

INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

HOW WELL DOES A CROWD RECOGNIZE DIFFERENCES BETWEEN PHOTOGRAPHS OF THE SAME ARTWORKS?

Cedric Kummer

Bachelorarbeit

Aufgabensteller	Prof. Dr. François Bry
Betreuer	Prof. Dr. François Bry, M.Sc. Stefanie Schneider
Abgabe am	26.04.2021

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.

München, den 26.04.2021
Matrikelnummer:



Cedric Kummer
11565365

Abstract

This bachelor thesis presents two novel algorithms that can be used in combination to determine the similarity of two photographs. The one algorithm analyses and compares the tags of two reproductions. These tags were assigned in advance, mainly by laypersons on the ARTigo platform. More precisely, the algorithm calculates the cosine similarity of the tags that have been assigned to the images. The information provided by the first algorithm can then be used to verify how much the tags annotated by the crowd differ between two reproductions that are already known to represent the same original. The ARTigo platform is part of an interdisciplinary project within the field of digital art studies, in which computer scientists and art history scholars conduct joint research. This bachelor thesis is part of that research project. The second algorithm is a computer vision algorithm that first calculates the feature vectors of the images. These are then matched so as to estimate how similar the two images are. Then, in a second step, the second algorithm calculates the general colour difference of the images in the Lab colour space. The results of the second algorithm can then be used to draw conclusions about the actual difference between the images. These results can also be used to classify whether, in the case of a high degree of similarity between the tags, two digital reproductions of the same original are involved, or whether both reproductions merely show the same or a similar motif. Finally, the thesis reports how the similarity of the assigned tags and the results of computer vision can be used to draw conclusions regarding whether two images represent the same original. This thesis is concluded with reflections on further possible application of the approach described in this thesis as well as how the work reported about in this thesis could be deepened.

Acknowledgments

I would like to thank Professor François Bry and Stefanie Schneider very much for their excellent supervision during this bachelor thesis. Unfortunately, due to the still ongoing pandemic, we were never able to meet in person, except via Zoom. I would also like to thank my family and friends who have always supported me where they could during this time. I was able to learn a lot of things during this time, from data analysis in R to image processing with OpenCV and to the process of scientific work.

Contents

1 Introduction	1
2 ARTigo	6
2.1 What are social tags and why are they used by ARTigo?	6
2.2 Data creation on an example	7
2.3 Data structure	7
3 Related Work	9
4 Cleansing the ARTigo dataset	11
4.1 Merging and cleansing the data	11
4.2 Correcting misspelled tags	12
5 Data Analysis	15
5.1 Creation of the sparseMatrix	15
5.2 Computing the cosine similarity	16
5.3 Evaluation of the calculated cosine similarities	17
6 Computer Vision	25
6.1 Feature detection	25
6.1.1 Computing the features	26
6.1.2 Comparing the features	27
6.2 Computation of the colour differences	28
6.3 Results	34
7 Conclusion	43
7.1 Evaluation of the results	43
7.2 Further thoughts	44
Bibliography	45

CHAPTER 1

Introduction

Duplicates in image databases are not uncommon. There are often several scans or photographs of one and the same work of art. Filtering reproductions of the same original from these databases is a very resource-intensive task. Several employees have to sift through all the material and then decide which reproductions show the same original. The ARTigo project at Ludwig Maximilians Universität Munich uses a large number of users to categorise these images using so-called social tags mainly assigned by laypersons.

The question arises whether the differences between the reproductions are reflected in the tags. Furthermore, the question arises whether, if very similar tags are assigned to different reproductions, this can also be used to identify reproductions that show the same original.

In the scope of this bachelor thesis, an algorithm was developed which determines whether two reproductions represent the same original by comparing the tags assigned to them. For this purpose, a small subsample of the ARTigo database was used, which only contains reproductions of which it is already known that they show the same original. The tags of each reproduction are first generalised to make them more comparable. For this purpose, the Levenshtein neighbours of each tag that has been assigned to a resource are determined. A spelling check is then performed on these neighbours. The frequencies of the other tags are then summed up with the frequency with which the correctly spelled tag was assigned to the resource. This filters out any typos or misspellings and eventually makes it easier to compare the tags of the resources. Afterwards, a matrix is formed which contains all reproductions and tags. The values of the matrix are the frequencies with which a tag has been assigned to a reproduction. A principal component analysis is then applied to this matrix. The result of this is then multiplied by the original matrix. Finally, the cosine distance of the reproductions to each other is determined.

Since it is already known which reproductions portray the same original, a high match value is to be expected for these. After this analysis, a computer vision algorithm determines to what extent the reproductions actually differ. Ideally, the algorithm confirms the results of the data analysis.

This algorithm, in combination with the ARTigo game, saves researchers from having to filter the reproductions by hand and validate whether they represent the same original. Applied to the entire dataset of the ARTigo project, the algorithm could identify those reproductions that represent the same original based on the similarity of their tags. The

researchers only have to decide in the last instance which of the two reproductions can be removed from the database.

The following figures each show two pictures of which it is already known that they represent the same original.



(a) Katsushika Hokusai, Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji, Museum für Kunst und Gewerbe Hamburg - 1830 [15]



(b) Katsushika Hokusai, Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji, Museum für Kunst und Gewerbe Hamburg - 1830-1831 [16]

Figure 1.1: Two different woodblock prints showing the same motif

These two woodblock prints by Katsushika Kokusai, which can be seen in [Figure 1.1](#) show the same motif, but the horizon of the two pictures differs. The details in these images are not too high. Therefore, it can be expected that rather few different tags have been assigned to the two images. The problem with these two reproductions, however, is that this is a false positive assignment. However, it is not expected that this difference will be detected by the computer vision algorithm.



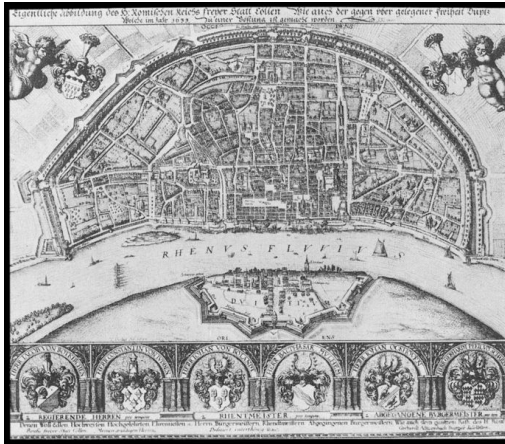
(a) Paul Gauguin, Felsige Meerküste, Göteborgs Konstmuseum - 1886 [21]



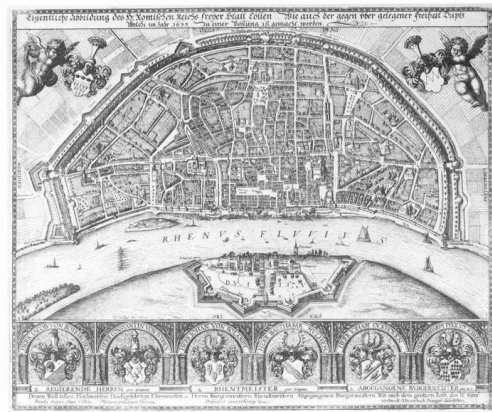
(b) Paul Gauguin, Rochers au bord de la mer, Göteborgs Konstmuseum - 1886 [22]

Figure 1.2: Two reproductions showing the same original but differing in brightness

In [Figure 1.2](#) one can clearly see two different photographs of the artwork "Rocky Sea Coast" by Paul Gauguin. Since the level of detail of the two images is not great, very similar tags are to be expected. The computer vision algorithm should detect that the two images show the same original, with the right image being considerably darker than the left.



(a) Wenzel Hollar, Ansicht von Köln und Deutz aus der Vogelschau, Stadtmuseum Köln - 1635 [39]



(b) Wenzel Hollar, Vogelansicht von Köln und Deutz, Staatliche Kunsthalle Karlsruhe - unknown [40]

Figure 1.3: Two copper engravings depicting the same original

The two copperplate engravings by Wenzel Hollar, which can be seen in [Figure 1.3](#) have a very high level of detail, it is to be expected that many different tags have been assigned to them. It is therefore to be expected that the tag similarity of the two images will not be that high. The computer vision algorithm should recognise that the two images are two reproductions from the same original. It should also recognise that the right image is slightly brighter than the left.



(a) Rembrandt, Christus heilt die Kranken, genannt das Hundertguldenblatt, Staatliche Kunsthalle Karlsruhe - 1601-1649 [26]



(b) Rembrandt, Das Hundertguldenblatt, Albertina Wien - 1647-1649 [27]

Figure 1.4: Different executions of the same theme

The two reproductions in [Figure 1.4](#) both show Rembrandt's "Hundred Guilder Leaf". They differ greatly in their contrast, brightness and general colouring. However, since a very familiar scene is depicted here, it is quite reasonable to assume that the tags assigned will be quite similar. However, the computer vision algorithm should be able to detect a clear difference here.



(a) Bartolomé Esteban Murillo, Einladung zum Pelota - Spiel, The trustees of Dulwich Picture Gallery London - 1670 [3]



(b) Bartolomé Esteban Murillo, Einladung zum Pelota - Spiel, The Trustees of Dulwich Picture Gallery London - 1670 [4]

Figure 1.5: Two reproductions that differ in their colouring

The two reproductions of the artwork "*Invitation to the Pelota - Game*" by Bartolomé Esteban Murillo in Figure 1.5 differ clearly in their colouring. The left image has a clear yellow/red cast. In general, quite similar tags are to be expected. The computer vision algorithm should be able to show clear differences.



(a) Utagawa Hiroshige, Die Komagata-Halle und die Azuma-Brücke, Blatt 62 aus der Serie: 100 berühmte Ansichten von Edo, Museum für Kunst und Gewerbe Hamburg - 1857 [34]



(b) Utagawa Hiroshige, 100 berühmte Ansichten von Edo, Institution not known 1856-1858 [33]

Figure 1.6: Two different woodblock prints showing the same motif

These two reproductions, shown in [Figure 1.6](#) made by Utagawa Hiroshige, both from the series "100 famous views of Edo", are clearly two different woodblock prints. Since there is a significant difference in colour, it is to be expected that these differences will also be reflected in the tags annotated by the crowd.

In the following chapter, the ARTigo project is introduced. It also illustrates how the data set used in this bachelor thesis has been created.

The following chapter presents some of the work and tools that were necessary to make this thesis possible.

Chapter 4 shows which data was selected as a basis for the algorithm. It also explains why these data was chosen and why it had to be cleansed again.

Chapter 5 then describes how the existing data has been analysed and how the algorithm decides which reproductions represent the same original.

Chapter 6 then follows with the colour subtraction of the images. This is done because it is possible to quantify how good the analysis or even the assignment of the tags was.

Finally, there is an outlook on what could not be realised in this thesis due to time constraints, but which can help the developed algorithm to become even more accurate. In addition, there is an outlook on what this algorithm could be used for.

The ARTigo project [24] at LMU Munich, an interdisciplinary project of the Institutes for Art History and for Informatics, uses various games which allow the user to playfully participate in art history. One of the games shares its name with the project and web gaming platform. In ARTigo, two players are randomly selected. They are then shown 5 reproductions of a work of art, which can be a scan or a photograph, for example. These reproductions are randomly selected initially from the Artemis image database of the Ludwig Maximilians Universität Munich. By the time other image databases joined ARTigo. Among them for example the one of "Rijksmuseum". The image database of the "Museum für Kunst und Gewerbe Hamburg", for example, was added through a cooperation. For each picture, one has 60 seconds to annotate the features he is seeing. Both players gain points for each identical tag. It is therefore advisable to annotate the most prominent features of the picture in order to collect as many points as possible. ARTigo relies on so-called social tags, the reason for this being useful in art history is explained in the next section. The method used in ARTigo is based on that of Luis von Ahn, who developed a game¹ that makes it possible to transfer the involvement of laypersons in the creation process so successfully practised in the online encyclopaedia Wikipedia to the field of image annotation while at the same time controlling misuse. [9]

In each run, the players get to see five pictures. For each picture, they have one minute to enter keywords that they recognise in the picture and that they think the other player also recognises. So the limited time puts them under some time pressure. Among other things, this is also reflected in the typos of some tags. The challenges that come along with this are discussed in chapter 4

2.1 What are social tags and why are they used by ARTigo?

Social tagging refers to the free, i.e. arbitrary, assignment of keywords (tags) by a group of people, whereby they explicitly do not have to be experts. The tags assigned in each case are aggregated and thus form a keyword index. [1] The great advantage of social tagging is therefore that the tagging is done by a large interdisciplinary crowd. This group consists mainly of laypersons, in the case of ARTigo precisely not of studied art scholars.

¹<http://www.gwap.com/gwap/gamesPreview/espgame/>

Although this does not mean that there are no experts to be found in this group. ARTigo uses these social tags without quality control of the input. Players can freely choose any number of terms to describe a picture. Thus, both form aspects and semantic content aspects can appear in the created keyword catalogue.^[9]

2.2 Data creation on an example

In Figure 2.1 you can see a running ARTigo game. Shown directly below the picture is a text field in which the player enters the tag he can recognise in the picture. Once he has entered the tag, he presses the button next to it to assign the word respectively these words to the picture. At the left edge the player can see which terms he has entered and how many points he has received for them. If the entered tag does not yet exist in the database, a new one is created and a new entry is created in the Tagging relation, which receives this new tag, this image and the number 1 as the frequency for the first time. If the entered tag already exists, the tagging relation is checked to see if this tag has already been assigned to this image. If this is the case, the frequency of this entry in the database is increased by one. If this is not the case, a new entry is created in the tagging relation as described above.

Above the word list, the player can see the time he has left to assign more tags to this picture.



Figure 2.1: A running ARTigo game (Image: Paul Cézanne, Olympia, 1875/1877, New York (New York), Collection Stern)

2.3 Data structure

As already mentioned in the previous section, the tags that a player gives to an image are stored in a database. The frequency with which these tags have been assigned to a particular resource is essential for this thesis, as the algorithm that has been developed needs to know the frequency with which a particular tag has been assigned to a particular resource. This information can then be used to decide how well the tags of two resources match.

The ARTigo data model (Figure 2.2) used for this thesis is engineered as follows:

Tag:

This relation reflects an actually assigned tag. This tag can be identified by a unique id. Name describes exactly the word that was entered by a player. Language gives information about the language in which the tag was written. In addition to German, English and French are also available.

Artist:

This relation contains all the different artists who can be assigned to their artworks on the basis of their id.

Resource:

Describes an image from the database. Id again stands for the unique identification number. The remaining part contains further information about the painting, which is not relevant for this thesis.

Tagging:

This relation contains the actual assignments of a tag to a resource and the frequency of annotation. Resource.id refers to the image and tag_id to the assigned tag. Frequency describes how often this tag has been assigned to this image.

Relation:

There are two images here which are knowingly reproductions of the same original. The entry in the type field is also important. This can either be "variant of" or "part of". In the case of variant of, the two images differ in colouring, contrast, brightness or similar. In the case of part of, one of the pictures only shows a section of what can be seen in the other picture.

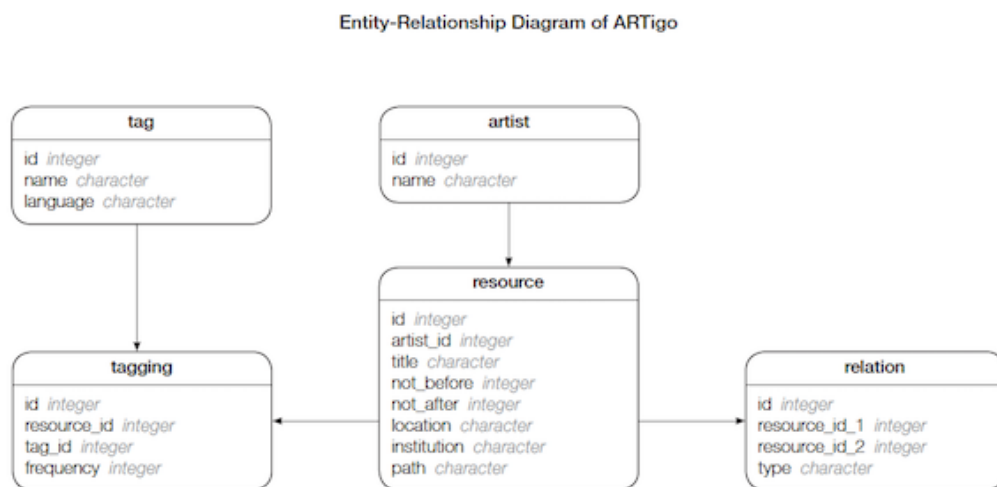


Figure 2.2: ER diagram of the 2018 data structure [5]

CHAPTER 3

Related Work

In her article "Social tagging als Methode zur Optimierung kunsthistorischer Bilddatenbanken – eine empirische Analyse des ARTigo-Projekts" [9], Laura Commare writes about how the ARTigo project uses social tags to help ensure that images in a database are tagged with high quality by a large group of art experts and laypeople. Based on this article, it can be assumed that it is possible to develop an algorithm that can eventually group reproductions that show the same original based on the tags used.

In their article "The Computer as Filter Machine: A Clustering Approach to Categorize Artworks Based on a Social Tagging Network" [32], Stefanie Schneider and Hubertus Kohle present the application of a clustering algorithm that uses crowdsourced annotations given by laypersons to segment artworks into groups.

In her doctoral thesis entitled "Art History Calculated: Interdisciplinary image data analysis of crowd-sourced annotations" [30], Sabine Scherz writes about the possibilities offered by computer games, such as ARTigo, to tag art data benches with keywords.

Writing a tool that makes it possible to carry out the data analysis would not have been possible within the scope of this bachelor thesis. For the data analysis, the project R [25] and some libraries developed for it were used. This includes the package *vwr* [19], which contains the *levenshtein.neighbors* function, with which it was possible to eliminate the typing and spelling errors of the assigned tags. The package *dplyr* [41] was used to search the data more easily.

In her article "Rot rechnen" [23] Waltraud von Pippich writes about the software Redcolor-Tool, which is able to assign a certain red value to any colour. She also addresses the fact that red in images can be interpreted as a symbol of domination and power. This illustrates how much colouring can influence human perception. On this basis, it is interesting for this bachelor thesis to analyse to what extent the resources examined differ in their colouring and whether conclusions can be drawn about the correspondence of the assigned tags.

For the computer vision part, openCV [6] was used. OpenCV is a tool that, among other things, allows you to read in images, create feature vectors of these images and compare them with each other, and calculate colour differences.

In his paper entitled "Distinctive Image Features from Scale-Invariant Keypoints". [20], David G. Lowe presents a method for extracting distinctive invariant features from images. With these so-called keypoints it is then possible to compare images with each other and to make a statement as to whether they represent the same thing.

Stefanie Schneider describes in her lecture entitled "*Über die Ungleichheit im Gleichen. Erkennung unterschiedlicher Reproduktionen desselben Objekts in kunsthistorischen Bildbeständen*" [2] (pp. 92-93), how the SIFT algorithm was used to identify digital reproductions in a database that represent the same original.

Stefanie Schneider has written another essay on the application of the SIFT algorithm. This can be found in the book entitled "*Das digitale Objekt – Zwischen Depot und Internet*", published by the Deutsche Museum. The title of the essay is "*Paare aufdecken. Das digitale Bild in historischen Inventaren*" [31]. Stefanie Schneider reports on how it is possible to use the SIFT algorithm to link so-called *Near-duplicates* resp. *Near-replicas*, which is the description for images in a database that show the same original.

Cleansing the ARTigo dataset

The ARTigo dataset consists of 6.805 images, 8.706 relations, and 4.654.646 taggings. This is a quiet large dataset. In data analysis, it is common to eliminate superfluous data and, if possible, improve the significance of the remaining data. This thesis is limited to the taggings of the resources, which can be found inside the relation called *relation*.

This chapter describes how the tags not needed for the subsample of the ARTigo dataset were eliminated. Chapter 2 described, among other things, the advantages of social tagging. Besides those advantages, they also cause some problems for the data analysis: The fact that the user is not restricted in his or her input leads to multiple typos and spelling mistakes, which is partly due to the limited time available to him or her for assigning tags.

Because of that, the Levenshtein distance is then applied to the remaining tags to group similar ones. Within these groups, each tag is then subjected to a spelling check, which is realised using web scraping. At the end of the process, the frequencies of the misspelled tags are added to the frequency of the correctly spelled tag. When comparing the tags later, this ensures that misspelled tags do not affect the result of the algorithm.

The analysis is based on a dataset from November 15th 2018 from Open Data LMU (Figure 2.2).

4.1 Merging and cleansing the data

This thesis focuses on digital reproductions of the same artworks, which may differ in colour, contrast, brightness or similar. These images are stored in pairs in the relation "relation" and have the value "variant of" in the column "type".

The analysis of the taggings is limited to the German-language tags, which are identified by the value "de" within the "language" column.

Initially, there are 8,706 relations containing 6,805 different reproductions.

In addition to reproductions of the type "variant of", there are also reproductions of the type "part of". These reproductions differ in the section they show of the original work of art.

After eliminating every relation that is not of the type "variant of", 5,614 relations and 4,839 different images remain.

In addition to the German-language tags, there are also English and French tags. For this thesis, only the 409,130 of the total 454,996 taggings are of interest whose assigned tag is in German.

4.2 Correcting misspelled tags

As described earlier in [chapter 2](#) the assigned tags are so-called social tags. While these bring some advantages as already mentioned, they also pose some major challenges for data processing. The most common one are misspelled words or typos.

In order to find out which tags could possibly refer to the same word, the Levenshtein algorithm resp. the Levenshtein distance, which is very common in spell-checking, was used. In R, this algorithm is located inside the package *vwr* [\[25\]](#) [\[19\]](#).

The Levenshtein distance returns a result that describes how much it costs to convert the source word into the target word. If the source word and the target word are identical, the Levenshtein distance returns the value 0. This also applies to two empty words.

The algorithm works as follows: Given are the tags *adler* and *adle*. The algorithm should now calculate the distance between the words. First the length of the source and target word is required.

$m := \text{length}(\text{adler}) = 5 = \text{source}(j);$

$n := \text{length}(\text{adle}) = 4 = \text{target}(i);$

Now a matrix with the *distanced* $[n + 1, m + 1]$ needs to be created. Since the starting value of the matrix is 0, each string is extended by one.

To calculate the Levenshtein distance, the following loop is run: [\(Figure 4.1\)](#)

```

0  for(i = 0; i < n; i++)
1  {
2      for(j = 0; j < m; j++)
3      {
4          distance[i, j] = MIN(
5              distance[i-1, j] = insertion_costs(target), // insertion costs 1
6              distance[i-1, j-1] = exchange_costs(source, target), // exchanging are 2
7              distance[i, j-1] = delete_costs(source) // deletion costs 1
8          )
9      }
10 }
11
```

Figure 4.1: The Levenshtein Algorithm [\[10\]](#)

There are 3 operations that can be performed per letter.

1. If a letter has to be inserted: $d[i-1, j]^1 + 1$
2. If a letter has to be exchanged: $d[i-1, j-1]^2 + 2$
3. If a letter has to be deleted: $d[i, j-1]^3 + 1$

The operation that returns the minimum value is always selected.

$\min(d[i-1, j], d[i-1, j-1], d[i, j-1])$

If both letters at $d[i, j]$ are the same, no operation costs occur. The lowest value of those three cells is then selected.

¹(the value found underneath the current cell)

²(the value found diagonally left of the current cell)

³(the value found on the left hand side of the current cell)

First Step

In the first step, zero is inserted where the cross sign is in both column and row.

$$d[0,0] = 0$$

Second Step

In the second step, the first column and the first row are filled in ascending order from zero. The last letter of each word thus contains its length.

Third Step

Since the first two letters match, the smallest number in this case is $d[i-1, j-1]$. For the following letters, the 3rd operation is the most favourable.

Fourth Step

The first two letters are A and D. The most minimal operation is number 1.

$$d[i-1, j] = 0 + 1 = 1$$

This is followed by the letters D and D, which is why 0 is inserted from position $d[i-1, j-1]$. For the following letters, operation 3 is again the minimum.

Fifth Step

For the first two letters, operation 3 is the minimum solution. Since source and target match at $d[i, j]$ and 0 is found at $d[i-1, j-1]$, it is entered in this cell. For the following letters, operation 1 is again the minimum solution.

Sixth Step

Up to the position $d[5,5]$, operation 3 is always the minimum. At $d[5,5]$ the letters match again and at $d[i-1, j-1]$ a 0 can be found again. For the last letter, operation 1 is the most minimal.

$\begin{bmatrix} \mathbf{R} & & & & & \\ \mathbf{E} & & & & & \\ \mathbf{L} & & & & & \\ \mathbf{D} & & & & & \\ \mathbf{A} & & & & & \\ \# & 0 & & & & \\ & \# & \mathbf{A} & \mathbf{D} & \mathbf{L} & \mathbf{E} \end{bmatrix}$	$\begin{bmatrix} \mathbf{R} & 5 & & & & \\ \mathbf{E} & 4 & & & & \\ \mathbf{L} & 3 & & & & \\ \mathbf{D} & 2 & & & & \\ \mathbf{A} & 1 & & & & \\ \# & 0 & 1 & 2 & 3 & 4 \\ & \# & \mathbf{A} & \mathbf{D} & \mathbf{L} & \mathbf{E} \end{bmatrix}$	$\begin{bmatrix} \mathbf{R} & 5 & 4 & & & \\ \mathbf{E} & 4 & 3 & & & \\ \mathbf{L} & 3 & 2 & & & \\ \mathbf{D} & 2 & 1 & & & \\ \mathbf{A} & 1 & 0 & & & \\ \# & 0 & 1 & 2 & 3 & 4 \\ & \# & \mathbf{A} & \mathbf{D} & \mathbf{L} & \mathbf{E} \end{bmatrix}$
$\begin{bmatrix} \mathbf{R} & 5 & 4 & 3 & & \\ \mathbf{E} & 4 & 3 & 2 & & \\ \mathbf{L} & 3 & 2 & 1 & & \\ \mathbf{D} & 2 & 1 & 0 & & \\ \mathbf{A} & 1 & 0 & 1 & & \\ \# & 0 & 1 & 2 & 3 & 4 \\ & \# & \mathbf{A} & \mathbf{D} & \mathbf{L} & \mathbf{E} \end{bmatrix}$	$\begin{bmatrix} \mathbf{R} & 5 & 4 & 5 & 2 & \\ \mathbf{E} & 4 & 3 & 4 & 1 & \\ \mathbf{L} & 3 & 2 & 2 & 0 & \\ \mathbf{D} & 2 & 1 & 0 & 1 & \\ \mathbf{A} & 1 & 0 & 1 & 2 & \\ \# & 0 & 1 & 2 & 3 & 4 \\ & \# & \mathbf{A} & \mathbf{D} & \mathbf{L} & \mathbf{E} \end{bmatrix}$	$\begin{bmatrix} \mathbf{R} & 5 & 4 & 5 & 2 & \mathbf{1} \\ \mathbf{E} & 4 & 3 & 4 & 1 & 0 \\ \mathbf{L} & 3 & 2 & 2 & 0 & 1 \\ \mathbf{D} & 2 & 1 & 0 & 1 & 2 \\ \mathbf{A} & 1 & 0 & 1 & 2 & 3 \\ \# & 0 & 1 & 2 & 3 & 4 \\ & \# & \mathbf{A} & \mathbf{D} & \mathbf{L} & \mathbf{E} \end{bmatrix}$

In the upper right corner of the matrix, the distance between the two words is now shown.

So "adler" and "adle" are class one neighbours. 14 10

The package *vwr* also contains the so-called *levenshtein.neighbours* function. This function returns a list, which is sorted according to the neighbourhood classes, i.e. 1, 2, 3, etc. Class 0, which contains the identical words as described above, is not included.

In order to identify possibly matching tags, the entries in the list are only of interest if they belong to the neighbourhood class of 1. The algorithm does not add the input word to the list. Therefore, this must be done manually afterwards.

To determine which tags were the ones spelled correctly in this list, a query in a German online encyclopaedia was made for each tag. The frequencies of the tags for which no official entry could be found in the dictionary were summed up and assigned to the tag from the list that had the highest frequency. The original list was then filled with the tags again. Tags for which a dictionary entry was found, but which did not have the highest frequency, have been inserted into the original list unchanged.

This process eliminated 33,021 tags. The total number of tags to be compared is now at 377,109.

After data cleaning, 4,587 different images and a total of 38,096 different tags remained. The information on how often a tag was assigned to a resource can be displayed in a $4,587 \times 38,096$ dimensional matrix. The cells of the matrix contain the frequency with which a tag was assigned to a resource. However, since only a fraction of the total number of assigned tags was assigned to a resource, the matrix consists mostly of zeros. For this reason, a *sparseMatrix* is used, which is designed to consist mostly of zeros. After the *sparseMatrix* has been filled, the cosine similarities of the resources are calculated. Then a list is created containing all known reproductions showing the same original and its cosine similarity. The first section of this chapter demonstrates how the required *sparseMatrix* was created. The second section first explains why cosine similarity was used to determine the tag similarity of the resources. This is followed by the actual calculation of the cosine similarity. The third section of this chapter then shows seven examples. These examples each contain two reproductions. The first six examples are the same as those shown in the introduction. The seventh example shows the two reproductions with the worst cosine similarity.

After these seven examples, which were already known to represent the same original, three more images follow, which have been newly discovered through cosine similarity.

5.1 Creation of the *sparseMatrix*

The creation of the *sparseMatrix* in *R* is shown in [Figure 5.1](#). All taggings that remain after data cleansing must first be read in. The required values are *resource_id*, *tag_id* and *frequency*. The rows and columns must be in ascending order. Therefore, the ids of resource and tag must first be converted as a factor (lines 7 & 8). It is important to save which factor belongs to which id in order to be able to draw conclusions about the actual data later (line 9).

In line 10, the *sparseMatrix* is eventually filled. The function argument *i* defines the rows of the matrix. The argument *j* defines the columns. Finally, the frequencies are passed to the argument *x*. Each cell has the value of the frequency with which the tag in column *j* has been assigned to the resource in column *i*. If a tag has not been assigned to a resource, the value at this point is 0. To clean up the *sparseMatrix* at the end, the function *drop0* is applied, which removes all zero values from the matrix.


```

0 library(data.table)
1 library(Matrix)
2 // Resource_id, tag_id and frequency of all resources cleansed in chapter 4
3 data <- readRDS("variantRelationTaggings.Rds")
4 resource.ids <- data$resource_id
5 tag.ids <- data$tag_id
6 frequencies <- data$frequency
7 resource.ids.factored <- resource.ids.as.factor(resource.ids)
8 tag.ids.factored <- tag.ids.as.factor(tag.ids)
9 factor.id.mapping <- data.table(resource_id = resource.ids, resource_id_
  factored = resource.ids.factored, tag_id = tag_id, tag_id_factored = tag.ids.
  factored, frequency = frequencies)
10 matrix <- sparseMatrix(i = factor.id.mapping$resource_id_factored, j = factor.
  id.mapping$tag_id_factored, x = frequencies)
11 matrix <- drop0(matrix)
12

```

Figure 5.1: Creation of the sparseMatrix

5.2 Computing the cosine similarity

In chapter 2 of their book "Data Mining (Third Edition)" [11], Jiawei Han, Micheline Kamber and Jian Pei explain why cosine similarity is so well suited to comparing documents. This is because the frequencies of how often words show up in each document are compared. Zero matches are ignored, since it is not important which words do not appear in the documents.

Cosine similarity is often used to determine similarities between documents. The principle behind this is as follows:

For each word, the number of times it occurs in the document is stored. Then the frequencies of the words that occur in both documents are compared. The better the frequencies of these words match, the higher the cosine similarity.

In Figure 5.2 you can see the formula used to calculate the cosine similarity. x and y here stand for the vectors of the two documents. $||x||$ and $||y||$ are the Euclidean norm of vector x and y , which is given as $\sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$. The closer the value of $\text{sim}(x, y)$ approaches 1, the more equal the documents are. [11]

Since the ARTigo data is already available in this form, the cosine similarities between the resources can be calculated by an algorithm. The result of this algorithm will be a $n \times n$ matrix whose values in the diagonal will always be 1, because on the diagonal the similarity to itself is calculated and therefore always 1. The matrix will also be mirrored. This implies that the value at $\text{matrix}[1, 2]$ will be identical to the value $\text{matrix}[2, 1]$. The algorithm is shown in Figure 5.3. Two parameters are passed to the function, $x_i[1]$ represents the row of the new matrix, $x_i[2]$ represents the column of the new matrix. Since it is known that the matrix is mirrored at the diagonal and the value of the diagonal is always 1, the performance of the algorithm can be improved considerably, since the cosine similarity only has to be calculated for the lower half of the matrix. Therefore, it is first checked whether $x_i[1] > x_i[2]$. If this is the case, the cosine distance of the two resources that are in series $x_i[1]$ or $x_i[2]$ in the origin matrix is calculated. If $x_i[1] == x_i[2]$, it is just on the diagonal of the new matrix. Based on this, the value 1 can simply be returned. If neither of the two is true, then one is in the upper half of the matrix and the value 0 can be returned.

The algorithm took a total of 11 minutes and 29 seconds to calculate the 4587×4587 matrix.

In the next step, the cosine distances of the already known reproductions are extracted from the matrix.

$$\text{sim}(x,y) = \frac{x \cdot y}{\|x\| \|y\|}$$

Figure 5.2: The formula to calculate the cosine similarity

```

0   calculate.cosinus.similarity <- function(xi) {
1   if (xi[1] > xi[2]) {
2   A = pca.matrix[xi[1],]
3   B = pca.matrix[xi[2],]
4   similarity = sum(A*B)/sqrt(sum(A^2)*sum(B^2))
5   return(similarity)
6   } else if (xi[1] == xi[2]) {
7   return(1)
8   } else {
9   return(0)
10  }
11  }
12  n <- nrow(pca.matrix)
13  cmb <- expand.grid(i=1:n, j=1:n)
14  similarity.matrix <- matrix(apply(cmb,1,calculate.cosinus.similarity),n,n)
15

```

Figure 5.3: Algorithm to calculate the cosine similarity matrix

5.3 Evaluation of the calculated cosine similarities

In [Figure 5.4](#) one can see that 94.29% of the already known reproductions depicting the same original have a cosine similarity of at least 0.5. Another 85.3% of the images have a cosine similarity of at least 0.6. More than half of the known reproductions, namely 63.64%, have a cosine similarity of at least 0.7. On the other hand, only 26.11% of the reproductions still have a cosine similarity of 0.8. 2.46% of the images have a cosine similarity of at least 0.9.

The following are 7 examples of reproductions known to show the same original.

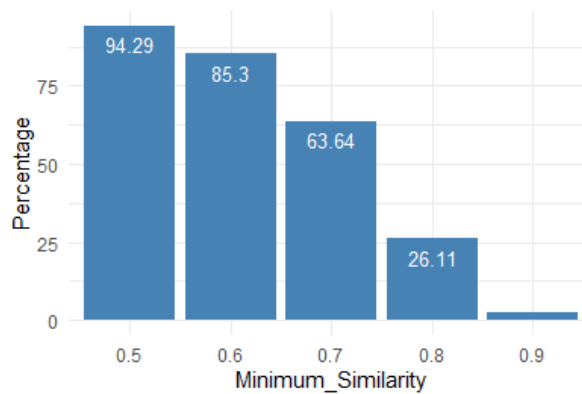


Figure 5.4: The cosine similarity distribution



(a) Katsushika Hokusai, Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji, Museum für Kunst und Gewerbe Hamburg - 1830 [15]



(b) Katsushika Hokusai, Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji, Museum für Kunst und Gewerbe Hamburg - 1830-1831 [16]

Figure 5.5: Cosine similarity of 1

The two reproductions seen in Figure 5.5 have been found to have the highest possible cosine similarity. However, when viewed with the human eye, it is doubtful whether these two images are reproductions of the same original at all. The foreground of the pictures is identical, but the horizon of the two pictures is completely different.

However, looking at both sets of tags, it is noticeable that both resources were actually assigned exactly the same tags. And since each tag has been assigned to the resource exactly once, there is a cosine similarity of 1. It can also be assumed that the computer vision algorithm, which will be discussed in the next chapter, will recognise the two images as identical. However, differences should be detected by the algorithm during the subsequent colour examination.



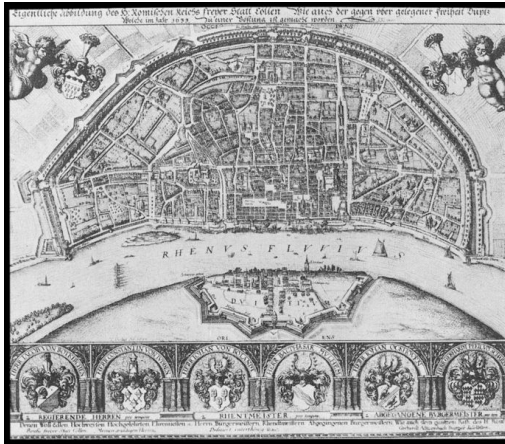
(a) Paul Gauguin, Felsige Meerküste, Göteborgs Konstmuseum - 1886 [21]



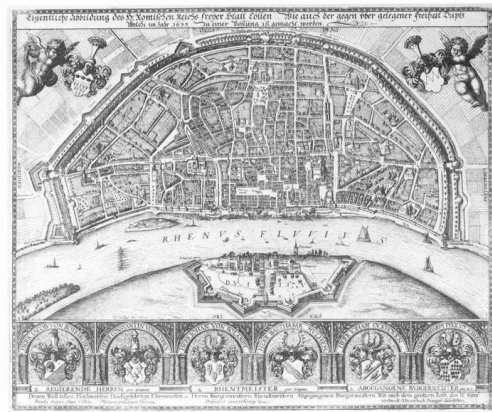
(b) Paul Gauguin, Rochers au bord de la mer, Göteborgs Konstmuseum - 1886 [22]

Figure 5.6: Cosine similarity of 0.9

The two images in Figure 5.6 have a cosine similarity of 0.9. It can be seen that the right image is significantly darker than the left one. In contrast to the images in Figure 5.5 these images were assigned significantly more tags. On the other hand, the frequencies with which the tags were assigned to the images also vary here.



(a) Wenzel Hollar, Ansicht von Köln und Deutz aus der Vogelschau, Stadtmuseum Köln - 1635 [39]



(b) Wenzel Hollar, Vogelansicht von Köln und Deutz, Staatliche Kunsthalle Karlsruhe - unknown [40]

Figure 5.7: Cosine similarity of 0.85

The images seen in [Figure 5.7](#) have a cosine similarity of 0.85. Only 29 different tags were assigned to the left image. The image on the right, though, was given 106 different tags by the crowd. This is where the advantage of cosine similarity can be seen. Since only the term frequency of those tags that both images have in common is compared, a cosine similarity of 0.85 could be achieved here.



(a) Rembrandt, Christus heilt die Kranken, genannt das Hundertguldenblatt, Staatliche Kunsthalle Karlsruhe - 1601-1649 [26]



(b) Rembrandt, Das Hundertguldenblatt, Albertina Wien - 1647-1649 [27]

Figure 5.8: Cosine similarity of 0.76

As expected in the introduction, the tag correlation is relatively high, although on the right image less can be detected due to overexposure.



(a) Barolomé Esteban Murillo, Einladung zum Pelota - Spiel, The trustees of Dulwich Picture Gallery London - 1670 [3]



(b) Bartolomé Esteban Murillo, Einladung zum Pelota - Spiel, The Trustees of Dulwich Picture Gallery London - 1670 [4]

Figure 5.9: Cosine similarity of 0.67

The cosine similarity of these images is somewhat worse than expected. Nearly equal numbers of tags were assigned to each of the images.



(a) Utagawa Hiroshige, Die Komagata-Halle und die Azuma-Brücke, Blatt 62 aus der Serie: 100 berühmte Ansichten von Edo, Museum für Kunst und Gewerbe Hamburg - 1857 [34]



(b) Utagawa Hiroshige, 100 berühmte Ansichten von Edo, Institution not known - 1856-1858 [33]

Figure 5.10: Cosine similarity of 0.1

The images shown in Figure 5.10 only share a cosine similarity of 0.1. The two images show approximately the same motif. However, they clearly do not show the same original.



(a) Grotteske Figuren mit Knorpelwerk, Blatt aus dem 'Neuw Grotteßken Buch', Museum für Kunst und Gewerbe Hamburg - 1610 [8]



(b) Christoph Jamnitzer, Gevleugelde draak, Rijksmuseum Amsterdam - 1573 [7]

Figure 5.11: Cosine similarity of 0.007

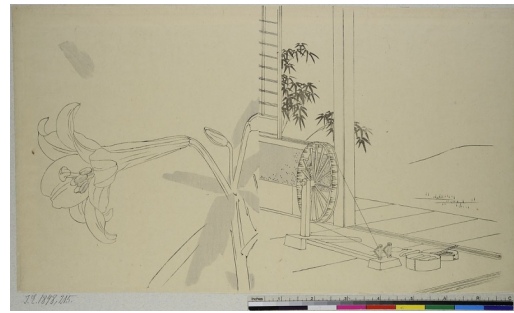
These two photographs of Vincent van Gogh's "*Grotesque Figures with Cartilage*", to be seen in [Figure 5.11](#) are the reproductions with the worst cosine similarity. It is striking that the two images were assigned different numbers of tags. In addition, each tag was annotated a maximum of once in the right image.

After analysing the cosine similarities of already known reproductions representing the same original, it can be said that the threshold value of the cosine similarity of tags of two images should be set between 0.5 and 0.6 in order to pre-sort images that could possibly show the same original.

The following three examples are not already known reproductions that depict the same original. All have a cosine similarity between 0.6 and 1.



(a) Utagawa Toyokuni I., Aus: Holzschnittbuch über zeitgenössische Kleidung, Museum für Kunst und Gewerbe Hamburg - 1802 [\[35\]](#)



(b) Utagawa Toyokuni I., Aus: Holzschnittbuch über zeitgenössische Kleidung, Museum für Kunst und Gewerbe Hamburg - 1802 [\[36\]](#)

Figure 5.12: Cosine similarity of 1

The tags of these two woodblock prints ([Figure 5.12](#)) by Utagawa Toyokuni I, which belong to the dataset of the Museum für Kunst und Gewerbe Hamburg and are titled "*From: Woodblock Book on Contemporary Clothing*", have a cosine distance of 1. The two images show the same scene, although they differ in colour. This example shows that the crowd has recognised the similarity of the images, which stands for a very good tag quality. It is to be expected that the computer vision algorithm will classify these two images as very similar. It should also recognise that the images differ in their colouring.



(a) Kawanabe Kyosai, Die Reise nach Westen:
Das Holen der Sutraschriften aus Indien, Mu-
seum für Kunst und Gewerbe Hamburg - 1630
29



(b) Kawanabe Kyosai, Die Reise nach Westen:
Das Holen der Sutraschriften aus Indien, Mu-
seum für Kunst und Gewerbe - 1629 28

Figure 5.13: Cosine similarity of 0.64

The two pictures from [Figure 5.13](#) both show a self-portrait by Rembrandt. They originate from the data holdings of the Rijksmuseum and the Rijksprentenkabinett in Amsterdam. The cosine similarity of 0.64 indicates that the crowd has recognised some differences, but overall has annotated the two images very similarly. The cosine similarity of the tags of the two reproductions could clearly show that they are very similar. The computer vision algorithm, should recognise that they do not show the same original.



(a) Vincent van Gogh, Gipsmodell eines kniendne Mannes, Van Gogh Museum Amsterdam - 1806 [13]



(b) Vincent van Gogh, Gipsmodell eines kniendne Mannes, Van Gogh Museum Amsterdam - 1805 [12]

Figure 5.14: Cosine similarity of 0.73

The paintings in [Figure 5.14](#) are clearly reproductions of the same original, although the way they were painted differs. These are two paintings by Joseph Anton Koch entitled "*Heroic Landscape with Rainbow*". [Figure 5.14a](#) was painted around 1806 according to the ARTigo database, but unfortunately it is not noted by which institution this painting was provided. [Figure 5.14b](#) was created around 1805 and was provided by the Staatliche Kunsthalle Karlsruhe.

A cosine similarity of 0.73 also proves here that the crowd has recognised and annotated the similarities of both images. The computer vision algorithm should recognise that the two images are quite similar. It should also show that the images are very different in colour.

In this chapter, all known reproductions are analysed using computer vision. Using the SIFT algorithm [20] provided by OpenCV [6]. The SIFT algorithm can calculate an individual feature vector for each image. How this works is described in section 6.1. The feature vectors of two images, are checked for similarity, can then be calculated using a method described by Davig G. Lowe describes in his paper [20]. Depending on how many matching feature vectors are found, this can then be used as a guide to judge whether the two images being compared are reproductions of the same original. To have another indicator for the similarity of two images, the colour channels of both images are then subtracted. The general colour difference between the two images can then also be used to draw conclusions about their equality.

6.1 Feature detection

In his paper "Distinctive Image Features from Scale-Invariant Keypoints" [20], David G. Lowe presents a method that extracts distinctive invariant features from images, which can then be used for reliable comparison between different views of an object or a scene. The method works as follows:

The first step is to detect scale-space extrema (keypoints). To do so, first locations and scales have to be identified, that can be repeatably assigned under differing views of the same object. This is done using the Gaussian function. "The scale space of an image is defined as function, $L(x, y, \sigma)$, that is produced from the convolution of a variable-scale Gaussian, $G(x, y, \sigma)$, with an input image, $I(x, y)$:"

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

where $*$ is the convolution operation in x and y , and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

" [20]

Scale spaces are computed by convolving the initial image for each octave of scale space with Gaussians to produce a set of scale space images. The different scale spaces of an image can be imagined as blur views of differing intensity of the initial image. Local maxima

and minima are then calculated by comparing each sample point to its eight neighbors in the current image and nine neighbors in the scale above and below. Only the points that are either smaller or larger than all their neighbours are selected as local extrema resp.

Keypoints. [20]

After all keypoints are selected, a detailed fit to the nearby data for location, scale and ratio of principal curvatures is performed on each of them. *"This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge."* [20]

After that, all keypoints are assigned a consistent orientation based on local image properties. The later assigned keypoint descriptor will be represented relative to this orientation. Throughout all these parameters a 2D coordinate system is created, in which the local image region is described. *"The next step is to compute a descriptor for the local image region that is highly distinctive yet is as invariant as possible to remaining variations, such as change in illumination or 3D viewpoint"*. [20]

To create this keypoint descriptor, first the gradient magnitude and orientation at each image sample point in a region around the keypoint location has to be computed. The descriptors are then weighted by a Gaussian window and then accumulated into orientation histograms summarizing the contents over 4×4 subregions.

After creating keypoints and descriptors for both images, they have to be compared. This is done by assigning the nearest neighbour of a keypoint in image two to the nearest in image one. However, if only the nearest neighbour is determined, no statement can be made as to whether it is actually a good match. David G. Lowe writes in his paper that the best two matches from image one should be assigned to the keypoint in image two. Next a global threshold value is set, which according to Lowe should be around 0.8. Now the distances of all knn(k nearest neighbour) matches to the keypoint in image two are compared. It is evaluated whether the distance of the first, better match is smaller than that of the second match multiplied by the threshold value. If this is the case, it is a good match. According to Lowe, this measure works well because correct matches must have a significantly smaller distance to their nearest neighbour than to their second best neighbour.

In order to present these good matches in a comprehensible way, the image with the lower number of keypoints is searched for. The percentage of good matches is then calculated on the basis of this value.

6.1.1 Computing the features

As it can be seen in [Figure 6.1](#), both images are first converted to greyscale, as the SIFT algorithm cannot operate on coloured images. Then a new instance of the SIFT algorithm is created. Then the two vectors are initialised, which will contain the keypoints of image 1 and image 2 respectively. The calculation described in [section 6.1](#) is then performed in lines 10 and 11.

```

0  Mat img1_grey, img2_grey, img1_descriptors, img2_descriptors;
1  cvtColor(image1, img1_grey, COLOR_BGR2GRAY);
2  cvtColor(image2, img2_grey, COLOR_BGR2GRAY);
3  // Create SIFT
4  Ptr<SIFT> sift = SIFT::create();
5
6  // Create Keypoints vector
7  vector<KeyPoint> img1_keypoints, img2_keypoints;
8
9  // Compute and detect
10 sift->detectAndCompute(img1_grey, noArray(), img1_keypoints, img1_descriptors)
11 ;
12 sift->detectAndCompute(img2_grey, noArray(), img2_keypoints, img2_descriptors)
13 ;

```

Figure 6.1: Computing keypoints and descriptors with sift

6.1.2 Comparing the features

Figure 6.2 demonstrates, as described in section 6.1, the calculation of the good matches. In line 1, the vector that will later contain the good matches is initialised. One line below, the same happens with the vector that will contain all the knn matches that will later be passed over step by step. In line 5, the so-called matcher is initialised. *DescriptorMatcher::FLANNBASED* is an algorithm provided by OpenCV that will find all knn matches in an efficient time. The function call that results in finding the two best matches from image one per keypoint in image two is done in line 8. *img2_descriptors* describes the descriptors of image two calculated in the previous step. The same applies to *img1_descriptors*. As already mentioned, *knnMatches* references the vector in which the matches are to be stored. The 2 at the end of the function indicates how many k-neighbours the algorithm should determine.

Line 10 then defines the global threshold of 0.8 addressed in section 6.1. Lines 11 – 18 contain the loop that decides whether the match is a good match or not. If $m.distance < ratio_thresh * n.distance$, the better of the two matches, *m*, is added to the vector of good matches.

The calculation of the percentage of good matches is shown in Figure 6.3

```

0 // Initialize vector of good matches
1 vector<DMatch> goodMatches;
2 // Initialize vector of knnMatches
3 vector<vector<DMatch>> knnMatches;
4
5 // Initialize matcher
6 Ptr<DescriptorMatcher> matcher = DescriptorMatcher::create(DescriptorMatcher
  ::FLANNBASED);
7
8 // Compute knnMatches
9 matcher->knnMatch(img2_descriptors, img1_descriptors, knnMatches, 2);
10 // Define Threshold
11 const float ratio_thresh = 0.8f;
12 // Select good matches
13 for (auto & knnMatch : knnMatches)
14 {
15     const auto m = knnMatch[0];
16     const auto n = knnMatch[1];
17     if (m.distance < ratio_thresh * n.distance)
18     {
19         goodMatches.push_back(m);
20     }
21 }
22

```

Figure 6.2: Calculating the good matches

```

0 double min_kp;
1 if (img1_keypoints.size() > img2_keypoints.size())
2 {
3     min_kp = 100.00 / (double) img2_keypoints.size();
4 } else
5 {
6     min_kp = 100.00 / (double) img1_keypoints.size();
7 }
8 kp_percent = min_kp * (double) goodMatches.size();
9

```

Figure 6.3: Calculating the percentage of good matches

6.2 Computation of the colour differences

How do people perceive colours? Probably, the best known colour model is the RGB model, which represents all colours based on the intensity of red, blue and green. However, it is known that humans do not see colour in terms of red, blue and green. A colour model that is closer to the human eye than the RGB model is the L*a*b model. The L*a*b is constructed as follows:

L stands for the lightness. The value is between 0 and 100.

a represents the axis between green and red, whose value can be between -128 and 127.

b represents the axis between blue and yellow, whose value can be between -128 and 127.

This algorithm computes the colour differences of two resources. The corresponding resources and their save paths are stored in a .csv file, which will initially be parsed by the program.

Resource_1_id, it's path, resource_2_id it's path and the cosine_similarity will be extracted. For each resource pair, the percentage match is then first calculated using the feature vectors described in [section 6.1](#). This is followed by the calculation of the colour differences between the two images.

In order to be able to compare the two images pixel by pixel, they must first both be scaled to the same size ([Figure 6.5](#)).

The algorithm ([Figure 6.7](#)) does the following:

Both images will be split up into a *vector* that contains their three colour-channels. Then, for each entry inside the Mat, the three L*a*b-channels are extracted.

In lines 15 and 16 the L-channel of both images is saved. Next the a- and b-channel is extracted and summed by 128. This is because the value of either a or b is between -128 and 127 . By summing the value with 128, the range will be between 0 and 255.

The next step at lines 24 to 26, each channel of the two images will be subtracted. The values at channel a and b again be subtracted with 128 to fit the original range. At the end each channel-vector will be pushed into a result *vector* of `<vector<double>>`, which will be returned.

After that the values of each channel will be summed up to calculate the general difference of the two images ([Figure 6.7](#)). The results will again be stored inside a .csv file to be able to compare the tag differences to the colour differences.

An example showing the calculated keypoints and the good matches of two images can be found in [Figure 6.8](#).

```

0 // images have the same size
1   Size img_1.size, img_2.size, new_size;
2   int new_size_1, new_size_2;
3   img_1.size = img1.size();
4   img_2.size = img2.size();
5   cout << "img1 size: " << img_1.size << endl;
6   cout << "img2 size: " << img_2.size << endl;
7   if (img_1.size.width != img_2.size.width || img_1.size.height != img_2_
size.height)
8   {
9       if (img_1.size.width > img_2.size.width)
10      {
11          new_size_1 = img_1.size.width;
12      } else
13      {
14          new_size_1 = img_2.size.width;
15      }
16      if (img_1.size.height > img_2.size.height)
17      {
18          new_size_2 = img_1.size.height;
19      } else
20      {
21          new_size_2 = img_2.size.height;
22      }
23      new_size = Size(new_size_1, new_size_2);
24      resize(img1, img1, new_size);
25      resize(img2, img2, new_size);
26  }
27
28 // Convert images into L*a*b
29 cvtColor(img1, img1_lab, COLOR_BGR2Lab);
30 cvtColor(img2, img2_lab, COLOR_BGR2Lab);
31

```

Figure 6.5: Resizing the images if necessary and converting their colour into Lab

```

0  string res_1_id, res_2_id, img1_path, img2_path, cosine_similarity, percent-
    conformity, img_1_relative_path, img_2_relative_path;
1      if (element.size() != 6)
2      {
3          cerr << "Element.size() was expected to be 5 but is " << element.size
    () << endl;
4          return EXIT_FAILURE;
5      }
6      // element[0] is ignored because it is the row id
7      res_1_id = element[1];
8      img1_path = "Images/" + element[2];
9      img_1_relative_path = element[2];
10     res_2_id = element[3];
11     img2_path = "Images/" + element[4];
12     img_2_relative_path = element[4];
13     cosine_similarity = element[5];
14     cosine_similarity.erase(cosine_similarity.length() - 1);
15
16     // Define all Mats for image variations
17     Mat img1, img2;
18     Mat3b img1_lab, img2_lab;
19
20     // Read image 1
21     img1 = imread(img1_path);
22
23     // Read image 2
24     img2 = imread(img2_path);
25
26     // Check if both images could be loaded
27     // if not return EXIT_FAILURE
28     if (img1.empty() || img2.empty())
29     {
30         if (img1.empty())
31         {
32             cout << "Could not load image 1" << endl;
33         }
34         if (img2.empty())
35         {
36             cout << "Could not load image 2" << endl;
37         }
38         return EXIT_FAILURE;
39     }
40
41

```

Figure 6.4: Parse the images

```

0      vector<vector<double>>> new_channels = processLabColor(img1_lab, img2_lab);
1      double l_diff = 0.0;
2      for (double l_val : new_channels[0])
3      {
4          l_diff += l_val;
5      }
6      l_diff /= new_channels[0].size();
7      double a_diff = 0.0;
8      for (double a_val : new_channels[1])
9      {
10         a_diff += a_val;
11     }
12     a_diff /= new_channels[1].size();
13     double b_diff = 0.0;
14     for (double b_val : new_channels[2])
15     {
16         b_diff += b_val;
17     }
18     b_diff /= new_channels[2].size();
19
20     // convert diffs to string
21     string l_diff_str, a_diff_str, b_diff_str;
22     l_diff_str = to_string(l_diff);
23     a_diff_str = to_string(a_diff);
24     b_diff_str = to_string(b_diff);
25     // append to csv file
26     write_to_csv(
27         write_csv_path, counter_string, res_1_id,
28         img_1_relative_path, res_2_id, img_2_relative_path,
29         cosine_similarity, percent_conformity, l_diff_str,
30         a_diff_str, b_diff_str
31     );
32

```

Figure 6.6: Resizing the images if necessary and converting their colour into Lab


```

0   vector<vector<double>> processLabColor(Mat &image1, Mat &image2) {
1   vector<Mat> channels;
2   vector<Vec3b> img1_lab_channels, img2_lab_channels;
3   vector<double> l_channel, a_channel, b_channel;
4   int rows, cols;
5   rows = image1.rows;
6   cols = image1.cols;
7   for (int i = 0; i < rows; i++) {
8       for (int j = 0; j < cols; j++) {
9           double img1_l_channel, img2_l_channel, img1_a_channel, img2_a_channel,
10          img1_b_channel, img2_b_channel, new_l_channel, new_a_channel, new_b_channel;
11          Vec3b img1_labPixel(image1.at<Vec3b>(i, j));
12          Vec3b img2_labPixel(image2.at<Vec3b>(i, j));
13          img1_lab_channels.push_back(img1_labPixel);
14          img2_lab_channels.push_back(img2_labPixel);
15          // extract channels
16          img1_l_channel = img1_labPixel[0];
17          img2_l_channel = img2_labPixel[0];
18          // add 128 to a and b channel to get a range from 0 to 255
19          img1_a_channel = img1_labPixel[1] + 128.0;
20          img2_a_channel = img2_labPixel[1] + 128.0;
21          img1_b_channel = img1_labPixel[2] + 128.0;
22          img2_b_channel = img2_labPixel[2] + 128.0;
23
24          // process channels
25          new_l_channel = img1_l_channel - img2_l_channel;
26          new_a_channel = img1_a_channel - img2_a_channel;
27          new_b_channel = img1_b_channel - img2_b_channel;
28          l_channel.push_back(new_l_channel);
29          // subtract 128 from a and b channel to get original range
30          a_channel.push_back(new_a_channel);
31          b_channel.push_back(new_b_channel);
32      }
33  }
34  vector<vector<double>> result;
35  result.push_back(l_channel);
36  result.push_back(a_channel);
37  result.push_back(b_channel);
38  return result;
39  }

```

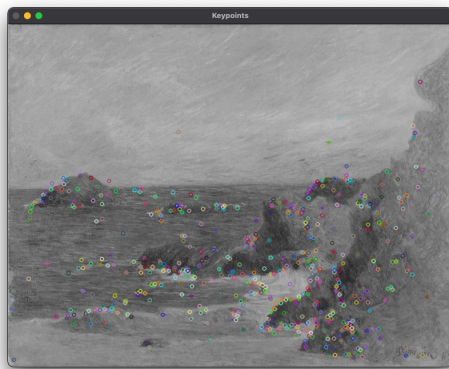
Figure 6.7: The raw calculation of pixel difference



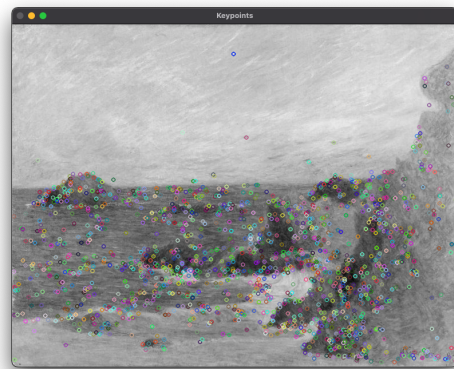
(a) Paul Gauguin, Rochers au bord de la mer, Göteborgs Konstmuseum - 1886 [22]



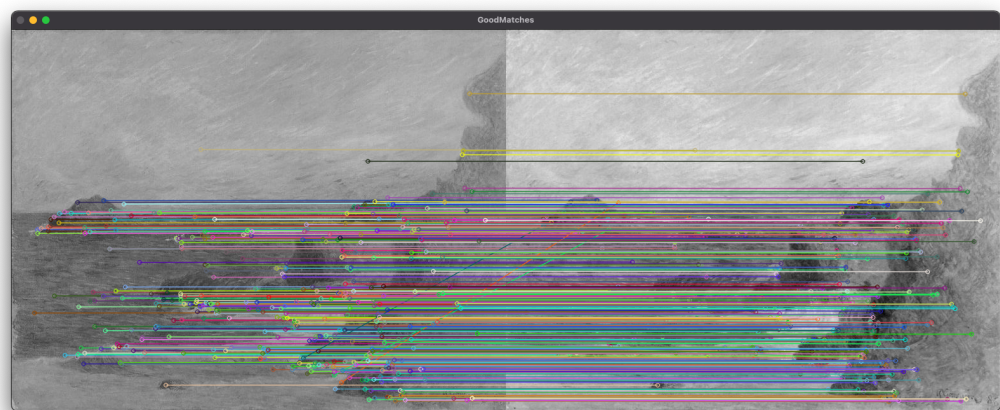
(b) Paul Gauguin, Felsige Meerküste, Göteborgs Konstmuseum - 1886 [21]



(c) Greyscale image with Keypoints drawn of Figure 6.8a



(d) Greyscale image with Keypoints drawn of Figure 6.8b



(e) The computed good matches of both images.

Figure 6.8: Illustration of the SIFT algorithm.

6.3 Results



(a) Katsushika Hokusai, Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji, Museum für Kunst und Gewerbe Hamburg - 1830 [15]



(b) Katsushika Hokusai, Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji, Museum für Kunst und Gewerbe Hamburg - 1830-1831 [16]

Figure 6.9: Cosine similarity of 1

For the two images in Figure 6.9 the computer vision algorithm calculated a match of about 30%. The difference in their brightness value was about 6.1, their difference on the a-axis was about 0.63 and their difference on the b-axis was about 3.49. It was found that both images differ only marginally in brightness, green/red and blue/yellow. Despite these minor differences, only 30% of the matches found could be described as good. Without knowing these images and their cosine similarity beforehand, it can be assumed that they are not reproductions depicting the same original. And that was exactly what was expected in the introduction.



(a) Paul Gauguin, Felsige Meerküste, Göteborgs Konstmuseum - 1886 [21]

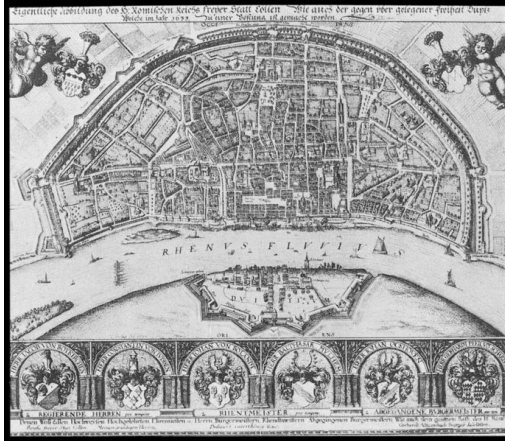


(b) Paul Gauguin, Rochers au bord de la mer, Göteborgs Konstmuseum - 1886 [22]

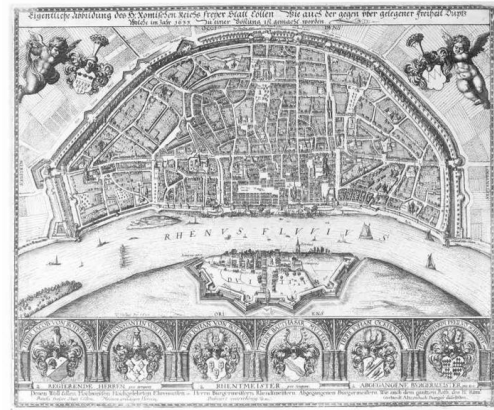
Figure 6.10: Cosine similarity of 0.9

The calculated correspondence of the images in Figure 6.10 is approximately 63.32%. Their L-value differs by about 24.34, the a-axis by about 2.61 and the b-axis by 5.08. Compared to the images in Figure 6.9 a significantly higher colour difference was found in these two images. At the same time, the percentage of good matches is more than twice

as high. If only the data were considered, it would have to be assumed that these two images show the same original. The algorithm has thus confirmed exactly what is also visible to the human eye. Although the images differ noticeably in their brightness, it can nevertheless be recognised that these two reproductions show the same original.



(a) Wenzel Hollar, Ansicht von Köln und Deutz aus der Vogelschau, Stadtmuseum Köln - 1635 [39]



(b) Wenzel Hollar, Vogelsicht von Köln und Deutz, Staatliche Kunsthalle Karlsruhe - unknown [40]

Figure 6.11: Cosine similarity of 0.85

The two images shown in Figure 6.11 only have a match of 35.39% according to the computer vision algorithm. They differ by 30.17 in brightness. Since these are black and white images, the values of the a- and b-axes are always zero, which is why no differences can be detected for them.

With these images the algorithm did not work as expected. If one were to proceed here as in Figure 6.9 and only start from the computer vision data, it would be difficult to make a concrete statement about the two images.



(a) Rembrandt, Christus heilt die Kranken, genannt das Hundertguldenblatt, Staatliche Kunsthalle Karlsruhe - 1601-1649 [26]



(b) Rembrandt, Das Hundertguldenblatt, Albertina Wien - 1647-1649 [27]

Figure 6.12: Cosine similarity of 0.76

According to Computer Vision, the images shown in Figure 6.12 only match by 21%. Their

L-value differs by 25. The a-value by 0.64. Their b-value differs by 7.8, suggesting a slight yellow tint in one image, which is also visible here to the naked eye. Here, too, the algorithm had problems determining the similarity of the two images.



(a) Barolomé Esteban Murillo, Einladung zum Pelota - Spiel, The trustees of Dulwich Picture Gallery London - 1670 [3]



(b) Bartolomé Esteban Murillo, Einladung zum Pelota - Spiel, The Trustees of Dulwich Picture Gallery London - 1670 [4]

Figure 6.13: Cosine similarity of 0.67

The images seen in [Figure 6.13](#) match by 51.83%, according to Computer Vision. Their L-value differs by 15, their a-value by 2 and their b-value by 1. Since the a and b values are both shifted in the same direction, this shows that the images differ in the warmth of their colours, which is also visible to the naked eye.

These two images again fulfil the expectation from the introduction. With relatively small differences in colour, a high number of features of both images could be matched.



(a) Utagawa Hiroshige, Die Komagata-Halle und die Azuma-Brücke, Blatt 62 aus der Serie: 100 berühmte Ansichten von Edo, Museum für Kunst und Gewerbe Hamburg - 1857 [34]



(b) Utagawa Hiroshige, 100 berühmte Ansichten von Edo, Institution not known - 1856-1858 [33]

Figure 6.14: Cosine similarity of 0.1

The two images from Figure 6.14 have 22% good matches. Their L-value differs very significantly by 80. Their a-value by 5.7 and their b-value is virtually identical. In the introduction, it was assumed that these two images would produce a similar result to those in Figure 6.9. However, the result shows more clearly than expected that these two images have more significant differences than those from Figure 6.9.



(a) Groteske Figuren mit Knorpelwerk, Blatt aus dem 'Neuw Grotteßken Buch', Museum für Kunst und Gewerbe Hamburg - 1610 [8]



(b) Christoph Jamnitzer, Gevleugelde draak, Rijksmuseum Amsterdam - 1573 [7]

Figure 6.15: Cosine similarity of 0.007

The images from Figure 6.15 were identified in section 6.1 as those with the least cosine similarity. Whether these are two reproductions showing the same original is difficult to determine even with the naked eye. If you look at the right-hand image, you can see a small caption, which is not to be found in the left-hand image. Nevertheless, the computer vision algorithm gives the two images a similarity of 50.36%. The L-value is also not very different at 3.37. The a-value differs by 1.39 and the b-value by 2.25.

Figure 6.16 shows the distribution of percentage matches. Most relations had a similarity between 50 and 60%. A threshold to determine if two images are similar enough could be set between 20 and 30% based on this data.

52 pairs of images were found to be 100% identical by the algorithm. All their features matched and they were also identical in colour.

The two reproductions by Kawanabe Kyosai, which are entitled "*The Journey to the West: Retrieving the Sutra Scriptures from India*" and can be seen in Figure 6.17, also possess a cosine similarity of 1.

The example which can be seen in Figure 6.18 shows the artwork "*plaster model of a kneeling man*" by Vincent van Gogh. The cosine similarity of the tags assigned to the two images is 0.64.

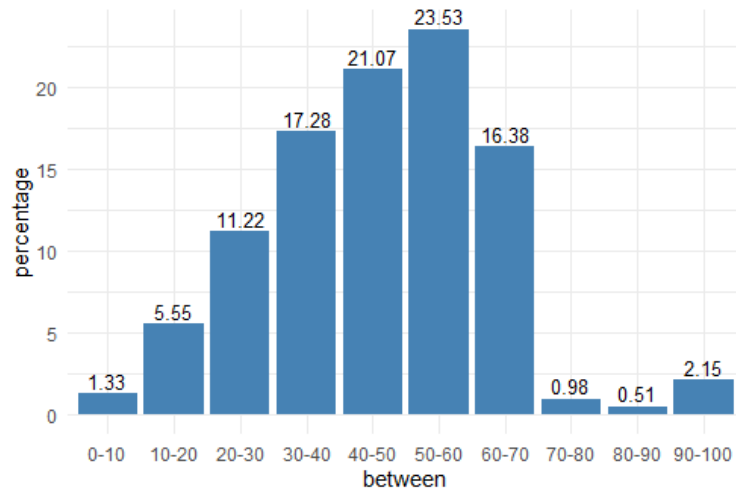


Figure 6.16: Distribution of percentage matches

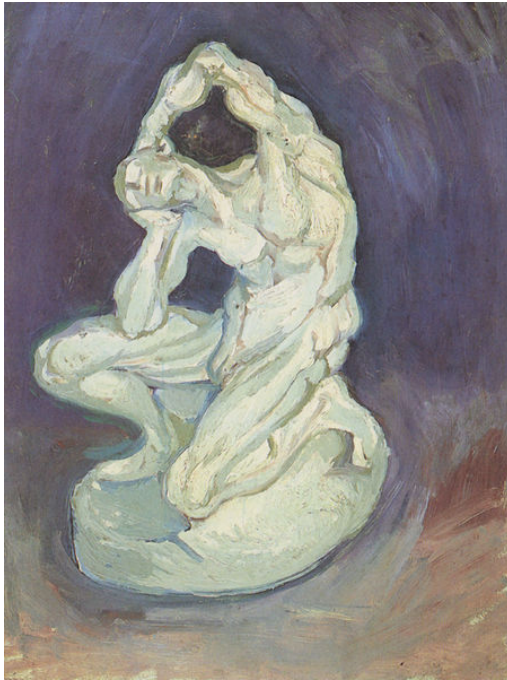


(a) Kawanabe Kyosai, Die Reise nach Westen: Das Holen der Sutraschriften aus Indien, Museum für Kunst und Gewerbe Hamburg - 1867 [18]

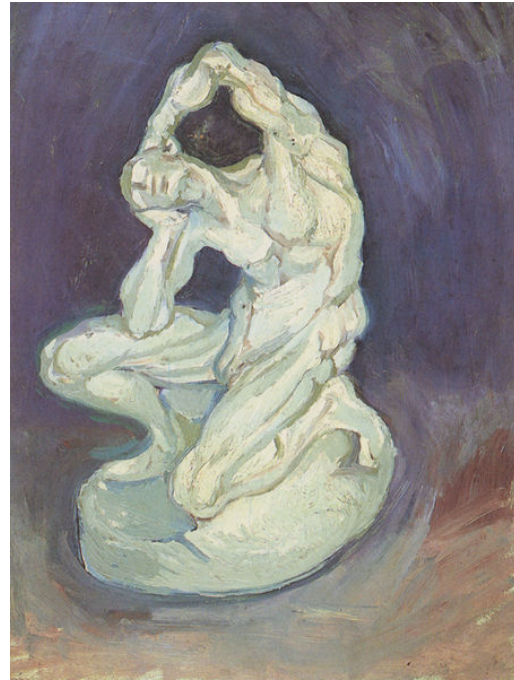


(b) Kawanabe Kyosai, Die Reise nach Westen: Das Holen der Sutraschriften aus Indien, Museum für Kunst und Gewerbe - 1864 [17]

Figure 6.17: Computer Vision Match of 100% with a cosine similarity of 1



(a) Vincent van Gogh, Gipsmodell eines knien- den Mannes, Van Gogh Museum Amsterdam - 1886 [37]



(b) Vincent van Gogh, Gipsmodell eines knien- den Mannes, Van Gogh Museum Amsterdam - 1886 [38]

Figure 6.18: Computer Vision Match of 100% with a cosine similarity of 0.64



(a) Utagawa Toyokuni I., Aus: Holzschnittbuch über zeitgenössische Kleidung, Museum für Kunst und Gewerbe Hamburg - 1802 [35]



(b) Utagawa Toyokuni I., Aus: Holzschnittbuch über zeitgenössische Kleidung, Museum für Kunst und Gewerbe Hamburg - 1802 [36]

Figure 6.19: Cosine similarity of 1

The computer vision algorithm assigned a similarity of 29.72% to the two images in [Figure 6.19](#). This is not a bad result, as the images appear to be very similar in many respects. The algorithm has calculated that the images differ by 3 in their L-value, by 1 in their a-value and by 1.7 in their b-value.



(a) Rembrandt, Zelfportret met open mond, alsof schreeuwend: buste, Rijksmuseum Amsterdam - 1630 [29]



(b) Rembrandt, Selbstbildnis mit lockigem Haar und weissem Kragen, Rijksprentenkabinett Amsterdam - 1629 [28]

Figure 6.20: Cosine similarity of 0.64

The images in Figure 6.20 are only 3.27% similar according to the algorithm. Their L-value differs by 12.5, the a-value by 0.75 and their b-value by 8.3. The b-value thus reflects the yellow tint of Figure 6.20a compared to Figure 6.20b. The low similarity shows that this is not the same original.



(a) Joseph Anton Koch, Heroische Landschaft mit Regenbogen, Institution not known - 1806

13



(b) Joseph Anton Koch, Heroische Landschaft mit dem Regenbogen, Staatliche Kunsthalle Karlsruhe - 1805

12

Figure 6.21: Cosine similarity of 0.73

At first sight, one might think that the two pictures in Figure 6.21 are the same, but with differing colours. This actually is not the case. Both pictures show roughly the same theme, but the execution is clearly not the same. The Computer Vision algorithm obtains the same result. The algorithm only assigns a match of 4.13% to the reproductions. The algorithm's colour calculations clearly show that Figure 6.21b has a distinct yellow cast compared to Figure 6.21a and has generally been painted with darker colours. On very close inspection of these two images, the following stands out:

- The general motif of the scene seems to be equal.
- There are three immediately recognisable people in the foreground. However, they are not the same three people. While the dress of the person on the right edge is very similar in both pictures, differences can be seen in the two persons on the left.
- In Figure 6.21a a flock of sheep with a shepherd can be seen down the mountain. In Figure 6.21b the shepherd is missing and the flock is much easier to recognise, possibly because of the different brightness.
- In general, the animals and where they can be found are clearly different in both pictures.
- The lower right part of the picture is different.
- The cloud formation is similar but not the same.
- The prominent tree at the left hand side of the picture also is not the same.
- Even the city is slightly different in both pictures.

Because of all these small differences, the poor match between the two images is a correct calculation of the algorithm, as the images clearly differ in their small details.

CHAPTER 7

Conclusion

How well does a crowd recognize differences between photographs of the same artworks? The reproductions examined, which were already known to represent the same original, showed that the similarity of the tags assigned was quite high on average. Nevertheless, the crowd only assigned completely identical tags in very rare cases. This leads to the conclusion that the crowd actually assigns different tags to images that differ in their colouring.

Additionally, this thesis has shown that the tags assigned by the crowd can be used to flawlessly match images that share the same or at least a very similar motif.

7.1 Evaluation of the results

The cosine similarities of the reproductions, which were already known to represent the same original, were very convincing. They show that the similarities in the images are reliably annotated by the players. Applying cosine similarity to the reproductions that were not known to depict the same original was also a great success.

Positive results could be detected throughout the samples. All the images examined showed similar scenes and it was clear why the cosine similarity of the tags was so high.

As shown in chapter 5 using the examples, there were also images that were annotated very differently by the crowd. The woodblock print by Utagawa Hiroshige shown there clearly demonstrates that the tags assigned by the crowd can also identify images that are ultimately very different.

The computer vision algorithm, which was to be used to mathematically calculate the actual differences between the compared images, was also convincing.

The feature matches of reproductions that knowingly represent the same original were consistently high. It was also possible to discover reproductions that were actually identical. On the one hand, they had a cosine similarity of 1 and, on the other hand, they were recognised by the computer vision algorithm as absolutely identical in their colour and features. It was interesting that not only cosine similarities of 1 were found. The cosine similarity of these completely identical images is absolutely diffuse.

The computer vision algorithm was also convincing in the three examples that were classified as potentially the same due to their high cosine similarity. These three examples are good proof that the crowd annotates images with similar or even almost identical motifs

with similar or even identical tags when assigning their tags. However, it can also be determined through computer vision that the bidders do not actually represent the same original and may only have been digitised differently.

This thesis has shown that the quality of the crowd-sourced tags in ARTigo is so good that it can be used to categorise images in databases so reliably that it is possible to identify reproductions that show the same original through similarly tagged images.

7.2 Further thoughts

This topic is far from exhausted by the thesis. On the contrary. It could be just the beginning, the basis for further research.

On the basis of this thesis, an attempt could be made, for example, to find and remove reproductions in databases. Furthermore, an attempt could be made to generalise the assigned tags even further. Perhaps a dictionary could be built over the entire ARTigo vocabulary, which would group the tags and thus also ensure that synonyms can be understood by the algorithm as the same tag. Through machine learning, for example with image net, the computer vision algorithm could be further developed to the extent that it analyses the objects in the images and, on the basis of these and the feature vectors of the images described in this thesis, makes a decision as to whether two images reflect the same original.

Through all of these improvements, the laborious task of categorising and sifting through the images by art historians could be greatly facilitated.

With these automations, resource-intensive work steps could be replaced and the staff would have more time to concentrate on the essential analyses.

Bibliography

- [1] Tom Alby. "Web 2.0". In: *Konzepte, Anwendungen, Technologien 2* (2007).
- [2] Autorinnen und Autoren der Abstracts. *DHd 2019 Digital Humanities: multimedial & multimodal. Konferenzabstracts*. ISBN 978-3-00-062166-6 (gedruckte Ausgabe). Frankfurt am Main: Zenodo, Mar. 2019. DOI: [10.5281/zenodo.2596095](https://doi.org/10.5281/zenodo.2596095) URL: <https://doi.org/10.5281/zenodo.2596095>
- [3] Bartolomé Esteban Murillo. *Einladung zum Pelota - Spiel*. London, The Trustees of Dulwich Picture Gallery. 1670.
- [4] Bartolomé Esteban Murillo. *Einladung zum Pelota - Spiel*. London, The Trustees of Dulwich Picture Gallery. 1670.
- [5] Matthias Becker, François Bry, et al. *ARTigo – Social Image Tagging [Dataset and Images]*. 2018.
- [6] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).
- [7] Christoph Jamnitzer. *Gevleugelde draak*. Rijksmuseum - Amsterdam. 1573.
- [8] Christoph Jamnitzer. *Groteske Figuren mit Knorpelwerk, Blatt aus dem 'Nieuw Grotteſken Buch'*. Museum für Kunst und Gewerbe Hamburg. 1610.
- [9] Laura Commare. "Social tagging als Methode zur Optimierung kunsthistorischer Bilddatenbanken–eine empirische Analyse des ARTigo-Projekts". In: *Kunstgeschichte. Open Peer Reviewed Journal* (2011).
- [10] Michaela Geierhos, Natalie Zink, et al. "Rechtschreibkorrektur".
- [11] Jiawei Han, Micheline Kamber, and Jian Pei. "2 - Getting to Know Your Data". In: *Data Mining (Third Edition)*. Ed. by Jiawei Han, Micheline Kamber, and Jian Pei. Third Edition. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 39–82. ISBN: 978-0-12-381479-1. DOI: <https://doi.org/10.1016/B978-0-12-381479-1.00002-2> URL: <https://www.sciencedirect.com/science/article/pii/B9780123814791000022>
- [12] Joseph Anton Koch. *Heroische Landschaft mit dem Regenbogen*. Staatliche Kunsthalle Karlsruhe. 1805.
- [13] Joseph Anton Koch. *Heroische Landschaft mit Regenbogen*. Not known. 1806.
- [14] Dan Jurafsky and James H. Martin. "Speech & language processing". In: Pearson Education India, 2000. Chap. 5, pp. 139–187.
- [15] Katsushika Hokusai. *Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji*. Museum für Kunst und Gewerbe Hamburg. 1830.

- [16] Katsushika Hokusai. *Der sieben Ri lange Strand von Soshu, Blatt 13 aus der Serie: 36 Ansichten des Fuji*. Museum für Kunst und Gewerbe Hamburg. 1830-1831.
- [17] Kawanabe Kyosai. *Die Reise nach Westen: Das Holen der Sutraschriften aus Indien*. Museum für Kunst und Gewerbe Hamburg. 1864.
- [18] Kawanabe Kyosai. *Die Reise nach Westen: Das Holen der Sutraschriften aus Indien*. Museum für Kunst und Gewerbe Hamburg. 1867.
- [19] Emmanuel Keuleers. *vwr: Useful functions for visual word recognition research*. R package version 0.3.0. 2013. URL: <https://CRAN.R-project.org/package=vwr>
- [20] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.
- [21] Paul Gauguin. *Felsige Meerküste*. Göteborgs Konstmuseum. 1886.
- [22] Paul Gauguin. *Rochers au bord de la mer*. Göteborgs Konstmuseum. 1886.
- [23] Waltraud von Pippich. "Rot Rechnen". In: *Zeitschrift Für Digitale Geisteswissenschaften. Sonderband 1* (2016).
- [24] François Prof. Dr. Bry, Martin Bogner, et al. 2021. URL: <https://www.artigo.org> (visited on 03/01/2021).
- [25] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: <http://www.R-project.org/>.
- [26] Rembrandt. *Christus heilt die Kranken, genannt das Hundertguldenblatt*. Staatliche Kunsthalle Karlsruhe. 1601-1700.
- [27] Rembrandt. *Das Hundertguldenblatt*. Albertina - Wien. 1647-1649.
- [28] Rembrandt. *Selbstbildnis mit lockigem Haar und weißem Kragen*. Amsterdam, Rijksprentenkabinet. 1629.
- [29] Rembrandt. *Zelfportret met open mond, alsof schreeuwend: buste*. Rijksmuseum - Amsterdam. 1630.
- [30] Sabine Scherz. "Kunstgeschichte berechnet: Interdisziplinäre Bilddatenanalyse crowdgesourcter Annotationen". Feb. 2017. URL: <http://nbn-resolving.de/urn:nbn:de:bvb:19-214650>
- [31] Stefanie Schneider, Andrea Geipel, et al. "Paare aufdecken. Das digitale Bild in historischen Inventaren". In: *Das digitale Objekt: Zwischen Depot und Internet*. Deutsches Museum, 2020, pp. 127–136.
- [32] Stefanie Schneider and Hubertus Kohle. "The Computer as Filter Machine: A Clustering Approach to Categorize Artworks Based on a Social Tagging Network". In: *Artl@s Bulletin* 6 3 (2017).
- [33] Utagawa Hiroshige. *100 berühmte Ansichten von Edo*. Not known. 1856-1858.
- [34] Utagawa Hiroshige. *Die Komagata-Halle und die Azuma-Brücke, Blatt 62 aus der Serie: 100 berühmte Ansichten von Edo*. Museum für Kunst und Gewerbe Hamburg. 1857.
- [35] Utagawa Toyokuni I. *Aus: Holzschnittbuch über zeitgenössische Kleidung*. Museum für Kunst und Gewerbe Hamburg. 1802.
- [36] Utagawa Toyokuni I. *Aus: Holzschnittbuch über zeitgenössische Kleidung*. Museum für Kunst und Gewerbe Hamburg. 1802.
- [37] Vincent van Gogh. *Gipsmodell eines knienden Mannes*. Van Gogh Museum - Amsterdam. 1886.

- [38] Vincent van Gogh. *Gipsmodell eines knienden Mannes*. Van Gogh Museum - Amsterdam. 1886.
- [39] Wenzel Hollar. *Ansicht von Köln und Deutz aus der Vogelschau*. Köln, Stadtmuseum. 1635.
- [40] Wenzel Hollar. *Vogelansicht von Köln und Deutz*. Staatliche Kunsthalle Karlsruhe. Not Known.
- [41] Hadley Wickham, Romain François, et al. *dplyr: A Grammar of Data Manipulation*. R package version 1.0.2. 2020. URL: <https://CRAN.R-project.org/package=dplyr>.