Virtual and Augmented Reality

Augmented Cards Game: Project Report

Work Group:
Hugo Cardoso - 201105625 - ei11154@fe.up.pt
Vasco Gomes - 201106906 - ei11161@fe.up.pt

Faculty of Engineering of the University of Porto
Masters in Informatics and Computing Engineering
Roberto Frias, sn, 4200-465 Porto, Portugal
Professors: António Augusto de Sousa and Jorge Alves da Silva

December 18, 2015

1 Introduction

The main objective of this project is the development of an application capable of assisting the playing of a simple card game. The application should be able to acquire an image of the cards played during a round of the game and announce the result, according to the game rules, by augmenting the acquired image with some information.

The application here described was fully developed in C++ using *OpenCV* (Open Source Computer Vision) version 2.4.

2 Proposed Algorithm

The algorithm proposed in this report is structured in three phases: image segmentation and extraction, card recognition and deck database preparation.

2.1 Image Segmentation and Extraction

Image segmentation is applied directly to the image acquired by the application. The processed is described in the steps below:

- **Noise removal**: achieved by running three different methods: histogram equalization, blur and threshold functions. The equalization enlarges the intensity range in the frame and is followed by a Gaussian function that functions as noise removal. The result is then filtered with a binary threshold.
- **Edge detection**: edges are detected by applying the Canny Edge Detection algorithm.
- **Contours extraction**: the edges from the previous step are then converted to lists of points, or contours, by using the *Suzuki85* algorithm. The resulting list is ordered by contour area, assuming each contour as a list of points that defines a closed section.
- **Cards extraction**: in order to allow for better comparisons with the deck, each card is converted to a new front perspective. Initially, a rectangle is extracted from a card's contour, using the equations for the lines formed between pairs of points. The rectangle's points are then mapped to a set of predefined points used to generate a new image with a single card in the correct perspective. An example of this process can be seen in figure 1.1.1.

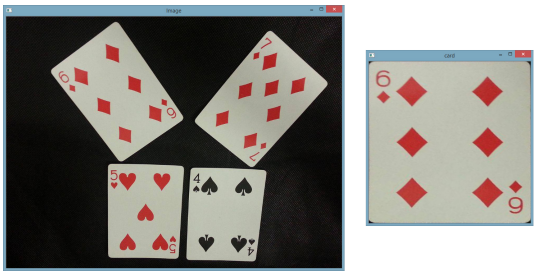

Figure 1.1.1: Example of the result of the extraction of a card in a given image.

2.2 Card Recognition

For card recognition, two methods have been implemented: binary absolute difference and *SURF* (Speeded Up Robust Features). Binary absolute difference attempts to find the lowest number of differences between two images in a pixel by pixel basis, while *SURF* based detection attempts to find the highest number of matches. Both methods use the same process for image segmentation.

In each method, detected cards are compared against all the cards in the deck database.

2.2.1 Binary Absolute Difference

- **Pre-processing**: achieved by applying a conversion to gray-scale, Gaussian function and adaptive threshold.
- **Matching**: given by the absolute difference, per-element, between two cards, resulting in a binary image. To compensate for slight differences in alignment, the resulting difference goes through one more iteration of Gaussian and threshold functions. This process returns the number of different pixels in the binary image (represented with a *1*).
- **Improvements**: As some cards are not symmetric, the matching process is also applied to vertically flipped versions of the detected cards.

2.2.2 SURF

- **Pre-processing**: this method requires no previous processing. Cards are compared directly.
- **Matching**: given by the number of matches between two cards. In order to reduce the number of incorrect matches, or outliers, two filtering processes are applied: by absolute value (distance) and *RANSAC* (Random sample consensus).
- **Improvements**: generally, computing features and descriptors for the matching processing is a slow process. As such, these values are initially calculated for all cards in the deck and then stored for later use.

2.3 Deck Database

Databases were created by the application of the corresponding method to images containing all the cards in a given deck. Although the creation of new databases is available, two databases are already provided:

- **Binary Absolute Difference**: stored as a single image containing side by side representations of all cards in the deck after a pre-processing phase.

Figure 1.3.1: Deck database used during the binary absolute difference method.

- **SURF**: stored as a single image containing side by side representations of all cards in the deck with no previous processing.

Figure 1.3.2: Deck database used during the *SURF* method.

3 Experiments, Status and Improvements

While developing the algorithm, several experiments were designed and tested to improve the efficiency of card recognition. Some relevant notes and conclusions are described below.

The binary absolute difference method is a good fit for images acquired from top perspectives. However, it has difficulty with images acquired at an angle. In these images, the extraction processed is unable to compensate for alignment differences. Although slower, *SURF* provides better results for such cases.

The extraction process is heavily reliant on the computation of a card's corners (rectangle). *OpenCV*'s internal method (*minAreaRect*) does not consider angled perspectives where rectangles have opposing sides with different sizes, so several experiments were conducted to develop an alternative. Two methods were found: the first one attempts to find the four lines (and respective intersections) that define a rectangle by comparing lines formed between pairs of points in a contour, while the second one calculates the diagonals by finding points with the largest distance in-between them. The solution here presented uses the first method.

The application was developed with re-usability in mind. In order to adapt to multiple games, the application represents a game as an abstract class so that multiple games can easily be incorporated. New methods and training decks can also be added without having to reorganize the code structure.

Taking into consideration both the complexity of the algorithm and its results, it is believed that it mostly succeeds in fulfilling all the tasks proposed. Considering the balance between planning, projections, work finished and work remaining, it is believed that 100% of the goals have been realized, and that the project has finished successfully.

4 References

[1] Opencv.org,. 2015. "Opencv — Opencv". http://opencv.org/.

[2] Docs.opencv.org,. 2015. "Welcome To Opencv Documentation! — Opencv 2.4.12.0 Documentation". http://docs.opencv.org/2.4/.

[3] Arnab Nandi,. 2015. "So I Suck At 24: Automating Card Games Using Opencv And Python — Arnab Nandi". http://arnab.org/blog/so-i-suck-24-automating-card-games-using-opencv-and-python.