Rozpoznawanie tablic rejestracyjnych Aplikacja na platformę Android

Laboratorium widzenia komputerowego

Maciej Nowak i Arkadiusz Wieczorek

30 listopada 2016r.

Spis treści

1 Wstęp

Celem aplikacji jest rozpoznawanie polskich tablic rejestracyjnych pojazdów. Aplikacja dzieli proces na 3 etapy. W pierwszej kolejności poddajemy zdjęcie wstępnej obróbce, która ma zapobiec niepotrzebnemu szumowi. W drugim etapie algorytm ustala pozycję naszej tablicy, zaś w trzecim wycinamy prostokąt odpowiadający proporcjom tablicy i przekazujemy do odczytania.

2 Algorytm - krok 1

Poniżej zostały przedstawione konkrente kroki algorytmu i ich implementacje.

2.1 Gray scale

Na początku wykonujemy dwie proste metody, aby przygotować nasze zdjęcie dla pozostałych kroków. Skala szarości pozwala zredukować przewagę istotnych bardzo nasyconych kolorów na zdjęciu.

Rysunek 1: Skala szarości

```
1 private void grayScale() {
2     grayScale = OpenCV.colorizeImage(image, Imgproc.COLOR_RGB2GRAY);
3 }
```

2.2 Gauss blur

Rozmycie gaussa umożliwia do pewnego stopnia odszumienie zdjęcia z niedoskonałości. W kontekście omawianego zdjęcia samochodu może to być pobrudzona tablica.

```
1 private void gaussBlur(Size size) {
2     gaussBlur = new Mat(grayScale.rows(), grayScale.cols(),
      grayScale.type());
3     Imgproc.GaussianBlur(grayScale, gaussBlur, size, 0);
4 }
```

Rysunek 2: Rozmycie Gaussa

3 Algorytm - krok 2

3.1 Sobel

Operacja sobel umożliwia uwypuklenie dominujących krawędzi na zdjęciu, czego efekty można zobaczyć poniżej.

Rysunek 3: Sobel

```
private void sobel(int depth, int x, int y) {
    sobel = new Mat();
    Imgproc.Sobel(gaussBlur, sobel, depth, x, y);
}
```

3.2 Adaptive Threshold (OTSU + BINARY)

Po uwypukleniu krawędzi na zdjęciu, stosujemy progowanie z opowiednimi parametrami, co pozwala wyszczególnić najjaśniejsze obiekty na zdjęciu.

```
private void adaptiveThreshold(double maxValue, int method, int
    type, int size, double C) {
    adaptiveThreshold = new Mat();
```

Rysunek 4: Adaptive Threshold (OTSU + BINARY)

```
3       Imgproc.adaptiveThreshold(sobel, adaptiveThreshold, maxValue,
        method, type, size, C);
4   }
```

3.3 Morphology (MORPH CLOSE)

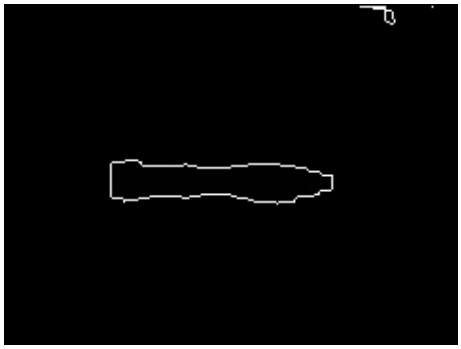
W tym przypadku operacja morfologii wykonuje coś na zasadzie zlewania w jeden obiekt. Po wykonaniu morfologii otrzymujemy jednolitą białą plamę w miejscu w którym progowanie pozostawiło wyraźnie przeważające jasne obiekty.

Rysunek 5: Morphology (MORPH CLOSE)

```
1   private void morphology() {
2       morphology = new Mat();
3       Mat element = getStructuringElement(Imgproc.MORPH_RECT, new
        Size(15, 5));
4       Imgproc.morphologyEx(adaptiveThreshold, morphology, Imgproc.
        MORPH_CLOSE, element);
5   }
```

3.4 Contours

Z białej plamy pozostawiamy kontur, który w kolejnym kroku zostaje przekazany do sprawdzenia proporcji odpowiadającej tablicy rejestracyjnej.



Rysunek 6: Contours

```
1 private void drawContours() {
2     contoursPoints = new ArrayList<>();
3     contours = morphology.clone();
4     Imgproc.findContours(contours, contoursPoints, new Mat(),
      Imgproc.RETR_LIST, Imgproc.CHAIN_APPROX_SIMPLE);
5     Imgproc.drawContours(contours, contoursPoints, -1, new Scalar
      (255,0,0));
6 }
```

3.5 Rectangle

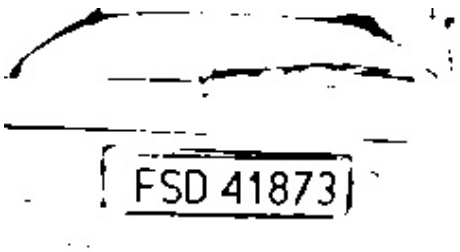
Następnie nakładamy na zdjęcie prostokąt na podstawie wyznaczonych proporcji i miejsca, w którym znajduje się tablica rejestracyjna.

Rysunek 7: Rectangle

4 Algorytm - krok 3

4.1 Adaptive Threshold (GAUSSIAN C + BINARY INV)

Znając miejsce występowania tablicy oraz jej rozmiar na zdjęciu dokonujemy ponownego progowania korzystając z rozymtego zdjęcia z kroku 1.



Rysunek 8: Adaptive Threshold (GAUSSIAN C + BINARY INV)

```
1 private void adaptiveThresholdCrop(double maxValue, int method, int
      type, int size, double C) {
2     adaptiveThresholdCrop = new Mat();
3     Imgproc.adaptiveThreshold(gaussBlur, adaptiveThresholdCrop,
      maxValue, method, type, size, C);
4 }
```

4.2 Cut

W przedostatniej operacji musimy wyciąć zdjęcie, tak aby pozostawić tylko i wyłącznie rejestrację. Nieprzycięta fotografia powodowałaby liczne szumy i zakłócenia dla OCR. Powierzchnia, którą otrzymujemy w wyniku przycięcia musi spełniać dwa warunki:

- nie może przekraczać minimalnej i maksymalnej wielkości i szerokości zdjęcia,
- stosunek białego obszaru do reszty zdjęcia musi wynosić przynajmniej 60%.

Rysunek 9: Wygenerowany plate

```
private void findPlate() {
    //PREPARE DATA
    File file = new File(OpenCV.root, fileNumber + "-plate.jpg");
    file.delete();
    ivPlate.setImageResource(0);
    rectangle = image.clone();

    //FOR EACH SET POINTS
    if (contoursPoints.size() > 0) {
        for (MatOfPoint matOfPoint : contoursPoints) {
            MatOfPoint2f points = new MatOfPoint2f(matOfPoint.
    toArray());
            RotatedRect box = Imgproc.minAreaRect(points);
            Imgproc.rectangle(rectangle, box.boundingRect().tl(),
    box.boundingRect().br(), new Scalar(255, 0, 0));

            //IF POINTS SET HAS REQUIRE RATIO AND AREA
            if(checkRatio(box)){
                plateCandidate = new Mat(adaptiveThresholdCrop, box
    .boundingRect());

                //IF POINTS SET HAS REQUIRE WHITE PIXELS RATIO
    INSIDE
                if(checkDensity(plateCandidate)){
                    Imgproc.rectangle(rectangle, box.boundingRect()
    .tl(), box.boundingRect().br(), new Scalar(0, 255, 0));
                    plate = plateCandidate.clone();
                    OpenCV.saveImage(fileNumber + "-plate.jpg",
    plate);
                    ivPlate.setImageBitmap(OpenCV.matToBitmap(plate
    ));
                } else {
                    Imgproc.rectangle(rectangle, box.boundingRect()
    .tl(), box.boundingRect().br(), new Scalar(0, 0, 255));
                }
                plateCandidate = null;
            }
        }
    }

    //SHOW RESULTS
    ivRectangle.setImageBitmap(OpenCV.matToBitmap(rectangle));
}
```

4.3 OCR

Na koniec przygotowany plate przekazujemy do OCR biblioteki Tesseract.

```
private String OCR(String fileName) {
    try {
        String text;
        TessBaseAPI tessBaseApi = new TessBaseAPI();
        tessBaseApi.init(OpenCV.root.getAbsolutePath(), "eng");
        tessBaseApi.setVariable("tessedit_char_whitelist", "
    ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789");
        tessBaseApi.setImage(new File(OpenCV.root, fileName));
```

```
8           text = tessBaseApi.getUTF8Text();
9           tessBaseApi.end();
10          return text;
11      }
12      catch (Exception e) {
13          return "";
14      }
15  }
```

5 Testy

5.1 Bezbłędne

Rysunek 10: Wynik OCR = PO 7H053

Rysunek 11: Wynik OCR = FSD 41873

5.2 Dobre

Poniżej przedstawiono sporo przykładów, w których natrafiliśmy na drobne błędy (1-2).

Rysunek 12: Wynik OCR = 7PO 7H053

Rysunek 13: Wynik OCR = 1PO 014HH

Rysunek 14: Wynik OCR = 3P0 DMHH

Rysunek 15: Wynik OCR = CTR 4KPZ

Rysunek 16: Wynik OCR = VCTR 4KP9

Rysunek 17: Wynik OCR = FSD 4187Q

Rysunek 18: Wynik OCR = JES FPY59

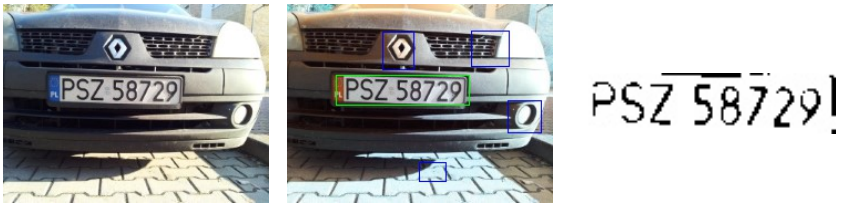
Rysunek 19: Wynik OCR = CO 3343

Rysunek 20: Wynik OCR = 2P0 984L

Rysunek 21: Wynik OCR = 63633

5.3 Słabe

Poniżej zostały przedstawione przykłady testów, w których wyniki znacząco odbiegają od stanu faktycznego jaki przedstawia tablica rejestracyjna.



Rysunek 22: Wynik OCR = PSZR7291

Rysunek 23: Wynik OCR = E140172

Rysunek 24: Wynik OCR = MEQBQ

Rysunek 25: Wynik OCR = HPO UBZCCI

Rysunek 26: Wynik OCR = 8729A

5.4 Brak odczytu

Poniżej zostały przedstawione przykłady testów, w których nie udało się znaleźć prostokąta odpowiadającego proporcjom tablicy. Poniższy przypadek prezentuje problem wielu jasnych punktów na zdjęciu.

Rysunek 27: Brak wyniku OCR z powodu dużej liczby drobnych jasnych punktów

Dwa kolejne przypadki przedstawiają typowy problem prześwietlonego zdjęcia, dla których przedstawione we wcześniejszym rozdziale zagadanienia nie działają.

Rysunek 28: Brak wyniku OCR z powodu bardzo dużego prześwietlenia

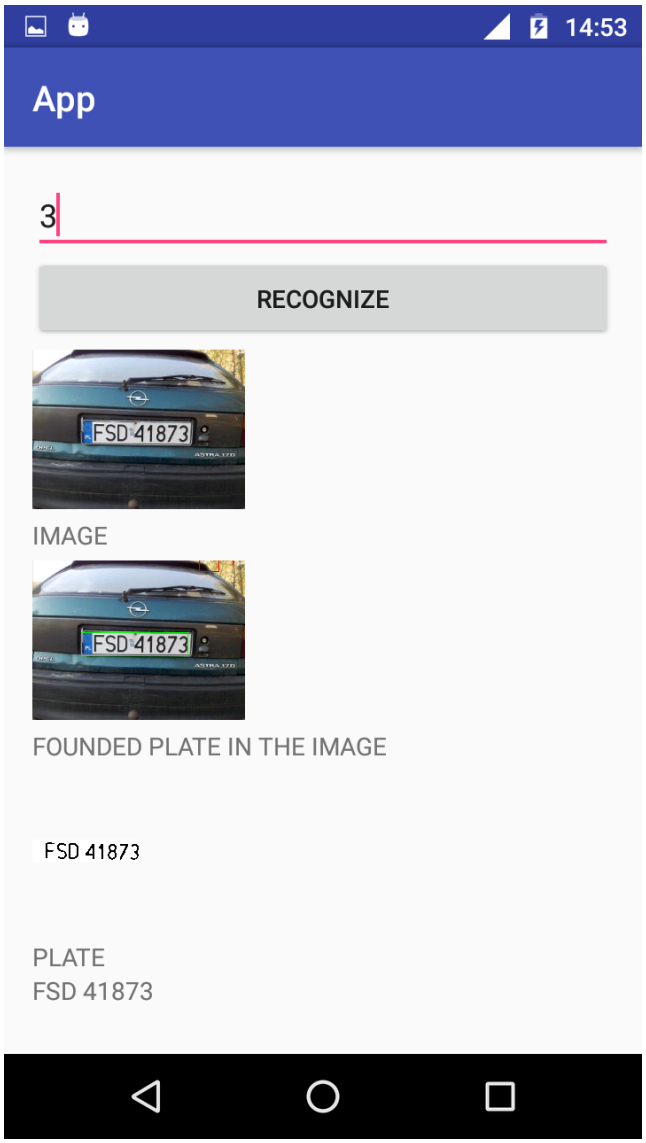
Rysunek 29: Brak wyniku OCR z powodu bardzo dużego prześwietlenia

Poniżej zaprezentowano dwa przypadki, w których problem dotyczy się cienia rzucanego na tablicę rejestracyjną. Jak możemy zauważyć, miejsce tablicy rejestracyjnej zostało odkryte, ale przez różny poziom natężenia światła nie udało się dopasować wymaganych proporcji.

Rysunek 30: Brak wyniku OCR z powodu zacienionej tablicy

Rysunek 31: Brak wyniku OCR z powodu zacienionej tablicy

6 Aplikacja



Rysunek 32: Zrzut ekranu aplikacji

7 Podsumowanie

Wykonując testy napotkaliśmy wiele problemów typowych dla widzenia komputerowego. Problemy, z którymi się zetknęliśmy to:

- charakterystyka i prześwietlenie zdjęcia testowego,
- wielkość zdjęcia testowego,
- duża liczba drobnych jasnych punktów,
- geometria zdjęcia testowego (czy zdjęcie zostało wykonane z boku czy na wprost auta),
- różne rodzaje aparatów,
- moc obliczeniowa urządzenia.

Zakładając, że charakterystyka zdjęcia byłaby podobna moglibyśmy w większości przypadkach odpowiadać na zadane zdjęcie testowe bezbłędnie. Jednak dla zadanego zbioru testowego zmiana dowolnego parametru w wybranym kroku modyfikuje wyniki dla wszystkich pozostałych. W omawianym zagadnieniu spróbowaliśmy ustawić takie parametry naszych kroków, aby wyniki były co najmniej dobre dla większości badanych zdjęć.