Large-scale interactive retrieval in art collections using multi-style feature aggregation

Nikolai Ufer [1*], Max Simon [1], Sabine Lang [1], Bjoern Ommer [1*],

1 Heidelberg Collaboratory for Image Processing (HCI), Heidelberg University, Heidelberg, Germany

* nikolai.ufer@iwr.uni-heidelberg.de; ommer@uni-heidelberg.de

Abstract

The search for motifs and objects is essential to art history as it helps to understand individual artworks and analyze relations between them. Digitization has produced extensive art collections that hold much information and promise new knowledge for art history. However, in order to extract this knowledge, scholars need sufficient tools to analyze them. This article presents a novel visual search algorithm and user interface to support art historians in analyzing extensive digital art collections. Computer vision has developed efficient search methods for photos. However, applied to artworks they show severe deficiencies due to massive domain shifts induced by differences in styles and techniques. We present a novel image representation based on a multi-style feature aggregation, which reduces the domain gap and improves retrieval results without additional supervision. Furthermore, we introduce a voting-based retrieval system with efficient approximate nearest-neighbor search [1], which enables finding and localizing small motifs within an extensive image collection in seconds. Our approach significantly improves state-of-the-art in terms of accuracy and search time on various datasets and applies to large and inhomogeneous collections. Besides the search algorithm, we developed a user interface that allows art historians to apply our algorithm directly. The interface enables users to search for single regions, constellations of multiple regions, and integrates an interactive feedback system. With our methodological contribution and easy-to-use user interface, this work manifests significant progress towards the machine-supported analysis of visual art.

Introduction

A central task of art history is to analyze the similarities between artworks to study reception processes, morphological variations of motifs or contextual changes in artworks, thus gaining more insight into artistic relationships. To study these visual similarities, art historians have to find the connection between specific image regions containing motifs or meaningful objects [2, 3].This is particularly important for the study of iconographic questions, where researchers aim to determine and interpret specific themes and pictorial symbols [4]. In Fig 1, we illustrate how different artists painted the motif of a skull throughout time but in varying contexts.

In recent years, archives, art institutions, and researchers have created large digitized art collections [5–12]. This offers new possibilities and allows art historians to apply their analyses to new and much larger collections to gain new insights. However, traditional manual methods, where art historians examine, sort, and compare paintings

Fig 1. Example of contextual variations of a motif. Artists have painted the skull in different epochs, genres, and techniques. It appears, for example, in Hans Holbein's The Ambassadors, Caravaggio's painting of Saint Jerome, and still lives by Pieter Claesz or Modern artist Paul Cézanne. While it is always the same object, its symbolic function, and meaning intended by the artist varies. For example, in Christian iconography, the skull relates to Saint Jerome, but more often, it refers to the concept of 'memento mori'– a reminder of human mortality. Finding and tracking such motifs through time and space helps art historians to identify relations between artworks, the meaning, and popularity of motifs, or how they have been adapted and sometimes altered in form and content.

by hand, are no longer feasible since this would take weeks or even months for large collections. At this point, computer-assisted systems can dramatically accelerate and simplify this work. However, current computer-guided approaches [9–12] are based on a simple text search through metadata. They allow a fast keyword search but require the collection of metadata, which is a costly procedure itself and requires expert knowledge. Furthermore, text-based searches cannot capture the visual variety and characteristics of paintings in which art historians are particularly interested in. Therefore, there is an urgent need for efficient algorithms capable of searching for visual similarities in extensive and diverse art collections based on user-defined queries. In the following, we tackle this problem and present an efficient search algorithm and an easy-to-use visual search interface that allows users to select image regions containing specific motifs or objects and find similar regions in an extensive collection of visual art.

Visual search systems consist of two main components. The first is the image representation with numerical feature descriptors, which is fundamental for the actual search and significantly influences the search results. The second is the search algorithm itself, which should find all relevant instances as fast and as precisely as possible. Although computer vision developed very efficient image representations for photos [13, 14], they show severe deficiencies for artworks. Digitized art collections pose new challenges due to various and unique motifs painted in different artistic styles and techniques. Hence, previous image representations have to cope with a domain shift from photos to artworks and within the art collection itself. Therefore, specifically tailored feature descriptors for the arts are necessary [15, 16]. These descriptors should be highly discriminative to find corresponding image regions and invariant to typical style variations at the same time. Supervised approaches tried to learn such descriptors based on image pairs annotated by human experts. However, this requires labeling thousands of images and is very time-consuming [17]. Furthermore, the learned representation is only suitable for very similar datasets. Alternative methods follow a self-supervised approach using spatial consistency to find corresponding image regions for training [16]. This approach shows promising results and avoids the need for expert annotations. However, it does not apply to datasets with many images without repeating structures, which is the rule rather than the exception in practice. Furthermore, the search for repetitions in large datasets is infeasible since it requires a pairwise comparison of a large fraction of all image pairs. In this article, we circumvent these issues and present a multi-style feature aggregation that improves feature descriptors color and style-invariance without the need for labeled data or curated image collections. For this, we use a generic on ImageNet [18] pre-trained feature extractor and current style transfer models. Given an image, we transfer it into a fixed set of different styles, extract their features, and aggregate them to obtain a single image representation. Since we average over the same styles for all images within the dataset, we reduce the domain gap which significantly improves retrieval results across artworks.

For practical users, besides the existence of efficient search algorithms, it is equally

important that they are accessible in some form of user interface. Both the commercial [19–21] and the academic [17, 22] side have made great efforts to develop such systems. On the commercial side, image search is not new, and large technology companies such as Google and Microsoft made this feature already available in their search engines. However, their algorithms are built for natural photos and cannot handle color and stylistic variations when searching across art collections. Furthermore, they allow only a holistic image search [19–21] and users have to publish their images to search through their own image collections [19, 20]. On the academic side, there exist several search interfaces specifically designed for the arts [17, 22–24]. However, for most of them, the search is either restricted to holistic images [22–24], which makes it impossible to search for specific motifs or objects. Or they are designed for a specific dataset [17], which heavily limits their usability. In collaboration with art historians, we developed a user interface which was designed as an analytical tool for art history. In this interface, the user can upload their own datasets, select an image, and search for one or multiple regions across the whole dataset using the presented search algorithm. This enables art historians not only to find similar regions but also to follow compositions or motifs characterized by multiple parts, which is essential for iconographic questions. In addition, we integrated various small features that add to the usability of the interface with the aim to simplify the workflow of art historians.

Contributions. This article presents a novel visual search algorithm and user interface to find semantically similar regions in an extensive art collection. It allows art historians to identify relations between artists and study motifs and topics in art history. Our main contributions are fourfold. (1) We present a novel feature aggregation strategy for the visual search in art collections, which successfully reduces the domain gap and improves overall retrieval results. (2) The introduced iterative voting combined with efficient approximate nearest-neighbor search enables finding and localizing small motifs in an extensive dataset within seconds. (3) We demonstrate that the proposed method significantly outperforms the state-of-the-art in terms of retrieval time and accuracy for the visual search in art collections. (4) Besides our search algorithm, we present a visual search interface that allows art historians to use our algorithm in practice. In addition to searching for single regions, it also allows the search for constellations of multiple regions and contains an interactive feedback system to refine search results further. The visual search interface is accessible on our project website `https://compvis.github.io/visual-search`. Therefore, with our easy-to-use visual search interface and methodological contribution, this work marks significant progress towards the machine-supported analysis of art.

Related work

In the following, we present most important research related to our work and highlight our contribution to it.

Computer vision in the arts

For several years, there has been a fruitful exchange between computer vision and the art community. This exchange ranges from the analysis of artworks with computer vision techniques [25–27] to the development of new algorithms inspired by new questions [28–31] to generative algorithms that transform photos into different artistic styles [32–34] or directly try to create new art [35, 36]. Applying computer vision techniques to analyze artwork is natural as the underlying questions are similar, and visual art is only one specific but very diverse and challenging domain for visual perception. In this spirit, researchers have successfully transferred object recognition

and classification algorithms from photographs to artworks. Here, the goal was to identify gestures, persons, iconographic elements, or everyday objects in paintings [27, 28, 37–40] or classify paintings according to their style, genre, or artist [8, 41–44]. In particular, art historians are interested in analyzing artworks and in studying relationships between them. To simplify this task, some works proposed algorithms to find visual relationships within art collections automatically [16, 45, 46]. However, these approaches are very time-consuming since they compare all image pairs within the dataset and are limited to small collections. Our goal is to help art historians to find visual connections within digitized art collections, but we focus on a large-scale search with a user-specified visual query. Art historians can find and investigate visual relationships faster and more targeted with such an visual search system compared to fully automatic systems.

Visual instance retrieval

Visual instance retrieval is a well-established research field in computer vision and deals with the task, given a query image or region to find and localize corresponding regions in other images of the dataset. Computer vision successfully developed classical [25, 47] and deep learning-based approaches [48–50] to solve this task. Compared to object detection with successful algorithms like FastRCNN [51] and YOLO [52], the particular challenge is that there are neither predefined object categories nor a large amount of training data. The provided visual queries consist of arbitrary image regions containing any motif or object. Previous methods for visual instance retrieval used feature point descriptors like SIFT [53] in a bag-of-words manner [25, 47], and many improvements have been made to this approach [54]. In recent years, Convolutional Neural Networks (CNN) have led to remarkable improvements in visual instance retrieval [55]. However, the primary research always dealt with photographs, either of the same places [56, 57] or objects [58]. If instance retrieval was applied to artworks, it was often limited to image level-based approaches [30, 59–62]. They do not allow to search for single regions, which is essential for art historians to investigate individual motifs or characteristics of an artwork. Just recently, Shen et al. [16] presented the first benchmark for finding and localizing individual motifs in a dataset of artworks with precise bounding box annotations. Most similar to our work is Shen et al. [16] and Seguin et al. [15], since both deal with visual instance retrieval in art collections on a regional level. In contrast to our work, Seguin et al. [15] use a triplet-loss to fine-tune a CNN in a supervised manner. However, they require a great deal of expert labor, and the learned features cannot be directly applied to other datasets. Shen et al. [16] avoids the need for labeled data and uses a self-supervised approach to learn dataset-dependent features. Similar to [63, 64], corresponding image regions are mined across the dataset using a spatial consistency verification, which are later used during the training stage. However, this is limited to small datasets with numerous repetitions. Furthermore, the presented retrieval system is based on a sliding-window approach and is not applicable to large-scale scenarios. In contrast, our method successfully reduces the domain gap within art collections and improves retrieval results without additional annotations or curated image collections. With our voting-based approach and efficient approximate nearest-neighbor search, we are able to find and localize small motifs in a few seconds across large and inhomogeneous datasets.

Visual search interfaces

Most search functionalities in digital image archives are still text-based searches through metadata [9–12, 24], which are limited since they require metadata and cannot capture the visual variety of images. In the following, we focus on visual search based systems.

Several companies, such as Google, Microsoft, and Idée Inc., offer engines to search the web for images [19–21]. However, these algorithms are not designed for artwork and do not allow users to search across their own datasets without publishing them. Only TinEye [21] offers a fee-based API for this purpose, but their service is restricted to find exact and altered copies and can not retrieve semantically similar images. Furthermore, their engines provide only a holistic image search. Although in Bing's search function, users can upload an image and mark an area to search for, this is not a proper regional search since images are still represented holistically on the dataset side and small objects or motifs within other images in the collection are not discovered. On the academic side, with the emergence of new visual search algorithms for the arts, significant effort has been made to provide user interfaces for their practical application. The Oxford Painting Search [22] is based on the work of Crowley et al. [25, 65] and is a search tool for the ARTUK dataset [6]. It allows searching for text queries, color, textures, object categories, and holistic images. However, their interface is restricted to a specific dataset and does not support searching on a regional level. Ahmed Elgammal et al. created the feed-based service ArtPI [23], a search tool for museums and auction houses. The system enables text and visual searches for holistic images. Besides visual content, users might also search for similar paintings regarding color or light treatment. However, their service also does not allow the search for single motifs or objects within images, which is essential for art history. In 2018, the Digital Humanities Laboratory of the École Polytechnique fédérale de Lausanne presented Replica [17]. In their interface, users can search for texts, holistic images, or image regions in the digitized art collection of the Giorgio Cini Foundation in Venice, which mainly includes artworks from the Italian Renaissance. It provides different viewing modes and allows user feedback to improve results. Their interface is most similar to ours. However, their search algorithm is based on expert annotations of related images and cannot be applied to other datasets without great effort. The most significant disadvantage of all previously mentioned systems is that they either allow only a holistic image search [19–23] or are limited to specific art collections [17], which dramatically limits their practical application in art history. In contrast, our interface allows users to search for entire images, single and multiple image regions, define spatial relations for regions, upload own collections, promote user feedback to refine results, search through metadata, browse through old searches and visualize results with different viewing modes. These functions ensure that the presented interface is a useful research tool for a broad audience in art history.

User interface

To enable art historians to apply our search algorithm in practice, we developed a user interface for the analysis of art collections. It was established in cooperation with art historians to meet their particular requirements of such an analysis tool. It allows users to upload and analyze individual image collections, where the algorithm retrieves not only identical but also similar semantic regions. This is important to reconstruct reception processes or investigate morphological variations of motifs over space and time. Besides searching for a single query region, the interface allows users to search for constellations of multiple regions, where they can specify how strong the algorithm should penalize geometric deviations from the query. This enables art historians to find similar regions and examine compositions or motifs characterized by multiple parts, which is essential for iconographical studies. The interface with a demonstration video is available on our project website: `https://compvis.github.io/visual-search`. In the following, we describe the interface architecture and workflow, additional features in more detail.

Architecture

Following modern web development standards, the user interface is divided into a front-end and a back-end. The front-end was developed using the Angular framework [66] and is served as a single-page application. The back-end consists of a REST API (webserver) and the search back-end, where the REST API manages the access to images and metadata and administrates the search back-end. The search back-end consists of multiple initialization and search workers running in the background, responsible for the actual initialization and search across the selected image collection. See Fig 2 for an overview.

Fig 2. Overview of the interface architecture. The application is divided into a front-end and back-end, where the front-end is served as a single-page application and represents the interface for the user. The back-end consists of a REST API and the search back-end, where the search back-end consists of multiple initialization and search workers itself.

Splitting the back-end into an API and a search back-end allows for asynchronous initialization and search requests and easy scaling. The search back-end can run on different machines because the communication with the webserver is managed through the API interface. For an efficient parallelization, the search back-end consists of multiple concurrently running workers that can perform different initialization and search requests. The webserver maintains a job queue which is read by the workers. Thereby, a search worker preferably takes over jobs whose search index is still in GPU memory. This accelerates further searches across the same dataset since it can take some minutes to load the search index to GPU for large datasets. All search results, user feedback, and refinements are stored in a SQL database. This allows for comfortable navigation through favorites and old search results and hence a convenient analysis of previous searches and visual relations within the image collections.

Workflow

The workflow of the interface is as follows. First, the user selects an image collection or uploads a new one and starts the initialization process. During this offline preparation stage, the algorithm extracts the features and builds the search index. This offline stage has to be done only once for each dataset. Afterwards, the index is stored and loaded to the GPU if needed. In the online search stage, users can mark one or multiple regions of interest by drawing bounding boxes and start the search process for these regions across the whole dataset. If users search for a composition of multiple regions, they can specify how strong the algorithm should take the geometric relationship between regions into account. If the desired geometric connectivity is set very low, the algorithm searches for images in which the selected regions occur somewhere in the image. And if it is set very high, the algorithm searches for the exact relative positioning of the selected regions, where small deviations are heavily penalized in the retrieval ranking. After the search algorithm is terminated, the results are visualized in the order of decreasing similarity. An overview of the workflow with the interface at different stages can be found in Fig 3.

Fig 3. Visualization of the interface workflow. First, the dataset to be searched is selected, or a new dataset is created and initialized (a). Then an image can be chosen, and one or multiple search boxes can be selected and searched across the dataset (b). The retrievals are displayed in an ordered list (c) or with a two-dimensional t-SNE embedding [67] (d).

Additional functions add to the the interface's usability: the possibility of adding and accessing metadata, storing and navigating through old search results and favorites, and alternating between a close-up and distant views of images and retrieved regions. It also provides an overview through a two-dimensional t-SNE embedding [67] of all search results based on the retrieval similarities. The layout of the interface supports easy and intuitive navigation through the search process. Each function aims to simplify the workflow for art historians. For example, the simultaneous view of selected favorites allows for comparative analysis, and a close-up view of retrieved regions enables a focused study of objects. The two-dimensional embedding of search results gives a quick overview and the possibility of finding patterns within search results faster.

Retrieval algorithm

From a computer vision point of view, the technical requirements of art historians towards a visual search algorithm are challenging. This includes the image representation as well as the retrieval system. Concerning the image representation, the search for arbitrary motifs across an art collection with large variations in style and technique poses severe challenges. The utilized feature descriptors have to be discriminative to find visual patterns but invariant to artistic styles at the same time. Therefore, off-the-shelf models trained on photos are insufficient. In the first part of this section, we address this problem and introduce a new multi-style feature aggregation that reduces the domain gap and significantly improves retrieval results. As far as the retrieval system is concerned, searching for image regions of any size in a large dataset within seconds is a major challenge. Encoding each image with a single feature descriptor cannot encode multiple objects or small items within the image. And searching over all possible regions with arbitrary size and aspect ratio is not feasible due to memory and time constraints. In the second part of this section, we address this problem and introduce a voting-based visual search. For all images and the selected query region, we extract a set of local patch descriptors, perform a nearest neighbor search on patch level, and predict retrieval bounding boxes by fusing all k-nearest neighbor votes of the local patches within the query region.

Multi-style feature aggregation

The following observation inspires the proposed feature aggregation. Feature descriptors generated with ImageNet [18] pre-trained CNNs are able to find similar regions when there are no or only slight stylistic differences. Our main idea is to use current style transfer models to transfer all images into the same averaged style domain, which reduces style differences and simplifies the retrieval task. For this, each image is stylized based on a fixed set of style instances and the extracted features are aggregated into one image representation. The approach consists of three steps. First, we determine a fixed set of images painted in different styles, which serve as our style templates. Second, we use a universal style transfer model to project each image into all styles of the style templates. Third, we aggregate the features of all style transformations into a single feature representation and use region-of-interest pooling to obtain descriptors of a given set of region proposals. In the following, we describe each step in more detail. See Fig 4 for an overview.

Style templates

We determine a set of style instances, which serve as our style templates and which we use for stylizing all dataset images. To find these templates, we proceed as follows.

Fig 4. Overview of the multi-style feature aggregation. It consists of three main steps: First, for all images, we extract their features using a classification network, cluster them in the feature space, and select the cluster centers as style templates. Second, given an input image, it is stylized concerning all style templates using a universal style transfer model. And third, all stylized image features are extracted and aggregated to the final multi-style feature representation. Please, see the text for more details.

First, we train a style classification network ϕ_{st} to get an image representation sensitive to artistic styles. For this, we use Wikiart [5] with 27 different style categories. The resulting network maps images into an m-dimensional feature space, $\phi_{st} \colon \mathcal{I} \to \mathbb{R}^m$, where $\mathcal{I}$ is the space of all images. We project all images in the dataset $\{I_1, \ldots, I_n\} \subset \mathcal{I}$ into this feature space and group them into k_s clusters according to their pairwise distances using K-means. Since ϕ_{st} is trained on style classification, different clusters contain different styles. We select the images which are closest to the cluster centers in the feature space and obtain a fixed set of images $\mathcal{S} = \{I_s | 1 \leq s \leq k_s\}$ with diverse styles, serving as our style templates. This selection is sufficient because our universal style transfer model is independent of the image content. The style templates do not have to represent the variation of all styles within the dataset but provide a subset of diverse styles that we can use to average over them. In our experiments, we set $k_s = 3$ since this gave the best performance.

Style transformation

We project all images of the dataset into all styles of the style templates using the universal style transfer model, which is a CNN $G \colon \mathcal{I} \times \mathcal{I} \to \mathcal{I}$. It takes a content image I_c and a style image I_s as input and synthesizes an image $G(I_c, I_s)$ with the content from I_c and style from I_s. For a given image $I \in \mathcal{I}$, we obtain all style transformations by calculating $G(I, I_s)$ for all $I_s \in \mathcal{S}$. The universal style transfer model is based on the network architecture of Li et al. [68] and consists of three major components: an encoder-decoder, a transformation, and a loss module. The encoder-decoder is trained to project the input image into a lower-dimensional feature space by the encoder so that the decoder is capable of reconstructing it. It is trained on MS-COCO [69], and the weights are fixed for the remaining training. The transformation module comprises two light-weights CNNs which receive the feature maps of the content and style image as input and generate a transformation matrix as output, respectively. The stylized image is generated by multiplying the encoder's content feature with the two transformation matrices and applying the decoder on the output. The loss module consists of a style loss and content loss. The style loss is computed at different layers with the Frobenius norm of the Gram matrix differences between the style and the stylized image. The contents loss is the Frobenius norm of the feature differences between the content image and the stylized image.

Feature fusion

In the last step, we average over the feature maps of all style transformations of a given image and obtain a single feature representation. In this way, we create an averaged style space for the image search that is more robust against style variations compared to the initial representation. Therefore, we proceed as follows. Given a feature encoder ϕ, an image $I \in \mathcal{I}$ and its transformed images $\{G(I, I_s) \mid I_s \in \mathcal{S}\}$, then we aggregate their

feature maps by taking their channel-wise mean, i.e.

$$\phi_{ms}\left(I, \mathcal{S}\right) = \frac{1}{1+|\mathcal{S}|}\left(\phi(I) + \sum_{I_s \in \mathcal{S}} \phi\left(G(I, I_s)\right)\right), \tag{1}$$

where we also consider the original image. This is beneficial since it contains fine-grained information that can be useful for the retrieval task but is lost during the transformation process. We have investigated alternative fusion strategies such as concatenation or taking the maximum. However, averaging has proven to be the most efficient method. Finally, given a set of local patches, we apply Precise ROI Pooling [70] on the new image representation and reduce their feature dimension using principle-component analysis with whitening (PCAW). To be stable against scale changes, the aggregated feature maps are generated in seven different scales and the most suitable scale is selected for each patch.

Retrieval system using iterative voting

Most visual search systems allow only a holistic image search based on full images as search queries. However, images consist of complex compositions of various motifs and objects, and searching on an image-level can only find the most dominant characteristics. However, for art history, searching for specific motifs or items is essential to study iconographic questions or trace reception processes of particular regions. Therefore, our algorithm should find similar instances of any region containing an arbitrary object or motif across an extensive visual art collection. This is a challenging requirement since encoding images with a single feature descriptor cannot capture small motifs accurately, and encoding all regions with various positions, sizes, and aspect ratios is not feasible due to memory and time constraints. Massively reducing the number of encoded regions using selection criteria is also not applicable since this requires knowing what the user is actually interested in. However, this varies from search to search and depends on the specific dataset.

Our goal is to introduce a generically applicable algorithm that provides accurate search results on various and diverse image collections and search queries. To tackle this challenge, we introduce an iterative voting based on a set of generic local patch descriptors. We encode each image using a moderate number of quadratic patches on multiple scales. Given a query region, we select the most discriminative patches within the query region and search for their nearest-neighbors across the whole dataset. Each of these nearest neighbors of local query patches votes for a retrieval bounding box, and multiple votes in an image are aggregated to the final retrieval bounding box. This approach has three main advantages. First, the search is conducted on a regional level and enables searching for small motifs that cannot be found with a holistic image search. Second, it allows precise localizations, independent of the patches used for the image encoding, and avoids the requirement of encoding all possible regions. Third, several weak search queries are combined with spatial verification, leading to more reliable search results. In the following, we describe the general structure of our approach and go into more detail about individual steps.

General structure

Our algorithm consists of an offline preparation and online search stage, see Fig 5 for an overview. In the offline preparation stage, the search index is initialized before the actual search is conducted. Therefore, the local patch descriptors are extracted using the multi-style feature extraction network (F1), compressed, and stored in the search index (F2). This step only needs to be done once. Afterwards, the search index is kept

in memory or saved and reloaded from disk if needed. In the online search stage, the users select a query region, most discriminative local patches inside this region are extracted (N1), and our voting is applied to their k-nearest-neighbors, which results in localized retrieval bounding boxes (N2). Finally, through local query expansion and re-voting, the retrieval results are further improved (N3).

Fig 5. Overview of the retrieval system. The retrieval system consists of an offline preparation and an online search stage. During the offline preparation stage, features of local patches on multiple-scales are extracted for all images (F1). Then, they are compressed and stored in the search index (F2). During the online search stage, descriptors of discriminative local patches within the query region are extracted (N1). Their k-nearest neighbors across the whole dataset are determined, and our voting procedure aggregates multiple local matches to retrieval bounding boxes (N2). Finally, the results are refined using local query expansion and re-voting (N3). Please, see the text for more details.

Image encoding with local patch descriptors

Our strategy for encoding images with local patch descriptors consists of two steps. See also (F1) in Fig 5. First, we generate a broad set of local patches by dividing each image into quadratic regions on multiple scales in a sliding window manner. Second, we filter them by selecting the most discriminative patches by applying non-maximum suppression on their feature activations, which we obtain by summing over the feature channel of the ROI pooled multi-style feature maps. By this, we get local patches on different scales that are more discriminative and more suitable for the retrieval task. We generate a maximum of 4000 proposals for each image $I \in \{I_1, \dots I_n\}$ in the dataset and extract their multi-style features $\mathcal{D}(I)$ that serve as our local patch descriptors. To keep the search index compact, we reduce the maximal number of local patches by 375 for each additional 20K images.

Search index and geometric database generation

For fast nearest neighbor search, we build a search index containing all local patch descriptors for all images, denoted by $\mathcal{D} = \cup_{i=1,\dots,n}\mathcal{D}(I_i)$, and a database including image names and positions. See also (F2) in Fig 5. For the search index, we utilize the Inverted File Index (IVF) and the Product Quantization (PQ) algorithm of [1, 71], which provide a fast GPU-parallelized implementation. The IVF algorithm clusters all feature vectors and computes their centroids using K-means. Given a query vector, the distance to all centroids is determined, and only the feature vectors assigned to a small subset of closest centroids are considered for the nearest neighbor search, which massively reduces the search space. Now, for the actual search the PQ algorithm is used. Here, the feature vectors are sliced into subvectors, and a codebook for each of them is learned. Based on these codebooks, the feature vectors can be stored efficiently using their ids, and a look-up table of all pairwise centroid distances allows a fast approximated nearest neighbor search. For a given query vector, the closest centroid of each query vector slice is determined, and the distances to all other vectors approximated using the pairwise centroid distances from the look-up table. For each dataset, we train an individual search index by randomly selecting 1000 images and applying IVF clustering and PQ codebook generation to their local patch descriptors.

Query formulation with local patch descriptors

The first step in the online search stage is the query formulation, see also (N1) in Fig 5. When the user selects an image and marks a rectangle q as the query region, we use this rectangle and the most discriminative local patches inside this region for the actual search. We extract local patches as described in the section on image encoding and select the most discriminative among them as follows. First, we remove all patches with too few overlap and are much smaller than the query region. On the remaining patches, we apply non-maximum suppression based on their feature activation and select up to 25 patches with the highest value. Finally, we extract the features of the query region and the selected local query patches, which we denote $\mathcal{D}(I, q)$ and are used for the voting in the following.

Voting based on local matches

Each image I in the dataset is encoded with a set of local patch descriptors on multiple scales, called $\mathcal{D}(I)$. All local patch descritpors of all images $\mathcal{D}$ are stored in the search index. For a given query region q, it's descriptor and additional local query patch descriptors $D(I, q)$ within the query region are extracted. Then, the voting consists of two main steps. First, we search for the k-nearest neighbors of all local query patch descriptors and perform majority-based voting to filter most promising images. Second, all local matches in the filtered images vote for retrieval bounding boxes, which are fused to final retrieval results. The two steps are visualized in (N2) of Fig 5 or Fig 6, and will be explained in more detail in the following.

Fig 6. Overview of our voting based on local matches. The voting consists of two main steps. First, for each local query patch, we search for its k-nearest neighbors across the dataset, order all images based on their sum of local matching scores, and focus on the images with the highest score. Second, each local match votes for a specific center and scale of the retrieval bounding box, which are aggregated to the final retrieval results.

In the first step, for each $f \in D(I, q)$ we search for its k-nearest neighbours $NN_k(f, \mathcal{D})$ across all local patch descriptors $\mathcal{D}$, where we set $k = 2048$ and call (f, g) a local match for any $g \in NN_k(f, D)$. For all local matches, we define their local matching scores via

$$s_f(g) = \exp\left(-\frac{\|g - f\|_2^2}{\|\hat{g} - f\|_2^2}\right), \tag{2}$$

where $\hat{g} \in NN_k(f, \mathcal{D})$ with a fixed rank of 512 and works as a reference distance. We define the similarity of query region q to an image I' by summing over all local matches in I', i.e.

$$S_q(I') = \sum_{f \in \mathcal{D}(I,q)} \max\left(\{s_f(g) \,|\, g \in NN_k(f, \mathcal{D}) \cap \mathcal{D}(I')\}\right), \tag{3}$$

where we consider at most one hit per image. Based on S_q we select the 500 most promising images containing q. By this pre-selection, we reduce the computational cost, and the second voting step becomes independent of the actual dataset size. Other approaches also perform such a pre-selection [17]. In contrast to those, we do this on a regional level instead of an image-level and avoid losing small corresponding regions by this filtering.

In the second step, we focus on the most promising images with their local matches to predict the final retrieval bounding boxes. We consider only rectangles with identical aspect ratios and limit ourselves to position and scale changes, neglecting more complex

transformations like rotation or shearing. By this, we do not have to increase the voting space and accelerate the search speed. In the following, we denote $\mathbf{c}_r$ and d_r as the center and diagonal of an arbitrary rectangle r. Now, each local match (f, g) with $g \in \mathcal{D}(I')$ votes for a specific rectangle r in image I'. We define $\mathbf{v}_f = \mathbf{c}_q - \mathbf{c}_f$ as the connection vector from the center of the local query patch $\mathbf{c}_f$ to the query rectangle $\mathbf{c}_q$. Then the local match (f, g) votes for a rectangle r in I' with center $\mathbf{c}_r = \mathbf{c}_g + \mathbf{v}_f \cdot d_g/d_f$ and diagonal $d_r = d_q \cdot d_g/d_f$. Since we presuppose identical aspect ratios as the query region, the retrieval rectangle is unambiguously defined by its center and diagonal length. We fuse all those matches by creating a voting map similar to [54]. Therefore, the vote of local matches (f, g) are inserted into the voting map of image I', where each point of the voting map represents a retrieval box at the respective position. Hereby, the voting map is a down-scaled version of the image and we use the local similarity score defined in Equ. 2 for the voting score. The voting map includes only the votes for retrieval bounding box centers. To estimate the scale, we average over the diagonals of the local matches voting for the same position. This way, we do not have to increase the voting space by another dimension and limit computational overhead. To reduce quantization errors and stabilize the voting, each local match votes for a small five by five window with a Gaussian kernel. Finally, we take the position of the maximum in the voting map and the averaged diagonal to obtain the center and diagonal of the best retrieval bounding box in image I'. In contrast to [15, 16], our voting based retrieval allows finding and localization at the same time without computational expensive sliding-window search over all image positions.

Local query expansion and re-voting

After the first search results, we apply local query expansion and repeat the search process. See (N3) in Fig 5. For this, we use the ten nearest neighbors of each local query patch and merge their feature descriptors by taking their mean and L_2 normalization. After the local query expansion, we obtain more generalized local query patch descriptors since they include multiple instances. Only regions with a small distance to all of these instances are ranked high, which stabilizes and improves the search results. The expanded local query patches do not have the same coordinates in the query image due to the modified feature descriptor and potential shifts of local matches. Therefore, we update the voting vectors and diagonals by determining their nearest neighbors in the query image and measuring the distance to the query bounding box center. Based on the expanded feature descriptor and updated voting vector, we repeat the voting procedure described previously, which leads to overall improved search results.

Experiments

This section presents diagnostic experiments to study the multi-style feature aggregation and voting-based retrieval system, as well as comparative evaluations on challenging benchmarks. For implementation details of our method, we refer to S1 File. PDF file with implementation details.

Datasets and evaluations

To show the efficiency and generalization capabilities of our system, we evaluate on five different datasets, including art and photo collections.

Brueghel

Our main evaluation is based on the image collection of Brueghel paintings [72] with annotations from Shen et al. [16]. This is currently the only publicly available dataset with annotations for localization and instance retrieval in the arts to the best of our knowledge. The collection consists of 1,587 paintings with different styles, techniques, and depicted scenes. It contains ten annotated motifs with 11 to 57 instances each, resulting in 273 annotations. Analogously to Shen et al. [16], we count retrievals as correct if the intersection over union (IoU) of predicted with ground-truth bounding boxes is larger than 0.3. The rather low threshold can be justified that this localization accuracy is sufficient for most retrieval applications. We compute the Average Precision (AP) for each query, average them per class, and report the class level mean Average Precision (mAP).

Brueghel5K & Brueghel101K

To cover the actual use case in art history, including vast and inhomogeneous art collections, we introduce two extensions of the Brueghel dataset, which we call Brueghel5K and Brueghel101K. For this, we sample 3,500 and 100,000 images from the Wikiart dataset [5], respectively, and add them to the Brueghel dataset. The included images contain different motifs in different techniques and styles and act as distractors, making the retrieval task much more difficult. For the queries and evaluation, we use the same annotations from the Brueghel dataset. The distractors are additional negatives for all queries, where we excluded all images painted by Peter Brueghel the Elder from the selection to circumvent false negatives. We follow the previously described evaluation protocol and report the mAP.

Large Time Lags Location (LTLL)

We also evaluate our algorithm on the Large Time Lags Location (LTLL) dataset, collected by Fernando et al. [73]. The dataset consists of 225 historical and 275 modern photographs taken from 25 cities and towns spanning over a range of more than 150 years. The goal is to recognize the location of an old image given a set of labeled modern photographs. The dataset's difficulty is also to bridge the gap between different domains as historical and modern photos are very different in their visual appearance and can be understood as two different domains. Since our retrieval system assumes that users mark image regions they are interested in, we provide and utilize annotated query bounding boxes for our and all baseline models. Analogously to the evaluation protocol of Fernando et al. [73], we report the accuracy of the first retrieval.

Oxford5K

Finally, we evaluate our approach on the Oxford5K dataset. It was collected by Philbin et al. [56] and is one of the most commonly used benchmarks for instance retrieval. It consists of 5,062 photos from Oxford, which were compiled from Flicker search terms. The annotations contain 11 different landmarks from Oxford, including towers, cathedrals and bridges, with five different queries for each. The occurrence per landmark ranges from 7 up to 220. We follow the evaluation protocol of [56] and compute the Average Precision (AP) for each query, average them per landmark and report the mean Average Precision (mAP).

Effect of the multi-style feature aggregation

To validate the effect of the proposed multi-style feature aggregation, we compare the performance of our retrieval system with different feature descriptors. We compare against features generated with VGG-16 with batch normalization, truncated after the conv-4 layer, which is either pre-trained on ImageNet [18] (ImageNet) or fined-tuned with the self-supervised approach of Shen et al. [16] (Artminer). For a fair comparison, we selected the same backbone architecture for all baselines. Moreover, we investigated alternative fusion strategies besides averaging, including concatenation (w/concat) and taking the maximum (w/max). The results are reported in Table 1.

Table 1. Comparison of feature representations and fusion strategies.

Features	Brueghel			LTLL	Oxford5K
	[16, 72]	5K	101K		
ImageNet	82.3	78.3	70.2	87.6	89.5
Artminer [16]	83.4	39.9	38.5	89.5	80.4
Ours w/max	87.2	83.7	76.2	92.3	90.7
Ours w/concat	86.4	84.1	**79.3**	90.9	**91.4**
Ours	**87.7**	**84.5**	77.5	**92.8**	**91.4**

Performance comparison of our multi-style feature aggregation with different fusion strategies (Ours) and features generated from VGG-16, which are either pre-trained on ImageNet [18] (ImageNet) or fined-tuned with the self-supervised approach of Shen et al. [16] (Artminer).

From the results, it can be seen that our approach improves retrieval results on all benchmarks. The performance gain compared to the initial ImageNet pre-trained model is particularly large for the art datasets. This is reasonable since there is a particularly large domain gap due to stylistic differences between images in the dataset. For the Oxford5K dataset we obtain a rather small improvement. Our multi-style features also achieve significantly better results compared to Artminer. Their approach shows severe deficiencies for large and inhomogeneous datasets, as well as for the Oxford5K dataset. The analysis of fusion strategies shows that taking the average gives superior or comparable results for most datasets. Only for Brueghel101K, the concatenation provides a slightly better retrieval performance.

Effect of the iterative voting

To analyze the iterative voting effect, we measure the retrieval performance with our multi-style feature aggregation and disable different components. We search only with the whole query region without voting (wo/voting), restrict ourselves to the first round of voting and remove the local query expansion and re-voting (wo/re-voting), and compare with our full system. To investigate the influence of the voting on the localization, we also measure the retrieval performance for different IoU thresholds on the Brueghel dataset. The higher the IoU threshold, the more accurate the localization has to be. All results are summarized in Table 2.

From the results, it can be seen that the iterative voting significantly improves the retrieval performance on all benchmarks. The first round of voting mainly increases the results on the LTLL and the Oxford5K dataset. For the Brueghel dataset, the first round of voting enhances finding additional instances only slightly, but the localization improves clearly, which can be seen by the performance gain for higher IoU values. The local query expansion and re-voting lead to further improvements. This effect is much smaller for the LTLL and Oxford5K compared to the Brueghel dataset. This can be

Table 2. Effect of the iterative voting.

Methods	Brueghel			LTLL	Oxford5K
	IoU@0.3	IoU@0.5	IoU@0.7		
Ours wo/voting	75.9	55.7	7.1	76.6	70.7
Ours wo/re-voting	76.0	61.5	27.7	91.4	89.5
Ours	**87.7**	**68.9**	**31.8**	**92.8**	**91.4**

We measure the performance for searching only with the whole query region (wo/voting), restrict ourselves to the first round of voting (wo/re-voting), and compare it to our full system. We also report the performance on the Brueghel dataset for different IoU thresholds.

explained by very similar instances of some motifs in the Brueghel dataset. Hence, the first retrievals are very similar to the query and thus are very well suited for the expansion and significantly improve the second round of voting. Furthermore, our ablation study also shows that even without the local query expansion and re-voting, our results are comparable to the state-of-the-art of Shen et al. [16] on the Brueghel dataset.

Computational cost

In the following, we examine the search speed and report the memory consumption of the search index. Since the search index has to be stored on the GPU for optimal speed, its size is a potentially limiting factor. We compare our algorithm with the approach of Shen et al. [16]. For a fair comparison, all measurements are conducted on the same server with 3 GPUs (NVIDIA Quadro P5000). For Shen et al., we assumed a perfect parallelization of their algorithm and divided their search times by 3. We also report the times for our algorithm if the search is performed on the CPU. In Table 3, we summarize the results for a single query and the memory consumption on the GPU for varying dataset sizes.

Table 3. Comparison of search speed and index size

Method	CPU/GPU	5K	20K	40K	60K	80K	100K
Artminer [16]	GPU	12.9 min	50.4 min	1.7 h	2.8 h	3.8 h	4.6 h
Ours	CPU	6.6 s	– s	9.5 s	10.6 s	12.0 s	13.5 s
Ours	GPU	**6.5 s**	– s	**7.7 s**	**8.4 s**	**8.7 s**	**8.9 s**
Ours	GPU	3.5 GB (1)	6.3 GB (1)	17.7 GB (2)	20.4 GB (2)	23.4 GB (2)	27.3 GB (3)

Comparison of the search speed of our method (Ours) with Shen et al. [16] (Artminer) for varying dataset sizes (upper part), where we also report the utilized memory on the GPU for the nearest neighbor search (lower part). With the numbers in brackets we note how many GPUs were used to store the index.

The table shows that our search algorithm needs about 9 seconds on a dataset with 100K images. This is much faster than the sliding-window based approach of Shen et al. [16]. Furthermore, it depends only moderately on the dataset size. We examine a large offset in the search time due to the multi-scale feature extraction of the search region, which can be further optimized in the future. A limiting factor of the number of images that can be searched is the index size. One of its determining factors is the number of extracted local patches per image. This number essentially determines how small the query regions can be so that they can still be found reliably. Since the search requirements and the available hardware depend on the application, this number should be adjusted according to the actual use case. Another possibility to decrease the index size is to reduce the dimension of the local patch descriptors. However, in our

experiments, we found that the retrieval performance significantly decreases for feature dimensions smaller than 96. Our investigation also shows that the nearest neighbor search on the CPU is a valid option and only slows down the search by a few seconds. And for smaller data sets this has no influence at all.

Benchmark performance

In the following, we provide quantitative and qualitative experiments on the described benchmarks.

Quantitative evaluation

We compare the performance of our algorithm against several baselines. This includes max-pooled features of an ImageNet pre-trained CNN (ImageNet, max-pool), and the state-of-the-art approaches of Shen et al. [16] (Artminer) and Radenović et al. [14]. For a fair comparison, we select the same backbone architecture for all methods. We also list the original performance reported by Shen et al. [16] with their fine-tuned ResNet-18 model. In contrast to Artminer, our approach utilizes marked query regions on the LTLL and Oxford5K dataset. However, according to their publication and our experiments, Artminer does not profit from query regions and obtains better results using full images, which allows their algorithm to utilize more context. Therefore, we report results of their so-called discovery mode, which utilizes full images as queries. We summarized all results in Table 4.

Table 4. Comparison of the retrieval performances on different benchmarks.

Methods	Net.	F.-tuned	Dim.	Brueghel			LTLL	Oxford5K
				[16, 72]	5K	101K		
ImageNet, max-pool	VGG-16	no	512	24.0	22.5	17.7	47.8	25.6
Radenović et al. [14]	VGG-16	no	512	15.5	12.7	5.9	59.3	53.4
Radenović et al. [14]	VGG-16	yes	512	15.8	12.8	5.7	76.1	87.8
Artminer [16]	ResNet-18	no	256	58.1	56.0	50.2	78.9	84.9
Artminer [16]	ResNet-18	yes	256	76.4	46.5	37.4	88.5	85.7
Artminer [16]	VGG-16	no	512	54.4	50.5	44.1	81.8	85.0
Artminer [16]	VGG-16	yes	512	79.9	39.5	36.4	88.9	81.5
Ours	VGG-16	no	96	87.7	84.5	76.6	**92.8**	91.4
Ours	VGG-16	no	128	87.4	86.4	79.1	**92.8**	**91.6**
Ours	VGG-16	no	256	**89.0**	**86.7**	**80.5**	**93.3**	91.3

Retrieval results of our method (Ours), max-pooled features, and state-of-the-art methods on the Brueghel [16, 72], Brueghel5K, Brueghel101K, LTLL [73] and Oxford5K [56] dataset. We also report the network architecture (Net.), whether features are fine-tuned on the retrieval task (F.-tuned), and the dimension of the underlying features (Dim).

We outperform all baselines on all benchmarks without fine-tuning on the retrieval task and with a much smaller feature dimension. Compared to Artminer [16], our algorithm is much more stable against distractors, i.e. images without any corresponding region in the dataset. For the self-supervised training of Artminer the chance to pick mainly image regions without any correspondence in the dataset is very high, resulting in too few and potentially spurious training pairs. In this scenario, their training worsens the performance, which can already be seen for Brueghel5K with a small number of distractors. In contrast, our method is much more robust against distractors, and the performance decreases moderately. On Oxford5K, our approach even outperforms Radenovic et al. [14], although it is designed explicitly for landmark

localization and trained on a large dataset of various landmarks in a self-supervised manner. Since their approach was trained on landmarks, it is not surprising that their approach performs relatively poorly on the art datasets.

Qualitative evaluation

In the following, we provide a qualitative comparison on the Brueghel dataset with the state-of-the-art [16] (Artminer) and further qualitative examples of our search system on all benchmarks. For a better overview, we show only a few retrievals per query at certain rankings that we have evenly distributed over the total number of annotated instances.

In Fig 7, we provide a direct comparison of our method with Artminer for different queries and ranks. For both approaches, the retrievals at first few ranks are correct. However, our method can find significantly more instances, which can be seen by the correct retrievals at higher ranks.

Fig 7. Qualitative comparison of our method with Artminer. Comparison of our approach (Ours) and Shen et al. [16] (Artminer) on the Brueghel dataset. Queries are visualized in blue on the left and search results on the right, where we draw green bounding boxes if the IoU is greater than 0.3 and red otherwise. For a better overview, we draw only the first and four additional search results with a distance of the ranking given by the total number of annotated instances of the query divided by four.

In Fig 8, we provide qualitative retrieval examples on Brueghel, LTLL, and Oxford5K. We show the retrievals on the entire image and in a close-up view to allow a detailed comparison of the query and retrievals. The search results show that our algorithm can find similar objects despite differences in color and artistic style. Furthermore, we see that even small objects can be precisely localized despite changes in perspective and partial occlusions.

Fig 8. Retrieval examples on Brueghel, LTLL and Oxford5K. We show retrieval examples on the Brueghel, LTLL, and Oxford5K dataset in rows 1-2, 3-4, and 5-6, respectively. For a better comparison, the first rows show retrievals in full screen (Full), and the second rows show zoomed-in versions (Zoom). Queries are visualized in blue on the left and search results on the right, where we draw green bounding boxes if the retrievals are correct and red otherwise. For a better overview, we draw only the first and four additional search results with a distance of the ranking given by the total number of annotated instances of the query divided by four.

.

Conclusion

In this article, we have presented a novel search algorithm and user interface to find similar regions in an extensive art collection. Our system allows a large-scale analysis of art datasets. This enables art historians to identify relations between artists, study the popularity of motifs and specific topics in art history, or to examine how they have been adapted and altered in form and content over time and space.

The presented search algorithm is based on a novel multi-style feature aggregation, which reduces the domain gap and improves instance retrieval across artworks. In contrast to previous methods, our algorithm requires neither labeled data nor curated image collections for self-supervised training. The presented retrieval system is based on

a voting procedure of local patch descriptors within the actual search region combined with recent GPU-accelerated nearest neighbor search [1]. It enables users to find and localize even small motifs or objects in an extensive dataset in seconds. We have validated our model's performance on various benchmarks, including collections of artworks [16, 72] and photographs [56]. We have also demonstrated that our method applies to real-world scenarios with large and inhomogeneous datasets. Besides the search algorithm, we developed a user interface that allows art historians to use our algorithm in practice. The interface integrates many useful functionalities and enable users not only to search for single regions but also for constellations of multiple regions and to refine search results based on user feedback. Therefore, our specifically tailored visual search interface and methodological contribution constitute a significant progress in the computer-assisted analysis of visual art.

Supporting information

S1 File. PDF file with implementation details. In this PDF file we give further implementation details of our algorithm.

S2 Video. Demonstration of the visual search interface. The purpose of the video is to illustrate the functionalities of the user interface. During the review process the interface is accessible as an online web application on our project website `https://compvis.github.io/visual-search`. For login please use the user credentials provided on the project website and follow the steps described in the video. If the visual search interface is not reachable or other technical issues occur, please contact the corresponding authors.

S3 Data. Additional data. The majority of utilized benchmarks are publicly available. To complete the data, we provide the download links of the Wikiart distractors used to create Brueghel5K and Brueghel101K. Furthermore, we provide the download links and labels of the Wikiart images used for training the style classification network and the additional query bounding boxes of the LTLL dataset. The data is available on our project website `https://compvis.github.io/visual-search`.

References

1. Johnson J, Douze M, Jégou H. Billion-scale similarity search with GPUs. arXiv preprint arXiv:170208734. 2017;.
2. Johnson CD. Memory, metaphor, and Aby Warburg's Atlas of images. Cornell University Press; 2012.
3. Hristova S. Images as data: cultural analytics and Aby Warburg's Mnemosyne. International Journal for Digital Art History. 2016;(2).
4. Van Straten R. An introduction to iconography. vol. 1. Psychology Press; 1994.
5. Wikiart;. `https://www.wikiart.org`.
6. ArtUk;. `https://artuk.org`.
7. Web gallery of art;. `https://www.wga.hu`.
8. Strezoski G, Worring M. Omniart: multi-task deep learning for artistic data analysis. arXiv preprint arXiv:170800684. 2017;.

9. Bayrische Staatsbibliothek;. https://www.bsb-muenchen.de.

10. Prometheus, das digitale Bildarchiv fúr Forschung und Lehre;.

11. Rijksmuseum;. https://www.rijksmuseum.nl.

12. Bildindex der Kunst und Architektur;. https://www.bildindex.de.

13. Radenović F, Iscen A, Tolias G, Avrithis Y, Chum O. Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking. arXiv preprint arXiv:180311285. 2018;.

14. Radenović F, Tolias G, Chum O. Fine-tuning CNN image retrieval with no human annotation. IEEE transactions on pattern analysis and machine intelligence. 2018;41(7):1655–1668.

15. Seguin B, diLenardo I, Kaplan F. Tracking Transmission of Details in Paintings. In: DH; 2017.

16. Shen X, Efros AA, Aubry M. Discovering visual patterns in art collections with spatially-consistent feature learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2019. p. 9278–9287.

17. Seguin B. The Replica Project: Building a visual search engine for art historians. XRDS: Crossroads, The ACM Magazine for Students. 2018;24(3):24–29.

18. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee; 2009. p. 248–255.

19. Google search;. https://www.google.com.

20. Bing search;. https://www.bing.com.

21. TinEye search;. https://www.tineye.com.

22. Oxford Painting Search;. http://zeus.robots.ox.ac.uk/artsearch/.

23. ArtPI - The Art API;. https://www.artpi.co.

24. Crissaff L, Ruby LW, Deutch S, DuBois RL, Fekete JD, Freire J, et al. ARIES: enabling visual exploration and organization of art image collections. IEEE computer graphics and applications. 2018;38(1):91–108.

25. Crowley EJ, Zisserman A. In search of art. In: Workshop at the European Conference on Computer Vision. Springer; 2014. p. 54–70.

26. Seguin B, Striolo C, Kaplan F, et al. Visual link retrieval in a database of paintings. In: European Conference on Computer Vision. Springer; 2016. p. 753–767.

27. Gonthier N, Gousseau Y, Ladjal S, Bonfait O. Weakly supervised object detection in artworks. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018. p. 0–0.

28. Schlecht J, Carqué B, Ommer B. Detecting gestures in medieval images. In: Image Processing (ICIP), 2011 18th IEEE International Conference on. IEEE; 2011. p. 1285–1288.

29. Ufer N, Souiai M, Cremers D. Wehrli 2.0: An Algorithm for "Tidying up Art". In: European Conference on Computer Vision. Springer; 2012. p. 532–541.

30. Takami M, Bell P, Ommer B. An Approach to Large Scale Interactive Retrieval of Cultural Heritage. In: GCH; 2014. p. 87–95.

31. Shen X, Pastrolin I, Bounou O, Gidaris S, Smith M, Poncet O, et al. Large-Scale Historical Watermark Recognition: dataset and a new consistency-based approach. arXiv preprint arXiv:190810254. 2019;.

32. Hertzmann A, Jacobs CE, Oliver N, Curless B, Salesin DH. Image analogies. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques; 2001. p. 327–340.

33. Gatys LA, Ecker AS, Bethge M. Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 2414–2423.

34. Sanakoyeu A, Kotovenko D, Lang S, Ommer B. A style-aware content loss for real-time hd style transfer. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018. p. 698–714.

35. Elgammal A, Liu B, Elhoseiny M, Mazzone M. CAN: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. arXiv preprint arXiv:170607068. 2017;.

36. Hertzmann A. Can computers create art? In: Arts. vol. 7. Multidisciplinary Digital Publishing Institute; 2018. p. 18.

37. Ginosar S, Haas D, Brown T, Malik J. Detecting people in cubist art. In: European Conference on Computer Vision. Springer; 2014. p. 101–116.

38. Westlake N, Cai H, Hall P. Detecting people in artwork with cnns. In: European Conference on Computer Vision. Springer; 2016. p. 825–841.

39. Yin R, Monson E, Honig E, Daubechies I, Maggioni M. Object recognition in art drawings: Transfer of a neural network. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2016. p. 2299–2303.

40. Jenicek T, Chum O. Linking Art through Human Poses. arXiv preprint arXiv:190703537. 2019;.

41. Karayev S, Trentacoste M, Han H, Agarwala A, Darrell T, Hertzmann A, et al. Recognizing image style. arXiv preprint arXiv:13113715. 2013;.

42. Mensink T, Van Gemert J. The rijksmuseum challenge: Museum-centered visual recognition. In: Proceedings of International Conference on Multimedia Retrieval; 2014. p. 451–454.

43. Tan WR, Chan CS, Aguirre HE, Tanaka K. Ceci n'est pas une pipe: A deep convolutional network for fine-art paintings classification. In: 2016 IEEE international conference on image processing (ICIP). IEEE; 2016. p. 3703–3707.

44. Wilber MJ, Fang C, Jin H, Hertzmann A, Collomosse J, Belongie S. Bam! the behance artistic media dataset for recognition beyond photography. In: Proceedings of the IEEE International Conference on Computer Vision; 2017. p. 1202–1211.

45. Seguin B, Striolo C, Kaplan F, et al. Visual link retrieval in a database of paintings. In: European Conference on Computer Vision. Springer; 2016. p. 753–767.

46. Garcia N, Renoust B, Nakashima Y. ContextNet: representation and exploration for painting classification and retrieval in context. International Journal of Multimedia Information Retrieval. 2020;9(1):17–30.

47. Jegou H, Douze M, Schmid C. Hamming embedding and weak geometric consistency for large scale image search. In: European conference on computer vision. Springer; 2008. p. 304–317.

48. Tolias G, Sicre R, Jégou H. Particular object retrieval with integral max-pooling of CNN activations. arXiv preprint arXiv:151105879. 2015;.

49. Yi KM, Trulls E, Lepetit V, Fua P. Lift: Learned invariant feature transform. In: European Conference on Computer Vision. Springer; 2016. p. 467–483.

50. Noh H, Araujo A, Sim J, Weyand T, Han B. Large-scale image retrieval with attentive deep local features. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 3456–3465.

51. Girshick R. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision; 2015. p. 1440–1448.

52. Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 779–788.

53. Lowe DG. Distinctive image features from scale-invariant keypoints. International journal of computer vision. 2004;60(2):91–110.

54. Shen X, Lin Z, Brandt J, Avidan S, Wu Y. Object retrieval and localization with spatially-constrained similarity measure and k-nn re-ranking. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 2012. p. 3013–3020.

55. Collins E, Süsstrunk S. Deep feature factorization for content-based image retrieval and localization. In: 2019 IEEE International Conference on Image Processing (ICIP). IEEE; 2019. p. 874–878.

56. Philbin J, Chum O, Isard M, Sivic J, Zisserman A. Object retrieval with large vocabularies and fast spatial matching. In: 2007 IEEE conference on computer vision and pattern recognition. IEEE; 2007. p. 1–8.

57. Philbin J, Chum O, Isard M, Sivic J, Zisserman A. Lost in quantization: Improving particular object retrieval in large scale image databases. In: 2008 IEEE conference on computer vision and pattern recognition. IEEE; 2008. p. 1–8.

58. Nister D, Stewenius H. Scalable recognition with a vocabulary tree. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2. Ieee; 2006. p. 2161–2168.

59. Shrivastava A, Malisiewicz T, Gupta A, Efros AA. Data-driven visual similarity for cross-domain image matching. In: Proceedings of the 2011 SIGGRAPH Asia Conference; 2011. p. 1–10.

60. Picard D, Gosselin PH, Gaspard MC. Challenges in content-based image indexing of cultural heritage collections. IEEE Signal Processing Magazine. 2015;32(4):95–102.

61. Crowley EJ, Parkhi OM, Zisserman A. Face painting: querying art with photos. 2015;.

62. Mao H, Cheung M, She J. DeepArt: Learning Joint Representations of Visual Arts. In: Proceedings of the 2017 ACM on Multimedia Conference. ACM; 2017. p. 1183–1191.

63. Ufer N. Deep Semantic Feature Matching. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 6914–6923.

64. Ufer N, Lui KT, Schwarz K, Warkentin P, Ommer B. Weakly Supervised Learning of Dense Semantic Correspondences and Segmentation. In: German Conference on Pattern Recognition. Springer; 2019. p. 456–470.

65. Crowley EJ, Zisserman A. The state of the art: Object retrieval in paintings using discriminative regions. 2014;.

66. Angular;. `https://angular.io/`.

67. Maaten Lvd, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008;9(Nov):2579–2605.

68. Li X, Liu S, Kautz J, Yang MH. Learning linear transformations for fast arbitrary style transfer. arXiv preprint arXiv:180804537. 2018;.

69. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft coco: Common objects in context. In: European conference on computer vision. Springer; 2014. p. 740–755.

70. Jiang B, Luo R, Mao J, Xiao T, Jiang Y. Acquisition of localization confidence for accurate object detection. In: Proceedings of the European Conference on Computer Vision (ECCV); 2018. p. 784–799.

71. Jegou H, Douze M, Schmid C. Product quantization for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence. 2010;33(1):117–128.

72. "Brueghel family: Jan Brueghel the elder." The Brueghel family database. University of California, Berkeley;. `http://www.janbrueghel.net`.

73. Fernando B, Tommasi T, Tuytelaars T. Location recognition over large time lags. Computer Vision and Image Understanding. 2015;139:21–28.