Interactive design wall over a shared screen GatherMind

Rafael Pereira 60700 & Renato Viola 60665

2 April, 2024

Contents

1 Introduction

Both design and creative work are dynamic by nature, and as such, demand an environment that not only fosters collaboration but also adapts to the evolving needs of a team.

This is especially true in a world so fast-paced and increasingly digital, like the one of today.

In response to this need, we propose **GatherMind**, a useful and easy-to-use collaborative workspace in the form of an interactive design wall, specifically designed to stimulate creativity and innovation among team members.

Across a large shared screen, participants will be able to engage in real-time collaborative design and discussion, sharing their own visions of certain design concepts whilst collectively refining and iterating on them.

To this end, we will be using **OpenFrameworks** and **OpenCV** to integrate somewhat advanced multimedia computing technologies into our system, such as facial and object recognition. The screen allows for a dynamic display of image/video designs, alternatives, previous approaches and new ideas. Furthermore, searching and comparison operations are enhanced through the metadata information associated to each media file in the system.

In order to be more accessible, interacting with the system doesn't require a physical input device, as all the controls are through touch and gesture recognition.

In the following sections, we will explore related systems that inspire our approach, and address the preliminary specifications that guide the development of this project.

2 Related Work

ShapeCompare [1] is a system that allows users to interactively generate and compare multiple design alternatives of a *CAD* (*Computer-aided design*) model, by having them distributed across a wall-sized display. These multiple representations of a *CAD* object foster a collaborative environment where design ideas can be quickly visualized and iterated on, by multidisciplinary teams that may even contain *non-CAD* experts.

By aiming to reduce the miscommunication between the experts and engineers in charge of modifying the *CAD* data, it avoids unnecessary iterations in the design process.

The study in which the system is presented also demonstrates some of the advantages for co-located collaboration in wall-sized displays over the standard desktop displays we are accustomed to, by allowing multiple users to interact with the information simultaneously.

Our project shares several objectives with this study, so there's a lot to take away from it.

Much like *ShapeCompare* [1], **GatherMind** aims to facilitate collaborative work by enabling real-time interaction and iteration on design concepts, by the team's members.

Not only that, the presented research demonstrates that being able to visualize multiple design alternatives/versions concurrently enhances the design process and speeds up decision making. We intend to incorporate a similar capability into our application, as it would greatly increase its utility as a brainstorming tool.

ShapeCompare's [1] success in enabling even *non-CAD* experts to actively participate in the design process and, through touch, manipulate complex data in an intuitive way did not go unnoticed.

Even though we won't be manipulating data as complex, we'll try to adapt this approach to our project, so that users can effortlessly modify content on the display.

Especially in a meeting room context, it's important that **GatherMind**'s interface is welcoming to all participants, regardless of their level of expertise.

In the context of the *VICE* [2] project and especially e-Learning, **MILOS** is a Multimedia Content Management System suitable for developing digital library applications and efficiently supports advanced features for persisting, searching and retrieving multimedia learning objects in the form of XML documents.

The paper demonstrates how a digital library's architecture focuses on the reutilization of existing multimedia content by enhancing the basic media files with metadata, thus sparing the system of the time and cost of creating new content and also annotating it.

Such reuse of multimedia documents is achieved by generating descriptors for this content, through automatic metadata extraction tools.

This process, alongside a web interface that offers users the possibility to perform a combination of full text searches with the selection of certain fields of the metadata's structure and/or similarity search on visual descriptors, allows for effective searches and retrievals.

From studying this project, we believe to have gathered some valuable ideas for our own, even though ours is focused on collaborative design rather than e-Learning.

By automatically extracting metadata from uploaded media content, similarly to the digital library mentioned above, we hope to ensure that interacting with design concepts in our application is intuitive, and that searches as well as retrievals are efficient.

3 Algorithms and techniques

3.1 Facial and Gestural Recognition

For facial and gestural recognition, we utilized *ofxCvHaarFinder* in conjunction with specialized Haar cascade classifiers. Such classifiers are a machine learning-based approach where a cascade function is trained with lots of positive and negative images to detect objects in other images.

Specifically for facial detection, we used the *haarcascade_frontal_face_alt_tree.xml* classifier, which seemed to be much more consistent than the default used in the **OpenFrameworks** example, and introduced much less artifacts. Gesture recognition was made possible by the *aGest.xml* classifier, the most accurate one that we could find after testing several.

3.2 Contour Detection

To detect and display contours in a frame of any media file, we've employed *ofxCvContourFinder* with an adaptive threshold. This seemed the best approach as there was need for decent detection of contours in varying lighting conditions.

In order to achieve this effect, we made use of the **THRESH_BINARY** and **THRESH_OTSU** techniques, and combined them. **THRESH_BINARY** converts grayscale images to binary images based on a fixed threshold, while **THRESH_OTSU** automatically calculates the optimal threshold value.

3.3 Similarity Comparison of Media Files

For comparing the metadata of different media files to determine their similarity, on all attributes aside from tags, we used the Euclidean distance metric - the straight-line distance between two points in a multidimensional space.

This appeared to be most suitable measure for comparing the numeric attributes of media metadata.

In determining how similar two files are, the number of tags in common between the files has the most weight in the calculation, the other attributes are mostly used to resolve the ties when several files have the same common tag count.

3.4 Extraction of edges and texture characteristics

To extract metadata information, specifically edge detection, several edge filters were applied, in order to highlight boundaries and transitions in varied orientations: Vertical, Horizontal, Diagonal 45º, Diagonal 135º and Non-Directional.

For extracting texture characteristics, we used 24 Gabor kernels with varying values for the theta (orientation) and sigma (scale) attributes.

These filters are highly effective in texture analysis as they can capture both spatial and frequency information, representing textures with reasonable degree of nuance.

3.5 Keypoint Extraction and Detection

For keypoint extraction and detection, our solution implements the Scale-Invariant Feature Transform (**SIFT**) method using **OpenCV's** Feature2D class, very much inspired by *example_16-02* of the **Learning OpenCV 3** repository.

SIFT detects and describes local features in images, enabling the identification of keypoints invariant to scale, rotation, and illumination changes.

To enhance the performance of extracting this information from several image and video files at once, we parallelized some of the solution using **OpenMP**, which leverages multi-core processors to accelerate the computation.

4 Application implementation

At launch, the application checks if every file in the system has an XML document associated to it. If the XML file isn't present, it will automatically generate the file and its correspondent metadata. This information consists of:

- **Tags**, that for now must be edited by the user themselves, to make it easier to find and locate the files;
- **Distribution of pixel values**, such as luminance, colour, edge distribution, and texture characteristics which can then be used to compare related files;
- **List of the eight closest related files**, which is only present when there has been a previous search for the file's versions and related files;

All pages aside from the screensaver and homepage, have a button (*Home icon*) at the top left corner of the screen, that redirects to the homepage, a home button so to speak.
Upon launching the application, a screensaver with the application's logo is displayed. If significant movement is detected, the application will transition to the homepage, consisting of images and video thumbnails arranged in two separate lists, which can be swiped left or right to select a different file.

With the file selected, the user can then click on it to view it in **full-screen mode**. This view mode includes a GUI which allows users to apply several filters to the files and persist these changes, in a dedicated XML document.
This viewing mode varies slightly depending on whether the selected file's type as the ASCII filter is exclusive to images, and videos may be paused and resumed.

In both the full-screen video and image screens, the user can click the bottom-most button in the top left corner of the screen (**version-control** icon), which redirects the application to a page circularly displaying the different versions of the product being presented as well as the files most closely related to the one that was selected (only eight thumbnails are displayed at once and versions take priority), which makes it possible to show alternative visualisations of the product's design and its different iterations.
When this sort of search for a specific file occurs for the first time, the eight most related files are explicitly computated and appended to the file's correspondent XML document. In subsequent searches, the related files no longer have to be computed, only having to be retrieved from the XML document to which they were appended earlier.

Versions of the file are annotated with their identifier and modification date, related files are annotated only with their type.
The user may open any of these files in full-screen by simply clicking on them. If the clicked file is a version of the selected file, it is loaded with the changes correspondent to that specific version.

From the homepage, there are two buttons in the top corners which allow the user to navigate either to the screensaver (lock icon) or to an interface in which the webcam image is enabled and displayed (camera icon).
In this interface, the user may either choose to enable facial detection (face icon) or present an object to the camera and request the system for the files where this object is most likely to occur (magnifying glass icon).

Once the system completes such request, the user is greeted with an interface in which the four best matching files are presented in a circular display around the center of the screen, and a live view of the camera in the middle.
In this interface, gesture recognition is enabled. The user may select the file it wants to open either by clicking on it or by placing its closed fist near it.

5 Class description

Our classes can be divided into 3 different types, each with its own objective.

5.1 DataTypes

The classes in this section focus on representing different types of data and include their own set of functionalities.

- **Media**: Represents any kind of Media that our application can interact with;
- **ImageMedia**: Extends the *Media* class by providing specific functionalities to manipulate and draw images.
- **VideoMedia**: Also extends the *Media* class, containing methods to play and manipulate videos.
- **Metadata**: A static class used primarily to manipulate and generate metadata for various items in the application.
- **metadata_struct**: Defines the structure of the metadata that each file contains.

5.2 UI Elements

The classes in this section consist of objects that the user can interact with to manipulate media objects.

- **Button**: Represents a button that can be pressed, triggering an event that the containing page can handle accordingly.
- **Carrousel**: Represents an array of media files that can be selected in a carrousel style.
- **FilterPanel**: Manages the panel with buttons for applying filters and saving changes.
- **MediaCircle**: Similar to the *Carrousel*, its main objective is to display media files in a circle, allowing the user select any of the files.

5.3 Pages

This section includes classes that represent different pages in our application.

- **CameraPage**: Captures webcam images and detects faces and objects for searching within the application.
- **HomePage**: Acts as the hub page, displaying two carousels—one for videos and another for images.
- **ImageEditor / VideoEditor**: These classes display a media item in fullscreen and use the *FilterPanel* to manipulate it. The *VideoEditor* class includes an additional button to play and pause the video.
- **ScreenSaver**: Displays the application logo initially and switches to the *HomePage* upon detecting any movement.
- **VersionControlPage**: Shows eight media files related to the previously selected file, prioritizing the versions of the given media.
- **FilteredPage**: Displays four files found through *CameraPage* search, each of which can be selected by gesture.

5.4 Other Classes

This section includes classes that do not fit into the previous categories, and are primarily used to manage the overall application.

- **ofApp**: Manages the active page, redirecting any triggered events to it, and calling its drawing and update functions.
- **screen_names**: Holds the enum with the index for each page, used in *ofApp* to change the active page.
- **main**: Sets up the application window and starts the application loop.

6 Conclusions

We believe to have implemented a very competent and accessible attempt at a collaborative design wall, having it be fully interactable through touch and gesture, without the need for physical input devices.

Our system was designed with ease-of-use and meaningful capabilities in mind.

The use of frameworks such as **OpenFrameworks** and **OpenCV** made this daunting task accomplishable, by simplifying most of the implementation process of nearly all the image processing logic.

Though we'd like to search for ways to improve our application even further, such as improving the parallelization in object recognition, the largest performance bottleneck in our application by far, for now we are very pleased with our results and trust that nearly any user can use our application at ease and enjoy its functionalities in a useful manner.

7 Annex

Note: To improve the readability of the diagram, the classes have been reduced.

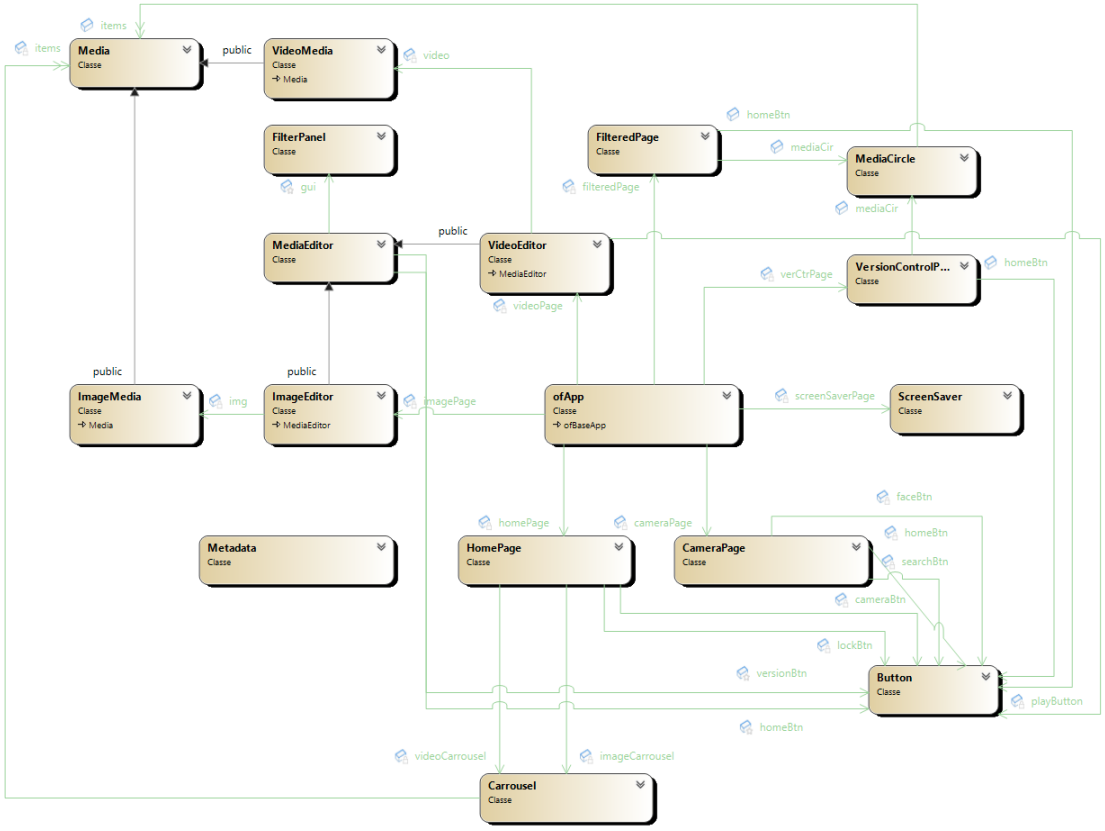

Figure 1: Simplified class diagram.

8 User Manual

8.1 Starting the Application

When the application starts, the app logo is displayed. Upon detecting any movement through the webcam, the app switches to the Home Page.

8.2 Home Page

On this page, the user is presented with a set of interactive objects:

- **Media Grid**: In the center of the screen, there are two rows of media files. The upper row contains image files, while the lower row contains video files. Users can select any file to view and manipulate by:
  1. Dragging the file to the center of the row.
  2. Clicking on the file in the center of the row, which will open the **Media Editor Page** (8.3).
- **Screen Saver Icon**: Clicking on the lock icon returns the user to the **Screen Saver Page** (8.1).
- **Camera Icon**: Clicking on the camera icon navigates to the **Camera Page** (8.5).

8.3 Media Editor

This page displays the selected media file in fullscreen, accompanied by three UI elements:

- **Home Icon**: Clicking this button returns the user to the **Home Page** (8.2).
- **Version Control Icon**: Below the **Home Icon** is another button that redirects the user to the **Version Control** page (8.4).
- **Filters Panel**: On the right side, there is a panel containing several toggle buttons, with the exception of the **Save button**:
  - **ASCII**: This toggle is available only for images and converts the image to ASCII art.
  - **Edge Filter**: This toggle highlights the edges of the media file.
  - **Inverted Colors**: This toggle inverts the colors of the media file.
  - **Blur**: This toggle applies a simple blur effect to the entire media file.
  - **Gaussian Blur**: This toggle applies a Gaussian blur to the media file.
  - **Dilate**: This toggle applies a dilation effect to the media file.
  - **Erode**: This toggle applies an erosion effect to the media file.
  - **Save**: Clicking this button generates a new version of the media file with the current active filters.

8.4 Version Control

This page displays eight media files related to the selected one in a circular shape, prioritizing its versions.

- **Home Icon**: Clicking this button returns the user to the **Home Page** (8.2).
- **Media Files**: Clicking on any of the media files will open it in the **Media Editor Page** (8.3).

8.5 Camera Page

- **Magnifying Glass Icon**: Clicking on this icon captures a frame from the webcam and it search's for related files using **SIFT**.
- **Face Detection Icon**: Enabling this button let's the application start detect the face of the users.
- **Home Icon**: Clicking this button returns the user to the **Home Page** (8.2).

8.6 Search Results

This page displays four media files in a circular shape, related to previous captured frame as the result of the search. Here the user has options to select one of the files displayed:

- The user can use touch to select one of the files, this action will open the file in the **Media Editor** (8.3).
- The user can also with his fist close and facing the camera point to the file he intents to open in the **Media Editor** (8.3).

There is also a Home button present in this page that redirects the user to the **Home Page** (8.2).

References

[1] Yujiro Okuya & Olivier Gladin & Nicolas Ladevèze & Cédric Fleury & Patrick Bourdot. "Investigating Collaborative Exploration of Design Alternatives on a Wall-Sized Display". In: (23 April 2020). URL: https://dl.acm.org/doi/10.1145/3313831.3376736.

[2] Paolo Bolettieri & Fabrizio Falchi & Claudio Gennaro & Fausto Rabitti. "Automatic metadata extraction and indexing for reusing e-learning multimedia objects". In: (28 September 2007). URL: https://dl.acm.org/doi/10.1145/1290067.1290072.

Links

Git Repository

For further clarification, a demonstration video will be made available soon. Its link may be found in the repository's README file.