

УДК 004.932

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ РАСПОЗНАВАНИЯ РЕКВИЗИТОВ БАНКОВСКИХ КАРТ ПОД УПРАВЛЕНИЕМ iOS

А.Ю. КУРИЛОВИЧ

(Представлено: канд. техн. наук, доц. Р.П. БОГУШ)

Представлено разработанное мобильное iOS-приложение для распознавания реквизитов банковских карт, которое предполагает предварительную обработку входных изображений для улучшения их качества и выделения блоков символов, распознавание которых выполняется с помощью библиотеки Tesseract. При разработке использовался язык программирования Objective-C, библиотека OpenCV 2.4.13, фреймворк iPhone SDK.

Для мобильного iOS-приложения распознавания реквизитов банковских карт используется алгоритм обработки видеоизображений, который состоит из этапов: детектирование области карты; сегментация изображения банковской карты; преобразование цветных сегментов к оттенкам серого; улучшение контрастности; подчеркивание границ символов с использованием операций математической морфологии; уточнение границ сгруппированных блоков символов; распознавание блоков символов библиотекой Tesseract.

В качестве основного языка для написания программной реализации взят за основу Objective-C, с использованием библиотеки OpenCV 2.4.13 (данная версия является самой актуальной для iOS), из основных фреймворков iPhone SDK, были использованы такие как: CoreMedia и AVFoundation - менеджмент медиа-данных [1, 2]; UIKit (UI) - работа с интерфейсами приложений; CoreGraphics (CG) - низкоуровневая, легковесная обработка 2D изображений на базе движка Quartz [3].

Интерфейс мобильного приложения (рисунок 1, а-в), снабжен областью просмотра изображения (1), захватываемого камерой устройства в реальном времени; областью вывода данных (3); меткой успешной фиксации на объекте распознавания (2). Область захвата имеет пропорцию равную 4:3, что является стандартом для вертикально ориентированного снимка iPhone/iPad. Область вывода содержит три вертикально расположенных текстовых поля, в которые последовательно выводится распознанная алгоритмом информация (номер банковской карты, дата срока ее истечения, фамилия и имя владельца карты). Метка детектирования карты, выполнена в виде рамки ярко-зеленого цвета с фиксированной толщиной границ, и отображается на экране в момент успешного определения положения карты, повторяя ее границы.

Захват видеопотока с основной камеры устройства реализован в отдельном потоке с помощью фреймворка AVFoundation и объекта ввода AVCaptureDeviceInput, который инициализируется объектом устройства захвата AVCaptureDevice в режиме видео съемки. Создается новая сессия AVCaptureSession, с задающимися через параметр размерами кадра. Далее при помощи метода addInput к ней добавляется ранее определенный объект ввода и через вызов startRunning - сессия запускается. Получение данных кадра осуществляется посредством метода обратного вызова didOutputSampleBuffer:fromConnectio с отображением фреймов на текущем контексте контроллера представления с помощью AVCaptureVideoDataOutput.

Согласно алгоритму работы, в первую очередь определяется ориентация кадра. После получения объекта изображения типа UIImage из данных потока, используя мета-данные от камеры устройства получим информацию о его ориентации типа UIDeviceOrientaton.

Используя объект CIDetector, и ряд опций: CIDetectorTypeRectangle (для обнаружения прямоугольных областей), CIDetectorAccuracyHigh (высокую точность распознавания), получим на выходе массив объектов типа CIRectangleFeature, с размерами и позициями детектированных на изображении прямоугольников. Вычисляя последовательно соотношение сторон каждого, выбираем только те, которые с наибольшей точностью подходят к эталонным размерам карты. Параллельным шагом становится поворот изображения кадра на угол, соответствующий вертикальному его отображению.

Отделить изображение карты по найденным выше параметрам типа CIRectangleFeature, из вертикально ориентированного снимка можно с помощью функции UIImageCreateImageInRect(), с первым параметром изображения-донора и вторым - области которую необходимо вырезать. Передавая в качестве донора - изображение банковской карты, а в качестве второго параметра последовательно вычисленные области для блоков номера карты, блока даты истечения срока действия и блока имени владельца карты получим изображения сегментов представляющих собой области интереса текущего алгоритма в формате указателей на объекты UIImage. Каждый из полученных сегментов поступает на обработку в качестве аргумента для следующего метода processBlock:cardBlockType.

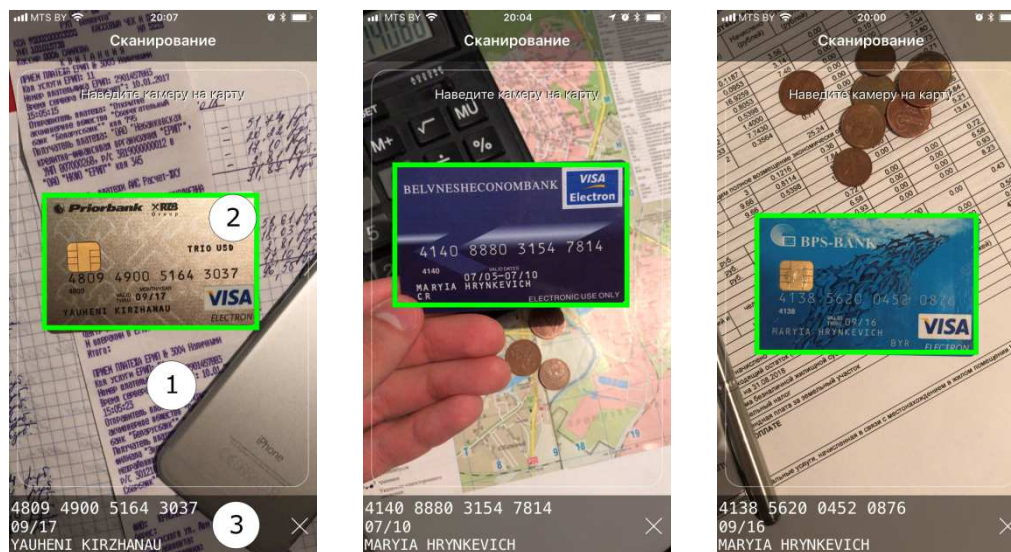


Рисунок 1. – Снимки экранов внешнего вида мобильного приложения, реализующего алгоритм распознавания реквизитов:

1 – область просмотра изображения; 2 – индикация успешного захвата карты; 3 – область вывода данных

Для поступившего на обработку изображения вычисляется его представление в формате массива OpenCV, типа Mat. Далее с помощью метода `cvtColor()` и параметра `CV_BGR2GRAY` выполняется его конвертация из цветного в оттенки серого.

Процедура повышения контрастности изображения методом нормализации гистограмм, выполняется в два этапа. Сперва вычисляется стандартная гистограмма изображения путем подсчета количества пикселей с заданными уровнем яркости в промежутке от 0 до 255, определяется средняя яркость, минимальные и максимальные значения яркости. Затем, принимая во внимание параметр погрешности ϵ , а также минимальные и максимальные уровни яркости, полученные на первом этапе, циклично с обеих сторон гистограммы выполняется подсчет количества пикселей начиная с их первоначальных экстремальных значений, когда это количество превосходит заданный в рамках погрешности порог, цикл прерывается и выбирается значения равное одному уровню яркости меньше чем текущий обрабатываемый. На выходе получаем два скорректированных значения минимальной и максимальной яркости с помощью которых вычисляем первый коэффициент контрастирования α и второй β . Далее с помощью функции OpenCV `convertTo(OutputArray, type, alpha, beta)` получаем изображение с нормализованной гистограммой [4].

Оценивая среднее значение яркости на изображении принимается решение о цвете символов на изображении, в случае если символы белые выполняется морфологическое преобразование `White Top Hat`, с помощью вызова функции `morphologyEx(inputArr, outputArr, operation, kernel)`, где параметр выбранной операции `MORPH_TOPHAT`; для темного цвета символов выполняется та же функция, но с параметром `MORPH_BLACKHAT` которая означает применение морфологической операции `Black Top Hat`. Ядро фильтра задается функцией `getStructuringElement()`, с указанием прямоугольной формы ядра `MORPH_RECT` и размеров через функцию `Size()`.

Адаптивная бинаризация выполняется путем выполнения функции OpenCV, `adaptiveThreshold()` в качестве параметров которой, передаются указатели на входной и выходной массивы данных, значение максимальной яркости для выходного изображения (всегда 255), параметр типа бинаризации `CV_ADAPTIVE_GAUSSIAN_C`, что означает применение алгоритма вычисления порога бинаризации по корреляции локально рассматриваемого блока данных с окном Гаусса (Гауссианом) [5]; `CV_TRESH_BINARY` указывает на прямой способ указания выходных значений (без инверсии). Через параметр функции `blockSize` задается значение размера рассматриваемого для определения порога бинаризации блока данных, а значение константы C носит характер смещения, по умолчанию равно 0 и может быть вычтено в процессе вычисления порога из весовой суммы [5].

Снова применяя функцию `morphologyEx(inputArr, outputArr, operation, kernel)` с параметром операции морфологии `MORPH_CLOSE`, а затем с параметром `MORPH_ERODE` получим последовательно сначала изображение к которому было применено морфологическое преобразование замыкания, а затем

операция эрозии. В обоих случаях в качестве ядра выступает матрица типа `Mat`, полученная от функции `getStructuringElement()`, с указанием формы эллипса `MORPH_ELLIPSE` и размера заданного через `Size()`.

В методе `calculateVerticalProjection:cardBlockType`: создается область окна с размером ширины изображения и с высотой равной высоте символа. Для каждого блока высота символов различна. Далее, смещая в цикле позицию окна на позицию вниз, с помощью функции `cv::countNonZero()` высчитывается количество фоновых пикселей попавших в область окна. Та позиция окна, при которой количество фоновых пикселей в ней минимально и считается позицией расположения символов в рассматриваемом блоке. Метод возвращает объект типа `CGRect` прямоугольной области занимаемой символами.

Далее выполняется процедура распознавания символов в указанной области изображения. Создается объект `G8Tesseract` с указанием в качестве инициализатора дескриптора языка для распознавания. Для случая распознавания блока номера карты, и блока даты истечения ее срока действия задается только английский язык ("`eng`"), для блока имени держателя - русский и английский ("`rus+eng`").

Экземпляр объекта `G8Tesseract` на следующем шаге получает по два варианта списков для каждого сеанса распознавания. Первый список `White List` содержит в себе символы с большим доверием, которые в процессе распознавания будут иметь больший вес для принятия позитивного решения в их сторону, и задается через параметр `charWhiteList`. Второй список `Black List`, противоположность первого варианта, содержащая набор символов с наименьшей степенью доверия. Задается этот список через параметр `charBlackList`.

При распознавании блока символов в области номера карты в качестве `White List` выступает набор содержащий только цифры: от 0 до 9. При распознавании даты срока истечения - цифры от 0-9 и символ косой черты `"/"`. В список недоверенных символов для поля держателя карты входят знаки препинания (пунктуации), а также спецсимволы, например, такие как знак равно `"="` и фигурные скобки `"{"`.

В параметр `gest` записывается экземпляр полученного на предыдущих этапах уточненного значения позиции и размера блоков символов. Указывается максимальное время на распознавание одного фрагмента, `maximumRecognitionTime`. Объекту `Tesseract` присваивается делегат на `self`, для получения результирующих данных через функцию обратного вызова, и выполняется метод `recognize`, дающий сигнал начать распознавание.

После завершения процесса распознавания системой `Tesseract`, данные результата можно получить вызвав метод `recognizedText`, экземпляра `G8Tesseract`.

После того как все блоки будут распознаны, текстовый результат каждого из них поступает в основной цикл алгоритма, там в свою очередь принимается решение о его выводе на экран с помощью объектов `NSRegularExpression` и `NSTextCheckingResult`, а также predefinedных регулярных выражений для каждого из распознаваемых блоков, принимая во внимания конкретные особенности каждого из них.

ЛИТЕРАТУРА

- 1 Developer Apple Docs [Электронный ресурс] / Core Media Framework. – Режим доступа: <https://developer.apple.com/documentation/coremedia>, свободный. – Дата доступа: 15.11.2017.
- 2 Developer Apple Docs [Электронный ресурс] / AV Foundation Framework – Режим доступа: <https://developer.apple.com/av-foundation/>, свободный. – Дата доступа: 18.11.2017.
- 3 Developer Apple Docs [Электронный ресурс] / Core Graphics Framework. – Режим доступа: <https://developer.apple.com/documentation/coregraphics/>, свободный. – Дата доступа: 22.11.2017.
- 4 OpenCV documentation [Электронный ресурс] / Miscellaneous Image Transformations. AdaptiveThreshold. – Режим доступа: https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html, свободный. – Дата доступа: 30.11.2017.
- 5 OpenCV documentation [Электронный ресурс] / Image Filtering. getGaussianKernel. – Режим доступа: <https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html>, свободный. – Дата доступа: 30.11.2017.