Eagle Eye
Visualizing large photo collections

Carlos Eduardo H. J. Fonseca
Universidade Técnica de Lisboa
Instituto Superior Técnico
Av. Rovisco Pais
Portugal
carlosefonseca@ist.utl.pt
2010

Abstract. Digital photography has been a part of people’s lives for the past decade, growing on hard drives, but image visualization has not been evolving enough to accompany this growth. Most people keep photos on folders or use software that can’t display a large number of photos at the same time, not providing a good overview of their collection. This work grew from the features of other works of alternative image visualizations and tries to bring a better way for regular people to explore their photo libraries and learn more about them. This work provides an extensible backend system intended for image and metadata processing, gathering information about the user’s photos, currently colors, faces, dates, paths and keywords, but, since it’s extensible, could be used for much more. We also created a visualization application to use the processed data and provide it to the users, in an interface that is clean and simple. It starts by displaying all the added images on the screen, at the same time, in a grid, grouped by dates. From here, the user can zoom in and out and change the display to organize the images by colors, faces or paths, and can filter using a mix of any data gathered by the backend. This way, users can view and explore their libraries in a different way from what they have been doing.

1 Introduction

The growth of the digital brings the problem that photo collections have started to grow more than they used to. But storing digital photographs isn’t that different from the past. There’s currently no good way to have a global view of the collection, understanding its evolution and characteristics. We propose that visualizing photographs in a birds-eye view, with a number of different ways to organize, group, sort and filter them by using features like events or dates, dominant color or people in photos will provide users a much more satisfying
experience of discovery of their libraries. To demonstrate this, we implemented Eagle Eye, a system that allows users to do just that. After testing, we observed that our approach allows users to navigate and explore their photo libraries in a whole new way, revealing patterns or evolution over time, while enjoying their photos.

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 Related work

Organizing and browsing photos using different feature vectors and their evaluations  Strong [14] focuses on the better experience provided by color organization of a large image collection. 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. In general, the best methods were found to be the ones where both color histograms and gradients are used to classify the images.

Phorigami: A Photo browser based on meta-categorization and origami visualization  Hsu et al. [7] 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.

    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. 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.

Flexible access to photo libraries via time place, tags, and visual features  MediaGLOW by Girgensohn et al. [4] is the application discussed in this paper. It's a content based image retrieval 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.

The photo display is graph based, allowing for overlapped images. 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.

Color coding is used to help the user understand better what is being selected.

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.

**PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps** This work presents PhotoMesa by B. B. Bederson [1], an application that supports browsing sets of images in a zoomable environment. Users can choose directories of images and they are all displayed in a space-filling manner. Users can zoom in to a group and then to a single image. The groups display their name and have different background colors for a better distinction. Text search is possible as well as selecting a group on a list.

For space filling uses Quantum Treemaps, a variation of the Ordered Treemap, aware of the constraints imposed by the use of images, and Bubble Maps, designed to have the least wasted space possible.

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

### 3 Solution Requirements

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.

#### 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 so it’s possible to organize and classify them. We chose face detection, color detection, image sequences and metadata extraction.
Provide an interaction with the full set of images and, if the want to, make smaller sets from the full library.

Efficiently display a large number of images in a single screen by using some techniques to reduce clutter.

Allow the manipulation of the display by different groups, sorts, filters, selections, zooming and panning.

With this capabilities, our work should be able to provide the user not only 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 ourselves some 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: Ease of use, Extensibility, Performance and Persistency.

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.

4.1 Overview

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.

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 and the control metadata required by the visualization.

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

The Eagle Eye part is the main application, containing the library and feature extractor managers. Both deal with files on the disk, JPEG image files and DLL extractor plugins, respectively. The user interacts with the core of Eagle Eye which currently provides a command line interface for its actions, like the image import and plugin execution. The import gathers the files and their metadata, to be later accessed by the plugins for processing. Plugins store the resulting data inside the library manager and can be accessed afterwards for outputting by a special plugin.

![Basic architecture of Eagle Eye's backend](image)

**Fig. 1.** Basic architecture of Eagle Eye’s backend

**Feature Extraction Plugins** We will now overview the extractors for the features referred back in Solution Requirements (3.1).

**Selection of useful image metadata** This plugin acts as a filter for all the available Exchangeable image file format (EXIF) tags picking the most relevant ones, coding them in a pre-defined way and appending them to the rest of the information to be exported for the visualization.
Timestamp of capture, device, path on disk and keywords are the currently extracted tags.

**Detection of image’s main color** We looked into a couple of different ways to tackle the color detection extractor but, due to time constraints, we only used a simple method that computes the histogram and returns a median color for each image. It works but induces some errors in images with high contrast.

**Face Detection** The face detection plugin is based on the open-source OpenCV library\(^1\) which processes every image file and detects existing faces.

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.

**Generation of multi-scale imagery** This plugin generates all the data files needed for the visualization to work. The visualization relies on the DeepZoom technology which needs to process the images before they can be displayed. This plugin does exactly that.

The plugin generates, for each image, 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 processed, 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 overview this part of Eagle Eye, its architecture, visualization techniques, sorting and filtering capabilities.

**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 processing, the visualization can be opened and all will appear in “the canvas”. After loading the metadata, the user is also presented with a set of options on the toolbar (fig. 2).

**The Canvas** is the most relevant part of the visualization as it displays the user’s images.

---

\(^1\) OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision. Available at [http://opencv.willowgarage.com](http://opencv.willowgarage.com)
Eagle Eye displays images very small, but because the interest is in users viewing their own collections, they user recalls what is that image without having to understand a totally new image. The grouping of events (e.g. by date or path) also helps 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 small images.

Eagle Eye allows manipulation of the canvas by panning and zooming up to fullscreen images.

**The toolbar** The user can use the functions on the toolbar to filter and sort differently.

**Disposition of Images on Canvas** We have come up with a some options for photo grouping, sorting and filtering.

**Sorting images into groups** Eagle Eye currently groups together images captured at the same day, with the same median color, from the same path or with the same number of faces, depending on user choice.

**Different dispositions** We implemented a Treemap for displaying most of different grouping options. This treemap is aware of the constraints of dealing with images. We also implemented a “linear” display for displaying images sequentially.
Filtering Eagle Eye also provides the user with the ability to filter images in two ways, using the filter bar for textual filtering or selecting pictures on the canvas.

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

Metadata indexing The visualization uses metadata embedded by the back-end, which is obtained, parsed and indexed for later use.

Canvas is a wrapper for the DeepZoom display. It stores settings and images positions and computes new ones. It also uses memoization to avoid repeated calculations.

Reducing Clutter Eagle Eye groups images that were taken within an interval of 4 seconds. A whole group is reduced to the space of one image, keeping the first one at its normal size, but not centered on the grid. The resulting space left is used for the other images. The user can always zoom in to see them.

5 Evaluation

We performed some user testing with 18 people, to test the interface. They liked the new approach but pointed out some flaws, most of them could be corrected with more work on Eagle Eye. We asked them to answer the System Usability Scale inquiry and it got a rating of 75, on a scale from 0 to 100.

We also performed some tests with a few people using their own libraries, so we could see how they react to their own photos being displayed on a new way. The general feedback was very positive, with users being engaged in the interaction. They also identified groups of photos much more easily than with another person’s library, as expected.

6 Conclusions

Photo visualization is an eternal problem with lots of compromises. Big software companies sell the simple ways with big lists of photos, but many other ways have been explored for years, some more eccentric than others. Some more useful than others. We explored their good and bad points and selected the features we thought important. Made a user survey and fine tuned our vision and Eagle Eye was born.

Our work aimed at being useful to users who have a regular collection of photos on their computers and want a better way to view and learn more about them. We built a system that gives users a completely different view of their
collection, by showing everything at the same time. While overwhelming at first, we have seen that users like it and, by already having knowledge about their own library, they can understand each group of photos, whether their photos were taken yesterday or ten years ago. From this, we allow users to change, reorder, filter, zoom and enjoy their pictures.

For this to be possible, we developed a system that extracts information from images and that can be expanded with more processes to obtain interesting and uncommonly seen information. This allows users to observe their collection in a variety of ways, by selecting different information and filtering down to specific sets of images.

Although we didn’t have time to implement many features we had in mind for an even better system, our user tests revealed that people liked the system, giving it a moderately good rating of 75 in a scale of 0 to 100.

We think we have a solid concept that, with some more development, could be made as an application that could actually be released to the public.

6.1 Future Work

During the realization of this work, many ideas popped in our minds but we didn’t have the time to work on them. The system needs some consolidation, specially on the backend. The feature extraction plugins should also be improved, like the color extraction, and a few could be added. The visualization should be improved, by providing more help to users, by increasing the features and by fixing the bugs. It would also be interesting if we could connect Eagle Eye to calendaring and location-based services to obtain more event and location data for photos.

References