

Projekt Podstawy Teleinformatyki
Rozpoznawanie twarzy i śledzenie ruchu

Maciej Marciniak

Damian Filipowicz

27 maja 2017

Spis treści

1	Dlaczego rozpoznawanie twarzy	5
2	Wymagania	6
2.1	Wymagania funkcjonalne	6
2.2	Wymagania pozafunkcjonalne	7
2.3	Wymagania sprzętowe	7
2.4	Środowisko pracy	7
3	Organizacja pracy	8
3.1	Podział pracy	8
3.2	Harmonogram pracy	8
4	Algorytmy rozpoznawania twarzy	10
4.1	EigenFace	10
4.2	FisherFace	11
4.3	LBPH	11
4.4	Wybór algorytmu	13
5	Biblioteka OpenCV	14
6	Architektura rozwiązania	17
7	Napotkane problemy	18
8	Przygotowanie środowiska pracy systemu	20

9	Użytkowanie systemu	21
9.1	Rozpoznawanie i zliczanie twarzy.py	21
9.2	Trenowanie klasyfikatora.py	24
10	Możliwe zastosowania systemu	28
11	Perspektywy rozwoju systemu	29

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.

Rozdział 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 praktyczna okazja poznania problematyki zabezpieczeń systemów informatycznych, która przyda nam się w przygotowaniach do tworzenia pracy inżynierskiej. Identyfikacja osób jest formą zabezpieczeń biometryczny, która jest ściśle związana z dziedziną bezpieczeństwa systemów informatyczny. Znajdujemy się na specjalizacji właśnie związanej z bezpieczeństwem, zatem wybrany przez nas temat projektu jest ściśle związany z problematyką jaką zajmuje się nasz kierunek studiów.

W przyszłości implementację projektu wykorzystać będziemy mogli podczas tworzenia pracy inżynierskiej.

Rozdział 2

Wymagania

2.1 Wymagania funkcjonalne

Wymagania funkcjonalne systemu zebrane zostały w Tabeli 2.1.

Tabela 2.1: Tabela wymagań funkcjonalnych systemu

Funkcja	Opis
Śledzenie ruchu	Dynamiczne zaznaczanie twarzy na klatce pobranej na żywo z kamery
Zliczanie liczby osoby	Podanie liczby osób znajdujących się w danych chwili w obiektywie kamery
Rozpoznawanie twarzy	Na podstawie wykrytych twarzy rozpoznanie osoby wg etykiety przypisanej podczas nauki klasyfikatora
Przypisanie danych osobowych do zdjęcia	Przypisanie imienia i nazwiska osoby na podstawie wykrytej etykiety i pobrania odpowiedniej pozycji z bazy danych
Dodawanie obrazów do bazy zdjęć z dysku	Możliwość dołączenia nowych zdjęć do istniejącej bazy zdjęć z folderu znajdującego się w pamięci komputera
Dodawanie obrazów do bazy zdjęć na żywo z kamery	Możliwość dołączenia nowych zdjęć do istniejącej bazy zdjęć bezpośrednio z kamery
Trenowanie klasyfikatora	Wykonanie pliku klasyfikatora na podstawie bazy zdjęć

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,
- ograniczenie liczby zdjęć w klasyfikatorze, plik klasyfikatora nie może przekraczać 300 MB,
- 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,
- pamięć karty graficznej ustawiona minimum na 256 Mb,
- 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,
- TeXStudio.

Rozdział 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 oraz LBPH,
- 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 3.1.

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 algorytmu 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 3.1: Harmonogram prac

Rozdział 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 składający 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, \dots, e_d) i odpowiadające im wartości własne $(\lambda_1, \lambda_2, \dots, \lambda_d)$ dla matryc rozproszonych.
4. Sortuj wektory własne, zmniejszając wartości własne i wybierz własne wektory \mathbf{k} z największymi wartościami własnymi, aby utworzyć macierz wymiarową $\mathbf{d} \times \mathbf{k}$ \mathbf{W} (gdzie każda kolumna reprezentuje wektor osobisty).
5. Użyj tej macierzy własnej $\mathbf{d} \times \mathbf{k}$, aby przekształcić próbki w nową podprzestrzeń. Można to podsumować przez pomnożenie macierzy: $\mathbf{Y} = \mathbf{X} \times \mathbf{W}$ (gdzie \mathbf{X} jest macierzą o wymiarach $\mathbf{n} \times \mathbf{d}$ reprezentującą próbki \mathbf{n} , a \mathbf{y} są transformowanymi próbkami o wymiarach $\mathbf{n} \times \mathbf{k}$ w nowej podprzestrzeni).

4.3 LBPH

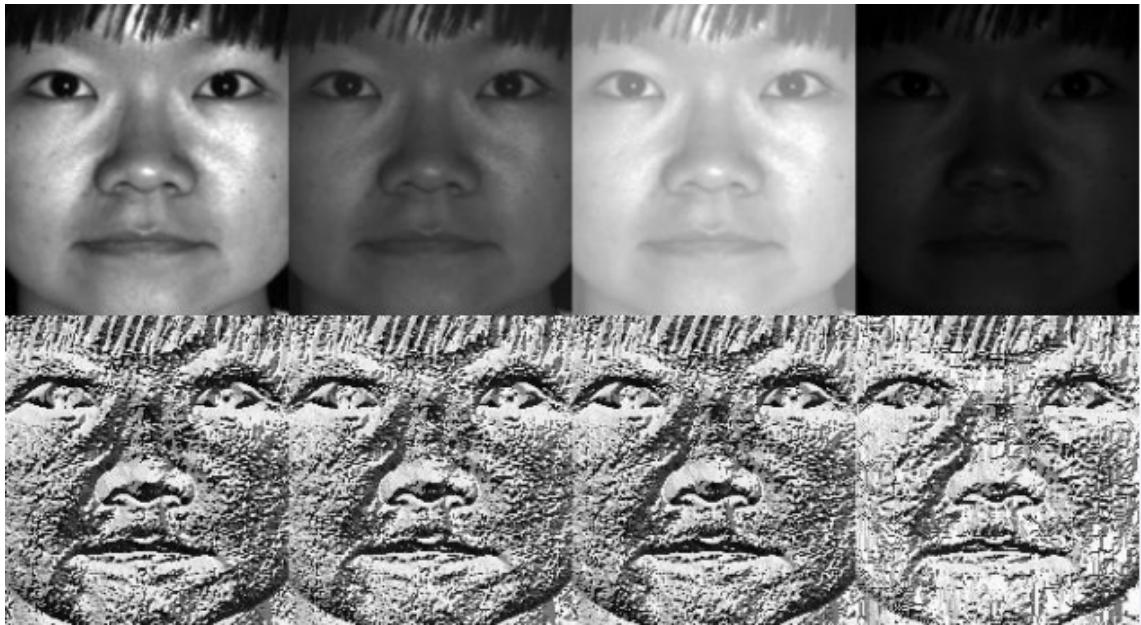
Local Pattern Binary Histogram jest metodą pozwalającą na zmianę zdjęcia w macierz wartości całkowitych opisujących mało-wymiarowe cechy obrazu. Histogram tych wartości jest następnie wykorzystywany do dalszej analizy.

LBPH nie bazuje na statystyce, więc nie jest potrzebna duża ilość zdjęć uczących. Dodatkową ważną cechą tego algorytmu jest możliwość douczenia klasyfikatora już na istniejącym pliku. Poprzednie dwie techniki rozpoznawania twarzy wyma-

gały ponownego treningu przy każdorazowej chęci zwiększenia liczby zdjęć danej osoby, czy dodanie nowej twarzy do systemu.

Główną ideą algorytmu jest sumowanie lokalnej struktury zdjęcia poprzez porównywanie każdego piksela z jego sąsiedztwem. Wybierany jest środkowy piksel kwadratowego obszaru (w wersji podstawowej 3x3 pikseli), a jego sąsiedztwo jest poddawane progowaniu. W zależności od tego, czy dany sąsiadujący piksel jest większy od progu, czy nie, przyjmuje wartość 1 lub 0. Następnie odczytuje się liczbę binarną zapisaną dookoła środkowego piksela. Dla ośmiu pikseli sąsiadujących istnieje 256 kombinacji, zwanych Local Binary Patterns.

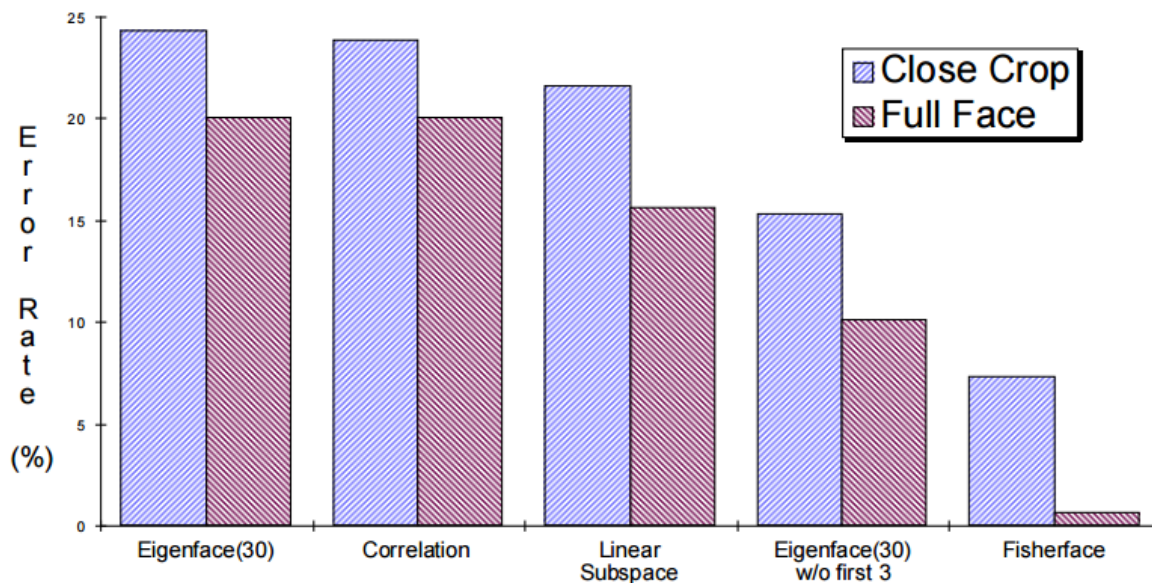
Algorytm bardzo dobrze sprawdza się przy zmiennym oświetleniu, można zauważyć na Rysunku 4.1 jak zmienia się obraz wynikowy przy różnym natężeniu światła.



Rysunek 4.1: Działanie algorytmu LBPH przy zmiennym oświetleniu

4.4 Wybór algorytmu

Podsumowując algorytm FisherFace obarczony jest mniejszym stopniem błędów niż EigenFace. Na Rysunku 4.2 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 4.2: Wykres efektywności algorytmów rozpoznawania twarzy

Algorytm EigenFace mimo stosunkowo niewielkich wymagań obliczeniowych posiada bardzo słabe wyniki przy rozpoznawaniu twarzy. Zależy nam jednak, aby system był w miarę możliwości niezawodny stąd metoda zostaje odrzucona z branych pod uwagę. Podczas testów własnych algorytm FisherFace okazał się mniej skuteczny niż metoda LBPH. Fisherface posiada jedną główną wadę w porównaniu do LBPH, jest nią brak możliwości doszkalania klasyfikatora. Z punktu widzenia systemu posiadającego osoby często przybywające do bazy jest to bardzo niewygodne i problematyczne.

Podsumowując, metoda Local Pattern Binary Histogram, w skrócie LBPH posiada najwięcej zalet, pomimo dużych problemów z rozpoznawaniem twarzy znajdującej się bokiem.

Rozdział 5

Biblioteka OpenCV

Biblioteka OpenCV jest bardzo rozbudowana i posiadająca wiele zastosowań, podczas realizacji projektu użyto zaledwie kilku metod do obróbki zdjęć oraz samego rozpoznawania twarzy. Wykorzystano bibliotekę w wersji 2.4.9, ponieważ dopiero od wersji 2.4 pojawiły się funkcje implementujące algorytmy FisherFace oraz LBPH. Nowsze wersje OpenCV są problematyczne w instalacji w języku Python 2.7 dla systemu Linux rasbian-jessie.

Metody biblioteki OpenCV użyte do realizacji projektu:

- `createLBPHFaceRecognizer(radius=1, neighbors=8, grid_x=8, grid_y=8, threshold=DBL_MAX)` —
Tworzy obiekt służący do rozpoznawania twarzy według algorytmu Local Pattern Binary Histogram. Parametr `radius`, domyślnie ustawiony na 1, oznacza długość promienia branego pod uwagę przy liczeniu macierzy sąsiedztwa. `Neighbors` (domyślnie 8) decyduje o, można tak powiedzieć, rozdzielczości wykonywanych obliczeń. Wartość ustawiona na 8 daje 256 różnych kombinacji ustawienia progowania. `Grid_x` i `Grid_y` wyznacza liczbę komórek odpowiednio na szerokość i wysokość długości od danej komórki branej pod uwagę w obliczeniach. `Threshold=DBL_MAX` jest progiem odległości do jakiej rozpoznawanie twarzy daje wynik, po przekroczeniu progu zwracana jest wartość -1 zamiast etykiety zdjęcia.
- `CascadeClassifier("haarcascade_frontalface_alt.xml")` —
Zwraca klasyfikator służący do wykrywania twarzy. Klasyfikator wgrywany jest z pliku podanego w argumencie, jest to plik XML.
- `cvtColor(frame, cv2.COLOR_BGR2GRAY)` —

Zwraca zmodyfikowany obraz, podany jako pierwszy parametr, w skali szarości.

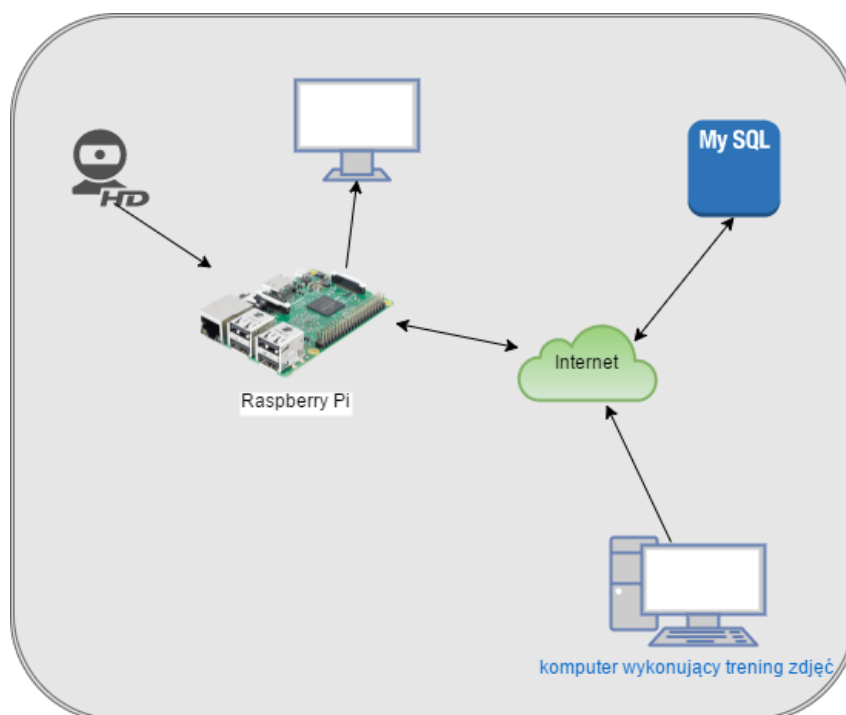
- `face_cascade.detectMultiScale(gray, 1.3, 8)` —
Funkcja wykrywająca twarz na podstawie wgranego klasyfikatora. Pierwszym parametrem jest zdjęcie, najlepiej w skali szarości, kolejne argumenty odpowiadają za strojenie.
- `recognizer.train(images, np.array(labels))` —
Obiekt `recognizer` służy do rozpoznawania twarzy, metoda wykonana na tym obiekcie ma na celu wytrenowanie klasyfikatora, aby potrafił rozpoznać twarze umieszczone w tablicy zdjęć (`images`) oraz przypisać do nich odpowiednie etykiety (`labels`).
- `recognizer.load(file)` —
Metoda `load` wykonana na obiekcie `recognizer` z argumentem w postaci pliku służy do wczytania wytrenowanego wcześniej klasyfikatora z zdjęciami do rozpoznania.
- `recognizer.update(images, np.array(labels))` —
Funkcja `update` ma podobne działania jak `train`, w przeciwieństwie do niej nie tworzy nowego klasyfikatora, lecz modyfikuje wcześniej wgrany. Metoda jest dostępna tylko dla recognizera utworzonego dla algorytmu LBPH.
- `recognizer.save(plik)` —
Metoda `save` wykonana na obiekcie `recognizer` z argumentem w postaci pliku służy do zapisania wytrenowanego wcześniej klasyfikatora z zdjęciami do rozpoznania.
- `putText(frame, text, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (b, g, r), 2)` —
Funkcja pozwalająca zmodyfikować obraz `frame` w taki sposób, aby dodać do niego tekst `text` w miejscu o współrzędnych `x,y` oraz w kolorze ustalonym według schematu BRG.
- `rectangle(frame, (x, y), (x + w, y + h), (b, g, r), 2)` —
Funkcja pozwalająca zmodyfikować obraz `frame` w taki sposób, aby dodać do niego ramkę w miejscu o współrzędnych `x,y` i długości boków `w` i `h` oraz w kolorze ustalonym według schematu BRG.

- `imshow(name, frame)` —
Imshow służy do wyświetlenia obrazu `frame` w oknie nazwanym `name`.
- `destroyAllWindows()` —
Funkcja wykonuje dokładnie to na co wskazuje jej nazwa, to znaczy zamyka (niszczy) wszystkie otwarte okna.

Rozdział 6

Architektura rozwiązania

Proponowane rozwiązanie polega na uruchomieniu programu śledzącego ruch i rozpoznającego twarze na urządzeniu Raspberry Pi oraz wgraniu wytrenowanego pliku z bazą zdjęć. Specjalny plik, do wczytania, tworzy się najlepiej na komputerze o większej mocy obliczeniowej, na przykład komputerze stacjonarnym, a następnie przesyła zdalnie do Raspberry Pi (posłużyć może w tym celu program WinSCP). Schemat połączeń elementów systemu znajduje się na Rysunku 6.1.



Rysunek 6.1: Ogólny schemat systemu

Rozdział 7

Napotkane problemy

Podczas implementacji systemu głównym problemem była niska wydajność urządzenia Raspberry Pi. Pomimo procesora posiadającego cztery rdzenie, każdy z nich o taktowaniu 1.2GHz, to program wykonuje się wolno.

Skrypt Pythona podzielony został pomiędzy wątki. Jeden wątek pobierał aktualne klatki z kamery. Drugi wykonywał wyszukiwanie twarzy na skopiowanym i zmodyfikowanym zdjęciu. Modyfikacja polega na sprowadzeniu obrazu do skali szarości. Trzeci wątek, główny wykonuje rozpoznawanie twarzy oraz przygotowuje wynikowy obraz do wydrukowania na ekranie.

Pomimo takiego rozłożenia obciążenia wciąż operacje wykrywania i rozpoznawania twarzy zajmują dużo czasu, około 5 sekund. Problem związany może być z ograniczeniami karty graficznej oraz pamięci ram urządzenia. Rozwiązaniem jest utworzenie dodatkowych wątków operujących na każdej twarzy osobno, powoduje niestety takie rozwiązanie bardzo częste miganie ramek, przez co obraz jest nieczytelny. Problem z obrazem jest również odpowiednie dobranie kolorów tekstu i ramek.

Inną napotkaną trudnością jest wielkość pliku z klasyfikatorem rozpoznającym twarze. Chcąc uzyskać dokładniejsze wyniki rozpoznawania twarzy algorytmem LBPH należało zwiększyć parametr neighbors z 8 do co najmniej 10, skutkuje to znacznym wzrostem pamięci jaką zajmuje plik. Zakładając parametr ustawiony na wartość 8, dokument zajmuje około 50 MB przy wczytanych 260 zdjęciach, zmieniając wartość na 10 rozmiar pliku wzrasta do 107 MB. Jest to zagrożenie, gdy dodając nowe zdjęcia osób przekroczymy dostępną pamięć ram Raspberry Pi. Do dyspozycji mamy 1 GB RAMu, ale należy odjąć od tej wartości pamięć współdzieloną dla karty graficznej (około 400 MB) plus pamięć zarezerwowaną na system operacyjny.

Problem również jest z tworzeniem klasyfikatora. Jest to operacja wymagająca większej mocy obliczeniowej urządzenia. Z tego powodu przeniesiono tworzenie klasyfikatora na osobny komputer posiadający większe możliwości. Kod programu trenującego jest przeznaczony również do pracy na Raspberry, lecz nie jest to zalecane, ponieważ istnieje duże prawdopodobieństwo zapełnienia pamięci RAM co skutkuje wyłączeniem lub zawieszeniem się urządzenia.

Kolejnym problemem jest kąt ułożenia twarzy względem obiektywu jaki algorytm jest w stanie poprawnie rozpoznać. Metoda LBPH skuteczna jest w dużym stopniu dla twarzy znajdujących się na wprost kamery. Niewielkie odchylenie głowy w bok lub w górę powoduje w skrajnym przypadku błędne rozpoznanie.

Ostatnim z problemów wartych uwagi jest trudność z wyznaczeniem progu przy którym twarz rozpoznawana jest prawidłowo, a od którego wyższe wartości są już nie przypisywane. Kłopot polega przede wszystkim w tym, że tak zwane odległości od wzorca są do siebie bardzo zbliżone.

Rozdział 8

Przygotowanie środowiska pracy systemu

System utworzony jest w języku Python 2.7, więc aby uruchomić program należy posiadać odpowiednią wersję interpretera. Jeśli na urządzeniu znajdują co najmniej dwie instancje interpretera to należy zwrócić uwagę, na to która uruchamia się domyślnie. Program można uruchomić również z konsoli poleceniem *python /nazwa_skryptu/ /parametry/*.

Przed uruchomieniem systemu Rozpoznawania i zliczania twarzy wymagane jest posiadanie zainstalowanych w systemie odpowiednich bibliotek. Dla programu „Rozpoznawanie i zliczanie twarzy” należy pobrać i zainstalować moduły odpowiedzialne za obsługę bazy danych oraz obróbkę graficzną. Są to odpowiednio biblioteka MySQLdb i OpenCV, pobrać oraz jednocześnie zainstalować można je za pomocą polecenia konsolowego *python -m pip install Python-MySQLdb Python-OpenCV* według schematu *python -m pip install /nazwa_biblioteki1/ /nazwa_biblioteki2/*. Program trenujący klasyfikator wymaga dodatkowo bibliotek Numpy, Pillow i Bar. W przypadku problemów z działaniem lub samą instalacją należy pobrać odpowiednie wersje modułów z Internetu (zwrócić uwagę na wersję interpretera dla Pythona 2.7 oraz wersję architektury systemu).

Sprawdzenie poprawności działania modułów przeprowadzić można przy pomocy uruchomienia z poziomu konsoli systemowej interpretera pythona (polecenie *python* bez żadnych parametrów), a następnie spróbować zaimportować biblioteki do programu poleceniem *import*. Dla powyższych modułów będzie to odpowiednio *import MySQLdb* oraz *import cv2*. Każde polecenie należy zakończyć klawiszem Enter. Poprawne działanie importowania bibliotek zakończy się bez żadnego błędu ani komunikatu. Wyjście z interpretera odbywa się za pomocą polecenia *exit()*.

Rozdział 9

Użytkowanie systemu

Po przygotowaniu i sprawdzeniu poprawności zainstalowanych bibliotek oraz weryfikacji wersji interpretera python można przejść do uruchomienia właściwych programów.

9.1 Rozpoznawanie i zliczanie twarzy.py

Tak jak zostało przedstawione w rozdziale wcześniej, program „Rozpoznawanie i zliczanie twarzy” uruchamia się poleceniem *python Rozpoznawanie i zliczanie twarzy* w wersji Linuksowej (raspbian) lub *python "Rozpoznawanie i zliczanie twarzy"*. Forma wywołania polecenia wymaga specjalnej formy, ponieważ znajdują się w niej spacje pomiędzy wyrazami. Program posiada parametry, domyślnie przy niepodaniu żadnego jest to użycie pliku klasyfikatora o nazwie „wytrenowany_plik.mdl” oraz użycie kamery dedykowanej dla Raspberry. Użycie innego zdefiniowanego klasyfikatora wymaga podania po nazwie programu ścieżki do pliku z danym klasyfikatorem. Program zaimplementowany jest również tak, aby mógł działać na urządzeniach innych od Raspberry, w tym wypadku należy w podać zawsze nazwę pliku klasyfikatora (nawet jeśli użyty zostanie domyślny) oraz podać po tym parametrze słowo „camera”. Zostanie w tym wypadku uruchomiona domyślna kamera ustawiona w systemie.

Po wywołaniu programu w konsoli pojawi się prośba o podanie adresu IP pod którym nasłuchuje bazy danych. Program nie posiada zabezpieczenia przed niepoprawnym wpisaniem adresu. Podanie błędnego IP wpłynie na dalsze działanie programu. Chcąc poprawić adres IP należy uruchomić program od początku. Następnym krokiem jest wczytanie klasyfikatora rozpoznającego twarz z pliku. Operacja ta może potrwać dłuższą chwilę w zależności od wielkości pliku jak i parametrów pamięci ram urządzenia. Rysunek 9.1 i Rysunek 9.2 przedstawiają

widok programu z poprawnie podanym adresem IP oraz w trakcie wczytywania klasyfikatora.

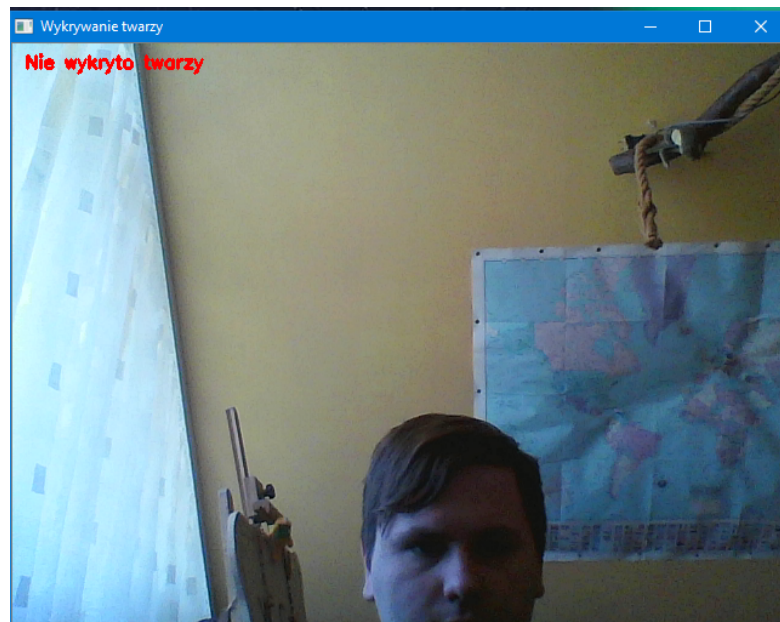
```
pi@raspberrypi:~/Desktop $ sudo python Rozpoznawanie\ i\ zliczanie\ twarzy.py
Podaj adres IP serwera: 192.168.137.172
Wgrywanie klasyfikatora...
```

Rysunek 9.1: Podanie adresu IP bazy danych (Raspberry)

```
python "Rozpoznawanie i zliczanie twarzy.py" wytrénowany_plik.mdl camera
Podaj adres IP serwera: 192.168.137.172
Wgrywanie klasyfikatora...
```

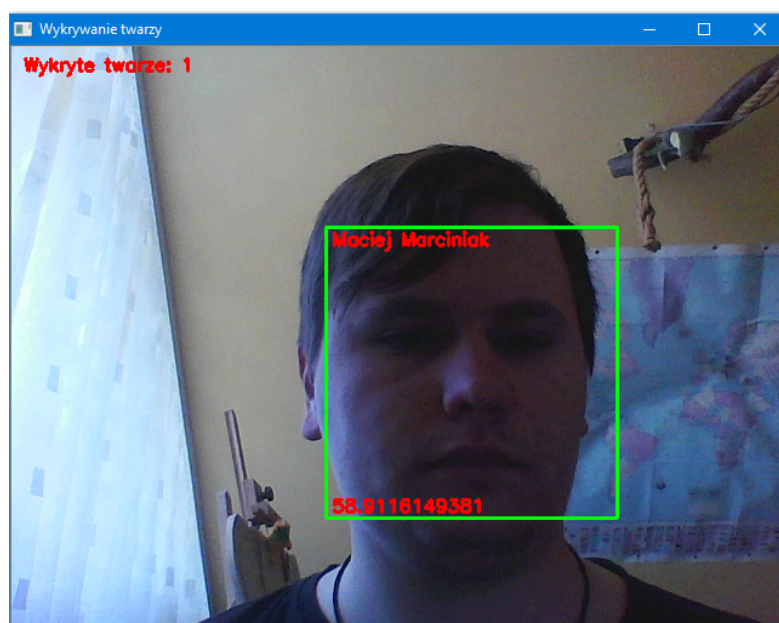
Rysunek 9.2: Podanie adresu IP bazy danych (Windows)

Po wgraniu klasyfikatora pojawi się nowe okno wyświetlające aktualne klatki z kamery, przedstawione na Rysunku 9.3. Niestety okno nie jest domyślnie aktywne, należy na nie kliknąć, aby je aktywować. Jeśli w obiektywie kamery nie znajdzie się żadna osoba (twarz) lub twarz nie będzie widoczna w całości, to wyświetlony zostaje napis, że nie wykryto twarzy w górnym lewym narożniku (czerwony napis).



Rysunek 9.3: Widok okna programu nieznajdującego twarzy

Rysunek 9.4 przedstawia widok programu jeśli twarz zostanie wykryta oraz rozpoznana. Napis wcześniej mówiący, że nie znaleziono twarzy zmienia się na „Wykryte twarze: 1”. Oznacza to, że w obiektywie wykryto jedną twarz, wartość liczbowa zmienia się w zależności od liczby wykrytych osób. Na obrazie pojawia się dodatkowo zielona ramka w około twarzy. Jest to obszar, który program traktuje jako twarz. Wewnątrz ramki znajduje się w lewym górnym rogu znajdują się dane wykrytej osoby oraz na w dolnym rogu względna odległość od oryginalnego obrazu znajdującego się w klasyfikatorze. Dane wykrytej osoby mogą pojawić się w dwóch różnych postaciach. Jedną z form jest wartość liczbowa (etykiety przypisanej przez algorytm rozpoznawania twarzy) oznaczająca, że powstał problem z połączeniem do bazy danych lub że dana osoba znajduje się w pliku klasyfikatora, lecz nie dostała dodana do bazy danych osób. Drugą postacią, przedstawioną na Rysunku 9.4 jest wartość tekstowa pobrana z bazy danych na podstawie etykiety liczbowej przypisanej do danej twarzy.



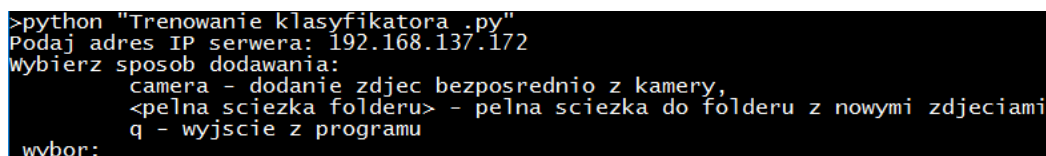
Rysunek 9.4: Widok okna programu znajdującego i rozpoznającego twarzy

Podczas pracy programu można aktualizować plik klasyfikatora, bez potrzeby ponownego uruchamiania aplikacji. Wgrywanie na nowo dokumentu odbywa się po naciśnięciu przycisku „n” w aktywnym oknie przedstawiającym aktualny widok z kamery. Spowoduje to zawieszenie wyświetlania obrazu na czas ładowania nowego pliku. Po zakończeniu wgrywania dokumentu program ponownie zacznie wyświetlać i rozpoznawać twarze lecz już na podstawie nowego, wgranego pliku.

9.2 Trenowanie klasyfikatora.py

Wcześniej opisy program nie może działać bez pliku klasyfikatora zdjęć. Ten właśnie dokument tworzony jest przy pomocy programu „Trenowanie klasyfikatora.py” uruchamianego poleceniem *python Trenowanie klasyfikatora.py* w wersji Linuksowej (rasbian) lub *python "Trenowanie klasyfikatora.py"* w wersji Windowsowej. Program nie posiada dodatkowych parametrów.

Po wywołaniu programu pojawia się widok konsoli. Tak jak w programie „Rozpoznawanie i zliczanie twarzy.py” również jesteśmy proszeni o podanie adresu IP serwera z bazą danych. Podanie błędnego adresu skutkuje niepodłączeniem do bazy danych, ale również wyłączeniem programu. Zatem ważne jest zwrócenie uwagi na poprawne wpisanie danych. Następnym krokiem jest wybór opcji pracy programu, widok aplikacji przedstawiający tę operację znajduje się na Rysunku 9.5.



```
>python "Trenowanie klasyfikatora .py"
Podaj adres IP serwera: 192.168.137.172
wybierz sposob dodawania:
    camera - dodanie zdjec bezposrednio z kamery,
    <pelna sciezka folderu> - pelna sciezka do folderu z nowymi zdjeciami
    q - wyjscie z programu
wybor:
```

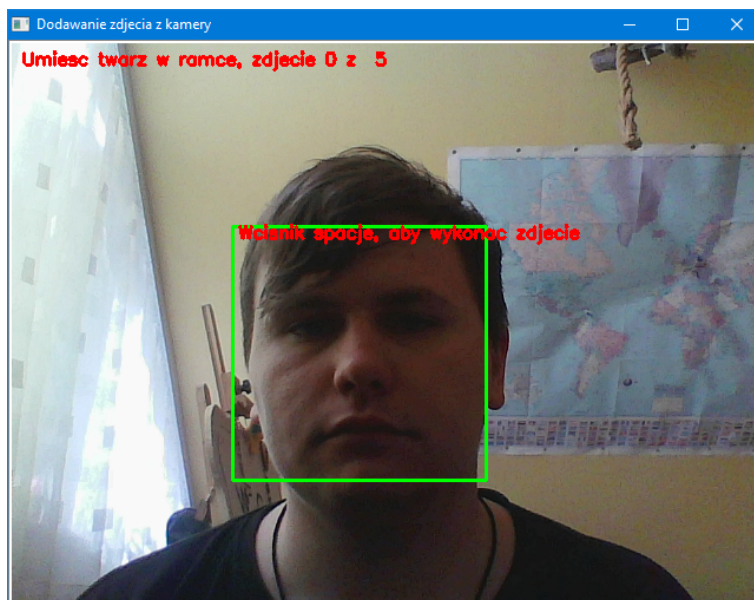
Rysunek 9.5: Widok okna programu trenującego — wybór opcji

Obok wymienionych opcji przedstawiono opis każdej z nich.

Dodawanie z kamery

Wybór dodawania za pomocą sposobu „camera” sprowadza się do uruchomienia kamery w urządzeniu oraz dodawania obrazów do bazy bezpośrednio na podstawie zdjęć robionych na żywo z kamery. Zanim jednak zostanie utworzone okno kamery zostaniemy poproszeni o podanie imienia i nazwiska osoby, której zdjęcia chcemy dodać. W przypadku braku podania którejś z danych zostaniemy poproszeni o podanie jednak wszystkich informacji. Jeśli dane osoby nie znajdują się w bazie danych osób, to zostaną automatycznie dodane, wraz z komunikatem o tym informującym oraz zostanie przypisana nowa etykieta zdjęcia, którą posłuży do dalszych operacji. Jeśli jednak osoba znajduje się w bazie danych to bez żadnych komunikatów przejdziemy do następnego kroku, czyli zostanie otwarte okno z widokiem z kamery. Okno interfejsu dodawania zdjęć z kamery przedstawione jest na Rysunku 9.6. Z racji, że istnieją różnej rozdzielczości kamery, dodawana jest biała ramka o rozmiarach 640 na 480 pikseli. Kamera przedstawiona na przykładzie ma dokładnie takie wymiary, zatem nie jest ramka dobrze widoczna. Twarz, aby

program próbował ją wykryć znajdować się musi wewnątrz białej ramki, informuje o tym czerwony napis w lewym, górnym rogu okna. W celach poprawienia jakości klasyfikatora, wykryta twarz na zdjęciu powinna mieć wymiary 250px \pm 50px.



Rysunek 9.6: Widok okna programu trenującego — okno dodawania z kamery

Jeśli warunek jest spełniony i zostanie wykryta twarz pojawi się ramka w okolicy twarzy z napisem wewnątrz, aby wcisnąć klawisz „spacja” w celu wykonania zdjęcia, jeśli jednak warunek nie zostanie spełniony zostaną wyświetlone odpowiednie komunikaty o tym, żeby się przybliżyć lub oddalić od obiektywu. Ponadto tym założeniem jest warunek, aby w obiektywie znajdowała się tylko jedna osoba, gdy znajdą się dwie lub więcej, program nie zareaguje na próbę wykonania zdjęcia i będzie komunikować o problemie.

Podczas procesu dodawania zdjęć założono, że dodawane jest 5 zdjęć danej osoby, większa liczba jest uciążliwa w pracy jak również nie zwiększa efektywności. Przy każdym poprawnie wykonanym zdjęciu zwiększany jest licznik u góry okna, informujący o tym ile zdjęć wykonano i ile zostało do wykonania. W każdej chwili można zrezygnować z procesu dodawania wciskając przycisk ESC. Po zakończeniu dodawania zdjęć lub anulowaniu procesu, okno kamery zostanie zamknięte oraz zostaniemy zapytani o to czy chcemy ponownie dodać zdjęcia danej osoby. Okno zapytania przedstawiono na Rysunku 9.7.

Należy w tym kroku zdecydować czy chcemy dodać więcej swoich zdjęć, podając odpowiedź „T” (akceptowane są też małe litery) lub jeśli chcemy przejść do następnego kroku, nie dodając nowych zdjęć podajemy odpowiedź „N”. Jeśli

```

>python "Trenowanie klasyfikatora .py"
Podaj adres IP serwera: 192.168.137.172
Wybierz sposob dodawania:
    camera - dodanie zdjec bezposrednio z kamery,
    <pelna sciezka folderu> - pelna sciezka do folderu z nowymi zdjeciami
    q - wyjscie z programu
wybor: camera
Podaj imie: Maciej
Podaj nazwisko: Marciniak
Czy chcesz ponownie dodac swoje zdjecia?: T/N
wybor:

```

Rysunek 9.7: Widok okna programu trenującego — zapytanie o ponowne dodawanie zdjęć

podana odpowiedź nie będzie prawidłowa będziemy pytani do skutku, aż zostanie podana odpowiedź poprawna. Następny krok odpowiada za decyzję, czy chcemy dodać nową osobę do bazy danych. Odpowiedzi są analogiczne do poprzedniego kroku. Na koniec pojawia się komunikat, że klasyfikator jest właśnie trenowany oraz pasek postępu prac, przedstawiony na Rysunku 9.8. Po zakończeniu trenowania pojawiają się komunikaty odpowiednio o zapisywaniu pliku klasyfikatora i zakończeniu treningu oraz status programu wraca do widoku wyboru opcji dodawania.

```

>python "Trenowanie klasyfikatora .py"
Podaj adres IP serwera: 192.168.137.172
Wybierz sposob dodawania:
    camera - dodanie zdjec bezposrednio z kamery,
    <pelna sciezka folderu> - pelna sciezka do folderu z nowymi zdjeciami
    q - wyjscie z programu
wybor: camera
Podaj imie: Maciej
Podaj nazwisko: Marciniak
Czy chcesz ponownie dodac swoje zdjecia?: T/N
wybor: n
Chcesz dodac kolejna osobe?: T/N
wybor: n
[K Wgrywanie bazy danych: |##### | 62/269

```

Rysunek 9.8: Widok okna programu trenującego — trenowanie

Dodawanie z folderu

Wybierając opcję drugą, czyli podając pełną ścieżkę do folderu z zdjęciami, zostaniemy również proszeni o podanie imienia i nazwiska jak to było przy wyborze dodania z kamery. W podanej ścieżce folderu znajdować powinny się zdjęcia tylko jednej osoby. Po zaakceptowaniu imienia i nazwiska automatycznie analizowane zostają zdjęcia umieszczone w folderze (brane są pod uwagę pliki tylko z rozszerzeniami .JPG oraz .PNG). Pojawia się pasek statusu analizy zdjęć. Widok procesu analizowania plików przedstawiono na Rysunku 9.9.

```

Podaj adres IP serwera: 192.168.137.172
Wybierz sposob dodawania:
    camera - dodanie zdjec bezposrednio z kamery,
    <pelna sciezka folderu> - pelna sciezka do folderu z nowymi zdjeciami
    q - wyjscie z programu
wybor: G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawanie twarzy\PT-Projekt-Rozpoznawanie-twarzy\baza_zdjec
Podaj imie: Maciej
Podaj nazwisko: Marciniak
[K251]
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 1/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 2/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 3/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 4/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 5/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 6/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 7/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 8/269
Wgrywanie folderu G:\Maciej\studia\Semestr VI\Podstawy Teleinformatyki\moje\Projekt rozpoznawani
e twarzy\PT-Projekt-Ro[Kznawanie-twarzy\baza_zdjec : | 9/269

```

Rysunek 9.9: Widok okna programu trenującego — analiza zdjęć z folderu

Po zakończeniu zadania, zostajemy zapytani czy chcemy dodać kolejną osobę, jest to analogiczne pytanie do wcześniej opisanego procesu. Po odpowiedzi przeczącej zostaje wykonane trenowanie klasyfikatora.

Ostatnia, trzecia opcja, czyli „q” powoduje wyłączenie programu.

Podczas dodawania zdjęć, niezależnie od metody dodawane zostają nowe pliki do folderu z dodanymi już zdjęciami o nazwie „baza_zdjec”. Każdy plik zostaje oryginalnie nazwany wg schematu:

/etykieta zdjecia/_/data dodania zdjecia (RRRR-MM-DD hh-mm-ss)/.JPG.

Rozdział 10

Możliwe zastosowania systemu

System odpowiednio zmodyfikowany mógłby posłużyć do zliczania osób wchodzących oraz wychodzących z pomieszczenia poprzez umieszczenie dwóch takich podsystemów komunikujących się po jednej i po drugiej stronie drzwi. Jeden system zliczałby osoby wchodzące, drugi zaś wychodzące.

Innym zastosowaniem mogłoby być zastosowanie rozpoznawania twarzy jako system dodatkowej weryfikacji osoby przy logowaniu do komputera. Samo rozpoznawanie twarzy łatwo oszukać poprzez podstawienie zdjęcia chociażby w telefonie do obiektywu kamery, dlatego rozwiązanie takie nie jest bezpiecznym jako jedyna metoda weryfikacji osoby.

System użyty może być również jako forma monitoringu w domu. Osoby upoważnione mogą poruszać się po mieszkaniu, nawet przy włączonym alarmie, nie aktywując go. Osoba nieznana pojawiająca się w obiektywie kamery wzbudzała by alarm.

Rozdział 11

Perspektywy rozwoju systemu

System na stan obecny posiada interfejs częściowo konsolowy. Dalszą perspektywą rozwoju projektu mogłoby być zastosowanie w pełni graficznego interfejsu. Usprawniłoby to funkcjonowanie użytkownika w systemie.

W przyszłości można zastosować bardziej wydajne urządzenie sterujące, pozwalające na efektywną pracę systemu.

Wnioski

Projekt przyczynił się do wzrostu naszej świadomości funkcjonowania systemów opartych o rozpoznawanie twarzy. Nauczyliśmy trzech metod rozpoznawania twarzy, z czego dwie zgłębiliśmy w większym stopniu (FisherFace oraz LBPH). Mieliśmy możliwość zapoznania się z bardzo zaawansowaną biblioteką jaką jest OpenCV, posiadającą wiele przydatnych metod do obróbki grafiki. Jest to dla nas cenne doświadczenie, które będziemy rozwijać w dalszej przyszłości.