Nicolas Früh
Felix Graule

Vision goes Vegas

Abstract

The goal of this P&S-project[1] was to program a computer to play blackjack. The computer received a photograph of the current card lying on a green carpet. The C++ software we developed was then able to detect the card, analyse it and take a decision in the blackjack game (either "stay" or "card"). For this project it was not necessary to distinguish different colour or face cards, as only the value of the card is relevant. We worked in groups of two persons, two ECTS-points were awarded.

How we analysed the card

Step 1: Detecting where the card is

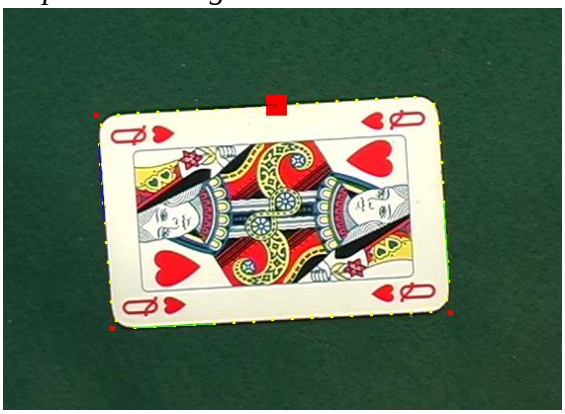

Figure 1: Boundary points (yellow), corners (red) and seed point (big red) on a card

To detect the card, the program starts in the middle at the top of the image. It checks the colour of the pixel and if it is too dark (meaning carpet) it moves to the next pixel below, until it finds the border of the card (a bright enough pixel). This process is repeated with different starting positions if necessary. Beginning from the seed point we just found, the program takes a radius and an angle from the seed point and checks whether the point is on the card or not (by analysing the brightness). If not, it will adjust the angle a bit and check again, until it detects a border. Using this method, we can detect the orientation (angle) of the card and follow its border which produces the yellow boundary points in the image. By fitting lines through these points and intersecting them, we find the four corner points that we use in the next step to extract the card from the image.

Step 2: Transformation of the image

Figure 2: card has been transformed

Once we have found three corner points of the image, a simple affine transformation as described by the matrix below is used to crop and rotate the card.

$$\begin{pmatrix} x_{\text{new}} \\ y_{\text{new}} \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{pmatrix} x_{\text{old}} \\ y_{\text{old}} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

In the equation system, there are six unknowns. By inserting the three boundary points, all unknowns can be determined and thus every pixel in the old image can be mapped to a pixel in the new (right part in Fig. 2) image.

Step 3: Creating a binary image

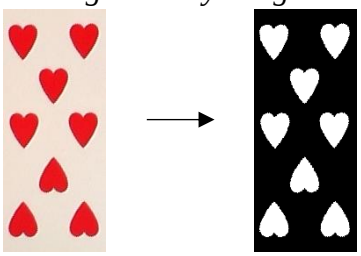

Figure 3: Binarised image

Being able to bring every card into a standardised position, we now want to count the number of symbols on the card. The type of symbol or it's colour are not relevant for our task. Thus, we remove all colour information and get a black and white (binary) image. The thresholds to do this have been chosen by testing, but could—in a more advanced system—be recalculated for every picture which would make the program more robust.

[1] https://www.ee.ethz.ch/studies/bachelor/third_year_regulations_2012/Practical_projects_seminars.html

Step 4: Labelling the image

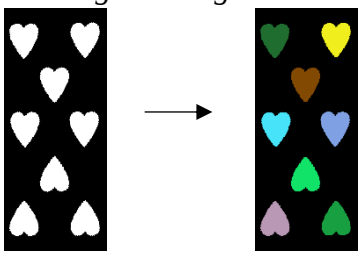

Figure 4: Labelled image. Connected pixels bear the same colour

The last step in our analysis is to find connected clusters of white pixels (these would each represent one shape). To do so, we walk through each pixel row by row and assign it a number. Beginning by one, the number stays the same as long as the pixels are white. If a black pixel is hit, we assign 0 to it but increase our counter for the white pixels. Thus, the next connected row of white pixels will be assigned two, and so on. In a second walk-through, we check for neighbouring white pixels in every possible direction (up, down, left, right) and create a table of pixel-numbers that neighbour each other. This table allows us to reassign new numbers to each cluster. Now, each shape is assigned a unique number. Very small pixel-clusters are neglected (they would most probably not be actual shapes but errors due to reflection). Finally, we can count the symbols and output the card value.

Handling of face cards

Face cards need a completely different approach, since we cannot count shapes there. We came up with a rather simple approach: After step 2 (transformation of the image) we perform a colour analysis. If certain thresholds of red, blue and green are exceeded after adding up all the colour values of every pixel in the image, we can decide if it is a face card. The type of the face card is not relevant for blackjack.

The thresholds have been determined by experimenting. Thus, our approach only works for the specific set of cards and lightning conditions we were working with. To generalise our solution, more advanced analysis would be required.

Playing the game

Once we could detect the card value, the implementation of the game was straight forward. We did not implement a complex strategy but looked at the first card of the dealer and classified it. If the card was good for the dealer, we were to play offensively, otherwise defensively. The program was finally tested in a proper blackjack game against our assistant, Andras Bodis-Szomoru.

Conclusion

The project was very educative. We could solve the task and our program performed well at the final game. Nevertheless, we understood that image recognition is not trivial and that there are many problems and obstacles one has to think of.

We were provided with a basic image processing library written by our assistant Andras Bodis-Szomoru. All the other code is our own work.

For more information, refer to http://www.vision.ee.ethz.ch/~ppsadm/