



**POLITECHNIKA
GDAŃSKA**

WYDZIAŁ FIZYKI TECHNICZNEJ
I MATEMATYKI STOSOWANEJ

Imię i nazwisko studenta: Filip Cyglicki

Nr albumu: 169668

Studia pierwszego stopnia

Forma studiów: stacjonarne

Kierunek studiów: Fizyka Techniczna

Specjalność: Informatyka stosowana

PRACA DYPLOMOWA INŻYNIERSKA

Tytuł pracy w języku polskim: Porównanie klasyfikatorów LBP i Haar

Tytuł pracy w języku angielskim: Comparison of LBP and Haar classifiers

Potwierdzenie przyjęcia pracy	
Opiekun pracy	Kierownik Katedry/Zakładu (pozostawić właściwe)
podpis	podpis
dr inż. Bartosz Reichel	

Data oddania pracy do dziekanatu:



**POLITECHNIKA
GDAŃSKA**

WYDZIAŁ FIZYKI TECHNICZNEJ
I MATEMATYKI STOSOWANEJ

OŚWIADCZENIE dotyczące pracy dyplomowej zatytułowanej: Porównanie klasyfikatorów LBP i Haar

Imię i nazwisko studenta: Filip Cyglicki

Data i miejsce urodzenia: 28.03.1996, Łąwa

Nr albumu: 169668

Wydział: Wydział Fizyki Technicznej i Matematyki Stosowanej

Kierunek: fizyka techniczna

Poziom kształcenia: pierwszy

Forma studiów: stacjonarne

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. 2018 poz. 1191 z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (Dz. U. 2018 poz. 1668 z późn. zm.),¹ a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

Potwierdzam zgodność niniejszej wersji pracy dyplomowej z załączoną wersją elektroniczną.

Gdańsk, dnia

.....
podpis studenta

Streszczenie

Praca powstała w celu porównania algorytmów HAAR'a i LBP (z ang. Local Binary Patterns) by sprawdzić w jaki sposób wpływa ilość pozytywnych i negatywnych próbek ich wysokość i szerokość czy ilość etapów mają wpływ na czas szkolenia oraz skuteczności jaką uzyskują wytrenowane klasyfikatory. Dodatkowym celem pracy jest zapoznanie się z narzędziami oferowanymi przez bibliotekę OpenCV. Efektem pracy jest powstanie programu komputerowego pozwalającego na masowe szkolenie klasyfikatorów z różną konfiguracją. Wyniki pracy zostały przedstawione w tabelach. W pracy przedstawiono w jaki sposób przebiega trening i użycie klasyfikatora jak również w jaki sposób zweryfikowano jego skuteczność.

Słowa kluczowe: sztuczna inteligencja, rozpoznawanie obrazów, widzenie maszynowe, optymalizacja.

Dziedzina nauki i techniki, zgodnie z wymogami OECD: informatyka techniczna i telekomunikacja, informatyka (dziedzina nauk technicznych)

Abstract

The goal of this paper is to compare HAAR and LBP algorithms, to check the influence of number of positive and negative samples, their height and width or number of stages result in different training times and effectiveness which is achieved by trained classifier. Additional target of it, is to get more knowledge about tools offered by OpenCV library. The outcome of this paper is computer program which helps in mass learning the classifiers with different configuration. The results are displayed in tables. In this paper explained how the training is done, how the classifier is used and how his effectiveness was verified.

Keywords: artificial Intelligence, image recognition, computer vision, optimisation

Wykaz ważniejszych oznaczeń i skrótów	6
• Wstęp i cel pracy	7
• OpenCV	9
• Tytuł podrozdziału	numer strony
• Tytuł punktu podrozdziału	numer strony
• Tytuł podrozdziału	numer strony
----- kolejne rozdziały, podrozdziały i punkty	numer strony
x. Podsumowanie	numer strony
Wykaz literatury	numer strony
Wykaz rysunków	numer strony
Wykaz tabel	numer strony
Dodatek A	numer strony
Dodatek B	numer strony

Wykaz ważniejszych skrótów i oznaczeń

LBP - Local Binary Patterns

1. Wstęp

Wraz z rozwojem technologii, naukowcy zaczęli zastanawiać się nad tym czy maszyny są w stanie myśleć. W 1956 roku podczas jednej z konferencji dotyczącej tego tematu, John McCarthy użył sformułowania 'sztuczna inteligencja'[1]. Od tego czasu minęło wiele lat a termin ten dotyczy już nie tylko podejmowania decyzji przez maszynę liczącą ale między innymi również rozpoznawaniem obrazów.

Widzenie maszynowe poruszone w niniejszej pracy jest złożonym tematem. W celu ułatwienia powstało wiele algorytmów pomagających w klasyfikacji obrazów jak i oprogramowania usprawniających ich implementację. Jednym z przykładów biblioteki wspomagających tworzenie oprogramowania do rozpoznawania obrazów jest OpenCV[2] (ang. Open Source Computer Vision). W skład niej wchodzi algorytmy optymalizujące, wspomagające uczenie maszynowe i mogą zostać użyte do rozpoznawania twarzy jak i śledzeniem ruchu źrenic czy usuwania efektu czerwonych oczu powstałych przy użyciu flesza.

Dużym problemem jest złożoność obliczeniowa, nawet małe zdjęcia składają się z wielu pikseli a przeprowadzenia operacji na nich operacji wymaga wiele zasobów. Z tego powodu powstały sposoby klasyfikatory zajmujące się optymalizacją tego zadania. Wybór konkretnej metody nie jest prostym zadaniem, często ze skutecznością wiąże się dłuższy czas uczenia lub większe zapotrzebowanie na czas procesora czy pamięć. Z tego powodu dobór odpowiedniego klasyfikatora jest zadaniem nietrywialnym.

W ramach pracy porównane zostaną dwa algorytmy Viola-Jones'a zwany również Treningiem Haara i LBP (ang. Local Binary Patterns). Jej rezultatem będzie tabela zawierająca czas jaki był potrzebny do nauki algorytmu, ile próbek zostało użytych a także skuteczność z jaką udało się zakwalifikować zdjęcia.

2. Cel i zakres

Celem niniejszej pracy jest porównanie klasyfikatorów widzenia maszynowego i opisanie ich różnic.

- Przegląd literatury na temat widzenia komputerowego i OpenCV
- Przegląd literatury na temat algorytmów Viola-Jones'a i LBP
- Wybór implementacji wymienionych klasyfikatorów
- Stworzenie aplikacji umożliwiającej porównanie
- Wykonanie testów i utworzenie pomiarów

3. OpenCV

To biblioteka z licencją open source do widzenia komputerowego i uczenia maszynowego. Powstała w celu wprowadzenia wspólnej infrastruktury dla programów do widzenia komputerowego i percepcji maszynowej w komercyjnych rozwiązaniach[2]. Wspomaga w tworzeniu aplikacji w C++, Javie, Pythonie na platformy Windows, Android, Linux czy Mac OS.

3.1. Historia OpenCV

Zapoczątkowany przez firmę Intel projekt, początkowo miał usprawnić zadania o wysokiej złożoności obliczeniowej. W projekt zaangażowani byli głównie eksperci od optymalizacji i wydajności których celem było:

- stworzenie otwartej i zoptymalizowanej infrastruktury do przetwarzania obrazów,
- rozpowszechnić wiedzę i narzędzia na których programiści mogą polegać i bazować,

Od 2015 roku OpenCV zostało przejęte przez organizację non-profit OpenCV.org[4].

3.2. Aplikacje wchodzące w skład OpenCV

Po zbudowaniu biblioteki OpenCV mamy dostęp do kilku aplikacji które wspomagają szkolenie w ich skład wchodzi:


3.1.1. Annotation (ang. adnotacja)

Aby uruchomić to narzędzie z linii komend potrzebujemy podać ścieżkę do pliku w którym znajdują się obrazy, a także gdzie i pod jaką nazwą chcemy zapisać plik -info. Jeżeli podamy poprawne parametry na ekranie powinno pokazać się pierwsze z naszych zdjęć na którym mamy możliwość zaznaczyć, zatwierdzić wyznaczony obszar lub usunąć ostatni a następnie przejść do kolejnego zdjęcia, tak jak pokazano na rysunku 3.1. Jeżeli chcemy utworzyć plik zawierający obrazy tła czyli takie które nie zawierają obiektu którego chcemy nauczyć obiekt, wystarczy nic nie zaznaczać. Kompletny dla zdjęć zawierających obiekt, powinien składać się ze ścieżki do zdjęcia, ilości obiektów jakie znajdują się na wybranym obrazie i ich współrzędne.

```

>opencv_annotation.exe --images=Images/Positive --annotations=pos.info
* mark rectangles with the left mouse button,
* press 'c' to accept a selection,
* press 'd' to delete the latest selection,
* press 'n' to proceed with next image,
* press 'esc' to stop.

```



Images/Negative\Negative10.jpg 0	Images/Positive\Positive10.jpg 1 10 2 85 34
Images/Negative\Negative100.jpg 0	Images/Positive\Positive100.jpg 1 12 8 80 28
Images/Negative\Negative101.jpg 0	Images/Positive\Positive101.jpg 1 10 5 84 31
Images/Negative\Negative102.jpg 0	Images/Positive\Positive102.jpg 1 9 5 83 30
Images/Negative\Negative103.jpg 0	Images/Positive\Positive103.jpg 1 10 5 87 30
Images/Negative\Negative104.jpg 0	Images/Positive\Positive104.jpg 1 9 7 88 28
Images/Negative\Negative105.jpg 0	Images/Positive\Positive105.jpg 1 9 7 84 29
Images/Negative\Negative106.jpg 0	Images/Positive\Positive106.jpg 1 7 9 88 30

Rys. 3.1. Przykład użycia programu annotation

3.1.2. Createsample (ang. stwórz próbkę)

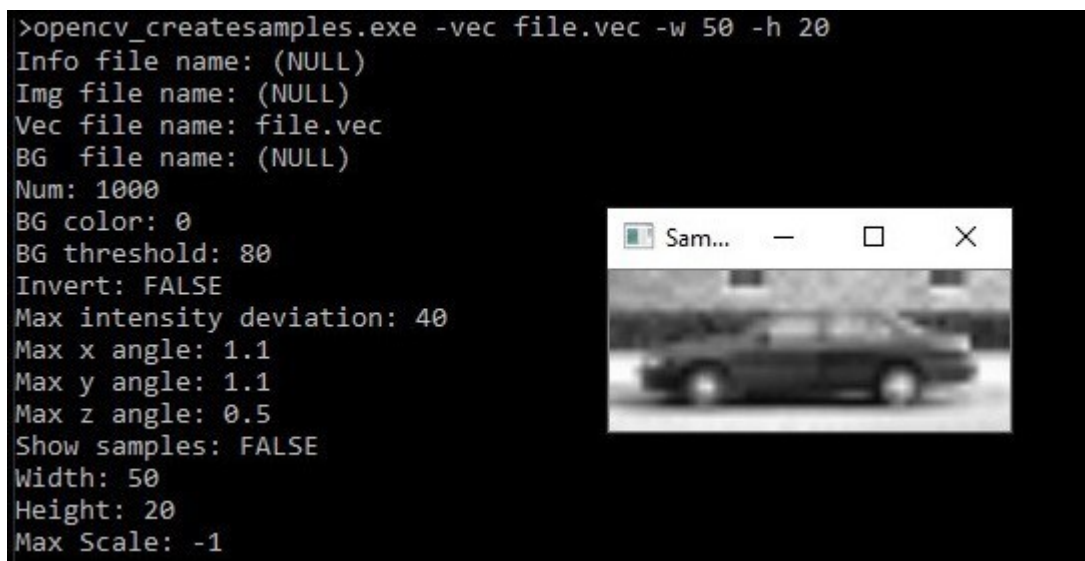
To narzędzie służy do wykonania pliku -vec zawierającego obrazy przygotowane do wyszkolenia klasyfikatora. Można go wygenerować na dwa sposoby, podając jeden obrazek na którym zawieramy nasz obiekt a następnie plik -info z obrazkami lub zamiast pojedynczego zdjęcia podać również plik -info z pozytywnymi obrazkami. Dobrą praktyką jest mimo wszystko używanie pliku z wieloma różnymi obrazami przedstawiającymi obiekt w ten sposób zwiększamy szanse na lepsze nauczanie klasyfikatora. Program wymaga także podania wysokości i szerokości próbek jakie chcemy uzyskać. Przykład użycia narzędzia do tworzenia próbek pokazano na obrazku 3.2.1. Jeżeli nie podamy plików -info lub pojedynczego obrazka a także ścieżka -vec będzie zawierała już istniejący plik to program wyświetli nam przygotowane próbki, tak jak przedstawiono na obrazku 3.2.2.

```

>opencv_createsamples.exe -vec file.vec -info pos.info -bg neg.info
-num 550 -w 50 -h 20
Info file name: pos.info
Img file name: (NULL)
Vec file name: file.vec
BG file name: neg.info
Num: 550
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 50
Height: 20
Max Scale: -1
RNG Seed: 12345
Create training samples from images collection...
Done. Created 550 samples

```

Rys. 3.2.1. Przykład utworzenia próbek za pomocą narzędzia createsample



Rys. 3.2.2. Wyświetlanie próbek za pomocą narzędzia createsample

3.1.3. *Traincascade* (ang. *Trenuj klasyfikator*)

Za jego pomocą można stworzyć klasyfikator. Jako parametry należy podać mu ścieżkę do katalogu w którym zostaną utworzone pliki klasyfikatora, plik -vec i -info z zdjęciami tła, liczbę pozytywnych i negatywnych próbek użytych w każdym etapie szkolenia, ilość faz, typ klasyfikatora a także wysokość i szerokość która powinna być zgodna z tą która została użyta do utworzenia pliku -vec, przykład użycia został przedstawiony na obrazku 3.3. Podczas każdego etapu, zostaje przygotowany plik który pozwala nam na przerwanie szkolenia i przywrócenia go na takim etapie na którym został ukończony.

```

>opencv_traincascade.exe -data path_to_classifier/ -vec file.vec
-bg neg.info -numPos 550 -numNeg 500 -w 50 -h 20 -numStages 1
PARAMETERS:
cascadeDirName: path_to_classifier/
vecFileName: file.vec
bgFileName: neg.info
numPos: 550
numNeg: 500
numStages: 1
precalcValBufSize[Mb] : 1024
precalcIdxBufSize[Mb] : 1024
acceptanceRatioBreakValue : -1
stageType: BOOST
featureType: HAAR
sampleWidth: 50
sampleHeight: 20
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: BASIC
Number of unique features given windowSize [50,20] : 487255

===== TRAINING 0-stage =====
<BEGIN
POS count : consumed    550 : 550
NEG count : acceptanceRatio    500 : 1
Precalculation time: 8.769
+---+---+---+---+
| N |   HR |   FA |
+---+---+---+---+
| 1 |     1 |     1 |
+---+---+---+---+
| 2 |     1 |     1 |
+---+---+---+---+
| 3 |     1 |     1 |
+---+---+---+---+
| 4 | 0.998182 | 0.548 |
+---+---+---+---+
| 5 | 0.996364 | 0.376 |
+---+---+---+---+
END>
Training until now has taken 0 days 0 hours 0 minutes 51 seconds.

```

Rys. 3.3. Przykład użycia narzędzia do trenowania klasyfikatora

3.1.4. Visualisation (ang. Narzędzie do wizualizacji)

Kiedy posiadamy wytrenowany klasyfikator, jednak chcielibyśmy sprawdzić jakie cechy wybrał, możemy użyć właśnie tego programu. Jako parametry należy podać obraz, na który zostaną naniesione cechy, ścieżkę do wytrenowanego klasyfikatora a także gdzie chcielibyśmy zapisać zdjęcie z naniesionymi cechami. Przykład dla klasyfikatora HAAR'a został przedstawiony na obrazku 3.4.



Rys. 3.4. Przykład pokazania cech za pomocą narzędzia do wizualizacji

// Spytać się o samo zaznaczanie obiektów na obrazie przy użyciu klasyfikatora

4. Haar-like

Opracowana w 2001 roku przez Paul'a Viola i Michael Jones'a

X. Wykaz Literatury

1. <https://pdfs.semanticscholar.org/d486/9863b5da0fa4ff5707fa972c6e1dc92474f6.pdf>
2. <https://opencv.org/about/>
3. <https://pdfs.semanticscholar.org/40b1/0e330a5511a6a45f42c8b86da222504c717f.pdf>
4. https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html

XI Spis rysunków

- 3.1. Przykładowe użycie programu annotation
- 3.2.1. Przykład utworzenia próbek za pomocą narzędzia createsample
- 3.2.2. Wyświetlanie próbek za pomocą narzędzia createsample
- 3.3. Przykład użycia narzędzia do trenowania klasyfikatora
- 3.4. Przykład pokazania cech za pomocą narzędzia do wizualizacji