**Sprawozdanie z laboratorium:**

**Komunikacja Człowiek-Komputer**

**Temat: Wykrywanie tablic rejestracyjnych w obrazach**

**Data: 28.11.2017r.**

**Prowadzący: Michał Tomczyk**

**Zajęcia: środa 8:00**

**Wykonali:**

**Katarzyna Jóźwiak 127237, gr. I4  
Piotr Pawlaczyk 127245, gr. I4**

1. **Wstęp**Celem naszego projektu było napisanie programu wykrywającego tablice rejestracyjne samochodów na zdjęciach. Systemy takie używane są w życiu codziennym , m.in. do wykrywania wykroczeń (np. na fotoradarach), rejestrowania samochodów korzystających z parkingów.
2. **Algorytm**
3. Na początku każdy z przetwarzanych przez nas obrazów poddaliśmy wyrównaniu histogramu na kolorowym obrazie



1. Następnie obraz z wyrównanym histogramem poddaliśmy działaniu filtru medianowego z oknem o rozmiarze 5 x 5



1. Po użyciu filtru medianowego zdecydowaliśmy się użyć na poprzednio uzyskanym wyniku filtru Gaussa z sigma 1,5



Rysunek Zdjęcie poddane obróbce filtrem medianowym

1. Na tak wstępnie przygotowanym obrazie zajęliśmy się detekcją krawędzi. Na przetworzonym obrazie zastosowaliśmy algorytm detekcji Canny’ego z niskim progiem = 100, a wysokim 150



Rysunek Wynik zastosowania algorytmu Canny'ego

1. Po znalezieniu krawędzi w naszym obrazie , zajęliśmy się wyszukiwaniem obszarów występowania naszych potencjalnych tablic przy użyciu funkcji findContours z biblioteki OpenCV, która znalazła sporą ich ilość. Następnie znalezione kontury poddaliśmy działaniu funkcji boxPoints, która ograniczyła je do prostokątów.

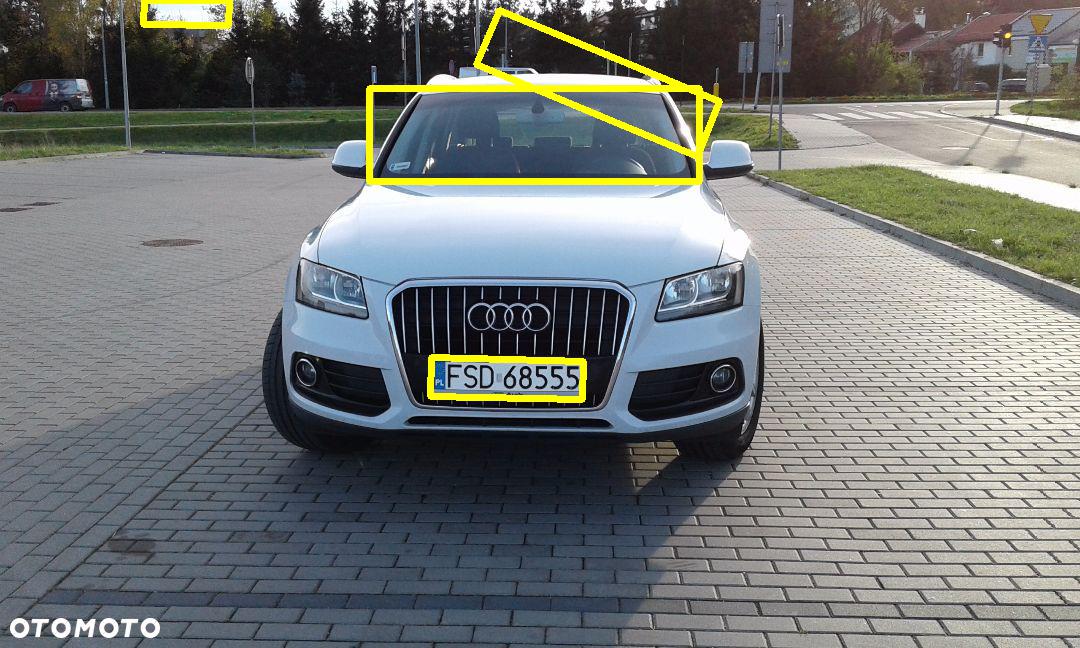
Jak widać na załączonym zdjęciu, ta liczba nadal była za duża.



Rysunek Zdjęcie z nałożonymi prostokątami zaznaczające potencjalne tablice rejestracyjne

1. Kolejnym etapem prowadzonych przez nas działań było ograniczenie prostokątów do tych, które w przybliżeniu mają rozmiar tablic rejestracyjnych w Polsce. Rozmiar tych tablic w Polsce do 520 x 114 mm, więc zdecydowaliśmy się brać pod uwagę te prostokąty, które mają stosunek szerokości do wysokości pomiędzy 3 a 6.5 (uwzględniliśmy, że zdjęcie może być wykonywane pod pewnym kątem).

Zaimplementowaliśmy też funkcję , która sprawdzała czy znalezionych przez nas prostokąt nie jest położony całkowicie pionowo na obrazie (większa wysokość niż długość).



Rysunek Zdjęcie po filtracji prostokątów

1. Niestety nawet takie ograniczenie nie przyniosło oczekiwanego przez nas efektu, ponieważ rysowały się także inne prostokąty nie będące tablicą rejestracyjną.

Postanowiliśmy więc, dla zostawionych obszarów, w obrębie każdego obrazka policzyć wartość unikalnych dla nich współczynników i stworzyć ich tablicę.

Zaimplementowaliśmy więc funkcję tworzącą wektor wartości współczynników dla danego pozostałych konturów obrazka bazującą na funkcji policzSumeKonturu(), która dla każdej z potencjalnych tablic najpierw wczytywała obraz z nałożonym filtrem Canny (punkt d) i dla konkretnych konturów obliczała ilość białych pikseli (tych zwróconych przez funkcję znajdywania konturów Cannyego) oraz zliczała liczbę pikseli zajmowanych przez jeden kontur (prostokąt). Współczynnikiem obrazka był iloraz sumyBiałychPikseli i rozmiaru konturu. To przyniosło nam najlepsze rezultaty.



Rysunek Ostateczny wynik działania programu

1. **Testowanie**

Testowanie naszego programu oparliśmy na samochodach znalezionych przez nas na popularnych serwisach aukcyjnych ( głównie otomoto.pl ). Przeglądając oferty samochodów w Polski wybraliśmy 50 pojazdów. Spośród nich nasz algorytm poprawnie zidentyfikował tablicę rejestracyjne w 46 pojazdach, co stanowi skuteczność działania naszego programu na poziomie ok. 90%.

1. **Przykłady poprawnie uzyskanych wyników:**



Rysunek Przykłady działania algorytmu

1. **Przykłady błędnych wyników podczas testowanie różnych wariantów filtracji konturów:**

Nie jest łatwo dostosować algorytm dla różnych rozmiarów tablic, naświetlenia, stopnia zabrudzenia tablic czy kąta pod jakim zostało wykonane zdjęcia. Początkowo wartość współczynnika dla finalnych konturów liczyliśmy poprzez różnicę wartości V (z przestrzeni barw HSV) sąsiadujących pikseli w obrębie każdego konturu i dzielenia jej przez liczbę pikseli w jego obrębie. Niestety nie przynosiło to zamierzonych efektów, ponieważ niektóre zaznaczone „tablice” znajdowały się np. na dachu domu lub na drugim planie zdjęcia. Podczas ewolucji naszego algorytmu doszliśmy do wniosku, że tablice nie będące w 100% prostokątami mają zawyżony obszar, ponieważ liczyliśmy je tak jakby były prostokątami, przez co w naszych obliczeniach uwzględnialiśmy także duży obszar nie będący tablicą i musieliśmy poprawić funkcję odpowiadającą za liczenie liczby pikseli w konturze.





Rysunek Przykłady błędnego działania programu podczas testów

Przed zastosowaniem funkcji z poprawnym zaznaczaniem obszarów, działało nam tylko około 25% przypadków. Następnie po jej poprawieniu uzyskaliśmy rezultat blisko dwukrotnie lepszy. Użycie współczynnika konturu jako sumy różnic wartości składowej V sąsiadujących pikseli w obszarze podniosło skuteczność naszego algorytmu do ok. 60%. Dopiero dobranie prawidłowych proporcji występowania konturów kształtów liter i cyfr w obrębie obszaru pozwoliło uzyskać rezultaty na poziomie około 90% dla testowanych przez nas przypadków.

1. **Źródła**  
   http://docs.opencv.org/2.4/

http://scikit-image.org/docs/dev/api/skimage  
http://opencv-python-tutroals.readt

hhttp://www.cs.put.poznan.pl/mtomczyk/