Autorzy
Szymon Krysztopolski 144619 i4.1
Michał Całka 141026 i4.1

# *Wstęp*

Celem projektu było zaimplementowanie rozpoznawania liczb pisanych odręcznie na białej kartce. Podzieliliśmy zadanie na dwa etapy. Pierwszym było skonstruowanie funkcji która dekoduje pojedynczą cyfrę. Drugi etap polegał na stworzeniu funkcji która podzieli napisaną liczbę (składającą się w domyśle z wielu cyfr) na pojedyncze cyfry, a następnie wykonuje wcześniej przygotowaną funkcję na każdej z nich dekodując tym samym liczbę.

# *Użyte metody*

Funkcja rozpoznająca cyfry działa na dość niskim poziomie. Analiza cyfr polega na rozpoznawaniu charakterystycznych cech dla danej cyfry. Cechy które są brane pod uwagę to położenie oraz ilość okręgów oraz prostych na danej cyfrze. Funkcje które są to wykorzystywane to HoughLines() oraz HoughCircles(). Paradoksalnie te dwie proste funkcje dają bardzo dużo użytecznych informacji. Przykładowo za ich pomocą zliczamy ilość prostych które są poziome, pionowe czy po skosie. Na samym początku funkcji używamy funkcji erosion() oraz dilation(). Ułatwia to programowi analizę cyfr.

W przypadku dzielenia liczby na cyfry stosujemy dodatkowo threshold(), dzięki czemu powstaje wyraźne oddzielenie cyfr od tła. Potencjalne cyfry są wyznaczane poprzez funkcję findContours(), a kontury spełniające kryteria – nie będące za małe ani nie znajdujące się wewnątrz innych konturów – są następnie kopiowane i zapisywane jako oddzielne pliki przy użyciu funkcji boundingRect().

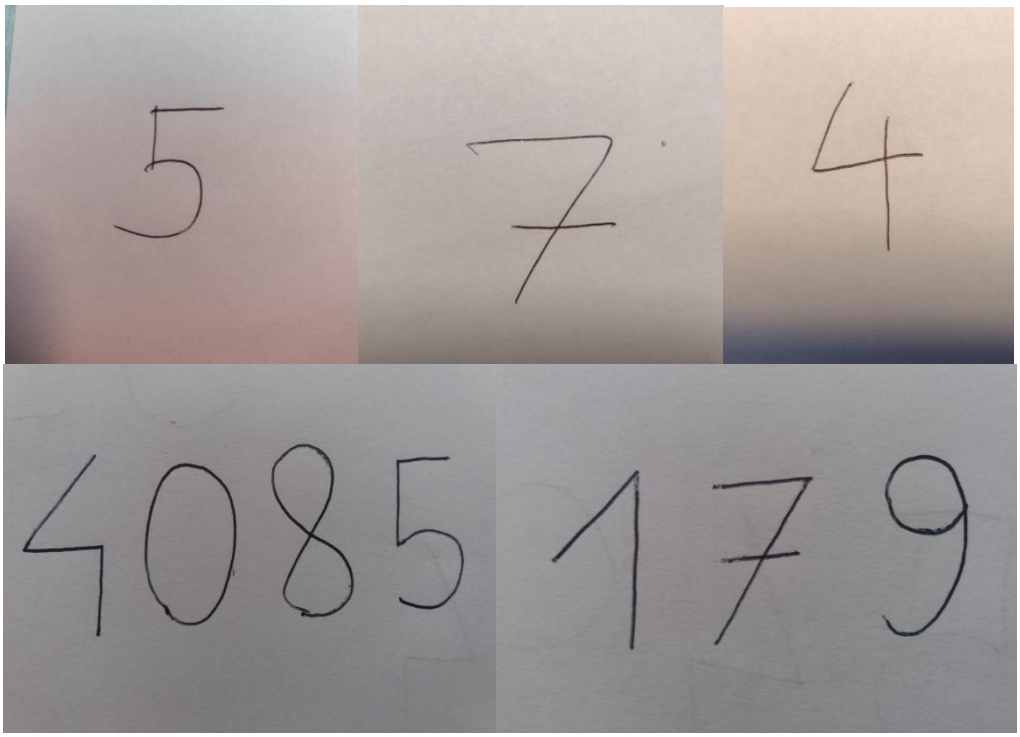
# *Badania, testy i wyniki*

Testy dzielą się na dwie części. W pierwszej badamy pojedyncze cyfry, aby zbadać skuteczność funkcji za odpowiedzialnej za rozpoznawanie ich. W dalszej części testujemy całe liczby. Ostatecznie ilość badanych cyfr jest równa 98. W momencie pisania sprawozdania poprawnych wyników jest 80.

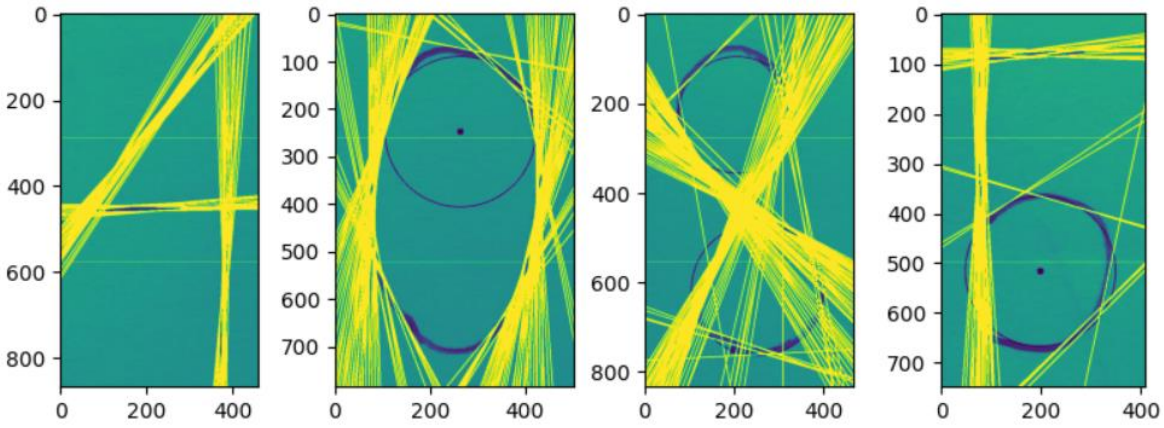
Ta liczba może ulec poprawie z uwagi na funkcjonalność, która jest zaimplementowana w naszym programie. Mianowicie program potrafi się samodzielnie uczyć. Z uwagi na to że mamy 21 argumentów ciężko byłoby to ręcznie testować. Zamiast tego stworzyliśmy automatyczne testy. Program losuje permutacje liczb (od 0 do N, gdzie N jest liczbą argumentów). Jest to kolejność testowania poszczególnych argumentów. Następnie program sprawdza wszystkie możliwości zmiany danego argumentu (wcześniej są ustalone parametry w postaci MIN, MAX oraz STEP). Gdy program znajdzie rozwiązanie niepogarszające, to przechodzi do niego. Dzięki takiemu podejściu i małym zmianom nakładającym się na siebie, możliwe jest uzyskanie lepszego wyniku. Program ma zapisane w pliku tekstowym obecnie najlepsze argumenty. W momencie znalezienia lepszego

rozwiązania plik jest nadpisywany nowymi wartościami. W ten sposób nie trzeba ręcznie modyfikować żadnych wartości.

Do testów zostały wykorzystane własnoręcznie wykonane zdjęcia, zawierające liczby napisane długopisem na białej kartce.



Rysunek 1: Przykładowe zdjęcia wykorzystane w programie




Rysunek 2: Wynik działania programu dla liczby 4085

# *Wnioski i możliwe kierunki rozwoju*

Program na ten moment działa poprawnie. Nie jest on jednak perfekcyjny. Żeby tak się stało należałoby zmodyfikować logikę funkcji rozpoznającej cyfry oraz możliwe że uwzględnić więcej cech charakterystycznych. Ulepszyć też można samą metodę uczenia się programu stosując jakiś rodzaj tablicy tabu, lub losując startowy zestaw argumentów. Celem takiej modyfikacji jest pozwolenie programowi na wyjście z lokalnego maksimum.