Analiza i Przetwarzanie obrazów
Sudoku

Sebastian Kałużny

3 lipca 2017

1 Wprowadzenie i metoda

Celem projektu było stworzenie programu, który będzie na podstawie zdjęcia z planszą sudoku rozwiązywał ją i wpisywał na orginalne zdjęcie rozwiązanie.

Program możemy podzielić na poszczególne części :

1. Załadowanie głównego obrazu
2. Znalezienie i obrysowanie konturu głównej planszy
   (a) przeprowadzamy wstępne operacje morfologiczne
   (b) przy pomocy wbudowanej funkcji cv::findContours wyszukujemy konturów obrazu
   (c) szukamy obszaru o największej powierzchni na podstawie konturów
   (d) określamy obszar planszy na której będziemy dalej przeprowadzać operacje („wycinamy" go z obrazka)
3. Przygotowujemy obraz do OCR - pozbywamy się lini planszy sudoku
   (a) na „wycietej" planszy z głównego obraza przeprowadzamy opercaje binaryzacji
   (b) przy pomocy funckji cv::HoughLinesP znajdujemy linie obrazu
   (c) ostnim krokiem jest usuniecie lini z obrazu
4. Wywołanie algorytmu OCR na obrazie z cyframi
   (a) w pierwszym etapie na podstawie klasy cv:::ml::kNearest i dostarczonych plików (classifications, image) trenujemy nasz algorytm
   (b) kolejno przeprowadzamy operacje morfologiczne przygotowując obraz

(c) wyszukujemy kontury znaków na obrazie i na podstawie wczesniej przetrenowanego algorytmu określamy znak (kNearest->findNearest)

5. Znalezienie poszczególnych kwadratów planszy sudoku 9x9
   (a) dostarczona klasa służy do określenia podzialu planszy sudoku na małe kwadraty (9x9 plansza)
   (b) dostarczamy jej także kontury znalezionych cyfr z OCR , klasa ta określa w którym polu znajduje się dana cyfra
   (c) zwaraca tablice dwuwymiarową przygotowną do wywołania algorytmu rozwiązania sudoku
6. Rozwiązanie sudoku metodą brute force
7. Określenie miejsca liczb na głównym zdjęciu i wpisanie rozwiązania sudoku

Porównując końcowe działanie programu a wstępne założenia umieszczone na stronie które wyglądały następująco:

1. Operacje morfologiczne. Przygotowanie obrazu.
2. Transformata Hougha, aby znaleźć linie -> wyznaczyć pole sudoku
3. OCR -> znalezienie liczby
4. Na podstawie wcześniejszych korków, wyznaczanie pól liczby
5. Rozwiązanie sudoku wybrany algorytmem
6. Zapis rozwiązania na obrazie.

Można stwierdzić, iż nie odbiegają w znacznej części od finalnego rozwiązania oczywiście biorąc pod uwagę, że wstępne wymagania były tworzone bez większej wiedzy czy te kroki zadziałają dla sudoku, były jedynie to założenia bez testów itp.

2 Program i testy

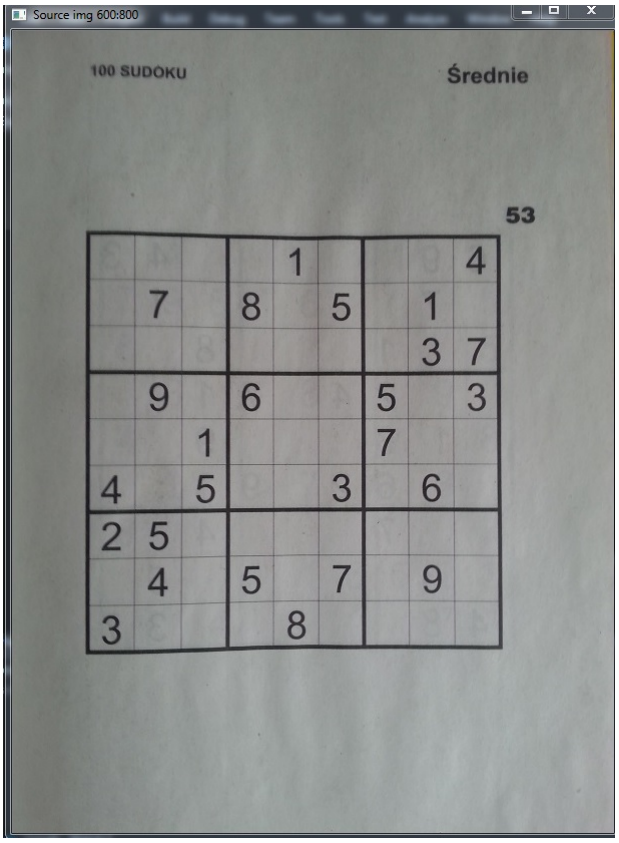
W programie został storzony plik nagłówkowy konfiguracyjny **SudokuConf.h**, najważniejsze flagi to :

```
#define DEBUG
#define TIME_MEASURMENT
```

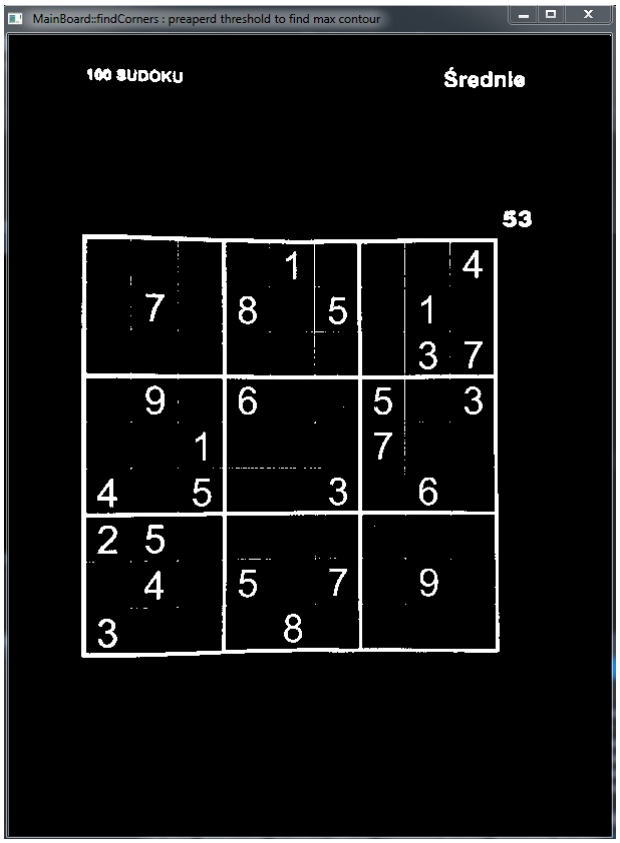
odkomentowanie ich powoduje włączenie dodatkowych opcji, dla pierwszej z nich znajdziemy debugowe printy w konsoli oraz obrazki z poszczególnych etapów programu, druga z nich mierzy jedynie czas wykonania programu podając wynik w milisekundach na konsole przygotowana ona jest do dalszego rozwijania tego programu poza ramami projektu z przedmiotu AIPO.

Program był testowany dla zdjęć sudoku załączonych w projekcie, przykładowe wywołanie dla jednego z nich z włączonymi flagami debugowymi:

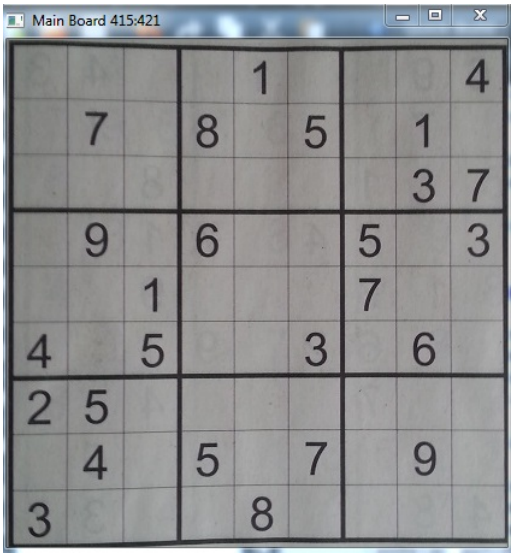
1) załadowanie zdjęcia

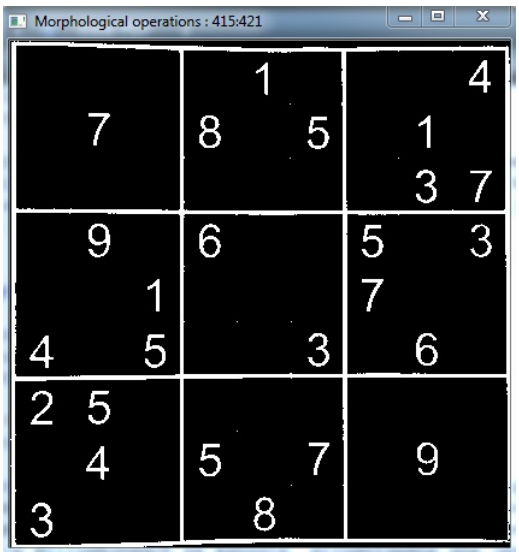


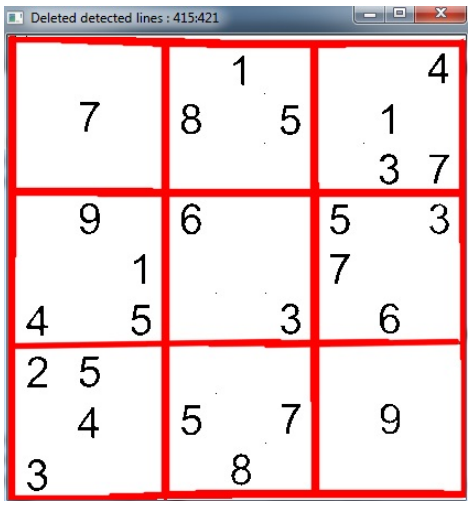
2) operacje morfologiczne + znalezienie głównej planszy sudoku



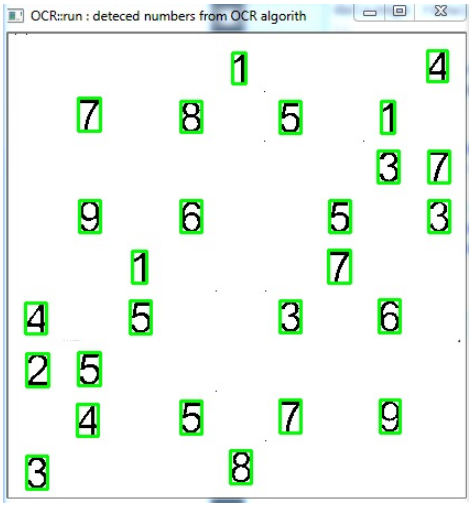
3) wycięta główna plansza sudoku + wydobycie samych cyfr



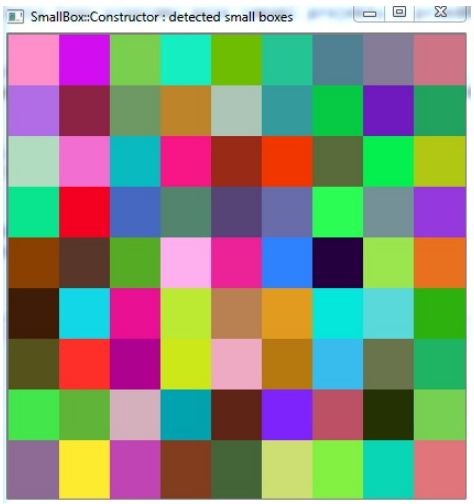




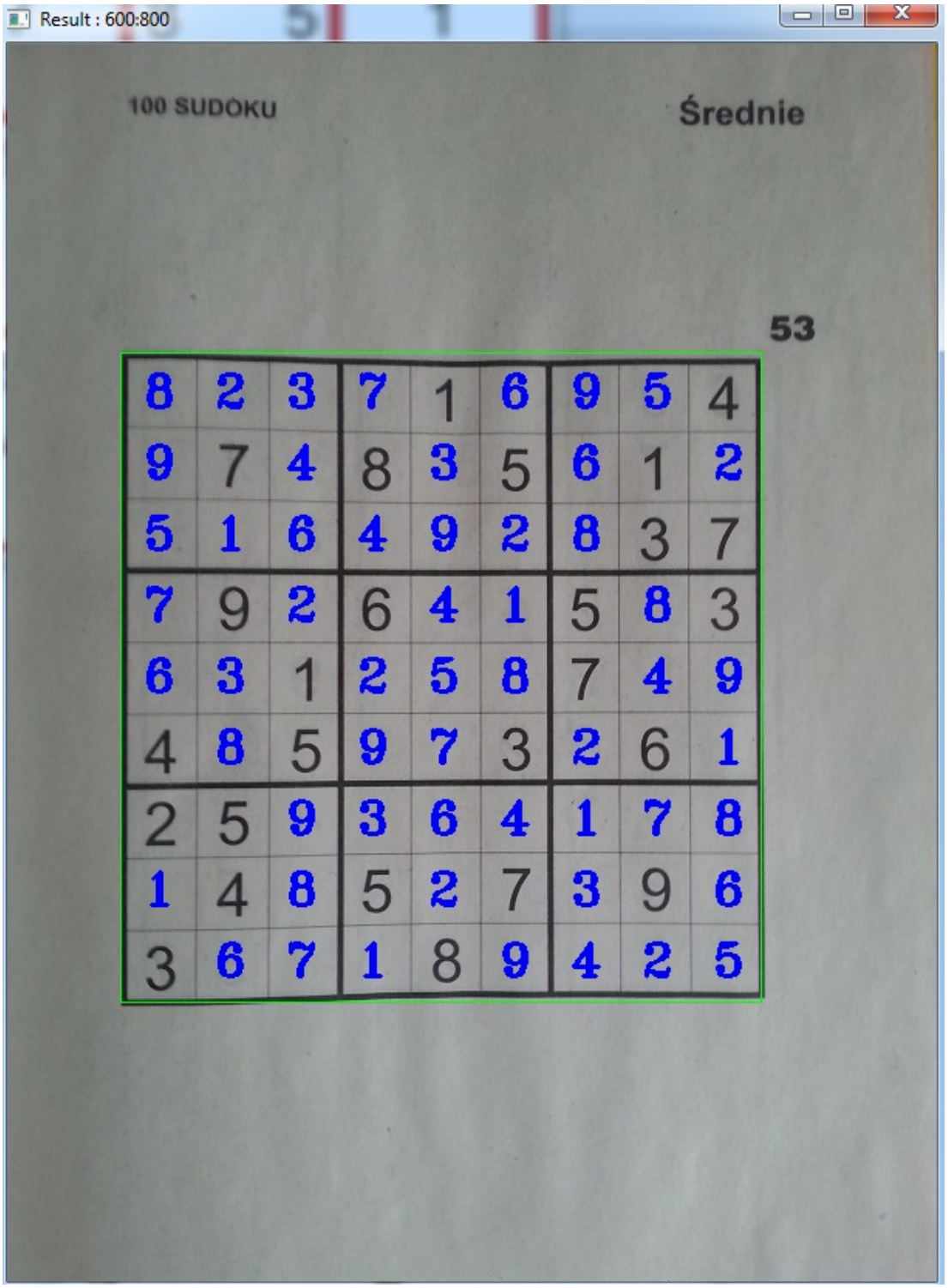
4) algorymt OCR



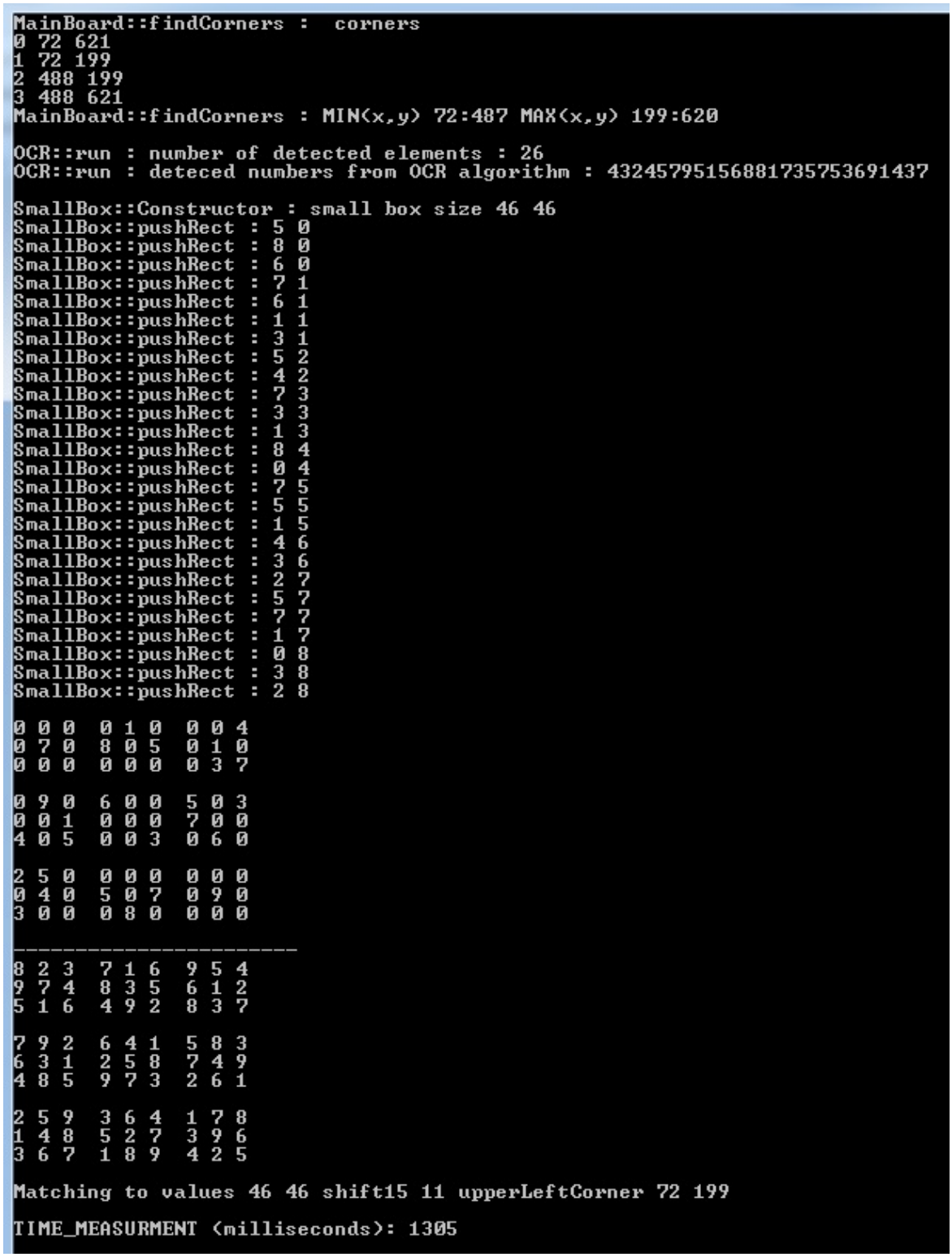
5) znalezienie poszczególnych kwadratów planszy (9x9 sudoku)

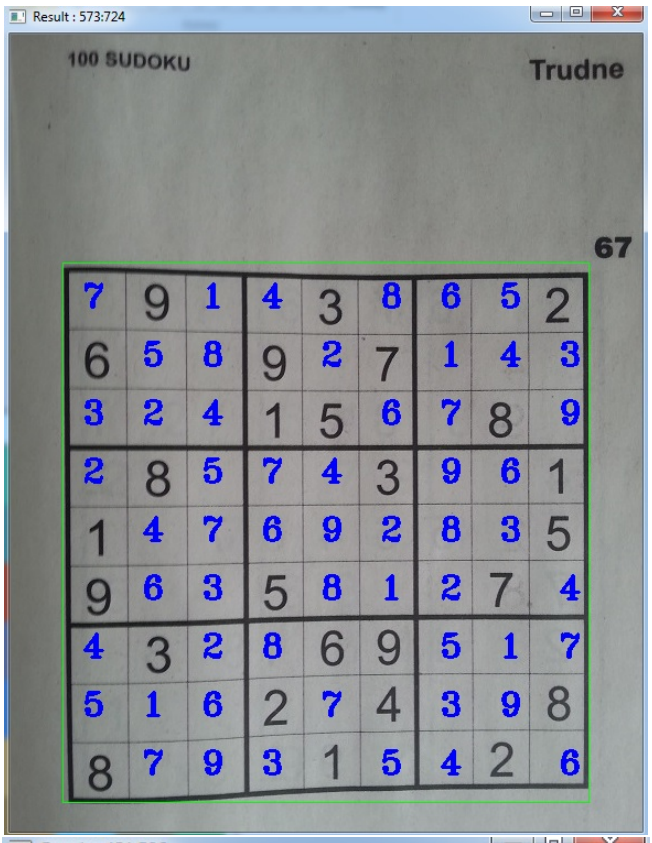


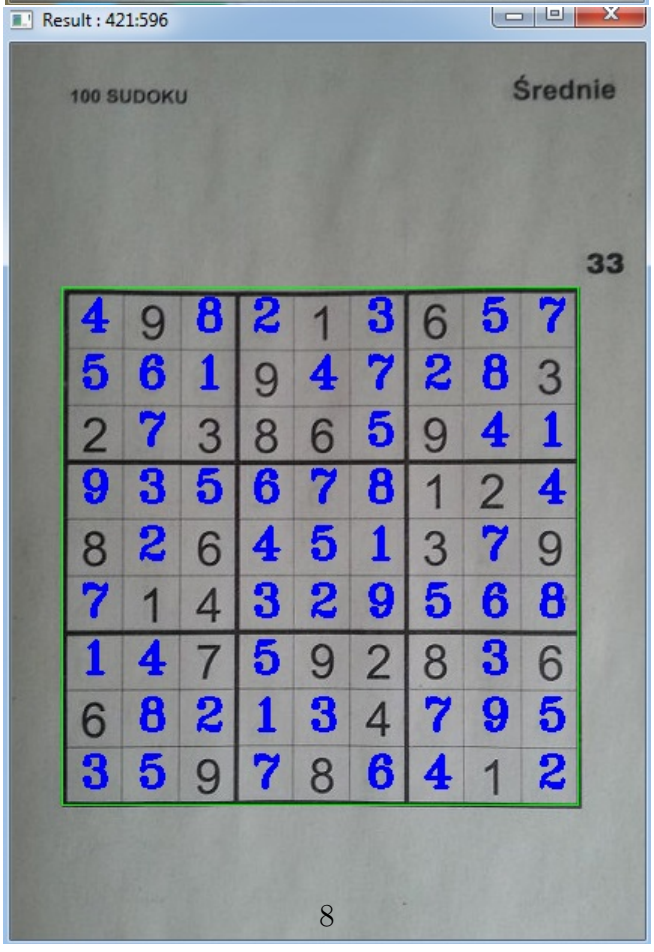
7) finalny efekt



+) konsola z debugowymi printami oraz inne rozwiązania




Result : 573:724
100 SUDOKU
Trudne
67
7 9 1 4 3 8 6 5 2
6 5 8 9 2 7 1 4 3
3 2 4 1 5 6 7 8 9
2 8 5 7 4 3 9 6 1
1 4 7 6 9 2 8 3 5
9 6 3 5 8 1 2 7 4
4 3 2 8 6 9 5 1 7
5 1 6 2 7 4 3 9 8
8 7 9 3 1 5 4 2 6


Result : 421:596
100 SUDOKU
Średnie
33
4 9 8 2 1 3 6 5 7
5 6 1 9 4 7 2 8 3
2 7 3 8 6 5 9 4 1
9 3 5 6 7 8 1 2 4
8 2 6 4 5 1 3 7 9
7 1 4 3 2 9 5 6 8
1 4 7 5 9 2 8 3 6
6 8 2 1 3 4 7 9 5
3 5 9 7 8 6 4 1 2

3 Podsumowanie

Projekt spełnia założenia „Rozwiązywanie sudoku na podstawie zdjęcia z kamery", został także przetestowany dla wyżej przedsawionych zdjęć. Pisząc sprawozdanie zdałem sobie sprawę, że małe zmiany wprowadziłbym w kodzie (programistyczne, nie mających wpływ na algorytm działania programu). Jedą ważna rzeczą, która mi przyszła do głowy także podczas pisania sprawozdania jest dodanie opcji testowania zdjęcia pod operacje morfologiczne mam tutaj na myśli na przykład taką rzecz jak dodanie pętli która będzie zmieniała paramet progru binaryzacji, która wykonuje się przed wykonaniem OCR razem z OCR następnie na podstawie otrzymanych cyfr będziemy testować tak długo aż znadziemy najwiekszą ilość cyfr. Podczas tworzenia projektu chciałem wykorzystać transformatę Hougha do znalezienia wszystkich lini (określenia wszystkich pól sudoku) i wtedy wykorzystania każdego pola z osobna do OCR niestety nie udało mi się uzyskać dobrej jakości zdjęcia po operacjach morfologicznych (nigdy nie otrzymałem dla testowanych zdjęć wszystkich 81 pól) dlatego postanowiłem zmienić koncepcje. Zdecydowałem się, aby znaleźć główne linie sudoku i je usunąć ze zdjęcia, tak aby nie kolidowały przy OCR, gdyż niekóre linie mogłyby zostać odczytane jako „l" lub „1". Chciałbym także w niedalekiej przyszłości dodać opcje „real time", dlatego została dodana funkcja mierzenia czasu. Ciekawi mnie także jak właśnie dodanie takich funkcjonalności np. ze znajdywaniem najlepszego progu dla binaryzacji będzie miało odzwierciedlenie w długości wykonywania całego algorytmu co przyczyni się do aplikacji „real time". Finalną wersją byłoby przenisienie tej stworzonej aplikacji na androida.