**Sprawozdanie z laboratorium:**

**Komunikacja człowiek - komputer**

Sprawozdanie I: Przetwarzanie obrazu – aplikacja

14 listopada 2017

**Prowadzący:** mgr inż. Paweł Liskowski

**Autorzy:** Mateusz Urbaniak 127345

Kajetan Zimniak 127229

Zajęcia wtorkowe, 8:00.

1. **Wstęp**

Celem projektu było znalezienie i zaimplementowanie najlepszej metody wykrywania i rozpoznawania cyfr drukowanych ze zdjęcia. Wśród wielu możliwych dróg stworzenia programu, na bieżąco wybieraliśmy najlepsze, naszym zdaniem, funkcje i algorytmy, aby osiągnąć pożądany efekt. Podział prac wynikał z kolejnych etapów projektu: implementacja algorytmu szukającego krawędzi obiektów, segmentacja obrazu, dopasowanie do wzorca, testowanie i modyfikacja programu.

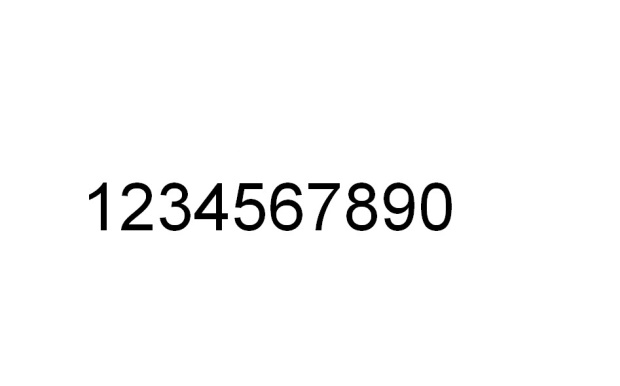
1. **Opis algorytmu i wykorzystanych filtrów, funkcji i metod**

W pierwszej kolejności za pomocą funkcji *io.ImageCollection()* ładujemy obrazy do programu, które następnie, w kolejności, poddawane są przekształceniu na odcienie szarości (*color.rgb2grey()*), w celu uzyskania dwuwymiarowej tablicy, na której dalej pracujemy. *Np.percentile()* i *exposure.rescale\_intensity()* służą do zmiany intensywności. W tym momencie jesteśmy gotowi do użycia *measure.find\_contours()* w celu znalezienia konturów, a dokładniej ich współrzędnych. Nie wystarczy to jednak do zakończenia pierwszej części programu. Na przykład dla cyfry „6”, funkcja wykryje nam dwa kontury, ponieważ znak składa się z dwóch kształtów. Przy użyciu minimalnych i maksymalnych współrzędnych x, y, usuwamy wewnętrzne części znaków, które nie pomogą nam w dalszej detekcji. W tym momencie jesteśmy gotowi do segmentacji, polegającej na wycięciu odpowiednich części macierzy obrazu. Jest to operacja nieskomplikowana, jednak potrzebna do użycia funkcji *match\_template().* Zwraca ona liczbę przedstawiającą dokładność pokrycia, znalezienia szukanego elementu na obrazie. W naszym przypadku konieczne było początkowe wycięcie każdego znaku, aby zachować kolejność odczytywanych cyfr, ponieważ każdy znak sprawdzany jest w pętli, z każdym przygotowanym szablonem cyfr od 0 do 9.

1. **Eksperymenty**

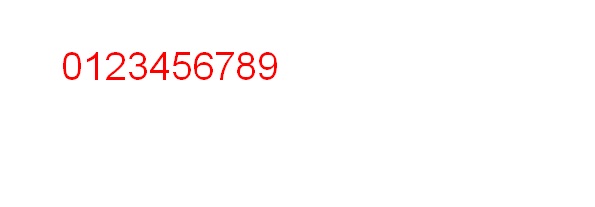
Program przystosowany jest do rozpoznania każdego rodzaju czcionki. W eksperymentach skupiliśmy się na czcionce Arial, po to, by dokładniej zbadać wpływ rozmiaru, koloru i tła zdjęcia.

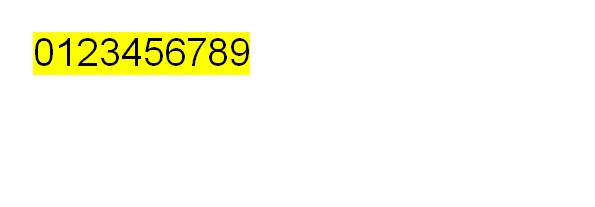
Na początku pracowaliśmy nad tym, aby program ze stu-procentową skutecznością rozpoznawał cyfry ze zrzutu ekranu *(rys.1)*, na białym tle. Problemem okazały się różnice w rozmiarze zdjęcia i szablonu. Łatwo je jednak wyeliminowaliśmy skalując obrazy. Następnie użyliśmy dylatacji i erozji, w celu uzyskania bardziej dokładnych „okrągłych” konturów zdjęcia, których jakość podczas wcześniejszych operacji, dążących do wycięcia pojedynczych znaków, zmiany rozmiaru itp., się pogorszyła.

*Rysunek 1:*

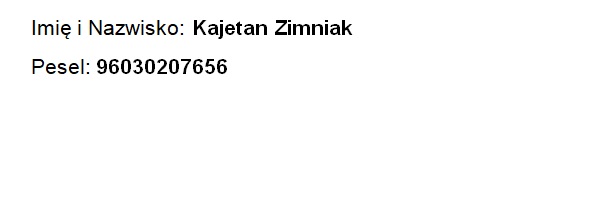
Po osiągnięciu sukcesu z powyżej przytoczoną sytuacją, zajęliśmy się badaniem podobnych przykładów, różniących się między sobą kolorem czcionki, wielkością i kolorem tła.

*Rysunek 2:*



  
We wszystkich powyższych przykładach program poradził sobie bezbłędnie. Wykonaliśmy więc próbę dla przykładu składającego się z liter i cyfr. Postanowiliśmy w tym miejscu dodać kolejną funkcjonalność, zaznaczanie na obrazie, w którym miejscu znajduje się cyfra, którą rozpoznajemy.

*Rysunek 3, obraz wejściowy:*



**Źródło**

[1] http://scikit-image.org

[2] http://matplotlib.org