

# POLITECHNIKA WROCŁAWSKA

## WYDZIAŁ ELEKTRONIKI

---

KIERUNEK: Automatyka i Robotyka (AIR)  
SPECJALNOŚĆ: Technologie Informacyjne w Systemach Auto-  
matyki (ART)

## PROJEKT INŻYNIERSKI

Aplikacja do identyfikacji i rozpoznawania twarzy  
użytkownika w celu zabezpieczenia dostępu przy  
użyciu technologii Qt

Qt based secure access application using face  
detection and identification

AUTOR:  
Dominik Guderski

PROWADZĄCY PROJEKT:  
dr inż. Andrzej Rusiecki, W-4/K-9

OCENA PROJEKTU:



*Projekt dedykuję rodzicom.*



# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>3</b>
1.1	Cel i zakres programu . . . . .	3
1.2	Układ pracy . . . . .	3
1.3	Koncepcja rozwiązania problemu . . . . .	4
1.3.1	Przechowanie nazwy użytkownika i hasła . . . . .	4
1.3.2	Przechowywanie wzorcowych zdjęć twarzy . . . . .	5
1.3.3	Konto administratora . . . . .	5
1.3.4	Detekcja i rozpoznawanie twarzy . . . . .	5
<b>2</b>	<b>Podstawy teoretyczne</b>	<b>9</b>
2.1	Przetwarzanie obrazów . . . . .	9
2.2	Rozpoznawanie twarzy . . . . .	9
<b>3</b>	<b>Opis programu</b>	<b>11</b>
3.1	Użyte technologie . . . . .	11
3.1.1	OpenCV . . . . .	11
3.1.2	Boost . . . . .	11
3.1.3	Qt . . . . .	12
3.1.4	Gmock . . . . .	12
3.2	Użyte algorytmy . . . . .	12
3.2.1	Detekcja twarzy . . . . .	12
3.2.2	Rozpoznawanie twarzy . . . . .	12
3.2.3	Szyfrowanie hasła . . . . .	12
3.3	Obsługa programu . . . . .	13
3.4	Struktura programu . . . . .	17
<b>4</b>	<b>Testowanie projektu</b>	<b>19</b>
4.1	Testowanie funkcjonalności uwierzytelniania kodem tekstowym i zarządzania użytkownikami . . . . .	19
4.2	Testowanie algorytmu wykrywającego twarz . . . . .	19
4.3	Testowanie algorytmu rozpoznającego twarz . . . . .	20
<b>5</b>	<b>Podsumowanie</b>	<b>21</b>
	<b>Bibliografia</b>	<b>21</b>



# Rozdział 1

## Wstęp

Rozpoznawanie twarzy jako technika biometryczna służąca do identyfikacji osób wykształciła się jako obiekt zainteresowań naukowców w latach osiemdziesiątych XX wieku. Pierwsze komercyjne systemy powstały w latach dziewięćdziesiątych XX wieku.[10]

Główną cechą rozpoznawania twarzy jest bezinwazyjność. Wystarczy spojrzenie w obiektyw. Systemy wykrywające i rozpoznające twarz wymagają szczególnych warunków otoczenia do poprawnego działania. Zmiany oświetlenia, położenia twarzy względem urządzenia rejestrującego mogą w znaczny sposób zaburzyć działanie algorytmu detekcji i rozpoznawania twarzy. W skrajnych przypadkach zmiana oświetlenia lub zmiana położenia twarzy względem urządzenia rejestrującego może uniemożliwić działanie algorytmów.

### 1.1 Cel i zakres programu

Niniejsza praca dotyczy praktycznego zagadnienia zabezpieczania dostępu do zasobów. Autor zaimplementował dwuelementowe uwierzytelnianie. Sprawdzana jest znajomość zdefiniowanego wcześniej hasła dla danego użytkownika oraz porównywana jest twarz odczytana z kamery z informacjami zawartymi w bazie danych. Moduł uwierzytelniania i obsługi informacji o użytkownikach został zaimplementowany przez autora od podstaw. Za bazę do stworzenia algorytmu identyfikacji oraz rozpoznawania twarzy posłużyła biblioteka openCV. Głównym celem było stworzenie programu, który realizowałby zadanie kontrolowania dostępu do zasobów poprzez rozpoznawanie twarzy w czasie rzeczywistym porównując z twarzami wzorcowymi oraz sprawdzanie znajomości hasła.

### 1.2 Układ pracy

We wstępie został opisany temat pracy, poruszone w niej zagadnienia oraz problemy związane z tematyką. W rozdziale drugim przedstawiono opis zagadnień teoretycznych związanych z tematem pracy. Opisano metody oraz biblioteki użyte w procesie realizacji założeń projektu. W podsumowaniu znajdują się wnioski czego udało się dokonać, czego udało się uniknąć i co sprawiło problemy. Podana też została propozycja jednej ze ścieżek dalszego rozwoju aplikacji.

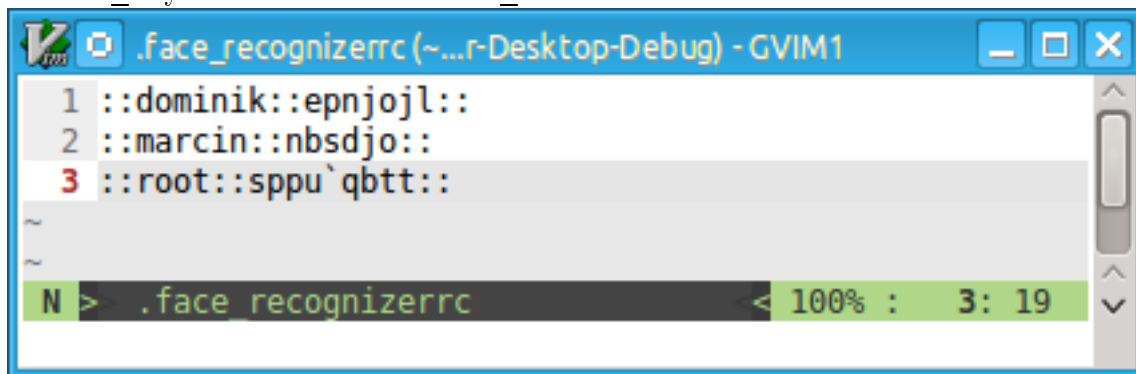
## 1.3 Koncepcja rozwiązania problemu

By zrealizować idee uwierzytelniania użytkownika na podstawie hasła oraz zdjęcia twarzy potrzebne są dwa elementy. Baza danych zawierająca nazwę użytkownika i hasło, oraz baza danych zawierająca wzorcowe zdjęcia twarzy powiązane z użytkownikiem. Oczywiście kwestią implementacyjną jest czy owe bazy będą jednym obiektem, czy też dwoma. W wypadku opisywanego projektu wybrane zostało podejście dwóch baz. Oczywiście obie te bazy są ze sobą synchronizowane i razem tworzą koncepcyjnie jedną bazę danych, nazywaną dalej bazą danych użytkowników.

### 1.3.1 Przechowanie nazwy użytkownika i hasła

Uwierzytelnianie użytkowników wymaga bazy danych. Owa baza została zrealizowana przy pomocy dwóch elementów. Pierwszy z nich to plik `.face_recognizerrc`, który przechowuje nazwę użytkownika oraz zakodowane hasło, w zdefiniowanym przez program formacie (zdefiniowanym w taki sposób by ułatwić przetwarzanie pliku przez program). Format ten ma postać:

```
::nazwa_użytkownika::zakodowane_hasło::  
::nazwa_użytkownika2::zakodowane_hasło2::
```



Czyli dla każdego użytkownika przeznaczona jest jedna linia w pliku. Nazwa użytkownika jest umieszczona w sposób niezakodowany, natomiast hasło umieszczone jest w sposób zakodowany prostym wariantem szyfru Cezara. Celem autora nie była implementacja nietrywialnego algorytmu szyfrowania hasła, ponieważ przyjęte zostało założenie, iż pliki konfiguracyjne programu są umieszczone w bezpiecznej lokalizacji. Można to osiągnąć uruchamiając program z konta administratora (wtedy pliki konfiguracyjne mogą zostać umieszczone w katalogu chronionym przed odczytem przez innych użytkowników niż administrator).

Innym rozwiązaniem jest uruchomienie programu w dowolnym miejscu, jednak ograniczenie interfejsu poprzez który użytkownicy programu mogą sterować programem. Takim ograniczeniem może być odpowiednie przygotowanie systemu operacyjnego, tak by uruchomiony program zajmował całą dostępną przestrzeń ekranu. Dodatkowo użytkownik ma do dyspozycji ekran dotykowy by móc sterować działaniem programu. Nie ma możliwości przełączenia czy też włączenia innego procesu w systemie. Takie podejście może zostać zastosowane przy projektowaniu interfejsu do sterowania inteligentnym budynkiem lub pomieszczeniem.



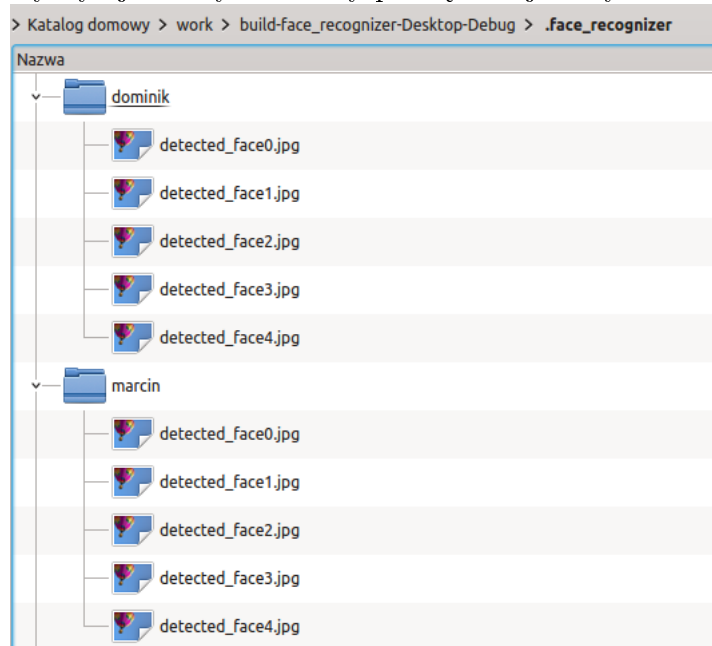
### 1.3.2 Przechowywanie wzorcowych zdjęć twarzy

Program oprócz wspomnianego pliku przechowującego dane tekstowe tworzy także katalog `.face_recognizer`. W tym katalogu dla każdego użytkownika z wyjątkiem użytkownika `root(administrator)` istnieje katalog o nazwie odpowiadającej nazwie użytkownika.

`.face_recognizer/`

- nazwa\_użytkownika/
- nazwa\_użytkownika2/

Katalog ten w momencie dodania użytkownika do bazy danych jest wypełniany dostarczonymi przez użytkownika pięcioma wzorcowymi zdjęciami twarzy. Wzorce te w momencie próby uzyskania dostępu do systemu przez użytkownika są porównywane z obrazem wykrytej twarzy z kamery podłączonej do systemu.



### 1.3.3 Konto administratora

Wyjątkowym przypadkiem, o którym należy wspomnieć jest konto administratora(login: root, domyślne hasło: pass). Owe konto jako jedyne jest dostępne już przy pierwszym uruchomieniu programu z domyślnym hasłem dostarczonym razem z programem. Bardzo ważna ze względów bezpieczeństwa jest zmiana domyślnego hasła. Wymaganiem stawianym administratorowi podczas procedury uwierzytelniania jest podanie prawidłowego hasła. Nie jest sprawdzany obraz z kamery. Oprócz tego konto root nie posiada katalogu root w katalogu `.face_recognizer/`, gdyż nie jest on potrzebny. Konto administratora jest potrzebne by dodawać i usuwać użytkowników.

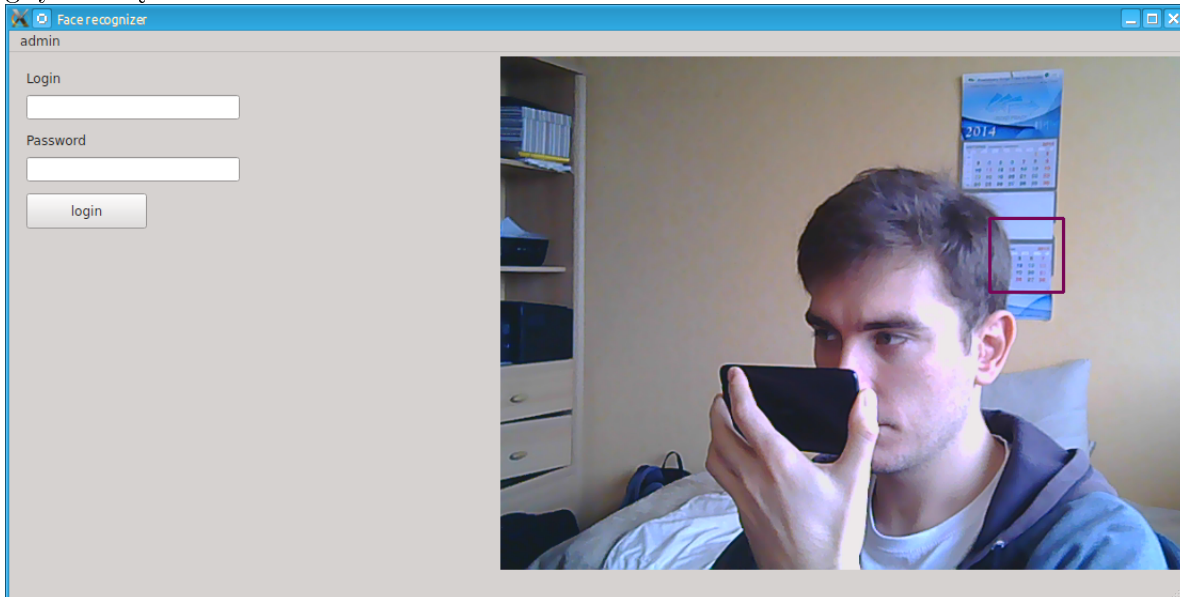
### 1.3.4 Detekcja i rozpoznawanie twarzy

W celu detekcji i rozpoznawania twarzy została użyta biblioteka OpenCV. Detekcja twarzy odbywa się przy użyciu funkcji Haar-like, umożliwiającej dość dobrą i szybką detekcję obiektów. Rozpoznawanie jest zrealizowane za pomocą metody EigenFace.

Podstawowy problem- szybkość przetwarzania obrazu, który zależy od mocy obliczeniowej komputera oraz zaimplementowanego algorytmu. Ważną kwestią jest poprawność

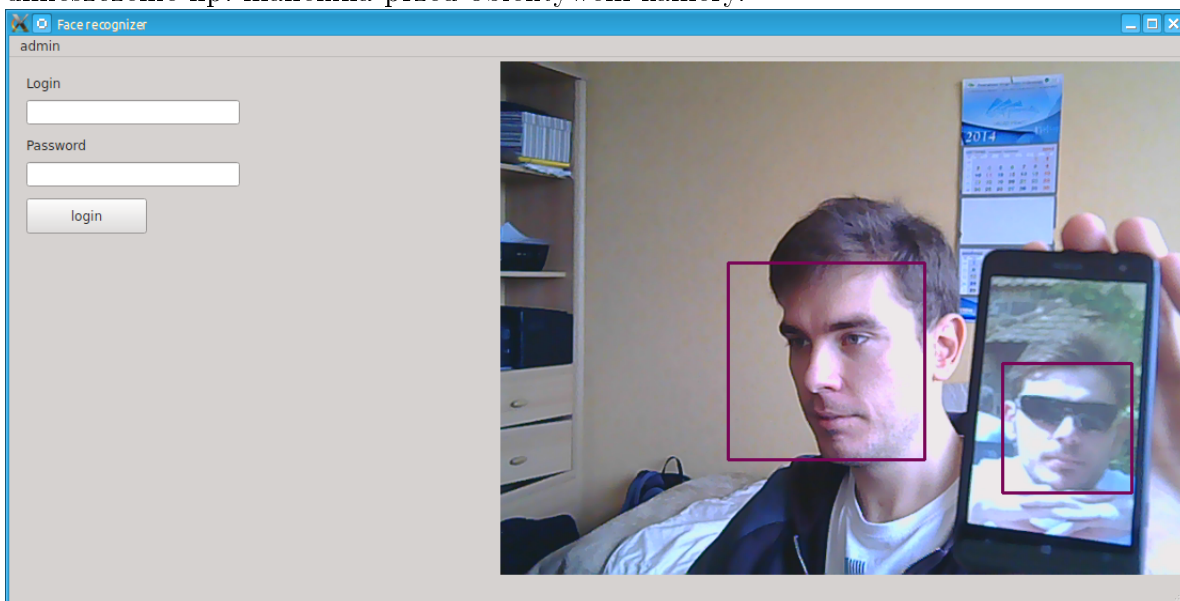
wykrywania twarzy. W wypadku gdy algorytm wykryje twarz w miejscu gdzie rzeczywiście jej nie ma może to prowadzić do błędnego działania programu, a także oznacza marnowanie zasobów. Z tego powodu w programie został przyjęty minimalny rozmiar wykrytej twarzy (50x50 pixeli). Wykryte obiekty mniejsze od tej granicznej wielkości nie są przetwarzane.

Takie podejście nie rozwiązuje jednak wszystkich problemów. Nadal może dojść do sytuacji, w której algorytm wykryje obszar obrazu twierdząc, że jest to twarz podczas gdy nie będzie to twarz.

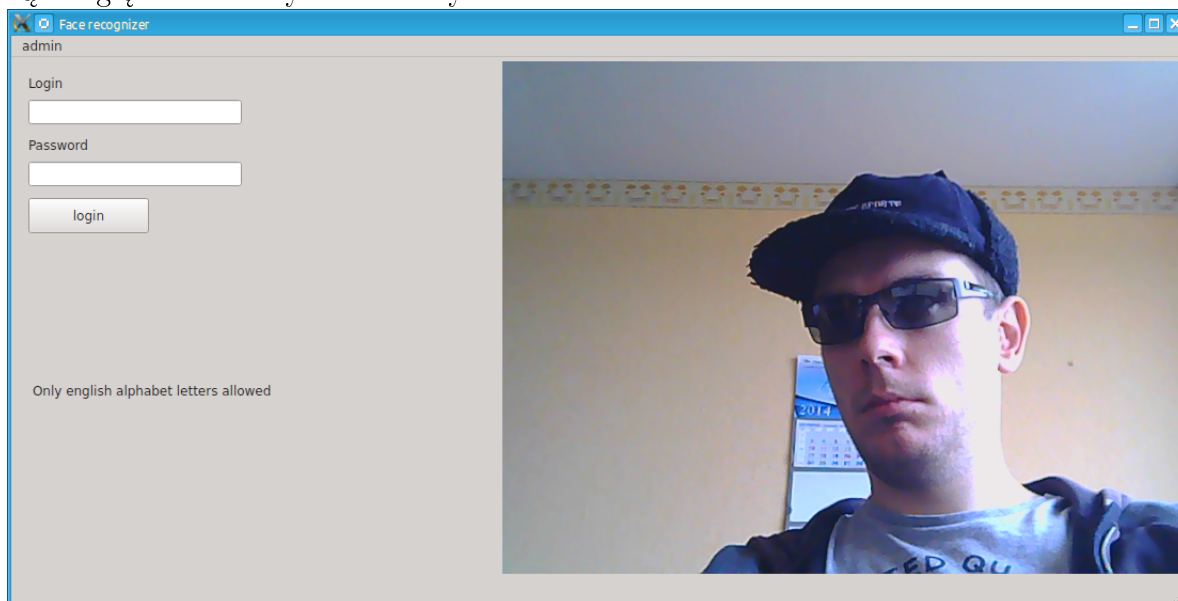


Program, który wykorzystuje funkcję Haar-like bazujący na rozpoznawaniu założonych wzorców jest szybki i dobrze pracuje w czasie rzeczywistym, jednak losowe zakłucenia w obrazie wejściowym (np. zła jakość obrazu, oświetlenie, niejednolite tło) mogą prowadzić do wskazania fragmentu obrazu nie zawierającego twarzy.

Podobny błąd może powodować umieszczenie przed obiektywem kamery ekranu lcd, na którym wyświetlane jest zdjęcie twarzy. Owym ekranem może być telefon komórkowy, tablet. Co oznacza, że podobny efekt działania algorytmu może być osiągnięty poprzez umieszczenie np. manekina przed obiektywem kamery.



Dużym wyzwaniem dla algorytmu wykrywającego jak i rozpoznającego twarz są także obroty twarzy względem kamery. Kolejnymi elementami są wszelkie ozdoby, czy też części garderoby. Okulary, czapka, zarost, tatuaże, kolczyki mogą w znaczący sposób zaburzyć owal głowy. Dla algorytmów wykrywających na podstawie koloru skóry obrót nie jest dużym problemem, jednak dla algorytmów opierających się na cechach twarzy różny jej kąt względem kamery stanowi wyzwanie.



Detekcja jak i rozpoznawanie twarzy nie pozostają obojętne na warunki świetlne, w których wykonywane jest pobieranie obrazu przez kamerę. Oprócz samego natężenia, równomierności oświetlenia jest jeszcze kwestia jakości urządzenia pobierającego obraz. Istnieją kamery, które posiadają moduły wyrównywanie natężenia oświetlenia na pobranym obrazie realizowane zarówno sprzętowo jak i poprzez wbudowane oprogramowanie. Nie bez znaczenia pozostaje rozdzielczość kamery. Niska sprawia, że obraz jest zniekształcony co utrudnia w sposób znaczny działanie algorytmów detekcji czy rozpoznawania. Kolejnymi problemami technicznymi związanymi z budżetowymi urządzeniami pobierającymi obraz są problemy z auto fokusem. Jednym z rozwiązań jest stosowanie wysokiej jakości kamery. Choć i tutaj w skrajnych przypadkach(bardzo małe natężenie światła) otrzymany obraz nie będzie nadawał się do opisywanych algorytmów. Skrajnym rozwiązaniem jest stosowanie kamery termowizyjnej. W wypadku takich urządzeń użytkownik zostaje uniezależniony od natężenia światła, dzięki czemu problemy opisywane wyżej nie występują. Wadą tego rozwiązania jest koszt kamery termowizyjnej. W chwili przygotowywania projektu koszt kamery termowizyjnej został oszacowany na 2 rzędy wielkości wyższy aniżeli koszt standardowej kamery. Przyjmuje się, że do działania opisywanego systemu dane wejściowe dostarczane przez urządzenie pobierające obraz są dobrej jakości.



# Rozdział 2

## Podstawy teoretyczne

### 2.1 Przetwarzanie obrazów

### 2.2 Rozpoznawanie twarzy



# Rozdział 3

## Opis programu

### 3.1 Użyte technologie

Program został stworzony w systemie operacyjnym Linux 3.13.0-39-generic #66-Ubuntu SMP, Ubuntu 14.04 w języku C++(ISO/IEC C++ 2003) przy użyciu kompilatora g++ (Ubuntu 4.8.2-19ubuntu1) 4.8.2. Środowiskiem służącym do projektowania graficznego interfejsu użytkownika był program Qt Creator 3.0.1 oparty na bibliotece Qt 5.2.1. Oprócz wspomnianych elementów zastosowane zostały biblioteki OpenCV 2.4.8, boost 1.54.0.1 przy tworzeniu kodu produkcyjnego- przeznaczonego do wytworzenie oprogramowania. Oprócz tego pomocniczo przy tworzeniu testów jednostkowych został użyty pakiet gmock 1.7.0.

#### 3.1.1 OpenCV

OpenCV to otwarta biblioteka stworzona przez Intel. W momencie tworzenia projektu udostępnione był interfejs biblioteki w językach C++, C, Python, Java. Biblioteka może być używana w systemie Windows, Linux, Mac OS, iOS oraz Android. Pracę nad biblioteką zostały rozpoczęte w 1999 roku. Wersja dojrzała została udostępniona w 2006 roku. Biblioteka jest wydawana na licencji BSD(Berkeley Software Distribution License). Ta bardzo liberalna licencja pozwala nie tylko na modyfikacje kodu biblioteka, ale także rozprowadzanie go w takiej postaci. Pozwala także na rozprowadzanie produktu bez postaci źródłowej czy wręcz włączenie do zamkniętego oprogramowania, pod warunkiem załączenia do produktu informacji o autorach oryginalnego kodu i treści licencji.

#### 3.1.2 Boost

Kolekcja bibliotek programistycznych poszerzających możliwości języka C++, objętych liberalną licencją(Boost Software License), która umożliwia użycie ich w dowolnym projekcie.

Dzięki restrykcyjnemu systemowi recenzowania i kontroli jakości, biblioteki Boost są poważane ze względu na ich wysoką jakość oraz często stawiane za wzorcowy przykład nowoczesnego projektowania i programowania w C++. Dziedziny zastosowania Boost są bardzo szerokie, pakiet dostarcza m.in. biblioteki ogólnego przeznaczenia (inteligentne wskaźniki, wyrażenia regularne), biblioteki stanowiące warstwę abstrakcji dla systemu operacyjnego (obsługa systemów plików czy wielowątkowości), jak i narzędzia przeznaczone głównie dla innych twórców bibliotek i zaawansowanych programistów języka C++ (np. biblioteka metaprogramowania MPL). Kilka bibliotek wchodzących w poczet Boost

zostało włączonych do pierwszego raportu technicznego komitetu standaryzacyjnego C++ (w jego skład wchodzi wielu spośród twórców Boost).[11]

### 3.1.3 Qt

Zestaw przenośnych bibliotek i narzędzi programistycznych dedykowanych dla języków C++, QML i Java. Ich podstawowym składnikiem są klasy służące do budowy graficznego interfejsu programów komputerowych, począwszy od wersji 4.0 Qt zawiera też narzędzia do tworzenia programów konsolowych i serwerów.

Środowisko Qt jest dostępne dla platform: X11 (m.in. GNU/Linux, BSD, Solaris), Windows, Mac OS X, Haiku oraz dla urządzeń wbudowanych opartych na Linuksie (Qt Extended), Windows CE, Symbian, Android. Qt jest podstawą dla m.in. uniksowego środowiska graficznego KDE oraz uniksowych wersji komunikatora internetowego Skype i programu Google Earth.

Biblioteki Qt dostępne są w języku C++ i Java; mogą też być wykorzystywane w programach napisanych w innych językach, m.in. Ada (QtAda), C# (Qyoto/Kimono), Pascal, Perl (Perl Qt4), PHP (PHP-Qt), Ruby (QtRuby) i Python (PyQt). Charakteryzują się w pełni obiektową architekturą. Licencje LGPL (v. 2.1), GPL (v. 3.0), komercyjna.[12]

### 3.1.4 Gmock

Biblioteka do testów jednostkowych oprogramowania dla języka C++, bazująca na architekturze xUnit. Jest wydawana na licencji BSD 3. Może być używana na wielu platformach: Linux, Windows, Mac OS X.

## 3.2 Użyte algorytmy

### 3.2.1 Detekcja twarzy

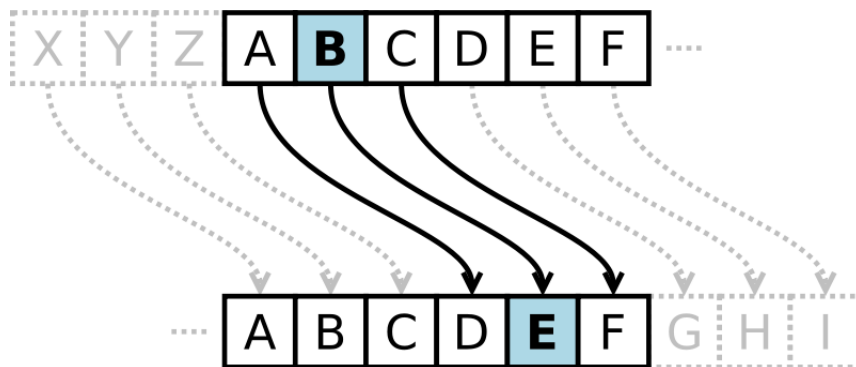
### 3.2.2 Rozpoznawanie twarzy

Nulla a nisl

### 3.2.3 Szyfrowanie hasła

Szyfr Cezara (zwany też szyfrem przesuwającym, kodem Cezara lub przesunięciem Cezariańskim) – w kryptografii jedna z najprostszych technik szyfrowania. Jest to rodzaj szyfru podstawieniowego, w którym każda litera tekstu jawnego (niezaszyfrowanego) zastępowana jest inną, oddaloną od niej o stałą liczbę pozycji w alfabecie, literą (szyfr monoalfabetyczny), przy czym kierunek zamiany musi być zachowany. Nie rozróżnia się przy tym liter dużych i małych. Nazwa szyfru pochodzi od Juliusza Cezara.[13]



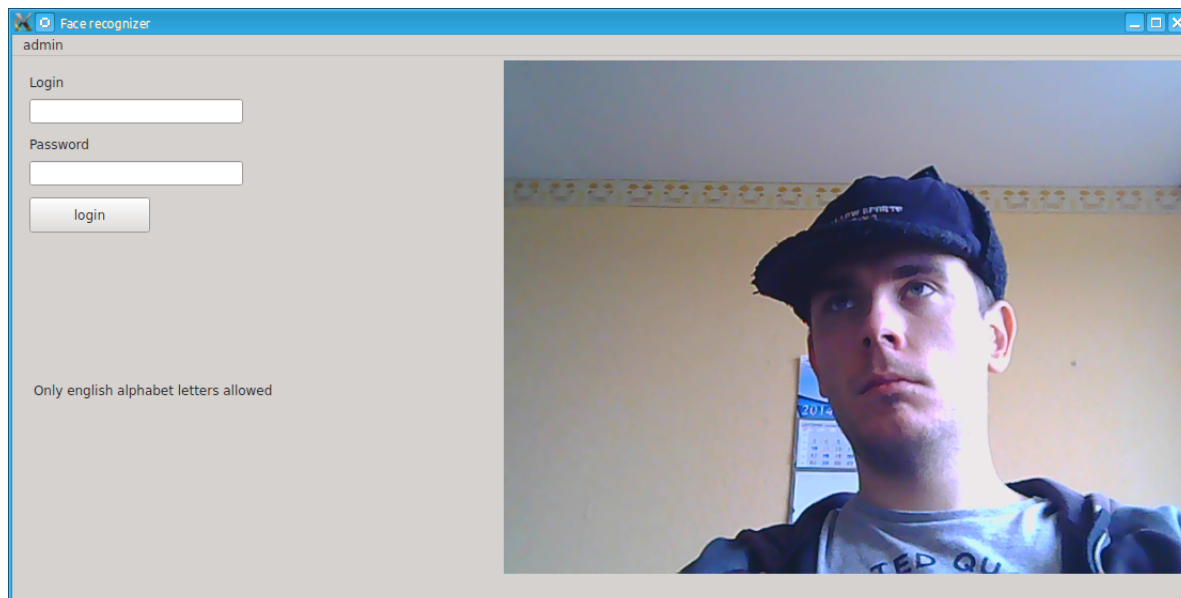


Użyta w projekcie metoda szyfrowania rozróżnia duże i małe litery i przypisuje im wagi zgodne z tablicą ASCII. Przesunięcie wynosi 1. Szyfrowanie zostało zastosowane w projekcie do celów innych niż najczęściej rozumiane zabezpieczenie informacji. Przyjęte jest założenie, że użytkownicy, którzy próbują uzyskać dostęp do zasobów poprzez opisywany system nie mogą uzyskać dostępu do pliku, w którym przechowywane są dane dotyczące nazw użytkowników i ich haseł. Takie ograniczenie może wynikać z ograniczeń w interfejsie dostępnym dla użytkownika systemu, lub wprost z konstrukcji systemu operacyjnego, w którym uruchamiana jest aplikacja (można założyć, że aplikacja uruchomiona jest w systemie linux z uprawnieniami użytkownika root i w katalogu domowym tego użytkownika, z czego wynika, że żaden użytkownik poza użytkownikiem root domyślnie nie będzie miał dostępu do czytania z bazy danych programu). Wprowadzenie szyfrowania zostało podyktowane możliwością przypadkowego otwarcia pliku przez administratora systemu. W takim wypadku administrator nie odczyta prawdziwego hasła. Dopiero celowa chęć odszyfrowania hasła może sprawić, że odczyta hasło innego użytkownika.

Algorytm szyfrowania zastosowany w kodzie Cezara bywa fragmentem bardziej złożonych systemów szyfrowania, takich jak szyfr Vigenère'a. Współcześnie szyfru Cezara używa się z przesunięciem 13 (ROT13), będącego prostym i szybkim sposobem na ukrycie treści.

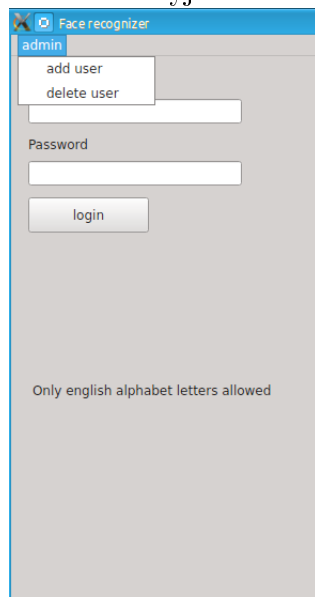
### 3.3 Obsługa programu

Program do działania w folderze, w którym jest uruchamiany potrzebuje pliku `haarcascade_frontalface_alt.xml` oraz kamery poprawnie podłączonej i rozpoznanej przez system operacyjny. Po spełnieniu powyższych warunków i uruchomieniu programu ukazuje się okno:

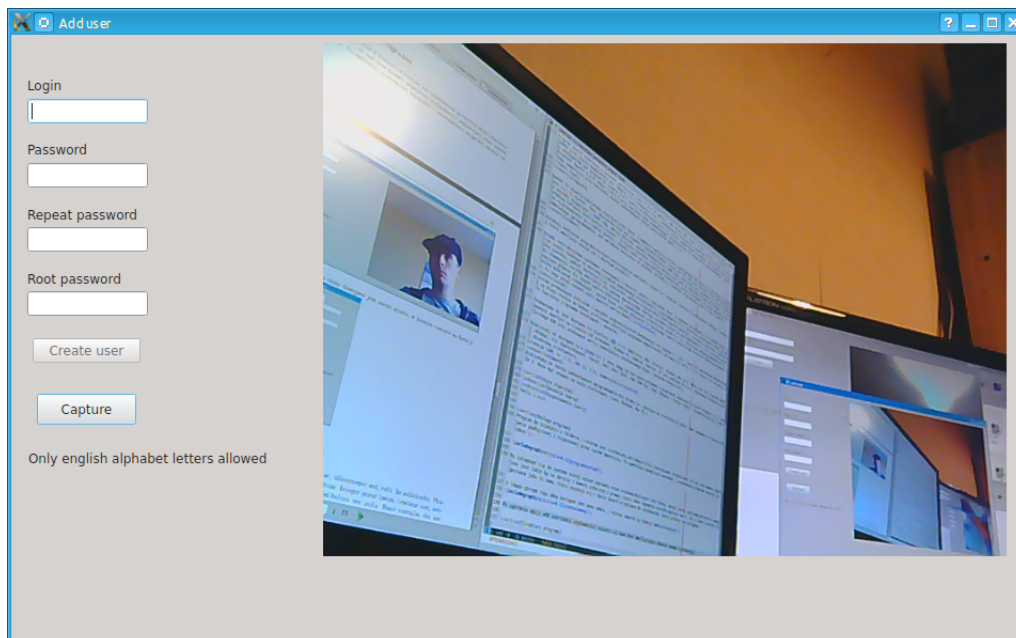


Każde pole przyjmujące informacje tekstowe od użytkownika (pola tekstowe takie jak login, password) przyjmują wyłącznie znaki należące do angielskiego alfabetu. Można wprowadzić inne znaki do samego pola formularza, jednak taki napis nie będzie dalej przetwarzany. By zalogować się do systemu należy wpisać poprawną nazwę użytkownika (login) oraz hasło. Oprócz konta root (administratora) wymagane jest także by na obrazie z kamery widocznym w prawej części okna logowania została wykryta twarz i by ta twarz została rozpoznana jako ta sama, która znajduje się w bazie danych przypisana do użytkownika, który próbuje się zalogować.

W lewym górnym rogu okna dostępne jest menu admin, w którym zawarte są funkcje administracyjne:

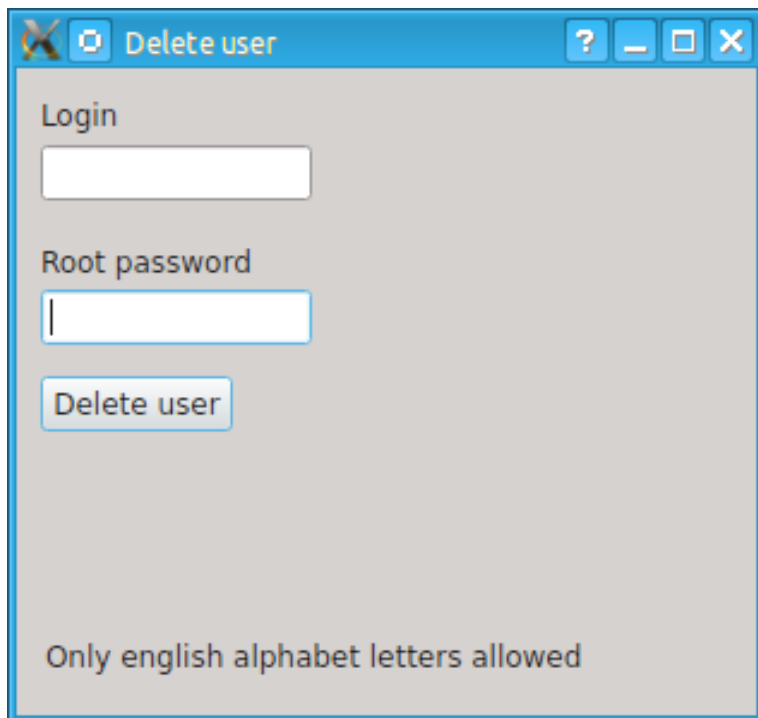


Po wybraniu opcji add user (dodaj użytkownika) otwiera się nowe okno umożliwiające dodanie nowego użytkownika:



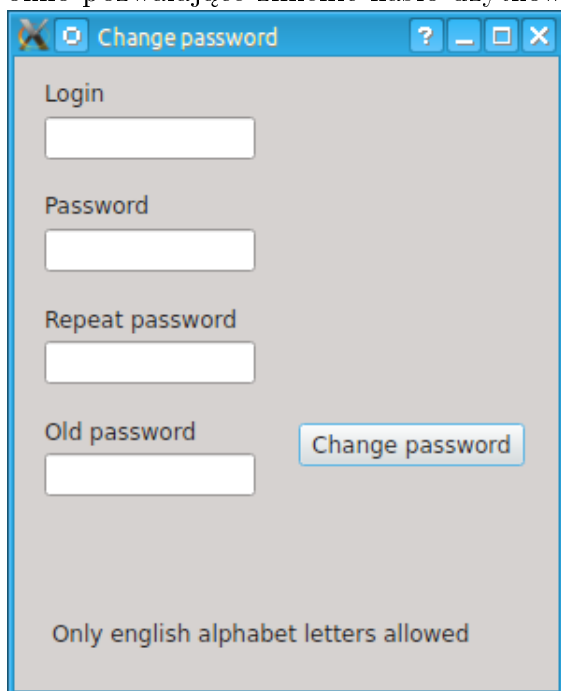
Aby dodać użytkownika należy wykonać 5 zdjęć, które będą używane przy logowaniu do systemu (sprawdzanie czy twarz widniejąca na obrazie kamery w momencie logowania jest tą samą, która została dodana do bazy danych dla danego użytkownika). Kiedy na obrazie po prawej stronie okna będzie widoczny prostokąt otaczający twarz użytkownika należy wcisnąć przycisk Capture. Czynność należy powtórzyć 5 razy. Gdy to zadanie zostanie wykonane pomyślnie przycisk Create user stanie się aktywny i będzie możliwe stworzenie nowego użytkownika po uzupełnieniu wszystkich pól formularza. Należy wpisać nazwę nowego użytkownika, hasło, powtórzyć hasło oraz hasło administratora (konto root). Po wykonaniu tych czynności i wcisnięciu przycisku Create user, jeśli nie pojawiły się inne błędy (błąd zapisu do pliku, ciąg znaków w polu password i repeated password nie są równe) nowy użytkownik zostanie dodany do bazy danych.

Analogicznie skonstruowane zostało okno, które otwiera się gdy w głównym oknie programu z menu kontekstowego zostanie wybrana opcja delete user. Tutaj dane tekstowe, które podawane są przez użytkownika to: login (nazwa użytkownika) i root password (hasło administratora). W tym oknie nie ma obrazu z kamery, ponieważ do uwierzytelniania konta administratora nie używa się obrazu z kamery. Jeśli wszystkie dane zostały wpisane poprawnie i użytkownik istnieje w bazie usunięcie powiedzie się.



The screenshot shows a window titled "Delete user" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are two text input fields: "Login" and "Root password". Below the "Root password" field is a button labeled "Delete user". At the bottom of the window, there is a text label that reads "Only english alphabet letters allowed".

Gdy w głównym oknie programu wpisujemy poprawne dane uwierzytelniające uzyskamy dostęp do chronionych zasobów. Taka akcja może spowodować uruchomienie dowolnego okna, otwarcie dowolnego dokumentu. Do celów demonstracyjnych zostało wprowadzone okno pozwalające zmienić hasło użytkownika:

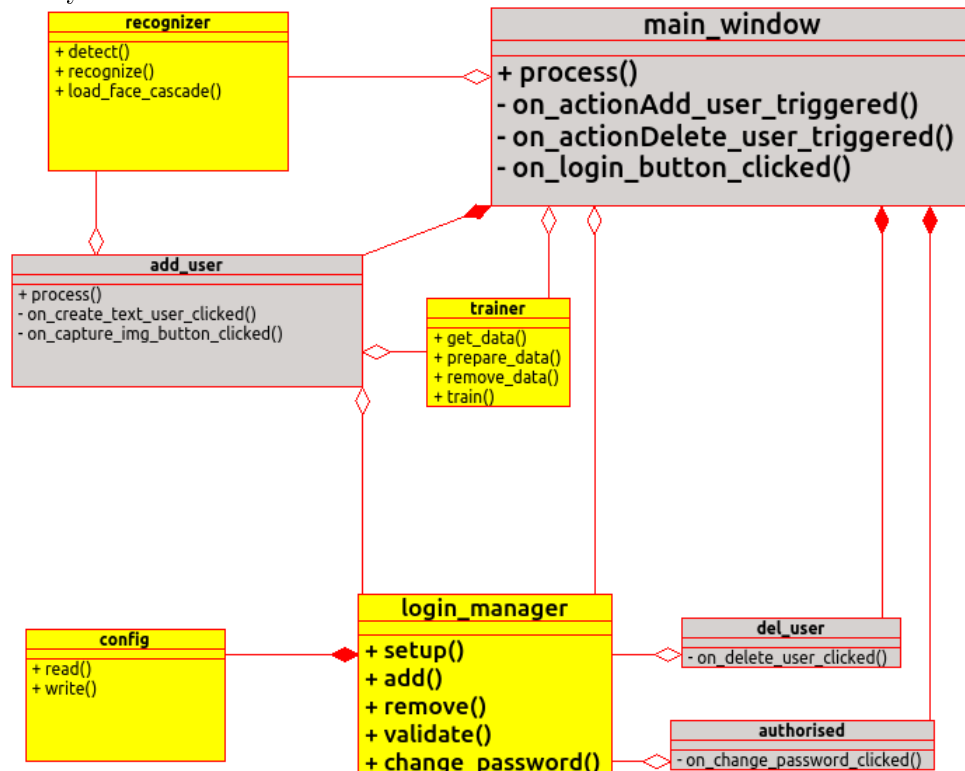


The screenshot shows a window titled "Change password" with a standard Windows-style title bar. Inside the window, there are four text input fields: "Login", "Password", "Repeat password", and "Old password". To the right of the "Old password" field is a button labeled "Change password". At the bottom of the window, there is a text label that reads "Only english alphabet letters allowed".

Po poprawnym wypełnieniu wszystkich pól następuje zmiana dotychczasowego hasła na nowe dla wybranego użytkownika.

## 3.4 Struktura programu

Program można podzielić koncepcyjnie na dwa elementy. Warstwę prezentacji i warstwę logiczną. Warstwę prezentacji stanowi kod odpowiedzialny za tworzenie i obsługę graficznego interfejsu użytkownika. Natomiast warstwa logiczna to klasy realizująca potrzebne algorytmy: komunikacje z bazą danych, kodowanie hasła czy też rozpoznawanie twarzy. Takie podejście jest zgodne z paradygmatem pisania oprogramowania dla systemów z rodziny Linux.



Jeżeli zajdzie potrzeba, można stworzyć graficzny interfejs użytkownika w zupełnie odmienniej technologii niż użyta w tym projekcie. Dzięki oddzieleniu dwóch warstw prezentacji i logicznej taka potrzeba może być zrealizowana w relatywnie krótkim czasie. Będzie wymagać mało prac reorganizujących kod.

Kolejną zaletą takiego rodzaju modularności kodu jest łatwość testowania. Począwszy od zautomatyzowanych testów jednostkowych, które w swoich scenariuszach testowych mogą tworzyć instancje poszczególnych klas wywoływać na nich metody ich interfejsów po czym sprawdzać wyniki. Istnieją nawet propozycje i możliwości zautomatyzowanego testowania wszystkich metod danej klasy, a nie tylko jej interfejsu jednak nie są one tak rozpowszechnione. Także testy manualne aplikacji, gdy uruchamiana z linii komend i odpowiednio dostosowana do scenariusza testowego mogą przebiegać sprawniej niż z graficznym interfejsem użytkownika.



# Rozdział 4

## Testowanie projektu

Testowanie było dwuetapowe. Na etapie tworzenie implementacji warstwy logicznej równoległe z tworzenie implementacji powstawały testy jednostkowe pozwalające wykryć ewentualne błędy. Oprócz tego po napisaniu warstwy logicznej dodany został kod odpowiadający za warstwę prezentacji. W dużej mierze ten kod był testowany manualnie przez użytkownika. Wyniki testów całości można podzielić na trzy grupy:

- Testowanie funkcjonalności uwierzytelniania kodem tekstowym i zarządzania użytkownikami
- Testowanie algorytmu wykrywającego twarz
- Testowanie algorytmu rozpoznającego twarz

### 4.1 Testowanie funkcjonalności uwierzytelniania kodem tekstowym i zarządzania użytkownikami

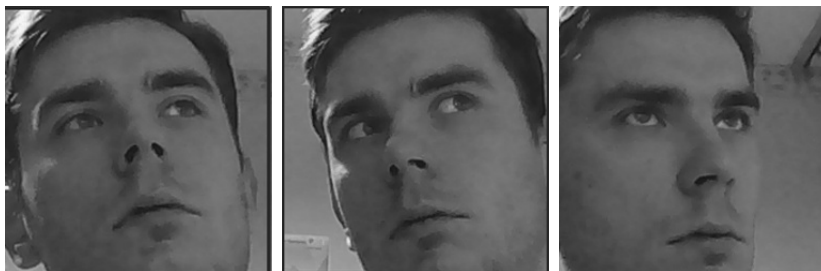
Program poprawnie komunikuje się z bazą danych: dodaje, usuwa, modyfikuje użytkowników. Ponadto umożliwia zalogowanie się. Jedną z tych operacji może się nie powieść w wypadku, gdy np. system operacyjny nie pozwoli utworzyć jednego z plików, który należy do bazy danych.

### 4.2 Testowanie algorytmu wykrywającego twarz

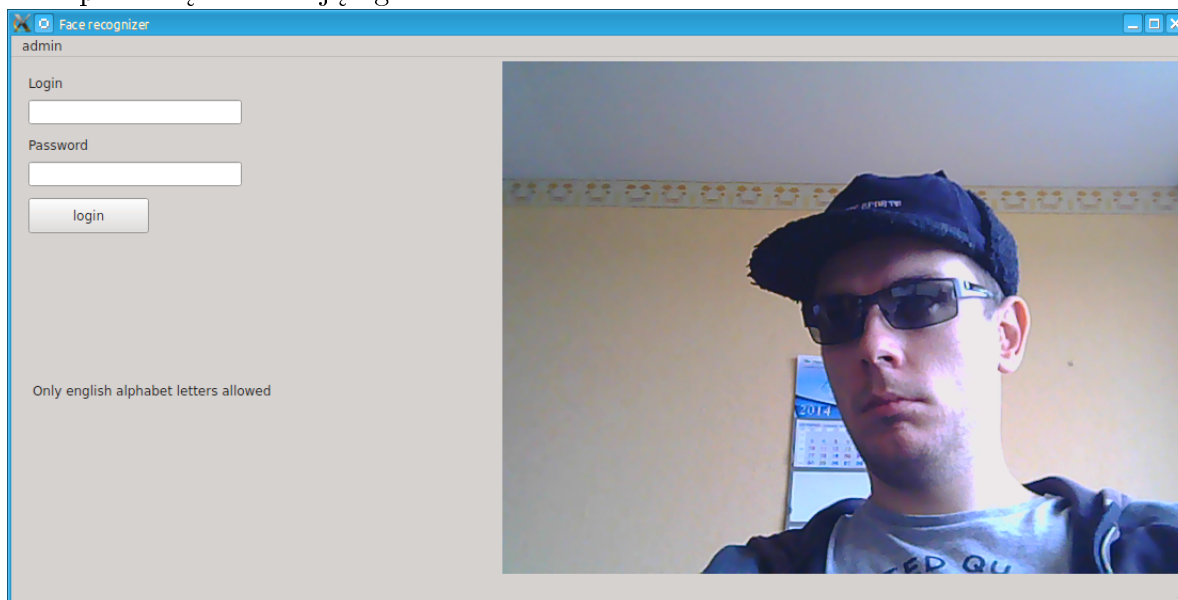
Algorytm działa z dużą skutecznością. Nawet w trudnych warunkach oświetlenia (nierównomiernie doświetlone zdjęcie, małe natężenie oświetlenia) wykrywa z powodzeniem twarz. W testach manualnych autora, podczas obracania twarzy pod różnym kątem i stosowania różnego rodzaju oświetlenia oraz elementów ubioru (okulary, czapka) wykrywalność twarzy

była na poziomie 80





Choć pojawiają się sytuacje, gdy przy trudnych warunkach twarz nie zostanie wykryta-  
brak prostokąta otaczającego twarz.



W praktyce nie da się precyzyjnie wyznaczyć granicznego kąta obrotu twarzy względem obiektywu kamery gdy twarz przestaje być wykrywana. Bowiem oprócz wspomnianego parametru pojawia się wiele innych zmiennych- oświetlenie, cechy charakterystyczne danej twarzy, mimika czy ubiór.

### 4.3 Testowanie algorytmu rozpoznającego twarz



# Rozdział 5

## Podsumowanie

Etiam scelerisque vulputate nulla. Phasellus facilisis vehicula lectus. Nullam adipiscing nisi sit amet sapien.



# Bibliografia

- [1] Daniel Lelis Baggio, Mastering OpenCV with Practical Computer Vision Projects
- [2] Adrian Kaehler, Gary Bradski, Learning OpenCV 2nd Edition
- [3] Ewaryst Rafajłowicz, Wojciech Rafajłowicz, Andrzej Rusiecki, Algorytmy przetwarzania obrazów i wstęp do pracy z biblioteką OpenCV
- [4] Jasmin Blanchette, Mark Summerfield, C++ GUI Programming with Qt 4, Second Edition
- [5] Robert C. Martin, Czysty kod. Podręcznik dobrego programisty
- [6] <http://www.boost.org/doc/>
- [7] <http://www.cplusplus.com/reference/>
- [8] <http://docs.opencv.org/>
- [9] <http://www.multimedia-computing.de/mediawiki//images/5/52/MRL-TR-May02-revised-Dec02.pdf>
- [10] History of Face Recognition <http://vismod.media.mit.edu/tech-reports/TR-516/node7.html>
- [11] <http://pl.wikipedia.org/wiki/Boost>
- [12] <http://pl.wikipedia.org/wiki/Qt>
- [13] [http://pl.wikipedia.org/wiki/Szyfr\\_Cezara](http://pl.wikipedia.org/wiki/Szyfr_Cezara)