Projekt Podstawy Teleinformatyki
Rozpoznawanie twarzy i śledzenie ruchu

Maciej Marciniak Damian Filipowicz

17 maja 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

Wymagania funkcjonalne systemu zebrane zostały w Tabeli 1.

Tabela 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.

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 LPBH,
- 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 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) $(e_1, e_2, ..., e_d)$ 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 $\mathbf{k}$ z największymi wartościami własnymi, aby utworzyć matrycę wymiarową $\mathbf{d} \times \mathbf{k}$ $\boldsymbol{W}$ (gdzie każda kolumna reprezentuje wektor osobisty).
5. Użyj tej matrycy własnej $\mathbf{d} \times \mathbf{k}$, aby przekształcić próbki w nową podprzestrzeń. Można to podsumować przez pomnożenie matrycy: $\mathbf{Y} = \mathbf{X} \times \mathbf{W}$ (gdzie $\mathbf{X}$ jest macierzą o wymiarach $\mathbf{n} \times \mathbf{d}$ reprezentującą próbki $\mathbf{n}$, a y są transformowanymi próbkami o wymiarach $\mathbf{n} \times \mathbf{k}$ w nowej podprzestrzeni).

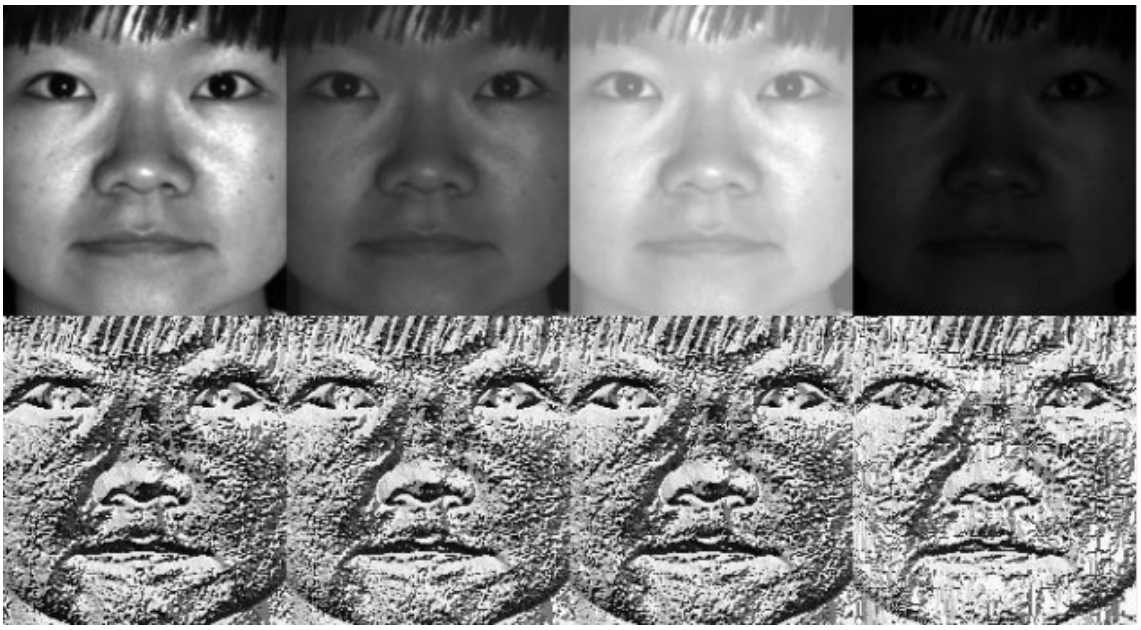
4.3 LPBH

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

LPBH 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 wymagały 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.

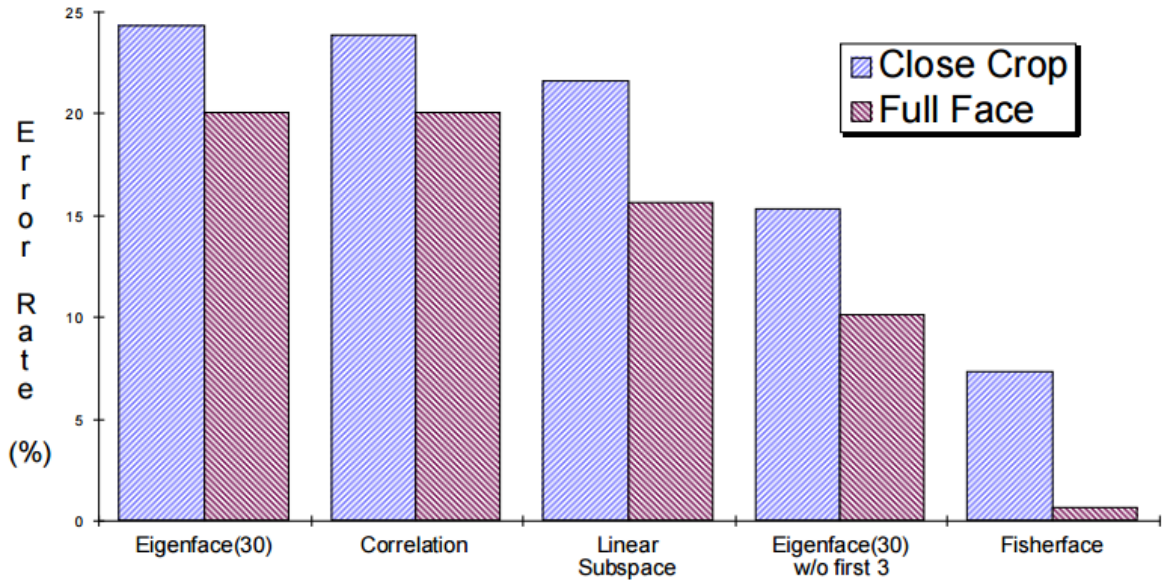
Algorytm bardzo dobrze sprawdza się przy zmiennym oświetleniu, można zauważyć na Rysunku 2 jak zmienia się obraz wynikowy przy różnym natężeniu światła.



Rysunek 2: Działanie algorytmu LPBH przy zmiennym oświetleniu

4.4 Wybór algorytmu

Podsumowując algorytm FisherFace obarczony jest mniejszym stopniem błędów niż EigenFace. Na Rysunku 3 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 3: 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 LPBH. Fisherface posiada jedną główną wadę w porównaniu do LPBH, jest nią brak możliwości doszkalania klasyfikatora. Z punktu widzenia systemu posiadającego osoby często przybywające do bazy jest to bardzo niewydajne i problematyczne.

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

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 LPBH. 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 rozpoznawnia 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 reconizer 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 LPBH.

- 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.

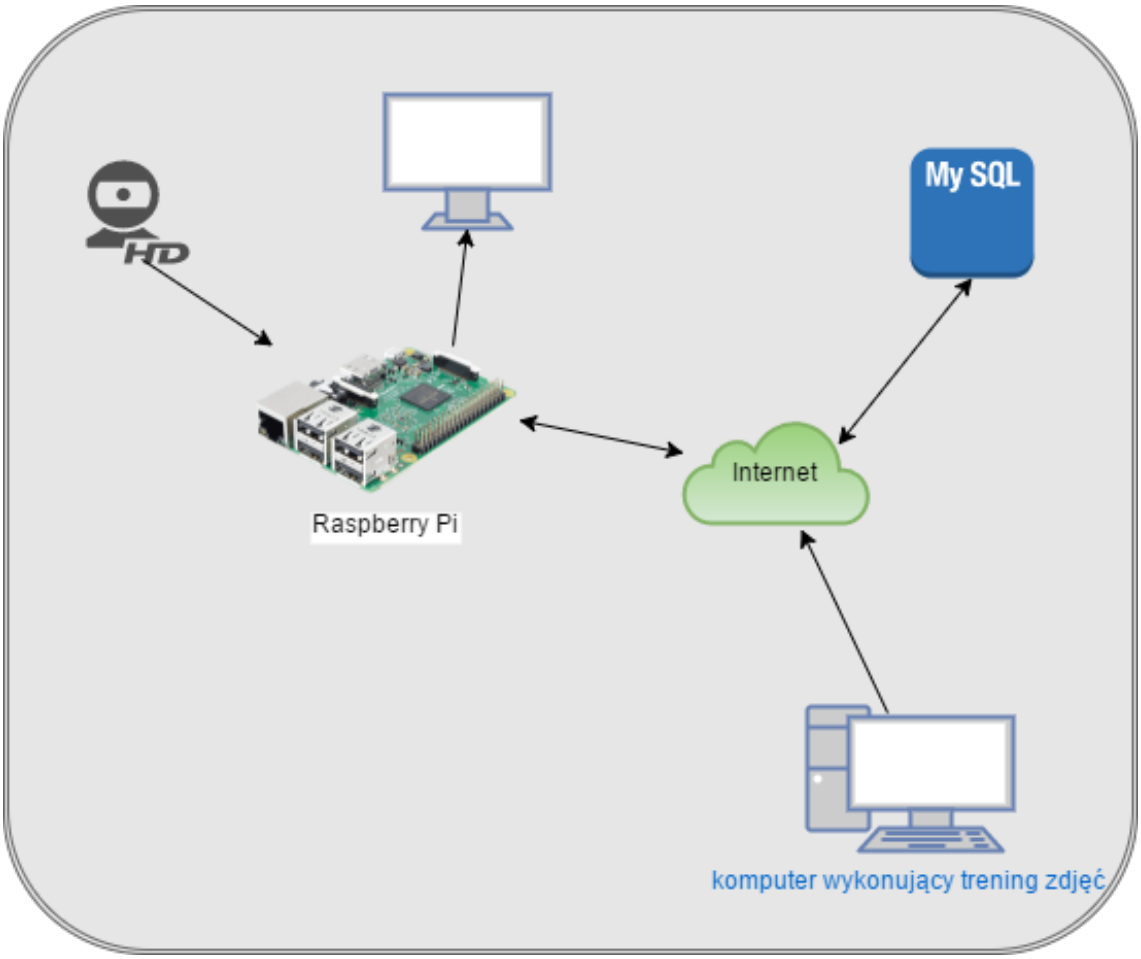
- rectangle(frame, (x, y), (x + w, y + h), (b, g, r), 2) —
  Funkcja pozwalająca zmodyfikować obraz frame w taki sposób, aby dodać do niego ramkę w miejscu o współrzędnych x,y i długości boków w i h oraz w kolorze ustalonym według schematu BRG.

- 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.

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 4.



Rysunek 4: Ogólny schemat systemu

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.

Inną napotkaną trudnością jest wielkość pliku z klasyfikatorem rozpoznającym twarze. Chcąc uzyskać dokładniejsze wyniki rozpoznawania twarzy algorytmem LPBH 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.

Kolejnym problemem jest kąt ułożenia twarzy względem obiektywu jaki algorytm jest w stanie poprawieni rozpoznać. Metoda LPBH 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.

8 Metody minimalizacji problemów