Politechnika Śląska
Wydział Matematyk Stosowanej
Kierunek Informatyka

Gliwice, 09.07.2020

Programowanie obiektowe i graficzne

Projekt zespołowy

"Find Undercover Cop"

Skład zespołu projektowego:

Konrad Lubera gr. lab. 1B

Robert Olej gr. lab. 1B

Wojciech Kajstura gr. lab. 1b

Wykaz zadań i prac zrealizowanych przez zespół projektowy:

Imię Nazwisko	Odpowiedzialny za:	Zrealizował zadania:
Kajstura Wojciech	1. UI	1A. Zaprojektował UI
		1B. Wykonał UI
	2. Zgodność ze wzorcem MVVM	2A. Napisał szkielet aplikacji
		2B. Nadzorował pozostałe zadania pod kątem wzorca
Lubera Konrad	1. Detekcję	1A. Zaprojektował algorytm detekcji
		1B .Wykonał algorytm detekcji.
	2. Konfiguracja Teseeract	2A. Zainstalował i skonfigurował Tessertact
		2B. Wykonał preprocesowanie obrazu
Olej Robert	1 Serializacje	1A. Zaprojektował mechanizm serializacji
		1B. Wykonał mechanizm
	2. Dokumentację	2A. Testował aplikację
		2B. Napisał dokumentację

1. Opis projektu.

Program ma za zadanie wykryć na zadanym mu zdjęciu samochodu jego tablice rejestracyjną, wyczytać jej numery i następnie zweryfikować czy pojazd należy do policji (tzn. jest nieoznakowanym radiowozem). Jeśli pojazd nie jest policyjną własnością to zostaje zwrócone województwo z jakiego pochodzi.

2. Wymagania

Do uruchomienia programu wymagany jest komputer z systemem Microsoft Windows.

3. Opis realizacji

Wykorzystane technologie

Program został stworzony w języku C# z wykorzystaniem technologii WPF oraz wzorca MVVM. Do detekcji tablicy rejestracyjnej na obrazie obrazie użyta została biblioteka Emgu CV (wrapper OpenCV dla C#), natomiast do odczytania tablicy użyto OCR Tesseract.

Skrócony opis wykorzystanych klas:

Model:

- Converters - statyczna klasa zawierająca konwertery typów wykorzystane w projekcie.
- Detection - klasa odpowiedzialna za detekcję tablicy rejestracyjnej na obrazie.
- Recognition - klasa odpowiedzialna za rozpoznawanie znaków na tablicy (OCR)
- LicensePlate - klasa w której znajdują się informacje o odczytanej tablicy
- Serialization - statyczna klasa odpowiedzialna za serializację.

ViewModel:

- RelayCommand, ViewModelBase - klasy bazowe viewmodelu
- MainViewModel - viewmodel projektu

View:

- MainWindow - widok naszej apliakcji

Opis Działania Detekcji

Za detekcję odpowiada klasa ***Detection*** w której znajduje się algorytm detekcji oraz mechanizm preprocesowania obrazu, mechanizmy te są opisaneponiżej, klasa znajduje się w pliku *Detection.cs.*

Detekcja tablicy rejestracyjnej na obrazie odbywa się poprzez znalezienie fragmentu obrazu o odpowiednich proporcjach do tablicy rejestracyjnej oraz posiadającym odpowiednie zagęszczenie kształtów odpowiadającym stosunkiem do liter.

Jednak najpierw aby można było znaleźć tablice należy przygotować obraz poprzez przekonwertowanie go do skali szarości.

Obraz passata w skali szarości.

Następnie następuje konwersja na obraz binarny oraz nałożenie filtru Gaussa aby wyeliminować szumy.

Binarny obraz passata.

Binarny obraz z filtrem.

Kolejnym etapem jest wyszukanie kontur na obrazie.

Oraz z zaznaczonymi konturami.

Ostatnim już etapem jest wykorzystanie algorytmu detekcji, wyszukanie tablicy, poprzez znalezienie miejsc w których zagęszczenie kontur pasujących stosunkiem do liter na tablicy jest odpowiednie, oraz przycięcie obrazu aby zawierał jak najmniejszy region wokół tablicy.

Obraz z zaznaczonym ROI.

Obraz który jest przekazywany do OCR.

Tesseract

Tesseract jest optyczne rozpoznawanie znaków silnika dla różnych systemów operacyjnych. To jest wolne oprogramowanie , wydany na licencji Apache w wersji 2.0, a rozwój został sponsorowany przez Google od 2006 roku.

W 2006 Tesseract został uważany za jeden z najbardziej precyzyjnych silników open source OCR wówczas dostępny. Dalej ponoć jest.

Do rozpoznawania znaków na tablicy rejestracyjnej użyliśmy OCR TESSERACT w wersji 3.3.0, nowsza wersja czyli 4.X zapewnia większa skutczność jednak nie ma jeszcze łatwych do zainstalowania pakietów dla .NET i wymaga to dużo pracy i pobierania paczek z

niewiadomego źródła. Przy odpowiednim przygotowaniu obrazu przed odczytanie przez ocr wyniki są zadowalające.

Aby efekty były satysfakcjonujące przede wszystkim należy zadbać o to aby tekst na obrazie był w jednej linii, przekrzywiony obraz bardzo obniża efektywność. W projekcie zastosowano algorytm który stara się znaleźć skrzywienie obrazu a następnie prostuje go.

```
        public void DeSkew()
        {
            if (outImage != null)
            {
                Image<Gray, byte> image = OutImage.Convert<Gray,
Byte>();
                double cannyThreshold = 180;
                double cannyThresholdLinking = 120;
                Image<Gray, Byte> cannyEdges =
image.Canny(cannyThreshold, cannyThresholdLinking);
                LineSegment2D[] lines = cannyEdges.HoughLinesBinary(1,
Math.PI / 180, 100, outImage.Width / 12,outImage.Width / 150 )[0];
                double[] angle = new double[lines.Length];

                for (int i = 0; i < lines.Length; i++)
                {
                    double result = (double)(lines[i].P2.Y -
lines[i].P1.Y) / (lines[i].P2.X - lines[i].P1.X);
                    angle[i] = Math.Atan(result) * 57.2957795;
                }
                double avg = 0;
                for (int i = 0; i < lines.Length; i++)
                {
                    avg += angle[i];
                }
                avg /= lines.Length;
                outImage = outImage.Rotate(-avg, new Bgr(0, 0, 0));
            }}
```

Aby zainicjować działanie OCRa należy utworzyć obiekt klasy Recognition. W której tworzony jest obiekt TesseractEngine który inicjuje OCRa.

```
       public Recognition(Bitmap image)
        {
            Bitmap new_dpi = new Bitmap(image);
            new_dpi.SetResolution(300, 300);
            ocr = new TesseractEngine("./tessdata", "eng",
EngineMode.TesseractAndCube);
            Read(image);
        }
```

```
private void Read(Bitmap image)
{
    processed = ocr.Process(image);
    Text = processed.GetText();
    Regex pattern = new Regex(@"[^0-9a-zA-Z]+");
    Text = pattern.Replace(Text, "");
}
```

Często przy odczytywaniu tekstu przez ocr pojawiały się w nim niepotrzebne spacji, znaki nowej linii czy symbole aby wyeliminować ten problem użyto wyrażenia regularnego. Metoda *Replace* pozbywa się wszystkich znaków poza literami oraz liczbami.

Mechanizm odczytywania tablicy rejestracyjnej zaimplementowano w klasie *Recognition*

Serializacja

By zweryfikować dokładność programu koniecznie było zapisanie wyników działania programu. Problem ten rozwiązano poprzez zapisanie do pliku w formacie json zserializowanego obiektu klasy LicensePlate oraz obrazu tablicy rejestracyjne w formatowaniu Base64.

Formatowanie Base64 pozwala nam skopiować tekst np. do onlineowego konwertera który przekonwertuje go na obraz.

Wykorzystany ocr wykorzystuje obraz typu Bitmap więc potrzebna była metoda która dokona konwersji z typu Bitmap na obraz zapisany w tekście o formatowaniu Base64. Metoda ta znajduje się w statycznej klasie Converters

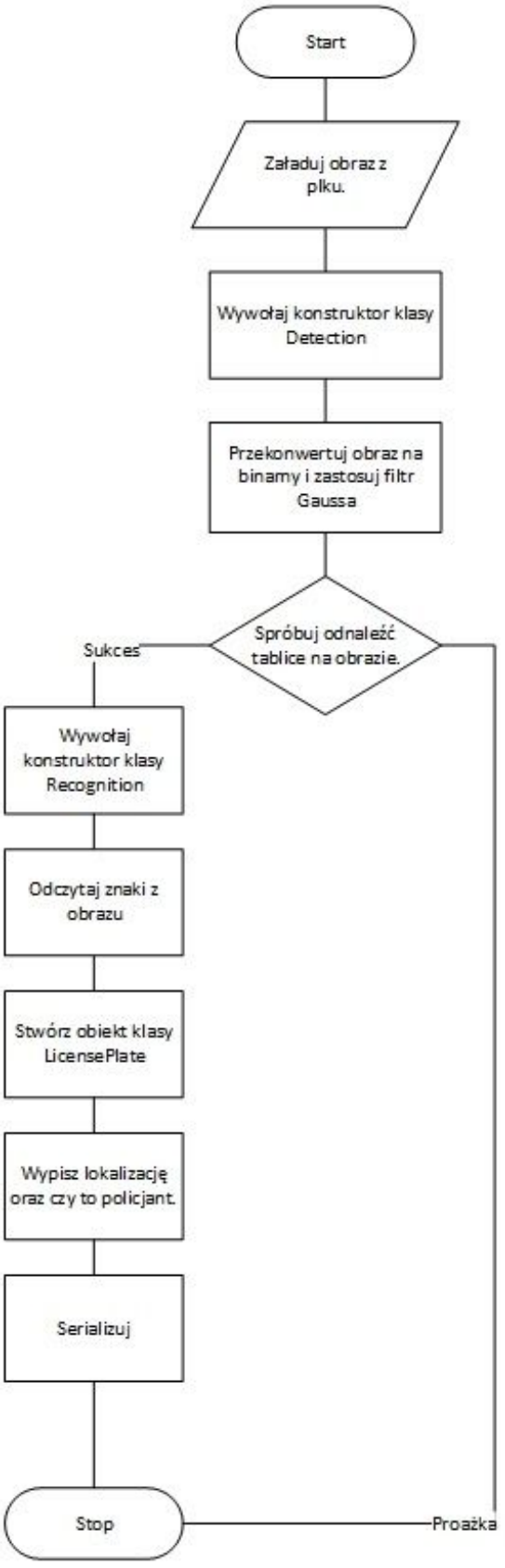
```
public static string ConvertImageToBase64(Bitmap image)
 {
    using (MemoryStream memory = new MemoryStream())
    {
       image.Save(memory, ImageFormat.Jpeg);
       byte[] byteImage = memory.ToArray();
       return Convert.ToBase64String(byteImage);
    }
}
```

Oto przykład skopiowanego tekstu w formatowaniu Base64 do konwertera codebeautify.org/base64-to-image-converter.

Przykładowa linijka z pliku json:
{"FullLicensePlate":"FSD81417","LocationShortcut":"FSD","LocationFullName":"powiat strzelecko-drezdenecki","LocationVoivodeship":"lubuskie","RandomCharacters":"81417","IsUndercoverCop":false}
/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsL…

Mechanizm serializacji został zaimplementowany w klasie *Serialization.*

Schemat blokowy działania aplikacji

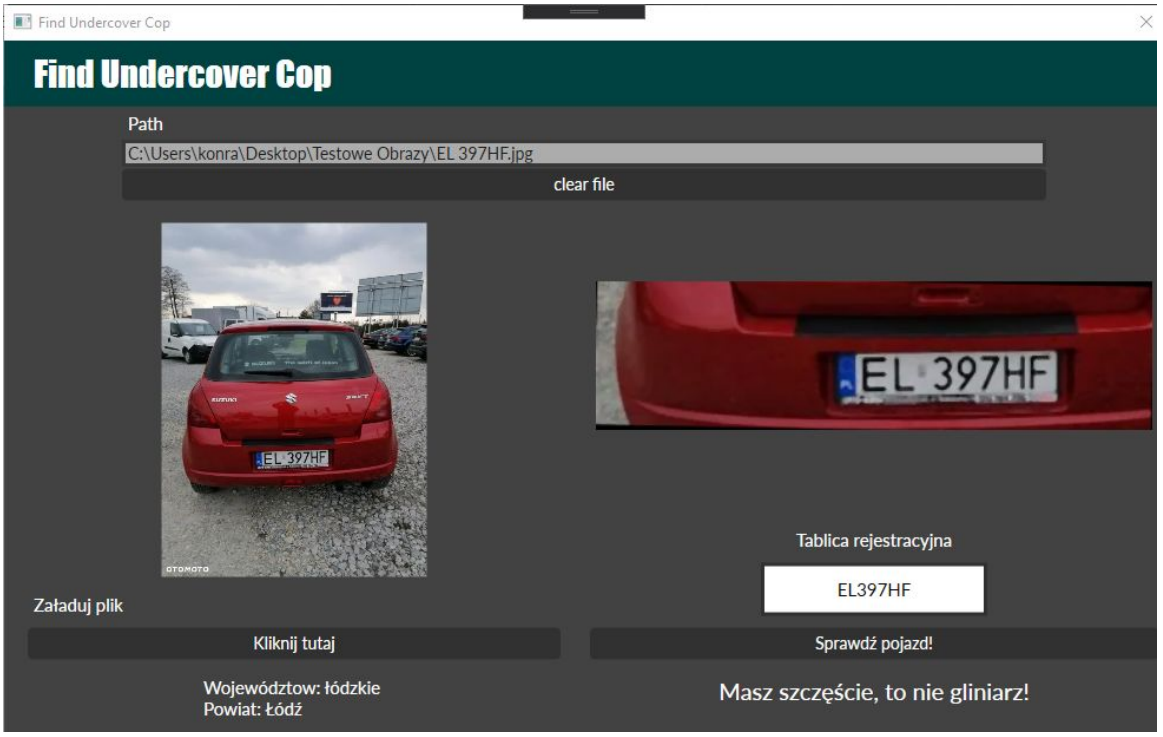


Testy

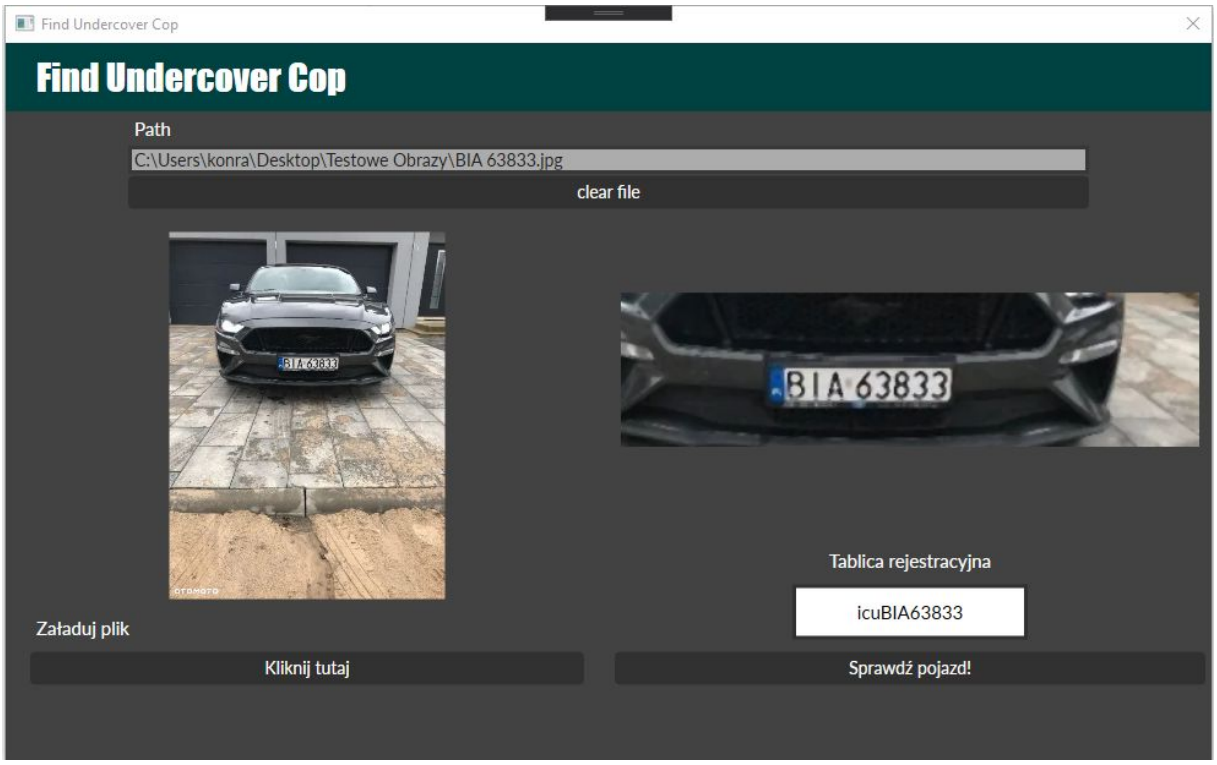
Dokładność programu można uznać za satysfakcjonującą wszystko jednak zależy od obrazu który wybierzemy. Niestety Tesseract posiada taką specyfikę, że najlepiej radzi sobie z obrazami wysokiej rozdzielczości, nie jest to zawszę reguła co zostanie pokazane poniżej, dodatkowo algorytm detekcji miewa problemy z rejonami w których jest dużo prostokątów np. okna sali gimnastycznej czy płoty.

Program nie posiada funkcji wykrywania więcej niż jednej tablicy na obrazie, jednak algorytm bez problemu na to pozwala, jest to kwestia implementacji, w przypadku gdy na obrazie jest więcej niż jedna znaleziona tablica program wybierze tą wyżej.

Oto przykład poprawnie działającego programu, tablica została znaleziona, litery poprawnie odczytane lokalizacja także:

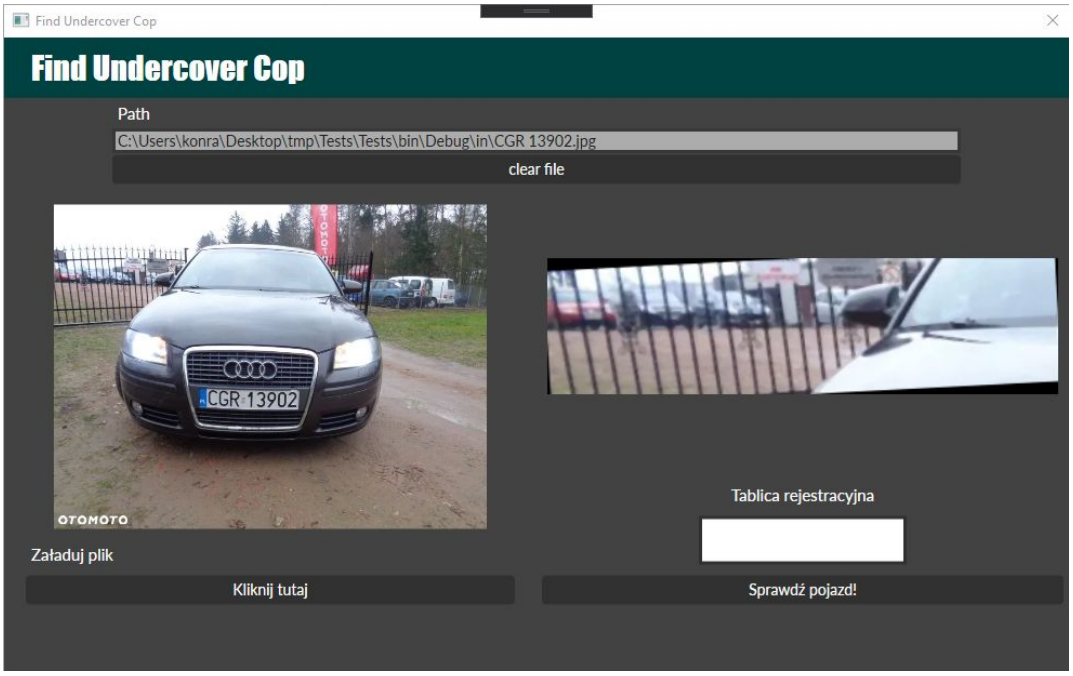


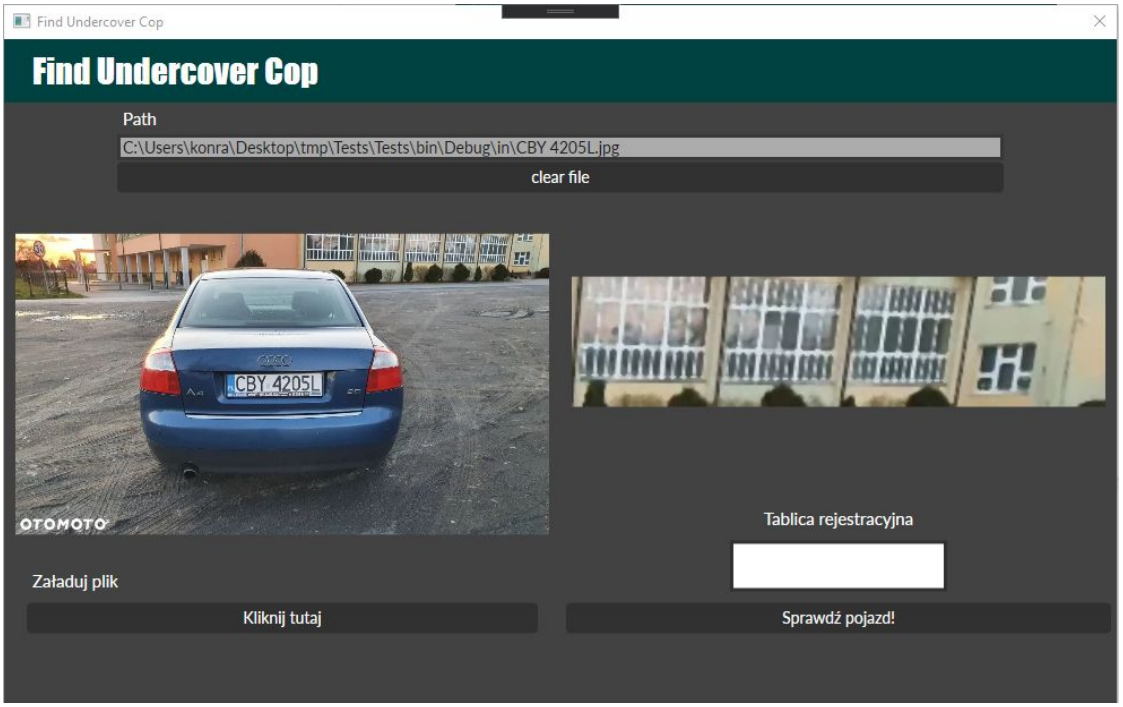
W wielu przypadka zdarza się, że OCR znajdzie litery których na obrazie nie ma jednak dodatkowa walidacja mogłaby wyeliminować ten problem:



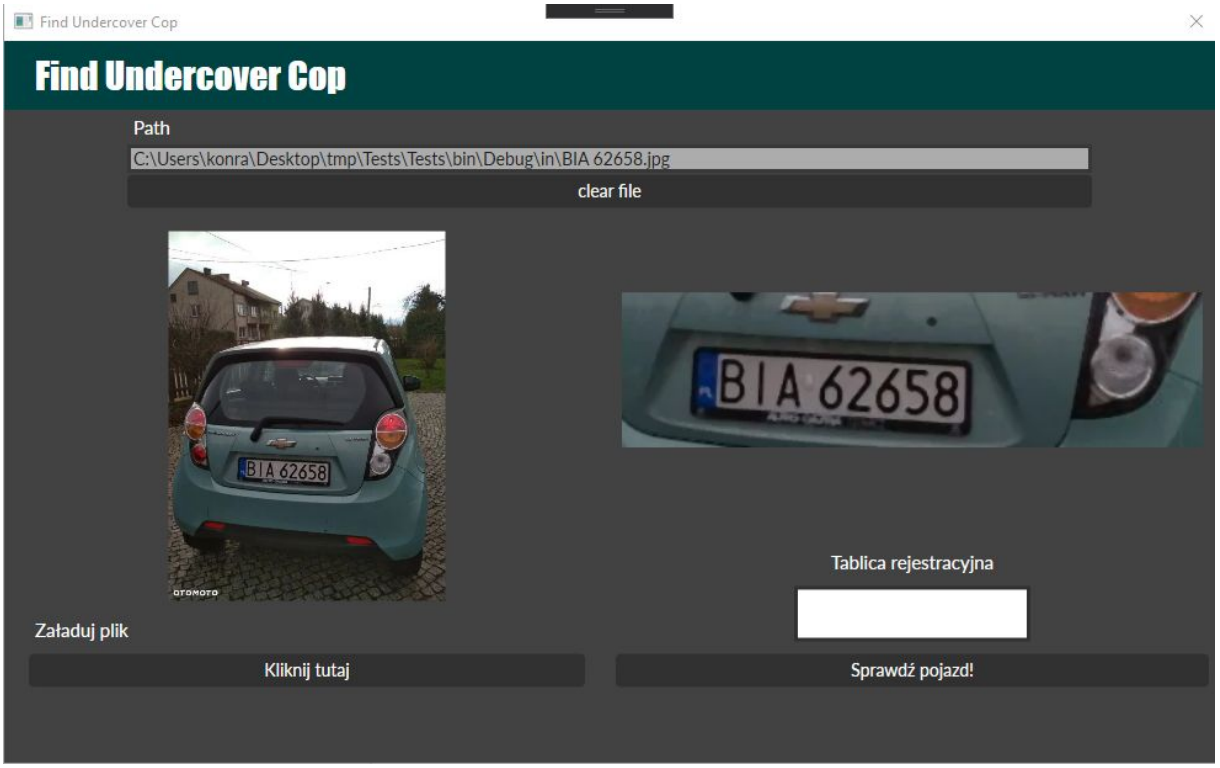
W tym dokładnie przypadku obraz tablicy jest w bardzo słabej jakości jednak OCR w miarę sobie z nim poradził.

Poniżej przykład w którym płot oraz okna sali gimnastycznej, przez zagęszczenie prostokątów, zostały sklasyfikowane jako tablica:

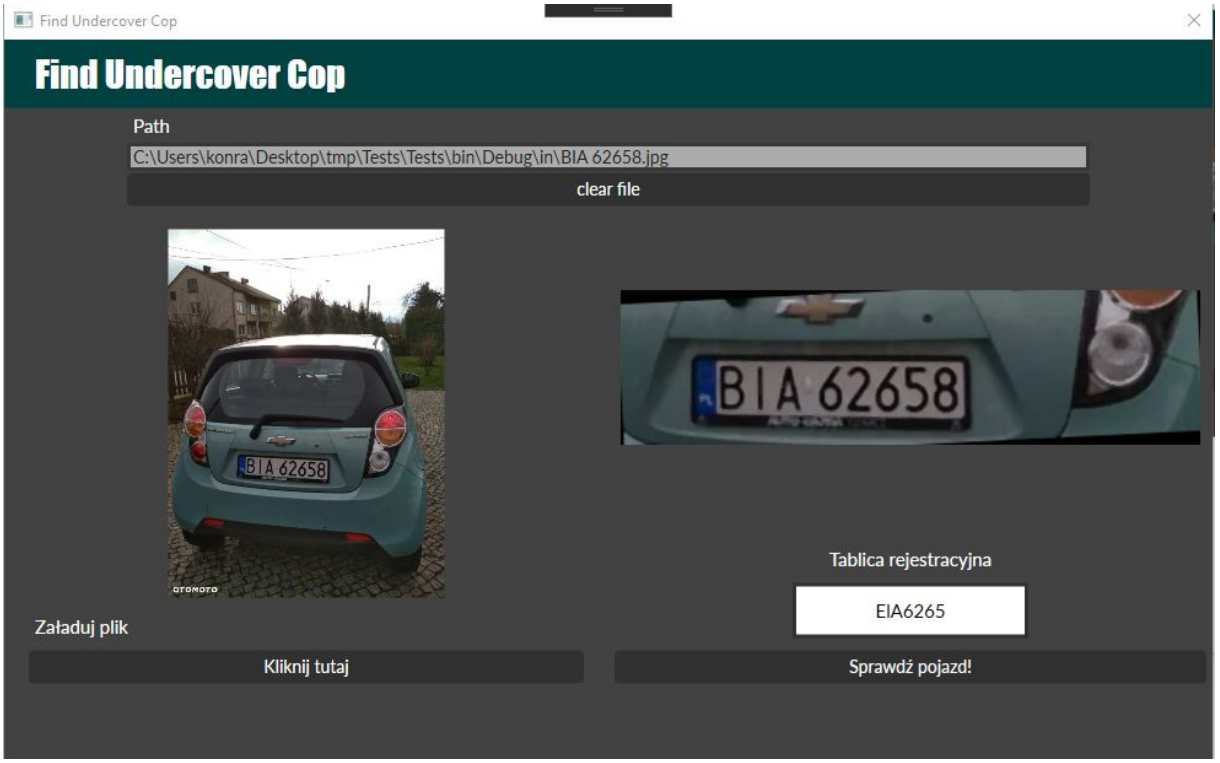




W sekcji o tesseract wspomniano iż bardzo ważne jest by obraz był prosty, poniższy test jest dość jaskrawym tego dowodem, obraz ma słabą rozdzielczość, jest ciemny a zdjęcie było robione pod kątem. Na obrazie który nie został wyprostowany, OCR nie znalazł nic jednak na obrazie prosty OCR zadziałał niemalże poprawnie.



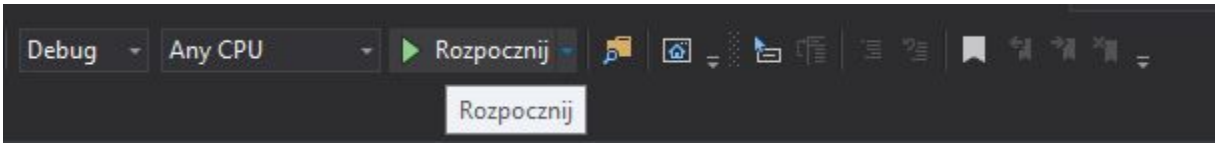
Funkcja DeSkew() nie została użyta.



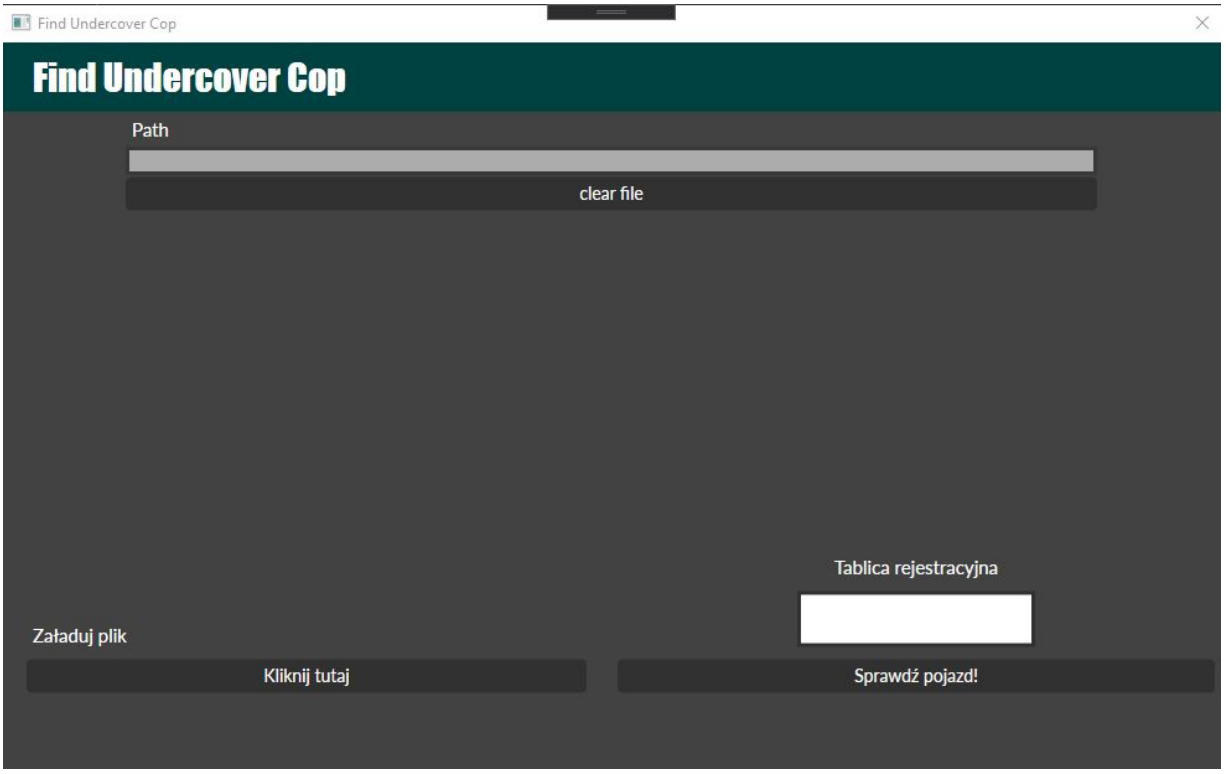
Funkcja DeSkew() użyta

4. Instrukcja użytkownika

Pracę programu rozpoczynamy skompilowaniem kodu w programie wspierającym język C Sharp (np. Visual Studio).

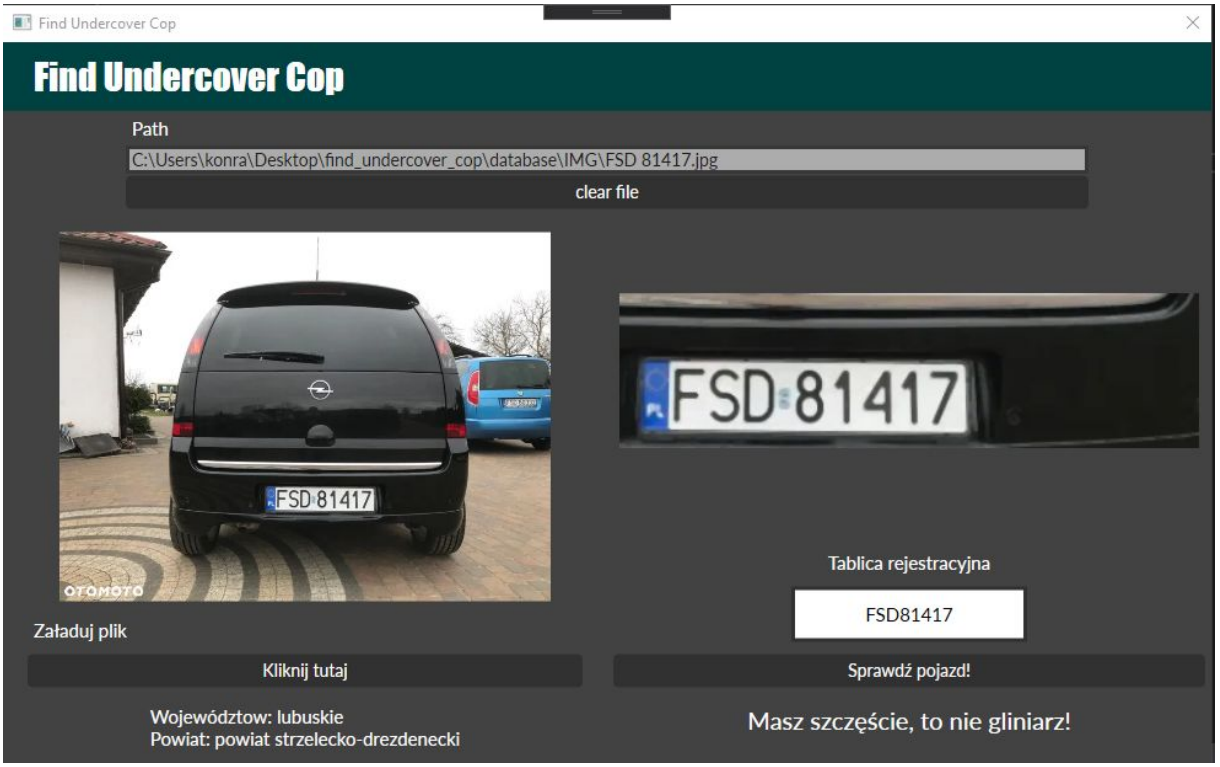


Następnie powinno ukazać się okno, z którego poziomu obsługujemy program.



Mamy możliwość dodania zdjęcia samochodu, który ma zostać sprawdzony przez program.

Jeśli to nieoznakowany radiowóz to program po naciśnięciu przycisku “Sprawdź pojazd!” poinformuje o tym, w innym razie dostaniemy informację z jakiego województwa jest sprawdzany samochód.



Obsługa jest prosta i intuicyjna.

5. Podsumowanie i wnioski.

Główną funkcję programu udało nam się zrealizować - wykrywanie znaków na tablicy i weryfikacja samochodu pod kątem nieoznakowanego radiowozu.

Niestety ta funkcje nie zawsze działają dokładnie.

Zdarza się, że rejon tablicy rejestracyjnej nie zostanie poprawnie znaleziony lub znaki wyświetlone przez program będą zamienione miejscami w porównaniu do tych na oryginalnej tablicy. Bywa też tak, że pewne podobne do siebie znaki program często wyczytuje źle (np. myli ‘1’ z ‘7’ lub ‘i’).

Aby poprawić działanie aplikacji można by użyć klasyfikatora Harra do wyszukiwania tablicy, jednak do jego poprawnego działania potrzebna jest bardzo duża liczba próbek, dodatkowo można by zastosować OCR Tesseract w nowszej wersji.

6. Dodatek - udokumentowanie wykorzystania systemu kontroli wersji.

Adres naszego repozytorium: https://github.com/konradluberapolsl/find_undercover_cop