Katedra Biosensorów i Przetwarzania Sygnałów Biomedycznych

Wydział Inżynierii Biomedycznej

POLITECHNIKA ŚLĄSKA

Uczenie Maszynowe

Projekt

Rozpoznawanie twarzy jako narzędzie uwierzytelnienia biometrycznego

Suchanek Kamil, Piecuch Edyta

Kierunek studiów: *Inżynieria Biomedyczna*

Specjalność: *Przetwarzanie i Analiza Informacji Biomedycznej*

Prowadzący projekt: Mgr inż. Konrad Duraj

ZABRZE – 2020

Spis treści

1. Cel i założenia

Celem projektu jest zaprojektowanie procesu uwierzytelnienia biometrycznego na podstawie obrazu twarzy. W ramach projektu zakłada się oprogramowanie przy pomocy dowolnych technologii realizacji następujących zadań:

- wyekstrahowanie danych biometrycznych z obrazów,

  Wykrycie twarzy w analizowanych obrazach przy pomocy algorytmu LBP oraz wyznaczenie ROI w celu dalszej analizy.

- zastosowanie maszyny wektorów wspierających w celu rozróżnienia konkretnych twarzy,
- przetestowanie rozwiązania i ocena skuteczności identyfikacji.

Zakłada się zastosowanie zaprojektowanego rozwiązania i ocenę jego skuteczności zgodnie z ogólnie przyjętymi metodami oraz ewentualną próbę poprawy jakości uzyskanych wyników.

2. Lokalne wzorce binarne

Lokalne wzorce binarne (LBP) są rodzajem wizualnego deskryptora stosowanego do klasyfikacji w wizji komputerowej.

Wektor funkcji LBP, w najprostszej formie, jest tworzony w następujący sposób:

- Podzielenie badanego okna na komórki (np. 16 x 16 pikseli na każdą komórkę).
- Dla każdego piksela w komórce porównanie piksela z każdym z 8 sąsiadów (na lewym górnym, lewym środkowym, lewym dolnym rogu, prawym górnym itd.). Idąc dalej zgodnie z pikselami wzdłuż koła, tj. Zgodnie z ruchem wskazówek zegara lub przeciwnie do ruchu wskazówek zegara.
- Jeżeli wartość środkowego piksela jest większa niż wartość sąsiada, wpisanie „0". W przeciwnym razie - „1". Daje to 8-cyfrową liczbę binarną (która dla wygody jest zwykle konwertowana na dziesiętną).
- Obliczenie histogramu, nad komórką, częstotliwości każdej występującej „liczby" (tj. Każdej kombinacji, których piksele są mniejsze, a które są większe niż środek). Ten histogram można postrzegać jako 256-wymiarowy wektor cech .
- Opcjonalnie znormalizowanie histogramu.
- Połączenie (znormalizowanie) histogramów wszystkich komórek, daje wektor funkcji dla całego okna.

Powstały wektor można wykorzystywać do uczenia maszynowego i nie tylko [1].

3. Histogram zorientowanych gradientów

Histogramy zorientowanych gradientów HOG są deskryptorami obrazu, które opisują kształt i pomagającymi w znalezieniu w obrazie konkretnego obiektu. Działają na zasadzie zliczania występowania gradientów, które występują w tej samej orientacji przestrzennej (pod określonym kątem), w konkretnym, dokładnie określonym fragmencie obrazu. W równomiernie rozmieszczonych fragmentach obrazu są liczone gradienty, w przeciwieństwie do innych deskryptorów. W celu poprawienia skuteczności wykrywania obiektów stosuje się lokalną normalizację kontrastu w tych regionach, które się na siebie nakładają [2].

Poprzez podzielenie obrazu na małe fragmenty (komórki), uzyskiwane są deskryptory HOG, a następnie dla każdej komórki obliczany jest histogram występowania orientacji krawędzi. Połączenie tych histogramów, obliczonych dla wszystkich komórek jest deskryptorem obrazu HOG [2].

Obraz, który zawiera poszukiwany obiekt, po zastosowaniu na nim dozwolonej transformacji nadal będzie zawierał cechy deskryptora HOG, pozwalające na wskazanie obecności poszukiwanego obiektu. Deskryptor HOG jest odporny na wszystkie transformacje geometryczne, poza rotacją [2].

Po fazie obliczenia gradientów dla każdej komórki tworzony jest histogram, który przedstawia rozkład orientacji krawędzi (rozkład tych gradientów) we wszystkich pikselach w komórce. Następnie komórki obrazu są grupowane w większe bloki, w celu przeprowadzenia operacji normalizacji kontrastu [2].

Na rysunku 1.1 przedstawiono realizację HOG (jedna komórka na blok, 1616 pikseli w komórce, 8 kanałów) dla przykładowego obrazu [2].

Rys. 3.1. Histogram zorientowanych gradientów: a) obraz wejściowy, b) histogram zorientowanych gradientów HOG dla obrazu wejściowego [2]

4. Maszyna wektorów wspierających

Rozpoznanie twarzy dzięki maszynie wektorów wspierających rodzi wyzwanie dla określenia funkcji reprezentujących poszczególne twarze. Problem leży w mnogości wariantów wizualnej prezentacji zależnych od punktu widzenia, oświetlenia i mimiki.

Metody geometryczne oparte na cechach wykrywa rysy twarzy, takie jak oczy, nos, usta i podbródek. Jako deskryptory twarzy używane są odległości, kąty i powierzchnie. To podejście umożliwia łatwą redukcję danych i jest niewrażliwe na zmiany oświetlenia i punktu widzenia. Niestety są one dość niepewne [3].

Metody szablonowe i neuronowe zwykle działają bezpośrednio na matrycach intensywności pikseli reprezentujących twarze. Wdrożenie i początkowa analiza obrazów jest znacznie uproszczona względem metod geometrycznych. Ekstrakcja cech oddana jest w ręce algorytmowi sieci.

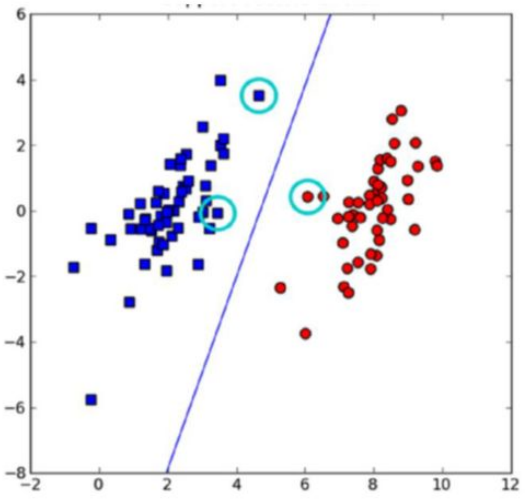
Support Vector Machine, czyli klasyfikator SVM jest techniką uczenia maszynowego pozwalająca analizować dane i rozpoznawać wzorce w celu klasyfikacji. Jest to klasyfikator, który jest liniowy, binarny i nieprobabilistyczny. Po fazie uczenia, w najprostszym przypadku SVM pozwala na określenie jednej z dwóch klas, do której należy zbiór danych wejściowych. Taki model uzyskany po fazie uczenia reprezentuje dane ze zbioru uczącego są od siebie oddzielone granicą z możliwie jak najszerszym marginesem. Punkty nie należące do zbioru uczącego są klasyfikowane poprzez określenie, po której stronie granicy znajduje się konkretny punkt, przez co przypisywana jest mu przynależność do określonej klasy [2].

SVM jest klasyfikatorem binarnym, jednak istnieją odmiany, które pozwalają na klasyfikację wieloklasową, którą stosuje się np. w przypadku przypisania danemu znakowi konkretnej cyfry lub litery, co może być zastosowane w rozpoznawaniu tekstu. W przypadku wielu klas, przy użyciu klasyfikatora SVM stosuje się podział problemu wieloklasowego na problemy dwuklasowe, w takiej sytuacji stosować można jedno z dwóch podejść:

1) One-versus-one – w tym przypadku tworzy się wszystkie dwuelementowe kombinacje klas, a dla każdej klasy uczony jest osobny klasyfikator. W rezultacie otrzymywana jest etykieta klasy, która była wynikiem klasyfikacji, największą liczbę razy.
2) One-versus-all – w tym podejściu trenowanych jest tyle klasyfikatorów, ile zadano klas. Dla każdego z nich dane podzielone są na dwie klasy, z których

jedna zawiera dane z pierwotnego zbioru uczącego, druga natomiast zawiera dane z pozostałych klas. Każda próbka jest analizowana przy użyciu wszystkich wytrenowanych klasyfikatorów, natomiast wynikiem jest klasa która została wybrana z największym wskazaniem przez przyporządkowany jej klasyfikator.

Na rysunku 2.1 przedstawiono ideę klasyfikacji dla klasyfikatora SVM. Zakreślone punkty to tzw. wektory nośne (ang. support vectors), a linia dzieląca zbiór danych to granica, zwana hiperpłaszczyzną (ang. hyperplane) [2].



Rys. 4.1. Idea klasyfikacji dla klasyfikatora SVM [2]

4.1. Maszyny wektorów wspierających w środowisku Matlab

W uczeniu maszynowym hiperparamerty kontrolują proces uczenia się. W programie Matlab zastosowana funkcja *fitcecoc()* nie stosuje domyślnie optymalizacji hiperparametrów, w przypadku ustawienia na automatyczną optymalizację, domyślnie Matlab wybierze następujące parametry:

- ‘Learners’ - ‘svm’ - {'BoxConstraint','KernelScale'}

Uczniowie	Kwalifikujące się hiperparametry (pogrubienie = wartość domyślna)	Domyślny zakres
'discriminant'	**Delta**	Log skalowane w zakresie [1e-6,1e3]
	DiscrimType	'linear', 'quadratic', 'diagLinear', 'diagQuadratic', 'pseudoLinear', i 'pseudoQuadratic'
	Gamma	Rzeczywiste wartości w [0,1]
'kernel'	**Lambda**	Wartości dodatnie skalowane logarytmicznie w zakresie [1e-3/NumObservations,1e3/NumObservations]
	KernelScale	Wartości dodatnie skalowane logarytmicznie w zakresie [1e-3,1e3]
	Learner	'svm' i 'logistic'
	NumExpansionDimensions	Liczby całkowite skalowane w przedziale [100,10000]
'knn'	**Distance**	'cityblock', 'chebychev', 'correlation', 'cosine', 'euclidean', 'hamming', 'jaccard', 'mahalanobis', 'minkowski', 'seuclidean', i 'spearman'
	DistanceWeight	'equal', 'inverse'i 'squaredinverse'
	Exponent	Wartości dodatnie w [0.5,3]
	NumNeighbors	Dodatnie wartości całkowite skalowane logarytmicznie w zakresie [1, max(2,round(NumObservations/2))]
	Standardize	'true' i 'false'
'linear'	**Lambda**	Wartości dodatnie skalowane logarytmicznie w zakresie [1e-5/NumObservations,1e5/NumObservations]
	Learner	'svm' i 'logistic'
	Regularization	'ridge' i 'lasso'
'naivebayes'	**DistributionNames**	'normal' i 'kernel'
	Width	Wartości dodatnie skalowane logarytmicznie w zakresie [MinPredictorDiff/4,max(MaxPredictorRange,MinPredictorDiff)]
	Kernel	'normal', 'box', 'epanechnikov', i 'triangle'
'svm'	**BoxConstraint**	Wartości dodatnie skalowane logarytmicznie w zakresie [1e-3,1e3]
	KernelScale	Wartości dodatnie skalowane logarytmicznie w zakresie [1e-3,1e3]
	KernelFunction	'gaussian', 'linear'i 'polynomial'
	PolynomialOrder	Liczby całkowite w zakresie [2,4]
	Standardize	'true' i 'false'
'tree'	MaxNumSplits	Liczby całkowite skalowane w przedziale [1,max(2,NumObservations-1)]
	MinLeafSize	Liczby całkowite skalowane w przedziale [1,max(2,floor(NumObservations/2))]
	NumVariablesToSample	Liczby całkowite w zakresie [1,max(2,NumPredictors)]
	SplitCriterion	'gdi', 'deviance'i 'twoing'

Rys. 4.2. Zestawienie ustawień hiperparametrów wg dokumentacji Matlab [4].

5. Realizacja zadania

Wykorzystano środowisko Matlab do wstępnego przetwarzania obrazów, wyszkolenia modelu i akwizycji nowych zdjęć. Do testowania rozwiązania posłużyły również fragmenty publicznej bazy danych BioID.

5.1. Przygotowanie zdjęć twarzy

Zdjęcia zostały podzielone na dwa równoliczne (z drobnymi odstępstwami) zestawy: testowy i treningowy. Każdy z zestawów został rozłożony do osobnego folderu z numerem: s0, s1 itd. Nazwy tych folderów posłużyły jako nazwy klas do rozróżnienia. Program sporządza listę dostępnych folderów obrazów w zadanych folderach zawierających tylko takie. Następnie program przeskakuje po każdym z folderów tożsamości i sporządza kolejną listę, tym razem z obrazami. Każde zdjęcie z listy podlega detekcji twarzy - w przypadku braku wykrycia owej, lokalizacja obrazu jest wskazywana w oknie Command Window, a program bezawaryjnie kontynuuje pracę. Sprawdzana jest również wymiarowość obrazu i w razie konieczności kolorowy obraz jest konwertowany na skalę szarości.

Proces wczytywanie obrazów i pozyskiwania ROI twarzy jest wspólny dla obu procesów. Dzięki informacjom zwrotnym, z folderów można wykluczyć obrazy bez wykrytych twarzy, oraz zwrócić uwagę na te błędnie sklasyfikowane.

Obrazy umieszczone w folderach o nazwach “sN”, gdzie N jest unikatowym numerem danej tożsamości, mogą być dowolnie dodawane do folderu z innymi obrazami treningowymi oraz testowymi. Oprogramowanie przyporządkuje temu folderowi oddzielną klasę do rozpoznawania o ile folder będzie wśród danych treningowych, kwestia obecności folderu testowego jest dowolna.

Zdjęcia użytkowe do bieżącej identyfikacji pozyskiwane są z kamerki internetowej przy pomocy dodatkowego modułu pomocniczego Matlaba do akwizycji obrazów. Następnie program upewnia się czy akceptujemy zdjęcie do identyfikacji i przeprowadza detekcję i rozpoznawanie twarzy.

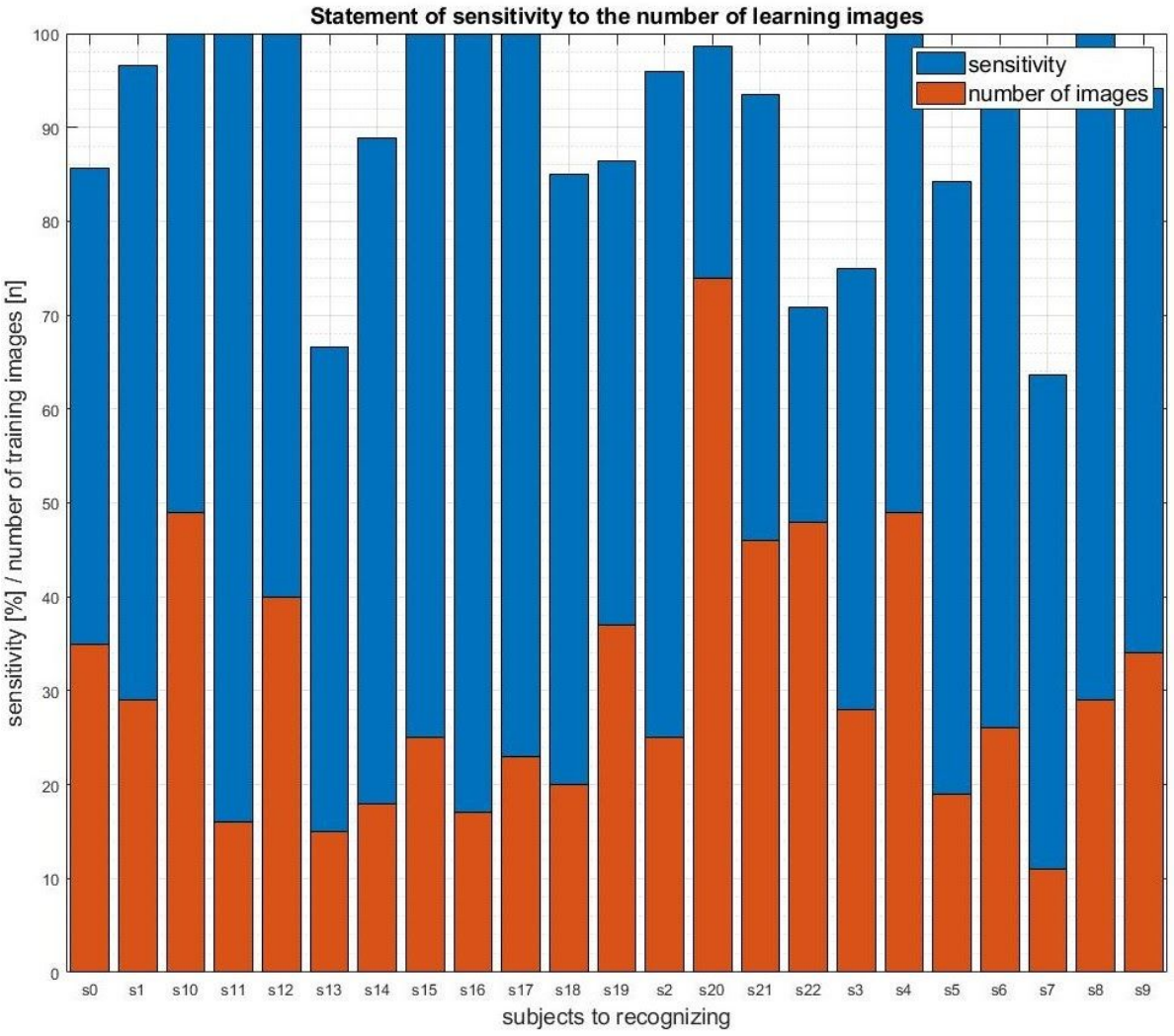
5.2. Ekstrakcja cech twarzy

Kwadratowe ROI obrazów zawierający twarze zostaje wycięty z oryginalnych obrazów i w skali szarości jest skalowany do rozmiaru 255x255 pikseli. W ten sposób przygotowany obraz trafia do algorytmu HOG. Powstały wektor 1xM, przy stałym wymiarze M bierze udział w procesie uczenia modelu SVM oraz rozpoznawania, na już przeszkolonym modelu.

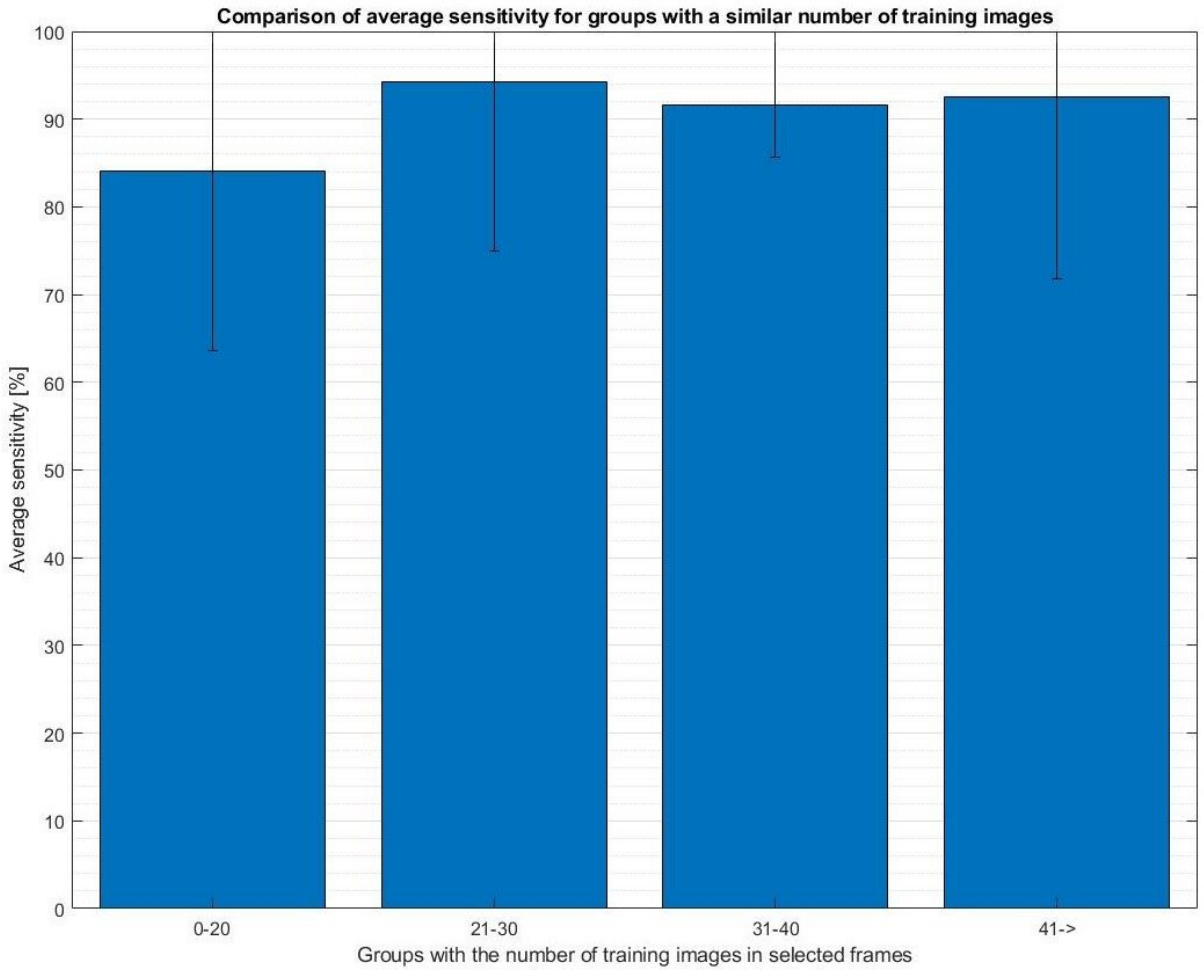
5.3. Skuteczność rozwiązania

Skuteczność metody została sprawdzona przy pomocy podstawowych statystyk i macierzy pomyłek. Wynika z nich (przy uwzględnieniu znajomości zdjęć, ich jakości), że w celu uzyskania zadowalającego poziomu rozpoznawania potrzeba około 30 zdjęć dobrej jakości, tzn odpowiednio zróżnicowanych i wyraźnych, dających możliwość uchwycenia pewnych zmienności w obrębie danej tożsamości. Większa liczba zdjęć wykonanych bez odpowiedniej dynamiki mogą dawać podobne lub gorsze rezultaty.

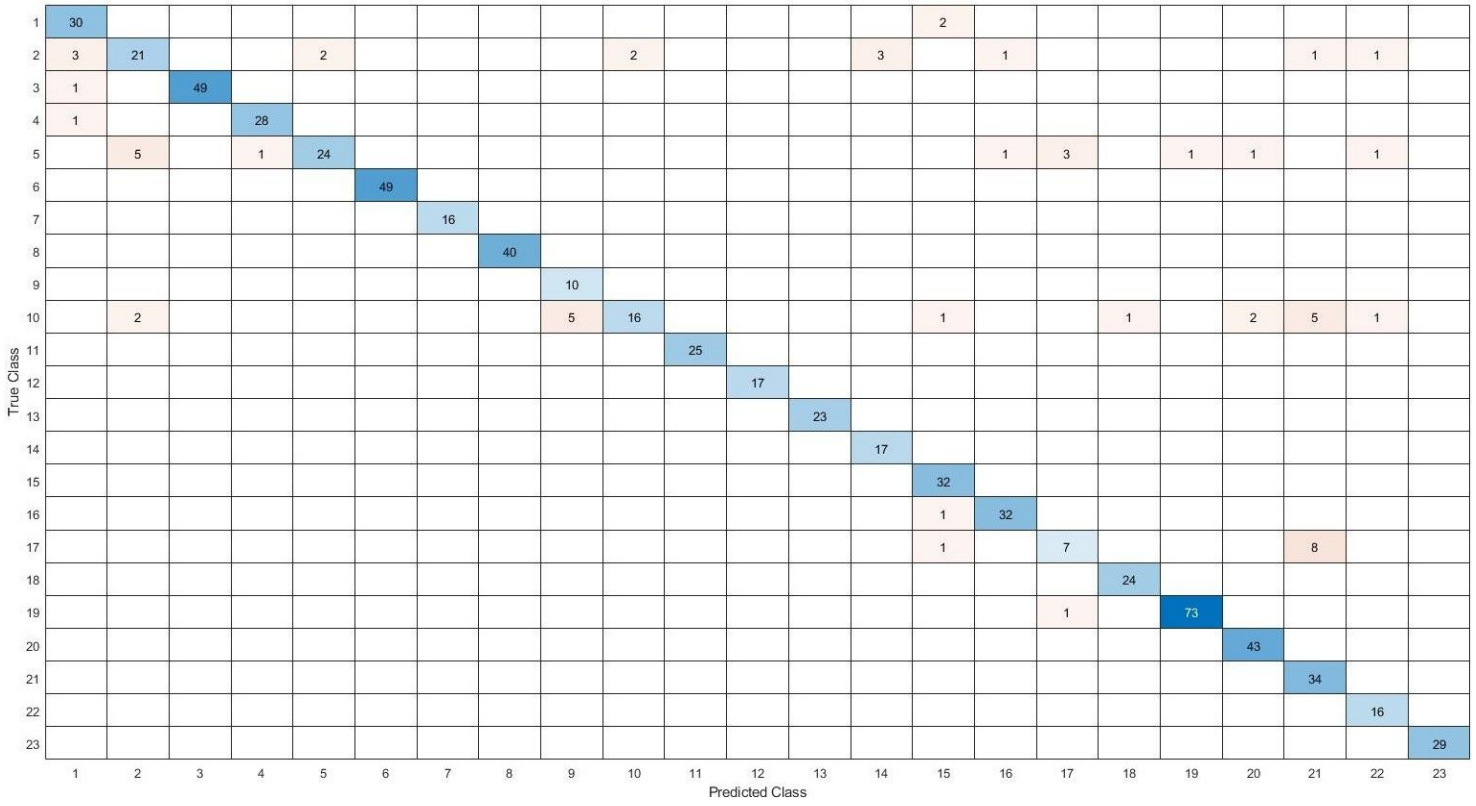
Na wykresie nr 1 przedstawiono zestawienie czułości i ilości obrazów treningowych dla każdej rozpoznawanej osoby. Na wykresie nr 2 przedstawiono uśrednienie tych wartości dla grup o zbliżonej ilości zdjęć wykorzystanych do nauki modelu. Całkowita skuteczność dla wszystkich osób wyniosła 91.8654%. Wykres nr 3 zawiera macierz pomyłek.



Rys. 5.1 Zestawienie czułości z liczbą obrazów uczących dla każdej klasy.

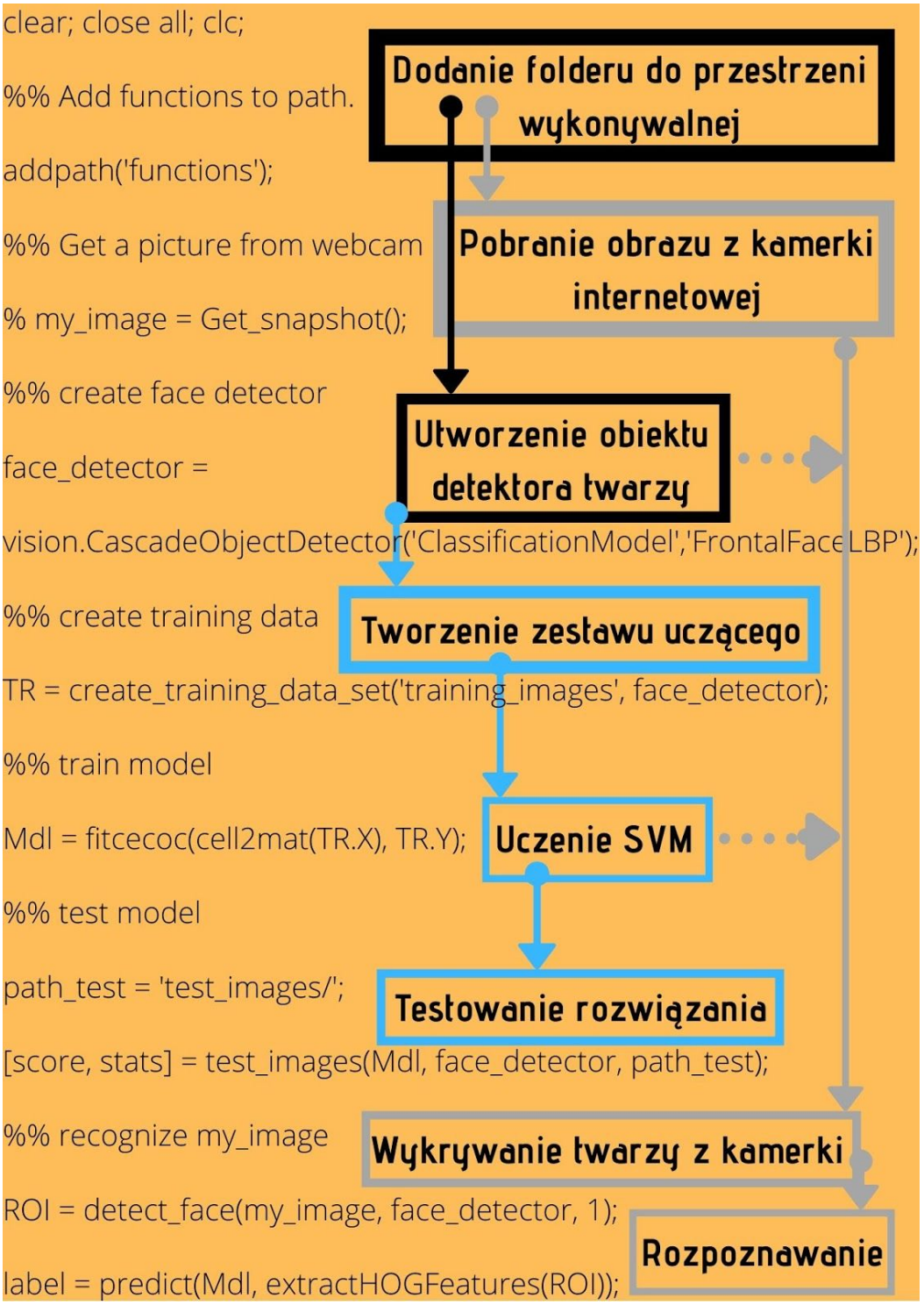


Rys. 5.2 Zestawienie średnich czułości dla klas o zbliżonej liczbie obrazów uczących.



Rys. 5.3 Macierz pomyłek.

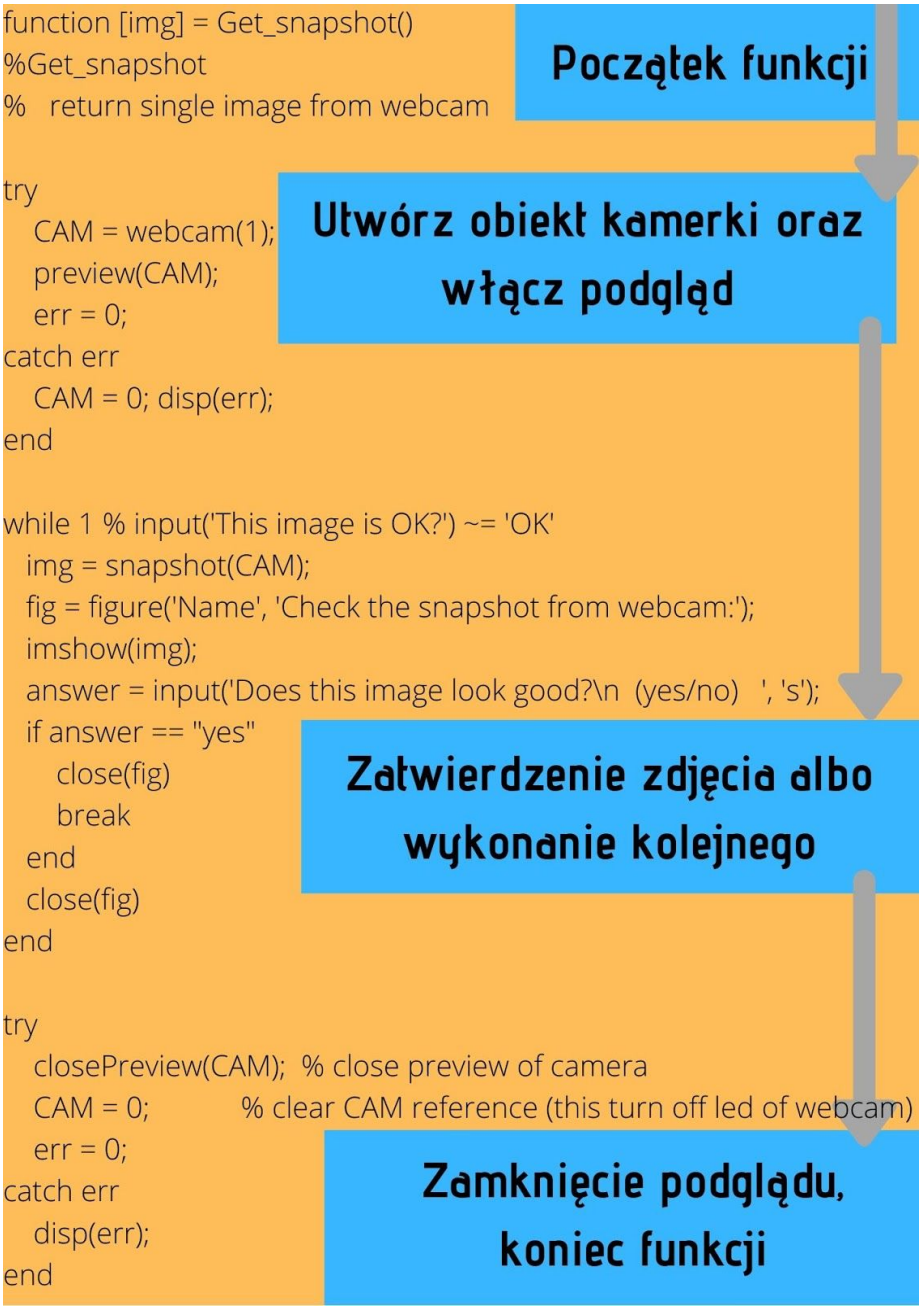
Program główny składa się z następujących poleceń:



Funkcja addpath(), dodaje odpowiednią lokalizację z plikami wykonywalnymi, w tym przypadku udostępnia funkcje programu z zewnętrznego folderu. Następnie funkcja

Get_snapshot() wykonuje zdjęcia przy pomocy kamerki internetowej i w razie powodzenia zapisuje je do zmiennej my_image. W kolejnym kroku tworzony jest obiekt wstępnie przeszkolonego modelu wykrywania twarzy przy pomocy algorytmu LBP. W celu przeszkolenia modelu tworzony jest zestaw danych uczących przy pomocy funkcji create_training_data_set. Następnie model jest szkolony funkcja fitcecoc() i jest gotowy do testów oraz rozpoznania osoby z obrazu w zmiennej my_image.

6.1. Funkcja Get_snapshot



Funkcja ta służy do pobierania obrazu z kamerki internetowej. W pierwszej kolejności próbuje uzyskać dostęp do kamerki, w przypadku powodzenia wyświetla podgląd i aktualny zrzut do zatwierdzenia. W oknie Command Window należy wpisać yes/no w przypadku odpowiednio zaakceptowania zrzutu w podglądzie albo no w celu wykonania nowego zrzutu. Ostatecznie zatwierdzony zrzut ekranu zapisywany jest do zmiennej i zwracany przez funkcję.

6.2. Funkcja Cut_the_ROI

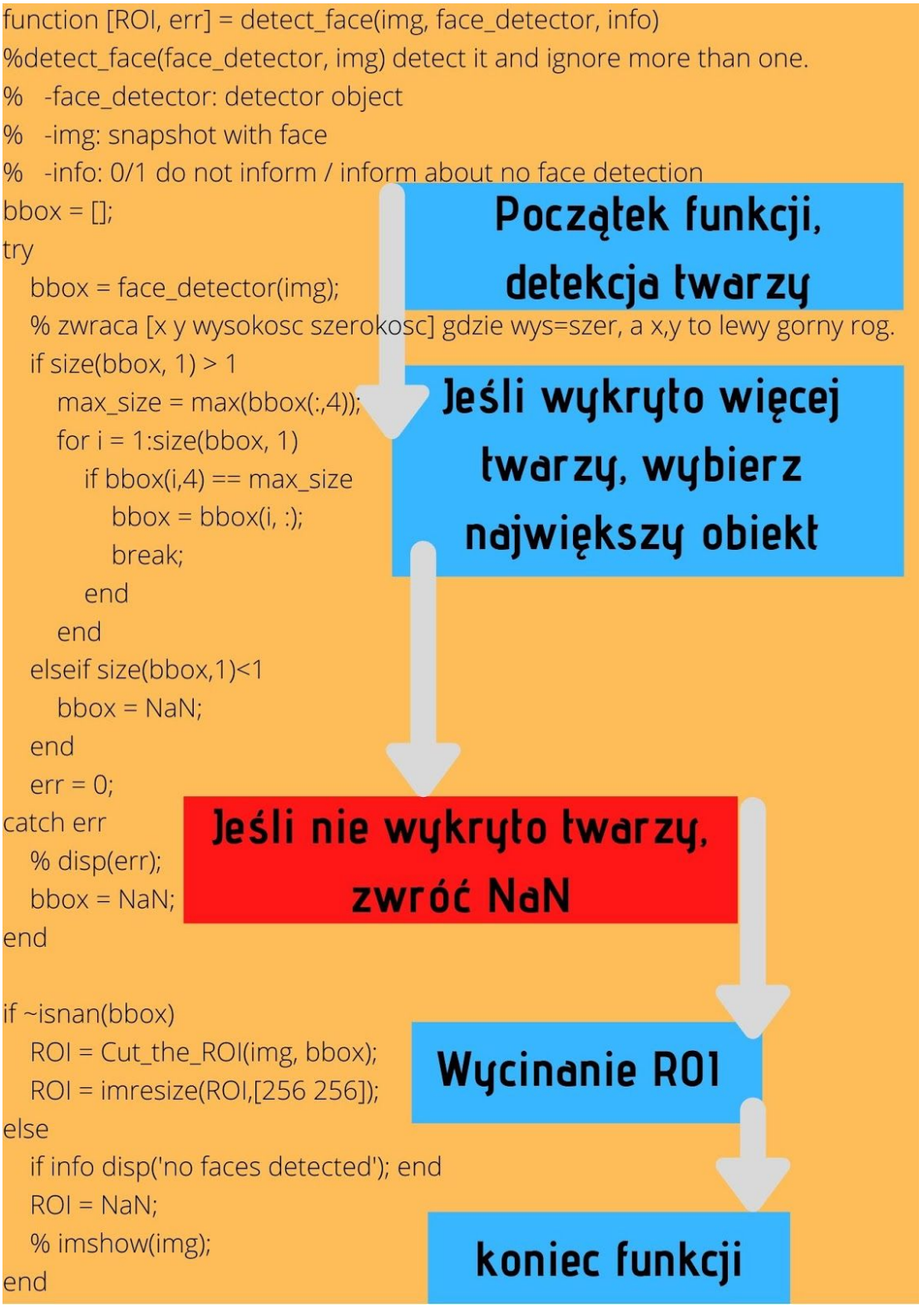
Funkcja odpowiedzialna za wydzielenie ROI z obrazu, na którym wykryjemy twarz. Wykorzystuje ona zmienną bbox z współrzędnymi z metody obiektu vision.CascadeObjectDetector programu MATLAB oraz obraz z stwierdzoną obecnością twarzy. Argumentem wyjściowym jest odpowiednio przycięty obraz.

6.3. Funkcja detect_face

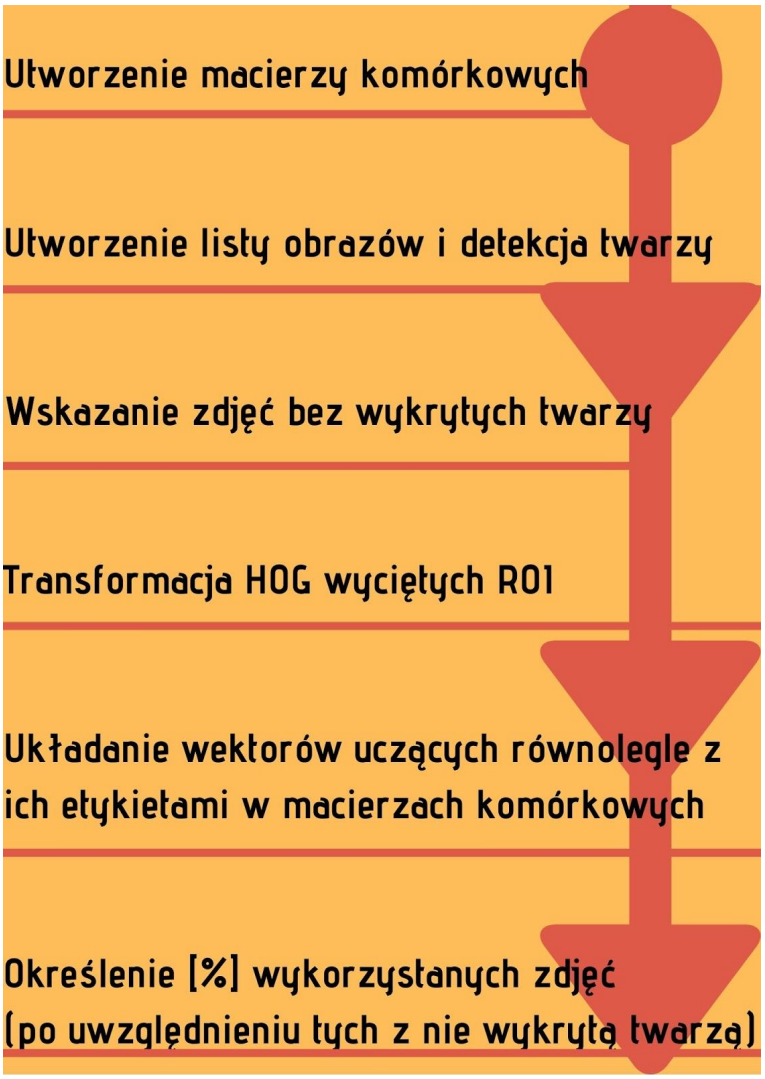
Funkcja kolejno wykrywa twarz przy pomocy obiektu vision i sprawdza, czy nie zostało wykrytych więcej niż jedna twarz. Dla pewnych obrazów to następuje, więc zdecydowano aby wziąć pod uwagę największą zwróconą ramkę, a resztę zignorować. Jest to słuszne, ponieważ twarz w zdjęciach, gdzie ich poszukujemy jest zazwyczaj największym tego rodzaju obiektem, na którym nam zależy, często mniejsze (nawet kilkukrotnie), niewyraźne elementy zostają natomiast błędnie uznane za twarz. W przypadku braku wykrytej twarzy, zamiast przeskalowanego ROI zostanie zwrócony NaN. Przy pomocy argumentu info możemy zdecydować, czy chcemy być informowaniu i takim przypadku.

```
function [ROI, err] = detect_face(img, face_detector, info)
%detect_face(face_detector, img) detect it and ignore more than one.
%   -face_detector: detector object
%   -img: snapshot with face
%   -info: 0/1 do not inform / inform about no face detection
bbox = [];
try
    bbox = face_detector(img);
    % zwraca [x y wysokosc szerokosc] gdzie wys=szer, a x,y to lewy gorny rog.
    if size(bbox, 1) > 1
        max_size = max(bbox(:,4));
        for i = 1:size(bbox, 1)
            if bbox(i,4) == max_size
                bbox = bbox(i, :);
                break;
            end
        end
    elseif size(bbox,1)<1
        bbox = NaN;
    end
    err = 0;
catch err
    % disp(err);
    bbox = NaN;
end

if ~isnan(bbox)
    ROI = Cut_the_ROI(img, bbox);
    ROI = imresize(ROI,[256 256]);
else
    if info disp('no faces detected'); end
    ROI = NaN;
    % imshow(img);
end
```



6.4. Funkcja create_training_data_set



Funkcja służy do generowania struktury danych uczących i raportowania o ilości wykorzystanych zdjęć treningowych, dzięki czemu można w prosty sposób wykluczyć niepotrzebne obrazy.

6.5. Funkcja test_images

Fragment komentarzy funkcji opisujący jej argumenty wejściowe i wyjściowe:

```
function [scores, stats] = test_images(Mdl, face_detector, path_of_test_images)
%test_images test model
%input:
%   Mdl: trained model
%   face_detector: trained object model of vision.CascaseClassifierObject
%   path_of_test_images: path with test images
%return:
%   scores: struct of scores and success
%       .success = {0 1 0} % 0 - failure, 1 - correct recognition
%       .score = {score score score} % score from predict matlab function
%   stats: struct of statistics
%       .TP = {TP for s0, ... for s1, ...}
%       .FN = {numel(images) - TP - cort for s0, ... for s1, ...}
%       .label = {"s0", "s1", ...}
%       .sen = {sensitivity % for s0, ...}
%       .miss = {miss rate % for s0, ...}
```

Funkcja odczytuje wszystkie obrazy testowe, przeprowadza klasyfikację i na bieżąco oblicza statystyki dla każdej z testowanych klas.

7. Instrukcja obsługi

Przygotowane oprogramowanie umożliwia podstawienie własnych danych i przeszkolenie domyślnie skonfigurowanego klasyfikatora SVM.

7.1. Przygotowanie danych uczących i testowych

Dane do nauki i testów należy umieścić w folderach odpowiednio: “training_images” oraz “test_images”. W obu folderach, obrazy odpowiadające określonym tożsamościom powinny znajdować się w odpowiadających sobie folderach s0, s1 itd...

7.2. Przygotowanie klasyfikatora

W celu zastosowania skryptu *main.m*, należy uruchomić jego ostatnią wersję, dostępną w czerwcu 2020 roku na repozytorium w serwisie github:
github.com/KamilSuchanek95/Biometria-twarzy-UM-m,
zakładając, że w odpowiednich folderach mamy już przygotowane obrazy do nauki i testów.

W wyniku uruchomienia programu, w oknie Command Window powinny pojawić się następujące komunikaty:

the percentage of images used from s0 is:100.00% (or 35 out of 35)

the percentage of images used from s1 is:100.00% (or 29 out of 29)

...

Oraz podczas procesu testowania:

test image folder: s0

incorrect classification as s3 on image: t0 (14).jpg score: -0.015977

incorrect classification as s3 on image: t0 (16).jpg score: -0.017234

incorrect classification as s4 on image: t0 (17).jpg score: -0.023112

incorrect classification as s3 on image: t0 (20).jpg score: -0.024862

incorrect classification as s1 on image: t0 (7).jpg score: -0.024719

s0 TP = 30 FN = 5 sensitivity = 0.85714 miss rate = 0.14286

==========

...

W wyniku pracy programu powstał przeszkolony obiekt Mdl oraz face_detector widoczne w oknie Workspace. Aby uniknąć ponownych obliczeń można zapisać owe zmienne do pliku poprzez zaznaczenie ich, kliknięcie prawym przyciskiem myszy oraz wybranie poprawnej opcji. Tak zapisane zmienne będzie można załadować przy pomocy kodu (funkcja load) oraz przeciągając owe do okna Workspace albo Command Window w dowolnej chwili.

7.3. Zastosowanie klasyfikatora

W ten sposób przygotowany klasyfikator i detektor można zastosować poprzez załadowanie przygotowanych wcześniej zmiennych oraz uruchomienie kolejno sekcji kodu o nazwach (tylko sekcji, nie całego kodu, chyba że niepotrzebne sekcje zostały własnoręcznie zakomentowane):

- %% Add functions to path
- %% Get a picture from webcam

Po uruchomieniu tej sekcji, pojawi się podgląd na żywo z kamerki internetowej oraz dodatkowe okno przedstawiające aktualnie uchwycone zdjęcie, program zapyta w oknie Command Window, czy takie ujęcie nam odpowiada:

Does this image look good?

(yes/no) yes

W tym przypadku wprowadzono słowo “yes” i zatwierdzono przyciskiem ENTER. Można wpisać również “no”, lub cokolwiek poza “yes” i po zatwierdzeniu pojawi się nowy podgląd uchwyconego ujęcia (samo zatwierdzenie bez wpisywania czegokolwiek również poskutkuje wykonaniem kolejnego zrzutu z kamerki).

- %% recognize my_image

Po uruchomieniu tej sekcji kodu program przystąpi do wykrywania twarzy, w przypadku braku wykrytej twarzy zostaniemy o tym poinformowani w oknie

Command Window. W tym przypadku zaobserwujemy również błąd programu ze względu na wykonanie predykcji w następnej linijce, W przypadku wątpliwości czy nasze zdjęcie będzie miało dość dobrą jakość można więc utworzyć nową sekcję dla kolejnej linii kodu. Aby uzyskać informację o wykrytej osobie należy wpisać w okno Command Window “label”:

```
>> label

label =

1×1 cell array

{'s6'}
```

W tym przypadku wskazało na tożsamość s6 (Kamil Suchanek, poprawnie). Na własne potrzeby można wprowadzić mapowanie wartości i osób, które umieściliśmy w poszczególnych folderach.

8. Podsumowanie

W ramach realizacji projektu powstał prototyp rozwiązania elementu systemu biometrii twarzy opartego o transformację HOG i SVM. Aktualne repozytorium projektu zawiera skrypty zgodne z funkcjonalnym opisem w tym dokumencie, zawierające drobne zmiany zmiennych i szyku linii kodu w celu poprawy czytelności. Założenia projektu zostały spełnione w podstawowym zakresie.

Bibliografia

1. www.en.wikipedia.org/wiki/Local_binary_patterns [dostęp: 17.06.2020]
2. Zacniewski A. - System Rozpoznawania Cyfr Oparty Na Histogramie Zorientowanych Gradientów - Akademia Marynarki Wojennej w Gdyni, Wydział Nawigacji i Uzbrojenia Okrętowego.
3. Guodong Guo, Stan Z. Li,, Kapluk Chan - Face Recognition by Support Vector Machines - School of Electrical and Electronic Engineering Nanyang Technological University, Singapore.
4. www.mathworks.com/help/stats/fitcecoc.html#d120e263351 [dostęp: 17.06.2020]