COMPUTER VISION AND COGNITIVE SYSTEMS HOMEWORK1 FIND PAINTINGS

Damiano Caprari, Matteo Pulega, Marco Dalai
Master Degree in Computer Engineering
Dipartimento di Ingegneria Enzo Ferrari
Università di Modena e Reggio Emilia
{189652, 85755, 187466}@studenti.unimore.it

Abstract

In this paper we will explain the methodology used to extract paintings from frames taken from a museum. Our approach is based on the capability to extract the connected components which represents the objects of the frame, construct a mask to isolate each component and then compute some indices that are significant for the discrimination.

1. INTRODUCTION

The aim of our work is to detect paintings inside frames captured in a museum. Since it is an indoor situation, the luminance condition is not variable, but images presented a large illumination gradient with which we had to deal.

We tried to remove the distortion caused by the camera lenses in order to obtain straight lines even at the borders of the image.

Once the components inside the image have been found, we isolate one at a time and then we analyse the characteristics of that portion of the image to be able to tell paintings and other objects apart. In particular, since we know that all paintings have a rectangular shape, we exploited some geometrical properties that helped us to crop them away from the frame.

2. RELATED WORKS

In order to be able to detect the paintings inside the image, we exploited some algorithms that were useful considering our approach.

Here we will present a list and then we will explain them, leaving for the approach section the explanation on how we used them:

- Hough Transform-based Radial Distortion Correction (HTRDC) [1]
- Adaptive Threshold
- Optimized Block-based Connected Components Labeling with Decision Trees [2]
- Canny Edge Detection
- Probabilistic Hough Transform for straight lines detection [3] [4]
- Totally Arbitrary 3D Texture Mapping [5]

2.1 HOUGH TRANSFORM-BASED RADIAL DISTORTION CORRECTION

Radial distortion is a very big problem for our approach because it can make straight lines appear as curves. This has a great significance since we use straight lines detection to find the 4 sides of the picture.

To correct the radial distortion we implemented the HTRDC algorithm [1] which uses the Hough Transform for straight lines in order to find the best distortion coefficient k that will be used to correct the image.

The resulting image will be cropped since not all original coordinates will be mapped in the undistorted image (Figure 1).

This is an iterative method which will stop when the range of k in which to search is lower than a fixed threshold. Since we have some prior knowledge, we can narrow the range from 0 to $1*10^{-4}$, causing the algorithm to converge faster.

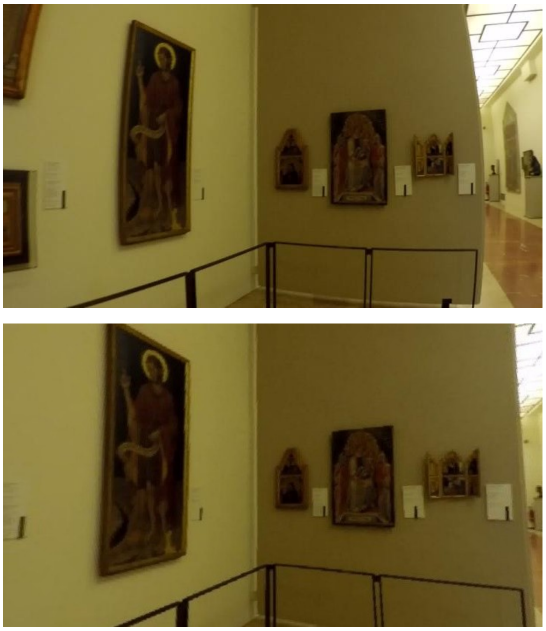


Figure 1 Above there is the distorted photo. Below is the same frame after the application of the HTRDC algorithm

2.2 OPTIMIZED BLOCK-BASED CONNECTED COMPONENTS LABELING WITH DECISION TREES

In order to obtain the connected components in the image, we used the Optimized Block-based Connected Components Labeling with Decision Trees [2]. We made this choice because this algorithm is faster compared to the Haralick algorithm and performs better in identifying the components. This algorithm is provided inside the opencv library, so we used that implementation.

The reason why this algorithm is faster is because it models the neighbourhood exploration with a Decision Table that will be converted into Optimal Decision Trees which allow to generate the code.

2.3 PROBABILISTIC HOUGH TRANSFORM FOR STRAIGHT LINES DETECTION

Probabilistic Hough Transform [3] [4] is an optimization of the Hough Transform. It takes into account only a random subset of the points. The only thing we must be careful of is to lower the threshold value since it uses fewer points.

2.4 TOTALLY ARBITRARY 3D TEXTURE MAPPING

Once the rectangle enclosing the picture is found, we need to rectify it. Due to the perspective, this sometimes leads to picture that have a dimension that is too tiny, therefore we need a method to adjust the size of the enclosing rectangle in order to obtain a good approximation of the size, despite the perspective.

This algorithm [5] tries to find a good approximation of the size of the rectangle using an iterative approach.

First of all, we need to find the 4 vertices of the rectangle, and we will name them from **A** to **D** starting from top left corner and moving anticlockwise. Through the intersection of diagonals **AC** and **BD** we find the center of the rectangle **O.** Then we have to compute the two vanishing points of the rectangle, we will call them $\mathbf{V_1}$ and $\mathbf{V_2}$. $\mathbf{V_1}$ can be computed as the intersection between lines **BC** and **AD**, while $\mathbf{V_2}$ can be computed as the intersection between **AB** and **CD**. Now we need to find the center of each edge of the rectangle as it would appear in 3D. This requires to intersect more lines in order to find the points that we will name from $\mathbf{i_1}$ to $\mathbf{i_4}$:

- $\mathbf{i_1}$ is the intersection between lines **AB** and $\mathbf{OV_1}$;
- $\mathbf{i_2}$ is the intersection between lines **BC** and $\mathbf{OV_2}$;
- $\mathbf{i_3}$ is the intersection between lines **CD** and $\mathbf{OV_1}$;

- $\mathbf{i_4}$ is the intersection between lines **AD** and $\mathbf{OV_2}$.

With those new points we can construct 4 small rectangles, we will name them from $\mathbf{R_1}$ to $\mathbf{R_4}$:

- $\mathbf{R_1}$ is (**A**, $\mathbf{i_1}$, **O**, $\mathbf{i_4}$);
- $\mathbf{R_2}$ is ($\mathbf{i_1}$, **B**, $\mathbf{i_2}$, **O**);
- $\mathbf{R_3}$ is (**O**, $\mathbf{i_2}$, **C**, $\mathbf{i_3}$);
- $\mathbf{R_4}$ is ($\mathbf{i_4}$, **O**, $\mathbf{i_3}$, **D**).

Now we select the biggest rectangle and we apply the same procedure to it. At the end we should have a rectangle small enough to minimize the perspective distortion. Then we multiply its dimensions by 2^N, where N is the number of iterations performed, to obtain the real measures of the rectangle in 3D (Figure 2).

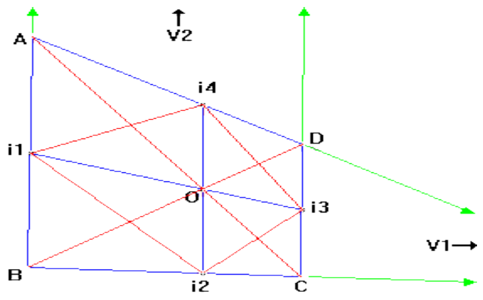


Figure 2 Computation of the rectangle size using perspective

3. APPROACH

In this section we will explain the approach and how we put together what we saw in section 2 to obtain images containing only paintings.

Since the image is distorted, the first thing we do is to apply the HTRDC method to obtain the radial distortion and correct the image. Pictures were taken with different cameras, so we do not know a priori with which camera the photo is taken and we do not have all the cameras, so we can not compute the intrinsic and all the extrinsic parameters, but with this method we can at least compute the radial distortion. Moreover, this method does not require the knowledge of the camera since it is completely based on the information provided by the image, so we can insert it in the execution of the program to compute the radial distortion on-line.

Once we have obtained the undistorted version of the image, we convert it into the grayscale version. This grayscale is blurred using a Gaussian Kernel with a strong standard deviation to delete the details of the paintings and of paintings' frames.

Even though we are in an indoor situation, pictures present a strong illumination gradient, therefore images cannot be assumed bimodal. Because of this we can not apply Otsu Binarization, but we decided to use an adaptive threshold. The adaptive threshold leaves some details of the items, to reduce this noise we apply some morphological operations in order to obtain thinner components.

After that, we compute the connected components of the image and, for each component, we compute the convex hull, which will be filled to form a mask of the entire component.

At this point we do not know if the component is actually a painting or some other item present in the museum. To discriminate them we use an entropy-based method. We extract the component from the colored image using its mask, and then we compute its histogram. Histogram of paintings will be variegated, therefore their entropy will be high compared to information labels or statues. We empirically computed a lower threshold below which we are sure that the component is not a painting, and an upper threshold above which we are sure that the component is a painting. We have a gray region in which we are not sure if the component is a painting or not, to overcome this problem we used the mean of the grayscale block. Paintings will have a low mean since they are mostly dark, so we followed the same criteria used with the entropy.

Once we are sure that the component is a painting, we compute its characteristics. In images we could have both regular paintings, paintings that are completely contained inside the image, and part of paintings. We saw that, if the painting is not fully contained in the image, it touches one of the four border of the image. To detect this, we sum the first and last rows of the component mask, as well as its first and last columns. If the sum at the borders is different from zero, the painting is not fully contained in the frame. Components which have this characteristics are marked as paintings parts. Moreover, we draw a 4 pixel black border around the frame in order to have always 4 sides for each component.

Now we can search for the corners of the paintings. To obtain that we first apply the Canny algorithm to compute the borders of the mask, and then we used the Probabilistic Hough Transform to get the segments representing straight lines. Once found these segments, we group them in two groups, based on their inclinations, with the KMeans algorithm, in this way we get groups with almost parallel lines.

Inside these two groups we search for the lines that are really parallel and we select the two that are further away from each other. In this way we obtain two groups of parallel lines with the maximum distance that may serve as the sides of the rectangle.

After having found the 4 sides, we compute the intersection between them. These 4 intersections serves as the corners of the painting. This is not a problem even for paintings parts since we draw a black border around them to be to use this algorithm.

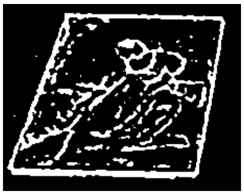
Using these 4 corners, we are now able to extract the painting from the source image and to compute an approximation of its real size in 3D using the algorithm explained in section 2.4. Once that is done, we rectify the painting.

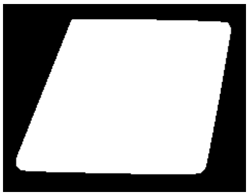
The final results are a list of all the paintings present in the picture and a mask representing the whole picture with the components segmented.

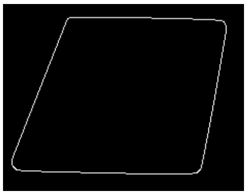
4. RESULTS

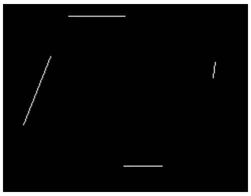
Following the steps described in section 3, the series of operations necessary for finding paintings is shown below. The process shown starts with the undistorted image because the operation of the HTRDC is already presented in Section 2.1.

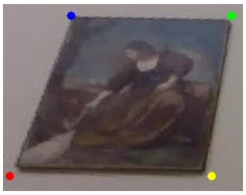
Undistorted image

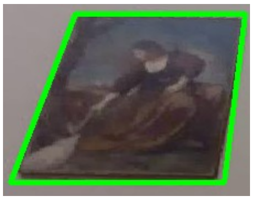

Adaptive Threshold

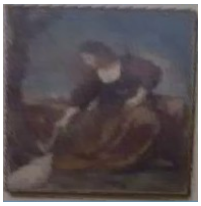

Connected Component


Canny Edge Detection

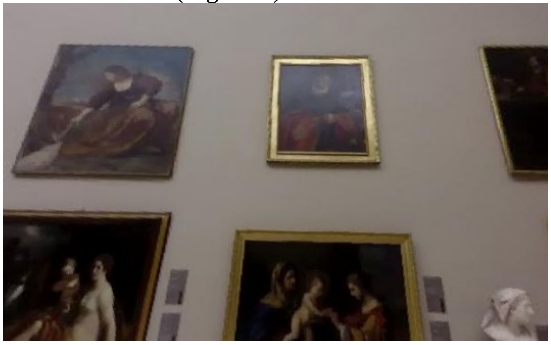

Hough Transform


Vertices detected


Detection


Rectified image

After detecting regular pictures and picture parts, as described in section 3, it is possible to create a defined segmentation and generate an image in which the paintings are clearly distinguishable from the rest of the frame. Picture parts and regular pictures are characterized by different colors (Figure 3).



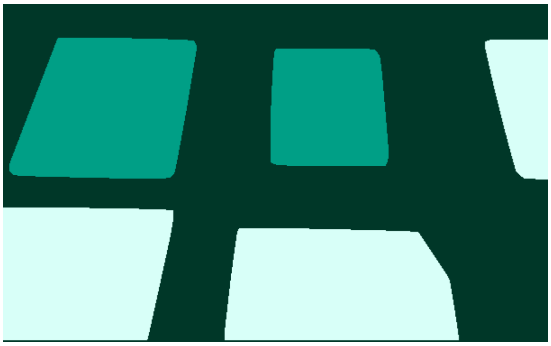


Figure 3

The algorithm was tested with a set of 50 images, frames from different clips taken in the Estense gallery. In addition to paintings, some frames contain statues, doors, people in front of targets, paintings that are very close to each other or in a perspective view. To evaluate the classification of the algorithm, we analyzed the results and classified as follows:

- a correct identification of a painting is considered a True Positive (TP);
- when a picture is not recognized it is a False Negative (FN);
- if a picture is identified but is not actually present, this is classified as a False Positive (FP).

Considering possible anomalies of detection and segmentation, we used these evaluation rules: when two or more paintings of the same category (regular picture or picture parts) are identified as a single one with the correct detection for their tyoe, the evaluation score is divided into 0.5 as TP and 0.5 as FN for each painting. Instead, if the detected category is wrong, we will consider each painting as a miss, that is a 1 FN.

With all the recorded TPs, FPs and FNs we calculated for each frame the Accuracy, number of correct predictions divided by the total number of predictions, the Recall, how many paintings were identified with respect to all those that should have been identified, and the Precision, how many paintings identified are really paintings. The goal of the algorithm is to identify paintings, not to identify where there are no paintings: so, the True Negatives (TN) are not considered and set them to 0.

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN}$$

$$Recall=\frac{TP}{TP+FN}$$

$$Precision=\frac{TP}{TP+FP}$$

Table 1 represents the averages of the different Accuracy, Precision, Recall and the calculation of the F1 score, which takes into consideration Precision and Recall of the test, keeping Regular Pictures and Picture Parts separated.

Regular Picture			
Accuracy	Precision	Recall	F1
0.583	0.618	0.605	0.612

Picture Part			
Accuracy	Precision	Recall	F1
0.585	0.736	0.667	0.700

Table 1

5. CONCLUSIONS

The proposed pipeline is used to detect paintings inside the Estense gallery. It deals very well when there is not a strong illumination gradient, which is a problem especially with corridors.

We believe that this pipeline is robust enough to be the base of a more sophisticated method, for example is possible to use this work to annotate images that will be used to train a Neaural Network, which will most likely perform better.

6. REFERENCES

[1] R. Cucchiara, C. Grana, A. Prati, R. Vezzani, “A Hough Transform-based method for radial lens distortion correction”, *12th International Conference on Image Analysis and Processing*, January 2003

[2] C. Grana, D. Borghesani, R. Cucchiara, “Optimized Block-based Connected Components Labeling with Decision Trees”, *IEEE Transactions on Image Processing*, vol. 19, issue 6, June 2010

[3] “Hough Transform and Probabilistic Hough Transform”, https://docs.opencv.org/4.1.0/d9/db0/tutorial_hough_lines.html

[4] C. Galamhos, J. Matas, J. Kittler, “Progressive Probabilistic Hough Transform for line detection”, *1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 1999

[5] “Totally Arbitrary 3D Texture Mapping”, https://web.archive.org/web/20160418004152/http://freespace.virgin.net/hugo.elias/graphics/x_persp.htm