УДК 004.3
DOI https://doi.org/10.32782/2663-5941/2024.5.1/33

Новак Д.С.
Київський національний університет технологій та дизайну

Олещенко Л.М.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Василенко В.М.
Національний авіаційний університет

Гуйда О.Г.
Таврійський національний університет імені В.І. Вернадського

Омецинська Н.В.
Таврійський національний університет імені В.І. Вернадського

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ СИМВОЛІВ КРЕДИТНИХ КАРТОК МОВОЮ PYTHON

Робота присвячена розробці програмного забезпечення для автоматизованого розпізнавання символів на кредитних картках за допомогою мови програмування Python. Актуальність теми обумовлена стрімким зростанням обсягів електронних платежів, що потребує вдосконалення технологій обробки фінансової інформації та підвищення рівня безпеки під час проведення транзакцій. У сучасних умовах, коли швидкість і точність обробки даних відіграють вирішальну роль, автоматизація процесу зчитування інформації з кредитних карток є надзвичайно важливою.

Основною метою роботи є створення ефективного та надійного інструменту для автоматичного розпізнавання даних з кредитних карток, який може бути використаний у фінансових установах та платіжних системах. Для досягнення цієї мети застосовуються сучасні методи комп'ютерного зору, що дозволяють забезпечити високу точність та швидкість обробки зображень. У ході розробки програмного забезпечення були проведені такі етапи:

1. Аналіз існуючих рішень та визначення вимог до програмного забезпечення: проведено дослідження поточних методів та технологій розпізнавання символів, що використовуються в різних галузях, зокрема у фінансовому секторі.

2. Вибір оптимальних алгоритмів та інструментів: обрані та адаптовані алгоритми комп'ютерного зору та обробки зображень, зокрема, технології OCR (Optical Character Recognition).

3. Розробка архітектури програми: створено архітектуру, яка включає модулі для попередньої обробки зображень, сегментації та розпізнавання символів. Це забезпечило гнучкість та масштабованість програмного забезпечення.

4. Реалізація програмного забезпечення: здійснено розробку з використанням бібліотек OpenCV, imutils для обробки зображень, NumPy для аналізу даних, PyQt5 для створення графічного інтерфейсу користувача та FPDF для генерації PDF-документів з результатами розпізнавання.

5. Тестування та оцінка ефективності: проведено тестування програмного забезпечення на реальних даних, що дозволило оцінити його точність, надійність та швидкість роботи.

6. Результатом роботи є програмне забезпечення, здатне розпізнавати символи на кредитних картках різних типів та форматів. Це рішення може бути використане в різних сферах, зокрема у фінансових установах, платіжних системах, а також інших галузях, де необхідна швидка обробка інформації з платіжних карток.

Ключові слова: *розпізнавання символів, кредитна картка, Python, OpenCV, обробка зображень, OCR.*

Постановка проблеми. У сучасному світі, де цифрові технології розвиваються швидкими темпами, автоматизація процесів обробки фінансової інформації стає все більш актуальною. Одним із ключових папраметрів цієї автоматизації є швидке та точне зчитування даних з кредитних карток [1–3]. Однак, незважаючи на значний прогрес у галузі комп'ютерного зору, існують проблеми, які потребують вирішення, а саме:

Різноманітність форматів карток. Кредитні картки випускаються різними банками та платіжними системами, що призводить до варіацій у дизайні, розташуванні та форматі інформації на картках.

Якість вхідного зображення. Фотографії кредитних карток можуть мати різну якість, освітлення, кути зйомки, що ускладнює процес розпізнавання.

Безпека та конфіденційність. Обробка фінансової інформації вимагає високого рівня захисту даних та має відповідати стандартам безпеки.

Швидкість обробки. Для забезпечення продуктивності фінансових операцій необхідно досягти високої швидкості розпізнавання при збереженні точності.

Інтеграція з існуючими системами. Розроблене програмне забезпечення повинно легко інтегруватися з існуючими фінансовими системами.

Адаптивність до нових форматів. З появою нових типів карток та зміною дизайну існуючих, система повинна бути здатною адаптуватися без значних модифікацій.

Обмеження обчислювальних ресурсів. Необхідність забезпечити ефективну роботу системи на пристроях з різною архітектурою та операційними системами.

Розроблене програмне забезпечення повинно бути достатньо гнучким для адаптації до нових викликів у сфері обробки фінансової інформації та здатним задовольнити потреби як великих фінансових установ, так і малого бізнесу у швидкій та надійній обробці даних кредитних карток.

Постановка завдання. Метою роботи є розробка програмного забезпечення для автоматичного розпізнавання символів на кредитних картках з використанням мови програмування Python та методів комп'ютерного зору.

Виклад основного матеріалу. Шрифт OCR-A, створений спеціально для алгоритмів оптичного розпізнавання символів. В роботі використовувався нижченаведений алгоритм для комп'ютерного зору та обробки зображень:

1. Локалізувати чотири групи з чотирьох цифр на кредитній картці.
2. Виділити кожну з цих чотирьох груп з подальшою сегментацією кожної з шістнадцяти цифр окремо.
3. Розпізнати кожну з шістнадцяти цифр та кожну літеру імені власника кредитної картки за допомогою зіставлення з шаблоном шрифту.

Шрифт OCR-A був розроблений наприкінці 1960-х років таким чином, щоб тогочасні алгоритми розпізнавання і люди могли легко розпізнавати символи. Шрифт підтримується організаціями зі стандартизації, зокрема ANSI та ISO. Незважаючи на те, що сучасні системи розпізнавання не потребують спеціалізованих шрифтів, таких як OCR-A, він все ще широко використовується на посвідченнях особи, виписках і кредитних картках.

Насправді, існує досить багато шрифтів, розроблених спеціально для OCR, зокрема OCR-B і MICR E-13B. Шрифт MICR E-13B використовується на паперових чеках та розшифровується, як розпізнавання символів магнітним чорнилом. Магнітні датчики, камери та сканери регулярно зчитують ваші чеки. Кожен з наведених вище шрифтів має одну спільну рису – вони призначені для легкого розпізнавання [4–6].

Розроблене програмне забезпечення зіставлення шаблонів для шрифту OCR-A, який часто зустрічається на кредитних і дебетових картках. Застосовано низку операцій обробки зображень, зокрема порогове значення, обчислення градієнтного представлення величин, морфологічні операції та виділення контурів.

```
FIRST_NUMBER = {
    "3": "American Express",
    "4": "Visa",
    "5": "MasterCard",
    "6": "Discover Card"
}
```

Рис. 1. Типи кредитних карток

```
def preprocessing_find_contours(path):
    rectkernel = cv2.getStructuringElement(cv2.MORPH_RECT, (9, 3))
    sqkernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray = imutils.resize(gray, width=300)
    tophat = cv2.morphologyEx(gray, cv2.MORPH_TOPHAT, rectkernel)
    gradX = cv2.Sobel(tophat, ddepth=cv2.CV_32F, dx=1, dy=0, ksize=-1)
    gradX = np.absolute(gradX)
    (minVal, maxVal) = (np.min(gradX), np.max(gradX))
    gradX = 255 * (gradX - minVal) / (maxVal - minVal)
    gradX = gradX.astype('uint8')
    gradX = cv2.morphologyEx(gradX, cv2.MORPH_CLOSE, rectkernel)
    thresh = cv2.threshold(gradX, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
    thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, sqkernel)
    cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    cnts = contours.sort_contours(cnts, method='left-to-right')[0]
    return cnts
```

Рис. 2. Функція для знаходження контурів

Типи кредитних карток, такі як American Express, Visa, MasterCard, Discover Card, можна визначити за першою цифрою в 16-значному номері кредитної картки (рис. 1). Ми завантажуємо еталонне OCR-A зображення, конвертуємо його у відтінки сірого, порогові значення, інвертуємо та знаходимо контури. Під час кожної з цих

операцій ми зберігаємо або перезаписуємо наше еталонне зображення, обходимо контури, виділяємо та асоціюємо області інтересу з відповідними значеннями (рис. 2).

```
def find_ROI(path):
    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    thresh = cv2.threshold(gray, 10, 255, cv2.THRESH_BINARY_INV)[1]
    cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    cnts = contours.sort_contours(cnts, method='left-to-right')[0]
    sample = {}
    for (i, c) in enumerate(cnts):
        (x, y, w, h) = cv2.boundingRect(c)
        roi = thresh[y:y + h, x:x + w]
        roi = cv2.resize(roi, (57, 88))
        sample[i] = roi
    return sample
```

Рис. 3. Функція для розпізнавання значень

Ми повинні переконатися, що кожна цифра масштабується до фіксованого розміру, щоб застосувати зіставлення з шаблоном для розпізнавання цифр та асоціюємо кожну цифру та літеру із зображенням зони інтересу (рис. 3). На цьому етапі ми закінчили розпізнавати цифри і літери з нашого еталонного зображення і пов'язували їх з відповідними значеннями та ініціалізували пари структурних елементів. Ви можете уявити собі це як невелику матрицю, яку ми переміщуємо по зображенню для виконання операцій, таких як згладжування, підвищення чіткості, виявлення меж та інших операцій з обробки зображення.

```
def for_digits(cnts,path):
    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray = imutils.resize(gray, width=300)
    locs_d = []
    for (i, c) in enumerate(cnts):
        (x, y, w, h) = cv2.boundingRect(c)
        ar = w / float(h)
        if ar > 2.5 and ar < 4.0:
            if (w > 40 and w < 55) and (h > 10 and h < 20):
                locs_d.append((x, y, w, h))
    locs_d = sorted(locs_d, key=lambda x: x[0])
    output = []
    for (i, (gX, gY, gW, gH)) in enumerate(locs_d):
        group_output = []
        group = gray[gY - 5:gY + gH + 5, gX - 5:gX + gW + 5]
        group = cv2.threshold(group, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
        grpcnts = cv2.findContours(group, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        grpcnts = imutils.grab_contours(grpcnts)
        grpcnts = contours.sort_contours(grpcnts, method='left-to-right')[0]
        for c in grpcnts:
            (x, y, w, h) = cv2.boundingRect(c)
            roi = group[y:y + h, x:x + w]
            roi = cv2.resize(roi, (57, 88))
            scores = []
            for (digit, digitROI) in digits.items():
                result = cv2.matchTemplate(roi, digitROI, cv2.TM_CCOEFF)
                (_, score, _, _) = cv2.minMaxLoc(result)
                scores.append(score)
            group_output.append(str(np.argmax(scores)))
        output.extend(group_output)
    return output
```

Рис. 4. Функція для розпізнавання цифр

Ми створили два таких елемента – прямокутний і квадратний. Прямокутний ми використовували для морфологічного виокремлення дрібних елементів та деталей із заданих зображень, виявлення світлих ділянок на темному фоні (наприклад, номерів кредитних карток), а квадратний – для операції закриття. Наступним кроком у нашій спробі виокремити цифри є обчислення градієнта Шарра для зображення в напрямку по осі абцис та для усунення прогалин, ми робимо операцію закриття. Далі ми виконуємо метод Оцу та бінарний поріг зображення за яким іде ще одна операція закриття, давайте пройдемося по контурах, фільтруючи на основі співвідношення сторін кожного з них, що дозволить нам відділити місця розташування груп цифр та літер від інших, несуттєвих ділянок кредитної карти. Значення співвідношення сторін та мінімальної ширини і висоти були знайдені експериментально на наборі вхідних зображень кредитних карток.

```
def for_alphabets(cnts,path):
    img = cv2.imread(path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray = imutils.resize(gray, width=300)
    locs_a = []
    for (i, c) in enumerate(cnts):
        (x, y, w, h) = cv2.boundingRect(c)
        if y > 145 and y < (gray.shape[0] - 8) and x < (gray.shape[1] * 5 / 8) and x > 10:
            locs_a.append((x, y, w, h))
    locs_a = sorted(locs_a, key=lambda x: x[0])
    output = ''
    for (i, (gX, gY, gW, gH)) in enumerate(locs_a):
        group = gray[gY - 5:gY + gH + 5, gX - 5:gX + gW + 5]
        group = cv2.threshold(group, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
        grpcnts = cv2.findContours(group, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        grpcnts = imutils.grab_contours(grpcnts)
        grpcnts = contours.sort_contours(grpcnts, method='left-to-right')[0]
        card_name = ''
        for c in grpcnts:
            (x, y, w, h) = cv2.boundingRect(c)
            roi = group[y:y + h, x:x + w]
            roi = cv2.resize(roi, (57, 88))
            scores = []
            for i in range(len(char)):
                result = cv2.matchTemplate(roi, char[i][1], cv2.TM_CCOEFF)
                (_, score, _, _) = cv2.minMaxLoc(result)
                scores.append(score)
            index_max_score = np.argmax(scores)
            card_name = card_name + char[index_max_score][0]
        output = output + " " + card_name
    return output
```

Рис. 5. Функція для розпізнавання літер

Використовуючи OpenCV, ми отримуємо параметри, необхідні для виділення області інтересу, що містить кожну цифру (рис. 4) та літеру (рис. 5). Для того, щоб зіставлення з шаблоном працювало з певним ступенем точності, ми змінюємо розмір області інтересу до того ж розміру, що і наші еталонні зображення шрифту OCR-A (57×88 пікселів).

OpenCV має зручну функцію, за допомогою якої ви порівнюєте два зображення: одне з яких є шаблоном, а інше – вхідним зображенням. Мета її застосування – визначити, наскільки зображення схожі. Порівнюємо еталонне зображення і область інтересу з кредитної картки та зберігаємо результат та додаємо його до списку результатів. Ми знаходимо цифру або літеру з максимальною схожістю, яка відповідає певному індексу. Цілочисельне представлення цього індексу представляє найбільш ймовірну цифру або літеру на основі порівняння з кожним шаблоном (індекси вже попередньо відсортовані).

Розроблене програмне забезпечення: зберігає типи кредитних карток у словнику; розпізнає цифри та літери з еталонного зображення; зберігає шаблони цифр і літер у словнику; локалізує чотири групи номерів (містить чотири цифри (загалом 16 цифр)), кредитних карток; виконує зіставлення шаблонів для кожної цифри та літери, порівнюючи кожну окрему область інтересів з шаблоном, зберігаючи результат для кожної спроби зіставлення; знаходить найбільш схожі цифри та літери і будує список, який містить номер, ім'я власника та термін дії кредитної картки; виводить отримані результати в розроблений графічний інтерфейс та зберігає в PDF файл.

Висновки. Розроблено програмне забезпечення, яке відповідає сучасним потребам фінансового сектору в автоматизації та підвищенні безпеки обробки даних кредитних карток. Застосування комп'ютерного зору та сучасних бібліотек дозволило створити ефективне рішення для розпізнавання символів. Охоплено всі основні етапи розробки програмного забезпечення, від аналізу вимог до тестування, що забезпечує повноту та якість кінцевого продукту. Розроблено архітектуру програми, яка включає модулі попередньої обробки, сегментації та розпізнавання, що забезпечує гнучкість та можливість подальшого вдосконалення та може стати основою для створення більш складних систем автоматизації в фінансовій сфері.

Список літератури:

1. Bychkov, O., Zubyk, L., Gololobov, D., Isaienkov, Y., Grynkevych, G. and Ivanytska, A., 2023. The System for Recognizing Useful Information of the Client's ID-Card Based on Machine Learning Technologies.

2. Sun, G. and You, F., 2020, November. Bank card number recognition system based on deep learning. In Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering (pp. 745-749).

3. Xin, Y., Shi, P. and Han, S., 2019, September. An Automatic Location and Recognition Method for Bank Card Number. In Proceedings of the 2019 International Conference on Robotics, Intelligent Control and Artificial Intelligence (pp. 728-732).

4. Ou, Z., Xiong, B., Xiao, F. and Song, M., 2020. ERCS: An efficient and robust card recognition system for camera-based image. China Communications, 17(12), pp. 247-264.

5. Maithani, M., Meher, D. and Gupta, S., 2023. Multilingual Text Recognition System. In Advances in Signal Processing, Embedded Systems and IoT: Proceedings of Seventh ICMEET-2022 (pp. 103-114). Singapore: Springer Nature Singapore.

6. Poudel, U., Regmi, A.M., Stamenkovic, Z. and Raja, S.P., 2023. Applicability of OCR Engines for Text Recognition in Vehicle Number Plates, Receipts and Handwriting. J. Circuits Syst. Comput., 32(18), pp. 2350321-1.

7. Padole, C. and Mitra, D., 2022. Information Extraction from Visiting Cards Using OCR and Post-Processing in Python. International Journal of Scientific and Technical Research in Engineering (IJSTRE), 7(05), pp. 1-7.

8. Yusman, N.S. and Ibrahim, M.M., 2021. Extracting Information from Identity Card into Electronic Form Using Image Processing Technique. INOTEK 2021, 1, pp. 135-136.

9. Gupta, M.K., Shah, R., Rathod, J. and Kumar, A., 2021, November. Smartidocr: Automatic detection and recognition of identity card number using deep networks. In 2021 Sixth International Conference on Image Information Processing (ICIIP) (Vol. 6, pp. 267-272). IEEE.

10. Manage, P., Ambe, V., Gokhale, P., Patil, V., Kulkarni, R.M. and Kalburgimath, P.R., 2020, December. An intelligent text reader based on python. In 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS) (pp. 1-5). IEEE.

Novak D.S., Oleshchenko L.M., Vasylenko V.M., Guida O.H., Ometsynska N.V. DEVELOPMENT OF SOFTWARE FOR CREDIT CARD CHARACTER RECOGNITION IN PYTHON

The work is devoted to the development of software for automated character recognition on credit cards using the Python programming language. The relevance of the topic is due to the rapid growth of electronic payments, which requires improving financial information processing technologies and increasing the level of security during transactions. In today's environment, when the speed and accuracy of data processing play a crucial role, automating the process of reading information from credit cards is extremely important.

The main goal of this work is to create an efficient and reliable tool for automatic recognition of credit card data that can be used in financial institutions and payment systems. To achieve this goal, modern computer vision methods are used to ensure high accuracy and speed of image processing. The following stages were carried out during the software development:

1. Analysis of existing solutions and determination of software requirements: a study of current methods and technologies for character recognition used in various industries, including the financial sector, was conducted.

2. Selection of optimal algorithms and tools: computer vision and image processing algorithms, in particular, OCR (Optical Character Recognition) technology, were selected and adapted.

3. Development of the program architecture: an architecture was created that includes modules for image preprocessing, segmentation, and character recognition. This ensured the flexibility and scalability of the software.

4. Software implementation: development was carried out using OpenCV libraries, imutils for image processing, NumPy for data analysis, PyQt5 for creating a graphical user interface, and FPDF for generating PDF documents with recognition results.

5. Testing and performance evaluation: the software was tested on real data, which allowed us to evaluate its accuracy, reliability and speed.

6. The result of the work is software capable of recognizing symbols on credit cards of various types and formats. This solution can be used in various fields, including financial institutions, payment systems, and other industries that require fast processing of information from payment cards.

Key words: *character recognition, credit card, Python, OpenCV, image processing, OCR.*