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

Chapter 1

Introduction

1.1 Prologue

Since the advent of digital photography, people have been storing their photographs on folders on their computers instead of photo albums on the self. 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 fact, they really don't do anything else then 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 that he was looking for appear.

This is true for the most common photo management software, 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 metadata and have lots of buttons and options for their editing capabilities.

We want to provide the user a totally different way to interact with his 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.

1.2 Context

Present concepts...?

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

1.3 Motivation

Alguns casos de uso

Chapter 2

Related Work

There has been a lot of discussion about the organisation and retrieval methods of large image libraries and some interesting and relevant are summarised here.

2.1 Visual guided navigation for image retrieval

Qiu et al. explore in [28] the requirements of a system intended for visualising 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.

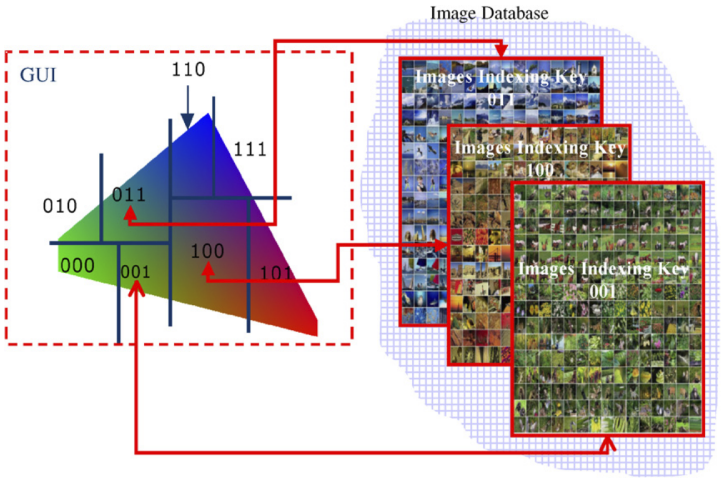

Figure 2.1: The chromacy diagram is split in parts and each image belongs to one of this parts. The diagram is part of the UI and when navigating through the diagram, only the images related to that part are shown.

The system shows all the photos arranged by colour, 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 colours, and arranges them by that value, on a colour map (fig. 2.1). The process is much faster but is also more error prone, specially on photos without a clear main colour.

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

2.2 Does organisation 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 [31] by Rodden and Sinclar was to evaluate how photo organisation 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 organised by similarity way. This organisation 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 labelling. 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 neighbours 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 describes in [27] a few visualisation methods for large collections of images and tests them with users. The purpose it to find ways or metaphors that provide a good visualisation experience in terms of time spent and quality of the visualisation.

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

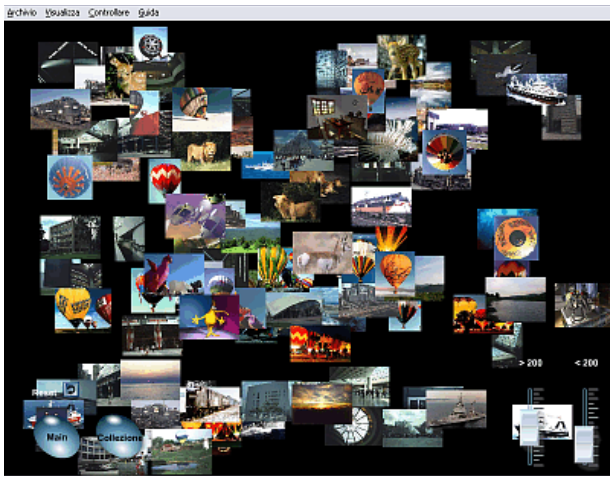

Figure 2.3: Spot display.

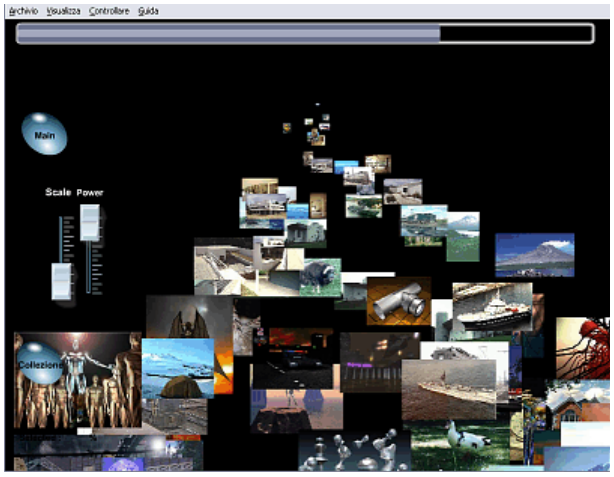

Figure 2.4: Shot display.

get closer to the user called Shot (fig. 2.4), a view where images quickly appear on random positions on screen named Spot (fig. 2.3), 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 [12] by Cooper et al. is not about large libraries but about what kind of interfaces for image showing has greater success of user recognition and preference (fig. 2.5).

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 recognise 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 [36] focuses on the better experience provided by colour organisation of a large image collection (fig. 2.6).

A self organizing map (SOM) is used to display the images on the screen featuring zooming, panning and sorting capabilities. The work is then based around the various methods used to determine the images' similarity.

Simpler methods are based on colour histograms, which aren't affected by rotations or scales but, by

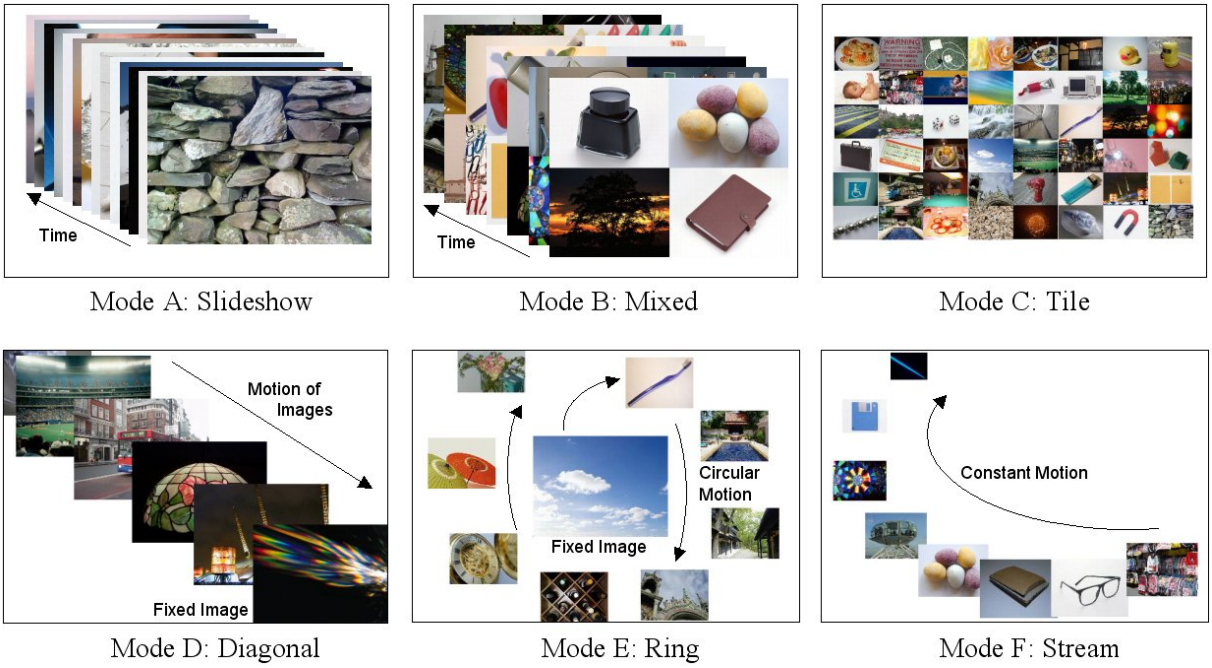

Figure 2.5: The six rapid serial visual presentation modes used in the experiments

not having spacial information on colours, 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 colour. In general, the best methods were found to be the hybrid ones, where both colour histograms and gradients are used to classify the images. No user testing is made in this project, neither is the speed of image categorisation referred about the methods used.

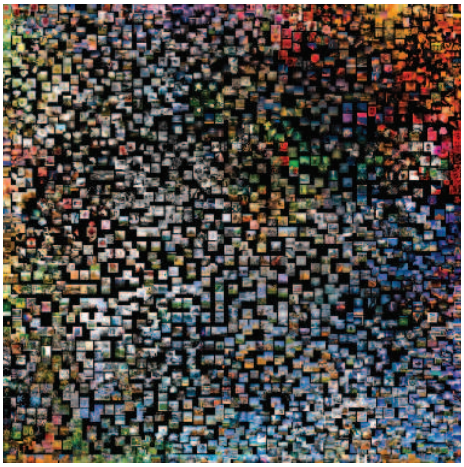

Figure 2.6: The result obtained for organizing 2200+ photos using colour autocorrelogram feature vector, using [36]

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

Tan et al. [38] present an evaluation of three colour-spatial image retrieval techniques.

The signature-based technique creates a signature for each image, based on the most frequent colours, according to a threshold, of each subdivision, or bin, of that image. The comparison between images is made by comparing the main colours 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 colour 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 colour and, therefore, two images will appear similar if both have similar coloured clusters on at roughly the same location. This method extracts the larger clusters and their colour 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 colour information, the brightness was also analysed 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. [25], is described a system that organises digital photographies 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 neighbour 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 organisation of large image databases

Chen et al. present in [9] 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 organised such that similar images are in close proximity

on a two-dimensional grid (fig. 2.7). Images are first organised into a binary tree through agglomerative clustering based on colour, 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.

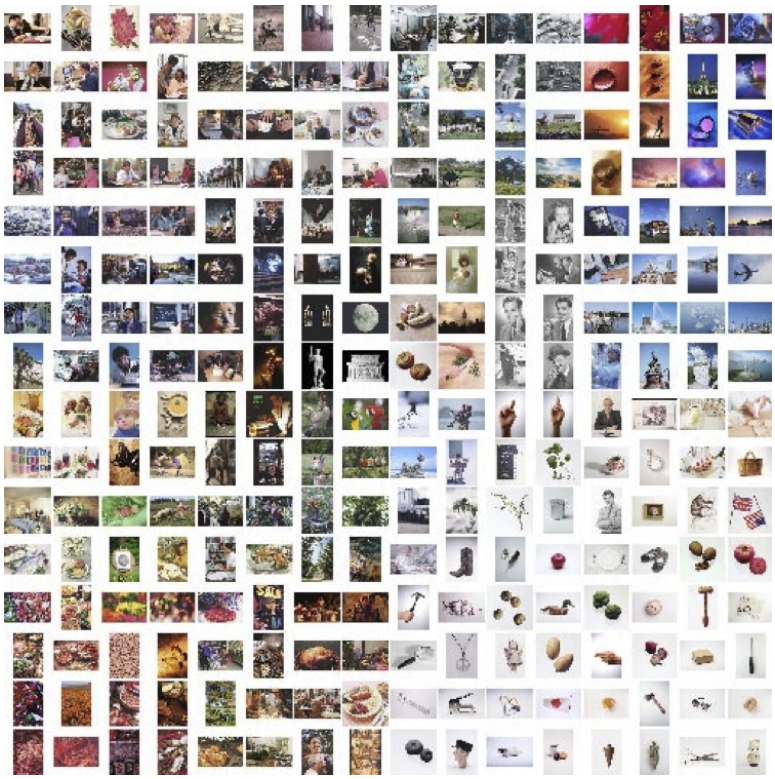

Figure 2.7: Images organised with [9]. Images with similar colour and texture are spatially adjacent.

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 organisation. Finally, a dispersion metric for objectively measuring pyramid organisation was proposed, and found that it correlated well with the author's subjective evaluations of pyramid organisation.

2.9 NN^k networks for content-based image retrieval

Heesch describes in [18] 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 neighbour of the other. Each arc is weighted by the proportion of feature combinations for which the nearest neighbour 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. 2.8) in the network.

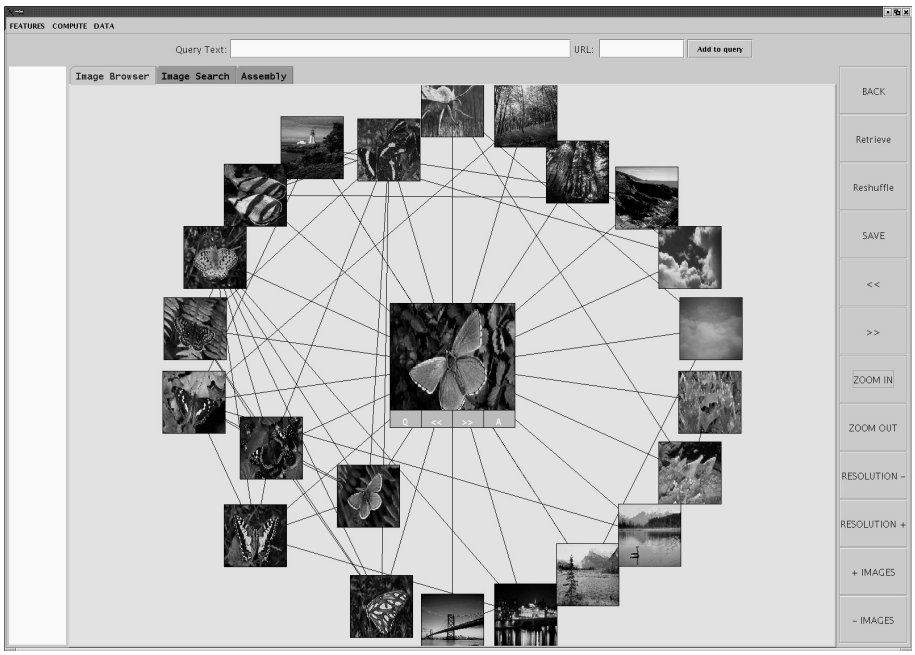

Figure 2.8: Local network around the chosen butterfly image depicted in the middle.

Seven low-level features are used for the classification of the images. HSV Global Colour Histograms maps the images by colour, saturation and brightness; Colour Structure Descriptor maps the distribution of colours by dividing each image in 64 windows, the colour space in 184 bins and associating colour 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. [20] try to ease the browsing problem by analysing the collections and identifying groups of related pictures. Each type of group is visualised 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 categorised 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. 2.9). 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.

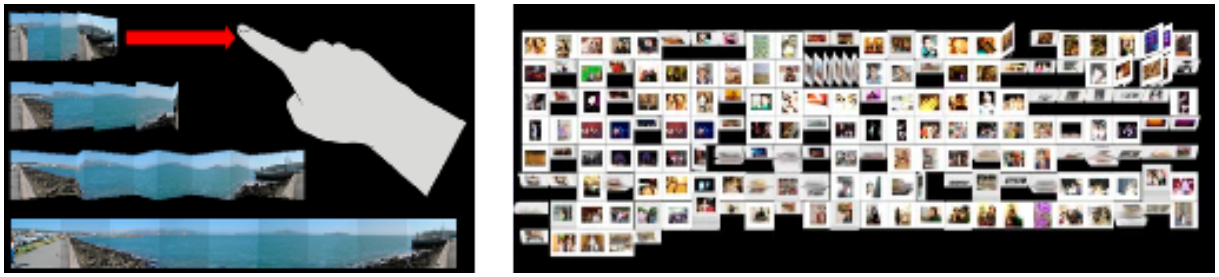

Figure 2.9: On the left, an example of an interaction on a group of photos that makes a panorama. On the right, a visualisation on 537 photos with some groups.

2.11 A next generation browsing environment for large image repositories

Schaefer [34] tries to take similarity based organisation of images from the 2D space to a 3D sphere, which allows interaction from the users. Rotating the sphere reveals images with different colours 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 colour is retrieved by calculating the image's median colour for it's efficiency over other methods like histograms. This two features are directly mapped onto the sphere's coordinates

Figure 2.10: Hue sphere of a dataset

and the entire structure is pre-calculated so browsing can be performed in real-time. Image overlapping is also avoided (fig. 2.10).

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. [16] 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. 2.11).

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.

Colour coding is used throughout the interface to help the user understand better what is being

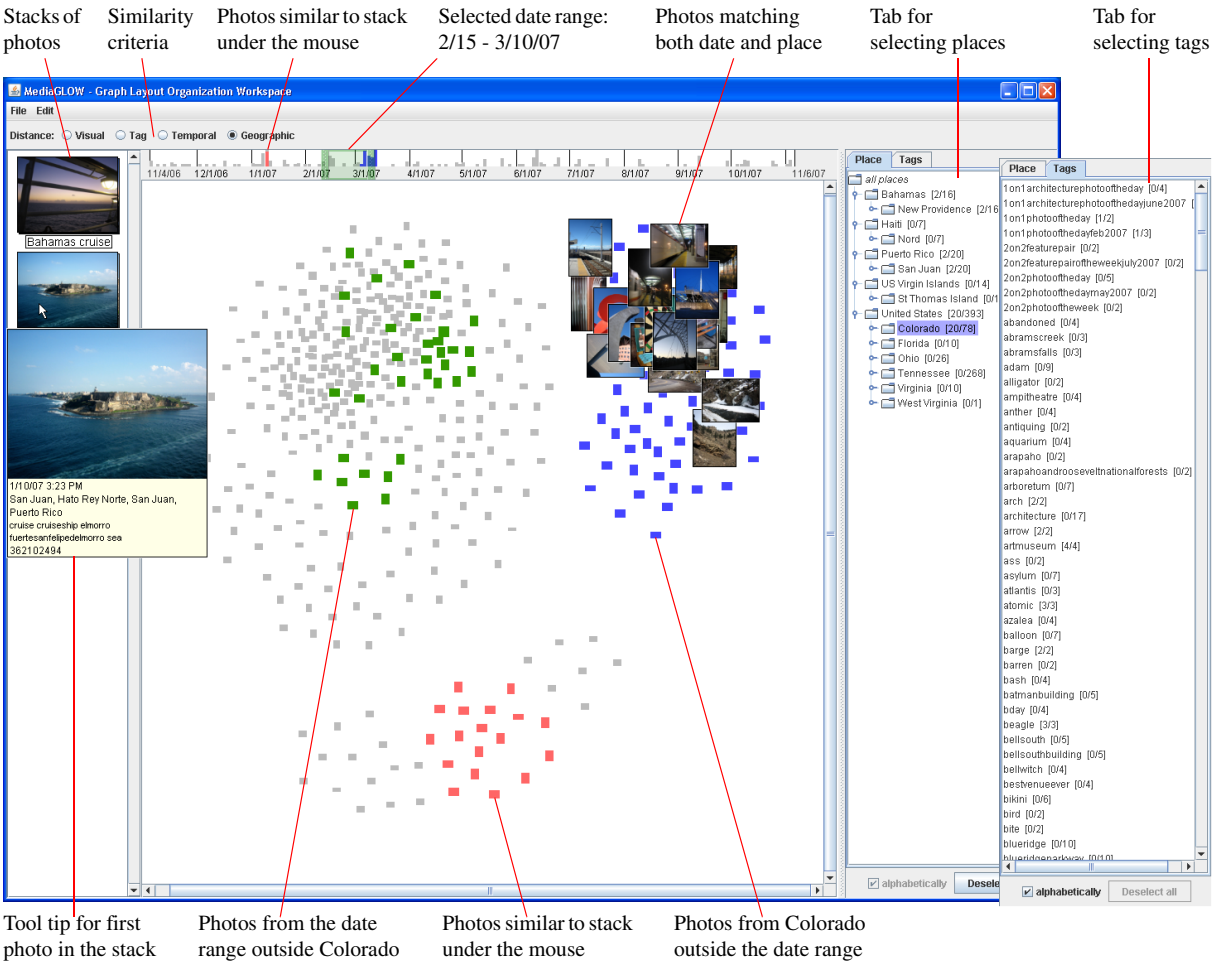

Figure 2.11: Photos Grouped by Geographic Similarity and Filtered by Date and Place.

selected. 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 coloured 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 organise and search the collections was emphasised 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 Discussion

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

Table 2.1: Different browsing methods

Work	Organisation				Visualisation	Focus	Size
	visual	date	gps	tags			
2.1 Qiu [28]	simple colour measures	—	—	—	Grid	Simplicity and Efficiency	60,000
2.2 Rodden [31]	—	—	—	✓	Grid	Similarity usefulness	100
2.3 Porta [27]	—	—	—	—	Spot (and others)	Unconventional visualisations	400
2.4 Cooper [12]	—	—	—	—	Static and animated	Usefulness of animations	—
2.5 Strong [36]	colour histograms and gradients	—	—	—	SOM	Evaluation of different features	2,200
2.6 Tan [38]	colour histograms of subdivisions	—	—	—	—	Evaluation of different features	12,000
2.7 Naaman [25]	—	✓	✓	—	—	Organization based on events	?
2.8 Chen [9]	colours, edges and textures	—	—	—	—	Efficient fast-sparce clustering	10,000
2.9 Heesch [18]	six different features	—	—	✓	Radial	Complex similarity network	32,000
2.10 Hsu [20]	—	—	—	—	Grid with groups	Interaction on grupped photos	1,333
2.11 Schaefer [34]	colour histograms of subdivisions	—	—	—	3D Sphere	3D mapping and interaction	4,500
2.12 Girgensohn [16]	—	✓	✓	✓	Overlapped graph	Having the best way to find photos	450

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 [16] 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 [25]. 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 from [34] is also interesting but doesn't provide a better interface to the collection

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

than expanding the sphere surface grid view to a full screen grid view, keeping the zoom function. The Phorigami work [20] 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 [16] visualisation approach, where images can be organised 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.

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.

For this to happen, the system must meet some requirements that we will specify on this chapter.

3.1 Goals

We will now expose the main goals for this thesis as well as some design goals that the work must feature.

3.1.1 Main Goal

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

More specifically, the work should:

- provide the user with a full view of his photo collection, by displaying a large number of images in a single screen;
- allow the manipulation of the display of images so that the user can create different views that allow new perceptions of his collection.

With this capabilities, the 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.

3.1.2 Design Goals

We also defined some additional requirements related to the design of the solution.

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 easier to use systems get the job done faster, are more enjoyable to use and allow less experienced users to be able to use it. We paid attention to this and did an effort to build an easy to use but powerful system.

Extensibility

The extensibility factor of a system is also important for the added value that can be obtained by quickly adding new features. 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 work.

Performance

Our work has to be performent enough so it doesn't frustrate its users.

What else?

Chapter 4

Eagle Eye

Eagle Eye 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 the canvas of the image collection, sorting through different methods and multiple ways of filtering, either textual or visual.

In this chapter, we will detail the various components that make up this work and how they interact.

4.1 Design Decisions

DeepZoom

After analyzing some visualization technologies that allowed easy display and manipulation of images TODO: Insert a bunch of "useless" tech, Microsoft's Silverlight, with its DeepZoom technology, proved to be the best choice.

DeepZoom enables the use of multiple resolution images for efficient display at various zoom levels and, for instance, is used for the display of gigapixel photos [1], where the user can view the whole image or zoom in the small details, or for collections of photos where the user can zoom between a view of multiple images and the details of a single image[2]. So far, usages of DeepZoom have been restricted to promotion websites, art galleries and other closed usages.

This should be about the multi-scale images and not about DeepZoom → Our work aims at bringing this technology to the regular user, in a much simpler and dynamic way.

Although DeepZoom seemed a great technology, it relies on Silverlight, which by itself isn't as good as using a full-fledged desktop application framework like Windows Presentation Foundation (WPF) or other frameworks native to their platforms. This created some undesired limitations like the need to have two separated parts of the system, the visualization and the backend, and other smaller problems like limited access to the disk from the visualization part.

To create a DeepZoom application, some pre-processing is required before hand to create multiple

[1] Gigapixel photo of San Francisco: `http://theeyegame.com/DZSL/TwinPeaks5DMk2`
[2] Hard Rock Memorabilia: `http://memorabilia.hardrock.com`

versions of each image in various resolutions and also to create imagery of a global view of the collection, also in multiple resolutions. This enables DeepZoom to only load the appropriate set of images for a certain zoom state, keeping the bandwidth (if used on the Internet) and memory requirements to a manageable level. This required pre-processing is done once on the backend. The visualization then uses the generated data to display the collection, being, at this point, totally independent from the original image files belonging to the user.

4.2 Backend

The backend is one of the two parts that make Eagle Eye. It is a command line utility that creates a library upon opening that the user must fill by adding the paths of folders containing images. The system will then read those images, gather their metadata, process them to extract visual features and, finally, generate and output the multi-scale imagery alongside with control metadata for the visualization.

We will now detail the various parts of the backend.

4.2.1 Architecture

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 JPG 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 Exchangeable image file format (EXIF)[3] metadata, information stored by the camera at the moment of the capture, like time, date, and camera information. This information is gathered upon import and is also available to other modules to use.

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[4], to crawl through the folder structure while identifying all the JPG images and returning their EXIF metadata which is then saved by our system (fig. 4.1)

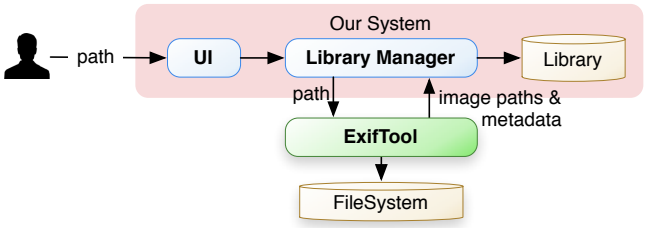

Figure 4.1: Process in use by our system of importing a folder using ExifTool.

Another option for importing metadata was to invoke EXIFTool as part of a feature extraction plugin

[3]EXIF is a standard that specifies the formats for tags used by digital systems handling image and sound files recorded by digital devices. `http://en.wikipedia.org/wiki/Exchangeable_image_file_format`

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

(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 arrange the images in 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.

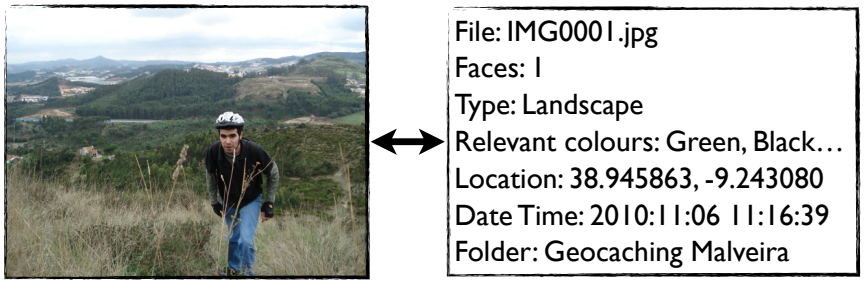

Figure 4.2: Example of information extracted from an image.

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 plugins in the future.

Feature Extraction Plugins need to implement a common interface and are given the ability to process images and save the gathered data. They have freedom to access the image file or its EXIF information and to structure the gathered data the way best suited for its purpose. They should, in the end, store the processed data in a specified way so it gets exported to the visualization. Existing 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 already gathered information. 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[5] 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's essential with libraries with tens of thousands of images, where keeping a complete library in memory is not feasible.

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

4.2.2 Feature Extraction Plugins

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, codes them in a defined way and appends them to the rest of the information to be exported for the visualization.

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. 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, for instance, that the photo had some dominant color, for instance, the light blue of sky, the green of the grass or from a park, the dark blue of the sea, maybe some big red ball that someone was playing with. This information can be helpful when searching for a photo in a collection.

Color extraction from images has been a long standing problem TODO: insert a bunch of references for color detection. One relevant work that we based this plugin is from Qiu et al[28] for exploring a method of image indexing that is fast and simple. On his 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 (fig. **??**). For our plugin we used the open source library AForge.Imaging[6] to obtain histograms for the images and compute their average color in RGB[7] and HSL[8] values. Both those values are then stored on the plugin-extracted data for each image.

this probably shouldn't be here This plugin demonstrates that color extraction is feasible and, with more time, a better method that does not average the colors but instead detects the most relevant ones could be implemented. TODO: references needed

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

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

[7] Red, green and blue `http://en.wikipedia.org/wiki/RGB`

[8] Hue, saturation and luminance `http://en.wikipedia.org/wiki/HSL_and_HSV`

Face Detection

The face detection plugin is based on the open-source OpenCV library[9] which processes every image file and detects existing faces (fig. 4.3).

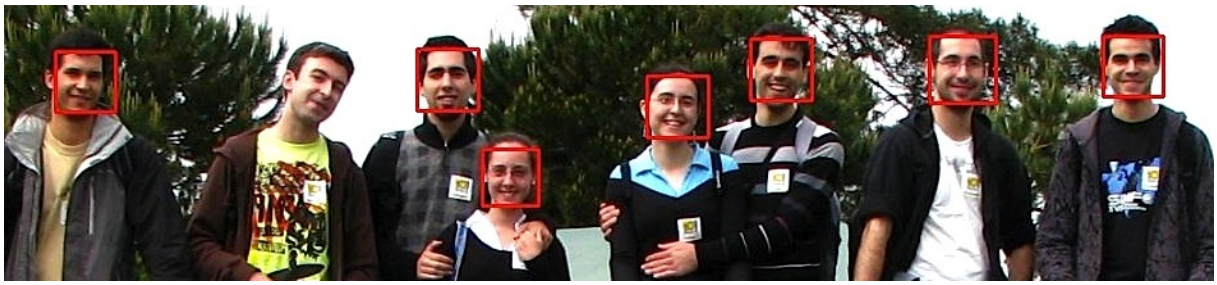

Figure 4.3: Example of the OpenCV library detecting faces on a common photo.

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. 4.3) 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. 4.4.

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. 4.4.

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[10] 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.

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

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

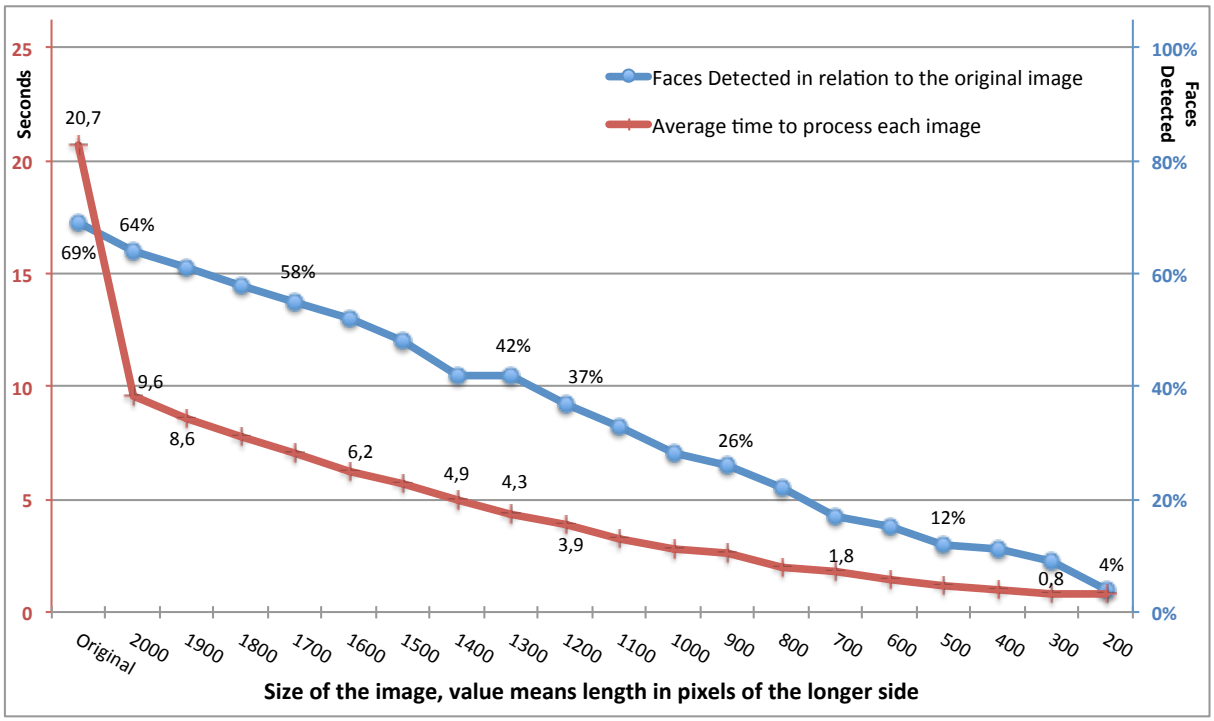

Figure 4.4: Results of the face detection test. 100% of faces is the actual faces present in the test images.

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.

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 enough to allow for an enjoyable 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. After loading the metadata, the user is presented with a set of options on the top toolbar, which is the only UI needed to use the system. TODO: Add picture

According to TODO: insert the reference of the dude who claimed that a 32x32px image was the minimum for recognition, an image with 32 pixels per side is the minimum size that allows a user to recognize an image.

Unsure about how well this fits here... Upon loading Eagle Eye's visualization system, thousands of images might be displayed and this 32 pixel might not be met and, therefore, it might be difficult for the user to recognize what is on display from a single image, but since a lot of them are being displayed, the user might be capable of making sense of the groups by their main colors.

The Canvas

The canvas is the most relevant part of the visualization as it dynamically displays all the images previously selected by the user, at the same time. This may make the images barely recognizable and, therefore, the user has the possibility of manipulating the canvas to see and enjoy the images. This means that 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, until the full screen view of one of them, and everything in between in a smooth way.

The toolbar

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

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

The middle section 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. One of the options is selected at all times and the content is presented accordingly. Changing the selected option causes the images in display to move around to the new position and form a different sort order. This sorting and disposition options will be explained in a later section. The other button in the display section of the toolbar 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 a different colors and presents a name for them, depending on the selection sort option. 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.

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 TODO: References needed . We chose a few options to allow some flexibility for the user, while trying to keep the interface simple.

We are now going to explain how do we sort images into groups and the different ways the user can arrange those groups on the canvas.

Sorting images into groups

As we've seen, the backend outputs metadata for each image. This metadata is loaded into the visualization application of Eagle Eye and is indexed by their 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 dividing the hue spectrum in bins TODO: specify which and add the spectrum image.

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 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 TODO: References needed and picked the ones that we thought that made sense and also that would be easier for the user to understand. We chose a tree map 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 like TODO: References needed 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 tree map. The problem with tree maps is that they are designed for areas that can take many forms, from squares to thin lines TODO: insert tree map image. Applying tree maps 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 TODO: gajo on his work in the Quantum treemaps. We tried to apply Quantum tree maps as our tree map algorithm but due to it's complexity and recurring problems, we adapted the tree maps of TODO: gajo, as used in Prefuse TODO: verify TODO: References needed to the reality of the image grid. This new algorithm was much easier to understand and implement, although it might require a small fine running for a couple of edge

cases.

Our tree map algorithm displays larger groups first, leaving the smaller ones to the end and makes an effort to layout groups as rectangles with an aspect ratio as close as possible to the screen's aspect ratio, for when the user zooms in, the groups fill the screen. There's also an effort to fill the space left between larger groups, making the display more compact and with less holes.

One problem of the tree map 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 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 collapse the couple pictures the user might take of his regular day between days 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 around the canvas 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 user to focus on a specific group of pictures that are his focus of interest at the 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 TODO: References needed : 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 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 have not been implemented yet, they are in the plans.

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 TODO: image.

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. So we made this feature allow various selections and only when the user forces the filtering that all the selections are unified and added to the rest of the intercession filters.

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 and images' metadata.

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.

Since the transport is a text file, we used a simple, text based, key/value capable encoding system: JSON TODO: References needed . The backend encodes key/value pairs of metadata for each image TODO: meter um exemplo em código and the visualization decodes and aggregates them in 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, using N bins for TODO: lalala
- 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 and retrieval of indexers.

This indexers are the base for the sorting and disposition techniques we've covered on 4.3.2. Each indexer has attached the preferred way to be displayed, for instance, the path indexer works better when displayed using a tree map and thats what the "Path" button on the toolbar will use to dispose the images, but the date has two buttons, "Date" and "Date (linear)" because it makes sense for it to be displayed using both disposition techniques. could be better

Canvas

The canvas is the most important 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 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 passed the filter, get the respective groups 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 for common 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, allowing a faster interaction.

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. [20] 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. TODO: images

Obviously that this can have both false positives or false negatives but in our tests those were in a small number. UI to fix that could be added but we don't feel that was essential.

should make a section just for the chapter conclusion?

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 Future Work (**??**) contains many ideas we have had which could bring this work to be a real world application.

Chapter 5

Evaluation

Chapter 6

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 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.

6.2 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 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.

[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`

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.

6.3 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.[16] 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. Slideshow is an 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.

maybe remove the "concluding" bellow?

6.4 Concluding

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.

Chapter 7

Conclusions

Insert your chapter material here...

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. pp. 71–80.

[4] BERMAN, A., AND SHAPIRO, L. A flexible image database system for content-based retrieval. *Computer Vision and Image Understanding* (Jan 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* (May 2000).

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

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

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

[9] 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 3299* (Jan 1998), 563–575.

[10] 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), 445–448.

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

[12] 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* (May 2006).

[13] CUNNINGHAM, S., AND MASOODIAN, M. Identifying personal photo digital library features. 401.

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

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

[16] GIRGENSOHN, A., SHIPMAN, F., TURNER, T., AND WILCOX, L. Flexible access to photo libraries via time place, tags, and visual features. pp. 187–196.

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

[18] HEESCH, D., AND RÜGER, S. Nnk networks for content-based image retrieval. *Advances in Information Retrieval* (Jan 2004), 253–266.

[19] HILLIGES, O., BAUR, D., AND BUTZ, A. Photohelix: Browsing, sorting and sharing digital photo collections. *Horizontal Interactive Human-Computer Systems, International Workshop on 0* (2007), 87–94.

[20] 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), 801–810.

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

[22] KRISHNAMACHARI, S., AND ABDEL-MOTTALEB, M. Image browsing using hierarchical clustering. *Computers and Communications, 1999. Proceedings. IEEE International Symposium on* (1999), 301 – 307.

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

[24] 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), 447 – 452.

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

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

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

[28] QIU, G., MORRIS, J., AND FAN, X. Visual guided navigation for image retrieval. *Pattern Recognition 40*, 6 (2007), 1711–1721.

[29] QIU, G., YE, L., AND FENG, X. Fast image indexing and visual guided browsing. *... Workshop on Content-Based Multimedia Indexing* (2010).

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

[31] 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), 197.

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

[33] RUI, Y., HUANG, T., AND CHANG, S. Image retrieval: Current techniques, promising directions, and open issues* 1. *Journal of visual communication and image ...* (Jan 1999).

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

[35] 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* (Dec 2008).

[36] 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* (Jul 2009).

[37] TAMURA, H., AND YOKOYA, N. Image database systems: A survey. *Pattern Recognition* (Oct 2002). Extensa lista de image browsers....

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

[39] 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. 30*, 11 (2008), 1958–1970.

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