SimilArt: The Design and Implementation of a Visual Exploration Tool for Artworks

Masoumeh Bakhtiariziabari, Kylian van Geijtenbeek, Barry Hendriks, Iulia Ionescu, and Martine Toering

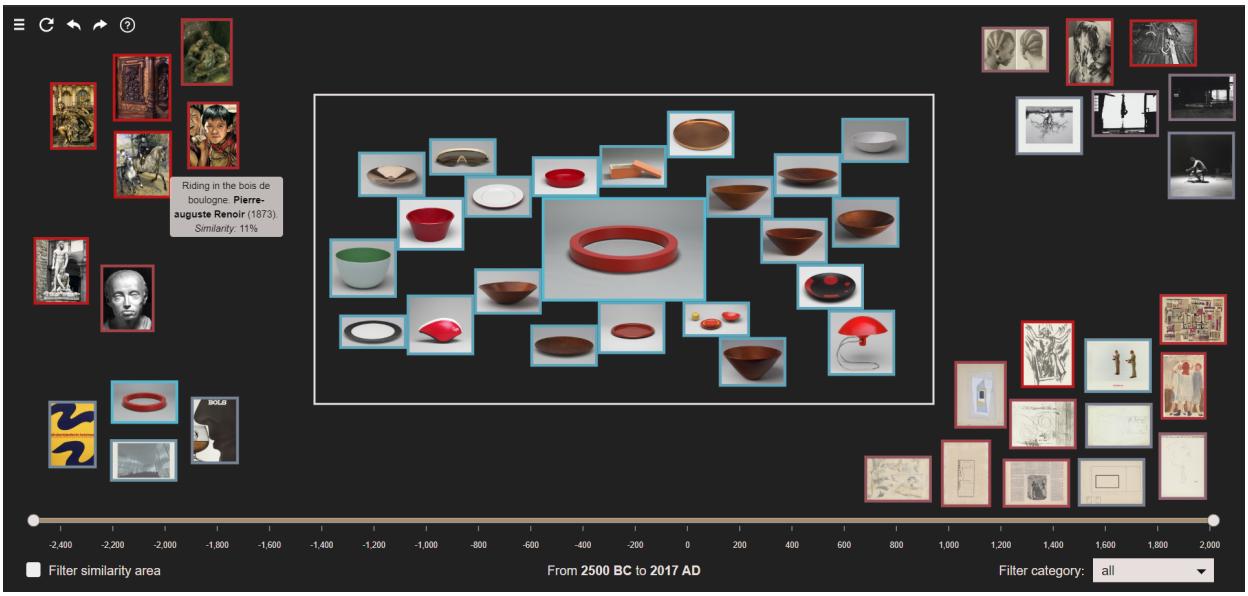

Fig. 1. **SimilArt:** a visual application that lets the user interact with different kinds of artworks and explore visual similarities. The colored borders around each image reflect the degree of visual similarity to the center image. The closer the color is to the *"Moderate Cyan"* the more similar it is perceived to the center image. Colors close to *"Boston University Red"* indicate a smaller ratio of similarity.

Abstract—Extracting and comparing visually similar artwork is an important task for studying art history, understanding its dynamic and analyzing the sources of inspiration for different artistic genres and streams. However, such an analysis would require analyzing artworks and their similarities at large scale, which is cumbersome – if not impossible – to be handled without specifically crafted automated systems. In this work, we design and implement *SimilArt*, a system that facilitates exploration and visual comparison of artworks in near real-time, using a deep-learning-based visual semantic extraction and efficient indexing that is offered via user-friendly interfaces. Our system covers functionalities such as feature-based search, filtering and visual comparison on a large dataset containing more than 150,000 artworks from numerous time periods, categories and genres.

Index Terms—Art, Artworks, Visual similarity, Exploration

1 Introduction

The OmniART dataset [12] contains over a million images of artworks by many different artists, both famous and unknown, whose works can be found from well known museums to online art forums. With so many different types of art made by a large variety of artists, orientating within the dataset and comparing individual artworks can become a frustrating and time expensive endeavor, having to filter and compare large amounts of images.

Comparing art can be done for many reasons, whether it is simply to find new art to enjoy or in order to conduct research into the similarities between great artists. However, in a large dataset like OmniART comparing art can be a difficult task. Lets say you enjoy a certain piece of art and would like to find similar artworks. There would be two main ways to find such similar art. The least time expensive way would be to 1) filter out all artworks that contain meta-data that is not similar to the meta-data of the artwork you enjoy, which would result in a set of similar artworks. A downside of this is that it might be very hard for a person to understand why a specific artworks sparks interest, resulting in a set of artworks that according to the meta-data are similar but none of which are enjoyable to you. The other way would be to 2) visually compare an overwhelming number of artworks. A large downside of this method would be the time it takes to visually compare many different artworks.

We aim to solve the problem of laborious searching in order to widely orient oneself within the world of art – from the oldest handcrafted fabrics to the most recent photographs. The objective is to provide a user with the ability to easily compare visual similarity with other artworks and focus on any domain or time period of interest. In our search for an intuitive and user-friendly tool, we aim to answer the following questions:

- How can one display artworks such that there is room for simultaneous exploration and comparison?
- How can one efficiently automate the task of finding visual (dis)similarity between artworks?
- How can one provide a user with the ability to efficiently filter or

- *Masoumeh Bakhtiariziabari, Kylian van Geijtenbeek, Barry Hendriks, Iulia Ionescu, and Martine Toering are with the University of Amsterdam. E-mail: { mbakhtiariz, kylian.vangeijtenbeek, barryhendriks98 } @gmail.com, { iulia.ionescu, martine.toering } @student.uva.nl.*

search for artworks without overwhelming the user with widgets?

This paper introduces the tool *SimilArt* that combines image features from a deep learning model in combination with a visual design, allowing users to explore similar artworks and making it easier to find new artworks or artists to enjoy. In this design we contribute:

- New and visually appealing ways to compare and explore new art.
- A combination of a deep learning model and visual design in order to visually compare art.
- An interactive design allowing users to filter on different art types, create own clusters of similar art, and search for specific art works.

2 Background

This section describes the OmniART dataset in detail after which related work in visualization is discussed.

2.1 OmniART Database

The application is based around a subset of the OmniART dataset [12]. The OmniART dataset is an artistic dataset containing 1,002,586 images of different artworks made throughout history, containing 15 different art collections, 36,599 artists, and 536 artwork types, spanning 22 periods of art. Each image contains meta-data specifying artist, artwork name, artwork type, school, creation date, and dominant color among others allowing the dataset to be filtered for specific artworks. The original dataset was created as part of research into multi-task deep learning for artistic data and has since been expanded from the original 33,418 images to the 1,002,586 image dataset it is today. The subset that is used in this paper consists of images with general type *print*, *photograph*, *sculpture*, *design*, *weapons and armor*, *drawing*, *painting*, *craft*, *installation* and *textile*. Each category was then limited to contain at most 25.000 artworks, resulting in a subset of just over 150.000 artworks.

2.2 Related Work

Previous research includes the main goal to facilitate exploration of the dataset based on color [11]. Users could explore the artworks contained in the database based on the colors used in the artworks, where additional filters, like time period, allow the user to restrict the shown artworks further.
Outside of the OmniART database research has been conducted into the visualization of similar images [10]. When browsing images sorting visually similar images close together helps in dividing the images in genres, with the downside that similar images sometimes "merge", resulting in the possibility that certain images are overlooked. Using labels or captions could reduce this effect.
Several measures for visualizing image similarity have previously been compared [9]. It was concluded that when working in a 3-dimensional space, advanced measures used for information retrieval lose their advantage. Consequently, such 3-dimensional applications perform just as well as simpler measures used in a 2-dimensional space. Therefore, one should not resort to 3-dimensional visualizations unless it adds significant value.
In previous research on multimedia analytics, multiple techniques for displaying a collection of images have been analyzed, taking image similarity into account [13]. It was concluded that for screen space efficiency, simple methods such as a basic grid view is optimal. However, when it comes to semantic navigability, clusters of (possibly overlapping) similar images allow the user to navigate through the similarity space most effectively. When wanting to sort images based on multiple information channels a spreadsheet view may instead prove most useful. Thus, a suitable presentation of images in a limited space depends on the goal of the application, the importance of similarity, and the dimensionality of the information that one wants to present the user with.

Fig. 2. **A screenshot of the loading screen**, shown to the user for the few seconds that the metadata is being loaded. During this time period images are flipped and and randomly substituted with new samples.

3 Design

In this section, we describe the design and the visual features of *SimilArt*. The focus in the design lies on user experience and story telling. One primary user goal is to enable the user to explore either based on knowledge about artworks or without any previous information beforehand. Allowing the user to be able to compare artworks based on artwork details such as artist name, artwork type, year and visual similarity is another objective. Lastly, our visualization tool could help in providing the user with a sense of the diversity and distribution in artworks present in the dataset. The main interface is shown in figure 1. Below, the choices made in the design are outlined.

3.1 Loading screen

Upon starting the application the artwork metadata has to be loaded. Although generally this loading takes no more than a few seconds, to prevent the user from being presented with a blank screen, an aesthetically pleasing loading screen entertains the user during startup. The loading screen presents a grid of randomly selected artworks from the dataset, with the middle cut out to present a clean display of the application title, as seen in figure 2. Independent of each other, each image flips over vertically (i.e. around its x-axis) in-place at random intervals to reveal a new image.

The loading screen and its flipping images serves three purposes. First of all, the user is presented with the title of the application. Secondly, as mentioned before, it engages the user while loading the required data in the background. Lastly, the loading page provides a first exposure of the available artworks to the user. This way the user is already being presented an opportunity to get a first impression of what artworks to expect.

3.2 Artwork area

The largest part of the application is occupied with an interface that shows artworks as these are central in our design. This interface is divided into two parts. The middle part of the application, surrounded by a border, contains visually similar artworks to a larger central image. The center image is the current selected artwork and can be updated by clicking any other artwork. Clicking the center image enlarges the center image. The surrounding area is the exploration area which contains artworks regardless of similarity. The purpose of this area is for the user to have a wide range of selection for viewing artworks or comparing artworks on visual similarity. The artworks in this area are draggable with the intent of enabling the user to make clusters of artworks. With this the user can compare notion of similarity with that of the image features from the network. For the size of the artworks a trade-off is present between resolution and sample number. We decided on slightly smaller images in favor of range and diversity. To account for the drawback of smaller images every image is clickable with Ctrl + Click to expand the image and show it in a larger overlay. The artwork area is sparsely filled with samples to avoid clutter. The top of the artwork area consists of five buttons for the menu: randomization of the exploration area, undo, redo and help button respectively. The menu

contains a search function, a section about the application and a contact section.

3.3 Search and filters

Other purposes of an artwork visualization tool can be for the user to compare and find artworks in a filtered selection as well as search for an artwork. For the filter we gave the user the option to filter by time (year of creation of the artwork), filter on category, and whether or not to include the similarity area in the filter. By default, the interface shows artworks from all years ranging from 2500 BC to 2017 AD. To filter the creation year we placed a moderate timeline in the tool that is fixed in position at the bottom. The current time selection is shown below the timeline so the user can immediately identify the applied selection. This same area contains the two other options for filtering and their current selection. The search feature consist of search based on artwork name as well as artist name. The search results are shown immediately below the search field. We decided for the search results to occupy the screen less and showing only one result at the time to avoid clutter. Figures 3 and 4 demonstrate the functionality of search and filter category buttons. A general overview of the filtering section can be found at the bottom of figure 1.

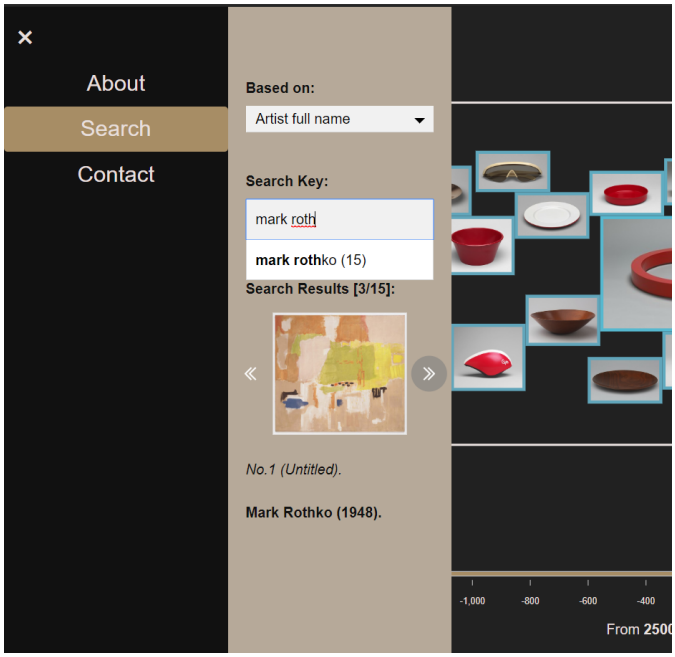

Fig. 3. **An overview of the search section**: search can be based on *artwork name* or *artist full name*, by typing the name in the *search key* field. There is an auto-complete option for suggesting the user the available options based on typed characters. The results of search are shown in the *search results* part, which can be traversed using left and right arrows. At the bottom of search result the *artwork's name*, *artist's full name* and *creation year* of it are illustrated.

3.4 Theme and colors

SimilArt uses a simple dark gray background color in order to reduce luminance and associated eye strain. Dark gray is chosen over black as dark gray is able to express a wider range of colors. A color that is closest to *sandal* or *desert sand* is used as accent color. This accent color is used in the timeline and in the menu, see figure 3. The accent color provides emphasis and contrast of these parts of the application but is used sparingly to ensure a clean look. The color provides a warm and light contrast to the background. The color relates to the topic of artwork of all times, taking the user back to ancient periods while still being a timeless color.

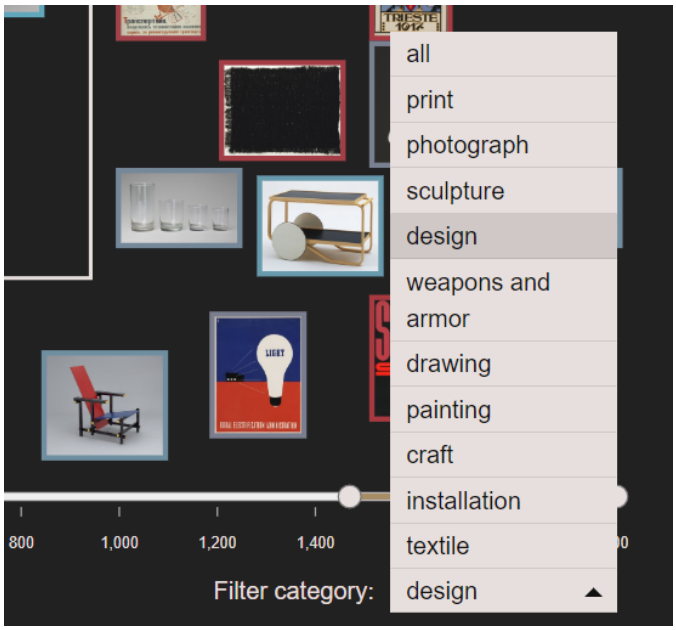

Fig. 4. **Category-based filtering**: By clicking on the dropup menu at the bottom right side of page, ten general types of categories appear for the user to select for filtering out the results of exploring area or both parts (exploring and similarity zones). Filtering on similarity area happens only when the checkbox at the left-bottom of the page ("*Filter similarity area*") is checked.

Similarity color spectrum To supplement the feature of having a similarity score associated with each artwork additional color was added that more clearly transfers similarity to the user. A supporting color spectrum is added to borders around images. For this color spectrum the blue palette is most suitable as this is a complementary color of the sandal accent color. Due to our dark and warm background color *moderate cyan* ensures an appropriate illumination contrast to the tool. A color spectrum is used ranging from *moderate cyan* for the maximum score to *Boston university red* for the minimum similarity score. The colors are complementary colors on the color wheel and provide our application with a vibrant feel. The gradient between these two colors are visually depicted in the *Info* part of our page (Figure 5) that is accessible by clicking the question mark icon at the top-left of the application (look through top left of figure 1).

Buttons The appearance of the buttons were modeled after the ion icons pack, an icon pack used in many websites and IOS, Android and desktop apps. Using these icons should give users a sense of familiarity and intuitive knowledge about what each button represents. The color of the buttons is a simple white, which gives the buttons some contrast to the background whilst not taking too much attention away from the artworks. When hovering over a button with the cursor the area around the button becomes lighter in a circle. This is a common approach in applications or browsers like Firefox or Google Chrome to indicate that the button is able to be pressed.

3.5 Utility buttons

To improve the user experience, situated at the top of the page we have included the following utilities accessible through buttons:

Randomization Refreshes exploration area with a new set of randomly selected images, following the requested filtering criteria.

Undo and Redo These two button are being used to track, save and revert changes related to the central image. More details about these two are elaborated in section 4.4.

Info illustrates useful information including pointers to the exploration area, artworks with visual similarity, the center artwork as well as the similarity visualization and filtering. Figure 5 represents the information being illustrated after pushing *Info* button (with the question mark symbol).

3.6 Written information

Meta data The information from the meta data provided by the OmniART database was raw and therefore not very consistent in style. We added minimal text preprocessing to ensure the data looked consistent for all artworks (e.g. names starting with capital letters, removing unnecessary spaces, etc.).

Tooltips When hovering over the images, a small summary of the artworks is given by displaying the name of the artwork, the artist and the corresponding year (see figure 6). The way this is visualized is based on the standard museum information sign. The similarity with the central artwork is given as a percentage between 0 and 100, which we believe gives an intuitive idea to the user regarding the similarity. The different parts of the text are executed in different styles (regular, bold, italics) to ensure that the text is readable and easily understandable without the need for a large tooltip box. The tooltips appear with a delay to ensure the users intent of seeking information.

Pop up screen central image The pop up screen displays the same information about the artwork, artist, and year as the tooltips, in a similar fashion. On top of this information, the general and artwork type are shown, together with the dominant color of the image, in a smaller font size to not distract from the main information (figure 7).

Loading If it takes a relatively long time to filter the images the user is interested in, the text "Loading..." appears to communicate to the user that the application is still responsive.

4 Implementation and features

SimilArt is a browser-based visualization application built with Javascript and d3.js [3]. During development, the application was run using a locally hosted server through the Python-based *Flask* library [1]. Any visual similarity calculations were done in Python, using Flask's TCP link to transfer data between the Python and Javascript functions. Development and testing has primarily focused on the *Firefox* and *Google Chrome* browsers. The rest of this section describes any further technical implementation details regarding *SimilArt*.

4.1 Visual Similarity

The goal of *SimilArt* is to compare artworks based on visual similarity such as color and shape. To obtain a similarity measure we vectorize each image by extracting features using a Convolutional Neural Network in Python. For the purpose of discovering visual similarity, the aim is not to pick the most complex network with the highest classification accuracy on popular image datasets such as ImageNet [4]. Instead we found that using the first few layers of a small ResNet-18 [6] model in PyTorch [8] pretrained on ImageNet works very well in practice. The first three convolutional blocks from ResNet-18 are used, resulting in 128 features of 28×28 pixels each per image. By average pooling each 28×28 feature down to 1 number, the output is reduced to a single 128-dimensional vector per image.

Once a feature vector has been extracted from each image in the dataset, similarity between artworks is measured by the Euclidean distance between their feature vectors. For any given artwork, the most similar artworks are retrieved using a k-nearest neighbor approach, using kd-tree [2] and ball-tree [7] data structures for supporting fast indexing. To provide each neighbor with an intuitive similarity score $s \in [0,1]$ we calculate the similarity score using a scaled cosine similarity:

$$s(\mathbf{x},\mathbf{y}) = max\left\{0,\ 1-\left(4\cdot\left(1-\frac{\mathbf{x}\mathbf{y}^T}{||\mathbf{x}||\ ||\mathbf{y}||}\right)\right)\right\} \quad (1)$$

Algorithm 1: Placement of images in similarity area

Arguments:
A: area in which the images should be placed
C: center image
I: list of images, sorted from most to least similar to center image
$borders \leftarrow getBorders(C)$;
$direction \leftarrow up$;
$numFailures \leftarrow 0$;
for $img \in I$ **do**
  $dirBorders \leftarrow getDirectionBorders(direction)$;
  $candidateLocs \leftarrow getCandidates(dirBorders, img)$;
  if $length(candidateLocs) = 0$ **then**
    $direction \leftarrow changeDirection(direction)$;
    $numFailures \leftarrow numFailures + 1$;
    if $numFailures \geq 8$ **then return**;
  else
    $bestLoc \leftarrow getClosestCandidate(C, candidateLocs)$;
    $placeImage(img, bestLoc)$;
    if *imgExtendsBounds(img, borders) or areaBordersReached(A, borders)* **then**
      $direction \leftarrow changeDirection(direction)$;
    $borders \leftarrow addBorders(img, bestLoc)$;
    $numFailures \leftarrow 0$;
  end
end

The reason for this re-scaling is the tendency of cosine similarities to be relatively high for most artwork pairs, providing little intuitive feedback on relative similarity between artworks. Instead, equation 1 magnifies the dissimilarity between artworks by a factor 4.

4.2 Image placement

In general, the goal for image placement is to have every image fully visible upon initial presentation, i.e. no images overlap each other. Yet, the desired placement of images differs a lot between the similarity area and the exploration area.

4.2.1 Similarity area

For the similarity area, one would like to pack images relatively close together while preventing a very static appearance by placing all images in a grid. To portray each image as truthfully as possible, one would like to maintain the correct aspect ratio of the image. For this purpose, each image is scaled such that the longest side of the image is of fixed length.

To create a visually interesting layout of images, we designed a custom algorithm inspired by the widely popular tag cloud generator Wordle [5]. In this custom algorithm, we keep track of the top, bottom, left and right borders of each image that has been placed thus far. Given a list of images sorted from most similar to least similar, the images are placed sequentially in a greedy fashion, changing placement direction between *up*, *right*, *down*, *left* (in this specific order, i.e. a clockwise spiral). When going in any direction, the candidate locations are searched in such a way that the new image would have no overlap with any existing image, yet connects to an existing image's border (top border for going up, right border for going left, bottom border for going down, left border for going right). One candidate location is calculated per existing border. Some randomness is involved in selecting a candidate location, yet such that the two images would still touch. If a candidate location is within close enough proximity of any outer border (i.e. the borders of the similarity area) or exceeds it, with 0.5 probability the candidate location is changed such that it connects to this outer border if possible.

Next, given all candidate locations, the image is placed at the candidate location where the center of the new image would be closest

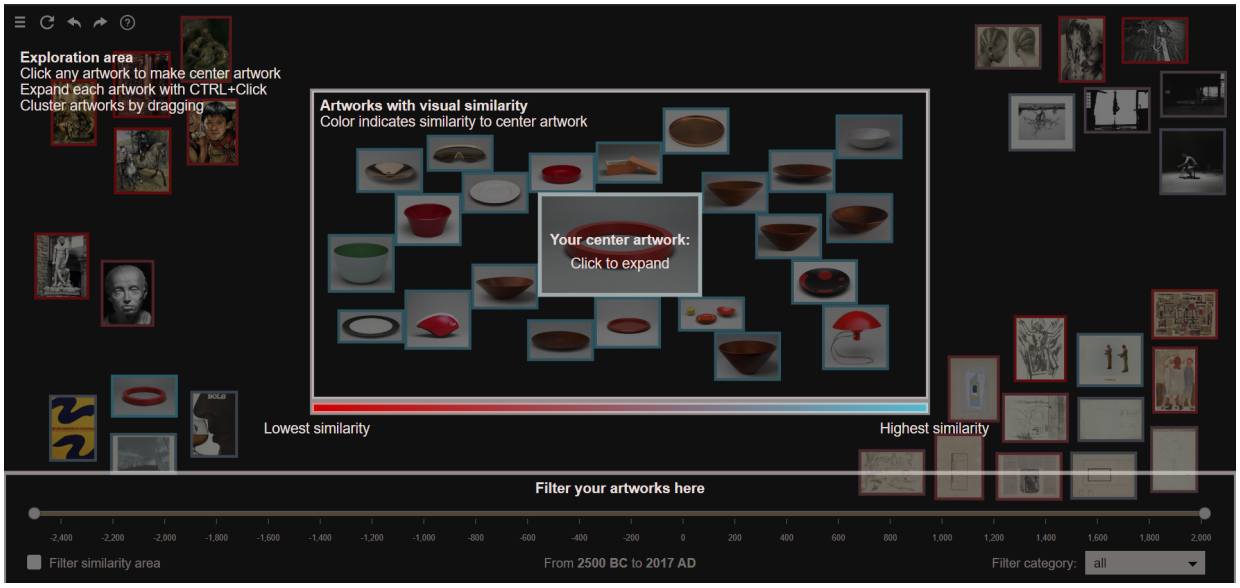

Fig. 5. **Info**: A snapshot of the application after pressing the Info button (question mark icon at the top left of the page). The colorful gradient from red to cyan depicts the colors being used for 0% to 100% similarity rate.

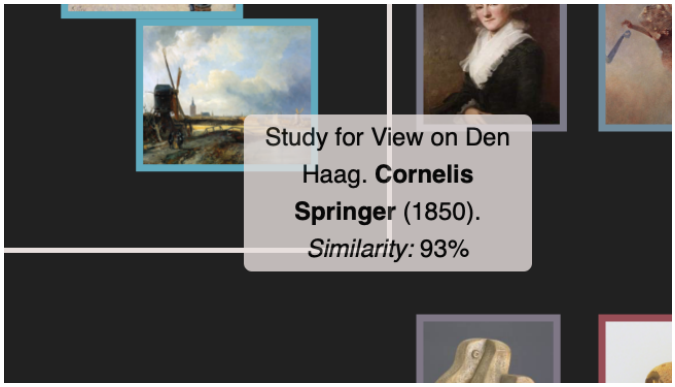

Fig. 6. **Tooltip**. By hovering over artwork, its name, artist's full name and creation year are illustrated.

Fig. 7. **The extended version of an artwork**. When the user (ctrl +) clicks an artwork, the full-resolution photo from the original online url of the corresponding image is loaded in a *popup* window. Next to the enlarged photo the following metadata information are displayed respectively: a) Artwork's name, b) Artist's full name, c) Creation year, d) General type, e) Artwork type and f) Perceived dominant color.

to the center image's center. Consider a tight bounding box $\mathscr{B}$ containing all placed images thus far. Given $\mathscr{B}$, placement direction is not switched until either 1) no valid location was found, 2) the newly placed image extends $\mathscr{B}$ in the current direction of placement, or 3) any image has previously been placed against the respective border of the similarity area that limits how far up/right/down/left we are allowed to place images. If the image cannot be placed successfully for two full passes through the up/right/down/left directions (i.e. eight consecutive direction changes), the algorithm terminates. Algorithm 1 provides a simple high-level description of this procedure.

4.2.2 Exploration area

The exploration area consists of random artworks apart from the ones visualized in the similarity area, after the filters are applied. To enhance this feeling of randomness in the artworks, we also chose for random, non-overlapping positioning of the images inside a grid. The choice was made to display a limited amount of images to keep the main focus on the similarity area. Showing less images leads to a lower cognitive effort needed from the user to compare all the images on the screen.

In case the user is interested in grouping different artworks from the exploration area or simply changing their location for other visual or preference reasons, the area allows for the artworks to change their location by dragging them with the mouse.

4.3 Filter

We enable the user to filter art work based on art creation time and general art type. Both filters are applicable on the dissimilar set of images as well as the visually similar set. A challenge in the implementation of the latter is to ensure the agility of recovering the filtered set of similar images in the vast majority of the queries while ensuring non-empty responses in cases of scarcity of visually similar artwork that satisfy the search criteria. To achieve this, we use an attractively growing query size approach, where we initially query k-nearest neighbors with small k values followed by the desired filtering; the query size gradually grows only if the filtering process does not leave enough samples. Algorithm 2 specifies this, where b is set to 5, guaranteeing no more than five queries in total, given the dataset size.

4.4 Undo and redo

The undo and redo buttons are implemented using two stacks. One stack maintains the state for undo and another for redo. For any forward action that potentially changes the state of the environment,

Algorithm 2: Iterative filtered nearest neighbor mining

Arguments:
X: dataset of samples
k_{query}: size of filtered nearest neighbors to be recovered
b: basis for the exponential growth of the query size
$i \leftarrow 0$;
while $n_{found} \geq k_{query}$ *and* $k_{current} \leq size(X)$ **do**
    $k_{current} \leftarrow k_{query} \times b^i$;
    $X_i \leftarrow kNearestNeighbors(X, k_{current})$;
    $n_{found} \leftarrow size(filter(X_i, criteria))$;
    $i \leftarrow i + 1$
end

e.g. search, filter, click, etc., first pushes the old state to the undo stack and then the new state is stored. The undo button pushes the current state onto the redo stack and retrieves the previous state by popping the most recent item from the undo stack. The redo action acts the other way around; first it pushes the current state onto the undo stack and then pops the most recent item from the redo stack to set it as the current state. The saved state of the environment consist of the image at the center, the corresponding filtered similar images and the corresponding locations they are illustrated at.

4.5 Search

In order to efficiently search for a specific artist or artwork a user-friendly search bar is implemented and placed in the expandable side menu. The user can, through a selection window, select to search for either *Art name* or *Artist full name*. The search bar is equipped with an autocompletion mechanism that, given the user's search query, finds any artwork names or artist names – depending on which type is being searched for – that start with the user's input. The list of autocomplete suggestions is created by simply searching through the list of possible artwork names or artist names, and comparing if they start with the same alphanumeric sequence regardless of casing. To quickly convince the user of each suggestion's relevance, the overlap between the user's query and the autocompletion suggestion is highlighted by displaying it in boldface. To provide the user with immediate feedback on how many search results a query would provide, the autocompletion shows in parentheses the number of results associated with the autocompletion suggestion, as seen in figure 3. A search query is being executed by either 1) clicking an item in the list of autocompletion suggestions, or 2) pressing the *enter* button while the search input field is in focus.

5 Discussion and Future work

In this section, we revisit our original posed questions, discuss insights on visualization gained and propose future work.

5.1 Reflection

In designing visualization tools the main aim is achieving the user goals and improving the user experience. *SimilArt* aims for the user to simultaneously compare artworks as well as explore artworks. We share the view that exploration is possible to a certain extent as the number of samples shown at one moment in time remains limited. For packing images in the similarity area in a visually interesting way we have introduced a fast algorithm. While providing good results in practice, it must be noted that this algorithm is highly non-deterministic due the randomness in finding valid candidate locations. The number of images that it is able to fit in the same space may differ significantly. The algorithm also partly promotes artworks with higher similarity to the center artwork to be placed closer to this center artwork. There is no hard guarantee that all images closer to the center image have a higher similarity than images further away, yet in practice this is generally true. To have this guarantee without drastic loss of valuable space on the canvas, one would have to accept slower performance and either increase the search space or use more complex force-based models.

One of the tasks is for comparison of artworks on visual similarity to be automated. *SimilArt* achieves this by employing a small neural network to extract features from each image and compare images by their feature vectors. This approach yielded satisfying results in practice, clearly focussing on both color and shape. However, the network was pretrained on ImageNet, a large *general* image database. Consequently, the network could distinguish well between colors, shapes and perform object recognition in general but may not be able to distinguish effectively between characteristics such as artistic style. For this, a network trained on an art dataset specifically may provide more appropriate results when it comes to comparison of artworks. One downside of using a neural network is that it may sometimes focus a great deal on background color of images of 3-dimensional objects such as textile and pottery, rather than focus on the artwork itself.

Another purpose of *SimilArt* is to allow the user to effectively filter and search in such a large collection of artworks while providing high quality user experience. By limiting the widgets to effective and easy to use filters and a search bar, users are able to narrow down the shown artworks. The filters allow users to select a time period and artwork type, whilst search function shows suggested artworks or artist based on your query, making it easier to find the intended artwork. A downside of limiting the widgets to only these three options is that while it does not clutter the design with widgets, it also severely limits how the art works can be filtered. Since *SimilArt* focuses on visual similarity, not having filters for visual attributes (e.g. dominant color) can result in less effective filtering.

5.2 Future work

Our application helps to find visual similarity of images of an enormous dataset yet there are other aspects that are interesting but due to time limitations are not covered in this work. At the current phase of the application, we compute image similarities based on some fixed regulations (scaled cosine similarity) and the neighbor indexing and search is based on features extracted from a pretrained ResNet-18 model. However, we may be unaware of the characteristics of an image that trigger a higher similarity score to the query image. For future work one could train a network to predict similarities of finer granularity, e.g. pixel-wise similarity heat-maps between images. Subsequently, the user could toggle a heat-map overlay to illustrate the regions of an image that contribute most or least to the similarity score. By adding this feature users can gain some insights about what causes two images look similar; color, illumination, texture, shape or other aspects of an artwork. Another interesting possibility is to expose the user with such higher level pair-wise similarity categories directly predicted by a trained model.

The current application allows for comparing a central artwork with visually similar artworks and randomly generated images. While filtering and shuffling in theory allows for almost any artwork to be displayed among the exploratory artworks, this would require the user to either be very fortunate during repeated shuffling, or narrowly filter for the specific category and period of creation, which the user would first have to search for. Instead, a possible addition to the application would be to allow the user to somehow compare two specific artworks of choice. This could be done by, for example, having more manual control over the exploration area, allowing for addition of exploration artworks through search.

Another topic that could be interesting is finding the influence of artists on each other's work. Being able to retrieve images similar to each artwork, we can measure the originality of an artwork by analyzing the percentage of similar arts that were created in the years before and after that specific artwork. By showing these values to users they can get a feeling if these artworks were influenced by other works or the artist was the actual influencer of that specific domain of art.

6 Conclusion

We have designed and developed *SimilArt*, an easy to use tool for exploring the *OmniArt* dataset and finding visually similar artworks to a selected central artwork. By dedicating most screen space to the display of the artworks, we allow for plenty of room to display a

central artwork along with visually similar artworks in a center region, while leaving enough room to explore the dataset and drag artworks around to manually form clusters. A carefully selected color scheme gives the application a pleasant and calming aesthetic, while allowing for effective expression of each artwork and its colors. Moreover, we conclude that small pretrained neural networks allow for effective visual similarity search, albeit lacking in detection of more subtle artistic characteristics. Lastly, we find that simple yet comprehensive filters suffice in filtering an art dataset for someone with the goal of exploration and comparison.

REFERENCES

[1] Flask. `https://palletsprojects.com/p/flask/`. Accessed: 2020-03-19.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[3] M. Bostock. D3.js - data-driven documents.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] J. Feinberg. Wordle. In J. Steele and N. Iliinsky, eds., *Beautiful Visualization: Looking at Data through the Eyes of Experts*, chap. 3, pp. 37–58. O'Reilly Media, Inc., 1st ed., 2010.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[7] S. M. Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

[9] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. A comparison of measures for visualising image similarity. In *The challenge of image retrieval*, 2000.

[10] K. Rodden, W. Basalaj, D. Sinclair, and K. Wood. Does organisation by similarity assist image browsing? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 190–197, 2001.

[11] G. Strezoski, I. Groenen, J. Besenbruch, and M. Worring. Artsight: an artistic data exploration engine. In *Proceedings of the 26th ACM international conference on Multimedia*, pp. 1240–1241, 2018.

[12] G. Strezoski and M. Worring. Omniart: Multi-task deep learning for artistic data analysis. *arXiv preprint arXiv:1708.00684*, 2017.

[13] J. Zahálka and M. Worring. Towards interactive, intelligent, and integrated multimedia analytics. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 3–12, Oct 2014. doi: 10.1109/VAST.2014.7042476