Computer vision and cognitive systems approaches in the Galleria Estense Museum

Gianluca Mancusi
228614@studenti.unimore.it

Daniele Manicardi
227246@studenti.unimore.it

Vittorio Pippi
219424@studenti.unimore.it

Abstract

The aim of this report is to describe a computer vision software able to analyze images and videos taken from the Galleria Estense Museum in Modena. The final result is intended to guarantee total hardware accessibility, thus imposing a low use of resources as main technical constraint.

1. Introduction

The project aims to provide an application capable of processing artistic images and videos, taken from the Galleria Estense in Modena. Specifically, the basic implemented functionalities are:

1. *Painting detection*: consists in automatically computing the bounding-boxes of the pictures contained in an image. The functionality can be easily extended to a video by iterating the algorithm on all the frames composing it.
2. *Painting rectification*: once having isolated the picture through the previous step, the goal is to correct its perspective, displaying it frontally.
3. *Painting retrieval*: given a rectified painting, the algorithm must automatically determine its title using a pre-built paintings' dataset.
4. *People detection*: it's the same type of detection described in the first step, but here people are the target.
5. *People localization*: starting from the detection performed through the fourth step, the goal is to determine the room in which that person lays.

Being the most important step as it influences all the others, this work initially focused completely on painting detection. The pipeline described in the document has been further divided into equal parts. Once finished and refined, everyone dedicated themselves to the implementation and research about a particular feature, not excluding a mutual adjustment and improvement with the collaborators.

Figure 1. Example of the final result of our image processing pipeline using standard computer vision techniques. The image is undistorted, the paintings have been detected and recognized. The room has also been localized.

2. Related works

In order to develop a high quality project it is always a good idea to study what has already been done, in a way to have a hint of the best techniques applied up until that moment. Starting from that point the decision that can be made is to reinvent the wheel or improve what has already been done. On the internet there are several works on painting detection and retrieval, most of them use OpenCV, but it's curious how the techniques and approaches are totally different.

A straightforward approach is to use a convolutional neural network for object detection such as YOLOv3 [9] or Faster R-CNN [10]. The result is guaranteed, but for educational purposes we decided to continue with traditional techniques, which as an advantage do not require a dataset. In any case, to avoid depriving ourselves of modern approaches we decided to experiment U-Net [11] in order to perform segmentation on the paintings. Going back to the classic computer vision, several jobs have been done in the past, let's see for example the case of [2] where the technique is to apply a gaussian blur and then work on the edges and contours to perform detection, finally applying SIFT for retrieval. Another outstanding work is [1] which focuses

on segmenting the paintings using contours and generating bounding boxes. It also transforms the boxes into a standard format and then applies the retrieval phase with ORB.

Other works focus strongly on the edges of the boxes for detection and apply preprocessing filters in order to have a qualitative result, like [5] which is only able to recognize rectangular boxes.

A more general approach that we found useful, because it applies the transformations and operations from an unconventional point of view, is [8] that particularly surprised us in the operations carried out in the first stages, where it works with the background of the paintings rather than with the foreground. We will take this idea, which we believe is also able to generalize well the shape of the paintings. Precisely because it does not need to know *a priori* what the shape of the paintings is like, but assumes that there is a uniform background behind the painting.

It is well known that most image processing operations suffer from camera lens distortion. There are various approaches to calibrate the camera: the most classic one is using a chessboard. Since the museum videos are not ours and we should not know how they were recorded, the chessboard is not a viable approach. One of the automatic approaches used in the literature is [3], which we found interesting due to the way it increases the number of lines in the frame through the Hough Transform. That is why we implemented it in the project.

For the painting rectification step there were several works. The rectification of a painting can be solved by taking the four corners of the painting and estimating the homography to make the image rectified. This procedure is explained in the work *Whiteboard Scanning and Image Enhancement* [13] that in addition to adjustments to improve a whiteboard, also rectifies it. We used this work for our purpose.

3. The approach

Working on the project, we have taken into account all the related works already done in the past and we have introduced a series of improvements to the several steps of the pipeline.

3.1. Camera calibration

First of all, we try to correct the image distortion automatically using our implementation of the HTRDC algorithm shown in [3]. The only change we made to the algorithm is to apply hough by focusing on vertical and horizontal lines in order to reduce, at least partially, the noise generated by curved lines.

3.2. Paintings detection

As already mentioned, the approach which inspired us the most is to separate the paintings from the background

(a) Original (b) Corrected image

Figure 2. The original image on the left and HTRDC-corrected one on the right.

with various techniques, analyzing each foreground object found later. Obviously this approach makes the assumption that the background is sufficiently homogeneous, the camera is not too close to the picture, and paintings' frames are well shaped and properly separate the painting from the background.

To allow every member of the team to work simultaneously, we have divided the pipeline into 6 macrosections

3.2.1 Pre-processing

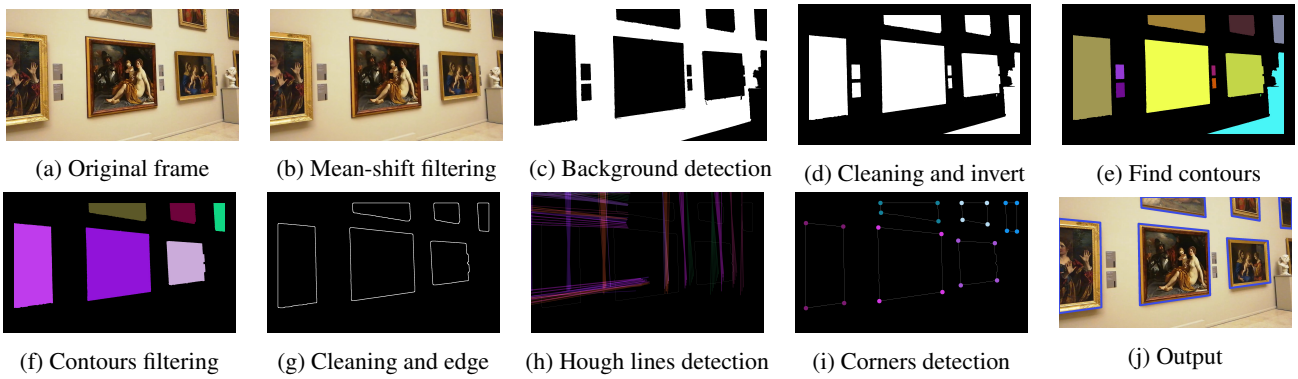
We have applied a mean-shift filter in order to "flatten" the background colors. This is really important since it will facilitate detection in the next sections, providing an image less noisy and more homogeneous but still having sharp edges.

3.2.2 Background detection

We applied OpenCV's *floodFill* function, which selects all pixels that are similar and contiguous to the starting one following a fixed tolerance. In this way, once one of the pixels in the background has been chosen as the starting point, the selection will spread a white area throughout the involved image portion. The frames of the paintings are really important to discriminate the background from the foreground. In this way, trying multiple points we determine the background mask as the largest selection found.

3.2.3 Cleaning

Once the background mask has been retrieved, we clean the image by performing a closing operation, aiming to clean the foreground from small details like paintings' labels or frame discontinuities. Then, the image is inverted and a constant-black padding around it is applied. The purpose of padding is to add a black border around the paintings that are not completely wrapped inside the image. This is very helpful to identify the borders of such cut pictures through the Hough transform. In the Figure (3d) we voluntarily applied a thicker padding to make it clearly visible, but a 1-pixel padding is enough for a more concrete context.



(a) Original frame (b) Mean-shift filtering (c) Background detection (d) Cleaning and invert (e) Find contours

(f) Contours filtering (g) Cleaning and edge (h) Hough lines detection (i) Corners detection (j) Output

Figure 3. The images reported above represent the most important building blocks of the pipeline. They have just a representative scope: starting from the 3f step every contour is processed individually

3.2.4 Components selection

Once having created the mask that separates all the elements from the background, we obtain the paintings' contours using the chain code algorithm applied with the *findContours* function of OpenCV. In order to clean the image from false paintings, all the areas not respecting the following constraints are discarded:

- The size of the retrieved rectangle must be above or equal a fixed threshold.
- The retrieved area must be smaller than $0.9 \times$ *the whole image area* and higher than a fixed threshold.

3.2.5 Contour cleaning and edge detection

Now we need to clean the paintings' borders and apply an edge detector. The last one is necessary in order to rectify the paintings' lines in the further sections.

The first transformation applied is an opening of the foreground, which removes from the paintings' areas their own shadow and part of the frame. Anyway, this operation is really useful because it removes many parts potentially disturbing the retrieval phase. The chosen algorithm for edge detection is Canny: the Figure (3g) shows the kind of results obtained.

As visible, also the paintings partially outside of the image are well localized, thanks to the padding inserted before.

3.2.6 Corners detection

Once the contours are ready, we use the hough transform to abstract its border's shape through vertical and horizontal straight lines. The intersections between such lines represent the corners of all the paintings in the image. In order to assign to each painting its corner group (obviously made of 4 points), we apply a k-means algorithm with $k = 4$. As visible from Figure (3i), this way of seeing the task as a clustering problem leads to the expected results. In the final step we removed the padding and we achieved the result shown in Figure (3j).

Figure 4. On the left a non-rectangular shape and on the right a rectangular painting

3.2.7 Non-rectangular shapes

In some cases it may happen that the picture does not have the classic rectangular shape or, for some reasons, the corners-retrieval steps failed. In such cases, the output will consist of the four corners of the bounding rect wrapping the non-rectangular shape as you can see in the Figure 4.

3.3. Painting rectification

Once the four corners of the paintings have been identified, they are used to estimate the perspective homography transformation for the rectification step. This procedure is described in [13], which we implemented to identify an approximation of the original aspect ratio of the picture in perspective. The [13] paper was written to scan the whiteboards and rectify them. We first estimate the actual aspect ratio of the painting from the detected quadrangle based on the fact that it is the projection of a rectangle in space. Besides the aspect ratio, we can also estimate the focal length

(a) Original

(b) Image corrected

(c) Original

(d) Image corrected

Figure 5. On the left you can see what the pictures look like after being extracted from the frame of the video, while on the right are the pictures after being rectified.

of the camera. From the estimated aspect ratio, and by choosing the "largest" painting pixel as the standard pixel in the final image (a pixel in the original image should be mapped to at least one pixel in the rectified image), we can compute the desired resolution of the final image. A planar perspective mapping (a 3 x 3 homography matrix) is then computed from the original image quadrangle to the final image rectangle, and the painting image is rectified accordingly. The results can be seen in the Figure 5.

3.4. Painting retrieval

Once the painting has been recognized and rectified, we calculate the descriptors of the rectified painting and we compare it with the descriptors of all the paintings in the descriptor database. In fact, we pre-compute all the descriptors of the paintings in the database at the first start of the painting retrieval step. The painting retrieval function will give a match only if there is a sufficiently high difference between the first and the second painting with the highest scores. The scores are the average of the distances of the descriptors.

The algorithm used to detect keypoints in the paintings is ORB [12]. The reason of this choice is that it is a fast algorithm. Moreover, it is easily available on OpenCV, unlike SIFT [6] which is a proprietary algorithm.

More specifically, before starting the algorithm we change the resolution of each image in the database in a standard resolution, empirically chosen at 400x400, and calculate the descriptor. Then we move to the standard resolution and calculate the descriptors of the original image. At each iteration of painting retrieval we will compare only the descriptors.

Figure 6. The overlay map in bottom left part of the screen. The recognized and localized room is the red one. In addition, the picture is undistorted using [3] and the recognized paintings have the names written on the blue label at the top-left of the frame. The painting on the left has no label because it is not in the paintings db.

3.5. People detection

To detect people in the videos it was straightforward to use YOLOv3 [9], but there are several fascinating approaches in the literature, such as RCNN [4] and its faster variants. We decided to use the state-of-the-art YOLOv3 for its speed, if we had to use a more precise recognition system, we would have chosen the Faster RCNN [10].

In order not to recognize the people portrayed in the paintings, we checked whether or not the bounding box of the recognized person was inside the painting. In such case, the person is not labeled since it would be a false positive.

3.5.1 Room detection

In order to locate which room the people are in, we have assumed that the videos were all shot in the same room where the people identified are also in, and that all the people in the video belong to the same room. So once you have identified one of the paintings in the video you can go back to the room where that painting is. To make the result visible, we decided to print the recognized room on the museum map. The map is in overlay in the bottom left part of the screen.

4. Experiments with neural networks

To calculate the effectiveness and metrics of our pipeline we randomly selected the 10% of all frames extracted from the videos and segmented them by hand. Having this data available we did some experiments to see what results could be obtained.

Target	Method	β	α	DSC	TI	IoU	Specificity	Precision	Recall
Paintings	Deterministic	-	-	0.585	-	0.512	0.804	0.638	0.649
	Supervised (with tiny dataset)	-	-	0.475	-	0.35	0.621	0.361	0.880
		1e-12	$1-\beta$	0.705	0.829	0.583	0.738	0.673	0.829
		0.25	$1-\beta$	0.668	0.738	0.545	0.705	0.590	0.864
		0.5	3	0.699	0.749	0.577	0.727	0.644	**0.849**
	Self-supervised + fine-tuning	**-**	**-**	0.716	**-**	0.601	0.760	0.710	0.811
		1e-12	$1-\beta$	0.701	**0.854**	0.584	0.733	0.642	0.854
		0.25	$1-\beta$	0.766	0.783	0.661	0.802	0.780	0.817
		0.5	3	**0.780**	0.830	**0.681**	**0.811**	**0.783**	0.835

Table 1. Results obtained from all experiments with the U-Net, compared with the deterministic approch

4.1. Architecture

Every test we performed consisted of using a U-Net convolutional network [7], which has been configured to do a background-removal with respect to the objective class (statue, painting).

4.1.1 Dataset

Since many high-framerate videos were provided, we built up a training dataset by performing a quantization of such videos every 45 frames, obtaining 3175 total images. The statue dataset required a longer time since we had to segment their shape through the whole dataset, obtaining in the end 693 samples. Finally, the painting dataset was created by manually segmenting the 10% of the 3175 total images randomly sampled.

However, the dataset will contain many frames from the same videos and therefore the data on which the training is done are not totally independent.

4.1.2 Data augmentation

For both statue detection and painting detection we tried to make some data augmentation in order to solve, even if only partially, the lack of images. In both cases we applied the following transformations as you can see here in *pytorch*-like code:

```
transforms.Compose([
    ToPILImage(),
    Resize(512),
    RandomCrop(512),
    RandomRotation(),
    RandomHorizontalFlip(),
    ColorJitter(0.4, 0.4, 0.4, 0.1),
    RandomGrayscale(),
    ToTensor(),
])
```

After some tests we decided to set the resize and the random crop to 512 because larger values have not led to improvements but only to a drop in performance. This is probably due to the fact the image becomes too small and you have an excessive loss of information.

During testing, we used instead the following transformations in order to have deterministic results:

```
transforms.Compose([
    ToPILImage(),
    Resize(512),
    CenterCrop(512),
    ToTensor(),
])
```

4.1.3 Loss function

Originally we used the Dice Coefficient (DSC) to compare the results obtained with ground truth. However, as we realized later, the DSC tends to give an inadequate weight to false positives, so we decided to use Tversky Index (TI), which is nothing more than a weighted DSC.

$$DSC = \frac{2TP}{2TP + FN + FP} \tag{1}$$

$$TI = \frac{TP}{TP + \beta FN + \alpha FP} \tag{2}$$

Indeed, we now have $\beta = \alpha = \frac{1}{2}$, and so $TI = DSC$.

4.2. Painting detection

Durante lo sviluppo della rete neurale per fare painting detection abbiamo utilizzato due approcci differenti. Il prima consiste nell'utilizzare il picolissimo dataset che avevamo a disposizione di 317 immagini per allenare la U-Net in modo supervisionato, ottenendo dei risultati molto promettenti che tendono ad eguagliare o addirittura superare il nostro metodo deterministico. Il secondo invece consiste nel carcare di sfruttare l'intero dataset composto da 3172 immagini e la nostra pipeline per allenare la rete in modo

Target	Method	β	α	DSC	TI	IoU	Specificity	Precision	Recall
		1e-12	$1-\beta$	0.302	0.239	0.206	0.946	0.734	0.241
		0.25	$1-\beta$	0.322	**0.335**	0.217	0.915	0.471	0.399
		-	-	0.241	0.243	0.166	0.914	0.406	0.267
Statue	Supervised	0.5	3	**0.381**	0.177	**0.27**	**0.948**	0.696	0.351
		0.5	5	0.351	0.182	0.264	0.925	0.508	**0.403**
		1	3	0.286	0.100	0.189	0.934	0.752	0.093
		1	5	0.314	0.093	0.217	0.952	**0.786**	0.266

Table 2. Results obtained from all experiments with the U-Net, compared with the deterministic approch

(a) Source (b) Mask

Figure 7. A segmentation result through the use of the U-Net.

self-supervised e poi effettuare un fine-tuning con le immagini sementate a mano. In particolare durante processo di training self-supervised la U-Net cercherà di predire l'output della maschera generata dalla pipeline, mentre durante la fase di fine-tuning verrà addestrata la rete utilizzando le label realizzate a mano. Com'è possibile vedere nella tabella

4.3. Statue detection

During the development of the neural network for the identification of the statues we had more difficulties in comparison to the detection of the paintings. In particular the problems found are mainly caused by the dataset, which is too small considering the large variety of statues in the museum and above all, it is strongly unbalanced. As can be seen in the table 2 the results obtained are not as promising as those of the paintings. In the case of the statues we obtained slightly better results by increasing the values of α and β in order to give more weight to false positives and false negatives, which is why the TI values are very low. In conclusion, having a larger and less unbalanced dataset would give much better results as in the case of painting detection.

5. Results

All the results obtained with the experiments have been summarized in the table 1. In particular, it is interesting to note, as far as the paintings are concerned, how the models

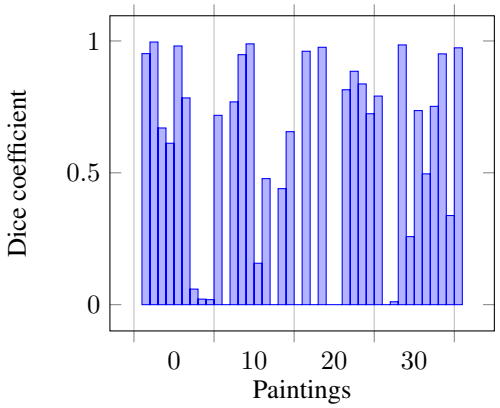


Figure 8. The Dice coefficient measure applied to our pipeline compared to ground-truth bounding boxes.

obtain generally better results in all the metrics except for the specificity that can be matched only by some models that exploit a self-supervised pre-training.

5.1. Results with painting detection

The results are fluctuating. This is clearly seen in the Figure 8.

The algorithm works well with certain types of images, while it fails completely with others. This is justified by the fact that our pipeline is perceptive for any type of image, without providing any cognitive concept of painting or statue.

6. Discussion

This work has been a good way for us to perceive the difference between classic computer vision techniques and the ones that apply neural networks. Despite the limited data available for neural networks, we have achieved more than promising results compared to the pipeline. By dedicating more time to data recovery and increasing the amount of data, we would obtain results that would be very difficult to replicate with traditional algorithms. We also tried tweaking the parameters of the algorithm but it can't be compared with the neural network.

In the attempts to tweak the parameters we noticed that standard measurements such as precision, recall, sensitivity and many more, continued to alternate. So we found a balance, giving more weight to false positives.

Indeed, the most important objective was to remove the outliers that can be found, not recognizing them as paintings. Examples of outliers are statues, fire extinguishers, picture labels and much more. Decreasing false positives means decreasing the number of outliers considered as paintings.

We also found interesting algorithmic solutions for the painting rectification step such as [13] that we were able to implement.

In conclusion using classical algorithms we obtain a considerable saving of computational resources, instead with deep learning we obtain a strong improvement of generalization capacity, but also a stronger consumption of resources. Using both these techniques allows us to compensate for the weaknesses of one and the other.

References

[1] Timothy Thiecke Bert De Saffel. Continuous room localization using painting detection.

[2] Conor Broderick. Robust recognition and identification of paintings using computer vision techniques.

[3] Rita Cucchiara, Costantino Grana, Andrea Prati, and Roberto Vezzani. A hough transform-based method for radial lens distortion correction. pages 182–187, 01 2003.

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[5] Ilya Kavalerov. Using pattern recognition to automatically crop framed art.

[6] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[7] milesial. Unet: semantic segmentation with pytorch.

[8] Geoff Natin. Locating recognising paintings in galleries.

[9] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[12] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 11 2011.

[13] Zhengyou Zhang and Li-wei He. Whiteboard scanning and image enhancement. In *Digital Signal Processing*, volume 17, pages 414–432, April 2007.