An approach for continuous painting detection and room prediction during a museum walk

Thomas Aelbrecht, Andreas De Witte, Jochen Laroy, Pieter-Jan Philips, and Gillis Werrebrouck

Ghent University, Valentin Vaerwyckweg 1, 9000 Ghent, Belgium

Abstract. Are you tired of exploring a museum on a map and figuring out where you are? This paper focusses on resolving this issue. It allows the detection of paintings in a video or on a live recording while walking through the museum. The paintings are detected using a mean-shift segmentation filter, floodfilling and contour detection. The detected paintings are matched against all known paintings in the museum resulting in a number of predictions. These predictions are then used as input for a model inspired by the Hidden Markov model. This model determines if they don't cause a teleportation through the museum. All information is displayed on a clear map so it's easy to know where you are in the museum. Never look at the map again, look at your phone while enjoying the paintings!

Keywords: Painting detection · Painting matching · Automated mask creation · Hidden Markov model

Introduction

This paper focusses on describing the different steps to reconstruct the path of an excursion through a museum. The first section describes how the paintings are cut out of high quality pictures. The following section explains an algorithm that can independently find paintings in a picture. Prior research has been done on this topic, one of which is by He et al. [6]. This paper focusses on the detection of irregular shaped objects in images and was used as inspiration for the automatic mask creation in the detection algorithm further described in this document. The third section is about how paintings are being matched which was inpired by the work of Liu et al. [9]. This section describes performant solutions to figure out which paintings are visible in a picture. The fourth section is about how to localize where a person is. This is done by using a custom implementation of the Hidden Markov model to decide which location is a logical result. This model was designed by consolidating the concepts of a paper about a basic Hidden Markov model [4], one about speech recognition [7] and a paper about combining historic probabilities [5]. The fifth and final section is about the visualization of the result. This section describes a modern way of visualizing the visited rooms in the museum.

1 Semi-supervised painting detection

1.1 The naive painting detection algorithm

The contour detection is the core part of the semi-supervised painting detection. It's the part that decides what contours are in the image and it works as described below.

Before any contour detection can be done, as many unnecessary details as possible have to be removed. OpenCV has several options to solve this problem, for example eroding, dilating, blurring (median, Gaussian…) or downscaling. The algorithm needs to get rid of as many details as possible to prevent incorrect detection. Although this won't be perfect, the removal of details will decrease the number of incorrectly detected contours.

So the first step of the algorithm is to remove details by resizing the image. The current implementation scales the image down and back up with a factor 5. To scale the image back up, pixel values are calculated using a pixel's area. The scale up was mostly done to be able to show the original image, but it's also a possibility to use the algorithm with the downscaled image. This will result in a slight increase in performance because the image it's working on would be smaller.

The next step is to convert the image to grayscale. This is done to make it easier to differentiate certain parts. This grayscaled image is then dilated and eroded to remove even more noise at the borders of the painting or on the walls. The last step of removing noise is applying a median blur. The biggest advantage of using a median blur is that it will preserve edges while removing noise. The idea behind these steps is to smeare as many details as possible, in other words to make bigger blots with the same color. This makes it easier to detect the borders and remove noise in the background.

The next step is to detect edges with Canny. The result of the Canny function will then be dilated once again to make the found edges stronger. Thereafter the contours can be detected using OpenCV's contour detection algorithm. This algorithm will make sure only the most outer contours are returned when a hierarchical structure of contours is found. A small final detail to prevent unlogical solutions is the following: any contour with a ratio smaller than 1:10 will be removed. This prevents very small contours to appear around noisy parts of the image.

1.2 Strengths and weaknesses

This biggest strength of this algorithm is that it will give an output in almost every image. An example of an image where no painting will be detected is an image where the framework of the painting is invisible, meaning the image only contains the painting itself, no wall and no framework. With this kind of paintings, the desired solution is a contour containing the entire painting, while this is impossible because the painting has no border at all. Another advantage

is that the algorithm is very fast in detecting contours due to the fact that all parts of the algorithm are standard functions that have a good performance.

However, the biggest weakness of this algorithm is that almost every found contour is not precise enough, so almost every solution needs a slight adjustment. For some solutions, the algorithm detects contours that are slightly larger than the actual painting, while for other paintings, it doesn't even include the border of the painting.

Another weakness of the algorithm is that paintings sometimes have overexposure or shadows, which makes it harder to detect it's contours. Some paintings even have a small tag with a description next to the painting, sometimes this small tag is detected as being a painting. This is a logical decision, because there's a big contrast in colors and the tag has a clear contour, but this is not a desired effect.

This algorithm is a first version to quickly be able to fill the database with the groundtruth. The actual algorithm in its final version is completely different and doesn't have these issues anymore. This algorithm will be discussed in the next section.

2 Unsupervised painting detection

2.1 The detection algorithm

The second detection algorithm drastically differs from the first attempt (see Section 1.1), it was designed from scratch with the weaknesses of the previous algorithm in mind. A little insipiration was taken from Geoff Natin ([10]).

The first step in the algorithm is a mean shift segmentation (see Figure 1)). This is a method to remove noise by taking the mean of the pixels within a certain range. A big advantage is that it partially removes color gradients and fine-grain textures which would cause issues in the further steps of the algorithm.

A common technique in object detection is by creating a mask of the object. The problem with this is that there's no certainty about the size and ratio of the painting as well as where the painting is located in the image and how many there are in the image. This can be solved by thinking the other way around. The one certainty that is consistent throughout all images is that all paintings hang on a wall. A mask for the wall will be made instead of making a mask for the paintings. The mask of the painting(s) can then be obtained by simply inverting this mask.

Another issue is that this mask can't be statically programmed in code for each image since the algorithm needs to be usable on any image or video frame. This can be solved by using a technique called flooding. Flooding will create a mask from a starting position in the image and will fill all neighbouring pixels if they have a color close to the color of the starting position. This is done recursively until no pixels are within the color range of the starting position anymore. [6]

The next problem is that there needs to be a good starting position to have a good and correct mask for the wall. To find this, the unsupervised algorithm

Fig. 1. An example image after mean shift segmentation

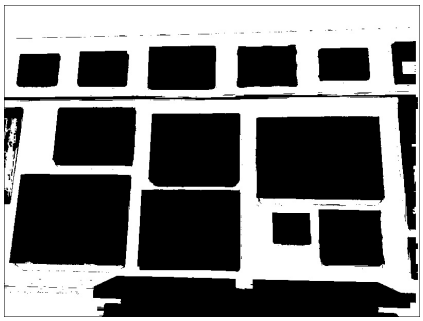


Fig. 2. The wall mask for the example image

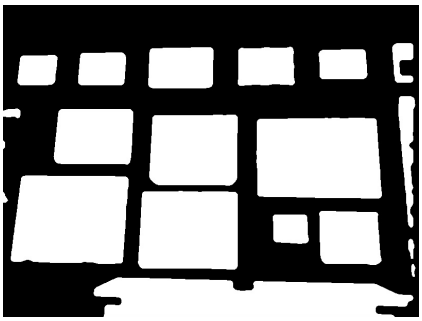


Fig. 3. The Painting mask for the example image

will iterate over all pixels of the image with a certain step size and perform the flooding with that pixel as starting position. Once a mask that has the same height and width as the image is found, then this mask will be used in the next steps of the algorithm. If no such mask is found after iterating over all pixels (with a certain step size), then the mask with the biggest size is used. This is because sometimes parts of the floor or other elements in the image will obstruct the floodfill algorithm to find a mask that has the same size as the image. See Figure 2) for an example of such mask.

Once the mask of the wall has been obtained, it is inverted to have the mask of the paintings (see Figure 3). The mask is then eroded to remove small imperfections and a median blur is used to smooth the edges in the mask.

The next step is to use Canny, the difference with the naive approach is that the two threshold values are determined by using the Otsu algorithm.

Next, a morphological transformation is performed on the Canny edges (see Figure 4). This will close edges that are close to each other as this will improve the detection of closed contours. This morphological transformation is in essence a dilation followed by an erosion.

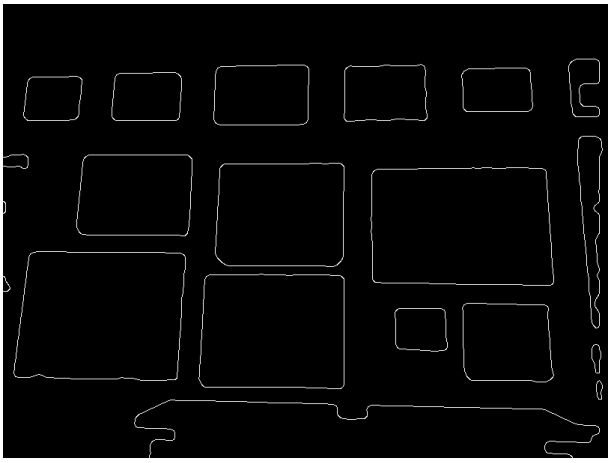


Fig. 4. The edges of the detected paintings

The next steps are exactly the same as the naive painting detection algorithm as described before. The contours are detected in the Canny edges and only fully enclosed polygons with 4 sides are returned as quadrilaterals.

2.2 Strengths and weaknesses

The most important improvement of the detection algorithm is that it isn't based on smearing out colors and finding contours in the color smudges.

The key to success with this algorithm is the use of automatic mask creation by using floodfill as main technique. Most of the paintings are detected in most of the images. The detected paintings also have a better fit compared to the naive algorithm.

With this new algorithm, the dataset has a bounding box accuracy of 81.31%. Only on some occasions there is a miss detection. One flaw that has been found is that some doorways are detected as a painting. This is logical because of how the mask creation works with the floodfilling. In the case of such a detection flaw, the doorways obstructs the floodfilling algorithm to correctly create a mask of the non-painting area. However, this is something that doesn't happen too often.

A minor problem with this algorithm, which also occurs with the naive algorithm, is that it sometimes detects dark shadows as part of the painting. Although, this is less of an issue in this algorithm then it was in the naive algo-

rithm. It doesn't affect the accuracy too much because if there is a shadow, it only appears on one side of the painting and it doesn't reach far.

A drawback of this algorithm is that it is less performant than the naive algorithm. This is mainly because of the mask creation, specifically the floodfilling. To solve this problem the critical and less performant parts of the algorithm were transformed into Cython. This resolved the whole performance issue and increased the processing performance drastically. [3]

2.3 Quantitative comparison

In order to make a quantitative comparison, two things are needed. First of all, the quadrilaterals have to be found autonomously (as described in Section 1.1). The second thing needed is the groundtruth of the dataset which has been generated by using the solution created with the first algorithm (see Section 1).

To measure the accuracy of the painting detection algorithm, three things are required; the amount of false negatives (= paintings that aren't found at all), the amount of false positives (= detected paintings that aren't paintings) and the bounding box accuracy (= average intersection divided by union).

With the creation of the solution for this problem, a new problem occured: how to find the intersection of these two shapes? The solution to this problem is made by using "Shapely". To do so, the quadrilaterals have to be transformed into a polygon. Once this is done, "Shapely" can compute the intersection. It has been tested whether this gives correct intersections if two polygons are not intersecting, or when they are sharing only a line. Also some more general intersections have been tested. Once the intersection is made, it's easy to find the area of the polygon, using "Shapely" once again.

An example of the detection can be found in Figure 5. The detected contours are shown in red and the groundtruth is shown in blue as determined with the use of the naive algorithm. The dataset consists of 553 images which all together contain 836 paintings. The detection algorithm had 86 false negatives and 32 false positives. The average bounding box accuracy is 81.30%. These results are displayed on the graphs in Figure 6.

2.4 Qualitative evaluation

For the qualitative evaluation, the unsupervised painting detection algorithm is used to check how accurate the paintings are found on a more difficult set. Hereby the test set and the video files are used. The painting detection is working well, most paintings are being found. However, some difficulties occur. The algorithm sometimes defines a window as being a painting and it almost always defines a doorway as painting.

Another problem is images shot at a sharp angle. For example when it seems like the borders of two adjacent paintings are overlapping. These images are sometimes not recognized. Paintings in images that are too blurred are sometimes not found either.

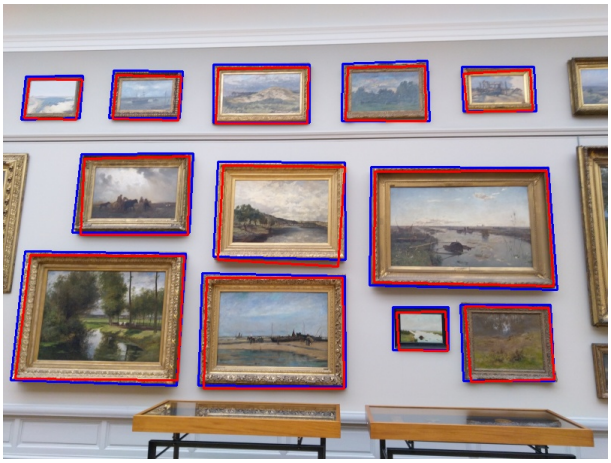


Fig. 5. A visualization of the detected paintings

3 Matching

3.1 Features

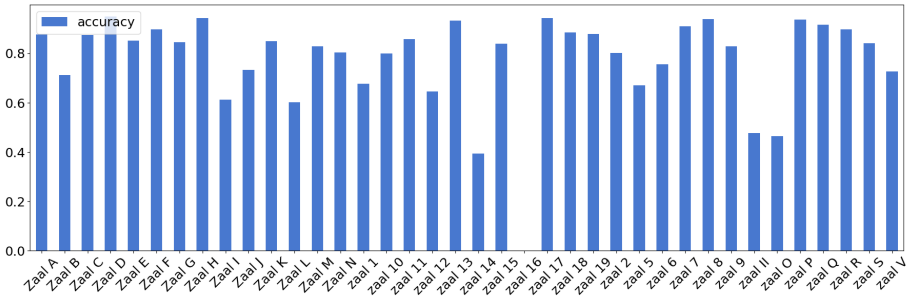
Before diving deeper into the matching algorithm, it's important to explain the features that are used to do the matching. One possibility is to use keypoints detected by algorithms like SURF [2] or ORB [12]. However keypoints turn out to be inefficient and inappropriate for object detection in museums [1].

The algorithm in this document only depends on two types of histograms from the detected paintings. The first type is a histogram of the full painting, the second type is a collection of histograms gathered from different blocks of the painting. The block size used in this algorithm divides the painting in 4 rows by 4 columns (or 16 blocks), independent of the painting's size. Both features are saved to the database for each of the labeled paintings and are both used in the matching algorithm explained in the next section.

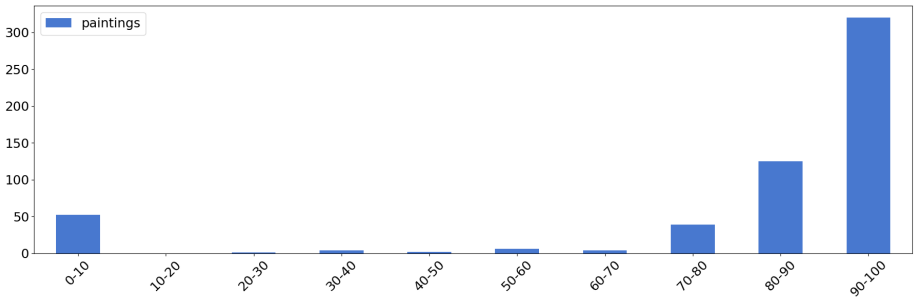
3.2 The matching algorithm

The first step in matching a given painting with the entire dataset of paintings is a light intensity equalization. This way different light intensities have no influence on the histograms gathered from a detected painting. Equalization is a technique that is only used on the video frames because of the rather bad frame quality, the dataset itself and the previously performed benchmarks don't need this step. [11]

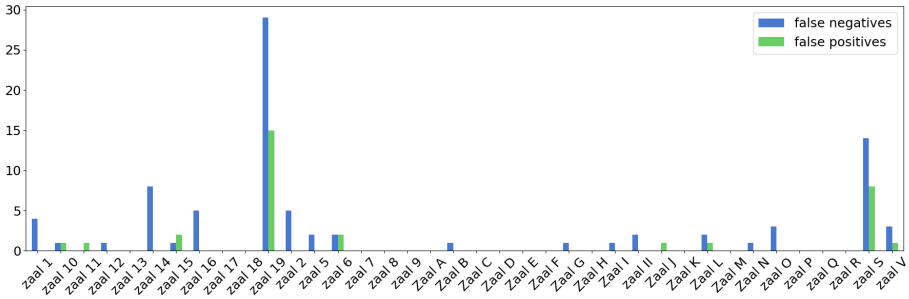
The second step consists of fetching the histograms as described in Section 3.1. Thereafter these histograms are compared against all histograms in the dataset. Per known painting the distance between the histograms is calculated using a technique called *correlation*. This way a chance between 0 and 1



(a) Average room detection accuracy



(b) Detection accuracy ranges for the dataset



(c) False negatives and false positives per room

Fig. 6. Performance measurements

is obtained. The histogram of the whole painting gives one chance, the block histogram gives a total of 16 chances which are combined by taking the average. These two chances are combined using Formula 1 which calculates a weighted average. In this formula, $P(X = P_i)$ stands for the chance that the detected painting X is painting P_i from the dataset, B stands for the average of the block histogram distances and F stands for the distance of the normal histograms. The block histogram gets a much higher weight because the predictions with these histograms are much more reliable than these with the normal histograms.

$$P(X = P_i) = \frac{(8 * B) + (1 * F)}{9} \tag{1}$$

The matching algorithm gives a list of possible rooms with the according chances per detected painting, even if the room is not accessible given the current room. How impossible rooms are treated, will be explained in Section 4. The algorithm has also an option to ignore chances that are below a given threshold.

3.3 Matching results

The matching algorithm has been tested in three ways. First, the dataset was matched against itself without using the detection algorithm but by using the corners from the semi-supervised algorithm. This test uniquely matched 835 of 836 paintings correctly. This means that only one painting could not be uniquely matched, this could be because some paintings appear in multiple images. The average matching probability is 92.24%.

The second test was performed on the dataset but in combination with the detection algorithm described in Section 2.1. This test indicates that 702 of the 762 detected paintings were matched correctly.

The third and last test was performed on the test dataset also in combination with the detection algorithm. The images have been divided in folders with the correct room as name to be able to count the amount of correct room matches. This test reveals that 158 of the 338 detected paintings had a correct room match. This is lower than the dataset because these images are meant to be more difficult to match becuase of the angle, rotation, light intensity, etc. These results are displayed on the graphs in Figure 7.

4 Localization

4.1 Hidden Markov model

In order to establish an intelligent localization of the user by eliminating teleportations, a model inspired by the Hidden Markov model ([4]), was designed. The reason why no pure Hidden Markov can be used is because the input and output of the model is identical. Another reason is that there's no way to determine the initial room of any given video without having a number of observations, which are rooms in this case. Therefor an alternative Hidden Markov model was designed. [8]

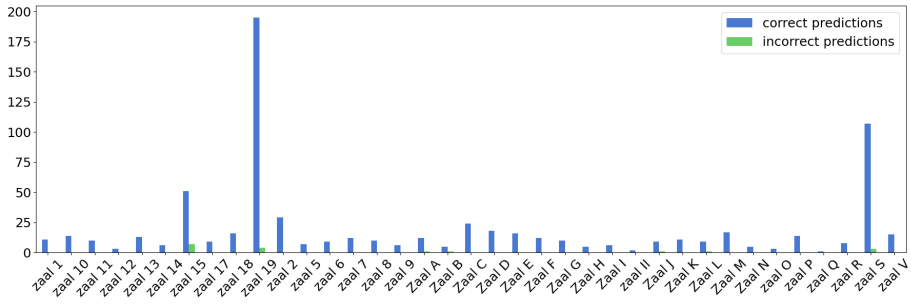


Fig. 7. Correct and incorrect painting predictions per room

4.2 Alternative Hidden Markov model

The alternative model basicly keeps a history of n observations and predicts the most common observation as the current room. Initially the model doesn't know the current room and therefor doesn't emit any prediction until enough observations are obtained.

The model takes a list of probabilities per painting as an input which could possibly contain different probabilities for the same room. However the model needs one probability per room. So Formula 2 ([5]) is used to combine all probabilities for a given room into one probability. In this formula, $P(X = R)$ stands for the chance that the user is in room R, $P_i(X = R)$ stands for the i^{th} chance, given by the prediction algorithm, that the user is in room R and $P_i(X \neq R)$ is logically the chance that the user is not in room R.

$$P(X = R) = \frac{\prod_i P_{test_i}(X = R)}{\prod_i P_{test_i}(X = R) + \prod_i P_{test_i}(X \neq R)} \tag{2}$$

Although this formula can combine probabilities, it doesn't take the current room into account. Therefor a weight is added to each of the chances. This weight is equal to 1 when the room is accessible from within the current room and can be choosen for rooms that aren't accessible, the default is 0.5. This results in Formula 3. If chances per room, e.g. from previous observations, are already known, then these are also taken into account in this calculation.

$$P(X = R) = \frac{\prod_i w * P_{test_i}(X = R)}{\prod_i w * P_{test_i}(X = R) + \prod_i w * P_{test_i}(X \neq R)} \tag{3}$$

By calculating the result of this formula per room, a list of probabilities per room is obtained. This list is then used to find the room with the highest probability which will be the observation for this input.

If the current observation is accessible from the current room, it is added to the history of observations. If this is not the case, the current room is added the history. Now, the most common room in the history is taken as the prediction for this input.

5 Visualization

In order to create a visualization of the algorithms described in the previous sections, a 3D model of the floor plan of the museum was created. First, this floor plan is converted into an SVG image so it can easily be altered in Python. Working this way makes it possible to dynamicly create an HTML page showing the SVG floor plan and the currently analyzed frame along with the detected paintings. Because some operations need a lot of processing power, two seperate processes are spawned. One process handles the painting detection and room prediction, the other one handles the visualization of the HTML page in a webview. One of the major points of attention during the creation of the visualization was performance, hence the HTML file is never written to disk in order to bypass the I/O operations. Therefor interprocess communication is used to send the HTML page to the second process. When this process receives an HTML page, it can be visualized in the webview. An example of this webview is shown in Figure 5. Besides showing the floor plan and the image, other information is shown as well. This includes information about the predicted room, which is also visualized in red on the floor plan, a probability for this prediction, the currently processed video's name and the currently analyzed frame.

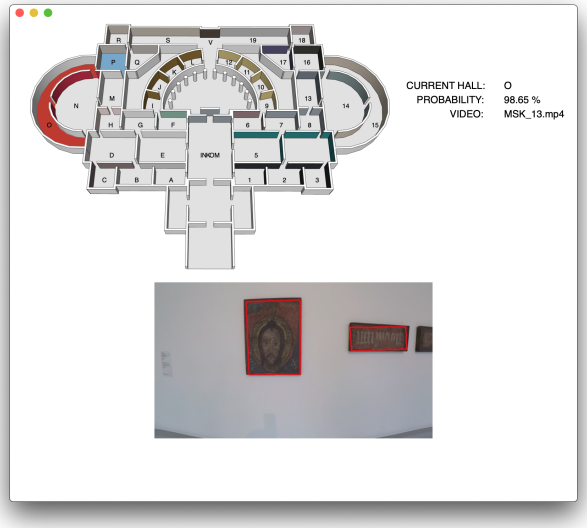


Fig. 8. Modern visualization of the floor plan

6 Conclusion

References

1. Bay, H., Fasel, B., Van Gool, L.: Interactive museum guide: Fast and robust recognition of museum objects. In: Proceedings of the first international workshop on mobile vision (2006)
2. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: European conference on computer vision. pp. 404–417. Springer (2006)
3. Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D.S., Smith, K.: Cython: The best of both worlds. Computing in Science & Engineering **13**(2), 31–39 (2011)
4. Eddy, S.R.: Hidden markov models. Current opinion in structural biology **6**(3), 361–365 (1996)
5. Genest, C., Zidek, J.V., et al.: Combining probability distributions: A critique and an annotated bibliography. Statistical Science **1**(1), 114–135 (1986)
6. He, Y., Hu, T., Zeng, D.: Scan-flood fill (scaff): an efficient automatic precise region filling algorithm for complicated regions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)
7. Juang, B.H., Rabiner, L.R.: Hidden markov models for speech recognition. Technometrics **33**(3), 251–272 (1991)
8. Jurafsky, D., Martin, J.H.: Speech and language processing. vol. 3 (2014)
9. Liu, X., Li, J.B., Pan, J.S., Wang, S., Lv, X., Cui, S.: Image-matching framework based on region partitioning for target image location. Telecommunication Systems pp. 1–18 (2020)
10. Natin, G.: Locating & recognising paintings in galleries, `https://github.com/nating/recognizing-paintings`
11. Patel, O., Maravi, Y.P., Sharma, S.: A comparative study of histogram equalization based image enhancement techniques for brightness preservation and contrast enhancement. arXiv preprint arXiv:1311.4033 (2013)
12. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: 2011 International conference on computer vision. pp. 2564–2571. Ieee (2011)