Playing Card Detection and Identification

Necip Şükrü AŞIK
Department of Computer Engineering, Eskisehir Technical University, Eskişehir
nsa@eskisehir.edu.tr

***Abstract*— In this report I present an algorithm for playing card detection and identification. This algorithm is able to detect and identify playing cards. This algorithm is implemented in MATLAB.**

Keywords-playing card detection; image processing; template matching;

I. INTRODUCTION

It is difficult to detect and identify playing cards in a correct background and in the right direction. For these stages, the correct algorithms should be chosen and applied. 58 images used in our project. Each image shows a different playing card. Each image is at different angles and in different directions. In order to match these cards with the correct methods, ranks and suits are divided into two folder. Then tried to reach the images in datasets and use correct image processing techniques.

II. ALGORITHM

A. Binary Image Thresholding

Some of the first steps I need to consider related to image processing in Matlab are the conversion to gray level and binary level. Binary thresholding was performed using Otsu's method. It determines the brightness threshold of the image and then converted into binary.

B. Region Detection

Region detection is a method that relies on the assumption that the neighboring pixels within one region have similar values. The common procedure is to compare one pixel with its neighbors. If a similarity criterion is satisfied, the pixel can be set to belong to the same cluster as one or more of its neighbors. In this project bwlabeln function used to partition an image into regions where each region represents a card.

C. Crop and Orientation Image

I cropped the images according to the regions I set, and then resized again. I determined the orientation of the images and make calculations according to that angle. Then I rotated the cards at the right angles. In this way, card was detected.

D. Template Matching with Cross-Correlation

Card identification was achieved using template matching with cross-correlation.

III. IMPLEMENTATION

First of all, I created our ranks and suites template that I will be using for 'template matching'.

Then, it was necessary to reduce the size of the image. The dimensions of the images in our dataset are close to 3000*3000 in general. Since I want the program to run faster, I tried to reduce the size of the pixel it was reading, and firstly, I reduced the size of the image to half the size by using the bilinear interpolation. If the size of the image being read is smaller than 1500 * 1500, then enlarged the size of the image by 3 times with the 'bilinear interpolation function again. The reason I'm doing this is because some of the images in the dataset are very small. I tried to find these small images and make them the same size as other images, so I tried to reduce the margin of error.Then, for segmenting the image the image had to be in binary form. For this challenge, first; imadjust function was used for histogram equalization. The image following contrast enhancement and binary thresholding,

Since some images are rotated in our given dataset, I had performed the rotation process to make the images straight.

Then I loaded our templates and went through pre-process separately for rank and suit.

In template pre-process, I made resize templates.
Using the Otsu's Threshold method, I converted the templates binary level. Otsu's Threshold method determine the best efficient threshold related to image and then use this threshold to make the image binary. I did this steps also the uploaded image.

After the image and templates come to the shape I want, I started processing each card. First I detected the card and then tried to identify what that detected card is by using template

matching.

After the pre-process step applied to both image and templates,I divided image into labels. Each label of the card is exposed with the 'regionprops'[1] method. It takes 'convexhull', 'area', 'centroid', 'orientation', 'boundingbox' as parameters. In this way, it extracts the features of the region like area, orientation, center points in the area etc.

If the area of the region is smaller than 70000, I made it perceive it as a 'noise' and bypassed it and look at the other label. The number of 70000 is actually the approximate total pixel number of each image. In this way, I prevented the detection process from perceiving non-object locations as objects. With BoundingBox, I take the smallest box containing the area and use it for cropping. Before proceeding to the template matching process, I wanted the image and templates to be as close as possible to each other in size. So I did the resize process again.

I rotated the image again. After rotating, I cropped the card again and calculated the top, bottom row and left, right column of the card. Since the smallest value represents the highest point of the image and the largest value means the lowest point of the image, I made the adjustments accordingly:

```
topRow = min (rows);
bottomRow = max (rows);
leftColumn = min (columns);
rightColumn = max (columns);
croppedImage_og = uprightImage(topRow:bottomRow, leftColumn:rightColumn);
```

And this way the card was detected. Time for the identification.

To match the image with the template, I take the top-left and bottom-right parts of the image as an angle. I set a size_template to get the small and large suit in the label and then I started the template matching process.
Here, in general, using the normxcorr2 function, there is a similarity ratio between the image and the template (separately for suit and rank). If this ratio is more than 90 percent, I concluded that the suit and rank was found. After the template matching process, I found the middle point of the card I detected and printed what the card is in the middle point.

At the same time, I framed the detected object and displayed it.