# A pipeline for detection, rectification and retrieval of paintings and detection and localization of people in the Galleria Estense

Marco Gambelli[1], Stefano Gavioli[1], and Nicholas Glorio[1]

[1]{218580, 214898, 217254}@studenti.unimore.it

## Abstract

*In this paper we analyze a pipeline to detect, rectify, retrieve paintings and detect and localize people in videos taken from different cameras inside "Galleria Estense" in Modena. We evaluate and use different Image Processing and Computer Vision algorithms for paintings and a neural network approach for people. At the end we make qualitative and quantitative considerations of our work based on which we draw some conclusions.*

## 1. Introduction

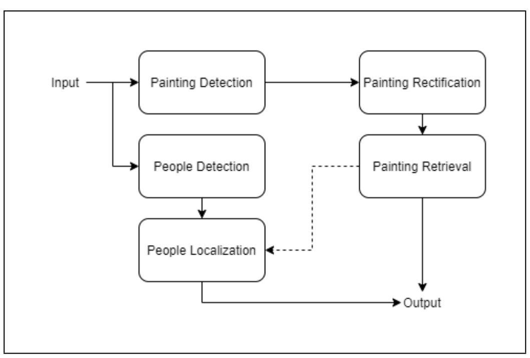



Figure 1. **Complete pipeline**

Nowadays the greater computational power and the availability of data have given space to a great variety of applications, once impossible to realize. In the area of Computer Vision in particular, we are able to process and extract useful information from images and videos.

Our attention focuses on the analysis and recognition of paintings in a museum, which can be useful, for example, if inserted in a smartphone application that provides useful information about the history of a painting and its location within the gallery, or to provide the location of people inside to ensure the safety of the place in a simple and smart way.

Our work is divided into three steps relating to paintings, which are:

2. Painting detection
3. Painting rectification
4. Painting retrieval

and two steps relating to people, which are:

5. People detection
6. People localization

The full project pipeline can be seen in Figure 1. In the next paragraphs we analyze in detail each of these steps.

## 2. Painting Detection

Painting detection is the first step inside the pipeline. It has to discover the paintings and define the corresponding bounding box. This step is fundamental and lays the base for the next steps.

### 2.1. Sub-Pipeline

The first step of our proposed pipeline is to convert the input frame into HSV and keep just the value dimension, after which a Gaussian Blur is applied to remove noise that inevitably is present either due to compression and to the camera used.

After these first steps we start by telling apart the background wall from the foreground paintings. This is done by applying a threshold computed with the Otsu method[4], that maximizes the inter-class variance. We also tried the Triangle method[10], but Otsu turned out to be by far better. The obtained foreground-background segmentation is flipped: the detected foreground is the wall and the detected background are the paintings. Before inverting the segmentation we apply a few morphological transformation: a dilation and five openings. This operation, done on the wall,

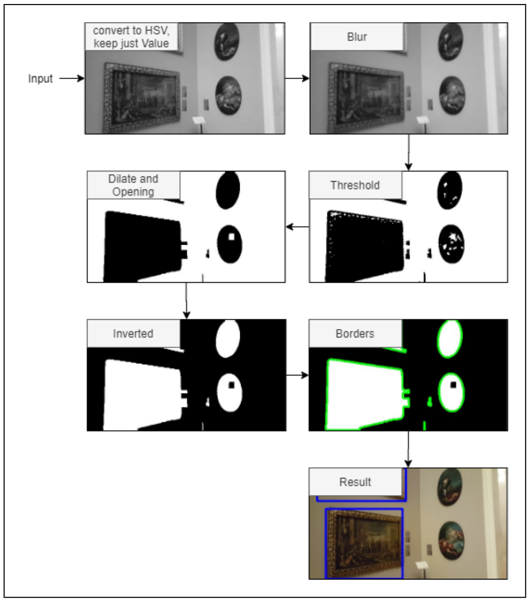



Figure 2. **Painting Detection**: Otsu sub-pipeline

is made to refine the detections by removing small size artifacts or small appendices (such as tags) that may connect to the painting, for instance due to the perspective or even shadow.

Using the Suzuki et al. algorithm[9] we find the contours from which we compute the polygonal approximation. These detection get filtered with the filters in the following section (2.2) to remove some false detections.

Among the various image processing pipelines we tested, this is the one that proved to get the best qualitative results on average. The full sub-pipeline can be seen in Figure 2.

### 2.2. Filters

We implemented several filters to remove bad detections, thus reducing false positives.

1. **Bounding box dimension**: This is a very simple filter that aims at filtering out either very small bounding boxes and too large ones. This is computed as a function of the frame dimensions $(f_{\text{width}}, f_{\text{height}})$:

$$\tau_{\text{bbd-H}} > \frac{bbox_{\text{width}} \cdot bbox_{\text{height}}}{(f_{\text{width}} - 1)(f_{\text{height}} - 1)} > \tau_{\text{bbd-L}} \tag{1}$$

2. **Average histogram distance**: To filter out those detections which are clearly not paintings we compute an average histogram on all the paintings in the given dataset and then measure the distance between the histogram of the bounding box area and the computed average one. We measure the distance between these and filter out those below a set threshold $\tau_{\text{avg-hst}}$. The distance used is the intersection:

$$d(H_1, H_2) = \sum_I \min(H_1(I), H_2(I)) \tag{2}$$

3. **Contour Area**: These filters consider the contour area, which is different than the bounding box area, and more generally $bbox_{\text{area}} \geq contour_{\text{area}}$
   (a) **Contour area**: a hard filter to filter out too small contours below a set threshold $\tau_{\text{ca}}$
   (b) **Contour area to bounding box area ratio**: Many wrong detections were defined by contours that did not cover most of the corresponding bounding box area, defining some strange shapes. For this reason a filter on the ratio of contour area to bounding box area is required to filter out those under a set threshold

$$\frac{contour_{\text{area}}}{bbox_{\text{area}}} > \tau_{\text{bbox-contour-ratio}} \tag{3}$$

4. **Number of sides/vertices**: Since we are looking for paintings we introduce a filter that removes all detections which do not correspond to quadrilaterals. This has the effect of removing other shaped paintings like rounded, hexagonal or other more exotic shapes

5. **Width to height bounding box ratio**: This is to filter elongated (either vertically or horizontally) bounding boxes that cannot represent a painting (or a part of it)

$$\tau_{\text{r}} > \frac{bbox_{\text{width}}}{bbox_{\text{height}}} > \frac{1}{\tau_{\text{r}}} \tag{4}$$

Through testing we found out and implemented the above filters with the following threshold values:

- $\tau_{\text{bbd-H}} = 0.99$
- $\tau_{\text{bbd-L}} = 0.005$
- $\tau_{\text{avg-hst}} = 3$
- $\tau_{\text{ca}} = 5000$
- $\tau_{\text{bbox-contour-ratio}} = 0.5$
- $\tau_{\text{r}} = 3$

## 3. Painting Rectification

After detection, paintings go through the rectification process. The scope is to straighten each painting before the retrieval process. We wanted to keep it as simple as possible and for this reason we decided to warp the painting vertices to the corresponding bounding box ones. This solution has obvious limitations since it is made to work with rectangular paintings (as the detection, see filter 4).

### 3.1. Sub-Pipeline

For each bounding box in a frame the painting vertices are taken from the ones of the polynomial approximation of the contour found during detection and then matched to the nearest bounding box vertex (using euclidean distance). We ensure not to pick the same painting vertex twice. Even with this condition, at times, in some tilted paintings, the vertices get wrongly associated and the rectification result is poor.

Once the vertices are matched we construct a homography matrix minimizing the back-projection error:

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 + \left(y_y' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 \tag{5}$$

This matrix is used to warp a copy of the original frame so that the painting is rectified and then cropped so that it can be used for the retrieval step.

The full sub-pipeline can be seen in Figure 3.


Figure 3. **Painting Rectification**: rectification sub-pipeline

## 4. Painting Retrieval

After painting rectification, painting retrieval begins. In this part we take the rectified painting and we match it with those of the database, returning a ranked list of results. This part is highly dependent on the previous steps, as an imprecise detection or an incorrect rectification will compromise good associations. Wrong associations can also depend on some other factors, like for example strong painting's lighting, excessive movement of the camera during shootings or video noise.

For each painting, we can extract some interesting points, called ”keypoints”, to provide a description of its characteristics. This description can then be used to identify the same painting in a frame which contains many other paintings. For a reliable recognition, it is important that the characteristics extracted from the painting are detectable even with changes in scale, rotation, or luminosity.

There are several algorithms that allow to meet all of those prerequisites. Initially we thought of using ORB[8], an unlicensed and very fast algorithm for detecting keypoints and computing the related descriptors. But in the end, we have experimentally proved that, due to the specific structure of the project, it gave us results which were not too satisfactory. Our choice therefore fell on SIFT which, although a little slower than ORB, proved to be a much more robust and suitable for our purposes.

### 4.1. SIFT

SIFT (Scale-Invariant Feature Transforms)[3] transforms an image into a number of descriptors, each of which is invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion. Major advantages of SIFT are:

1. **Locality**: features are local, so robust to occlusion and clutter
2. **Distinctiveness**: individual features can be matched to a large database of objects
3. **Quantity**: many features can be generated for even small objects
4. **Extensibility**: can easily be extended to a wide range of different feature types, with each adding robustness

According to this algorithm the retrieval process takes the following steps:

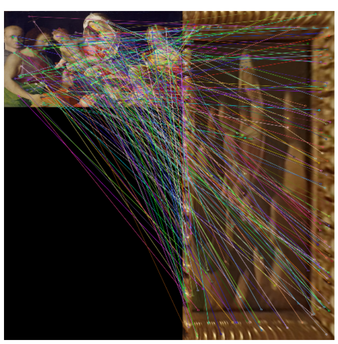
1. **Setup**: in offline mode we calculate the most significant keypoints of each database painting and their related descriptors, storing everything in a JSON file
2. **Preparation**: given a rectified painting we calculate its most significant keypoints and descriptors in the same way as before and recover everything previously saved

3. **Matching**: in order to do matching between images we use a brute force approach, which takes the best mutual correspondence between the descriptors of the video paintings and those of the database. In this way we find the best correspondences between keypoints and this process is repeated for each stored painting

4. **Retrieval**: to recover the closest paintings, we calculate the average match distance for each association, then sort the results in ascending order with respect to this distance and take the first ten results

5. **Association**: the rectified painting then corresponds to the first result of the ranked list

(a)



(b)

Figure 4. **Painting Retrieval**: examples of good associations using SIFT between the stored image (left) and the rectified one (right)

# 5. People Detection

Looking at our pipeline, the next step to perform is about people detection. This refers to the search for people in each of the frame of the input video. To accomplish good results by using a simple structure, we decided to opt for the YOLO[5] object detector, using pre-trained weights on the COCO dataset[2]. Since the weights were trained on 80 different classes, we filter out each class which differs from the ”person” one.

## 5.1. YOLO

YOLO (You Only Look Once) is a state-of-the-art, real-time object detection system. YOLO uses a totally different approach with respect to other classifier-based detectors, having several advantages over them. Firstly, the network predicts all bounding boxes across all classes for an image simultaneously, meaning that the network reasons globally about the full image and all the objects in the image. In addition, those predictions are made with a single network evaluation. This makes it extremely faster with respect to other systems like R-CNN[1] which require thousands of evaluations for a single image. This particular design enables end-to-end training and real-time speeds, while maintaining high average precision.

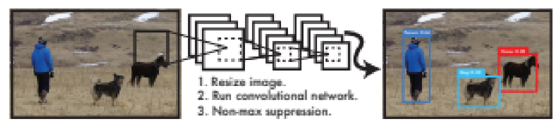



Figure 5. **The YOLO Detection System**. Processing images with YOLO is simple and straightforward. The system (1) resizes the input image to $448 \times 448$, then (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model’s confidence [7]

Specifically, citing what is written in the YOLO paper[5], the detection of an object goes through different steps:

1. Divide the image into a $S \times S$ grid.

2. Consider then that if the center of an object falls into a grid cell, then that grid cell is responsible for detecting that object.

3. Each grid cell predicts B bounding boxes and confidence scores for those. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is what it predicts. The confidence score is defined as

$$confidence = P(object) \cdot IoU_{truth} \tag{6}$$

If no object exists in that cell, the confidence scores should be zero. Otherwise, the confidence score should be equal to the intersection over union (IoU) between the predicted box and the ground truth. Each bounding box is described by 5 predictions: $x, y, w, h$, and the $confidence$ value. The $(x, y)$ coordinates represent the center of the box relative to the bounds of the grid cell. The $(w, h)$ values, width and height respectively, are predicted relatively to the whole image. Then, the confidence prediction represents the IOU between the predicted box and any ground truth box.

4. Each grid cell also predicts $C$ conditional class probabilities, defined as $P(class_i|object)$. These probabilities are conditioned on the grid cell containing an object. Regardless of the number of boxes, the detector predicts one set of class probabilities per grid cell.

Since the confidence is a user defined parameter, we decided to filter out each detection with

$$confidence \leq \tau_{\text{confidence}} = 0.8$$

### 5.2. YOLOv3

For the purpose of the project we decided to implement the third version of the base YOLO classifier: YOLOv3[7]. This uses a few tricks to improve training and increase performance like: multi-scale predictions, predicting boxes at 3 different scales and a better backbone classifier. This consists in a hybrid approach between the network used in the previous YOLO version, Darknet-19[6], and a residual network which uses successive $3 \times 3$ and $1 \times 1$ convolutional layers with some shortcut connections. This forms a significantly larger network, named Darknet-53 (see Figure 6) thanks to its total number of convolutional layers.

| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| 1× | Convolutional | 32 | 1 × 1 | |
| | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| 2× | Convolutional | 64 | 1 × 1 | |
| | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| 8× | Convolutional | 128 | 1 × 1 | |
| | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| 8× | Convolutional | 256 | 1 × 1 | |
| | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| 4× | Convolutional | 512 | 1 × 1 | |
| | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

Figure 6. **Darknet-53** used in YOLOv3 [7]

## 6. People Localization

People Localization is based both on people detection and on the retrieval results. By retrieving the paintings also the corresponding room is known. Instead of taking just the first or a random painting and assign the room to that one, we decided to have a more robust, yet simple, voting-based algorithm. This is described in Algorithm 1.

This has been implemented since there can be some errors in the retrieval and this refines the localization to have more accurate results.

**Algorithm 1: Voting Algorithm**

**Result:** Detected room
$bin_i = 0 \quad \forall i \in [1, 21]$;
**foreach** *retrieved painting in frame* $j$ **do**
  $bin_{room_j} += 1$;
**end**
**return** $\arg\max_j bin_j$;

## 7. Considerations

Given this pipeline we made qualitative and quantitative considerations on each step before drawing the final conclusions.

### 7.1. Painting Detection

For using exclusively image processing tools the qualitative results are on average quite good. We also decided to compute some quantitative measures, namely average intersection over union ($IoU$) and also an F1 score on a set of frames we extracted from the dataset and manually labeled.

Defining:

1. **True Positives (TP)**: Paintings that get detected with an $IoU > 0.5$
2. **False Positives (FP)**: Paintings that get detected with an $IoU < 0.5$
3. **False Negatives (FN)**: Paintings that get not detected with a bounding box with an $IoU > 0.5$

we obtained the following result:

| Average IoU | Precision | Recall | F1-score |
|---|---|---|---|
| 0.60 | 0.68 | 0.46 | 0.55 |

### 7.2. Painting Rectification

The rectification we did shows poor results on very warped paintings since it projects painting corners onto the bounding box vertices, resulting in very narrow rectifications.

This very simple method though provided qualitatively good results on average and helped keeping the pipeline clean and simple.

### 7.3. Painting Retrieval

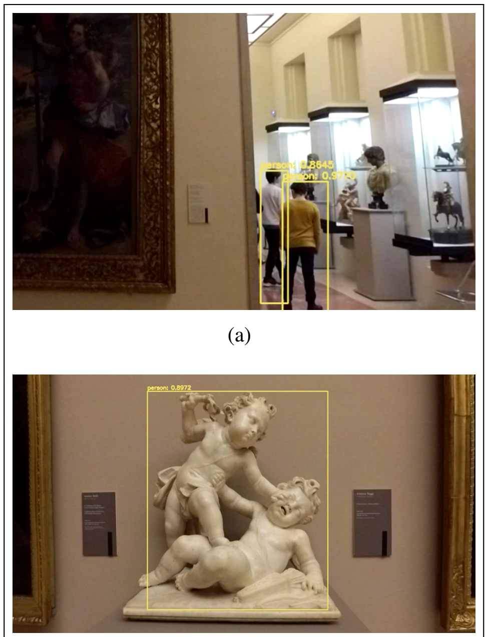
We evaluate the overall performance of the retrieval process by taking frames from some videos, trying to somehow achieve heterogeneity in the choice in order to have a more general view of its effectiveness. Then, over our test set, we take into account only the paintings belonging to the database which are correctly detected (i.e. have an $IoU$ with the ground truth $> 0.5$ and rectified and we found that:

$$accuracy = \frac{n_{\text{correct associations}}}{n_{\text{total associations}}} = 0.60 \quad (7)$$

We think that this result is quite good considering that some frames are very difficult to guess due to strong illumination, blurring, incorrect detection and rectification that sometimes occur.

### 7.4. People Detection

Using a confidence value of $0.8$ allowed us to have a qualitatively high precision (see Figure 7) while having a small number of false positives. Also, those were scenarios in which the false positives were statues or paintings depicting people. Something strictly related to humans, so, in the end, understandable for a network pre-trained on a dataset in which the "statue" class was not one of the classes.



(a)

(b)

Figure 7. **People Detection**: example of good (real people) (a) and bad (statue) association (b)

### 7.5. People Localization

The proposed algorithm (see Algorithm 1) is very robust in cases where there are few errors in the retrieval. When the number of paintings is reduced (like to one, in many cases) the localization result is even more dependant on the retrieval result.

## 8. Conclusions

This work showed the viability of Image Processing tools for present and relatively simple tasks; for others, instead, it showed the necessity for deep architecture and thus the lack of the former to be able to do all Computer Vision tasks.

The results we had, showed both the power of Image Processing and its weaknesses. Improved results may be reached by using state-of-the-art deep approaches.

This work at the end was particularly interesting for us. It represented a direct and complete application of heterogeneous knowledges learned in this sector. It can be seen by each of us as a good starting point for subsequent theoretical and practical insights.

## References


[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Zitnick. Microsoft coco: Common objects in context. 05 2014.

[3] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[4] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[6] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. pages 6517–6525, 07 2017.

[7] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. 04 2018.

[8] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[9] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985.

[10] G. W. Zack, W. E. Rogers, and S. A. Latt. Automatic measurement of sister chromatid exchange frequency. *J. Histochem. Cytochem.*, 25(7):741–753, Jul 1977.