Projekt Podstawy Teleinformatyki
Rozpoznawanie twarzy i śledzenie ruchu

Maciej Marciniak Damian Filipowicz

7 kwietnia 2017

Spis treści

Wstęp

Systemem rozpoznawania twarzy i śledzenia ruchu nazywamy komputer obsługujący kamerę cyfrową oraz program analizujący wykonane zdjęcie. Identyfikacja osoby odbywa się przez odnalezienie na obrazie charakterystycznych cech oraz porównanie ich z klasyfikatorami znajdującymi się w bazie danych.

Śledzenie twarzy jest częścią mechanizmu rozpoznawania osób. Wśród wielu obiektów program wykrywa kontury twarzy po wcześniejszym wyuczeniu algorytmu opartego o zbiór obrazów testowych. System posiadać będzie również dodatkową funkcjonalność do zliczania osób, a dokładniej twarzy, znajdujących się w danej chwili w obiektywie kamery.

Projekt składać się będzie z 3 podstawowych elementów:

- bazy danych MySQL,
- mikrokomputera Raspberry Pi 3,
- dedykowanej kamery Raspberry Pi 5Mpix.

1 Dlaczego rozpoznawanie twarzy

Projekt realizowany jest w ramach przedmiotu Podstawy Teleinformatyki. Wybrano temat „Rozpoznawanie twarzy i śledzenie ruchu" z wielu różnych możliwości, ponieważ jest to możliwość poznania problematyki, która przyda się nam w przygotowaniu się do tworzenia pracy inżynierskiej. Identyfikacja osób jest formą zabezpieczeń biometryczny, która jest ściśle związana z dziedziną bezpieczeństwa systemów informatyczny, jednocześnie z wybraną przez nas specjalizacją kierunku studiów.

2 Wymagania

2.1 Wymagania funkcjonalne

Podstawowymi wymaganiami jakie ma spełniać projekt są:

- śledzenie ruchu na podstawie wykrywania konturu twarzy,
- zliczanie liczby osób znajdującej się w obiektywie,
- rozpoznawanie twarzy oraz przypisanie danej osoby do pasujący zdjęć znajdujących się w bazie danych,
- automatyczne uczenie się rozpoznawania nowych osób.

2.2 Wymagania pozafunkcjonalne

Zakłada się następujące wymagania pozafunkcjonalne systemu:

- szerokość widzenia obiektywu to 70 stopni,
- ograniczona pamięć bazy danych do 32GB (pojemność karty pamięci),
- oprogramowanie zgodne z urządzeniem Raspberry Pi,
- wymagany interpreter języka Python w wersji 2.7.

2.3 Wymagania sprzętowe

Niezbędne, minimalne wymagania do uruchomienia systemu to:

- mikrokomputer Raspberry Pi 3,
- system operacyjny rasbian-jessie dla Raspberry Pi,
- dowolna dedykowana kamera Raspberry Pi,
- zasilacz micro USB 5V co najmniej 2A,
- karta pamięci micro SD minimum 32Gb klasy 10.

2.4 Środowisko pracy

- język programowania Python 2.7,
- Linux rasbian-jessie,
- IDE PyCharm,
- Latex.

3 Organizacja pracy

Link do repozytorium GitHub: Rozpoznawanie twarzy i śledzenie ruchu

3.1 Podział pracy

Zadania Damiana Filipowicza:

- zapoznanie się z algorytmem EigenFace,
- implementacja rozpoznawania twarzy na obrazie,
- utworzenie bazy danych zawierającej osoby do rozpoznania,
- przygotowanie korpusu zdjęć do wytrenowania klasyfikatora.

Zadania Macieja Marciniaka:

- zapoznanie się z algorytmem FisherFace,
- prowadzenie dokumentacji projektu,
- implementacja wykrywania i zliczania twarzy na obrazie,
- implementacja mechanizmu rozbudowy korpusu trenującego klasyfikator.

3.2 Harmonogram pracy

Przygotowano orientacyjny harmonogram pracy rozłożony na wszystkie zajęcia projektowe. Wyszczególniono zadania jak również osobę/osoby zajmujące się danym fragmentem. Zobrazowano harmonogram na Rysunku 1.

Zadanie	Kto	Zajęcia					
		24.03.2017	7.04.2017	21.04.2017	5.05.2017	19.05.2017	2.06.2017
Utworzenie repozytorium	Damian, Maciej						
Wybór środowiska pracy oraz sprzętu	Damian, Maciej						
Zapoznanie się z algorytmem EigenFace	Damian						
Zapoznanie się z algorytmem FischerFace	Maciej						
Wybór algorytmy do rozpoznawania twarzy	Damian, Maciej						
Implementacja wykrycia i zliczenia twarzy na obrazie	Maciej						
Implementacja rozpoznawania twarzy na obrazie	Damian						
Utworzenie bazy danych osób do rozpoznania	Damian						
Przygotowanie korpusu zdjęć do wytrenowania klasyfikatora	Damian, Maciej						
Implementacja mechanizmu trenującego klasyfikator o rozpoznawanie nowych osób	Maciej						
Prowadzenie dokumentacji projektu	Maciej						

Rysunek 1: Harmonogram prac

4 Algorytmy rozpoznawania twarzy

4.1 EigenFace

Twarze własne (eigenfaces) to zbiór wektorów własnych używany przy komputerowym rozpoznawaniu twarzy. Twarze własne są uważane za pierwszą skuteczną technologię rozpoznawania twarzy. Wektory własne są wyprowadzone z macierzy kowariancji rozkładu prawdopodobieństwa wysoko – wymiarowej przestrzeni wektorowej możliwych ludzkich twarzy. Koncepcja eigenface opiera się na wektorach własnych. Zbiór twarzy własnych może zostać wygenerowany przez wykonanie PCA na dużym zbiorze obrazów ludzkiej twarzy.

Obrazy składające się na „treningowy zbiór obrazów" powinny być zrobione w takich samych warunkach oświetleniowych i muszą być znormalizowane w taki sposób by oczy i usta na wszystkich zdjęciach znajdowały się na tych samych wysokościach. Wszystkie obrazy muszą też być w tej samej rozdzielczości. Natomiast same „Twarze własne" nie przechowują zdjęć tylko są listą cech przez co zajmują mało miejsca i pozwalają na wydajniejsze obliczenia.

4.2 FisherFace

Innym podejściem rozpoznawania twarzy niż EigenFace jest algorytm FisherFace. Jak mogliśmy zaobserwować w ramach poprzedniego algorytmu jego podstawowym celem było zmaksymalizowanie wartości wariancji w ramach naszej bazy danych (tak by próbki były znacznie różne od innych), natomiast w przypadku podejścia zaproponowanego w ramach algorytmu FisherFace za cel stawiane jest maksymalizacja średniego dystansu pomiędzy różnymi klasami oraz zminimalizowanie wariancji wewnątrzklasowej. Sposobem, który doprowadza do uzyskania takich warunków jest użycie liniowej metody znanej pod nazwą liniowej analizy dyskryminacyjnej (Fisher's Linear Discriminant - FLD). Obraz twarzy składają-

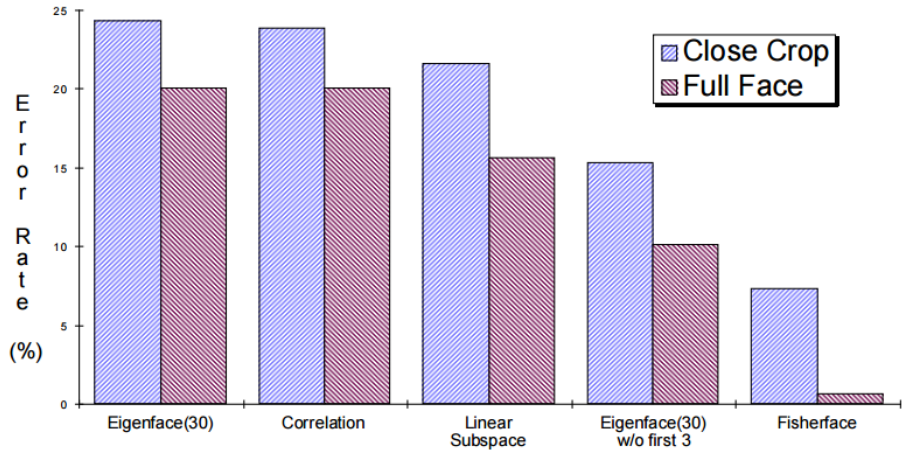
cy się z bardzo dużej liczby pikseli jest redukowany do mniejszego zbioru linowych kombinacji, które mogą następnie wykorzystane są do klasyfikacji.

Na algorytm FLD składa się 5 głównych etapów:

1. Oblicz średnią D-wymiarową wektorów dla różnych klas z zestawu danych.
2. Oblicz macierze rozproszone (matryca rozproszona w klasach i wewnątrz klasy).
3. Obliczyć wektory własne (eigenfaces) $(e_1, e_2, ..., e_d)$ i odpowiadające im wartości własne $(\lambda_1, \lambda_2, ..., \lambda_d)$ dla matryc rozproszonych.
4. Sortuj wektory własne, zmniejszając wartości własne i wybierz własne wektory **k** z największymi wartościami własnymi, aby utworzyć matrycę wymiarową **d** × **k** ***W*** (gdzie każda kolumna reprezentuje wektor osobisty).
5. Użyj tej matrycy własnej **d** ×**k**, aby przekształcić próbki w nową podprzestrzeń. Można to podsumować przez pomnożenie matrycy: **Y** =**X** ×**W** (gdzie **X** jest macierzą o wymiarach **n** ×**d** reprezentującą próbki **n**, a y są transformowanymi próbkami o wymiarach **n** ×**k** w nowej podprzestrzeni).

4.3 Wybór algorytmu

Podsumowując algorytm FisherFace obarczony jest mniejszym stopniem błędów niż EigenFace. Na Rysunku 2 przedstawiono wykres efektywności wybranych algorytmów, jak można zauważyć najbardziej zawodny jest EigenFace, zaś bezkonkurencyjny FisherFace. Najlepsza implementacja przy pełnym widoku twarzy ma poziom błędów rzędu 1% w porównaniu do blisko 25% EigenFace.



Rysunek 2: Wykres efektywności algorytmów rozpoznawania twarzy

Zakładając znacznie mniejsze zużycie pamięci i oszczędność obliczeń algorytmu EigenFace w porównaniu do FisherFace jest to algorytm mniej obciążający mikrokomputer Raspberry. Zależy jednak nam podczas realizacji projektu, by poziom błędów był jak najniższy, więc do rozpoznawaniu twarzy użyjemy algorytmu FisherFace.