Painting detection, extraction, recognition and people detection from museum videos

Bellei Francesco 218634@studenti.unimore.it
Gianelli Marco 205302@studenti.unimore.it
Petrucci Enrico 204385@studenti.unimore.it

1 Introduction

Computer vision's fields of application are varied and in many of them it is necessary to recognize objects in the real world. This is also the case for the project presented in this report. The material on which the work is based on is a series of videos collected inside the "Galleria Estense" museum. The goal of this project is to create an algorithm capable of detecting, extracting and recognizing the paintings inside the videos by comparing them with a database provided. The algorithm also detects the presence of people and their orientation with respect to the camera.

2 Pipeline

Due to the nature of the project and to the tasks that we were requested to accomplish, we decided to set up the entire work following a pipeline structure. The pipeline developed can be split into seven main steps, each one of them represented in Figure1 will be further explained in the next sections.

Fig.1: Pipeline structure: a) Painting Detection b) Rectification c) Retrieval d) Localization e) People face and eyes detection

2.1 Pre-processing

After various initial attempts at tackling the problems that this project presented, we realized that the performances of the whole process could be improved if we managed to get better starting data. For this reason, we decided to pre-process the videos provided, in order to improve the image quality, and undistort some of them since they were shot with a GoPro camera.

2.1.1 Video undistortion

Almost every camera generates some sort of distortion to an image, this is also the case for the GoPro used for filming the videos inside the museum. This means that objects which are supposed to be straight, like the frames of the paintings, are instead curved. This is mostly due to the shape of the lens and is typically called radial distortion. This is caused by the fact that the fisheye lens used in GoPro cameras causes increased distortion as you move away from the centre of the image. There is a second form of distortion, called translational distortion, which derives from the fact that the lens is not perfectly centred and parallel to the imaging sensor. To calibrate a GoPro, two important datasets are needed: the intrinsic matrix dataset and the distortion coefficients dataset. The intrinsic matrix is a 3x3 matrix which contains information about the focal length and the principal point. The principal point is the point on the image directly below the centre of the lens. This value should be in the centre of the image but as previously said is usually slightly off centre. Both the datasets were found and saved into a *.npz numpy file, which in the project is used to provide the OpenCV undistort function with the necessary parameters. Since the undistortion moves the pixels of the images closer to the centre, missing pixels tend to occur because the distortion is extreme and there is no information outside the video frame to fill in these areas. The OpenCV undistort function crops the image so there are no missing pixels, this is noticeable in a loss of information around the edges shown in the images below.

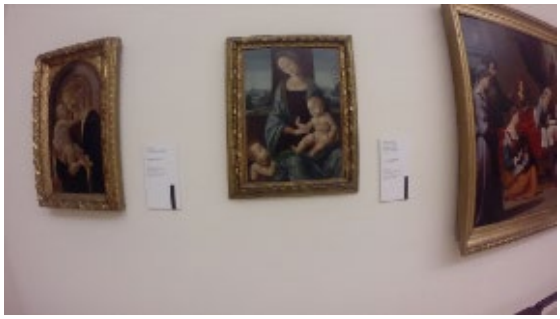

Fig.2: Frame with distortion

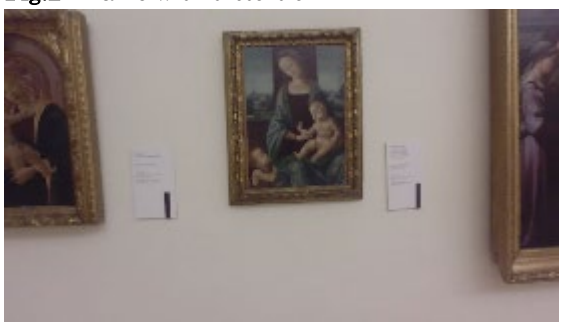

Fig.3: Frame without distortion

2.1.2 Deblurring

In an attempt to ease the process in the next steps, we decided to try to remove the blur present in the videos using a scale-recurrent network called SRN-DeblurNet, proposed by Tao et al [1]. With respect to CNN-based deblurring systems, here the network weights are shared across scales to significantly reduce training difficulty and introduce stability benefits. By doing so, the number of trainable parameters is reduced and the proposed structure can incorporate recurrent modules, where the hidden state captures useful information and benefits restoration across scales. The network takes as input a sequence of blurry images downsampled from the input image (the video frames) at different scales, and produces a set of corresponding sharp images. The

sharp one at the full resolution is the final output.

Fig.4: Frame pre deblurring

Fig 5: Frame post deblurring

2.2 Painting detection

The second step in the pipeline is the detection of the paintings. Since the speed of the algorithm was not a prerequisite of the project, the decision was to implement an algorithm that detects the paintings for every frame of the input video. The main objective of this step is to obtain bounding boxes for every painting in the frame, this means finding a ROI (Region Of Interest) for each one of them.

2.2.1 Thresholding

As a first step in the painting detection section, we tried to apply a threshold to every frame of the video, aiming at finding the edges present in the image to be able to separate the painting from the other elements in the scene. Due to the fact that various light conditions are present in the video and the point of view constantly changes, the pixel's values around the painting's frame are similar to the ones of said frame, impeding a correct separation. We evaluated two different approaches: Canny and adaptive thresholding. By empirical evidence we realized that the second option leaded to better results, in particular we decided to use an adaptive mean threshold combined with a bilateral filter to reduce the noise and preserve the edges. Although less pleasing for the eyes, since the next step of the process was to find the contours of the painting, having thick edges would have eased the operation. With this in mind, we dilated the lines found at the previous step, in order to avoid errors and create a more efficient detection.

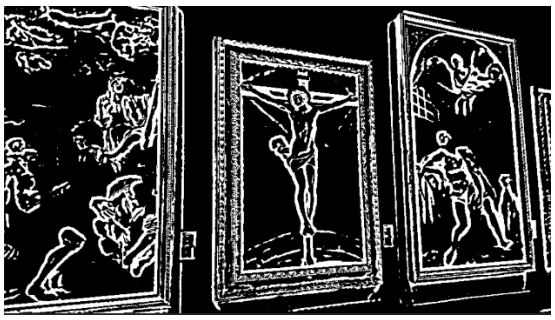

Fig.6: Thresholding

2.2.2 Finding contours

After having prepared the frame with thresholding, the actual part of the painting detection begins. To do so is necessary to find the contours of the objects, a curve for each one of them, represented by a Python list, joining all the continuous points having same colour or intensity. The contour approximation method utilized allowed us to remove all redundant points and compress the contour, thereby saving memory. After having generated a convex hull for every

contour found and approximated it into a polygon, we proceeded by drawing it into the frames to obtain the result shown in the image below.

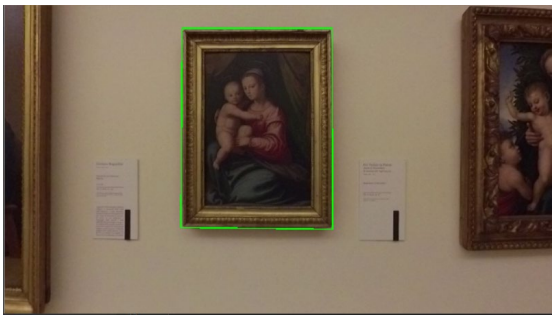

Fig.7: Painting with contours

2.3 Painting rectification

Once the contours have been found, we proceeded with the painting rectification. To achieve it we had to find the four angle points of the contour so that we could later warp the image. We then continued by cropping the picture obtained in order to provide the painting retrieval step a better input. Once that step is over, if we have a match with the database we recalculate the homography matrix to obtain an even better warp of the painting.

2.4 Painting retrieval

Once the painting's image has been correctly detected and rectified, the last step of the first section consists in matching it with the database provided, in order to determine the information needed to classify it that will be useful in the next pipeline steps. To accomplish this task, we opted to use a keypoint detector and descriptor called SURF (Speeded-Up Robust Features) by Bay et al [2]. The decision was taken due to the fact that Bay demonstrated that SURF is three times faster than SIFT while maintaining comparable performances. This is possible because with respect to SIFT, were the Laplacian of Gaussian is approximated with Difference of Gaussian, here is approximated with Box Filter, allowing the convolution of this approximation to be done in parallel for different scales. At this point a BFMatcher (Brute Force) is used to determine which image inside the database is the most similar to the one retrieved. We decided to keep only the images that had a certain amount of keypoints in common and discard the others by following the test proposed by Lowe [3], in order to avoid an incorrect matching, possible due to the fact that sometimes only the most alike image would be matched, but without it being the correct one. By looking at the matches obtained from the current frame and the database, we consider the distance from the keypoints as a similarity measure to have a confirmation of the correct result.

2.5 People detection

The next step of the project was to detect the presence of people inside the videos. To do so, we opted to use a famous real-time object detection system based on a neural network, YOLOv3. It consists in applying said neural network to the full image given as input (the single frames of the video) and dividing it into regions and predicting bounding boxes and probabilities for each one of them. After loading the YOLO object detector trained on COCO dataset, constructing a blob from the current

frame and then performing a forward pass to the detector, we obtained the bounding boxes needed and the associated probabilities. We filtered out weak predictions by ensuring the detected probabilities were greater than the minimum probability. Also in order to avoid the incorrect detection of painted people inside the artworks, we decided to delete the bounding boxes of the detections found inside the paintings' bounding boxes as shown in the image below.

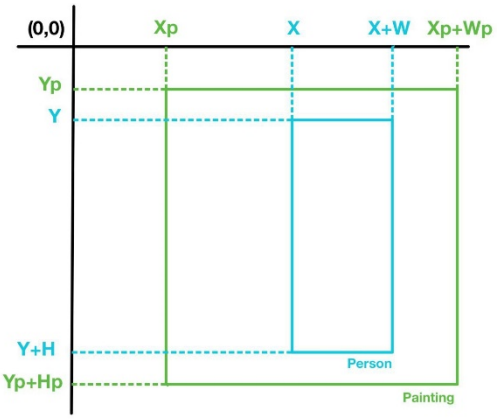

Fig.8: The person's bounding box coordinates included into the painting's bounding box coordinates

2.6 People localization

The next request consisted in localizing the people inside the museum given a map of the building. At our disposition we have the bounding box of the person detected in a video and the data of the paintings present at each frame. We look at the scores obtained by the painting retrieval and we consider the best: if this score is under an empirically predetermined threshold, we know that it's a true positive. If it is above, we are not sure about in which room we are, so we compare the results obtained by the other paintings in the frame and select the most frequent one. If two rooms are equally probable, we utilize a different approach. What we decided to do was to find the 2D distance from the person and the nearest painting. Although calculating the 3D distance between the two would have been more correct, we realized that to do that we needed parameters relative to the camera like the focal length that we could not obtain, hence the decision to opt for the 2D distance. Once determined the nearest painting, having realized the painting retrieval in a previous step, we already had all the information relative to that painting, even its location inside the building, that would be so the same of the person at that particular frame. Knowing this, we proceed at showing the map of the building with the room in which the person is, highlighted, like shown below.

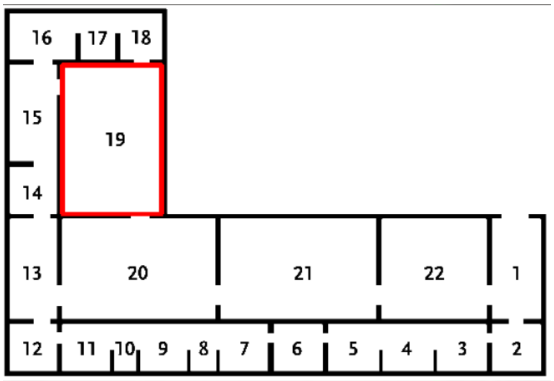

Fig.9: Map with highlighted correct room

2.7 Detecting person orientation

An optional task for the project was to determine whether a person is facing a painting or not. To accomplish this task, the solution proposed was to determine if the person is facing or not the camera: if it is, the assumption made is that the

person is not facing a painting, if it is not, the assumption made is that it is facing the painting detected at one of the previous steps in the pipeline. In order the realize if the person is facing the camera, an Haar feature-based cascade classifier has been used to detect the face and the eyes of the person. It is a classifier trained with a few hundred sample views of a particular object (in this case faces and eyes) called positive examples, and negative examples, arbitrary images. The word "cascade" in the classifier name means that the resultant classifier consists of several simpler classifiers (decision-tree classifiers) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. The Haar-like features used as input to the basic classifiers can be found in the scientific community for a lot of common objects that someone would want to detect. These features were used with the detectMultiscale OpenCV function, that detects objects of different sizes in the input image given (in our case the video frames). Since what we are trying to understand is whether or not a person is facing the camera, detecting the eyes would be an answer to the question. To speed up the process we looked for eyes only in frames in which a face has been detected first. Since the Haar-like features are not perfect, it is possible that in some frame a mouth would be detected as an eye, but this type of error does not compromise the aim of the task. Below are shown two images of the results of the detection.

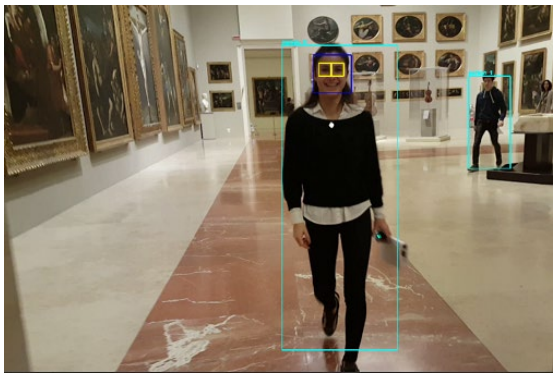

Fig.10: Correct face and eyes detection

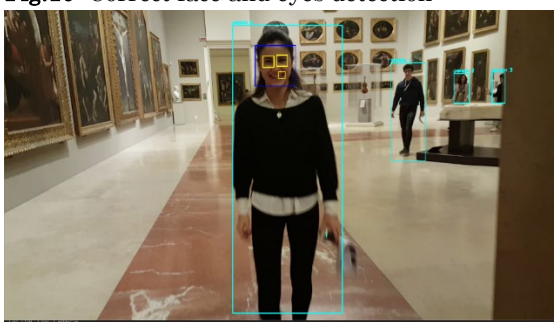

Fig.11: Incorrect face and eyes detection

3 Testing and results

In order to verify the performance of the described system, we realized various tests. For what concerns painting detection and retrieval, the main issue faced by the algorithm occurs when the correctly detected painting is not found inside the database since it is simply not present, with the results that the most similar painting is returned instead. In the section of the table concerning the paintings detection, True Positive is the number of correctly detected paintings. False positive is the number of wrongly detected paintings. False Negative is the number of wrongly non-detected paintings and the number of True Negative is assumed 0.

Video	Painting detection		
	TP	FP	FN
GOPR5826	720	0	0
20180206_113800	93	0	331
VIRB0400	413	3	76
20180206_11360	59	0	203
VIRB0395	300	0	0
IMG_9626	90	27	0
Total (1200 frame)	1675	30	610
	Precision 98,24%		
	Recall 73%		
	Accuracy 82,35%		

Table 1: Painting detection results

Video	Painting retrieval			
	TP	FP	FN	TN
GOPR5826	31	5	11	13
20180206_114604	4	0	0	10
20180206_113600	10	0	0	0
VIRB0395	0	0	0	10
IMG_2653	10	0	0	0
VIRB0392	10	0	0	0
Total (100 frame)	65	5	11	33
	Precision 92,85%			
	Recall 85,52%			
	Accuracy 86%			

Table 2: Painting retrieval results

For what concerns people and face detection, a measure to improve the results obtained at the beginning was to not draw the bounding box for people detected inside the bounding box of a painting. The following table represents the result measured by testing the detection on 15 random frames from the videos provided. The True Positive number represents the number of correctly detected people (respectively faces), the False Positive number represents the wrongly detected people (respectively faces). The False Negative number represents the wrongly non detected people (respectively faces). For what concerns the True Negative, in people detection is assumed 0, in face detection represent the non-detected faces due to the fact that the person is turning and confusing the detection.

Video	People detection			Face detection			
	TP	FP	FN	TP	FP	FN	TN
20180206_114604	32	0	5	14	0	3	5
GOPR5829	12	7	0	8	3	2	0
Total (15 frame)	44	7	5	22	3	5	5
	Precision 86,27%			**Precision 88%**			
	Recall 89,79%			**Recall 81,48%**			
	Accuracy 78,57%			**Accuracy 77,14%**			

Table 3: People and face detection results

Regarding the results obtained in the painting detection, we decided in our algorithm to not draw the bounding box of a painting if it is not completely in the frame, in order to not select something that is not a painting. Same goes for the painting retrieval, that's the reason behind some of the results. In the videos in which the image was not steady, we concluded that applying deblurring was a good idea, instead in the relatively stable videos, although pleasing, did not influenced the painting detection. For what concerns people detection, the results are satisfying, but since only few videos contained people we cannot assure the reliability and consistency of the detection.

References

[1] Tao, Xin & Gao, Hongyun & Wang, Yi & Shen, Xiaoyong & Wang, Jue & Jia, Jiaya. (2018). Scale-recurrent Network for Deep Image Deblurring.

[2] Bay, H. & Ess, A. & Tuytelaars, T. & Van Gool, Luc. (2008). SURF: Speeded up robust features. Computer Vision and Image Understanding. 110. 346–359.

[3] David Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: International Journal of Computer Vision 60 (Nov. 2004), pp. 91–. doi: 10.1023/B:VISI.0000029664.99615.94.