

Politechnika poznańska

Wydział Elektryczny



Rozpoznawanie twarzy i śledzenie ruchu

Agnieszka Gregulec
Hubert Krzepkowski
Bartosz Michta
Emilian Ossowski

Spis treści

Szkic harmonogramu	3
Opis i uzasadnienie wyboru tematu	3
Podział prac.....	3
Funkcjonalności	3
Wybrane technologie	4
Architektura	4
Rozpoznawanie twarzy	7
Współczesne rozpoznawanie twarzy przy użyciu uczenia maszynowego.	7
Rozpoznawanie twarzy - krok po kroku.....	7
Instalacja biblioteki na Ubuntu	17
Interesujące napotkane problemy i ich rozwiązania	20
Opis najważniejszych funkcji programu.....	20
Instrukcja użytkownika	21
Dodawanie osoby	21
Sprawdzanie obecności:	23
Testy.....	26
Podsumowanie	29
Perspektywy rozwoju.....	30

Szkic harmonogramu

05.04.2018 r. - prezentacja wstępna

19.04.2018 r. - I postęp prac

17.05.2018 r. - II postęp prac

14.06.2018 r. - prezentacja wyników pracy

Opis i uzasadnienie wyboru tematu

Celem projektu jest stworzenie systemu do identyfikacji osób wchodzących do pomieszczenia obserwowanego przez kamerę IP. Wybraliśmy podany temat, ponieważ:

- temat jest ciekawy
- styczność na czwartym semestrze studiów z przetwarzaniem obrazów
- chęć poznania nowych technologii

Podział prac

- zapoznanie się z tematyką i poszerzenie wiadomości
<https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>
- znalezienie przydatnych bibliotek
- implementacja
 - Front-end oraz rozpoznawanie twarzy
 - Back-end oraz dopasowywanie twarzy z bazy studentów z twarzami osób na zajęciach
- interfejs
- testy
- dokumentacja

Funkcjonalności

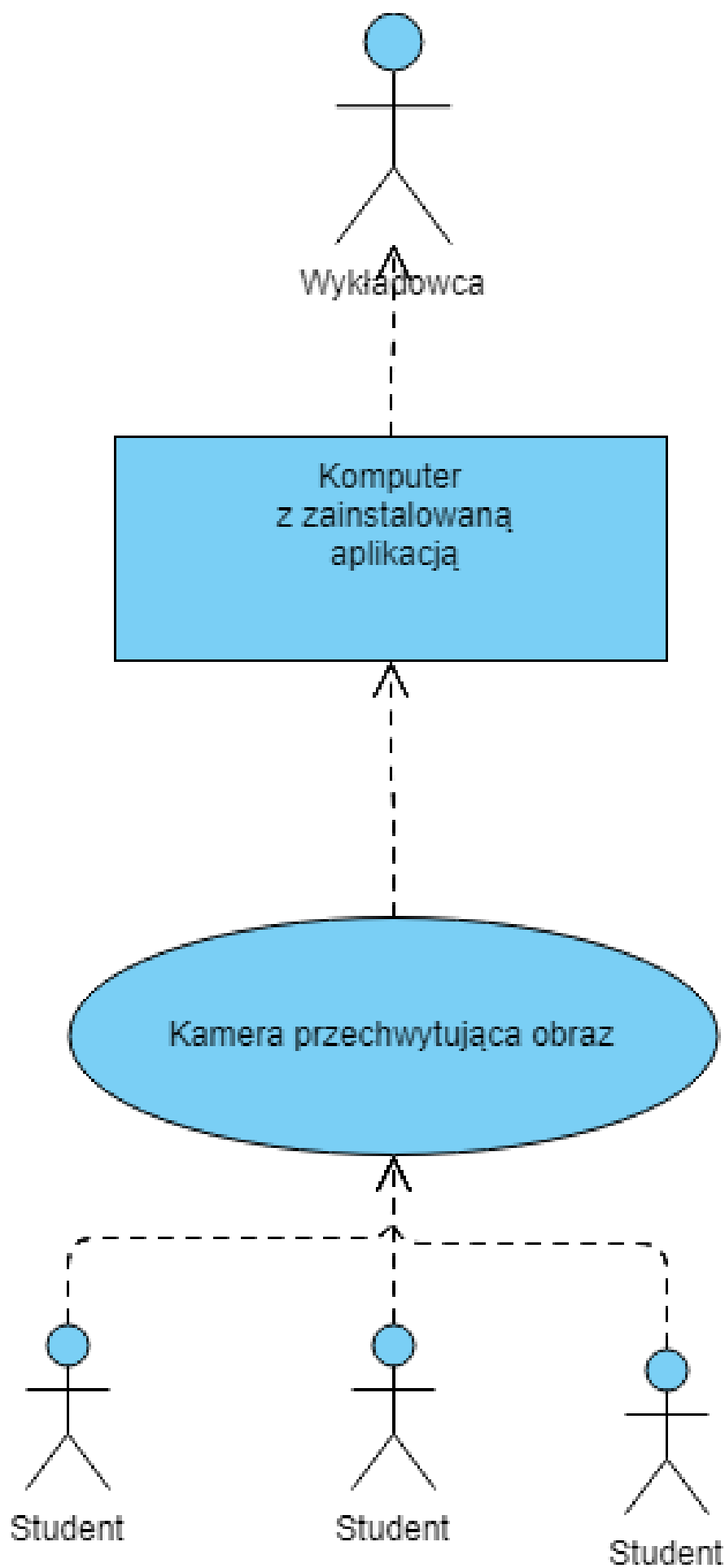
- aplikacja desktopowa
Aplikację można uruchomić na komputerze prowadzącego z podłączoną lub wbudowaną kamerką.
- zapisywanie na zajęcia
Aplikacja rozpoznaje twarz, którą można zapisać tj. zdjęcie z nazwą.
- sprawdzanie obecności na zajęciach
Na podstawie zapisanych osób można sprawdzić czy pojawiły się na zajęciach.

Wybrane technologie

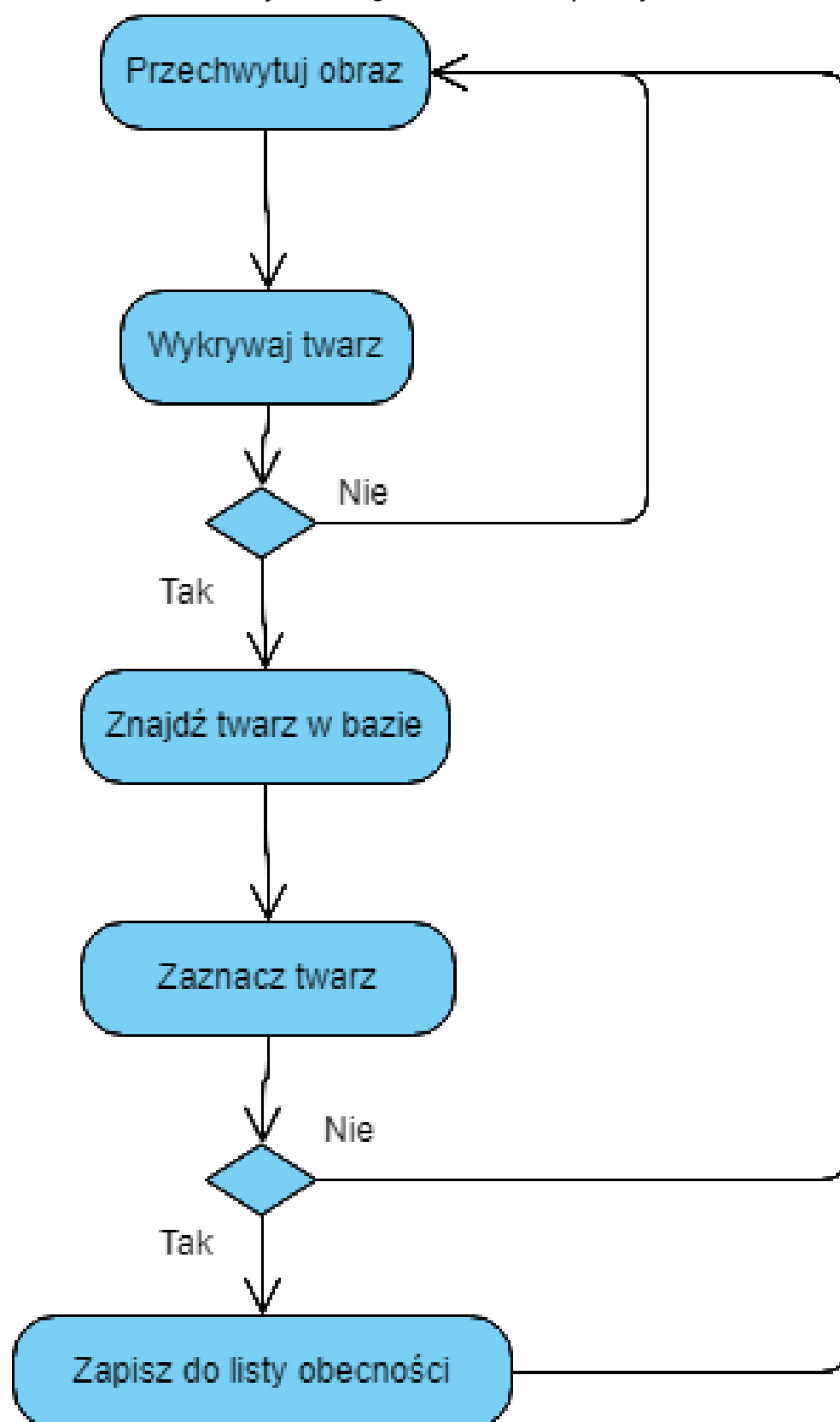
- OpenCV
Biblioteka do rozpoznawania twarzy, z którą pracowaliśmy przy rozpoznawaniu obrazów.
- biblioteka ageitgey/face_recognition
https://github.com/ageitgey/face_recognition/blob/master/README.md
Biblioteka znaleziona na potrzeby projektu.
- Python 3.3+ lub Python 2.7
Wybrany został język Python ponieważ, wcześniej z nim pracowaliśmy przy przetwarzaniu obrazów oraz biblioteka, z której korzystamy została napisana dla tego języka.
- Linux
System operacyjny został wybrany ze względu na łatwość instalacji biblioteki.

Architektura

Przy projektowaniu aplikacji zostały wykonane dwa diagramy. Pierwszy diagram przedstawia użycie aplikacji. Drugi diagram jest diagramem aktywności, czyli pokazuje co aplikacja wykonuje.



Rys. 1 Diagram działania aplikacji



Rys. 2 Diagram aktywności aplikacji

Rozpoznawanie twarzy

Współczesne rozpoznawanie twarzy przy użyciu uczenia maszynowego.

Rozpoznawanie twarzy jest serią kilku powiązanych problemów:

- Na początku spójrz na zdjęcie i znajdź wszystkie twarze
- Po drugie, skup się na każdej twarzy, żeby być w stanie zrozumieć, że nawet jeśli twarz jest zwrócona w dziwnym kierunku lub w złym oświetleniu, to wciąż ta sama osoba.
- Po trzecie, możesz wybierać unikalne cechy twarzy, które możesz użyć, by odróżnić ją od innych ludzi - np. Jak duże są oczy, jak długa jest twarz itp.
- Na koniec porównaj unikalne cechy tej twarzy z wszystkimi osobami, które znasz, aby określić dane osoby.

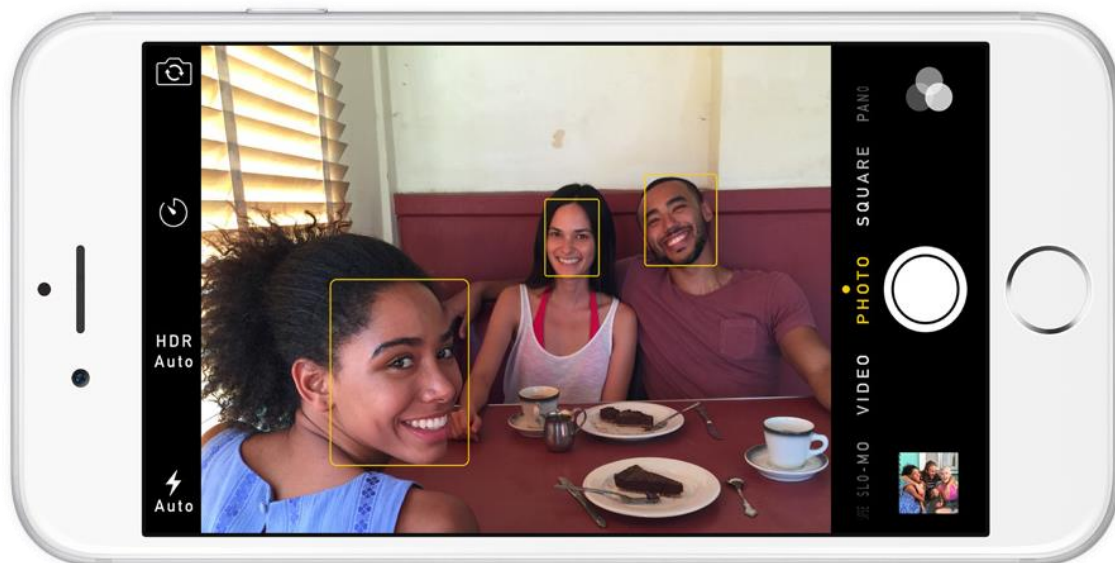
Mózg jest stworzony, aby zrobić to wszystko automatycznie i natychmiastowo. Komputery nie są w stanie wykonywać tego rodzaju zadania, więc musimy nauczyć ich, jak robić każdy krok w tym procesie osobno.

Musimy zbudować potok, w którym osobno rozwiązujemy każdy krok rozpoznawania twarzy i przekazujemy wynik bieżącego kroku do następnego kroku. Innymi słowy, połączymy razem kilka algorytmów uczenia maszynowego.

Rozpoznawanie twarzy - krok po kroku

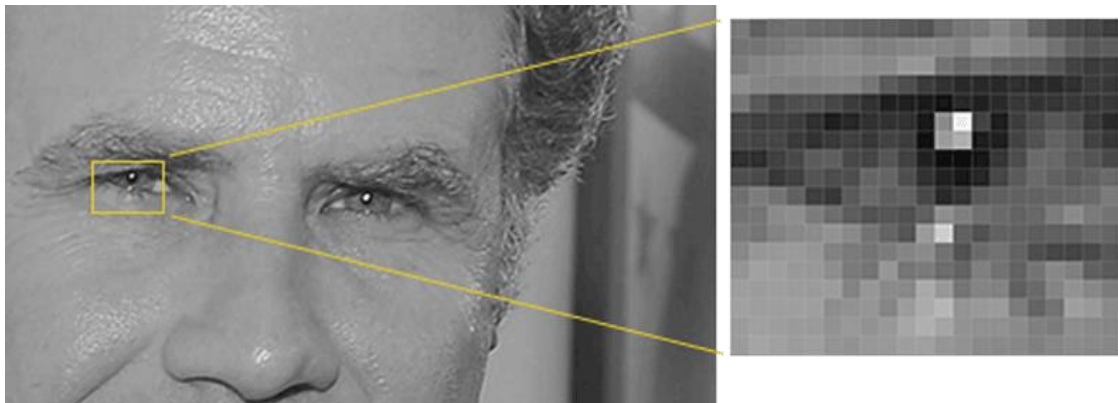
Krok 1: Wyszukiwanie wszystkich twarzy

Pierwszym krokiem w naszym procesie jest wykrywanie twarzy. Oczywiście musimy zlokalizować twarze na zdjęciu, zanim będziemy mogli spróbować je odróżnić.

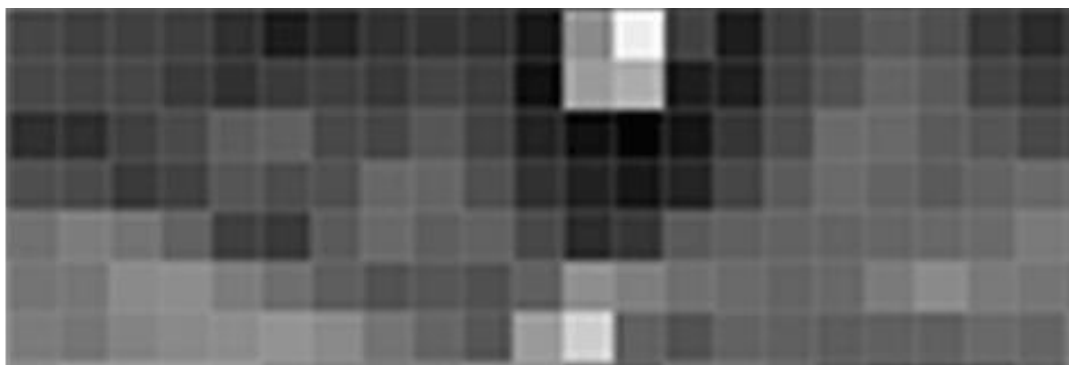


Wykrywanie twarzy to doskonała funkcja dla aparatów fotograficznych. Gdy aparat może automatycznie wykrywać twarze to może się upewnić, że wszystkie są ostre przed zrobieniem zdjęcia. Ale użyjemy go do innego celu - znalezienia obszarów obrazu, który chcemy przekazać, do następnego etapu w naszym ciągu. Detekcja twarzy stała się głównym nurtem na początku lat 2000, gdy Paul Viola i Michael Jones wymyślili sposób wykrywania twarzy, które były wystarczająco szybkie, by działały na tanich aparatach. Jednak obecnie istnieją znacznie bardziej niezawodne rozwiązania. Zamierzamy użyć metody wynalezionej w 2005 r. o nazwie Histogram of Oriented Gradients - lub po prostu HOG w skrócie. Aby znaleźć twarze na obrazie, zaczniemy od uczynienia naszego obrazu czarno-białym, ponieważ nie potrzebujemy danych o kolorze, aby znaleźć twarze.

Następnie przyjrzymy się każdemu pojedynczemu pikselowi na naszym obrazie po jednym na raz. Dla każdego pojedynczego piksela chcemy spojrzeć na piksele otaczające go bezpośrednio:



Naszym celem jest ustalenie, jak ciemny jest bieżący piksel w porównaniu do pikseli bezpośrednio go otaczających. Następnie chcemy narysować strzałkę pokazującą, w którym kierunku obraz robi się ciemniejszy:



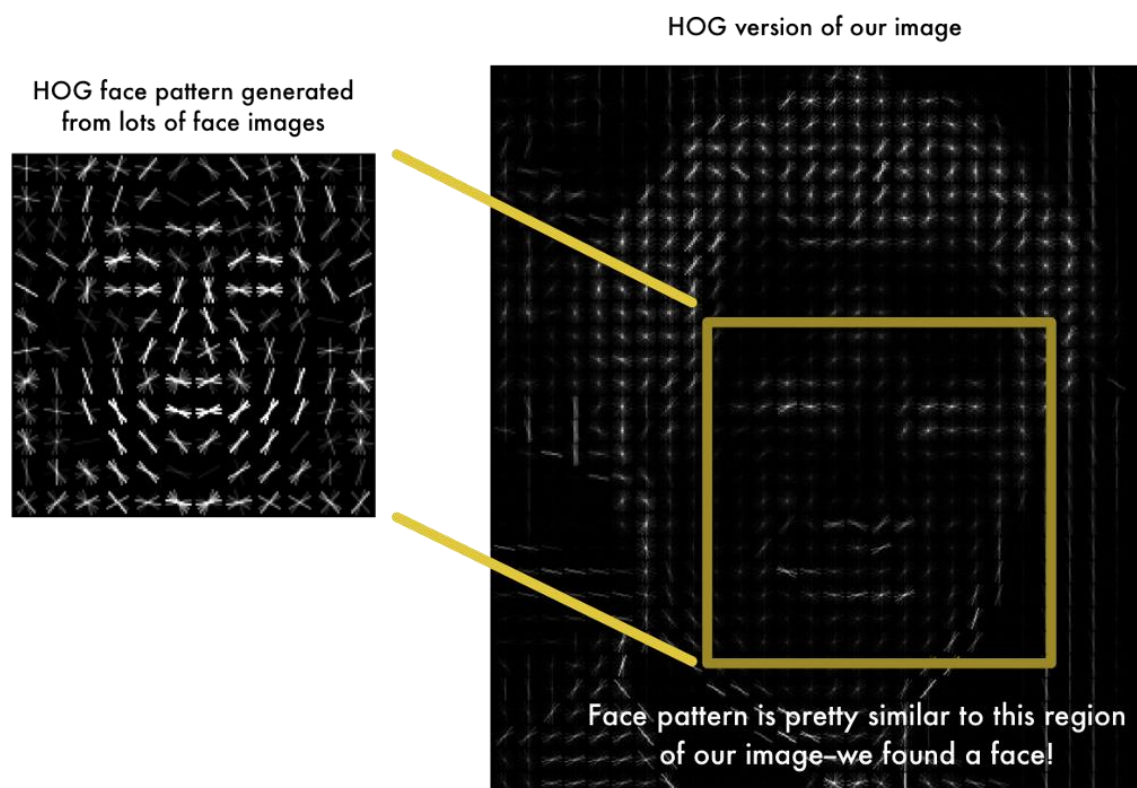
Jeśli powtórzysz ten proces dla każdego pojedynczego piksela na obrazie to skończymy na tym, że każdy piksel zostanie zastąpiony przez strzałkę. Te strzałki są nazywane gradientami i pokazują przepływ od światła do ciemności na całym obrazie.

Może się to wydawać przypadkową czynnością, ale istnieje naprawdę dobry powód do zamiany pikseli na gradienty. Jeśli przeanalizujemy piksele bezpośrednio, naprawdę ciemne obrazy i naprawdę jasne obrazy tej samej osoby będą miały zupełnie inne wartości pikseli. Ale biorąc pod uwagę tylko kierunek, w którym zmienia się jasność, zarówno naprawdę ciemne obrazy, jak i naprawdę jasne obrazy, będą miały dokładnie taką samą reprezentację. To sprawia, że problem jest łatwiejszy do rozwiązania. Ale zapisanie gradientu dla każdego pojedynczego piksela daje nam zbyt dużo szczegółów. Byłoby lepiej, gdybyśmy mogli zobaczyć podstawowy przepływ jasności / ciemności na wyższym poziomie, abyśmy mogli zobaczyć podstawowy wzór obrazu. Aby to zrobić, podzielimy obraz na małe kwadraty o wymiarach 16x16 pikseli każdy. W każdym kwadracie zliczymy liczbę gradientów w każdym głównym kierunku (ile punktów w górę, w prawo, w prawo, itd.). Następnie zastąpimy ten kwadrat na obrazie najsilniejszymi kierunkami strzał.

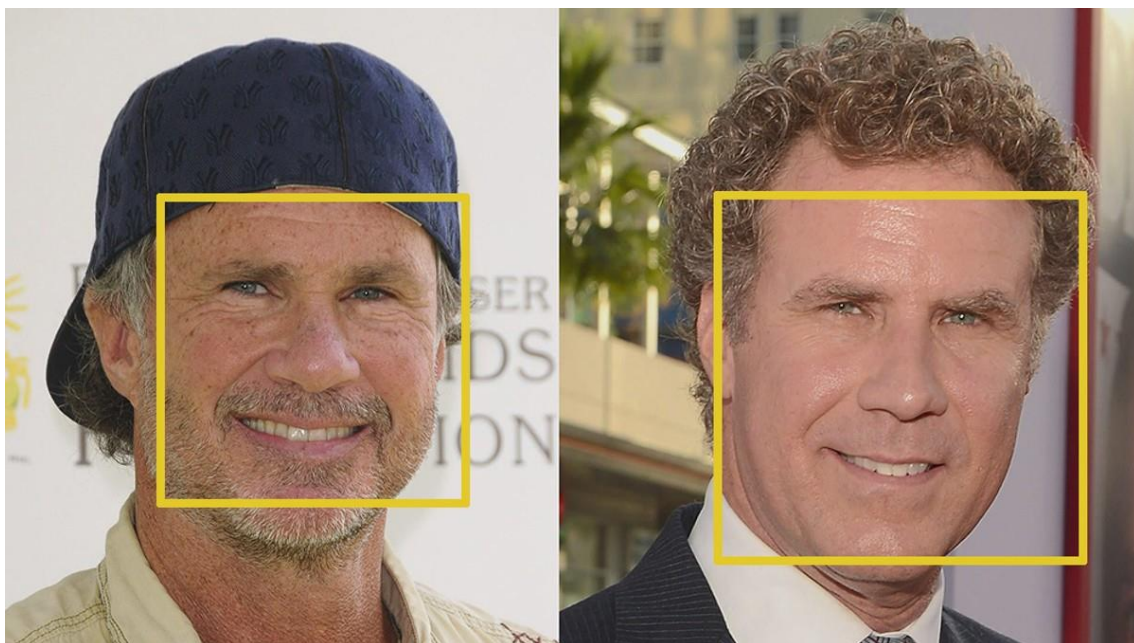
Końcowym rezultatem jest przekształcenie oryginalnego obrazu w bardzo prostą reprezentację, która w prosty sposób uchwyci podstawową strukturę twarzy:



Aby znaleźć twarze na tym obrazie HOG, wystarczy tylko znaleźć część naszego obrazu, która wygląda najbardziej podobnie do znanego wzorca HOG, który został wyodrębniony z wielu innych twarzy treningowych:

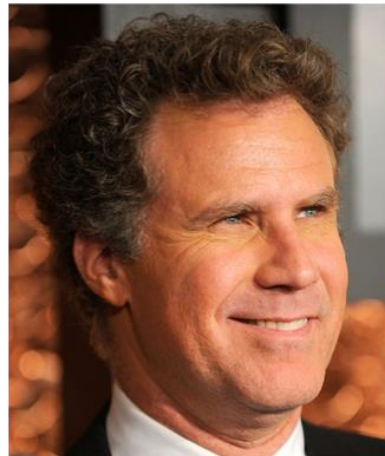
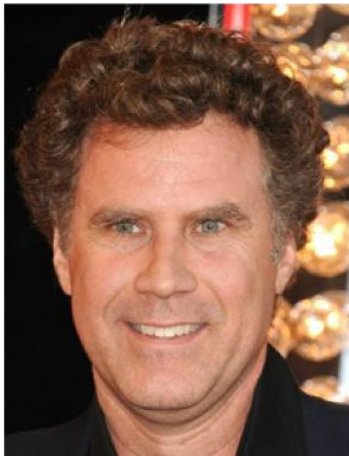


Dzięki tej technice możemy teraz łatwo znaleźć twarze w dowolnym obrazie.

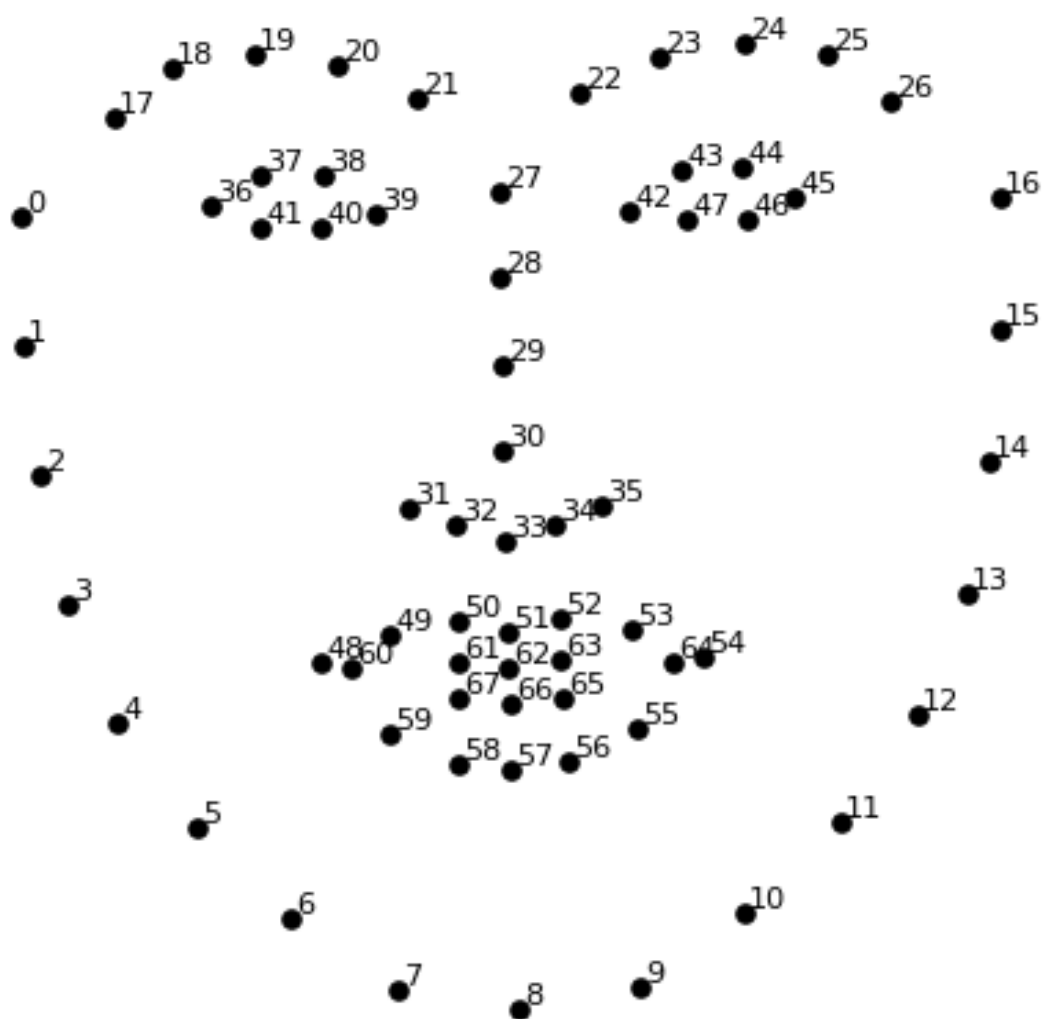


Krok 2: Pozowanie i projekcja twarzy

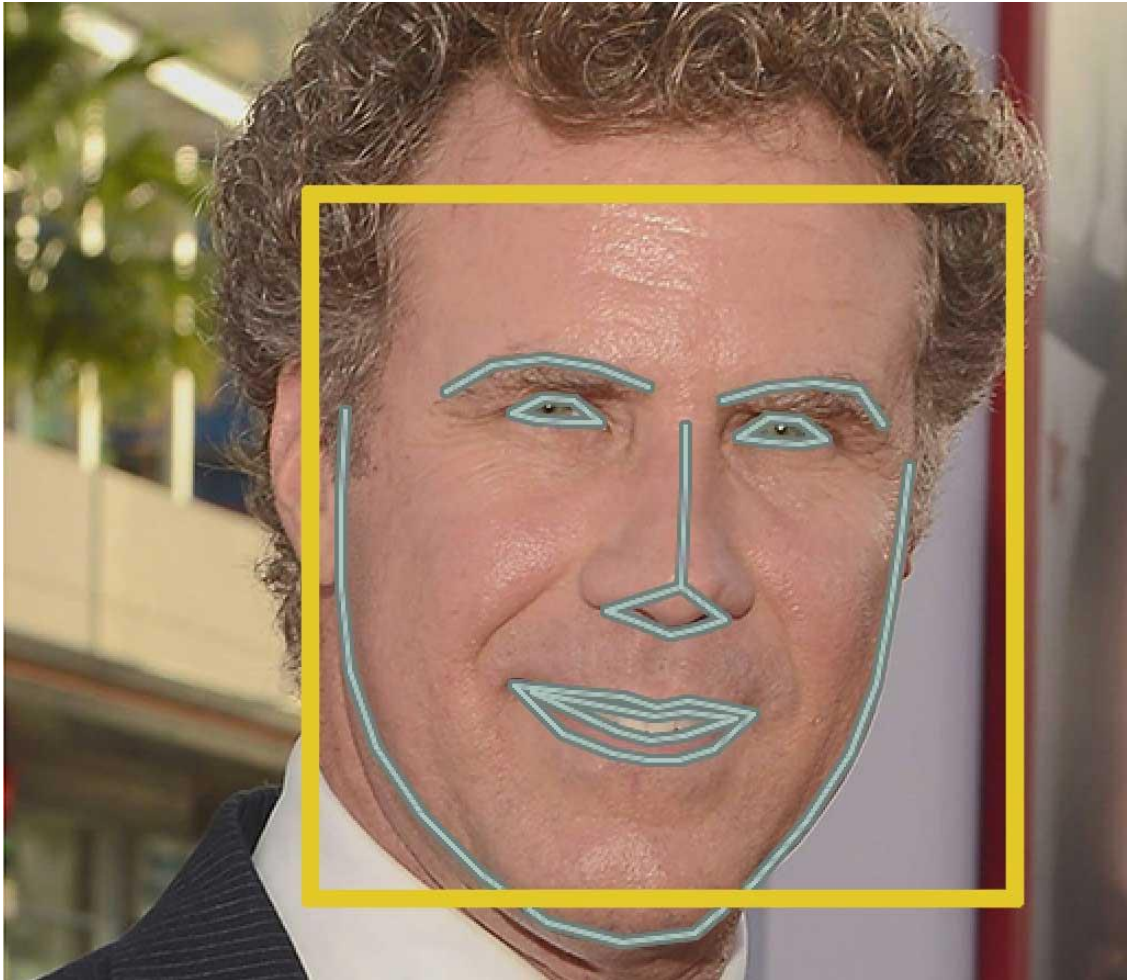
Wyróżniliśmy twarze na naszym obrazie. Teraz mamy do czynienia z problemem, że twarze zwrócone w różnych kierunkach wyglądają zupełnie inaczej niż na komputerze:



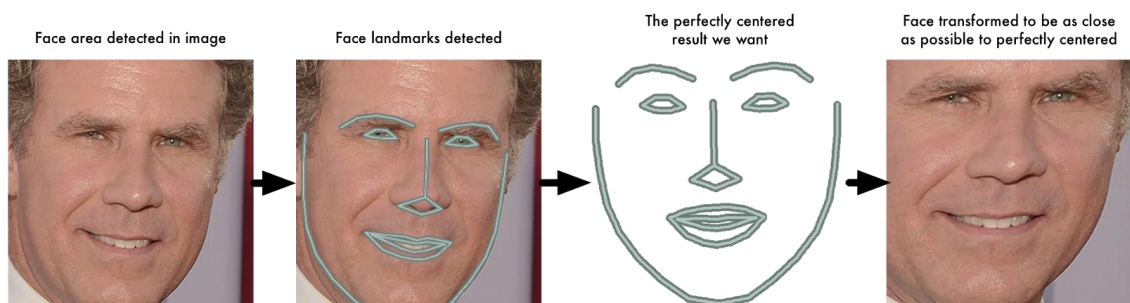
Ludzie mogą łatwo rozpoznać, że oba obrazy przedstawiają Will Ferrell, ale komputery widziałyby te obrazy jako dwóch całkowicie różnych ludzi. Aby to wyjaśnić, należy przeglądać każde zdjęcie, aby oczy i usta zawsze znajdowały się w próbce na obrazie. Ułatwi nam to porównywanie twarzy w kolejnych krokach. W tym celu użyjemy algorytmu zwanego szacowaniem punktu orientacyjnego twarzy. Jest na to wiele sposobów, ale zamierzamy zastosować podejście wynalezione w 2014 roku przez Vahida Kazemi i Josephine Sullivan. Podstawową ideą jest to, że znaleźliśmy 68 konkretnych punktów (tzw. Punktów orientacyjnych), które istnieją na każdej twarzy - na górze podbródka, na zewnętrznej krawędzi każdego oka, na wewnętrznej krawędzi każdej brwi itd. Następnie ćwiczymy maszynę algorytm uczenia się, aby móc znaleźć te 68 punktów na każdej twarzy:



Oto wynik odnalezienia 68 punktów orientacyjnych na obrazie testowym:



Teraz, gdy wiemy, że są oczy i usta, po prostu obracamy, skalujemy i ścinamy obraz tak, aby oczy i usta były wycentrowane najlepiej jak to możliwe. Nie robimy żadnych fantazyjnych 3d, ponieważ powoduje to zniekształcenia obrazu. Będziemy używać tylko podstawowych transformacji obrazu, takich jak rotacja i skala, które zachowują linie równoległe (zwane transformacjami afinicznymi):



Teraz, bez względu na to, jak odwrócona jest twarz, jesteśmy w stanie wyśrodkować oczy i usta w mniej więcej tej samej pozycji na obrazie. Dzięki temu nasz kolejny krok będzie dużo dokładniejszy.

Krok 3: Kodowanie twarzy

Najprostszym podejściem do rozpoznawania twarzy jest bezpośrednio porównanie nieznanej twarzy, którą znaleźliśmy w Kroku 2 ze wszystkimi zdjęciami, które mamy z osobami, które już zostały oznaczone. Kiedy znajdziemy wcześniej oznaczoną twarz, która wygląda bardzo podobnie do naszej nieznanej twarzy, musi to być ta sama osoba.

Z takim podejściem jest jednak naprawdę ogromny problem. Witryna taka jak na przykład Facebook z miliardami użytkowników i trylionowymi zdjęciami nie może przechodzić przez każdą poprzednio otagowaną twarz, aby porównać ją z każdym nowo przesłanym obrazem. To by zajęło zbyt dużo czasu. Muszą być w stanie rozpoznać twarze w ciągu milisekund, a nie godzin.

Potrzebujemy sposobu na wyodrębnienie kilku podstawowych pomiarów z każdej twarzy. Wtedy moglibyśmy zmierzyć naszą nieznaną twarz w ten sam sposób i znaleźć znaną twarz z najbliższymi pomiarami. Na przykład, możemy zmierzyć rozmiar każdego ucha, odstęp między oczami, długość nosa, itp

Najbardziej niezawodny sposób pomiaru twarzy

Takie pomiary powinniśmy zbierać z każdej twarzy, aby zbudować naszą znaną bazę danych twarzy.

Okazuje się, że pomiary, które wydają się nam oczywiste dla ludzi (np. kolor oczu), nie mają sensu dla komputera oglądającego pojedyncze piksele na obrazie. Naukowcy odkryli, że najdokładniejszym podejściem jest umożliwienie komputerowi obliczenia pomiarów.

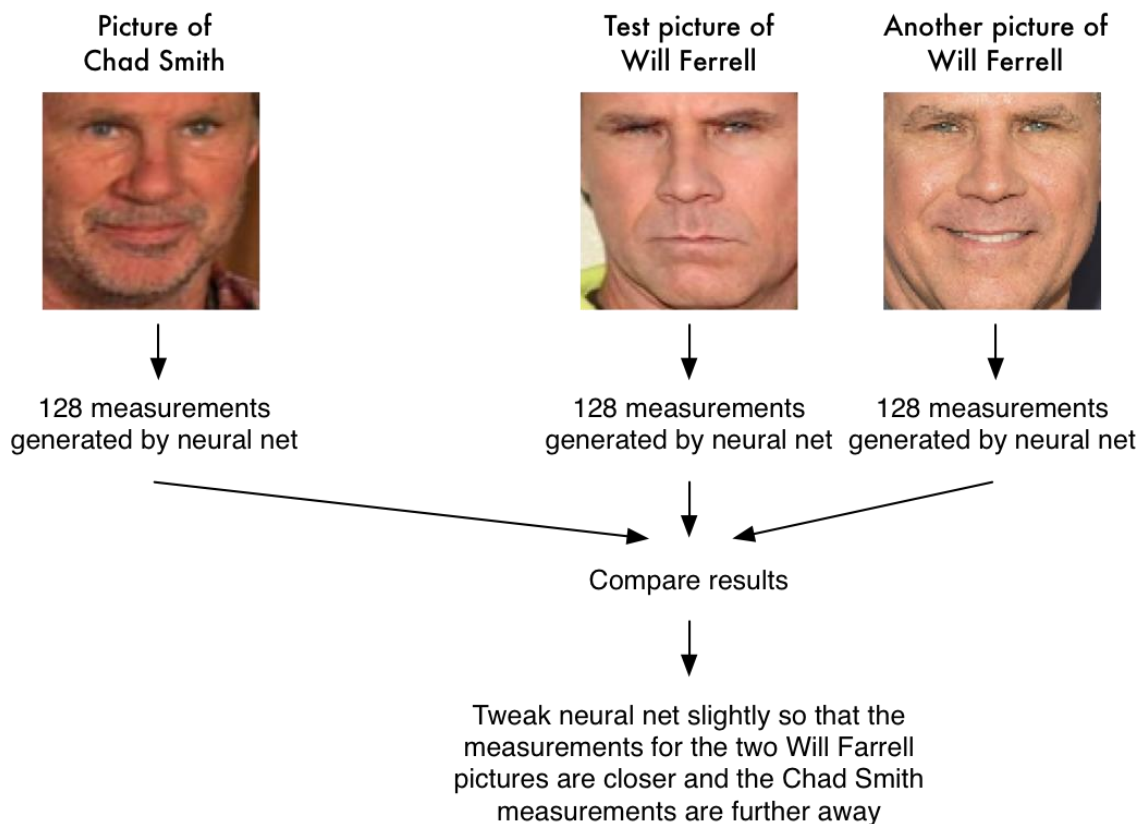
Rozwiązaniem jest szkolenie Deep Convolutional Neural Network (podobnie jak w części 3). Ale zamiast szkolić sieć, aby rozpoznać obiekty obrazów, takie jak zrobiliśmy ostatni raz, zamierzamy ją wyćwiczyć, aby wygenerować 128 pomiarów dla każdej twarzy.

Proces treningu działa, patrząc na 3 obrazy twarzy na raz:

- Załaduj szkolny obraz twarzy znanej osoby
- Załaduj kolejne zdjęcie tej samej znanej osoby
- Załaduj zdjęcie zupełnie innej osoby

Następnie algorytm analizuje pomiary generowane obecnie dla każdego z tych trzech obrazów. Następnie poprawia nieco sieć neuronową, aby upewnić się, że pomiary generowane dla # 1 i # 2 są nieco bliżej, upewniając się, że pomiary dla # 2 i # 3 są nieco dalej od siebie:

A single 'triplet' training step:



Po powtórzeniu tego kroku miliony razy dla milionów obrazów tysięcy różnych osób, sieć neuronowa uczy się rzetelnie generować 128 pomiarów dla każdej osoby. Każde dziesięć różnych zdjęć tej samej osoby powinno dać z grubsza te same wymiary.

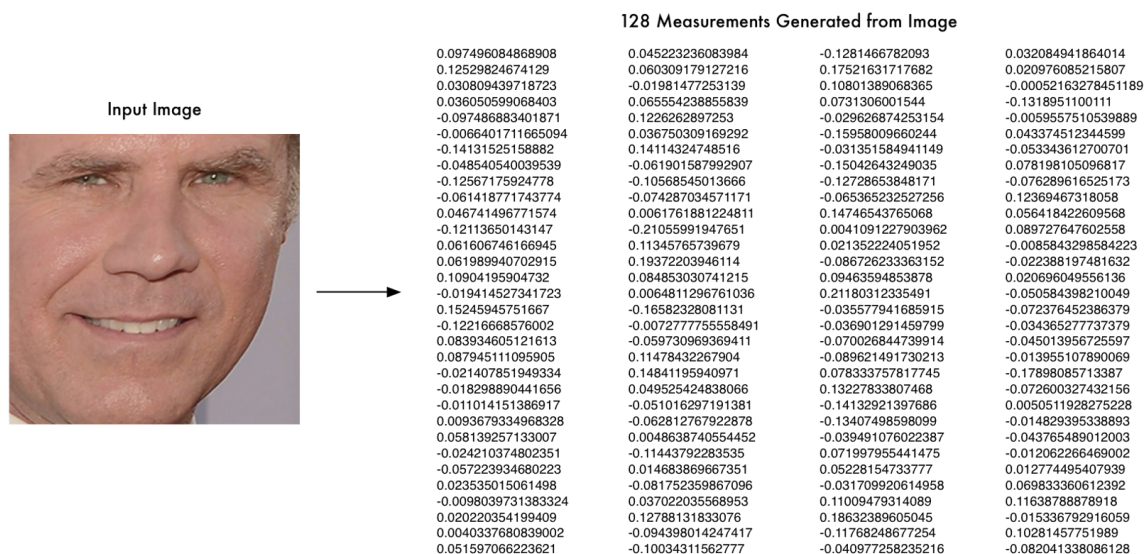
Pomysł zredukowania skomplikowanych danych surowych, takich jak obrazek, do listy liczb generowanych komputerowo, ma wiele wspólnego z uczeniem maszynowym (zwłaszcza w tłumaczeniu językowym). Dokładne podejście do twarzy, z którego korzystamy, zostało wynalezione w 2015 roku przez naukowców z Google, ale istnieje wiele podobnych podejść.

Kodowanie naszego wizerunku twarzy

Proces szkolenia splotowej sieci neuronowej wymaga dużej ilości danych i mocy komputera. Nawet przy kosztownej karcie graficznej NVIDIA Telsa potrzeba 24 godzin ciągłego treningu, aby uzyskać dobrą celność.

Ale gdy sieć zostanie już przeszkolona, może generować pomiary dla każdej twarzy. Ten krok należy wykonać tylko raz. Na szczęście dla nas, ludzie z OpenFace już to zrobili i opublikowali kilka przeszkolonych sieci, z których możemy bezpośrednio korzystać.

Więc wszystko, co musimy zrobić sami, to uruchomić nasze obrazy twarzy poprzez ich wcześniej wyszkoloną sieć, aby uzyskać 128 pomiarów dla każdej twarzy. Oto wymiary naszego testowego obrazu:



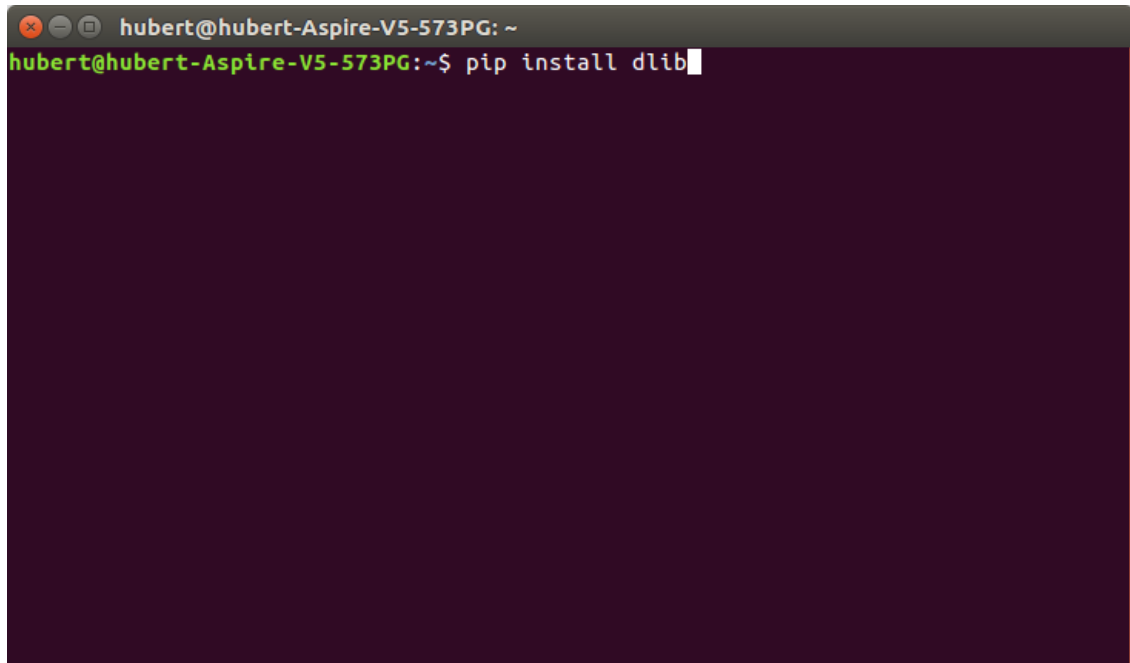
Zbudowany przy użyciu najnowocześniejszego rozpoznawania twarzy firmy dib, zbudowanego przy pomocy uczenia maszynowego. Model ma dokładność 99,38% w benchmarkach Labelled Faces in the Wild. Znajdowanie funkcji twarzy jest bardzo przydatne w wielu ważnych sprawach, ale można również użyć do zabawnych rzeczy, takich jak stosowanie makijażu cyfrowego.

Cechy:

- znajdowanie wszystkich twarzy które pojawiły się na obrazku
- znajdź i zmodyfikuj cechy twarzy na zdjęciach - uzyskaj lokalizacje i zarysy oczu, nosa, ust i podbródka każdej osoby.
- identyfikacja twarzy na zdjęciach - rozpoznaj, kto pojawia się na każdym zdjęciu.

Instalacja biblioteki na Ubuntu

1. Aby rozpocząć instalację biblioteki należy w pierwszy kroku uruchomić konsolę i wykonać polecenie: `pip install dlib`.

A screenshot of a terminal window with a dark purple background. The title bar at the top shows window control icons and the text 'hubert@hubert-Aspire-V5-573PG: ~'. The terminal prompt is 'hubert@hubert-Aspire-V5-573PG:~\$' in green. The command 'pip install dlib' is typed in white, followed by a white cursor. The rest of the terminal area is empty.

```
hubert@hubert-Aspire-V5-573PG: ~  
hubert@hubert-Aspire-V5-573PG:~$ pip install dlib
```

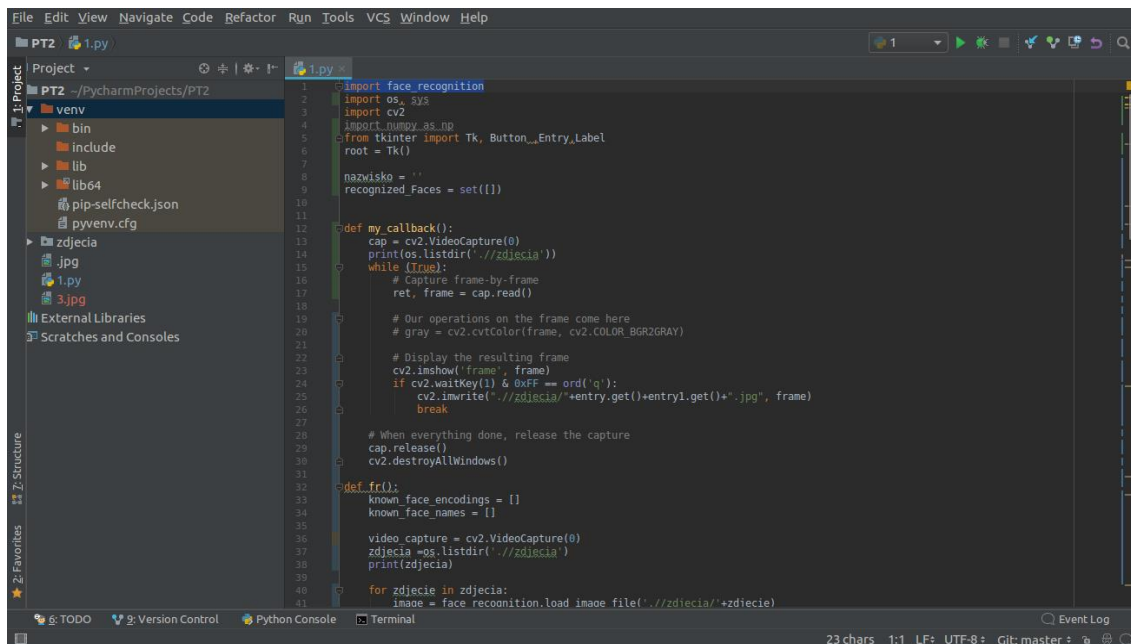
Rys. 3 Konsola w systemie Ubuntu z poleceniem `pip install dlib`

2. Po udanym zainstalowaniu jako następny krok należy wykonać polecenie: `pip3 install face_recognition`. W przypadku Pythona 2 należy użyć polecenia: `pip2 install face_recognition`.

```
hubert@hubert-Aspire-V5-573PG: ~  
Stored in directory: /home/hubert/.cache/pip/wheels/df/c1/5b/42d688c523f80cd0d4e5d3224fd39ec451de54583d22487c3c  
Successfully built dlib  
Installing collected packages: dlib  
  
You are using pip version 10.0.0, however version 10.0.1 is available.  
You should consider upgrading via the 'pip install --upgrade pip' command.  
hubert@hubert-Aspire-V5-573PG:~$  
hubert@hubert-Aspire-V5-573PG:~$ pip3 install face_recognition  
Collecting face_recognition  
  Downloading https://files.pythonhosted.org/packages/28/10/f153bbbc218fc169768aa1c02f2e9178e9241e4af8da56289bdca2c0c217/face_recognition-1.2.2-py2.py3-none-any.whl  
Collecting Pillow (from face_recognition)  
  Downloading https://files.pythonhosted.org/packages/07/52/8e27b9c54cb70d379244771a58483928b3a02db3c657d466ed84eb18f22b/Pillow-5.1.0-cp35-cp35m-manylinux1_x86_64.whl (2.0MB)  
    100% |████████████████████████████████████████| 2.0MB 542kB/s  
Collecting dlib>=19.7 (from face_recognition)  
  Downloading https://files.pythonhosted.org/packages/d5/5c/aa64510aa354d562ecba7edecd500b020280741140e5d91ff5ec1c6f8289/dlib-19.13.1.tar.gz (3.3MB)  
    100% |████████████████████████████████████████| 3.3MB 325kB/s  
Collecting face-recognition-models>=0.3.0 (from face_recognition)
```

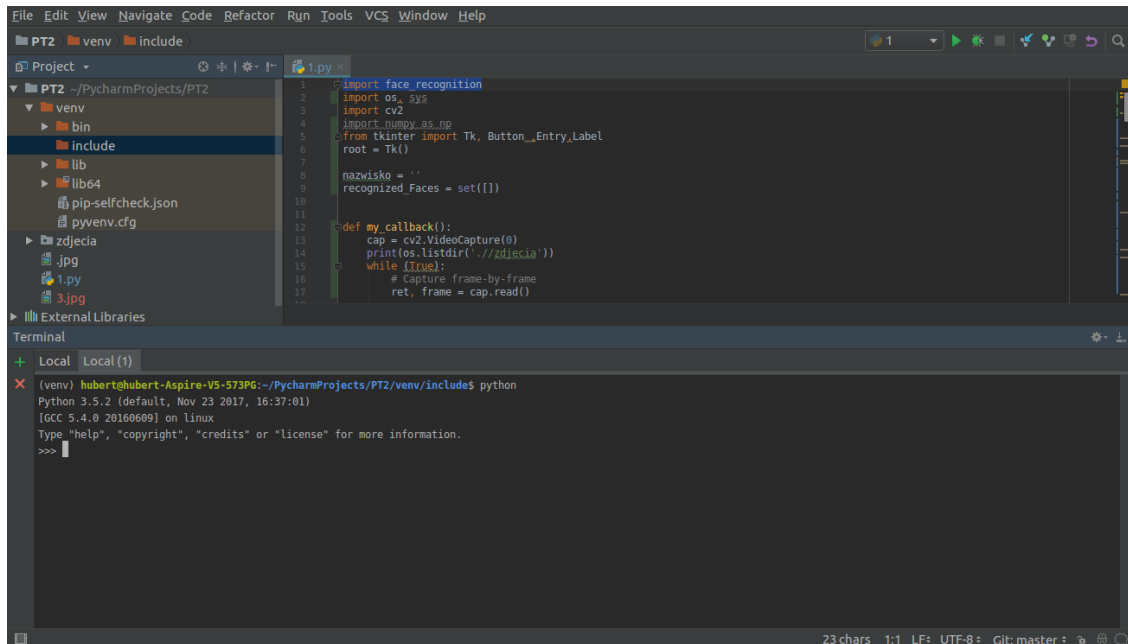
Rys. 4 Konsola w systemie Ubuntu z poleceniem `pip3 install face_recognition`

3. Aby użyć bibliotekę w projekcie należy zaimportować ją instrukcją: `import face_recognition`



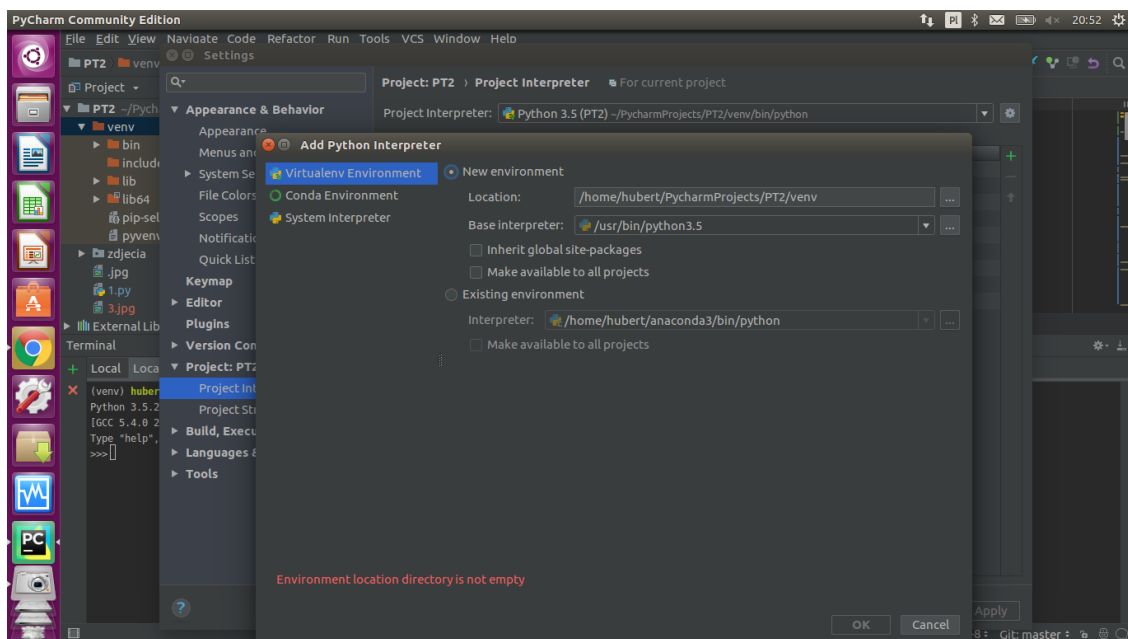
Rys. 5 Użycie biblioteki w projekcie

4. Jeśli wszystko poprawnie się zainstalowało, a nadal nie możemy zaimportować biblioteki należy sprawdzić z której wersji Pythona korzysta nasz projekt. W tym celu należy otworzyć terminal i wykonać polecenie: python.



Rys. 6 Sprawdzanie zainstalowanej wersji Pythona

5. W przypadku jeśli projekt korzysta z innej wersji niż ta, dla której zainstalowaliśmy bibliotekę face_recognition należy zmienić wersję Pythona dla projektu. W programie PyCharm należy otworzyć ustawienia programu, a następnie dodać interpreter Pythona:



Rys. 7 Dodawanie interpretera Pythona w programie PyCharm

Interesujące napotkane problemy i ich rozwiązania

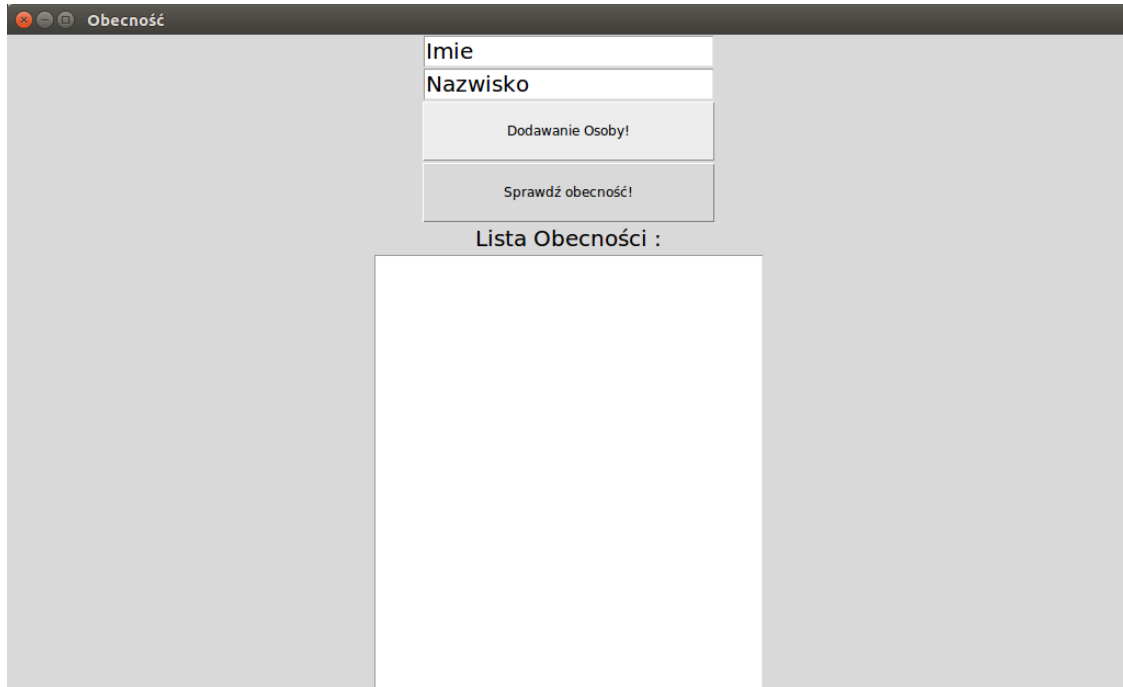
- niezgodność wersji zainstalowanego Pythona z wersją Pythona, w której została napisana biblioteka
Pierwszym napotkanym problemem był brak zgodności wersji Pythona zainstalowanego z wersją Pythona, w której została napisana biblioteka. Rozwiązanie tego problemu znajduje się w rozdziale Architektura w podrozdziale Instalowanie biblioteki na Ubuntu od punktu 4.
- rozpoznawanie twarzy na zdjęciu
Kolejnym problemem jest rozpoznawanie twarzy z wydrukowanego zdjęcia. Rozwiązaniem tego problemu jest sprawdzanie głębi, ale niestety nie udało się tego poprawić, ponieważ jest to trudne do wykonania.
- obsługa systemu kontroli wersji git
Brak synchronizacji w aktualizacji projektu doprowadził do powstania wielu różnych wersji i spowodował utrudnienia w ustaleniu wspólnej wersji programu.

Opis najważniejszych funkcji programu

- Photo
Funkcja służąca do zapisania zdjęcia zrobionego przy użyciu kamery oparta na OpenCV.
- face_recognition.face_locations
Funkcja służąca do odnajdywania lokacji twarzy na obrazie jako parametr przyjmuje pojedynczą ramkę z obrazu wideo.
- face_recognition.face_encodings
Funkcja, która generuje kodowania dla twarzy rozpoznanych na obrazie, jako parametry przyjmuje pojedynczą ramkę z obrazu wideo i lokalizację twarzy lub tylko zdjęcie.
- face_recognition.compare_faces
Funkcja porównująca podane w parametrach kodowania twarzy generowane z obrazu z kamery z tymi które zostały wczytane ze zdjęć.
- face_recognition.load_image_file
Funkcja pobiera zdjęcie z podanej w parametrze ścieżki.

Instrukcja użytkowania

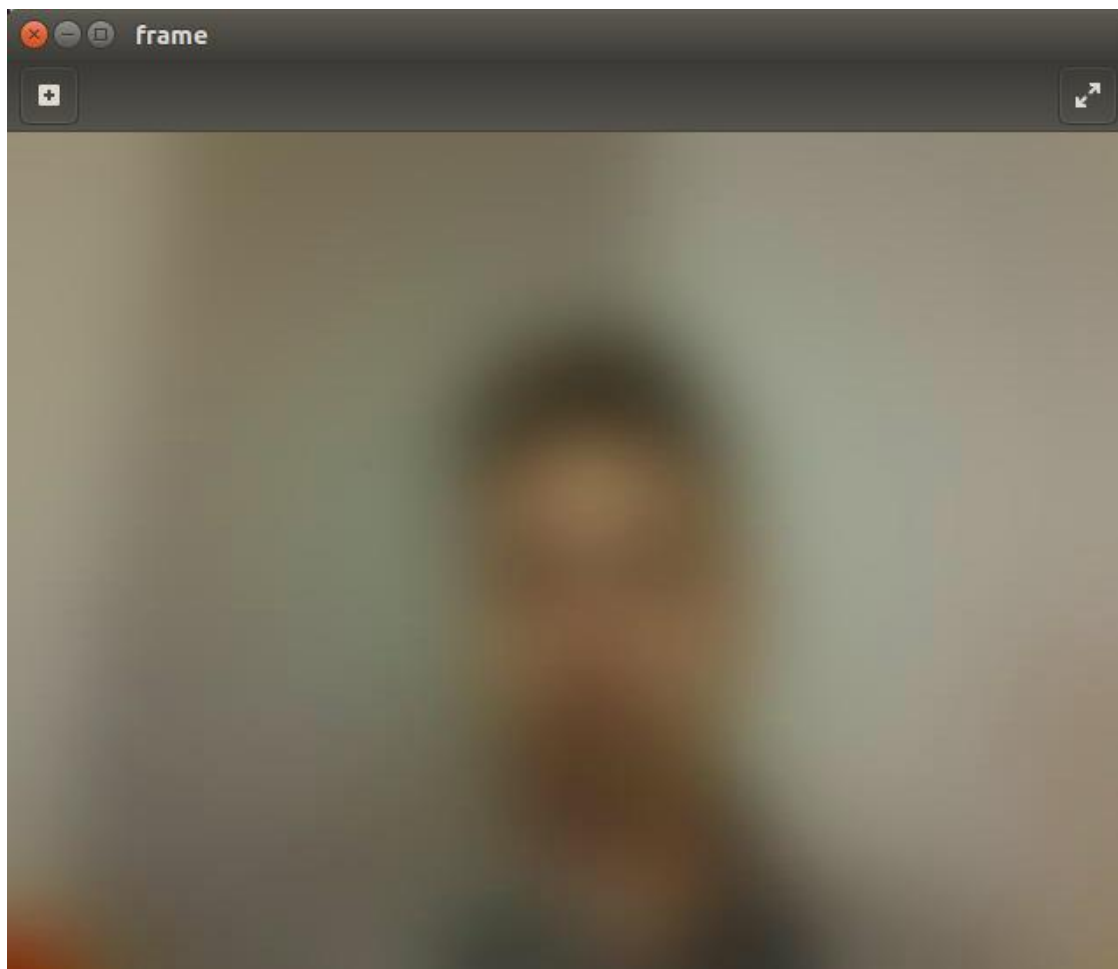
Dodawanie osoby



The screenshot shows a window titled "Obecność" with a light gray background. In the center, there is a white rectangular area containing a form. The form has two input fields: "Imie" (First Name) and "Nazwisko" (Last Name). Below these fields are two buttons: "Dodawanie Osoby!" (Add Person!) and "Sprawdź obecność!" (Check Presence!). At the bottom of the form area, there is a label "Lista Obecności :" (Attendance List :). Below this label is a large, empty white rectangular box, presumably for displaying the attendance list.

Rys. 8 Interfejs aplikacji zaraz po uruchomieniu aplikacji

W celu dodania osoby do zapisania wypełniamy imię i nazwisko w formularzu następnie klikamy przycisk Dodawanie Osoby! Otworzy się okno z bieżącym obrazem z kamery:



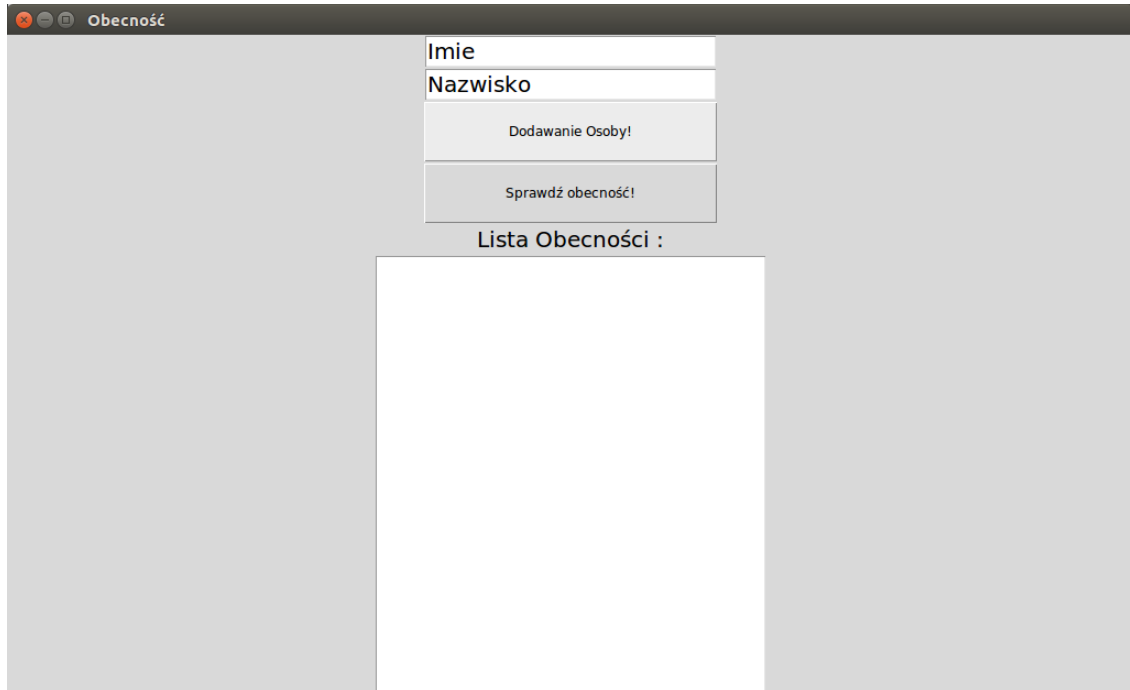
Rys. 9 Okno z bieżącym obrazem z kamery do wykonania zdjęcia

Należy wykadrować zdjęcie tak, żeby twarz była dobrze widoczna, nie była zasłonięta oraz dobrze oświetlona. Następnie należy przycisnąć klawisz *enter*, aby zrobić zdjęcie.

Zdjęcia można także dodawać w folderze „zdjęcia” projektu, zdjęcia muszą posiadać rozszerzenie *.jpg, a nazwa powinna składać się z imienia i nazwiska osoby dodawanej np. JanKowalski.jpg

Sprawdzanie obecności:

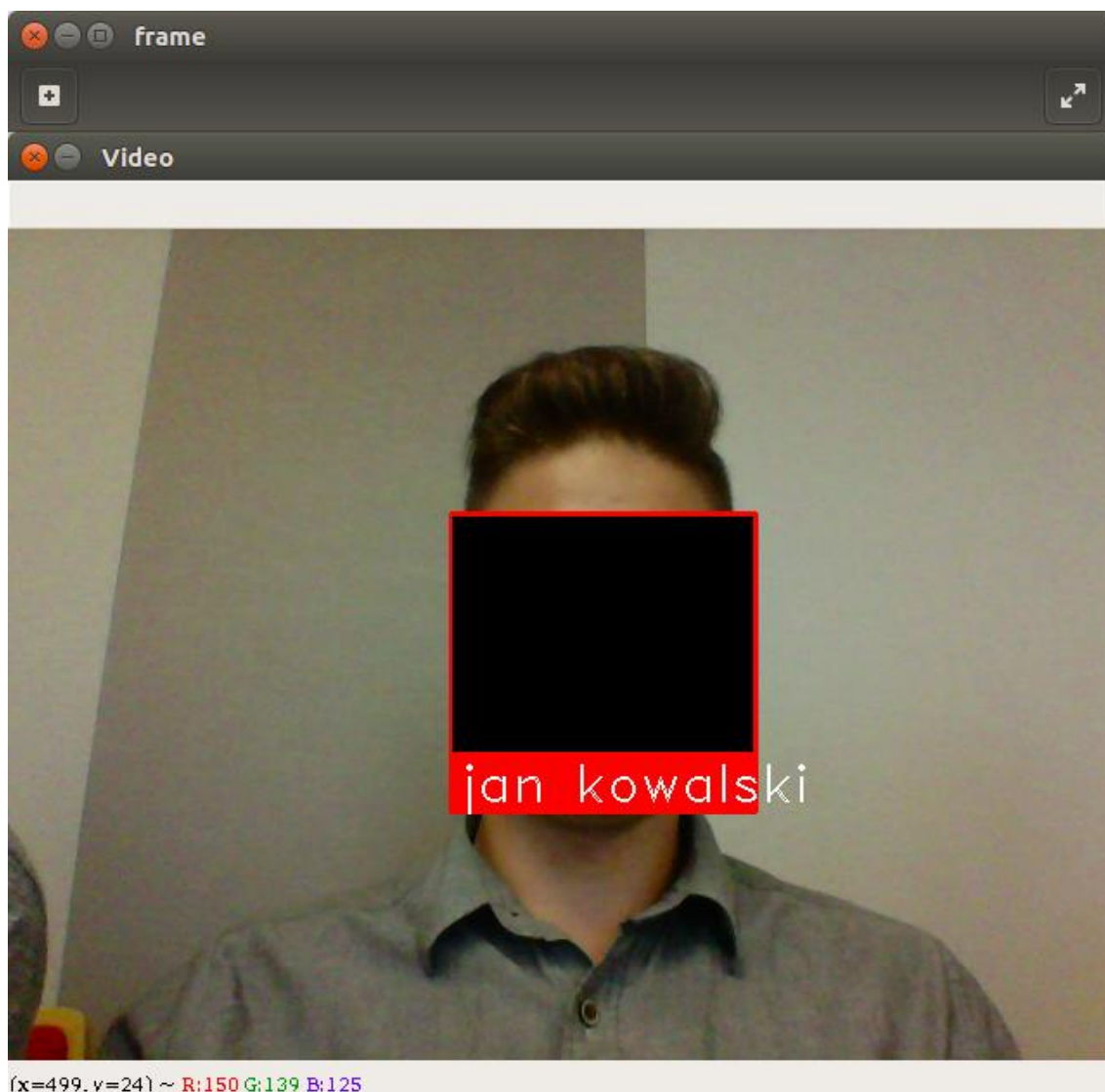
Aby sprawdzić obecność należy przycisnąć przycisk *Sprawdź obecność!* na ekranie startowym.



The screenshot shows a window titled "Obecność" with a light gray background. In the center, there is a white rectangular area containing the following elements from top to bottom: two input fields labeled "Imie" and "Nazwisko", a button labeled "Dodawanie Osoby!", and another button labeled "Sprawdź obecność!". Below these elements, the text "Lista Obecności :" is displayed, followed by a large, empty white rectangular box intended for a list.

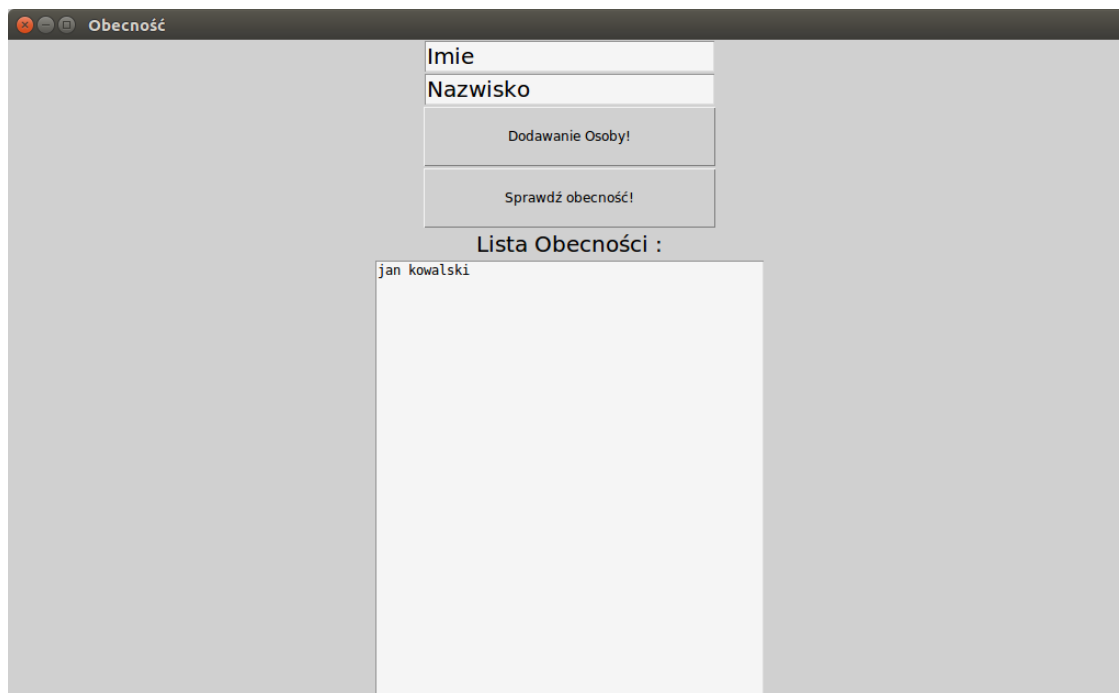
Rys. 10 Interfejs aplikacji zaraz po uruchomieniu aplikacji

Później pojawi nam się okno z obrazem z kamery, aby sprawdzić obecność studenci muszą być w polu widzenia kamery.



Rys. 11 Sprawdzanie obecności za pomocą rozpoznania twarzy

Kiedy wszyscy studenci pojawili się już przed kamerą należy zakończyć sprawdzanie obecności przyciskając klawisz *enter*.



Rys. 12 Ekran startowy z listą obecnych osób

Osoby obecne i wykryte przez system pojawią się w interfejsie aplikacji w polu Lista Obecności.

Testy

Program został poddany testom w różnych warunkach oświetlenia. Użyta została kamera zintegrowana z laptopem o rozdzielczości 1,3Mpix.

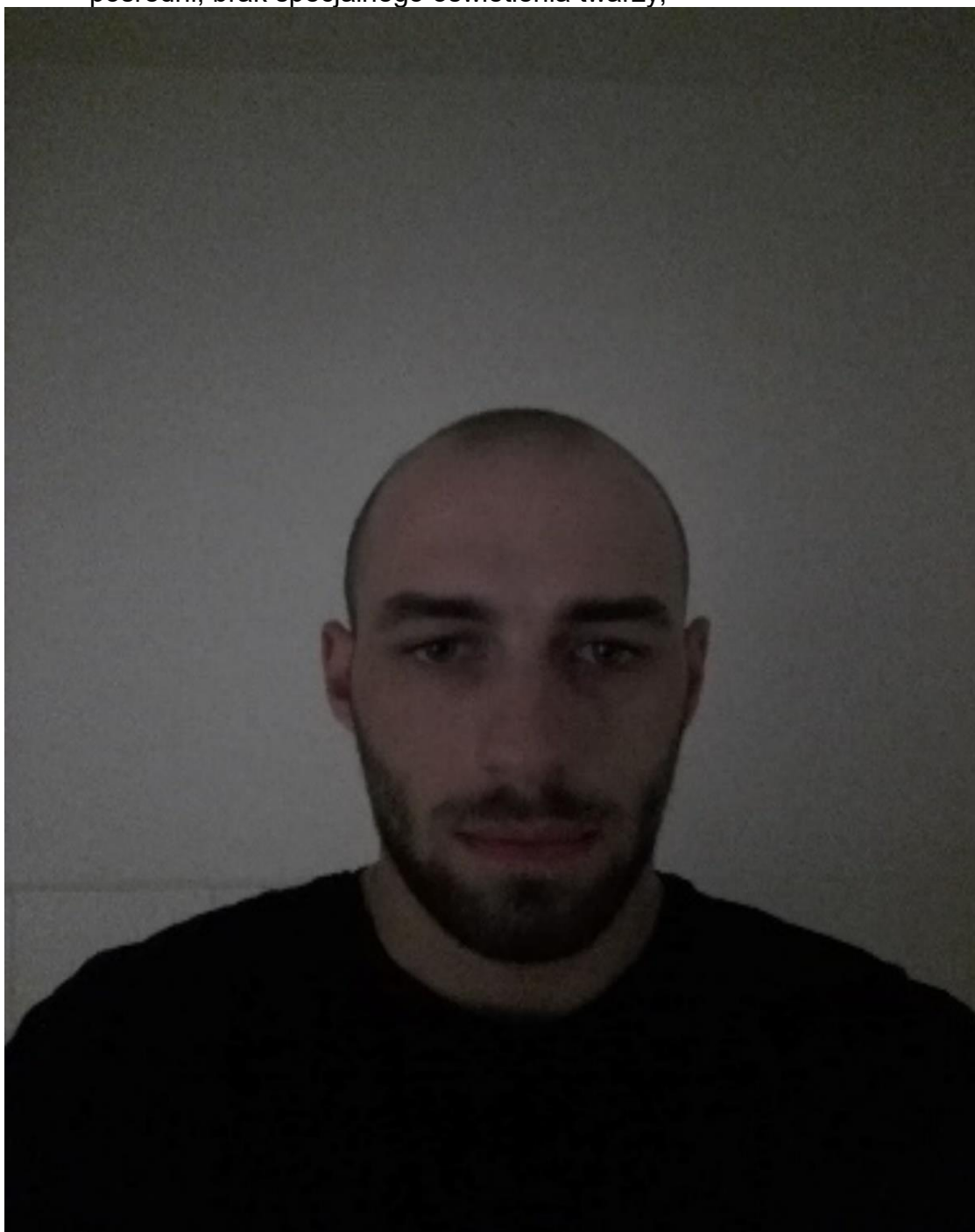
Wyróżniliśmy trzy poziomy oświetlenia obrazu:

- całkowity, oświetlona twarz od przodu,



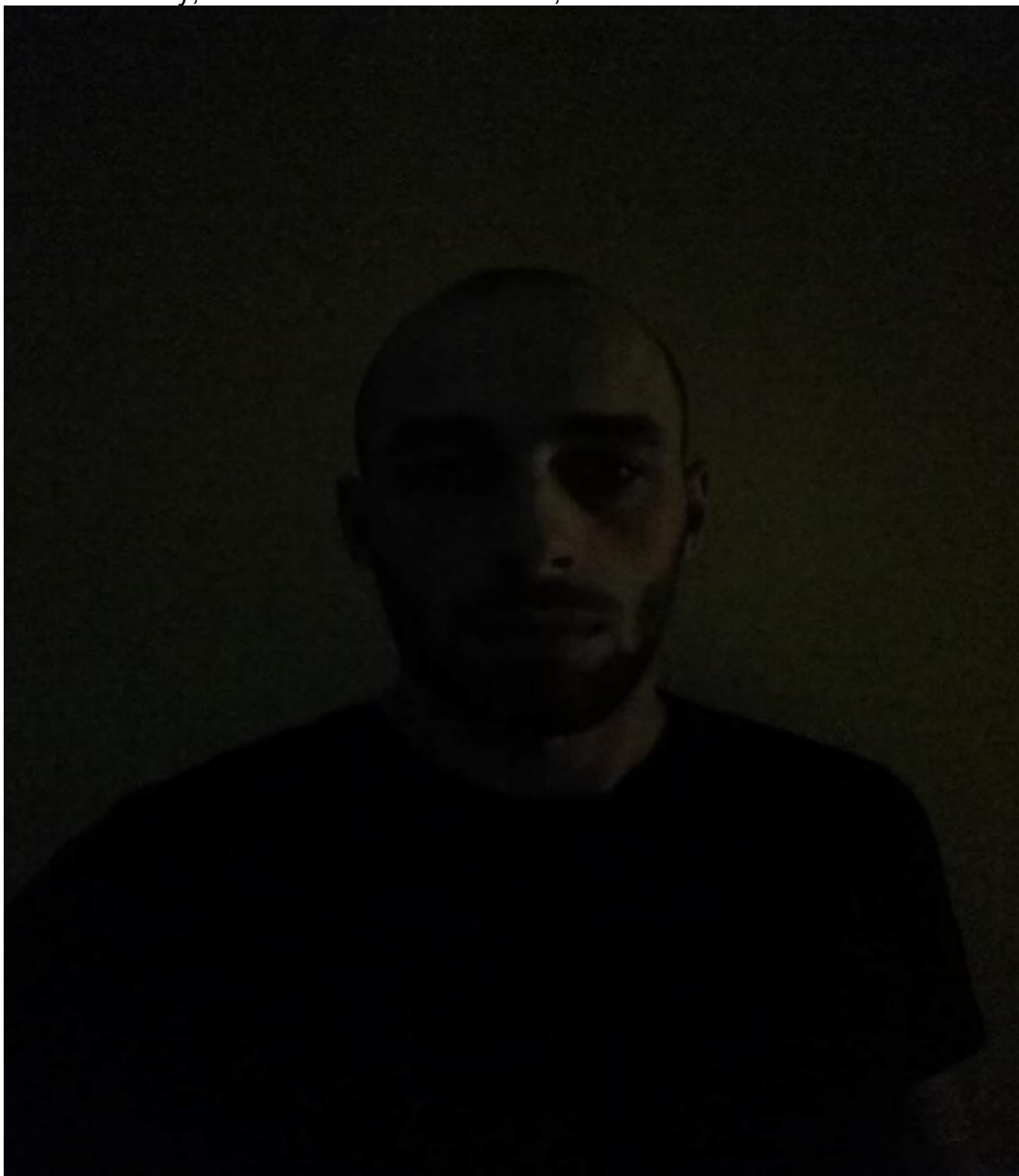
Rys. 13 Zdjęcie dobrze oświetlone

- pośredni, brak specjalnego oświetlenia twarzy,



Rys. 14 Zdjęcie średnio oświetlone

- znikomy, światło ustawione w oddali, twarz minimalnie oświetlona.



Rys. 15 Zdjęcie słabo oświetlone

Seria prób zawierała przechwycenie twarzy osoby 15 razy z różnego profilu, przez czas 2 sekundy.

W rezultacie otrzymaliśmy następujące wyniki:

1 – wykrycie oraz 0 – brak wykrycia (wykrycie twarzy, ale nie wykrycie osoby)

Tabela 1 Wyniki testów aplikacji

Nr próby	całkowity	pośredni	znikomy
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	0
6	1	1	0
7	1	0	0
8	1	0	1
9	1	1	1
10	1	1	0
11	1	1	0
12	1	0	0
13	1	1	1
14	1	1	1
15	1	1	0
Śr.	1.0	0.8	0.53

Z przeprowadzonych testów można wywnioskować, że algorytm rozpoznaje twarz danej osoby z minimalną skutecznością 53%, natomiast w dobrym oświetleniu ma 100% skuteczność.

Podsumowanie

W ramach pracy na zajęcia zostały zrealizowane postawione cele. Stworzony został program, który realizuje następujące cele:

- rozpoznawanie i detekcja twarzy,
- tworzenie listy obecności,
- możliwość dodawania osób do lokalnej bazy danych
- duża wydajność i szybkość działania programu

Perspektywy rozwoju

- Możliwość połączenia programu z zewnętrzną bazą danych w celu weryfikacji osób. Przykładowo przy znalezieniu nowej osoby program przeszukuje najpierw bazę lokalną, a jeżeli nie znajdzie, to wysyła zapytanie do api obsługującego zapytania w bazie danych.
- Możliwość dodania filtrów, efektów poprawiających jakość obrazu, w celu uzyskania lepszych wyników rozpoznawania twarzy poprzez manualne dostosowanie obrazu przekazywanego do algorytmu.
- Możliwość modyfikacji cech twarzy uzyskanych za pomocą algorytmu wykrywania twarzy.
- Możliwość utworzenia aplikacji wyświetlającej rozłożenie lokalizacji w sali, tworzenia statystyk.
- Możliwość stworzenia systemu wykrywającego poszukiwane osoby.
- Możliwość stworzenia systemu informującego rodziców o obecności dziecka na lekcjach.