Szkoła Główna Gospodarstwa Wiejskiego

w Warszawie

Wydział Wydział Zastosowań Informatyki i Matematyki

Sylwester Turski

167497

Inteligentny algorytm do rozpoznawania i zliczania miejsc parkingowych na podstawie serii obrazów z kamery

Intelligent algorithm to recognize and count the parking spaces on the basis of a series of images from the camera

Praca dyplomowa inżynierska

na kierunku informatyka

Praca wykonana pod kierunkiem

Dra Pawła Hosera

Katedra Zastosowań Informatyki

Warszawa, 2017 rok

**Oświadczenie promotora pracy**

Oświadczam, że niniejsza praca\*/wskazane przez autora rozdziały pracy dyplomowej przygotowanej zespołowo\* została/zostały\* przygotowana pod moim kierunkiem   
i stwierdzam, że spełnia\*/spełniają\* warunki do przedstawienia tej pracy w postępowaniu   
o nadanie tytułu zawodowego.

Data .................................... Podpis promotora pracy ...................................................

**Oświadczenie autora pracy**

Świadom odpowiedzialności prawnej, w tym odpowiedzialności karnej za złożenie fałszywego oświadczenia, oświadczam, że niniejsza praca dyplomowazostała napisanaprzeze mnie samodzielnie i nie zawieratreści uzyskanych   
w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności ustawą z dnia   
4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. Nr 90 poz. 631 z późn. zm.)

Oświadczam, że przedstawiona praca nie była wcześniej podstawą żadnej procedury związanej z nadaniem dyplomu lub uzyskaniem tytułu zawodowego.

Oświadczam, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Przyjmuję do wiadomości, że praca dyplomowa poddana zostanie procedurze antyplagiatowej.

Data ..................................... Podpis autora pracy .............................................

**Streszczenie**

**Inteligentny algorytm do rozpoznawania i zliczania miejsc parkingowych na podstawie serii obrazów z kamery**

Streszczenie pracy - do 1000 znaków (12)

Słowa kluczowe - do 6 (12)

**Summary** (12 bold)

**Title** (12 bold)

Summary - 1000 words max (12)

Keywords – 6 max (12)

Spis Treści

[Wstęp 7](#_Toc468279497)

[Cel i zakres pracy 7](#_Toc468279498)

[Przegląd piśmiennictwa 7](#_Toc468279499)

[Materiały i metodykę pracy 7](#_Toc468279500)

[Omówienie i dyskusję wyników 7](#_Toc468279501)

[Podsumowanie i wnioski 7](#_Toc468279502)

[Spis piśmiennictwa. 7](#_Toc468279503)

# Wstęp

# Cel i zakres pracy

# Przegląd piśmiennictwa

# Projekt rozwiązania

## Biblioteki i technologie użyte podczas pisania algorytmu

OpenCV (Open Source Computer Vision) – Popularna biblioteka funkcji do rozpoznawania obrazów w czasie rzeczywistym. Posiada między innymi funkcje do manipulacji obrazami, wykrywania cech obrazu i uczenia maszynowego. W 1999 roku projekt OpenCV został zainicjalizowany przez firmę Intel. Biblioteka jest napisana w języku C++. Jako wrapper w języku C# wybrałem bibliotekę OpenCvSharp. API tej biblioteki jest bardzo zbliżone do oryginału, dodatkowo posiada fluent API.

WPF (Windows Presentation Foundation) – Framework do tworzenia interfejsu użytkownika (ang. user interface) dla platformy .NET, stworzony przez firmę Microsoft. WPF kładzie nacisk na grafikę wektorową, dzięki której większość kontrolek można skalować bez utraty jakości, czy pixelizacji. Wybór na tą technologię zapadł ponieważ można w łatwy sposób projektować wygląd aplikacji jednocześnie zachowując pełną kontrolę nad zachowaniem aplikacji podczas zmiany wielkości okna. Rozważaną alternatywą do tej technologii były wbudowane w bibliotekę OpenCV funkcję do tworzenie interfejsu użytkownika. Jednak nie pozwalały one zarządzać położeniem i wielkością kontrolek, przez co ostateczny wybór padł na technologię WPF

LiteDB – Nie relacyjna, osadzona baza danych dla platformy .NET. Podobnie jak popularna baza MongoDB baza LiteDB jest bazą dokumentową, w której można przechowywać obiekty, jednak w przeciwieństwie do MongoDB LiteDB jest bazą osadzoną i nie wymaga serwera, a dane są zapisywane w pojedynczym pliku. Dzięki temu mogę w łatwy sposób przechowywać dane w aplikacji bez konieczności instalacji dodatkowego oprogramowania czy projektowania tabel do przechowywania skomplikowanych obiektów.

## Badane cechy obrazu

Organoleptycznie można zauważyć że samochody od podłoża odróżnia kolor karoserii, więc pierwszą badaną przeze mnie cechą było badanie saturacji na obszarze w którym użytkownik zaznaczył położenie parkingu. Poniżej przedstawiam uproszczony kod który oblicza tą cechę

public static double SaturationFeature(Mat mask, Mat src, int threshold)

{

var saturatedPixel = src

.CvtColor(ColorConversionCodes.BGR2HSV)

.GetSaturation()

.Mul(mask)

.Threshold(threshold, type: ThresholdTypes.Binary)

.CountNonZero();

var all = mask

.CountNonZero();

return (double)saturatedPixel / all;

}

[Wykres ROC]

Jak nie trudno się domyślić ta cecha nie jest skuteczna przy wykrywaniu samochodów o kolorze karoserii który posiada niską saturację (np. białym, czarnym, szarym) dając *false negative*. Również szum RGB generowany przez kamerę przy słabym oświetleniu może generować wyniki *false positive*.

Rozszerzeniem tej cechy jest badanie wartości średniej i odchylenia standardowego dla składowych kolorów HSV

[kod ]

[wykres ROC]

Kolejną cechą którą można zauważyć organoleptycznie są krawędzie. Na obrazie na którym zastosano wykrywanie krawędzi np. metodą *Cannego*, można zauważyć że obszary na których znajdują się samochody dają widocznie więcej krawędzi od podłoża. Na podstawie tej obserwacji wyliczam stosunek pixeli z krawędziami do całości obszaru. Poniżej przedstawiam uproszczony kod który oblicza tą cechę.

public static double EdgeFeature(Mat mask, Mat src)

{

var all = mask.CountNonZero();

var edgePixelCount = src

.Canny(40, 50)

.Mul(mask)

.CountNonZero();

return (double)edgePixelCount / all;

}

[Wykres ROC]

# Omówienie i dyskusja wyników

# Spis piśmiennictwa

Wyrażam zgodę na udostępnienie mojej pracy w czytelniach Biblioteki SGGW   
w tym w Archiwum Prac Dyplomowych SGGW

.................................................................

*(czytelny podpis autora pracy)*