UDC 004.932

SOFTWARE FOR SCAN BANK CARDS IN IOS MOBILE DEVICES

ALEXANDER KURILOVICH, RYKHARD BOHUSH
Polotsk State University, Belarus

In this article we propose the preprocessing algorithm for identifying information fields on the bank cards images. Bank cards details recognition software for iOS mobile devices, iPhone SDK frameworks, OpenCV and Tesseract libraries is implemented based on this algorithm. Results of the time cost for bank cards details recognition on the iPhone7 for video with different sizes of the input frame are presented.

Nowadays many M-banking apps require manual entry of bank card details (all information fields data applied on the card) into the system for payment transactions and it is not an easy task for a user. This process is time-consuming, requires attentiveness and diligence. Therefore, development of the algorithmic provision and software for recognition of bank card details for iOS mobile devices is relevant.

Offered recognition algorithm of bank card details for iOS mobile devices requires the following steps: card region detection; segmentation of the bank card image; converting colored segments to grayscale; contrast enhancement; emphasizing the boundaries of symbols using operations of mathematical morphology; adjusting the boundaries of the grouped blocks of symbols; recognition of symbol blocks by the Tesseract library. General scheme of the developed algorithm is shown in fig. 1.

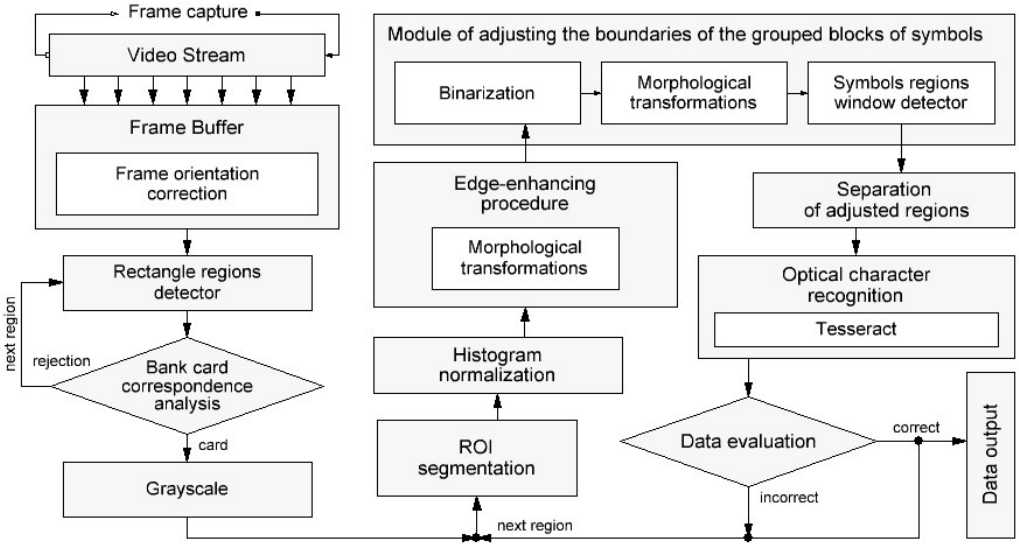

Fig.1. General scheme of the developed algorithm

The input data of the algorithm is video stream received from the mobile device's camera. The stream is divided into frames by time and represents images in the RGB color space. Considering characteristics of the iPhone 6 (2014), the camera has an 8-megapixel CCD, dual-LED flash and autofocusing. The frame sizes can vary depending on the particular device model, and can reach 3840 × 2160 pixels. All devices younger than the above have better characteristics. The operating system (greater than iOS 2) is capable of automatically holding the focus on the subject, automatically adjusting the brightness and white balance in the photo. The above indicates that the image coming directly from the camera system will have the optimal input parameters: depth of field, brightness, sufficient frame size for accurate perception the details of the bank card.

Separation of the bank card image from the background is carried out by processing it with iOS algorithms, which are based on the Viola-Jones high-speed object detection method [1] and deep learning using the OverFeat method [2].

The main criterion for selecting a contour of the bank card from the set of detected rectangles *R*, where the number of rectangular areas found- *N*, is the ratio of its sides. Since dimensions of the card sides ($m_o \times n_o$) are defined by ISO / IEC 7811-1: 2017, the ratio of its sides is a constant value $\frac{n_o}{m_o}$. At first we represent the object of the found rectangle r_i as $R_i\ (m_i, n_i)$, where $i = 0, 1, ..., N-1$. Then the rectangles *r* for which this condition (1) is satisfied belong to regions of interest. The exact match is the rectangle with the longest side n_{max} among all selected.

$$\frac{n_i}{m_i} = \frac{n_o}{m_o} \tag{1}$$

Dimensions of the rectangular region $r_o\ (m_{max}, n_{max})$ and its location relative to the whole image $p_o\ (x_o, y_o)$ are used to extract the card region from original frame.

Regions of algorithm interest, containing information about the bank card number, the expiration date, the cardholder's name, are being separated from the card image *I* with sizes $m_{max} \times n_{max}$ (further as $m_I \times n_I$). The size and location of these regions are defined by ISO / IEC 7811-1: 2017 and can be written in the following presentation:

- region $C(x_C, y_C, n_C, m_C)$ - the bank card number;
- region $D(x_D, y_D, n_D, m_D)$ - the expiration date;
- region $E(x_E, y_E, n_E, m_E)$ - cardholder's name.

Since the sizes $m_I \times n_I$ of the obtained image of the card *I* can vary, it is necessary to convert *C*, *D* and *E* to the form that will be applicable for their correct separation from the image *I*. To this end, it is necessary to define two scaling factors: by width (2) and by height (3).

$$C_{scale.width} = \frac{n_I}{n_o}, \tag{2}$$

$$C_{scale.height} = \frac{m_I}{m_o}, \tag{3}$$

Multiplying each of regions parameters indicated above by corresponding scaling factor, we obtain the sizes of these regions relative to the size of the obtained image of the card *I*.

$$C_I(x_C \cdot C_{scale.width}, y_C \cdot C_{scale.height}, n_C \cdot C_{scale.width}, m_C \cdot C_{scale.height})$$
$$D_I(x_D \cdot C_{scale.width}, y_D \cdot C_{scale.height}, n_D \cdot C_{scale.width}, m_D \cdot C_{scale.height})$$
$$E_I(x_E \cdot C_{scale.width}, y_E \cdot C_{scale.height}, n_E \cdot C_{scale.width}, m_E \cdot C_{scale.height})$$

Separating fragments of regionsC_I, D_I, E_I from the image of the bank card *I*, we obtain images: I_C, I_D, I_E, respectively, the card number, the expiration date and the cardholder's name.

According to ISO / IEC 7810 - 2006, the bank card number contains 16 digits distributed into 4 equal groups of 4 digits each. Based on this fact, the image I_C selected in the previous step with the card number region C_I is being divided into 4 equal regions C_{I1}, C_{I2}, C_{I3}, C_{I4} (4).

$$C_{Ij}(x_{CI} + (\frac{n_{CI}}{4} \times (j-1)), y_{CI}, \frac{n_{CI}}{4}, m_{CI})$$

$j = 1,2,3,4$ (4)

Separating the calculated regions C_{I1}, C_{I2}, C_{I3}, C_{I4} from the image I_C, we obtain respectively the images I_{C1}, I_{C2}, I_{C3}, I_{C4} - groups of the card number.

The next step is to convert the image to grayscale. After that, to increase the brightness difference between the symbol contours and the background of the images, a procedure of contrast increasing is used. The most effective for this task is method of histogram normalization, since it does not stretch the entire intensity range, but only it's most informative section. This allows us not to consider the true extremal values of the

brightness and to designate conditions for their determination with a given accuracy, which enhances the contrast effect due to the loss of noise regions with rarely encountered intensities [3].

To determine the tone of the background and the tone of the symbols we calculate the average brightness of the image $Y_{aver.}$. If $Y_{aver.} > 127.5$, then we take the background color of the received image as light, and the color of the symbols as dark, otherwise - vice versa. Using operations of mathematical morphology, the boundaries of symbols are emphasized. If the color of the characters is light, the WhiteTopHat morphological transformation function [4] is used. For dark symbols, the BlackTopHat function is used.

The structuring element for the kernels of both filters has a rectangular shape and a size of $n_b \times m_b$, calculated by the formula (5), where n_I, the width of the image received for the morphological transformation.

$$n_b = n_I \cdot 0.06$$
$$m_b = \frac{n_b}{3} \quad (5)$$

The next step is the binarization of images with an adaptive threshold based on the analysis of the local region [5, 6]. The method converts the image in grayscale *I* to a monochrome image according to (6).

$$bin_{x,y} = \begin{cases} g_{max}, if\ Y_{x,y} > T(Y_{x,y}, block\_size, c) \\ 0, else \end{cases} \quad (6)$$

where $Y_{x,y}$ is the brightness value of the pixel with the (*x, y*) coordinates of the image *I*; $g_{max}=255$ is the maximum brightness value, $T(Y_{x,y}, block\_size, c)$ is the adaptive binarization threshold, calculated individually for each pixel; where the function *T* - is the weighted sum [5] (cross-correlation with the Gaussian window, Gaussian (7) [6]) of a block of pixels with size *block_size×block_size* adjacent to the processed pixel (with coordinates *x, y*) minus the coefficient c [5].

$$G_i = \alpha \cdot e^{-\frac{\left(i - \frac{(block_{size}-1)}{2}\right)^2}{2 \cdot sigma^2}}$$

where i=0... block_size-1,

α *- is chosen in such a way that* $\sum_i G_i = 1$ (7)

To reduce noise, to remove unnecessary non-informative details from the binarized image, and roughly approximate the symbols to isolated painted blocks different from the background of the image, successive morphological operations of closing and erosion are applied to it.

Structuring element *b* of the kernels of both filters has an ellipse shape, which is better suited for processing characters which font, OCR-B, according to ISO / IEC 7811-1: 2017, has smoothed edges. The size of the kernels $n_b \times n_b$, where n_b - should not exceed the size of a third of the thickness of the symbol $\frac{w_{symb}}{3}$, otherwise there is a probability of losing the symbol outline during processing.

The thickness of the symbol w_{symb} does not depend on the type of the processed fragment of the card image, and is strictly defined for OCR-B. The projection of the font width value (mm) to the size in pixels relative to the size of the full bank card region is calculated based on its full width (8), and is a constant within the processing of the current card region.

$$w_{symb} = n_0 \cdot 0.004884 \quad (8)$$

To adjust the edges of the symbol region on the processed image, the vertical sliding window method is used. The height of the window varies from the font size of the characters. For the bank card, these are values determined by ISO/IEC 7811-1: 2017, H_{cn} - card number font height (4,0mm), H_{exd}- expire date font height (2,85mm), H_{hn}- cardholder's name font height (2,65mm). Then their projections onto a height in pixels relative to

the size of the region of the full card are: h_{cn}, h_{exd}, h_{hn}. The general form of the calculation is represented by formula:

$$h = \frac{H \cdot m_0}{54.0} \tag{9}$$

where 54.0 mm is the height of the card according to ISO 7810 ID-1, and m_O is the height of the detected region of the card (px).

Images are sequentially transmitted to the input of the Tesseract system for recognition, with parameters: the adjusted symbol area obtained in the previous step; language.

The proposed algorithm is implemented in the Objective-C programming language, using the OpenCV 2.4.13 library (this version is the most relevant for iOS), with the help of the iPhone SDK frameworks, such as: CoreMedia and AVFoundation - media data management; UIKit (UI) - work with application interfaces; CoreGraphics (CG) - low-level, lightweight processing of 2D images based on the Quartz engine.

The mobile app interface (fig. 2) is equipped with an image viewing area, captured by a mobile device's camera in real-time, a data output area, a mark of successful fixation on the recognition object. The capture area has a proportion of 4:3, which is the standard for a vertically oriented iPhone/iPad. The data output area contains three vertically arranged text fields into which the information recognized by the algorithm (the bank card number, expire date, cardholder's name) is sequentially displayed. The card detection mark is made in the form of a bright green rectangle with fixed thickness of borders and is displayed on the screen by repeating its boundaries when the card position is successfully determined. The capture of the video stream from the main camera of the device is implemented in separate thread using the AVFoundation framework and the AVCaptureDeviceInput object, which is initialized by the AVCaptureDevice object in the video capture mode. A new session of AVCaptureSession is being created, with the frame size parameters (for iPhone 7, the maximum size is 3840 × 2160). Using the addInput method, a previously defined input object is added to it and via the call startRunning - the session starts. The frame data is retrieved using the didOutputSampleBuffer: from Connection method, which shows the frames in the current context of the view controller using AVCaptureVideoDateOutput. The results of the time consuming estimation of the algorithm were run on the iPhone 7 for video with different sizes of the input frame (fig. 3).

Fig. 2 - Appearance of the mobile app:
1 – the image viewing area; 2 – the mark of successful fixation of bank card;
3 – the data output area

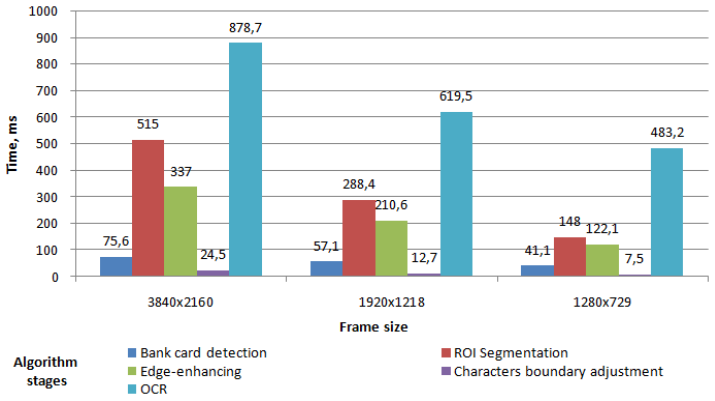

Fig. 3 - Time consuming bar graph

Conclusion

In this article, the task of recognizing information fields of bank cards is solved. The developed algorithm represents the processing of the sequence of frames received from the mobile device's camera and includes: bank card detection using the Viola-Jones high-speed object detection algorithm and the OverFeat method; ROI segmentation; improving the quality of symbols using the method of histogram normalization and morphological

transformations TopHat; adjusting the boundaries of symbols using the binarization with the adaptive threshold, performing morphological transformations on the results and searching for the most suitable region using the vertically sliding window. Recognition of adjusted regions is implemented by the Tesseract library.

REFERENCES

1 Robust Real-time Object Detection Using a Boosted Cascade of Simple Features : Proceedings of the Computer Vision and Pattern Recognition Conference, 2001 / Viola, P. and Jones, M.J.
2 OverFeat: Integrated Recognition, Localization and Detection Using Convolutional Networks, December, 2013 / Sermanet, Pierre, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun.
3 Kustikova, V.D. Development of multimedia applications using OpenCV and IPP libraries : textbook / V.D. Kustikova. – Nizhny Novgorod : Nizhny Novgorod State University. N.I. Lobachevskogo, 2012.
4 Gonzalez, R. Digital image processing / R. Gonzalez, R. Woods. – M. : Technosphere, 2005. – 1072 p.
5 OpenCV documentation [Electronic resource] // Miscellaneous Image Transformations. Adaptive Threshold. – Access mode: https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html, free. – Access date: 30.11.2017.
6 OpenCV documentation [Electronic resource] // Image Filtering. getGaussianKernel. – Access mode: https://docs.opencv.org/2.4/modules /imgproc/doc/filtering.html, free. – Access date: 30.11.2017.