by painting them with different colors and presenting names for them, depending on the selected sort option (fig. 4.11 shows an example of this). Grouping will also be explained bellow.

The third and final section of the toolbar is the filter section. It contains controls to filter images by using simple text and to visually select images on the canvas. This options will also be explained bellow.

Figure 4.11: By enabling overlays, the user can see names for the groups and a much more clear group distintion.

4.3.2 Disposition of Images on Canvas

The different ways to dispose the images on the canvas was a matter that required some exploration of possibilities, like the ones seen on the Related Work ([3, 6, 11, 26, 29, 33, 46, 50, 53, 55]).

We have come up with a some options for photo grouping, sorting and filtering. We kept a sensible set of the choices so its easy and understandable for users. To make sense of how all this pieces work together, we are going to explain how do we sort images into groups and the different ways the user can arrange those groups on the canvas, followed by how filtering is done.

Sorting images into groups

As we've seen previously, the backend outputs metadata for each image. This set of metadata is loaded into the visualization application of Eagle Eye and is then indexed by type.

For instance, each image has an associated creation timestamp which will be aggregated by days, generating an image group for each day.

Similarly, the mean color associated with each image is indexed and groups are generated by splitting the hue spectrum. We used a quick approach and divided it into 12 equal bins with names, as can be seen in figure 4.12. From the image, it's possible to see that some colors have a greater usage of the spectrum, like green, which takes the most part of three bins, while other colors use only one. This gives the user multiple tones for some colors and only one for others. We made it this way so the number of bins could be quickly changed to something else if we saw fit but the most sensible option might have been to manually define the bins with different sizes and more accurate names.

Another option is the grouping by device name, which usually allows to distinguish between who took the pictures if, for instance, different people have different cameras on the same event.

The last option currently available to the user is grouping by path which groups together the images

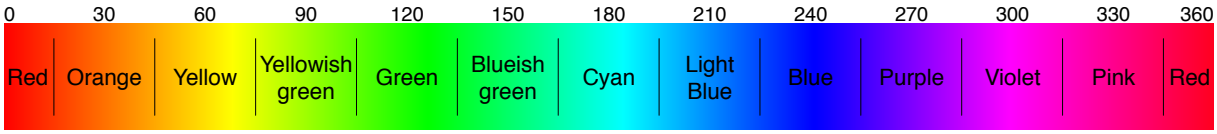


Figure 4.12: Division of the hue values in 12 equal parts, 30 points apart from each other.

that were already grouped by the user, on the file system. This allows for the display of an organization that is recognizable to the user, which can make a good starting point.

On the canvas, group boundaries are identifiable by discrete gray borders and, when the Show Overlays function is active, by color rectangles that also contain the groups' names.

Different dispositions

After having the images grouped by any of the sorting options referred on the previous section, the system has to know how to display them on the canvas.

We looked into various options ([3, 6, 11, 26, 29, 33, 46, 50, 53, 55]) and picked the concepts that we thought that made sense and that would be easier for the user to understand. We chose a treemap view and a column-based linear view. Both of these are grid-based layouts, meaning that images are positioned inside a defined grid on the canvas. We also looked into free positioning systems ([26, 50, 53, 55]) but they make it harder for the user to understand the images within, since some images will be covered by others. When displaying thousands of images, it's important to make them easy to see, and mixing them up wasn't the appropriate thing to do. We focused on other ways for making it easier to the user see what matters and we will talk about them later on.

The first layout technique we employed was the treemap. Introduced by Brian Johnson and Ben Shneiderman [35] in 1991, treemaps recursively subdivide areas into rectangles. The problem is that areas can take many forms, from squares or large rectangles to almost thin lines, like can be seen on fig. 4.13. Applying treemaps to images calls for the adaptation of the algorithms to make sure the areas can correctly hold the images and that all images in all groups have the same size and are positioned in the same grid, to make them easier to view. This ideas are supported by B. Bederson on his work with the Quantum Treemaps [3]. We tried to apply quantum treemap as our treemap algorithm but, due to its complexity and other recurring problems, we had to move to a simpler method and adapted the squarified treemaps [6] by Mark Bruls to support the grid of images that we required. This algorithm was much easier to understand, adapt and implement.

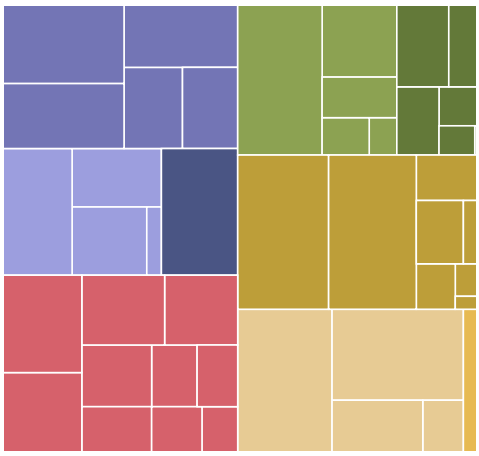
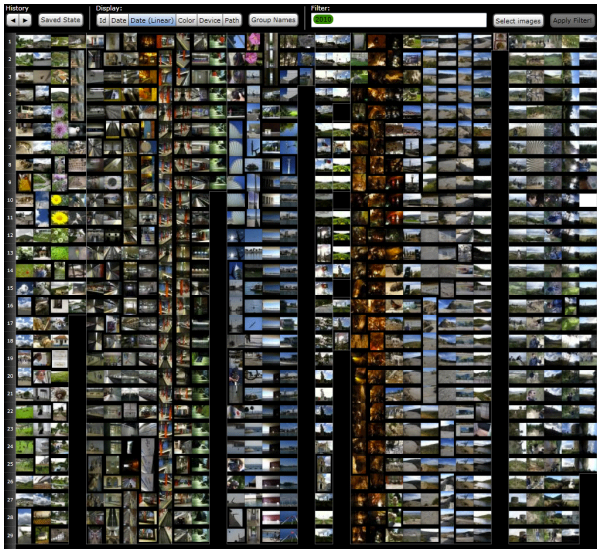


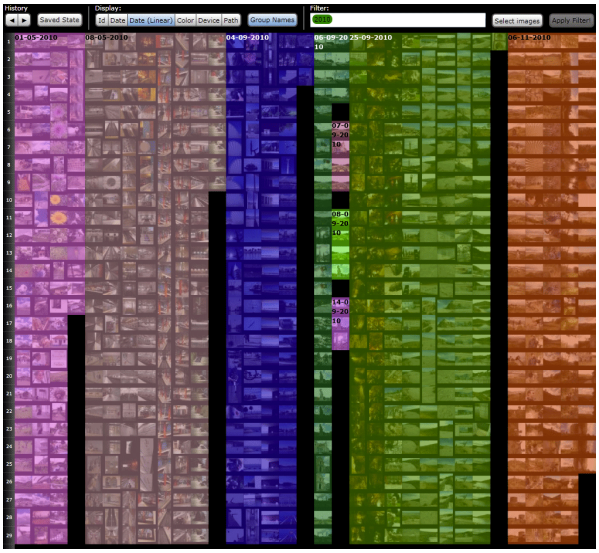
Figure 4.13: Common treemap based on space filling.

Our resulting treemap algorithm displays larger groups first, leaving the smaller ones to the end, making an effort to layout groups as rectangles with an aspect ratio as close as possible to the screen's aspect ratio, for filling the screen when the user zooms on one. There's also an effort to fill the space

left between larger groups and the edges of the screen, making the display more compact and with less empty space across the screen edges.



4.14.1: Images only.



4.14.2: With group divisions overlayed.

Figure 4.14: Example of the linear display.

One problem of the squarified treemap display is that group sequence is irrelevant. Groups are positioned by their size, which is unacceptable for sorting options that require some sequence, like sorting by time. For this we created a linear display (fig. 4.14) that uses columns and displays groups sequentially. Each group may fill part of a column or various columns, depending on their size. With the aim of reducing wasted space, groups that fit on the wasted space left by the previous group use that space to display themselves. This is useful to group the few pictures the users may capture of their day-to-day, between days (like weekends) that he went on a trip and took a much larger number of photographs.

Currently we are using this layout system only for the date display since the use of columns makes visualization harder, requiring either some panning from top to bottom or selecting the group using the filter tools explained ahead. This is an area we must improve, and we will discuss some ideas later on.

Filtering

In addition to the sorting options explained above, Eagle Eye also provides the user with the ability to filter images. This allows the users to focus on specific groups of pictures that are of interest at a particular moment.

For instance, the user might want to only see photos of a day, or a person, or person in a specific day or even pictures that are mostly blue.

For all this, Eagle Eye provides two ways to specify this kinds of constraints: using the filter bar or selecting pictures on the canvas, and we will now look into both them.

The filter bar's purpose mimics a regular search box, similar to the ones that exist on applications like internet browsers, file browsers or photo browsers: it provides a way for the user to type what he

is looking for and get some suggestions to help him with the search. Since we wanted a simple UI, we figured that we had to provide smart suggestions to the user, so that he can feel comfortable using it for multiple types of searches. Searches are performed immediately upon selecting the desired filter. Currently, searches are only intersections (commonly “and”) of the entries in the filter bar but after adding adequate UI, it will be possible to also use unions (commonly “or”).

Every piece of the metadata can be used as a filter and the suggestions list shows what are the available options and what type of metadata they relate to, for instance, when looking for a person’s name, the “keyword” metadata type is shown, when searching for a place, both “keyword” and “path” types might be presented, if the path to the pictures included the place where they were taken.

This examples are quite simple but we wanted to go further and allow searches like “Summer”, for all photos taken around the summer time in all years, “May 2010”, for all pictures taken during that month, “Has 2 people” for photos that have been detected to have people in them. Although some of these examples have not been fully implemented yet, they are in the plans. See figure 4.15 for some examples.

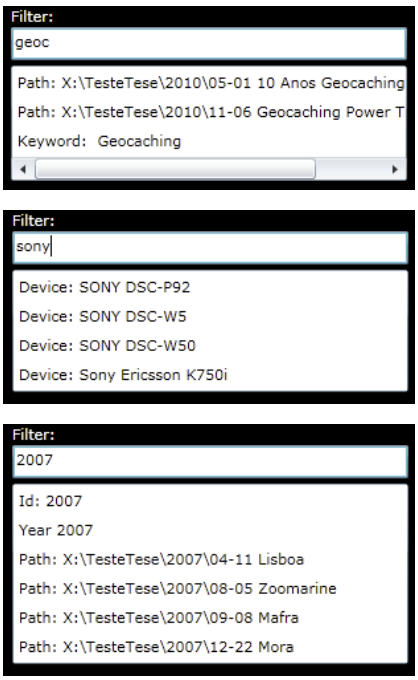


Figure 4.15: Examples of filter suggestions.

The second filter method allows the manual selection of images from the canvas, either by a common mouse drag-and-drop on the canvas or by selecting entire groups with a mouse click. This two interaction options can be activated on the respective button on the end of the toolbar.

As explained above, the filter bar intersects its elements immediately after being added but that behavior in the function would make it impossible for the user to make multiple selections, therefore they are only unified and the filtering will only be performed when the user presses the Apply Filter.

Since the selections are displayed in the same way text filters are, the removal of a selection is made in the same way.

4.3.3 Architecture

We will now take a look into the architecture of the visualization part of Eagle Eye.

Metadata indexing

At the center of the visualization is the DeepZoom canvas. By default, it displays images in the order they are specified on the “collection.xml” file, the file generated by the backend that identifies the multi-resolution imagery.

The “collection.xml” also contains some metadata for each image, added by the backend. The visualization parses the file and extracts this metadata which is then processed and indexed.

The backend encodes textual key/value pairs of metadata in JSON for each image and the visualization decodes and aggregates them on various indexers, one indexer for each type of metadata. Eagle Eye has a few indexers that know how to read and process different types of data:

- a generic indexer, for basic text or number values
- a date indexer, that gathers images by day
- a color indexer that gathers images by color
- a keyword indexer that indexes images by each one of the associated keywords
- a path indexer that indexes images by the folder where they resided when added

The indexers are managed by a metadata collection manager that handles, for instance, the XML parsing and creation, management and retrieval of indexers.

The indexers are the base for the sorting and disposition techniques we've covered on 4.3.2. The indexers can have disposition types associated with them, like the date indexer, which has the treemap and the column dispositions associated with it. Other indexers only have the treemap disposition (e.g. Color, Path) and others don't have dispositions at all, like the Keyword indexer that is only accessible from the Filter bar. This indexer to disposition association is our decision and aims at providing the user a simpler interface while displaying images in the best available way.

The indexers also provide data for the filter bar's suggestions to provide the suggestions we have already seen on section 4.3.2.

Canvas

The canvas is the most visible part of this work. Its core is from the DeepZoom framework and we use it to display lots of images in certain positions, according to the state of many options and variables. To hold all those options together and to help developing some more, we have created a state manager for the canvas.

A new canvas state object is created for each different set of options for the canvas. The object stores the options and computes the positioning of all images by picking those that have been selected by the filter, get the groups for that image set from the selected indexer, dispose them using the selected disposition algorithm and finally computing and saving the position for each image.

We use memoization to avoid repeated calculations when changing display types. If a certain display has already been calculated, it will reuse the information on the saved state object to rearrange the canvas without going through the position calculations again, allowing a faster interaction.

The navigation buttons on the toolbar interact directly with the state, for instance displaying a previous state or one that was saved.

Reducing Clutter

This work displays a great quantity of images at the same time and we thought that we could help the user by reducing some unnecessary clutter.

Sometimes users take more than one photo of the same thing and they do it for different reasons, some of them simple and others more technical. Some people just didn't like the previous photo or they want to make sure they have a few options to pick the best one and so they take more. Bursts of photographs are a similar case of this behavior but resulting in a larger image count.

Some more advanced photographic techniques require that the user takes more than one shot of something for merging in post processing. Examples of this techniques are panoramas, where a sequence of shots are merged to create a bigger image that what the camera lens is capable of, exposure bracketing, to overcome partial under or overexposure or focus bracketing, to increase the are of the image that is correctly focused.

Various shots of the same subject are generated by all of this motives and they clutter any image browser since they end up being multiple copies the same thing.

To help the user abstract himself from this similar images, we decided to come up with a way to collect them. Hsu et al. [33] used the idea of image grouping but probably not with a great implementation since it can be hard to see what is in the groups when they are collapsed and grouping was done manually. We chose a different path.

For our implementation, we decided to group images automatically based on the proximity of their capture timestamps, so every two images that were taken in a time span less then t seconds are grouped together. After some experiments we decided on $t = 4$ seconds which is more than enough for sequences of images that were taken automatically, gives some margin for manually taking another photo of the same thing, but is short enough for a regular user to take a picture, look at something else move the camera and take a picture of that.

For the display we decided to show all images of the group in the space of one, but displaying the first image at regular size and reducing the others to a smaller size, perceptible when looking closer and keeping the ability to zoom in on each of them and see them in fullscreen. This way we can reduce the clutter without fully hiding repeated images.

Obviously that this can have both false positives or false negatives but our tests revealed this would be on a very small number. UI to fix that could be added but we don't think that it is sufficiently important to clutter the interface.

Figure 4.16 is a selection of success cases for this feature. It reduced a set of 80 images to just 10. It's possible to see cases where multiple pictures were taken in sequence and cases where the aim was to make a panorama. Portrait and landscape images are both equally well handled. The main image has the same size as it would be without the grouping, but is aligned to the top left corner, maximizing the free space to the left or to the bottom of the grid. This space is then filled with the other images which are adjusted in ways that can maximize their size.

Figure 4.17 is a real example of a trip to the beach, where some photos were taken normally, some were taken using the HDR or the Panorama techniques and some others where taken as a quick sequence. All this sets of photos are correctly grouped together. This figure contains a total of 71 photos, taking the space of only 36. On the bottom left of the figure, can be see two sets of sequential photos of the same scenes that were not stacked together due to the longer period of time between their captures.

Figure 4.16: Selected examples of photo stacking as presented by Eagle Eye.

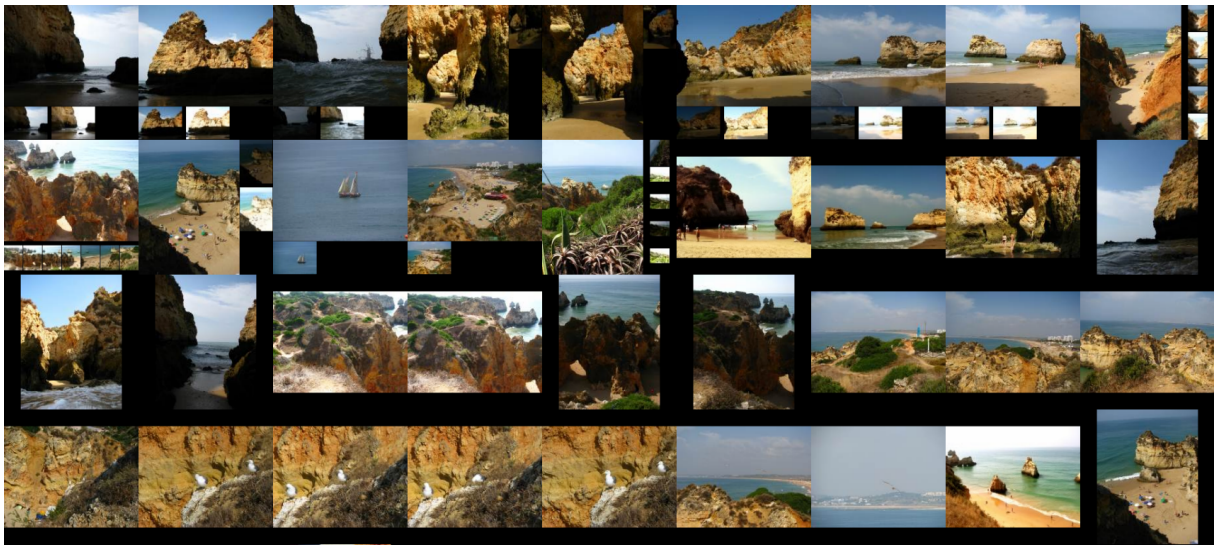


Figure 4.17: Example of a trip where HDR and Panorama techniques where applied for some photos. Other shots were repeated, some quickly and others not so much.

To correctly group this photos in an automatic way, their visual similarity must be determined to avoid grouping unrelated photos. This is one of the features already referred on the Solution Requirements chapter (3.3).

To conclude this chapter, where we've seen what this work does and how it does it, we would like to add that many features and improvements could be added for an even better experience, but time and the focus of this work didn't allow us to do it. Even so, the section on Future Work (6.1) contains many ideas we had which could bring this work closer to a real world application.

Chapter 5

Evaluation

CRUZ

- * só usa ”programas básicos”
- biblioteca com 833 fotos
- Foi demonstrada a interface. . .
- * rapidamente identificou padrões de cores
- só usou uma câmara. organização por device useless
- * acha giro/gosta bastante
- não usa keywords
- * algumas sugestões de melhorias
- tem fotos semelhantes mas que passam o tempo das stacks (não faz bursts nem braketing)
- * nunca teve a experiencia de mexer com tantas fotos ao mesmo tempo
- * consegue distinguir eventos
- ecra de 22” 1600x1080
- * identifica a falta de slideshow e rápido fullscreen de fotos
- pedi todas as fotos que ele tirou no interrail à noite. . . ele não tinha fotos nenhumas. . .
- depois de algumas experiencias, escolheu as fotos escuras , ordenou por pasta e verificou que não tinha fotos à noite
- pedi fotos onde eu e ele aparecemos. procurou em photowalks onde fomos, escolheu grupos na organização por pasta e seleccionou imagens onde pareceu ter pessoas
- queixou-se de não conseguir navegar quando está nas selecções de fotos
- sugeriu usar o right click para seleccionar e left click para navegar
- mudou-se para a lib de teste com fotos minhas
- pedí para ver as fotos mais antigas, ordenou por data (linear) e seleccionou uma data de fotos no principio do canvas
- pedi quantas vezes é que eu fui tirar fotos ao parque das nações
- procurou tags com naçoes. . . queixou-se que não tinha o OR

- outra hipotese era ir pelo path e escolher grupos
- meteu por data e viu
- muito preto
- qual é o evento onde tenho mais fotos escuras...
- procure pro path
- pede um botão de reset dos filtros
- com que camara tiro fotos mais escuras
- cor $\rightarrow$ device

D4rch

- 2000 fotos
- gostou da interface e da interação
- pedi fotos tiradas fora do país
- pediu pelo OR
- path $\rightarrow$ seleccão de grupos
- parece que seleccionar images e aplicar um filto n funca
- pedi que fotos não eram dele
- Devices $\rightarrow$ grupos
- botao limpar filtros
- pedi para encontrar uma foto fixe, verde... como se tivesse que a por na parede
- procurou por cor e escolheu uma
- dá jeito para ver muita coisa
- diz que se percebe bem as fotos e identificou alguns conjuntos de fotos no seu tamanho reduzido
- –
- A aplicação demora um pouco a carregar nas fotos,
- Pedi para encontrar uma foto de um golfinho, foi pelas fotos azuis e rapidamente encontrou uma foto de um golfinho a saltar
- Pedi para descobrir e mostrar quantas vezes fui tirar fotos ao parque das nações
- Ordenou por path e viu os nomes
- Pedi para procurar as primeiras fotos que tirei
- –
- bugs nos filtros
- or
- mais rápido
- considera que a disposição das imagens é mais ou menos decente.
- as stacks não são explicitas, não se percebe porque estão juntas
- não dá pra seleccionar uma imagem dentro de uma stack
- concorda com a falta de um slideshow

- gosta das cores do group names mas questiona se pode acontecer alguém confundir-se pois as a organização por cores tem as cores mas as outras não
- pedi para tentar identificar fotos em zoomout e identifica algumas, embora já conheça algumas. teve problemas a perceber fotos mais cinzentonas.

Chapter 6

Conclusions

We will now go through what we think could be improved and conclude this work.

6.1 Future Work

During the realization of this work, many ideas popped in our minds but we didn't had the time to work on them.

6.1.1 Consolidation

The most important effort that must be done to this work is a rearrangement of the backend and its connection to the visualization.

As we've seen on chapter 4.2, both parts of this work are currently separated, with the backend being a command line utility. Our idea was to create an graphical application where the user could enter the folder paths he wants to add to Eagle Eye and then the application would do the needed work on the background, using free resources, and when it was done, the user could enter in the visualization mode without the need to open the internet browser.

This interface would also provide some configuration options, for the system and its plugins.

6.1.2 Feature Extraction Plugins

The second important thing to do is to improve the feature extraction plugins.

The color plugin, for instance, should be able to better identify colors in images instead of simply make a color average.

More data should also be extracted from the EXIF. As example, location data should be translated into place names[1]. Another plugin we see some benefit in creating would be a generator of keywords related to the capture dates. Currently we have the visualization reading the capture dates and generate keywords like "Summer" or "May 2011" for use as search tags. By transferring this to a feature extraction

[1] The action of turning GPS coordinates into addresses is called reverse geocoding. An example of a provider of such service: `http://code.google.com/apis/maps/documentation/geocoding/index.html#ReverseGeocoding`

plugin, this data will be generated only once. It would then appear in the Suggestions of the Filter Bar as date options, allowing the gathering of sets of images from various years or seasons.

An additional plugin that would be interesting to develop would be some kind of hook into other application's metadata about photos. As an example, it's possible to access Picasa's face recognition and identification data about photos. By bringing this data into Eagle Eye, we could allow the user to perform searches for people's names without having to add those names as keywords.

6.1.3 Visualization

On the visualization part, the linear disposition, used mainly by the date view, should be improved to make more clear the distinction between days, months, years. A great idea that would be interesting to use would be the inclusion of something like the date bar in the work of Girgensohn et al. [26] (fig. 2.10). Another idea for a different disposition born from the stacking of similar photos discussed in section 4.3.3, where one image is bigger than the rest, to help identify what is on what group.

We also think it needs to be more slideshow friendly. Slideshows are important part of any photo application, be it on the desktop or even on the web, since people always like to view their images. This work already displays images in fullscreen, it just needs a better interaction to repeatedly move to the next image, something that can be achieved easily with a button on the UI and bindings to keyboard keys.

6.2 Conclusion

In this work we have explored the current state of image browsers, their goods and bads. We have then defined multiple requirements for the work to be done and created an implementation of them.

Eagle Eye is . . . something something something. . .

We tested it and found the users generally liked the new way to view their images and found how interesting it is to identify such small images

It still has space for some improvements but its close to a point where it could be put on the market.

conclusions

old: In our view, with this additions, Eagle Eye would be a useful photo viewer that could be used by anyone that has some computer knowledge and a photo collection spread on the computer.

Bibliography

[1] ALBUZ, E., KOCALAR, E., AND KHOKHAR, A. A. Scalable Color Image Indexing And Retrieval Using Vector Wavelets.

[2] AZZAM, I., LEUNG, C., AND HORWOOD, J. A fuzzy expert system for concept-based image indexing and retrieval.

[3] BEDERSON, B. B. PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th annual ACM symposium on User interface software and technology* (2001).

[4] BERMAN, A., AND SHAPIRO, L. A Flexible Image Database System for Content-Based Retrieval. *Computer Vision and Image Understanding* (1999).

[5] BRUIJN, O., AND SPENCE, R. Rapid serial visual presentation: a space-time trade-off in information presentation. *AVI '00: Proceedings of the working conference on Advanced visual interfaces* (2000).

[6] BRULS, M., HUIZING, K., AND WIJK, J. J. V. Squarified Treemaps. *Proceedings of the joint Eurographics and IEEE TCVG Symposium on Visualization* (2000).

[7] CAMPBELL, I., AND VAN RIJSBERGEN, C. The ostensive model of developing information-needs. *Citeseer* (2000).

[8] CHANG, S., YAN, C., AND DIMITROFF, D. An intelligent image database system. *IEEE Transactions on ...* (1988).

[9] CHANG, Y.-C., AND LEE, D.-J. Color Image Quantization Using Color Variation Measure. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing* (2007).

[10] CHEN, J.-Y., AND BOUMAN, C. A. Hierarchical browsing and search of large image databases. *Image Processing* (2002).

[11] CHEN, J.-Y., BOUMAN, C. A., AND DALTON, J. C. Similarity pyramids for browsing and organization of large image databases. *Human Vision and Electronic Imaging III* (1998).

[12] CHEN, L., HU, B., ZHANG, L., AND LI, M. Face annotation for family photo album management. *International Journal of Image . . .* (2003).

[13] CHRISTMANN, O., AND CARBONELL, N. Browsing through 3D representations of unstructured picture collections: an empirical study. *Proceedings of the working conference on Advanced visual interfaces* (2006).

[14] COMBS, T., AND BEDERSON, B. Does zooming improve image browsing? *. . . of the fourth ACM conference on . . .* (1999).

[15] COOPER, K., BRUIJN, O., SPENCE, R., AND WITKOWSKI, M. A comparison of static and moving presentation modes for image collections. *AVI '06: Proceedings of the working conference on Advanced visual interfaces* (2006).

[16] COOPER, M., FOOTE, J., AND GIRGENSOHN, A. Temporal event clustering for digital photo collections. *ACM Trans. Multimedia Comput. Commun. Appl.* (2005).

[17] CUNNINGHAM, S., AND MASOODIAN, M. Identifying personal photo digital library features.

[18] DANIELS, J., HA, L., AND OCHOTTA, T. Robust smooth feature extraction from point clouds. *computer.org* (2007).

[19] DATTA, R., JOSHI, D., LI, J., AND WANG, J. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)* (2008).

[20] DATTA, R., AND LI, J. Content-based image retrieval: approaches and trends of the new age. *. . . on Multimedia information retrieval* (2005).

[21] DENG, J., DONG, W., SOCHER, R., LI, K., AND FEI-FEI, L. Imagenet: A large-scale hierarchical image database. *computer.org* (2009).

[22] FLEMING, M. Categorization of faces using unsupervised feature extraction. *Neural Networks* (1990).

[23] FOGARTY, J., TAN, D., KAPOOR, A., AND WINDER, S. CueFlik: interactive concept learning in image search. *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (2008).

[24] FONSECA, M., AND JORGE, J. NB-Tree: An indexing structure for content-based retrieval in large databases. *Proc. of the 8th Int'l Conf. on Database Systems for Advanced Applications. Kyoto: IEEE Computer Society* (2003).

[25] GABBOUJ, M., BIRINCI, M., AND KIRANYAZ, S. Perceptual color descriptor based on a spatial distribution model: Proximity histograms. In *2009 International Conference on Multimedia Computing and Systems (ICMCS)* (2009).

[26] GIRGENSOHN, A., SHIPMAN, F., TURNER, T., AND WILCOX, L. Flexible access to photo libraries via time place, tags, and visual features. In *Proceedings of the 10th annual joint conference on Digital libraries* (2010).

[27] GIRGENSOHN, A., SHIPMAN, F., WILCOX, L., TURNER, T., AND COOPER, M. MediaGLOW: organizing photos in a graph-based workspace. In *Proceedings* (2009).

[28] HEESCH, D. A survey of browsing models for content based image retrieval. *Multimedia Tools and Applications* (2008).

[29] HEESCH, D., AND RÜGER, S. NNk networks for content-based image retrieval. *Advances in Information Retrieval* (2004).

[30] HILLIGES, O., BAUR, D., AND BUTZ, A. Photohelix: Browsing, Sorting and Sharing Digital Photo Collections. *Horizontal Interactive Human-Computer Systems, International Workshop on* (2007).

[31] HONG, Z. Algebraic feature extraction of image for recognition. *Pattern Recognition* (1991).

[32] HSU, R.-L., ABDEL-MOTTALEB, M., AND JAIN, A. K. Face detection in color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2002).

[33] HSU, S., CUBAUD, P., AND JUMPERTZ, S. Phorigami: A Photo browser based on meta-categorization and origami visualization. *Human-Computer Interaction. Novel Interaction Methods and Techniques* (2009).

[34] JAIN, A., AND VAILAYA, A. Image retrieval using color and shape. *Pattern Recognition* (1996).

[35] JOHNSON, B., AND SHNEIDERMAN, B. Treemaps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd conference on Visualization '91* (1991).

[36] KRISHNAMACHARI, S., AND ABDEL-MOTTALEB, M. Image browsing using hierarchical clustering. *Computers and Communications, 1999. Proceedings. IEEE International Symposium on* (1999).

[37] LEE, D.-J., MITRA, S., AND KRILE, T. F. Analysis of sequential complex images, using feature extraction and two-dimensional cepstrum techniques. *Journal of the Optical Society of America A* (1989).

[38] LIU, Y., ZHANG, D., AND LU, G. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition* (2007).

[39] MINKA, T., AND PICARD, R. An image database browser that learns from user interaction. *Master's thesis* (1996).

[40] MINKA, T., AND PICARD, R. Interactive learning with a "Society of Models". *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on* (1996).

[41] MOJSILOVIC, A., HU, H., AND SOLJANIN, E. Extraction of perceptually important colors and similarity measurement for image matching, retrieval and analysis. *IEEE Transactions on Image Processing* (2002).

[42] NAAMAN, M., SONG, Y., AND PAEPCKE, A. Automatic organization for digital photographs with geographic coordinates. *Digital Libraries* (2004).

[43] NIXON, M. Feature extraction and image processing. *books.google.com* (2008).

[44] PANCHANATHAN, S., PARK, Y. C., KIM, K. S., KIM, P. K., AND GOLSHANI, F. The role of color in content-based image retrieval. In *Proceedings. 2000 International Conference on Image Processing* (2000).

[45] PLATT, J., AND CZERWINSKI, M. Phototoc: Automatic clustering for browsing personal photographs. *... Communications and Signal ...* (2004).

[46] PORTA, M. Browsing large collections of images through unconventional visualization techniques. *AVI '06: Proceedings of the working conference on Advanced visual interfaces* (2006).

[47] QIU, G., MORRIS, J., AND FAN, X. Visual guided navigation for image retrieval. *Pattern Recognition* (2007).

[48] QIU, G., YE, L., AND FENG, X. Fast image indexing and visual guided browsing. *... Workshop on Content-Based Multimedia Indexing*.

[49] RODDEN, K. How do people organise their photographs. *BCS IRSG 21st Ann. Colloq. on Info. Retrieval ...* (1999).

[50] RODDEN, K., BASALAJ, W., SINCLAIR, D., AND WOOD, K. R. Does organisation by similarity assist image browsing? *Proceedings of the SIGCHI conference on Human factors in computing systems* (2001).

[51] RODDEN, K., AND WOOD, K. R. How do people manage their digital photographs? *Proceedings of the SIGCHI conference on ...* (2003).

[52] RUI, Y., HUANG, T., AND CHANG, S. Image Retrieval: Current Techniques, Promising Directions, and Open Issues* 1. *Journal of visual communication and image ...* (1999).

[53] SCHAEFER, G. A next generation browsing environment for large image repositories. *Multimedia Tools and Applications* (2010).

[54] STRONG, G., AND GONG, M. Browsing a Large Collection of Community Photos Based on Similarity on GPU. *ISVC '08: Proceedings of the 4th International Symposium on Advances in Visual Computing, Part II* (2008).

[55] STRONG, G., AND GONG, M. Organizing and browsing photos using different feature vectors and their evaluations. *CIVR '09: Proceeding of the ACM International Conference on Image and Video Retrieval* (2009).

[56] SURAL, S., QIAN, G., AND PRAMANIK, S. Segmentation and histogram generation using the HSV color space for image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on* (2002).

[57] TAMURA, H., AND YOKOYA, N. Image database systems: A survey. *Pattern Recognition* (2002).

[58] TAN, K., OOI, B., AND YEE, C. An evaluation of color-spatial retrieval techniques for large image databases. *Multimedia Tools and Applications* (2001).

[59] TERROSO, E. M. *An Image Organizer with Content-Based Image Retrieval*. PhD thesis, Universidade Federal do Rio Grande do Sul, 2008.

[60] TORRALBA, A., FERGUS, R., AND FREEMAN, W. T. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* (2008).

[61] VASCONCELOS, N., AND LIPPMAN, A. A multiresolution manifold distance for invariant image similarity. *IEEE Transactions on Multimedia* (2005).

[62] WAN, S., JIN, P., AND YUE, L. An Effective Image Retrieval Technique Based on Color Perception. In *Sixth International Conference on Image and Graphics (ICIG)* (2011).

[63] WOJCIECH, K., BASALAJ, W., AND SINCLAIR, D. Evaluating a Visualisation of Image Similarity as a Tool for Image Browsing. *the IEEE Symposium on . . .* (1999).

[64] WORRING, M. Interactive access to large image collections using similarity-based visualization. *Journal of Visual Languages & Computing* (2008).

[65] YUNG, M. Content-Based Image Retrieval. *csse.monash.edu* (2006).

[66] ZHANG, R., AND ZHANG, Z. M. Addressing CBIR Efficiency, Effectiveness, and Retrieval Subjectivity Simultaneously. In *MIR '03: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval* (2003).