Report for Vision and Cognitive systems project

Enrico Ghidoni, Simone Ferrari, Tommaso Miana

Abstract

This report is based on an University project, precisely from the master's course of *Vision and Cognitive systems*. The aim of the project is to develop and execute 5 main cognitive vision tasks upon some videos taken in the Galleria Estense Museum in Modena. The first three tasks are focused on paintings. In particular: painting detection, drawing bounding boxes, painting rectification, to correct the perspective and, given a specific and annotated dataset, painting retrieval. The second part of the project, composed by the last two tasks, is about the detection and localization of people inside a video, to understand in which room of the museum they are located.

Given the structured form of this project every chapter in this paper will go through a specific task.

Introduction

A first approach to the problem was an initial understanding process to understand the pipeline needed and all the algorithms that we must use.

A particular approach has been taken in the calculation of the aspect ratio of the measures of a painting in the Painting rectification process. In this case we decided to use the algorithm described by Z.Zhang in [1] (see *Painting rectification* for further details).

After this crucial passage, the painting retrieval has been done using the keypoints and descriptors detector algorithm ORB [2].

In parallel with the operations on the paintings, two main steps are carrying on to detect and localize people presented in each frame.

The people detection step was realized using the YOLOv3 object detector pre-trained with the COCO dataset. The bounding boxes of all the people detected, as output of this task are then passed to the following step, in charge of localizing people in the museum.

In the people localization process it was decided to use the information retrieved by the painting retrieval step to infer the room of each person detected.

Architecture

The project involves different tasks and many algorithm. A visualization can be useful to better understand the complessive architecture of the pipeline.

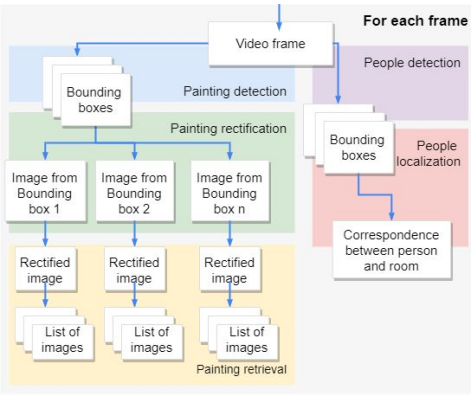

Fig.1 Architecture of the project

As outlined in *Fig.1* there are 5 different tasks:

1. Painting detection
2. Painting rectification
3. Painting retrieval
4. People detection
5. People localization

The input of the first step is the raw video, that is processed frame by frame. Each frame passes through painting detection and people detection to obtain the bounding boxes useful to create the visualization of the objects in the images.

The painting rectification and people localization steps receive the list of bounding boxes as input and return

- a list of rectified images (one for each detected painting)
- the room each person is in inside the museum

respectively.
The final step of the "paintings" branch (left in *Fig.1*) is the painting retrieval process, that receives the rectified images as input and returns a list of images ordered by similarity with the rectified image.

Painting detection
The goal of this first part is to identify the shapes enclosing paintings within each frame of a given video. The fundamental assumption of this first part of the process is that it is better to keep the detection more sensitive than to make it cut out potential painting frames. It was deliberately decided to design this step to be more prone to have false positive detections whilst minimizing false negative samples. The reason for this choice is that the following steps will only match actual paintings included in the provided database, thus filtering out non relevant objects that may be detected as paintings.
Given the above mentioned considerations, the detection part is implemented with the following steps:

1. Since the input of this section is the raw frame captured by the video recording, a Gaussian filter is applied in order to reduce the noise in the image. At first, the kernel size was set to two-to-three times the standard deviation of the image but this choice, made on the assumption that on average the standard deviation is close to values lower than ten, turned out to be too harsh in smoothing the image, thus resulting in a detail loss. The kernel size was then reduced to a 5x5 filter.
2. A thresholding process is applied to the smoothed image to separate the regions containing paintings from the background. It was not possible to manually set the threshold to a common value that would accurately separate the paintings due to the heterogeneity of the dataset (different lighting conditions, frames colours) an adaptive Otsu thresholding was used. Although being reliable in adapting to different conditions, it sometimes struggle to provide a clear separation between what would be considered as element of interest and the rest of the scene. In some cases, floor or ceiling elements are joined with a painting resulting in a poor isolation of the painting itself. By the tests performed, this point would be the single most effective change that could improve the painting detection step.
3. Before actually detecting the bounding boxes of the paintings, a dilation process is applied to the thresholded image in order to smooth out and remove small sized white areas inside a painting, providing a uniform area to be considered as a region of interest for the painting detection. This process also smoothes the edges of a painting enclosing area (similar to step n.2 in [3]) .
4. Contours are detected using the *CHAIN_APPROX_SIMPLE* algorithm from OpenCV, which greatly reduces the memory consumption [4]. At this point, small sized contours are still present despite the dilation process. Since most of them are caused by either the painting frame or the painting itself, only the outermost contours are kept.
5. Lastly, rectangular bounding boxes are approximated from the contours. In order to reduce the amount of non relevant objects detected as paintings (such as the description plaques on the walls), only the bounding boxes with

area above the average area of all the bounding boxes in the image are kept.

The presented solution gives good results, the amount of false negatives is greatly reduced and, as previously discussed, false positives do not represent an issue for the following sections. A lot is left to the thresholding technique though, which is not optimal and often yields unclean contours.

Painting rectification

This task is focused on the rectification of the bounding boxes from the previous step, all the coordinates are useful to crop the frame in *n* images. From each frame *n* images are considered, each image is obtained by a bounding box detected in the previous step.

To simplify the description, all the next considerations are about only one image for a single frame, not all the *n* images. But of course the algorithm works for every *n* images for each frame.

The steps of this task are:

1. Find the contour points in an image. The main algorithm used for this step is Canny edge detection, that works very well after an initial filtering process of the Gaussian noise in gray scale image. After Canny, the *cv2.dilate()* function was used to remove the small regions inside a painting and to obtain a well connected border of the painting, because only the external points are needed. After finding the contour points with the functions: *cv2.findContours(),* which finds contours in a binary image, and *cv2.approxPolyDP(),* which approximates a polygonal curve with precision [5]. At the end of this process there is a great deal of contour points.
2. Find the 4 corners of a painting. For this part the only thing to do is taking the 4 external corners detected by the previous step. Of course these points are the ones with the biggest difference in distance to each other.
3. Calculate the aspect ratio. The task is based on [1] which describes (in chapter 3.2) how to obtain an approximately correct aspect ratio to create a good rectified image.

   The paper presents a well defined procedure to obtain an aspect ratio and describe precisely every step. There are two main steps: calculate the *Geometry of a rectangle* and *Estimating camera's focal length and rectangle's aspect ratio*, respectively in sections 3.2.1 and 3.2.2. Section 3.2.1 derives the basic constraints from a single view of a rectangle.

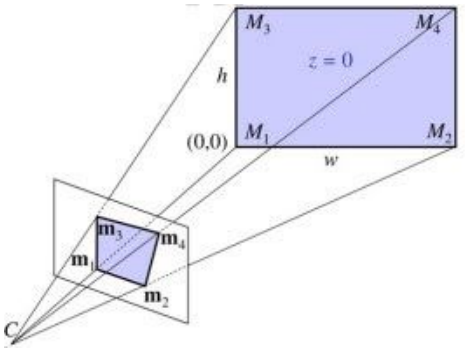

Fig.2 Geometry of a rectangle

Section 3.2.2 describes how to use these constraints to estimate the camera's focal length and the actual aspect ratio of the rectangle.

In this project, the formulas provided in the above mentioned sections were used to obtain all the parameters used to calculate a real aspect ratio.

Unfortunately, sometimes the measurement of the focal length of the camera fail and a different aspect ratio misuration was adopted (less precise, but less complicated). At the end of the process a real aspect ratio for the warping process is obtained.

4. Warp the perspective of the image. At the end, the Homography *H,* the source (4 corners calculated in step 2) and destination point (external corners of the new image) are calculated to execute the function *cv2.warpPerspective()*. Of course, the shape size of the rectified image are the coordinates calculated in the previous step.

All these processes work well for the majority of the paintings tested (92% over 102 images from video frames), with a lot of paintings both present and missing in the database. All the paintings in the procedure were taken with the frame picture, this to obviate the problem of paintings with not rectangular shape.

Unfortunately, the procedure had some trouble with the rhomboidal paintings. An issue is that sometimes the image is too blurred or the perspective is not good enough for the measurement of the real aspect ratio, that obviously fail to create a image with correct shape. Another problem is the camera calibration that sometimes causes issues in the algorithm. Also, the creation of the "stick" images is a problem, a possible solution is adding a threshold for the aspect ratio. Although it was decided not to implement such filter (see *Conclusion*).

Painting retrieval

The retrieval task was achieved using the ORB algorithm [2] which, as stated in the title of the paper itself, is an efficient alternative to SIFT or SURF. ORB is faster than the other two algorithms and it has similar matching performances, the choice fell on this algorithm mainly because it is not patented. This allows for use without any sorts of license.

ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performances. First it uses FAST to find keypoints, then applies Harris corner measurement to find the top *n* points among them. It also uses pyramid to produce multiscale-features [6].

The high speed and lightness of ORB comes with the downside of a worse invariance on image distortion. For this reason, the rectictification task previously discussed is crucial for the retrieval step to work.

The final algorithm can be summarized as follows:

1. Compute with ORB the descriptors of each image of the database. This operation of course is done only once at the start-up of the program.
2. For each rectified frame in input, the actual retrieval algorithm is carried out:
   a. First of all, to achieve better results, the contrast of every image in input is increased by a factor of 1.5.
   b. Matches detection: for each painting in the database every descriptor is associated to the most probable corresponding descriptor of the query image. This association is done by the brute force algorithm *BFMatcher*, a function of OpenCV which returns *DMatch* objects.

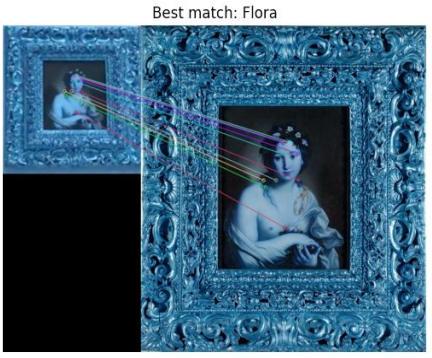

Fig.3 Example of group of matches. Every pair of points represent a match, which can be a good or a bad one, depending on his distance attribute

   c. Now to sort the painting, from the most likely to the least likely, we need a way to compute the quality of the matches that were found. To do this, the attribute *distance* of the

object *DMatch* was exploited. In fact, this attribute expresses the certainty of the match. But because for every painting multiple matches are found, the mean of all the distances for each painting is taken as baseline.

d. So now, after sorting the paintings based on the mean-distance, the result is, for each frame in input, a list of *k* paintings sorted from the one which is most probable to the one that is more different, where *k* is a parameter chosen by the user.

3. It is also needed to check the goodness of the input data, for two main reasons:
   a. The peaces of frame in input could be false positives (partial paintings, too distorted figures etc.);
   b. The paintings database is not complete, which means that the images in input could contain paintings unretrievable by default.

   It is necessary, then, to ignore in some way this two types of input.

   To do that we developed a simple but effective method: the first three best matches of the most probable painting retrieved, are taken under consideration to determine the goodness of the retrieval process. If the distances of these three ranges are all less are less than 30, then the algorithm find (with an high probability) the correct painting in the database. This number has been found empirically: a painting retrieval can be considered pretty accurate if at least the first three matches have a distance less than 30. Otherwise it means that the algorithm "is not sure" about the result, so it is probable that the image in input contains a painting not present in the database or which is too bad to be correctively associated with a painting in the database.

   The threshold as the all algorithm in general has been developed to avoid as much as possible false positives instead of the false negatives: an input image will almost never mismatch with an incorrect painting. It can happen instead, which a correct match will be considered as incorrect.

A curious final observation must be done regarding the painting "Ritratto d'uomo". This picture appears in the best 10 possible paintings of almost every retrieval operation. Also, in case of a bad input, this painting is the one which most of the times gives the best results, sometimes even too good. This bias has been taken into account in the algorithm, even if it is not rare to see the algorithm fail because of this phenomenon.

Fig.4 Example of incorrect retrieval

Fig.5 Example of correct retrieval

People detection

This task was realized using the pre-trained YOLOv3, an object detector based on a convolutional neural network called Darknet-53.

You only look once (YOLO) is a state-of-the-art, real-time object detection system. Though it is no longer the most

accurate object detection algorithm, it is a very good choice when you need real-time detection, without loss of too much accuracy, as in the case of this project. As oppose to other networks like R-CNN and Faster R-CNN, YOLO forwards the whole image only once through the network (as the name suggests). YOLOv3 is the latest version of the popular convolutional neural network and it is more accurate and faster than its predecessors. It composed by residuals blocks, upsampling layers and three different detection steps, which guarantees a very good multi-scale detection.

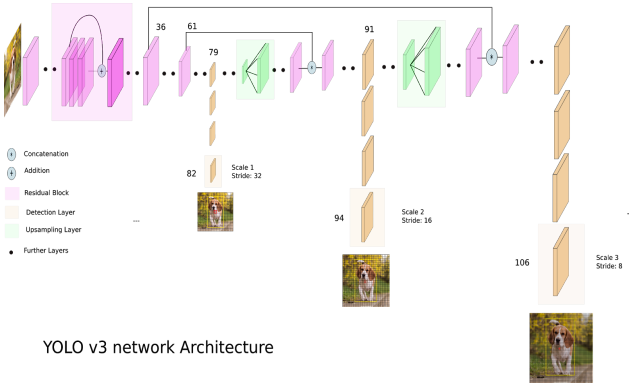

Fig.6 YOLOv3 network architecture [8]

To avoid reinventing the wheel, the network use a set of weights already trained for the detection of 80 classes of COCO dataset. Both the Python code of the network and the weights are available on the site of the course. Some changes have been applied to the configuration file to make the network detect only people, and then some refactoring of the code has been done to integrate it in the project.

The first approach, actually, was to try to re-train the network on COCO dataset, specifying to detect only people. This would have updated the weights and increased the accuracy. For a matter of time, lack of capable hardware and other difficulties it was decided to keep the pre-trained and more general weights, that however still guarantee good performances and accuracy.

As for some logical choices, it was decided to keep a relatively high *confidence* factor, for two main reasons:

- the videos are full of paintings representing people which could fool the network (the network still fail sometimes in these cases).
- Considering that this task is needed for the following one (the localization of people in the museum), it is not strictly necessary to detect one person in each frame but it is enough to have some correct detections for each video.

In other words, it is better to have some missing detection, but still manage to obtain some bounding boxes for the following task, being able to localize the room of the museum, rather than having some false positive detections of people inside a painting. To achieve this goal the *confidence* coefficient was set to 0.85.

People localization

This final task is straightforward as the information regarding the rooms of the museum are contained in the painting retrieval output. To assign a room to each person, the room numbers are retrieved from the matching paintings descriptions and the most frequent one is assigned to each detected person in the frame. This does not take into account people being spread across multiple rooms though, which is a limitation of the presented solution.

Conclusion

One particular issue that could be better addressed is the filtering of ill detected painting bounding boxes. One possible solution to the issue mentioned in the painting rectification section would be to add an aspect ratio filter in the painting detection task, removing all the bounding boxes that do not conform to a minimum aspect ratio. As example, the minimum aspect ratio of the

paintings in the groundtruth database could be used as threshold for the filter. It was decided not to implement this in the current version because not every painting is present in the database, thus some actual paintings would be left out and not detected.
It is noticeable that within the same video, the same paintings are not always matched depending on the examined frame. This is due to the motion blur introduced by rapid movements of the camera. In fact, the proposed solution has a much greater accuracy when the camera movements are performed at a slow pace. This consideration may be particularly useful in the case of the implementation of a self-driving mechanism.

Although no significant groundbreaking solution was explored to address the various tasks of the project, the results are consistent and the overall accuracy can be improved with more fine-tuning of both the techniques and parameters adopted.

References

1. Zhang, Z., Li-Wei, H. (2006). Digital Signal Processing, *Whiteboard scanning and image enhancement,* Volume 17, Number 2, March 2007, pp. 414-432
2. Rublee, E., Rabaud, V., Konolige, K., Bradski, G. (2011). Proceedings of the IEEE International Conference on Computer Vision, *ORB: an efficient alternative to SIFT or SURF*
3. *Locating & Recognizing Paintings in Galleries*, GitHub https://github.com/nating/recognizing-paintings
4. *Contour Approximation Modes*, OpenCV Documentation https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga4303f45752694956374734a03c54d5ff
5. *Structural Analysis and Shape Descriptors*, OpenCV Documentation. https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html
6. *ORB (Oriented FAST and Rotated BRIEF),* OpenCV Documentation. https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html
7. Redmon, J., Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*
8. *What's new in YOLO v3?,* Medium *https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b*