

# IVR 1

Katrina Yankova and Francisco Vargas

February 26, 2015

## 1 Introduction

The following report aims to detail how several vision techniques are employed to extract features and classify a deck of playing cards.

As an initial stage we employed edge detection techniques to detect the bounding box of the card and crop it in order to eliminate the back ground as much as possible.

Furthermore we employed auto-calibrating threshold based segmentation to extract the bounding boxes of the features within the cropped cards. The final stage consisted in extracting image signatures and complex moments, allowing us to create feature vectors for classification via Naive Bayes. The classification via Nave Bayes yielded excellent results due to the high quality feature extraction and astute noise reduction techniques employed throughout this practical.

We used a great chunk of material from the course slides, the lab code and inbuilt matlab functions; This being said we did not apply at any time the 'black box' of employing things without understanding. In to depth research was applied for every inch of this practical along with a bit of hacking and creativity, which we hope to justify in the following chapters.

## 2 Methods

### a Bounding Box and Image Cropping

Since the cards have well defined edges it seemed like a good idea to detect these edeges employ its connected region as a bounding box and crop out the original greyscale card.

The laplacian takes the general definition of  $\nabla^2 = \sum_{i=1}^n \frac{\partial^2}{\partial x_i^2}$  which for our case we are only interested in two dimensions thus  $n = 2$ . The continuous Laplacian of the gaussian kernel is defined as follows:

$$\nabla^2 \mathcal{N}(\underline{x}; \sigma) = -\frac{1}{\pi\sigma^2} \left[ 1 - \frac{x^2 + y^2}{2\sigma} \right] e^{-\frac{x^2 + y^2}{2\sigma}}$$

This formula is used to approximate a discrete kernel, we have the following example:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

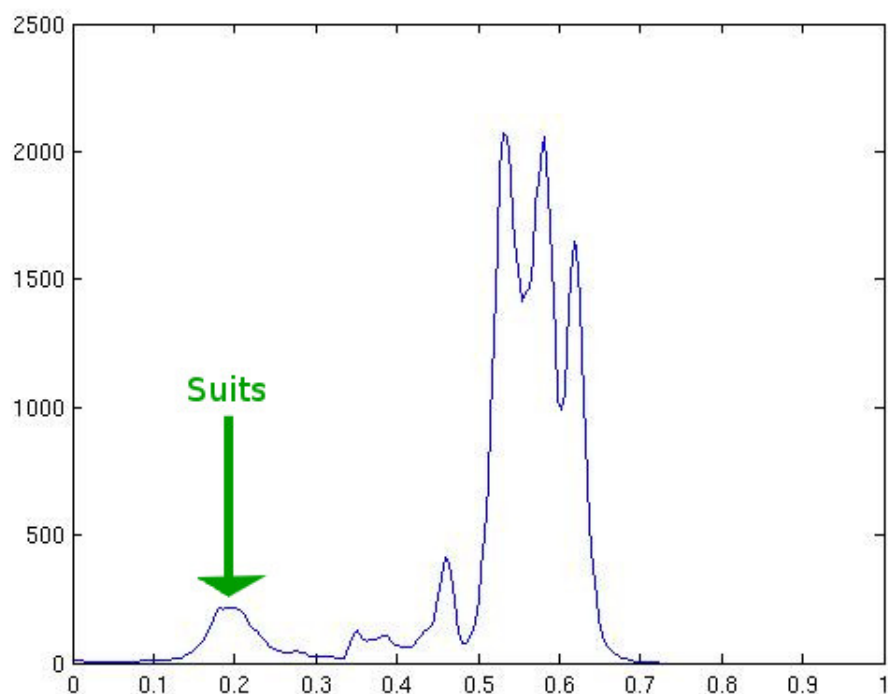
The kernel is applied via 2D convolution with the image. In a sense one may picture this procedure as first convolving with a Gaussian and then applying the laplacian which aims to find the areas of rapid change in other words the edges.

```

1 I = rgb2gray;
2 % binary image yielding
3 bw = edges(I, 'log');
4 % remove noise using open
5 % open is a mix of both erosion and dilation
6 bw = bwareaopen(bw, 15);
7 % label connected regions
8 lables = bwlabel(bw);
9 %{ obtain properties for lables
10 such as BoundingBox for example
11 Which is later used to crop the
12 greyscale of the original
13 image (normalized by rgbnorm.m)
14 %}
15 properties = regionprops(lables, 'all');
```

## b Thresholding the Features

Before thresholding the features the histograms we inspected the histograms and by a method of trial and error we devised the location of the suits within the cropped card and tweaked the findthresh.m algorithm to target the correct valley. It is important to note that our gray intensity is conveniently between 0 and 1 due to a witty cast. The histograms were smoothed before iteration over then and as we went through the project:



The tiny peaks within a bigger peak were sources of distortion and mis-calibration thus we increased the size of the Gaussian kernel to blur these out more when convolving. Empirically we determined the kernel size to be 1x40 and it yielded histograms of the following form:

INSERT IMAGE

After thresholding we removed small area (less than 150 px) elements and used the following morphological transform.

```
1 bw = bwmorph(bw, 'majority');
```

An additional noise removal technique (for long rectangles) was to look at the mayor axis to minor axis ratio and check that it is not greater than 3. If it was greater than 3 we removed it. All these constants for removal were determined empirically throughout the practical.

We then found the bounding boxes for the remaining elements and cropped the original image (gray scaled) for each single one of these features. On every new cropped feature the following was applied to threshold

```
1 % Inverted global threshold
2 % graytrhesh computes global
```

```

3 % threshold level.
4 % we invert this to extract the suit
5 bw = ~im2bw(Icropped,graytrhesh(I));

```

resulting in: INSERT PICTURE HERE

### c Construction of Feature Vector for Suit classification

For every suit symbol in each card we generated a feature vector. For example 2 of diamonds would ideally yield 4 feature vectors. The feature vectors consisted of 4 dimensions compactness, the first two rotation and scale invariant moments and the average of the normalized red channel of the suit which is closest to the center. The reasoning behind this is that the region detected closest to the center should be a suit (which yields a lot more pixels with color than a number); The closer you are to the center of the card the less likely the region is noise (again this is based on empirical evidence but also some common sense).

$$\underline{x}_{feature} = \langle compactness, ci_1, ci_2, \mathbf{mean}(red\_channel) \rangle$$

The other  $ci_k$ ,  $k \in [3, 7]$  yield significantly small values along the order of magnitude of  $10^{-5}$  and produced no changes in the classification problem, hence they were discarded due to their poor discrimination power. The normalization of the red channel before averaging accounts for dealing with brightness. We chose not to build two different classifiers since it seems quite hacky to identify red and black by assuming there normalized rgb values are approximately 0.3333 and 0.6 (which they are nonetheless when averaging within the bounding box of the suits these values may be averaged out resulting in confusion).

### d Counting the number

Out of 480 features (+ approximately 30 incorporated later) the detection of the features was perfect it drew the bounding box on all features and no noise created any issues. This fact lead us to deciding that counting the number of regions and subtracting 4 was the correct decision for determining the number of the card. We opted out of naive Bayes since it is almost impossible to differentiate between 9 and 6 since they are exactly the same. A possible solution to classification via machine learning would be to compute the global moments of the card and incorporate them as dimensions since the 6 has less symbols than 9 the global moments and other global signatures could have relatively high discrimination powers.