# Interfacing CBIR: Designing Interactive Widgets to Query Attribute Data in Face Image Retrieval

Ted Davis

Visual Communication Institute, The Basel School of Design, HGK FHNW,
Vogelsangstrasse 15, 4058 Basel, Switzerland
theodore.davis@fhnw.ch

**Abstract.** This paper establishes a unique method in visual search for the querying of face image attribute data, through a modular interface composed of interactive widgets. These widgets enable the user to define a model result through abstracted visual representations of each portrait attribute. The combined inputs construct compound queries for comparing quantitative values. Such a technique can help bridge the semantic gap within image retrieval by avoiding the continued and prevalent reliance on keywords and text-based inputs for the description and querying of pictorial content. Rather than a graphical user interface being an afterthought to a novel image processing technique, this research utilizes existing image datasets as a future given and addresses how *content-based image retrieval* (CBIR) can advance when reconsidering the role and importance of design.

**Keywords:** content-based image retrieval, CBIR, image search, visual search, query, portrait, face, attribute, interaction, interface, design, semantic gap, graphical user interface, GUI, widget, relevance feedback.

## 1 Introduction

With an exponential growth in both the creation and sharing of digital photographs, the field of *content-based image retrieval* (CBIR) continues to emerge as a potential relief to the unsustainable model of manually tagging contextual contents. Even though significant developments have been made to automate the process of generating tags, by maintaining textual descriptions of visual media, both the stored labels and attempts to query them are limited by the semantic gap [1][2], thus requiring a correlation in the keyword annotator and its retriever. In contrast, the raw data generated through image processing contains a much wider range of gray-values and probabilities, which can be stored in nearly their original floating-point (decimal) precision format, rather than being rounded off to a finite keyword. This introduces a primary hypothesis, that *the interface of CBIR won't see innovation until numbers replace words in the storage of image attribute data.*

A. Marcus (Ed.): DUXU 2014, Part II, LNCS 8518, pp. 425–434, 2014.

### 1.1 Background

In contrast to the topic of CBIR typically being approached by the more technical field of computer vision, this problem has been investigated from the vantage point of an interface designer. Rather than producing novel algorithms, this research proposes a novel design approach for the user-intuitive interfacing of quantifiable data.

### 1.2 Related Work

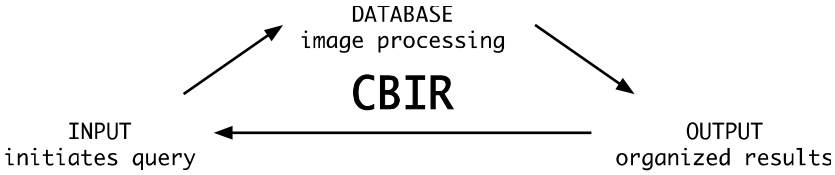


**Fig. 1.** A triangle of distinction should be made regarding three co-dependent aspects of CBIR: the *INPUT* initiates the search query; the *DATABASE* is populated through image processing; the *OUTPUT* organizes the retrieved results

The term CBIR, synonymous with *visual search,* encompasses a wide variety of definitions, all of which touch one or more of the three aspects above (Fig. 1). Following an extensive survey of CBIR implementations [3][4][5], it became clear (with the exception of *reverse image search*) that keywords and text-input query remain predominate components in the *content-based* retrieval of imagery. Whether the hindrance to get away from semantic descriptions lies in the population of a database or is out of convention from text-based html form elements remains unknown. The role of design is most commonly provoked when it comes to the output, however, the actual *visual search* has already occurred by this point. Therefore this research focuses on the visual possibilities of that initial input. The most relevant methods in this direction are described below.

**Sketch and Low-Level Widgets.** Low-level attributes, including color, texture and contour have seen the most development as interactive visual inputs, although they typically depend on text-based inputs for contextual results. One pioneering method is the query by sketch, in which the user is offered a reduced set of drawing tools to provide a depiction of the sought after imagery [6][7][8]. The largest obstacle with this technique is that the query can only be as accurate as the user's own drawing abilities.

**Reverse Image Search.** While many CBIR models focus on finding similar imagery, a niche has been developed for finding the exact same image through a visual signature. Useful for finding alternative sized copies of the same image, this technique has also been adopted for identifying image copyright infringers [9][10].

Additionally, the Internet connected camera of a user's smartphone has seen dramatic exploration in the consumer goods market [11][12]. Also using visual signatures, the resulting query is typically returned as a set of keywords, which are passed on as plain text searches.

**Attribute Extraction.** In models specific to portrait imagery, significant progress can be found through advances in face recognition and automated name tagging [13][14]. Further developments have taken automation further by automatically extracting multiple attributes from each portrait [15][16]. Nevertheless, such systems are still relying heavily on and are interfaced solely by text-based input queries to match an exact attribute label. This requires the end user to know ahead of time the exact wording that was used for each attribute, which can be slightly optimized through the use of a synonym generator.

## 2 System Overview

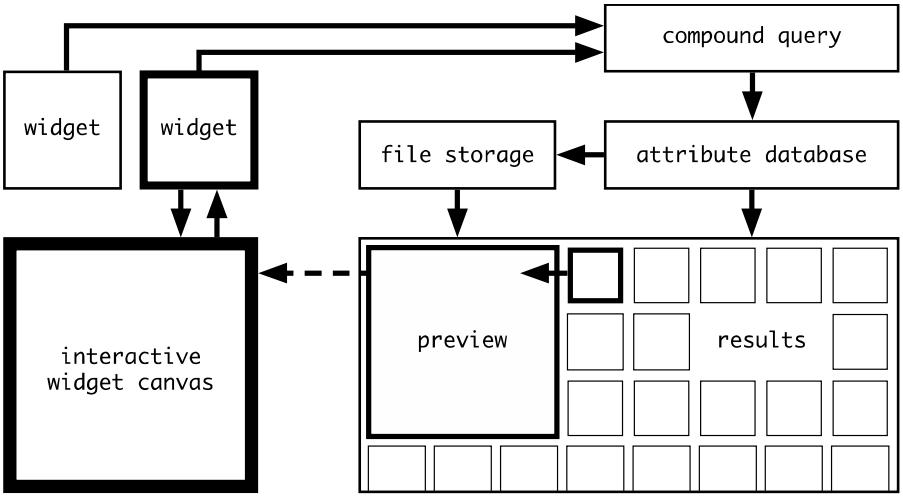


**Fig. 2.** Schematic for Retrieving Relative Attributes With Widgets (RRAWW) search system

The proposed system can be depicted with the schematic shown above (Fig. 2). Multiple *widgets* can be activated, which are singularly selected and modified using the *interactive widget canvas*. They form a *compound query,* composed of individually tallied ranks within the *attribute database*. This is averaged to form results based on matching rank. The user may choose to further adjust the widgets, or by clicking on a given result, activate an enlarged *preview*, enabling them to optionally adopt the attribute values for the single selected widget or all available widgets.

### 2.1 Storing Values

**Raw Data.** Focusing on the image processed values found in a dataset such as the *Annotated Facial Landmarks in the Wild* (AFLW) [17], one finds attributes such as image width and height, pixel coordinates of a face bounding box, width and height of the face bounding box, estimated rotation of head, among a small set of binary ones. This poses two problems that must be solved before the data can be accessed. The data is specific to one particular image, with its own width and height and therefore must be made relative to all images. Another problem is that the range of values each attribute provides is unique to that attribute. Rotation is a prime example, considering pitch, yaw and roll all have varying human restrictive ranges based on π. Therefore all raw values must be mapped to a common relative range before they can be compared.

**Mapped Data.** In order to accommodate the diverse range of each attribute, a mapping formula is used to convert all values to a new range between -1.0 and 1.0.

$$y = (x - a) * (d - c) / (b - a) + c \ . \quad (1)$$

Here *x* represents the input value, *a* and *b* are the current min and max values of the input's range, *c* and *d* are the new min and max values of the desired range.

### 2.2 Interfacing Values

With every attribute value in the database now within a range of -1.0 to 1.0 (stored as floating point numbers containing up to 8 decimal places), interactive widgets could be designed on an identical mapping. For example, the exact position of the head can be determined by setting the center of the image as coordinates [0, 0], whereas the most upper-left position would be [-1, -1] and the most bottom-right position would be [1, 1]. The interactive widget can then dynamically map the visual representation of a head position within its own frame dimensions to output an identically relative value for querying.

### 2.3 Relevance Feedback

Due to the precision of values being stored, combined with a visual interaction capable of abstracting their query, an engaging user experience of *relevance feedback* is created [18]. The user positions the model head roughly in one location and results mirroring that decision are returned. If the results are offset from the desired location, minor adjustments can be made in that direction for new more relevant results. Furthermore, as described in depth below, the user can utilize a similar search function to import a desired result's exact attribute value to the interactive widget, allowing them to bypass a guess and check game.

# 3 Interface

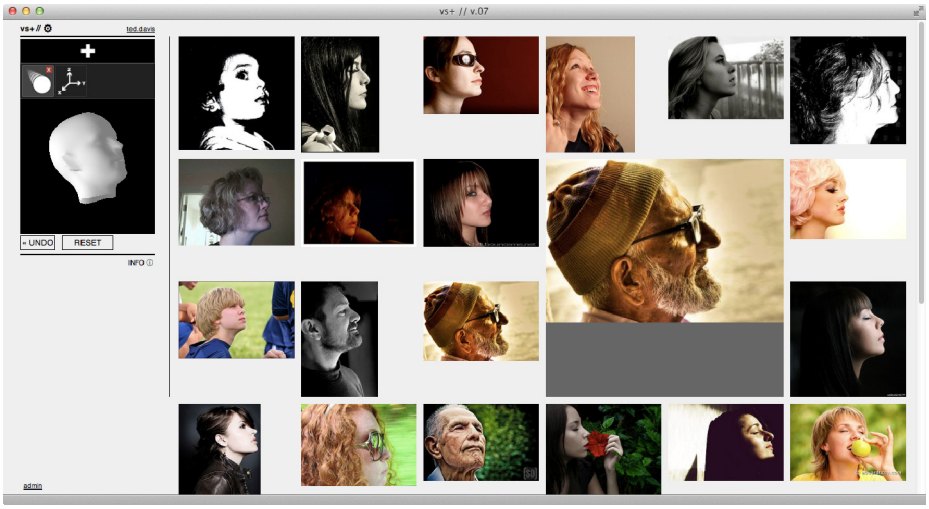


**Fig. 3.** Overview of the RRAWW search graphical user interface in a preliminary stage

Although the emphasis in this work was placed on the *input* widgets alone, it grew increasingly difficult to isolate them from the greater context in which they exist. With the realization that each widget would benefit from directly relevant feedback, being modifiable from the results, it was clear the *OUTPUT* needed equal attention in order to craft a seamless user experience.

## 3.1 Technical Implementation

**Overall.** The backend has been realized through a Linux, Apache, MySQL, PHP (LAMP) web application environment. The frontend is composed of basic HTML5, whose functionality is heavily extended through JavaScript. This includes use of the Dynamic Object Model (DOM) and Asynchronous JavaScript and XML (AJAX), allowing all queries and retrievals to be completed and displayed without refreshing the page. JavaScript objects are used extensively for storing retrieval results and widget parameters, allowing the user to easily undo and redo their compound queries.

**Widgets.** The widgets interaction has been implemented with the use of Processing.js, a derivative of Processing[1], enabling their interactions to be embedded within the HTML5 *canvas* element. Basic mouse operations of clicking and dragging manipulate a visual facade while exporting values to share with the entire system.

[1] Processing is a simplified programming environment aimed at making Java more accessible to artists and designers.

### 3.2 Navigation

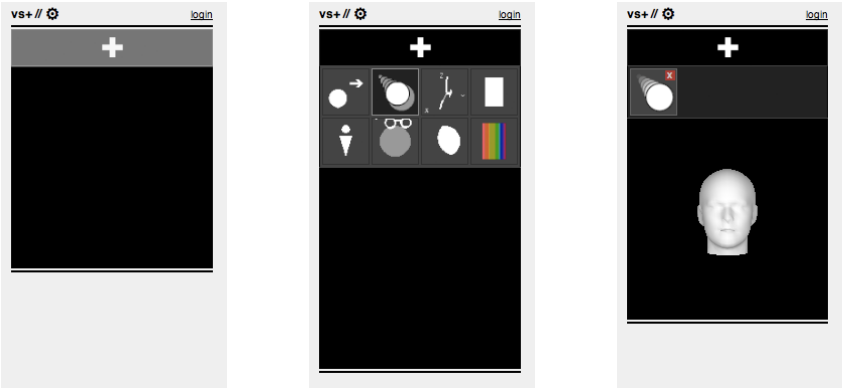


**Fig. 4.** Navigation steps for adding a widget

Upon first loading the application, the navigation has been reduced to a single pulsating button with a plus symbol placed above a blank widget canvas (Fig. 4 left). Clicking on this button reveals a set of dynamic icons (short animations) revealing the functionality of each inactive widget (Fig. 4 middle). Clicking on an icon closes the inactive widget panel, as it now appears in a new panel for active widgets on its own, while the previously blank canvas reveals a 3D model head, ready for interaction (Fig. 4 right).

### 3.3 Widgets

**Fig. 5.** Time lapse of widget interaction. *left*: position / *middle*: size / *right*: rotation

Each available attribute from a given dataset has been used to construct an interactive widget, visually representing an abstraction of the mapped values they output. Each widget interaction should capture the essence of that particular trait, while abstract enough to maintain its modeling purpose. In the case of face position, size and rotation, it became critical for their values to be shared with one another if active, in order to keep a consistent model for the query across each widget.

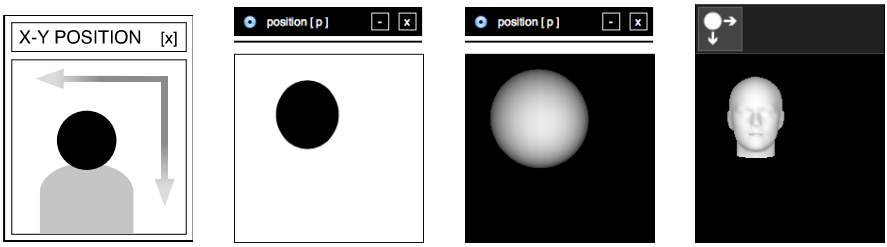


**Fig. 6.** Evolution of widget representation, starting from an initial sketch on the left

**Evolution.** The first implementation of widgets (Fig. 6) began with a basic solid ellipse to represent the face and enable the query of both relative face size and position within the image. The same program was used to generate both the images and data for 1,000 images containing random values within range. This made an initial proof of concept possible, populating a database with exact values to match the generated images, which could then be retrieved. It was a success and returned anticipated results, even given the small pool of images to search from. After a set of 5,000 generative 3D face-images were supplied by our partners, the Graphics and Vision Research Group (GRAVIS) at the University of Basel, the 2D ellipse evolved into a 3D sphere enabling the development of a widget for querying rotation data. Created using the 3D Morphable Face Model [19], this had the fortune of also generating an average head 3D object file, which was then imported and used for a more precise, albeit still abstract, representation of the desired face.

**Photos in the Wild.** Unsurprisingly both generated test image sets were problematic in their generative nature. In demonstrating these proof-of-concepts, the results were mistaken to have been created on the fly, rather than retrieved, due in part to their precise correlation to the inputs given by the user and their non-photographic nature. For that reason, the AFLW image- and dataset was implemented, a collection of over 20,000 images, extracted from Flickr, which are representative of the diverse conditions in which portrait imagery is most often found when uploaded to the Internet. Beyond the dataset having manually placed landmarks, which allow for widget implementations of face position, size and rotation, it also contains a set of binary attributes including gender, glasses, occlusion and image saturation. Although these binary qualities don't require the precision other widgets afford, they can still benefit from having their attributes trait displayed and selected in a visual way. An additional widget for image format (portrait or landscape respectively) was deducted by comparing the image width to its height. A simple attribute to consider, nevertheless it proves helpful when requiring an image for a particular usage.

**Compound Query.** For each activated widget, property settings are first used independently to find and rank the closest match for that given attribute. This is then combined and compared in an optionally weighted manner to each other, forming an average distance from the ideal match.

### 3.4 Returned Results

Sorted by their matching rank to the collective widgets, results are displayed in a modular square format grid, responsive to window size changes (Fig. 3). Although thumbnails have been created for all images, a dynamic image resizing script has also been implemented, enabling the option of rendering only the face bounding-box crop, as well as being prepared for scalable image collections.

**Endless Scroll.** An *endless scroll* was implemented by setting a default query limit of 1,000 results, which is far more than necessary. This is stored as a JavaScript object and seamlessly parsed through in groups as needed client side by the user scrolling to the last visible result. Although only a microsecond difference in initial querying time, the benefit is quickly experienced once images below the fold are requested and instantaneously displayed. Nevertheless, extensive remote server testing in an isolated condition would be necessary to confirm this speculative benefit.

**Enlarged Preview.** Dynamic image resizing was implemented for the expanded preview size image when clicking on a given thumbnail. The size is set to match a multiple of the thumbnail module size. Aware of a thumbnail's placement to the window frame, the preview is automatically offset in a direction, making sure that it is entirely visible within the window. This is necessary, because the preview has been given a fixed position, remaining static on the window while scrolling. Combined with the ability to drag it freely, a comparison between two images is made easier.

### 3.5 Similar Search

A *similar search* feature was developed which allows the user to utilize attribute values from a given result. In contrast to typical black-box surprises in how similar searches are conducted, here the user can either apply the attribute values of the resulting image to the single active widget in use or automatically enable and set all available widgets.

**Active Similar.** The active similar setting is particularly useful when attempting to set one of the gray value widgets (face position, size, rotation) of the portrait. In this case, a given example may better represent the desired setting than the user had provided. It is especially useful when attribute settings need to be maintained, which a resulting image contradicts. For example, the desired portrait must be looking to the left, however the face size of a result is closer to that in mind than had been set, even though the face was not looking far enough to the left. To accomplish this, the face size widget is set to active, and only that attribute's value is used for another query.

**All Similar.** Similar search using all widgets can be particularly useful if finding that a result further down the list is more desired than what is being retrieved in the first few. The neighboring results will immediately reflect these new settings, often displaying additional images of the same person or particular photo shoot. In some

instances, it reveals any existing duplicate images[2]. Because this similar search is composed of modular attribute widgets, a specific attribute can easily be modified or removed altogether if the results were too narrowing.

## 4 Conclusion

The proposed method of transparent similar search is made possible due to the modular qualities of this interface, reinforcing the benefits of a widget-based query. Furthermore, by maintaining attributes in a computer readable manner, with floating-point precision numbers, rather than a human readable selection of subjective semantic keywords, the data is in its ideal form for the actual retriever of all the heavy lifting. Perhaps at the apex of making the RRAWW search system [20] possible is its second 'R', for *relative*. By mapping the raw image processed data to a commensurate range, the many components of this interface could communicate with one another.

Returning to the initial hypothesis, *interface innovation of CBIR will require numbers to replace text in the storing of image attribute data* rings especially true as abstract digits are transcoded back into the visual representation from which they were first extracted.

**Further Research.** Although the design of widgets was limited to the set of attributes made available by the AFLW dataset, further processing of these images or incorporating alternative datasets, would open the opportunity for additional widgets to be designed. Such attributes could include: lighting conditions or direction, eye gaze, hair features, facial expression, mouth opening, sharpness and group portraits – to name a few.

While this research focused on datasets of face images, its methods could be implemented to include that of any imagery that can be broken down into individually extracted attributes.

**Acknowledgements.** The author would like to thank the Graphics and Vision Research Group (GRAVIS) at the University of Basel who enabled a dialog and support for interfacing existing and developing computer vision technologies. This research was supported by the Swiss National Science Foundation (136840).

## References

1. Dorai, C., Venkatesh, S.: Bridging the semantic gap with computational media aesthetics. IEEE MultiMedia 10, 15–17 (2003)
2. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence. 22, 1349–1380 (2000)

[2] Surprisingly this has resulted in the identification of over 1,000 duplicate images in the AFLW dataset. It is acknowledged that the scraping algorithm collected duplicates, however these were never identified or purged. Complete with slightly offset landmarks due to human placement, they were still identified as close matches to one another. The duplicate dataset will be provided to the AFLW team for suggested inclusion– or exclusion.

3. Jörgensen, C.: Image retrieval: theory and research. Scarecrow Press, Lanham (2003)
4. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image Retrieval: Ideas, Influences, and Trends of the New Age. ACM Comput. Surv. 40, 5:1–5:60 (2008)
5. Thomee, B., Lew, M.S.: Interactive search in image retrieval: a survey. Int. J. Multimed. Info. Retr. 1, 71–86 (2012)
6. Niblack, C.W., Barber, R., Equitz, W., Flickner, M.D., Glasman, E.H., Petkovic, D., Yanker, P., Faloutsos, C., Taubin, G.: QBIC project: querying images by content, using color, texture, and shape (1993)
7. Engel, D., Herdtweck, C., Browatzki, B., Curio, C.: Image Retrieval with Semantic Sketches. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011, Part I. LNCS, vol. 6946, pp. 412–425. Springer, Heidelberg (2011)
8. Giangreco, I., Springmann, M., Kabary, I.A., Schuldt, H.: A User Interface for Query-by-Sketch Based Image Retrieval with Color Sketches. In: Baeza-Yates, R., de Vries, A.P., Zaragoza, H., Cambazoglu, B.B., Murdock, V., Lempel, R., Silvestri, F. (eds.) ECIR 2012. LNCS, vol. 7224, pp. 571–572. Springer, Heidelberg (2012)
9. TinEye Reverse Image Search, `http://tineye.com`
10. PicScout, `http://www.picscout.com`
11. Google Goggles, `http://www.google.com/mobile/goggles`
12. CamFind, `http://camfindapp.com`
13. Picasa, `http://picasa.google.com`
14. Facebook, `https://www.facebook.com`
15. Kumar, N., Belhumeur, P.N., Nayar, S.K.: FaceTracer: A Search Engine for Large Collections of Images with Faces. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part IV. LNCS, vol. 5305, pp. 340–353. Springer, Heidelberg (2008)
16. Kumar, N., Berg, A.C., Belhumeur, P.N., Nayar, S.K.: Describable Visual Attributes for Face Verification and Image Search. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, 1962–1977 (2011)
17. Kostinger, M., Wohlhart, P., Roth, P.M., Bischof, H.: Annotated Facial Landmarks in the Wild: A large-scale, real-world database for facial landmark localization. In: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pp. 2144–2151 (2011)
18. Zhou, X.S., Huang, T.S.: Relevance feedback in image retrieval: A comprehensive review. Multimedia Systems 8, 536–544 (2003)
19. Blanz, V., Vetter, T.: A Morphable Model For The Synthesis of 3D Faces (1999)
20. RRAWW Search, `http://www.rraww.net`