Katedra Biosensorów i Przetwarzania Sygnałów Biomedycznych

Wydział Inżynierii Biomedycznej

POLITECHNIKA ŚLĄSKA

Systemy Wbudowane i Mobilne w Biomedycynie

Projekt

Trimodalny system uwierzytelnienia biometrycznego

Kamil Suchanek
Edyta Piecuch
Joanna Chwał
Michał Letniowski
Bryan Mierzwa

Kierunek studiów: *Inżynieria Biomedyczna*

Specjalność: *Przetwarzanie i Analiza Informacji Biomedycznej*

Prowadzący projekt: Dr hab. inż. Dariusz Komorowski

ZABRZE – 2020

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

Spis treści

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

Podział prac

- Moduł wykrywania i rozpoznawania twarzy

  Kamil Suchanek, Edyta Piecuch

- Moduł rejestracji EKG

  Michał Letniowski, Joanna Chwał

- Moduł rejestracji temperatury

  Bryan Mierzwa

1. Cele i założenia

Projekt zakłada wykonanie systemu uwierzytelnienia biometrycznego składającego się z następujących modułów:

- Moduł wykrywania i rozpoznawania twarzy,

Zakłada się, że podsystem ma wykrywać obecność osoby do zidentyfikowania na podstawie obrazów zdjęć/chwilowych obrazów z strumienia wideo oraz rozpoznawać osoby zatwierdzone wcześniej.

- Moduł rejestracji EKG,

Moduł odpowiedzialny za rejestrację EKG powinien w sposób zręczny umożliwiać akwizycję sygnału, w celu pozyskania danych umożliwiających biometryczną identyfikację konkretnej osoby, która została wcześniej wprowadzona do systemu.

- Moduł rejestracji temperatury.

Rejestracja temperatury ma na celu dodatkowe zabezpieczenie systemu przed podstawieniem do identyfikacji przedmiotów nieożywionych zamiast osoby zatwierdzonej.

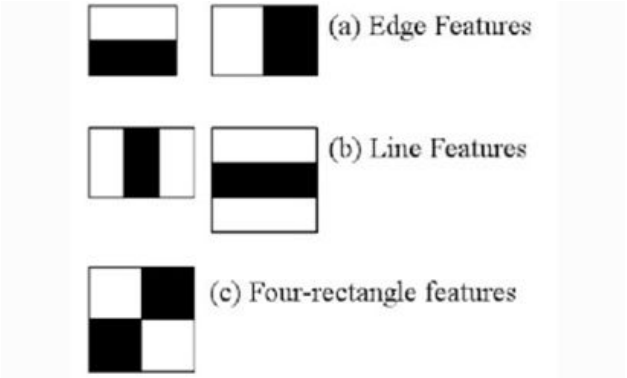
2. Metodologia

2.1. Kaskada klasyfikatorów haaro-podobnych

Kaskady zawarte w plikach XML zostały stworzone przez projekt OpenCV na podstawie wielu danych testowych (twarzy). Każda testowa twarz została przeanalizowana w celu wyciągnięcia ciemniejszych i jaśniejszych miejsc. Wartości jasności tych obszarów (progi kolorów szarości dla których obszar spełnia wymagania) jak i same obszary zostały zapisane w tym pliku XML. Na podstawie wielu takich próbek testowych, kaskada potrafi dokładnie wydedukować jak powinna wyglądać twarzy, tj. posiada informacje jakie obszary względem jakich są jaśniejsze lub ciemniejsze oraz zna progi koloru jakie powinny spełniać te obszary aby można było nazwać je twarzą. Taką kaskadę zastosowano do implementacji tego projektu.

Wykrywanie obiektów za pomocą klasyfikatorów kaskadowych opartych na cechach Haara jest skuteczną metodą wykrywania obiektów zaproponowaną przez Paula Violę i Michaela Jonesa. Jest to podejście oparte na uczeniu maszynowym, w którym Funkcja kaskadowa jest szkolona z wielu pozytywnych i negatywnych obrazów. Następnie służy do wykrywania obiektów na innych obrazach.

W przypadku wykrywania twarzy, początkowo algorytm potrzebuje wielu pozytywnych obrazów (obrazów zawierających twarzy) i negatywnych (obrazów bez twarzy), aby wyszkolić klasyfikator. Następnie musimy wyodrębnić z niego funkcje. W tym celu wykorzystywane są funkcje haar pokazane na poniższym obrazku. Są jak nasze jądro splotowe. Każda funkcja jest pojedynczą wartością uzyskaną przez odjęcie sumy pikseli pod białym prostokątem od sumy pikseli pod czarnym prostokątem.



Graficzna prezentacja funkcji Haar'a.

Teraz wszystkie możliwe rozmiary i lokalizację każdego jądra są używane do obliczania wielu funkcji. Dla każdego obliczenia funkcji należy znaleźć sumę pikseli pod białymi i czarnymi prostokątami. Aby rozwiązać ten problem, wprowadzono integralne obrazy.

Spośród wszystkich obliczonych cech większość z nich jest nieistotna. Na przykładzie obrazu poniżej, można zauważyć, że górny rząd pokazuje dwie dobre cechy, pierwsza wybrana funkcja wydaje się skupiać na właściwości polegającej na tym, że obszar oczu jest często ciemniejszy niż obszar nosa i policzków. Druga wybrana cecha polega na tym, że oczy są ciemniejsze niż grzbiet nosa. Ale te same okna obowiązujące na policzkach lub w innym miejscu nie mają znaczenia.

W tym celu stosuje się każdą funkcję na wszystkich obrazach szkoleniowych. Dla każdej funkcji znajduje najlepszy próg, który klasyfikuje twarze jako pozytywne i negatywne, jednak pojawią się błędy lub błędne klasyfikacje. Należy wybrać funkcje o minimalnym poziomie błędu, co oznacza, że są to cechy, które najlepiej klasyfikują obrazy twarzy i innych. (Proces ten nie jest tak prosty. Każdy obraz ma na początku jednakową wagę. Po każdej klasyfikacji zwiększa się waga błędnie sklasyfikowanych obrazów. Następnie wykonuje się ten sam proces. Obliczane są nowe poziomy błędów. Również nowe wagi. proces jest kontynuowany do momentu osiągnięcia wymaganej dokładności lub poziomu błędu lub znalezienia wymaganej liczby funkcji).

Ostateczny klasyfikator to ważona suma słabych klasyfikatorów. Nazywa się go słabym, ponieważ sam nie może sklasyfikować obrazu, ale wraz z innymi tworzy silny klasyfikator.

Na obrazie większość obszaru obrazu jest regionem innym niż twarz. Dlatego lepiej jest użyć prostej metody sprawdzenia, czy okno nie jest obszarem twarzy. Jeśli nie jest, odrzuca się go za jednym razem. Nie jest on przetwarzany ponownie. Zamiast tego algorytm skupia się na regionie, w którym może znajdować się twarz. W ten sposób powstaje możliwość znalezienia większej ilości czasu na sprawdzenie możliwego regionu twarzy.

W tym celu wprowadzono koncepcję Kaskady Klasyfikatorów. Zamiast stosować wszystkie 6000 funkcji w oknie, zgrupowano je w różne etapy klasyfikatorów i stosuje jeden po drugim. (Zwykle kilka pierwszych etapów będzie zawierało znacznie mniej funkcji). Jeśli okno zawiedzie w pierwszym etapie, odrzuca się je. Nie rozważa się pozostałych funkcji. Jeśli przejdzie pomyślnie, stosuje drugi etap funkcji i kontynuuje proces. Okno, które przechodzi przez wszystkie etapy, jest obszarem twarzy.

2.2. Metody rozpoznawania twarzy

Istnieją trzy podstawowe metody klasyfikacji twarzy oferowane przez oprogramowanie OpenCV, są to Eigenface, Fisherface oraz LBPH. Metody te stanowią kombinację różnych metod przetwarzania, ekstrakcji i minimalizowania ilości cech z obrazów.

2.2.1. Eigenface

Jest to metoda oparta głównie na analizie głównego składnika - PCA (ang. Principal Component Analysis). Wymaga ona nauki na obrazach o tych samych wymiarach. Polega ona na obliczeniu wektorów własnych dla analizowanego obrazu oraz posortowaniu ich względem wartości własnych. Wektory odpowiadające największym wartościom własnym uwzględniane są w procesie rozpoznawania twarzy.

Obrazy testowe są przetwarzane, a wyekstrahowane w ten sposób cechy zestawiane są z istniejącymi w przeszkolonym modelu i zwracana jest etykieta dla minimalnej odległości euklidesowej. Oznacza to, że nawet obrazy osób spoza zbioru uczącego zostaną przyporządkowane do jednej z znanych klas, podobnie jak w pozostałych metodach.

2.2.2. Fisherface

Fisherface jest metodą opartą o liniową analizę dyskryminacyjną - LDA (ang. Linear Discriminant Analysis), która polega na znalezieniu takich zależności w analizowanym zbiorze danych, które najlepiej rozróżniały by istniejące klasy. Metoda ta jest mniej wrażliwa na zależności wynikające z oświetlenia i innych czynników mogących generować częstości w obrazie uznawane za istotne z punktu widzenia PCA. Podobnie jak w przypadku poprzedniej metody, zbiór obrazów uczących i testowych muszą mieć te same wymiary.

2.2.3. LBPH

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

Metoda oparta o cechy lokalne obrazu w przeciwieństwie do już wymienionych oparta jest na ustalaniu lokalnych wzorcach binarnych (LBP ang. Local Binary Pattern) badających sąsiedztwo każdego piksela w celu generowania liczb dziesiętnych na ich podstawie. Obraz uzyskany w ten sposób podzielony zostaje na regiony , dla których wyodrębnione zostają histogramy. Wektor cech tworzony jest poprzez konkatenację owych histogramów.

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

3. Metodyka projektu

Pracę nad poszczególnymi modułami należy rozdzielić w celu sprawnego przebiegu postępów oraz rozdzielenia odmiennych tematycznie zadań.

3.1. Moduł wykrywania i rozpoznawania twarzy

Wykrywanie twarzy na obrazie przy pomocy kaskady klasyfikatorów haaro-podobnych oraz typowych metod rozpoznawania twarzy oferowanych przez oprogramowanie OpenCV.

3.2. Moduł rejestracji EKG

Wykonanie prostego wzmacniacza dla I odprowadzenia EKG, wymagającego przyłożenia po jednym palcu z obu dłoni do elektrod.

Zarejestrowany sygnał ma podlegać przetwarzaniu do postaci przestrzeni cech sygnału zgodnie z wcześniej przygotowanym algorytmem, zastosowanym do generowania danych uczących.

Algorytm musi zostać zaimplementowany dla płytki rozwojowej STM.

3.3. Moduł rejestracji temperatury

Prosty algorytm na platformę STM weryfikacji temperatury ciała rejestrowanej z powierzchni skóry dowolnym, wiarygodnym czujnikiem. Zakres akceptowalnych temperatur zostanie dobrany eksperymentalnie.

4. Realizacja projektu

4.1. Moduł wykrywania i rozpoznawania twarzy

W pierwszym etapie należy wykryć twarz na obrazie aby uzyskać dane ROI dla algorytmu rozpoznającego konkretne osoby.

4.1.1. Wykrywanie twarzy kaskadą Haara

```
# initialize the video stream and allow the camera sensor to warmup
print("[INFO] warming up camera...")
vs = VideoStream(usePiCamera=args["picamera"] > 0).start()
time.sleep(2.0)
```

do zmiennej vs przypisujemy ścieżkę z kamery internetowej, a następnie wyświetlamy to co zostało przechwycone do wcześniej przygotowanego okna aplikacji.

```
faces=self.findface(image)
for (x,y,w,h) in faces:
    image = cv2.rectangle(image,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = image[y:y+h, x:x+w]
    eyes = self.eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

image = Image.fromarray(image)
image = ImageTk.PhotoImage(image)
```

Wykrywanie twarzy na obrazie pobranym z kamery internetowej opiera się na wykorzystaniu algorytmu kaskady Haara. Kolejne piksele obrazu są porównywane z poprzednimi pod względem poziomu szarości, ponieważ algorytm pracuje na obrazach monochromatycznych. następnie po odnalezieniu właściwego obszaru, za

pomocą cech , które ten algorytm wykorzystuje, powyższy kod “rysuje” na obrazie kwadrat, w którego wpisana jest twarz.

4.1.2. Pobieranie obrazu z kamerki internetowej

Metoda klasy Face_detector pobiera obraz z kamerki internetowej i przypisuje je do zmiennej instancji zapisując je w pliku z identyfikatorem obiektu Pythona. Program otwiera podgląd kamerki i umożliwia zapis obrazu przy pomocy klawisza “s” oraz przerwanie procesu poprzez użycie klawisza “q” albo kombinacji “ctrl+c”.

```
def get_photo(self):
    key = cv2.waitKey(1) # obiekt przycisku dla okna kamerki 1ms
    webcam = cv2.VideoCapture(0) # obiekt kamerki
    while True:
        try:
            check, frame = webcam.read() # odczytaj okno
            cv2.imshow("Capturing", frame) # wyświetl okno
            key = cv2.waitKey(1) # obiekt przycisku, czekaj 1ms z oknem
            if key == ord('s'):  # jeśli "s" to zapisz zdjęcie
                cv2.imwrite(filename="images\\image_object_" + str(id(self)) + '.jpg', img=frame)
                webcam.release()
                cv2.waitKey(1650)
                cv2.destroyAllWindows()
                print("Image saved!")
                break
            elif key == ord('q'): # jeśli "q" to wyłącz kamerkę
                print("Turning off camera.")
                webcam.release()
                print("Camera off.")
                print("Program ended.")
                cv2.destroyAllWindows()
                break
        except(KeyboardInterrupt): # jeśli ctrl+C albo coś innego również zakończ
            print("Turning off camera.")
            webcam.release()
            print("Camera off.")
            print("Program ended.")
            cv2.destroyAllWindows()
            break
```

Metoda get_photo().

4.1.3. Rozpoznawanie wykrytej twarzy

Aktualna wersja programu umożliwia rozpoznawanie osób z puli obrazów testowych bez interakcji z kamerką internetową w celu sprawnego opracowania podejścia tworzenia modelu, weryfikacji jego działania i ustaleniu protokołu rozpoznawania.

Fragment programu training_script.py odpowiedzialny za przygotowywanie klasyfikatora.

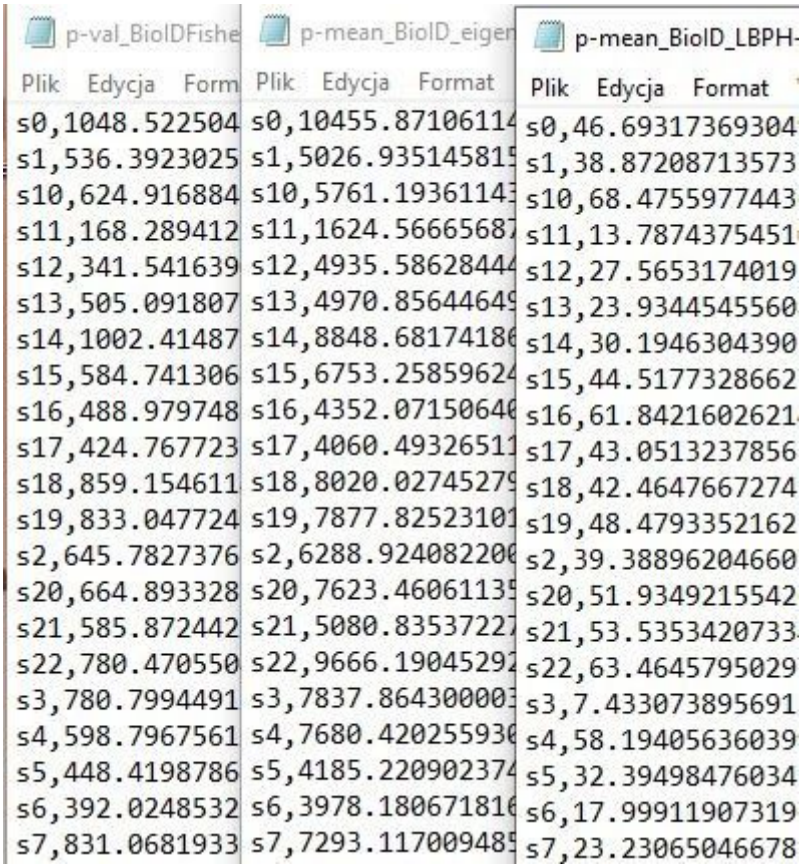
Szkolenie obrazów polega na zastosowaniu kaskady klasyfikatorów haaro-podobnych do detekcji twarzy oraz zwrócenia potrzebnych ramek ROI w celu wycięcia interesujących nas części obrazów, następnie obrazy są konwertowane na skalę szarości i ewentualnie zmienia się ich rozmiar na 256x256 pikseli w przypadku użycia metody Fisherface oraz Eigenfaces.

4.1.4. Poprawki i statystyki opisowe

Program został poprawiony pod względem wyboru ROI. Początkowo, pod uwagę brane było pierwsze wykryte ROI z listy, w przypadku, gdy kaskada wykryje więcej “twarzy”. Po korekcie, w przypadku wykrycia więcej niż jednej “twarzy” pod uwagę brane jest najszersze wykryte ROI, ponieważ dodatkowe “twarze” często są drobnymi obiektami, o cechach podobnych do twarzy. Na wygenerowanych macierzach pomyłek pojawiła się poprawa o zaledwie kilka właściwych detekcji.

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

Uzyskane odległości euklidesowe są zbyt zróżnicowane dla każdej z metod oraz dla każdego obiektu badawczego, co uniemożliwia ustalenie uniwersalnego progu odrzucenia rozpoznania (uznanie za osobę nie zatwierdzoną).

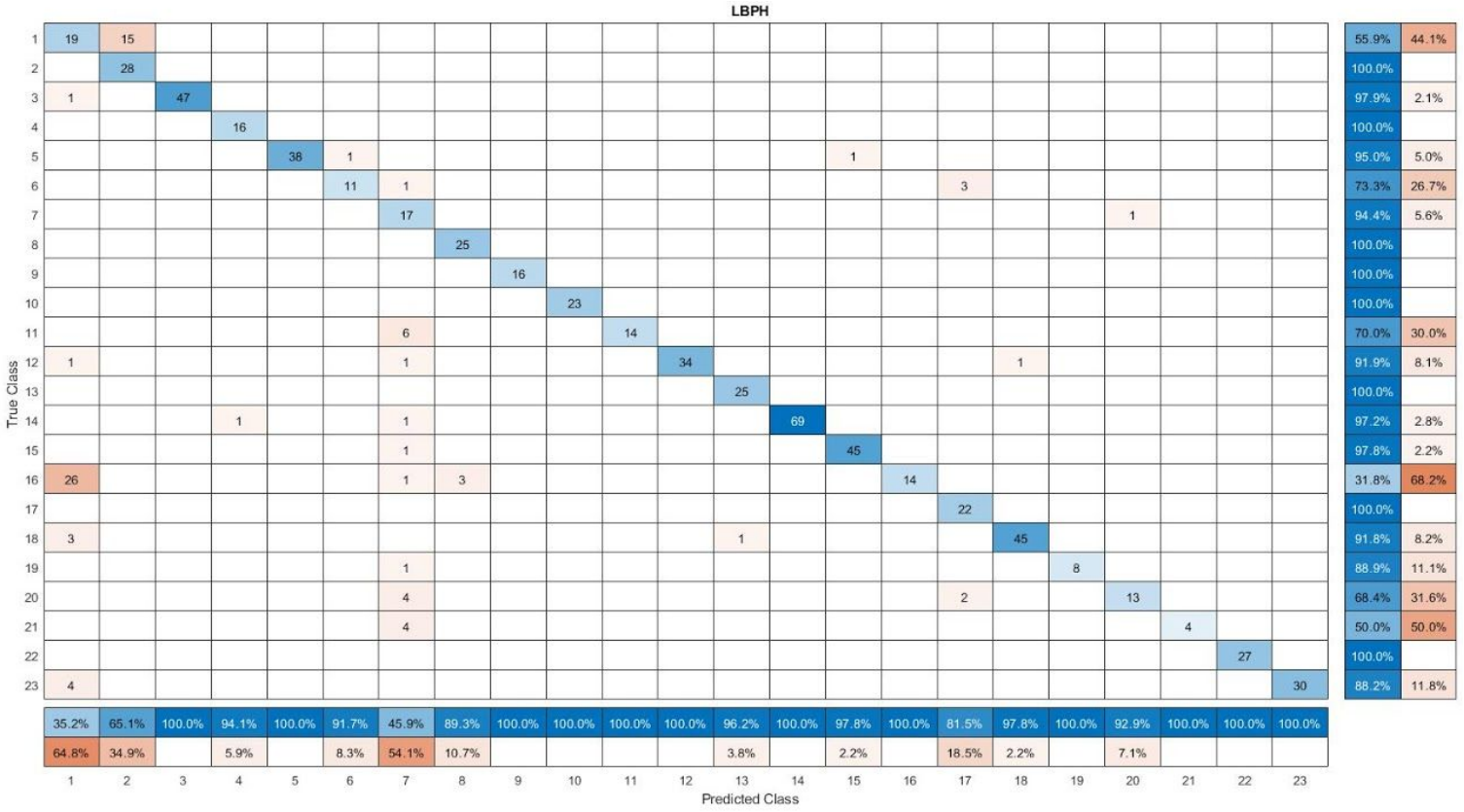


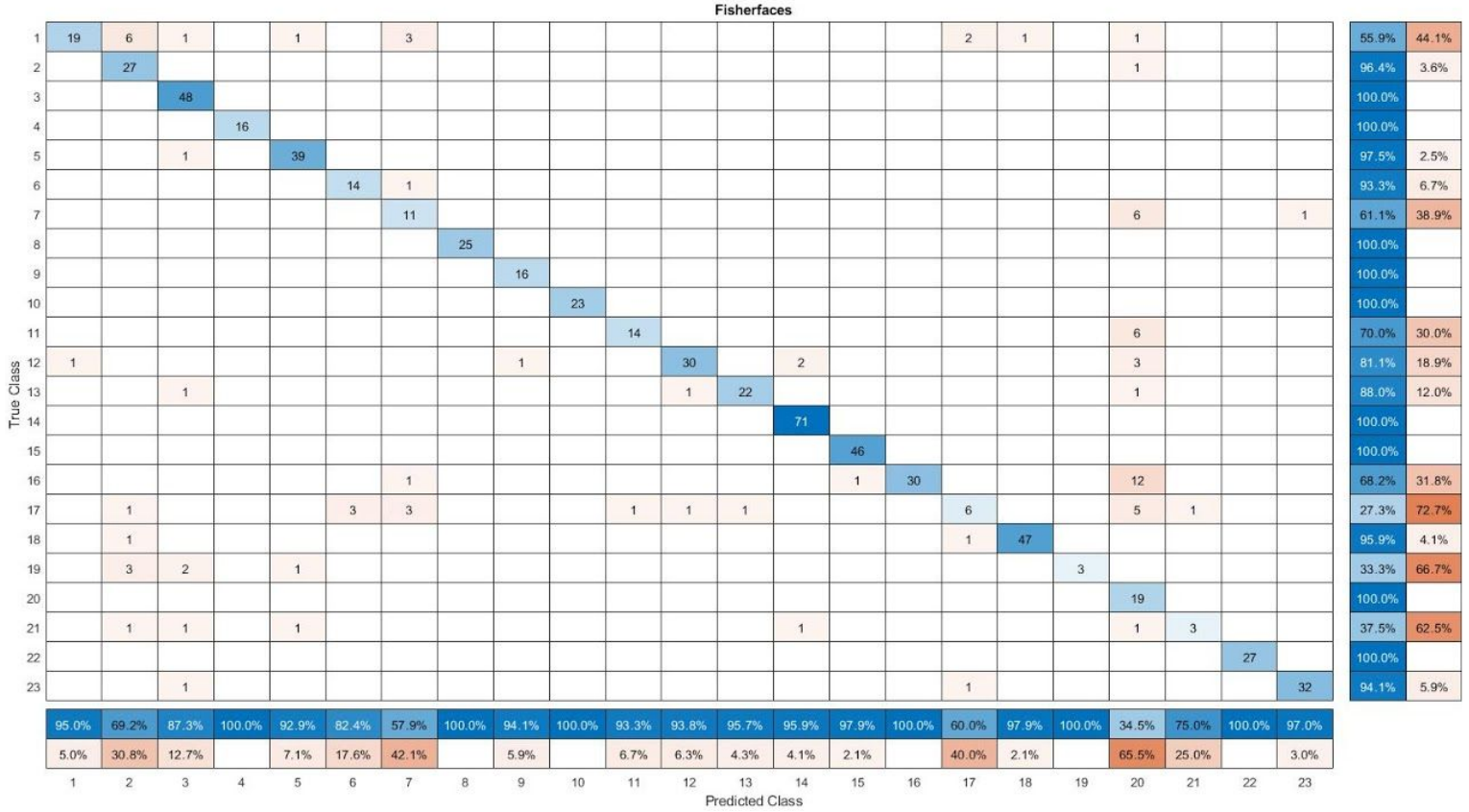
Przykładowe średnie wartości odległości euklidesowych dla danej tożsamości (s0, s1, ...), dla trzech metod od lewej: Fisherface, Eigenfaces, LBPH.

Proponuję zapis typowych wartości odległości euklidesowych dla każdej słusznie predykowanej tożsamości w zbiorze obrazów testowych. Przykładowo dla metody LBPH dla moich zdjęć, typową wartością dla poprawnych przewidywań była odległość 13-14 jednostek, więc aby program uznał moją tożsamość przy weryfikacji, wymagałby podobnych odległości jako progu. Nie byłby to środek dostateczny do pozbycia się błędów identyfikacji, ale na pewno zmniejszy to liczbę błędnie zweryfikowanych osób.

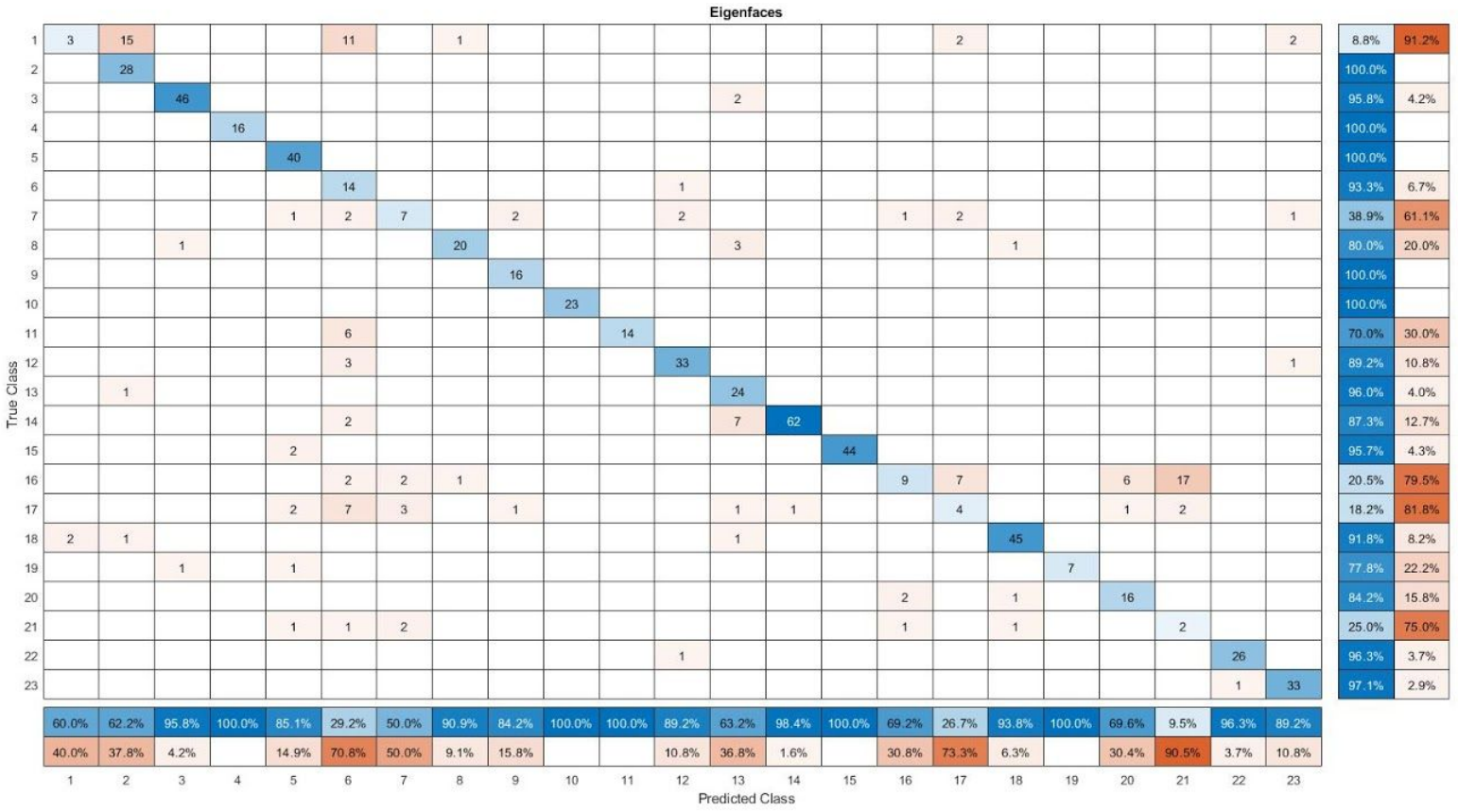
W celu dobrania metody progowania utworzono program zbierający wartości minimalne, średnie oraz maksymalne odległości euklidesowych dla każdej tożsamości, osobno dla przypadków poprawnej klasyfikacji i niepoprawnej. Ten sam program odpowiada za zbieranie danych o skuteczności rozwiązania (recognition_test.py).

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__





__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__



Macierze pomyłek dla metod LBPH, Fisherface oraz Eigenfaces.

	LBPH [%]	Fisherfaces [%]	Eigenfaces [%]
Dokładność	98,92	98,89	98,17
Czułość	86,64	82,59	76,78
Specyficzność	99,43	99,42	99,05
Precyzja	90,76	87,81	76,63
F-score	86,60	82,65	73,71

Podsumowanie statystyk skuteczności każdego z trzech algorytmów rozpoznawania twarzy.

Ze względu na niewielką poprawę w skuteczności, nie wykonano nowych statystyk po uwzględnieniu poprawki wybierania największego ROI. Pomimo zdecydowanie pozytywnego podsumowania, niektóre zbiory zdjęć nie mają zadowalającej szansy na właściwe rozpoznanie, jest to spowodowane m.in. ich niewielką ilością i jakością. Wiele zdjęć pochodzi z publicznej bazy danych BioID, są to zdjęcia dobrej jakości, wykonywane w powtarzalnych warunkach, natomiast pozostałe zdjęcia pochodzą z różnych kamerek internetowych i nie tylko, wykonywane były w różnych warunkach, a ich ilość wahała się pomiędzy 19 a 80 razem dla zestawu uczącego i treningowego, natomiast w przypadku zdjęć z bazy BioID było to około 30

zdjęć na każdy z zestawów. Na projekcie z innego kursu okazało się, że potrzeba przynajmniej 30 zdjęć w zestawie uczącym aby uzyskać zadowalające efekty (inna metoda) [3].

4.1.5. Protokół weryfikacji

W celu przeprowadzenia weryfikacji użytkowników systemu obmyślono protokół rozpoznawania, na który składają się następujące czynności:

- Część przygotowująca aplikację:
  - Przygotowanie danych zgodnie z specyfikacją oprogramowania,
  - Przeszkolenie modelu i przetestowanie w celu określenia maksymalnych odległości euklidesowych, osiąganych podczas poprawnych predykcji,
  - Wprowadzenie tożsamości w postaci tekstu, dla każdej osoby,
  - Zapisanie stanu aplikacji i nie ingerowanie w określone podfoldery.
- Część użytkowa:
  - Uruchomienie aplikacji i automatyczne wczytanie zapisanego stanu,
  - Uruchomienie kamerki internetowej i przystąpienie do weryfikacji,
  - Weryfikacji podlega:
    - Zadeklarowana tożsamość, np: Imię i Nazwisko (zmapowana na odpowiednią klasę w programie),
    - Odległość euklidesowa dla weryfikowanego zdjęcia (musi być mniejsza bądź równa niż maksymalna wartość uzyskana podczas testów).

4.1.6. Specyfikacja wewnętrzna oprogramowania

Oprogramowanie napisane w ramach projektu opiewa o:

- Skrypt SWiMwB.py - program obiektowy przeznaczony do zręcznej implementacji podstawowych czynności w ramach tworzenia modelu i rozpoznawania twarzy,
- Skrypt rec_app.py - aplikacja z interfejsem graficznym stworzona przy pomocy biblioteki Tkinter, w Pythonie.
- Skrypt recognition_test.py - skrypt przeznaczony do testowania modelu w wersji testowania podejścia do realizacji zadania projektu, został zaadaptowany do jednej z funkcji w aplikacji graficznej,

- Skrypt training_script.py - skrypt przeznaczony do trenowania modelu w wersji testowania podejścia do realizacji zadania projektu (opisany w punkcie 4.1.3),
- Inne skrypty nie podlegające dokumentacji, utworzone w celu opracowania wymienionych skryptów (wszystkie znajdują się na wspomnianym repozytorium).

Repozytorium projektu: github.com/KamilSuchanek95/Biometria_twarzy_SWiMwB_py

4.1.6.1. Skrypt SWiMwB.py

Skrypt ten zawiera dwie klasy:

- Face_detector - klasa odpowiedzialna na wykrywanie twarzy przy pomocy kaskady klasyfikatorów haaropodobnych. Implementuje ona następujące metody:
  - get_photo() - odpowiada ona za uruchomienie kamerki internetowej, przechwycenie zdjęcia w przypadku wciśnięcia przycisku “s” oraz zrezygnowanie z akwizycji zdjęcia w przypadku wciśnięcia przycisku “q”. Metoda zwraca obiekt obrazu, zapisuje go w folderze “images” oraz zapisuje go w zmiennej instancji obiektu klasy. W trakcie działania danej sesji programu, możliwe jest odwołanie się do zapisanego zdjęcia dzięki unikalnemu identyfikatorowi obiektu klasy, zawartego w zapisanym obrazie. Metoda została zaprezentowana w punkcie 4.1.2.
  - detect_face(image = None) - Metoda odpowiedzialna za wykrycie twarzy na obrazie. Zwraca ona obiekt obrazu w skali szarości, zredukowanego do obszaru ROI (zostaje on również zapisany w zmiennej instancji klasy). Metody należy użyć podając jako argument zrzut z kamerki internetowej, w przypadku, gdy argument zostanie pominięty, zostanie odczytane zdjęcie zapisane wcześniej w pliku (z odpowiednim ID obiektu klasy).
- Face_recognitor - klasa odpowiedzialna za rozpoznawanie twarzy przy pomocy trzech domyślnych algorytmów biblioteki OpenCV. Klasa ta implementuje następujące metody:
  - read_model(model_path, subjects_path) - Metoda ta jest odpowiedzialna za wczytanie do programu wcześniej wyszkolonego modelu, argumentami jest ścieżka do pliku z modelem w formacie .xml oraz ścieżka do pliku z klasami, które rozróżna klasyfikator w formacie .csv.
  - prepare_training_model(data_folder_path, eq = 0) - Metoda służąca do przygotowywania danych uczących. Przyjmuje ona dwa argumenty, w tym jeden opcjonalny - “eq”. Argument “eq” należy ustawić na 1 w przypadku, gdy potrzebujemy danych do szkolenia modeli dla

algorytmów Fisherface oraz Eigenface, w przypadku algorytmu LBPH wystarczy domyślna wartość argumentu, czyli 0. Pierwszy argument jest ścieżką do folderu zawierającego inne foldery, po jednym dla każdej osoby, którą chcemy rozpoznać. Foldery należy umieścić np.: w folderze o nazwie "training_images", a zdjęcia poszczególnych osób umieszczać w folderach wewnątrz owego i nadawać im nazwy "s0", "s1", itd... Metoda zwraca listę wykrytych twarzy przy pomocy klasy Face_detector, listę etykiet oraz listę klas, czyli nazw poszczególnych folderów ("s0", "s1", itd...).

- train_model(path = "training_images") - Metoda służąca do szkolenia modelu. Przyjmuje ona opcjonalny argument ścieżki do folderu z danymi uczącymi (wymagania co do folderu są takie same jak w przypadku metody prepare_training_data). Metoda zapisuje model jako zmienną instancji, zapisuje do pliku w formacie .xml w folderze "models" pod nazwą "<nazwa algorytmu>_model.xml", gdzie <nazwa algorytmu> należy zastąpić słowami: "lbph", "fisherface" albo "eigenface". Zostaje zapisana również lista klas do pliku o końcówce "_subjects.csv", także z nazwią użytego algorytmu rozpoznawania.
- predict(img, eq) - Metoda służąca do predykcji klasy zadanego zdjęcia twarzy. Przyjmuje ona obraz ROI zwracany przez metodę detect_face() oraz 0 albo 1 w argumencie "eq", gdzie 1 posłuży do zmiany rozmiaru obrazu, w przypadku gdy używamy algorytmu innego niż LBPH. Metoda zwraca zastosowane ROI, najmniejsza wartość odległości euklidesowej dla zadanego ROI i etykieta w postaci tekstu, wskazująca na klasę najbliższą zadanej twarzy. W przypadku nie wykrycia twarzy do rozpoznania, metoda zwróci trzy wartości None.

4.1.6.2. Skrypt rec_app.py

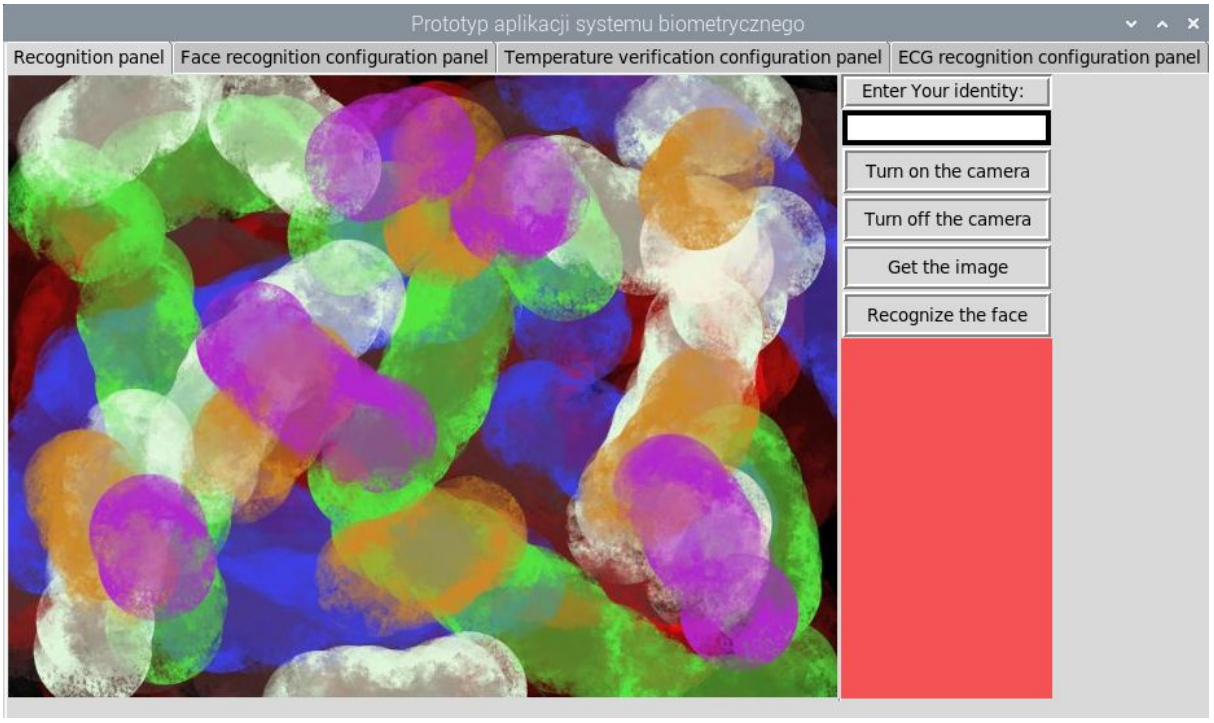
Jest to aplikacja z interfejsem graficznym, stworzona przy pomocy biblioteki Tkinter w Pythonie. Składa się ona z okna, wewnątrz którego umieszczone jest 4 paneli:

- Panel rozpoznawania - zawiera przyciski kontroli kamerki, etykietę do uzupełnienia tożsamości, oraz przycisk uruchamiający weryfikację.
- Panel konfiguracji rozpoznawania twarzy - umożliwia przeszkolenie nowego modelu, bądź jego załadowanie, razem z niezbędnymi plikami (klasy do predykcji, tożsamości, odległości euklidesowe). Można wybrać jeden z trzech algorytmów.
- Panel weryfikacji temperatury i panel rozpoznawania EKG - pozostaje pusty, ale zakładam że znalazłby się tam przycisk wywołujący osobny program, albo

została uzupełniona sekcja w kodzie aplikacji przygotowana po to, aby ułatwić dodanie nowych funkcji i kontrolek.

Główne okno aplikacji zostało przedstawione poniżej. Dotyczy ono panelu “Recognition panel”. Wybór panelu jest dostępny na górnej belce aplikacji. W tym panelu dostępne są następujące przyciski:

- Turn on the camera - włączenie kamerki internetowej oraz przesyłanie obrazu do początkowo - kolorowego obrazu początkowego na lewo od przycisków,
- Turn off the camera - wyłączenie kamerki internetowej oraz zastąpienie widoku obrazem początkowym,
- Get the image - Zapisanie aktualnego podglądku z kamerki do pliku, którego przedrostek nazwy będzie stanowiła treść wprowadzona do białego pola tekstowego poniżej etykiety “Enter Your identity”,
- Recognize the face - Przycisk wywołuje procedurę rozpoznawania twarzy.



Okno główne aplikacji.

W drugim panelu zawarte są kontrolki potrzebne do przeszkolenia albo wczytania gotowego modelu i innych plików potrzebnych aplikacji oraz zapisania dokonanej konfiguracji narzędzia.

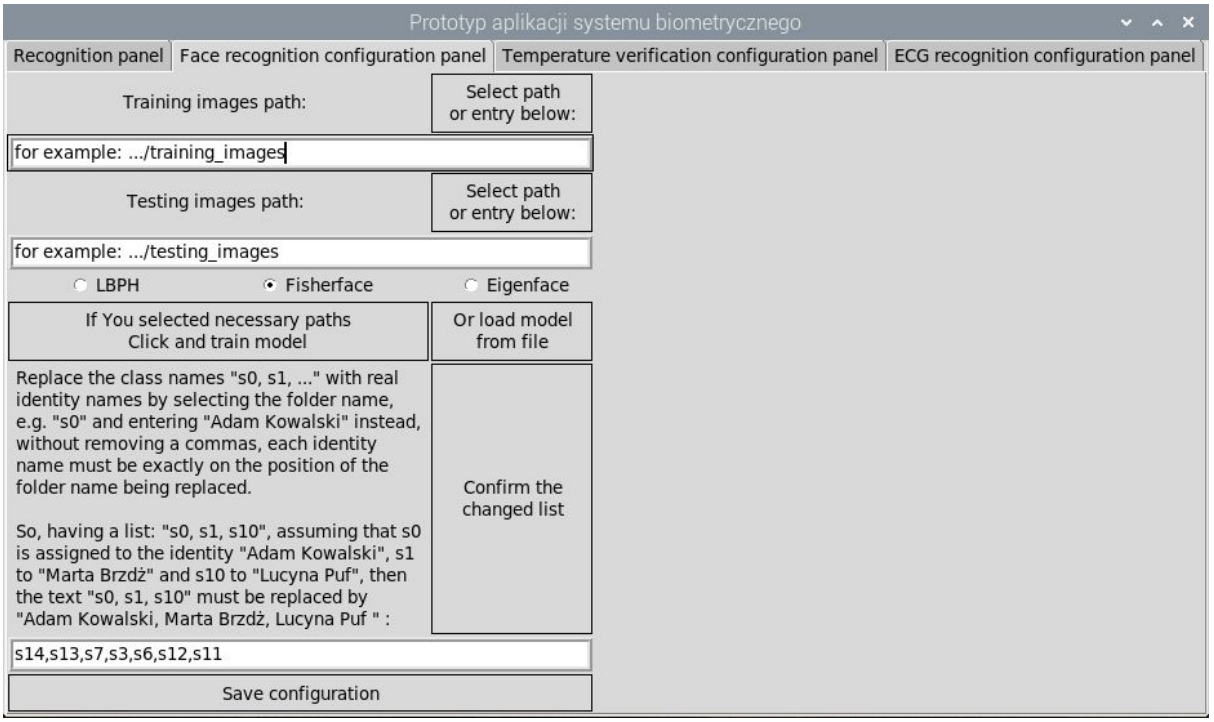
Począwszy od góry znajduje się segment do ustawienia ścieżki, prowadzącej do folderu z danymi treningowymi. jak wskazuje na to etykieta “Training images path:”. Po prawej stronie od etykiety jest przycisk służący do wybrania folderu, a poniżej

znajduje się pole tekstowe, które można uzupełnić ręcznie albo użyć wspomnianego przycisku.

Drugi segment jest taki sam jak pierwszy, w tym, że służy do wskazania ścieżki do folderu z obrazami testowymi.

Poniżej znajduje się belka z wyborem algorytmu rozpoznawania twarzy. Następnymi elementami są przyciski rozpoczęcia nauki modelu, a na prawo od niego jest przycisk służący do załadowania wcześniej przygotowanych w ten sposób plików.

Przedostatnim segmentem jest instrukcja uzupełniania tożsamości i przycisk ich zatwierdzenia. w polu tekstowym pod instrukcją należy wpisać listę tożsamości, po ukończeniu trenowania modelu. Na samym dole panelu znajduje się przycisk zatwierdzenia konfiguracji, które będą stosowane od następnego startu aplikacji.

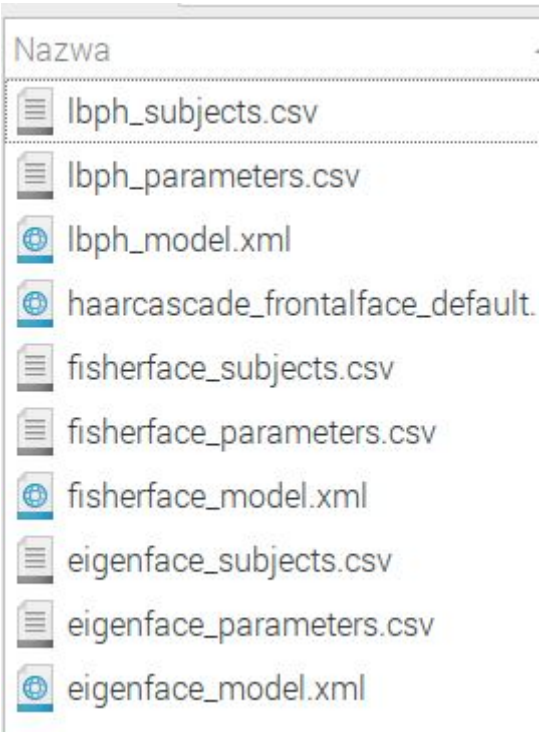


Panel ustawień wykrywania twarzy.

Aplikacja musi mieć dostęp do następujących folderów:

- models - folder przechowujący model w formacie .xml, klasyfikator do wykrywania twarzy w formacie .xml, oraz listy klas “<algorytm>_subjects.csv” oraz parametrów “<algorytm>_parameters.csv”. Aplikacja może dysponować tylko jednym zestawem plików dla danego algorytmu, stworzenie nowego poskutkuje nadpisaniem istniejącego. W celu zachowania poszczególnych danych, należy je skopiować do innej lokalizacji. Natomiast każdy zestaw plików, który został skonfigurowany poprzez wczytanie, zamiast szkolenia,

powinien zostać umieszczony w tym folderze jeśli z losowych powodów się tam nie znajduje (wcześniejsza kopia zabezpieczona w innym miejscu).



Przykładowa zawartość folderu models, gdy zdążyliśmy przeprowadzić konfigurację dla trzech algorytmów.

Plik subjects jest produktem nadmiarowym, jednak potrzebnym w pewnych momentach działania aplikacji, zawiera on listę nazw folderów (sugerowane s0, s1, itd...) oddzielonych przecinkami.

Plik parametrów zawiera po przeszkoleniu 2 kolumny, a po zatwierdzeniu nazw tożsamości 3. 1 kolumna zawiera nazwę klasy (folderu z obrazami), druga kolumna zawiera maksymalne wartości odległości euklidesowej , osiągniętej przy poprawnych predykcjach modelu, trzecia kolumna zawiera tożsamości, które należy podać w panelu głównym podczas weryfikacji.

Przykładowa zawartość pliku parametrów.

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

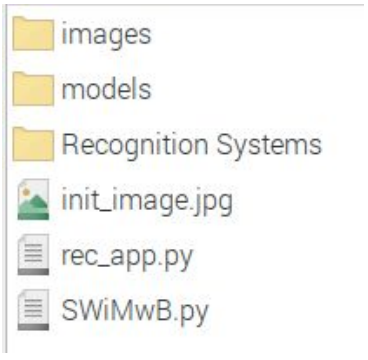
- ❖ Recognition Systems - folder zawierający plik tekstowy “set.txt”, decydujący o początkowym stanie programu.

Plik zawiera dwie wartości oddzielone przecinkiem:

- 1 albo 0, gdzie 1 oznacza, że aplikacja została skonfigurowana i model z pozostałymi plikami zostaną automatycznie wczytane do pamięci programu przy starcie aplikacji, natomiast 0, albo brak pliku oznacza, że konfiguracji nie zwieńczono.
- nazwę algorytmu, aby aplikacja wiedziała które pliki należy wczytać.

- ❖ images - folder przeznaczony do przechowywania nowych zdjęć, oraz zdjęć zapamiętanych w przeprowadzonych procesach weryfikacji, mogących posłużyć do przyszłego usprawniania modelu, bądź ukierunkowanego wykluczania osób nieporządanych (dołączenie losowej osoby, która próbowałaby przejść weryfikację, licząc na podobieństwo, zostałaby włączona do zbioru uczącego i nazwana losowym ciągiem znaków, przez co system stwierdziwszy, że sprawdzana osoba jest do niej najbardziej podobna, nie dopuściłaby do zatwierdzenia, ponieważ do poprawnej weryfikacji, potrzeba wprowadzić poprawne np.: Imię i Nazwisko, a w przypadku losowego ciągu znaków, byłoby to prawie niemożliwe).

Foldery te należy stworzyć w tym samym miejscu, gdzie umieściliśmy skrypt aplikacji. W tym samym folderze również nie może zabraknąć pliku SWiMwB.py oraz init_image.jpg.



Struktura folderów aplikacji graficznej.

4.1.7. Specyfikacja zewnętrzna - instrukcja obsługi

Moduł wykrywania i rozpoznawania twarzy umożliwia następujące scenariusze użytkowania aplikacji:

- Uruchomienie - utworzenie modelu - zapisanie konfiguracji - weryfikacja tożsamości,
- Uruchomienie - załadowanie modelu - zapisanie konfiguracji - weryfikacja tożsamości,
- Uruchomienie - weryfikacja tożsamości.

4.1.7.1. Tworzenie modelu, konfiguracja i weryfikacja twarzy

W celu przygotowania modelu należy przygotować następujące foldery:

- folder z obrazami uczącymi, np: “training_images” - wewnątrz tego folderu należy umieścić po folderze na każdą tożsamość, którą chcemy rozpoznawać, najlepiej używając nazw s0, s1, s2 itd... W każdym folderze konkretnej osoby powinno się zamieścić przynajmniej kilkanaście zdjęć.
- folder z obrazami testowymi, np: “test_images” - struktura musi być analogiczna jak w przypadku folderu z obrazami treningowymi. Jeśli folder s0 w folderze obrazów treningowych należy do np: Jana Kowalskiego, to folder o tej samej nazwie w folderze obrazów testowych również musi. Liczba obrazów może być w nich mniejsza niż w przypadku folderu z obrazami treningowymi.

Najlepiej, jeśli format obrazów ograniczy się do .JPG. Zdjęcia twarzy powinny być wykonane zachowując naturalny pion twarzy, tak aby linia oczu była równoległa do podstawy zdjęcia.

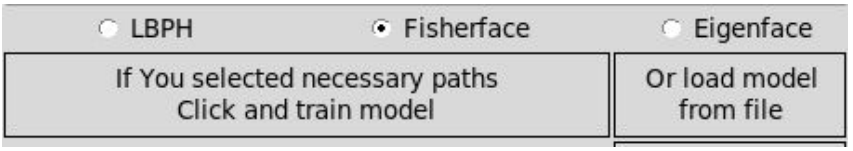
Gdy już przygotujemy potrzebne foldery, możemy przystąpić do konfiguracji, po uruchomieniu aplikacji, należy wybrać z górnej belki aplikacji panel konfiguracji rozpoznawania twarzy, naszym oczom powinno ukazać się okno z polami tekstowymi, zawierającymi przykładowe ścieżki do plików (pokazany w punkcie 4.1.6.2).

Należy wprowadzić całkowitą ścieżkę do pliku, lub ścieżkę względną, jeśli potrzebne foldery znajdują się w pobliżu pliku programu. Można również wybrać przycisk “Select path or entry below” w celu otworzenia natywnego eksploratora

plików i wskazania właściwej ścieżki do folderu z obrazami uczącymi. Analogicznie postępujemy w przypadku obrazów testowych.

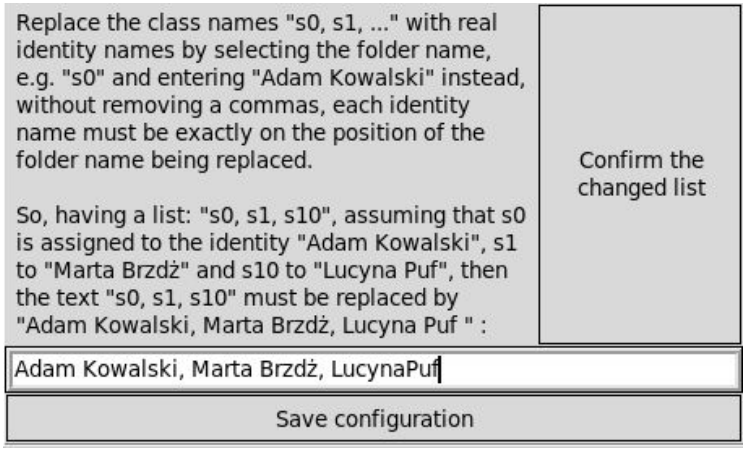
Uzupełnione ścieżki do plików.

Następnie należy wybrać algorytm, zaznaczając jedną z trzech opcji: LBPH, Fisherface, Eigenface oraz wcisnąć przycisk "If You selected necessary paths Click and train model". Należy poczekać aż przycisk się odblokuje a na trzecim polu tekstowym pojawią się klasy, czyli jak zakładamy, nazwy folderów. Proces uczenia i testowania modelu może zająć nawet kilka minut, wybrany przycisk będzie miał jaśniejszą barwę, póki jest zablokowany.



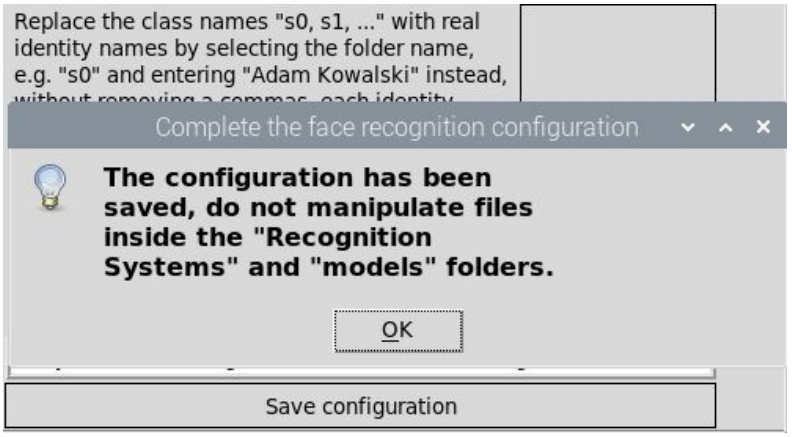
Zaznaczenie algorytmu Fisherface.

Przed zapisaniem konfiguracji przy pomocy przycisku "Save configuration" należy zastąpić nazwy klas nazwami tożsamości, których aplikacja będzie wymagała podczas weryfikacji oraz zatwierdzić owe zmiany przyciskiem "Confirm the changed list".



Przykład wypełnienia pola tożsamości.

Po zapisaniu konfiguracji powinno pojawić się powiadomienie informujące o powodzeniu procesu oraz napominające, aby nie ingerować w strukturę folderów aplikacji.



Zwieńczenie konfiguracji.

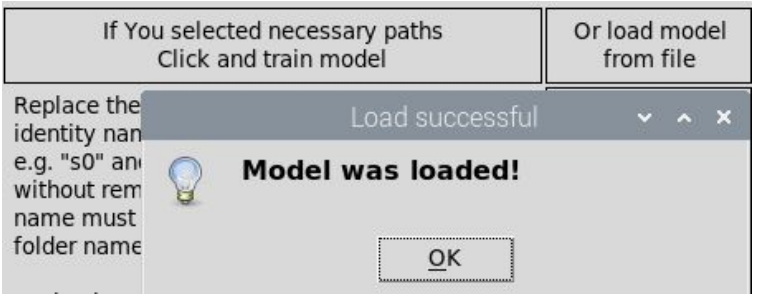
Następnie proces weryfikacji nie różni się od przedstawionego w punkcie 4.1.7.3.

4.1.7.2. Wczytanie modelu i weryfikacja twarzy

Wczytanie modelu należy przeprowadzić z panelu konfiguracji rozpoznawania twarzy, wybierając przycisk “Or load model from file”, następnie jak będą sugerować nazwy okien należy wybrać i zatwierdzić kolejno trzy pliki:

- <algorytm>_model.xml
- <algorytm>_subjects.csv
- <algorytm>_parameters.csv

Powodzenie operacji zostanie podsumowanie stosownym powiadomieniem.

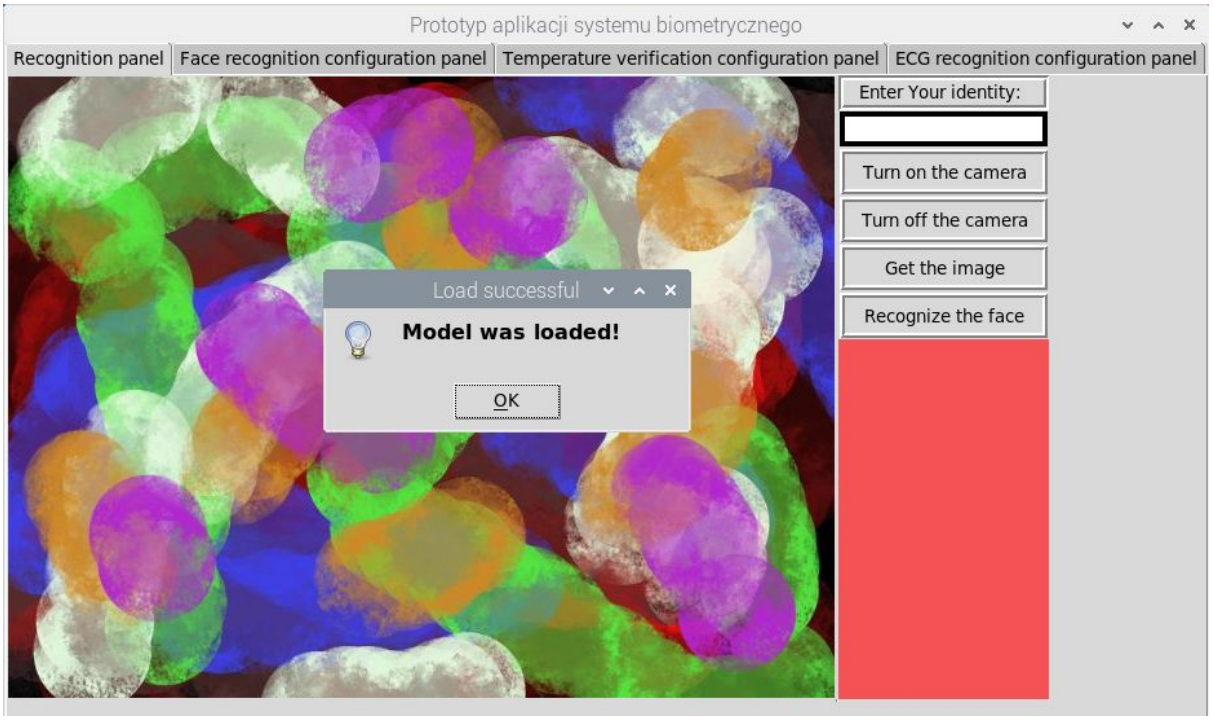


Powiadomienie o powodzeniu operacji.

Następnie proces weryfikacji nie różni się od przedstawionego w punkcie 4.1.7.3.

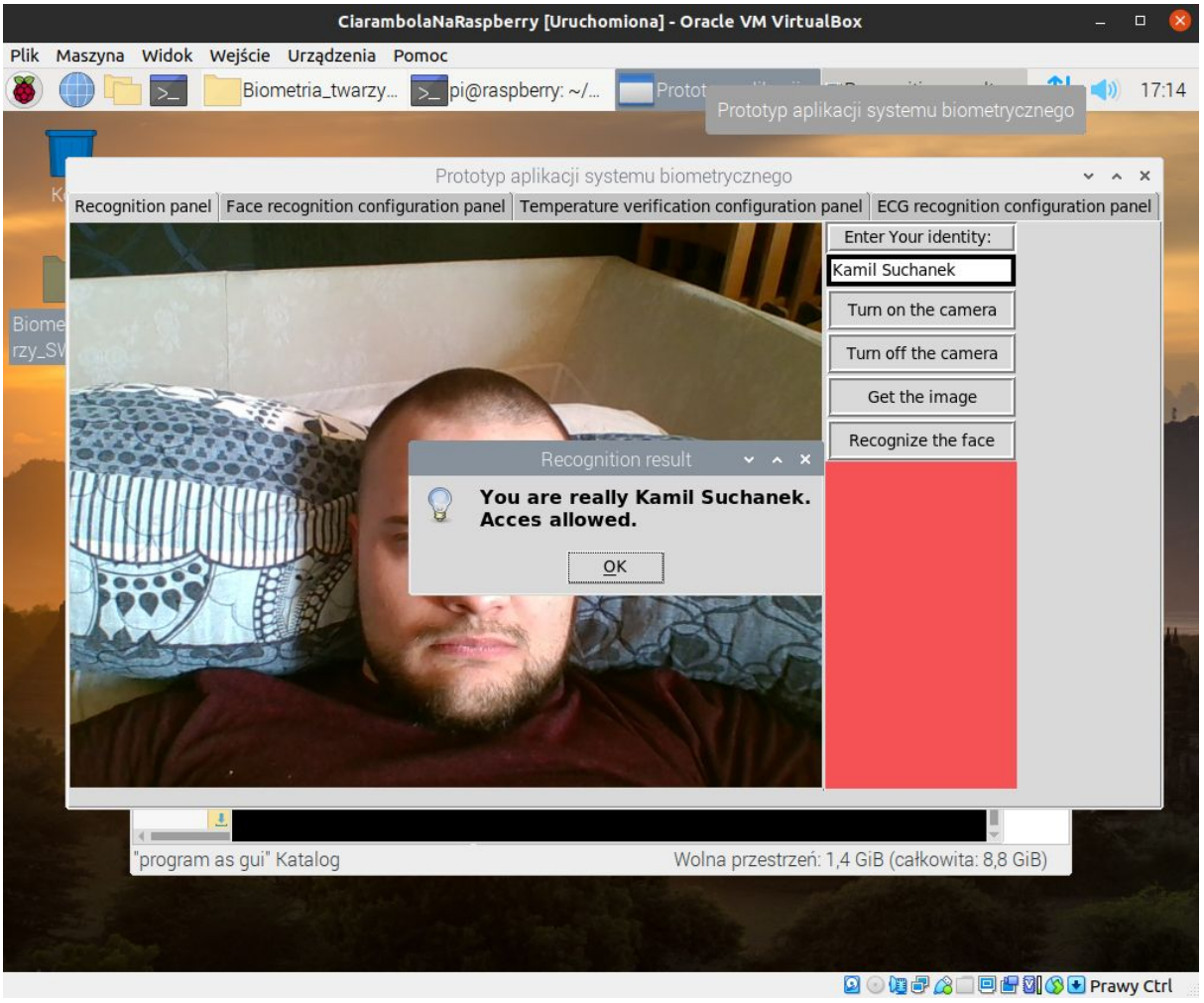
4.1.7.3. Użytkowanie po konfiguracji

Uruchomienie aplikacji po wcześniejszej konfiguracji poskutkuje informacją o tym, że stworzony wcześniej model został załadowany do pamięci programu.



Następnie należy wpisać swoje np: Imię i Nazwisko, wybrać przycisk “Turn on the camera” oraz “Recognize the face”, aż uda się nam skutecznie uchwycić twarz. W przypadku niepowodzenia przy uruchamianiu kamerki internetowej, należy upewnić się, że urządzenie jest dostępne a oprogramowanie ma dostatecznie uprawnienia.

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__



Pomyślne przejście weryfikacji.

4.2. Moduł rejestracji EKG

Sygnał EKG będzie rejestrowany na podstawie trzech elektrod umieszczonych na lewej i prawej ręce i lewej nodze.

Głównym klasyfikatorem będzie sieć neuronowa, która klasyfikacje będzie opierać o cechy charakterystyczne w sygnale ekg dla danej osoby. Dodatkowym klasyfikatorem będą algorytmy przetwarzania podstawowych parametrów sygnału w celu zawężenia obszaru klasyfikacji, tj. charakterystyczny przebieg sugeruje osobę wysoką i szczupłą, osoby o krępej budowie ciała zostaną pominięte w klasyfikacji.

4.3. Moduł rejestracji temperatury

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

Zastosowanie czujnika cyfrowego MLX90614 wykorzystującego podczerwień do rejestracji temperatury w zakresie -40°C do 85°C oraz modułu Bluetooth w celu przesyłu danych w czasie rzeczywistym.

R-Pi potrzebuje dodatkowych modułów umożliwiających komunikację Bluetooth, symulując środowisko Debiana na urządzenia tego rodzaju podejmujemy założenie, że taki moduł jest obecny.

__Kamil__ __Edyta__ __Asia__ __Michał__ __Bryan__

5. Podsumowanie

W ramach projektu opracowano podejście służące do weryfikacji ludzi na podstawie zdjęć twarzy oraz zaimplementowano je w języku Python. Zastosowano algorytmy będące częścią oprogramowania OpenCV. Powstał skrypt wykorzystujący bibliotekę OpenCV oraz aplikacja graficzną przy pomocy biblioteki Tkinter. Przedstawione rozwiązanie spełnia założenia projektu.

Bibliografia

1. https://www.docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html
2. http://michalbereta.pl/dydaktyka/PatternRecognition/Rozpoznawanie_wzorcow_w5.pdf
3. https://github.com/KamilSuchanek95/Biometria-twarzy-UM-m/blob/master/Sprawozdanie_UM_Suchanek_Piecuch.docx.pdf
4. .