

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

1	Wstęp	3
1.1	Cel i zakres programu	3
1.2	Układ pracy	3
1.3	Koncepcja rozwiązania problemu	4
1.3.1	Przechowywanie 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	6
2	Podstawy teoretyczne	9
2.1	Model barw	9
2.2	Identyfikacja twarzy	11
2.3	Szyfrowanie hasła	11
3	Opis programu	13
3.1	Użyte technologie	13
3.1.1	OpenCV	13
3.1.2	Boost	13
3.1.3	Qt	14
3.1.4	Gmock	14
3.2	Konfiguracja środowiska programistycznego Qt	14
3.3	Podręcznik użytkownika	14
3.4	Struktura programu	19
4	Testowanie projektu	23
4.1	Testowanie funkcjonalności uwierzytelniania kodem tekstowym i zarządza- nia użytkownikami	23
4.2	Testowanie algorytmu wykrywającego twarz	23
4.3	Testowanie algorytmu rozpoznającego twarz	24
5	Podsumowanie	25
	Bibliografia	25

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.

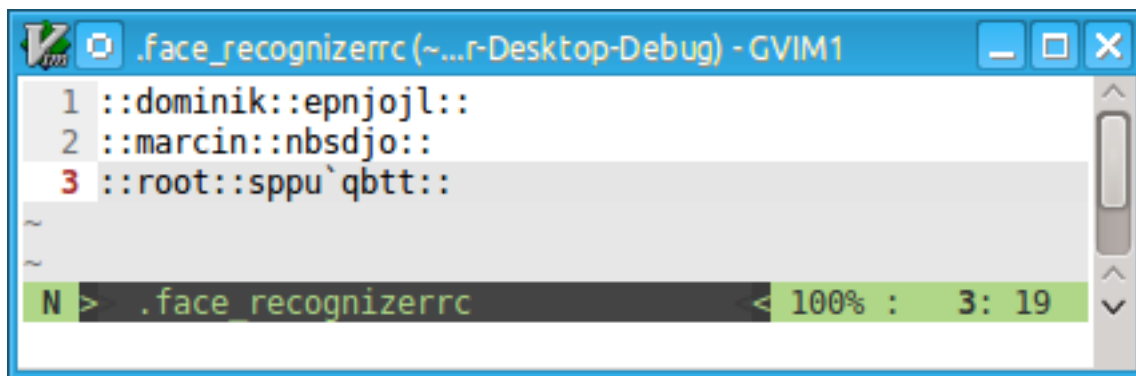
1.3 Koncepcja rozwiązania problemu

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::
```



Rysunek 1.1 Format pliku przechowującego login i hasło

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

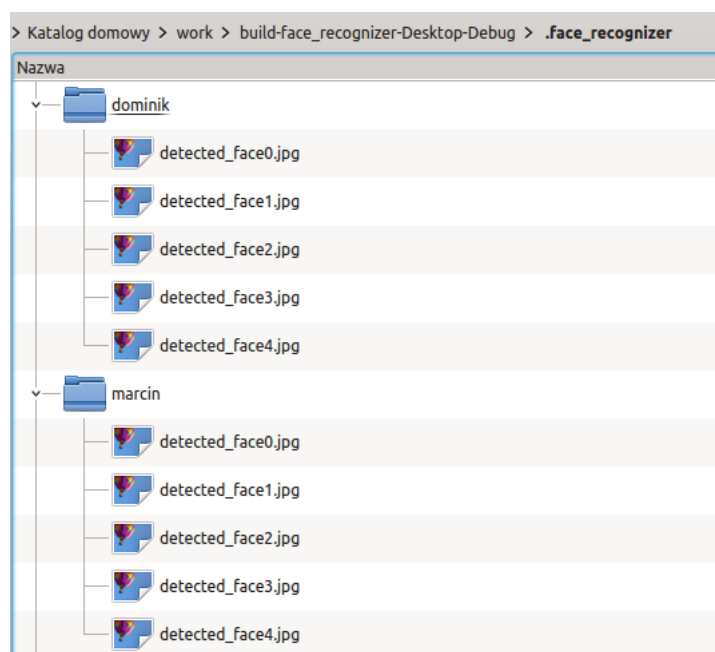
zastosowane przy projektowaniu interface'u do sterowania inteligentnym budynkiem lub pomieszczeniem.

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.



Rysunek 1.2 Format katalogu przechowującego wzorcowe zdjęcia twarzy

1.3.3 Konto administratora

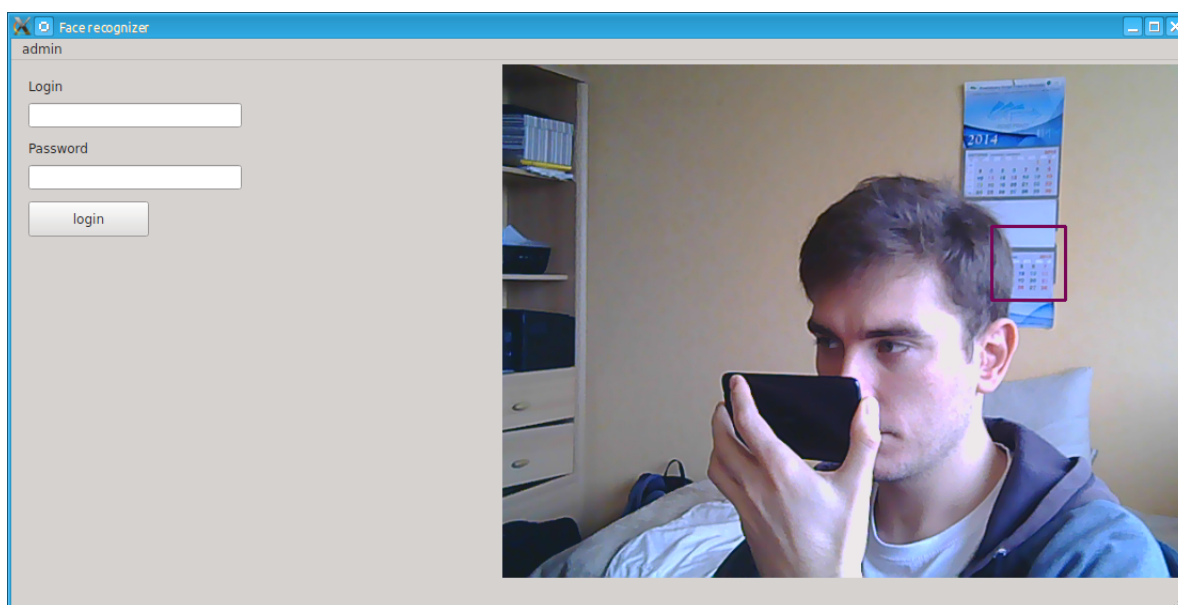
Wyjątkowym przypadkiem, o którym należy wspomnieć jest konto administratora(login: `root`, domyślne hasło: `pass`). Owe konto jako jedyne jest dostępne już przy pierwszym uruchomieniu programu z domyślnym hasłem dostarczonym razem z programem. Bardzo ważna ze względów bezpieczeństwa jest zmiana domyślnego 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.

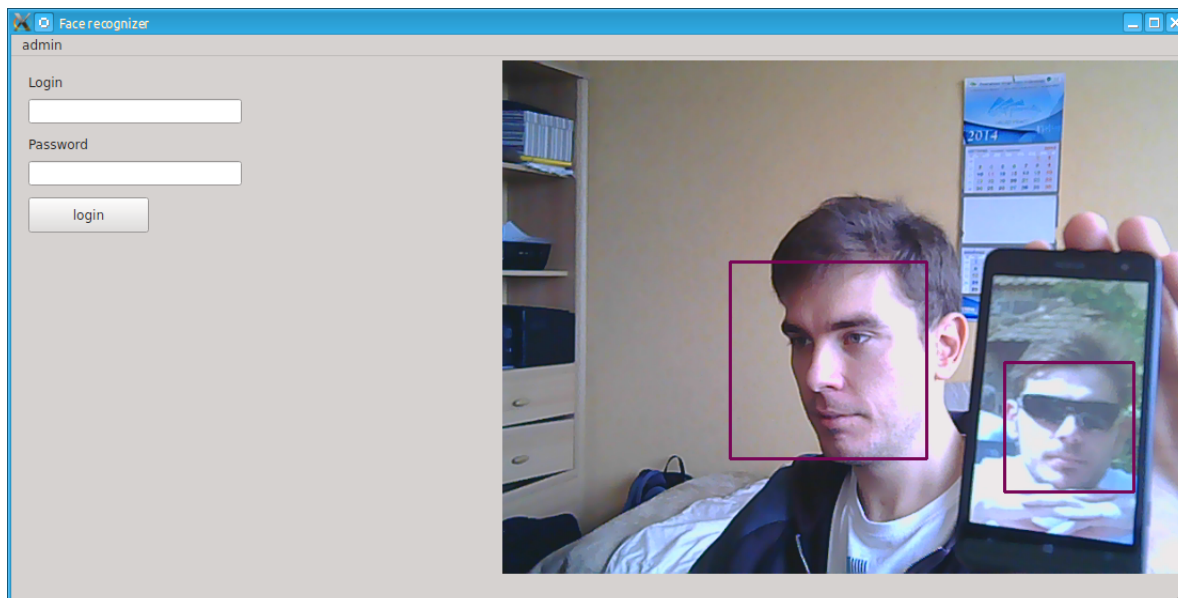


Rysunek 1.3 Twarz wykryta w niewłaściwym miejscu

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.

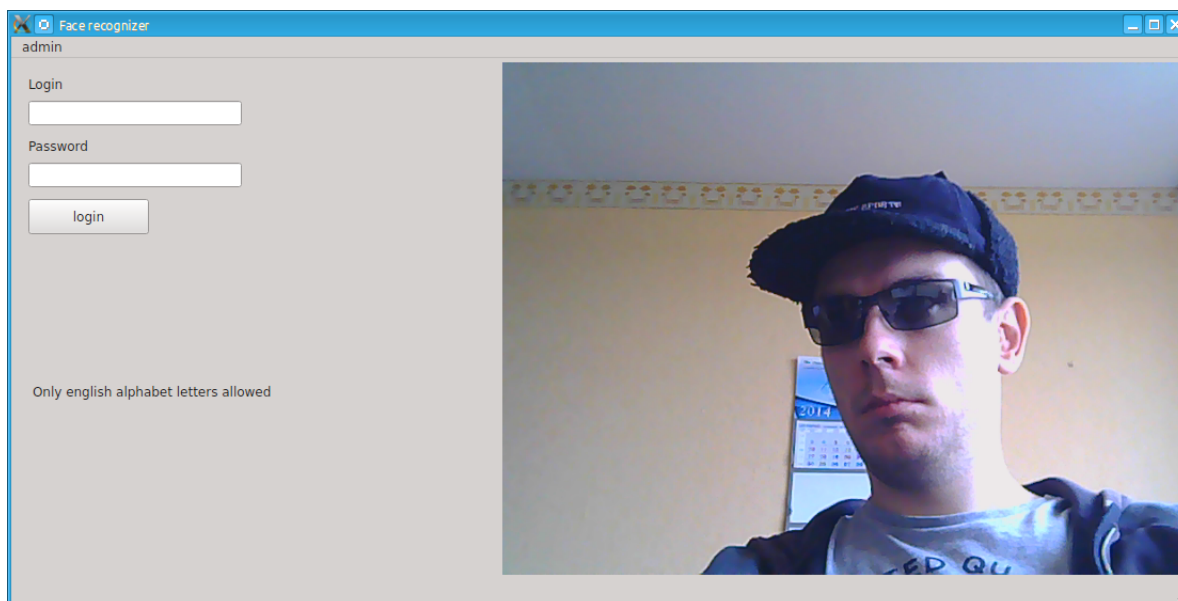
Podobny błąd może powodować umieszczenie przed obiektywem kamery ekranu lcd, na którym wyświetlane jest zdjęcie twarzy. Owym ekranem może być telefon komórkowy, tablet. Co oznacza, że podobny efekt działania algorytmu może być osiągnięty poprzez umieszczenie np. manekina przed obiektywem kamery.

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



Rysunek 1.4 Twarz wykryta na ekranie LCD

dużym problemem, jednak dla algorytmów opierających się na cechach twarzy różny jej kąt względem kamery stanowi wyzwanie.



Rysunek 1.5 Twarz nie wykryta- czapka i okulary

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 wyrównywania 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 Model barw

RGB – jeden z modeli przestrzeni barw, opisywanej współrzędnymi RGB. Jego nazwa powstała ze złożenia pierwszych liter angielskich nazw barw: R – red (czerwonej), G – green (zielonej) i B – blue (niebieskiej), z których model ten się składa. Jest to model wynikający z właściwości odbiorczych ludzkiego oka, w którym wrażenie widzenia dowolnej barwy można wywołać przez zmieszanie w ustalonych proporcjach trzech wiązek światła o barwie czerwonej, zielonej i niebieskiej.

Z połączenia barw RGB w dowolnych kombinacjach ilościowych można otrzymać szeroki zakres barw pochodnych, np. z połączenia barwy zielonej i czerwonej powstaje barwa żółta. Do przestrzeni RGB ma zastosowanie synteza addytywna, w której wartości najniższe oznaczają barwę czarną, najwyższe zaś – białą. Model RGB jest jednak modelem teoretycznym, a jego odwzorowanie zależy od urządzenia (ang. device dependent), co oznacza, że w każdym urządzeniu każda ze składowych RGB może posiadać nieco inną charakterystykę widmową, a co za tym idzie, każde z urządzeń może posiadać własny zakres barw możliwych do uzyskania.

Model RGB miał pierwotnie zastosowanie do techniki analogowej, obecnie ma również do cyfrowej. Jest szeroko wykorzystywany w urządzeniach analizujących obraz (np. aparaty cyfrowe, skanery) oraz w urządzeniach wyświetlających obraz (np. telewizory, monitory komputerowe).

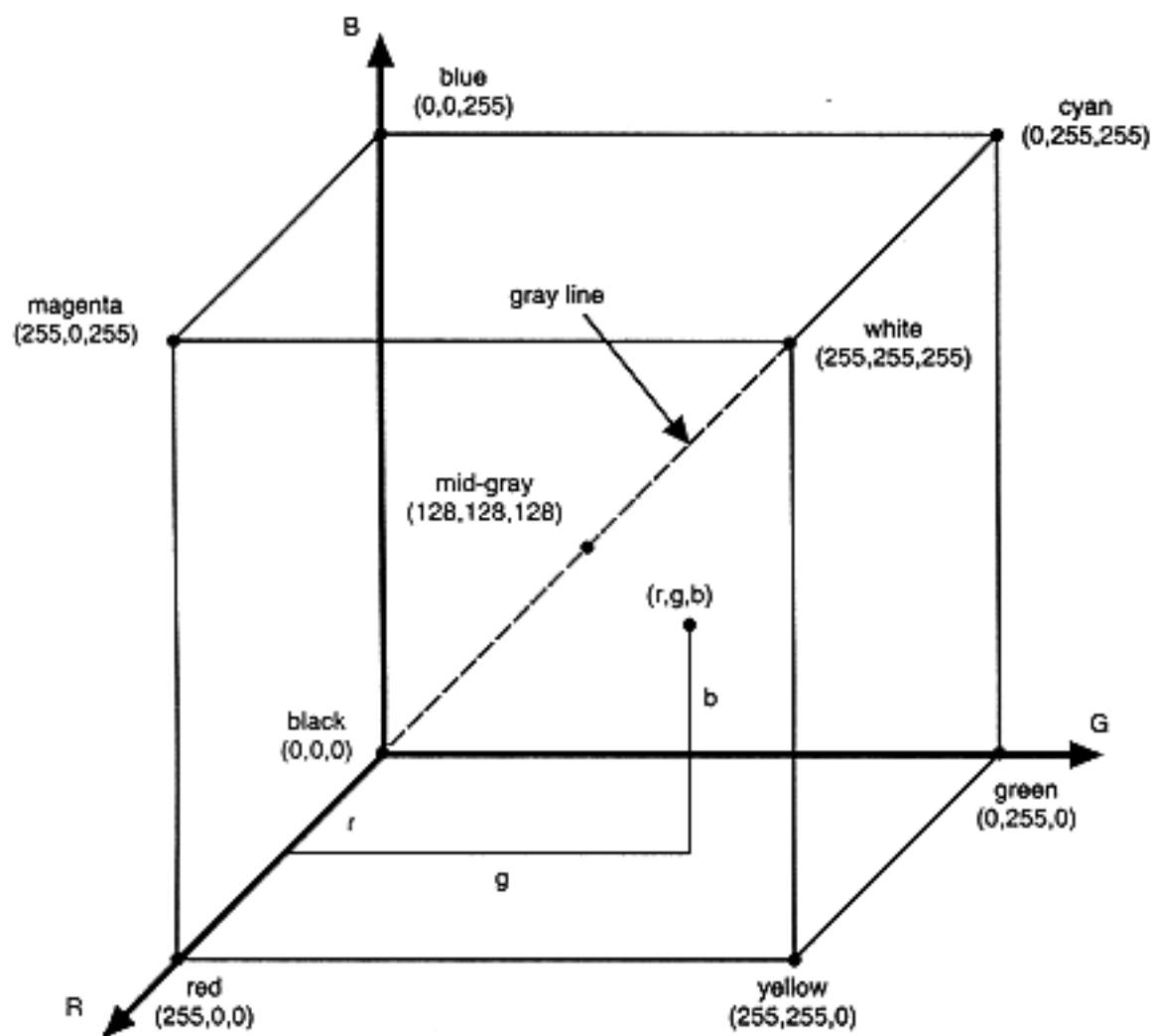
Zapis koloru jako RGB często stosuje się w informatyce (np. palety barw w plikach graficznych, w plikach html). Najczęściej stosowany jest 24-bitowy zapis kolorów (po 8 bitów na każdą z barw składowych), w którym każda z barw jest zapisana przy pomocy składowych, które przyjmują wartość z zakresu 0-255. W modelu RGB wartość 0 wszystkich składowych daje kolor czarny, natomiast 255 – kolor biały. W rzadszych przypadkach stosuje się model, w którym przypada po 12 lub 16 bitów na każdą ze składowych, co daje dużo większe możliwości przy manipulowaniu kolorem.[14]

Kolor RGB można obliczyć ze wzoru:

$$\text{numer koloru} = R * 65536 + G * 256 + B$$

gdzie R, G i B przyjmują wartość od 0 do 255.

Oprócz wspomnianego modelu istnieje także BGR(używany np. w OpenCV), który różni się od wyżej opisanego kolejnością następowania po sobie 3 bazowych kolorów.



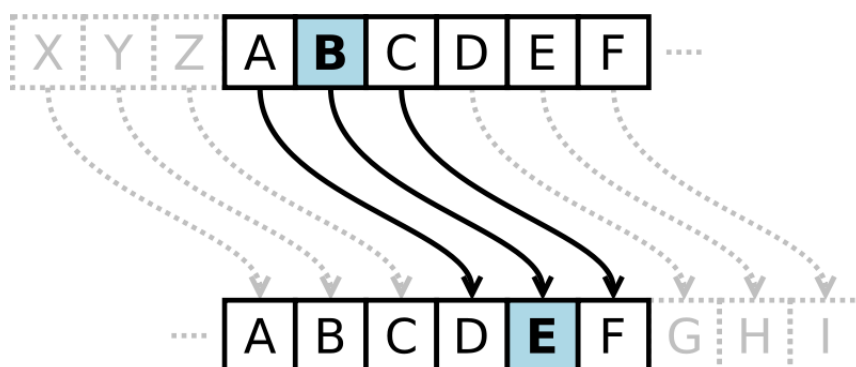
Rysunek 2.1 Przestrzeń RGB

2.2 Identyfikacja twarzy

Klasyfikator Haar'a jest to element zaimplementowany w bibliotece OpenCV. Może służyć do wykrywania obiektów na obrazie. W projekcie posłużył do wykrywania twarzy. Klasyfikator bazuje na kaskadzie stworzonej z wektora uczącego. Kaskada ta, w formie pliku xml z danymi jest dostarczana jako część biblioteki OpenCV. Bazując na zaimplementowanym w bibliotece algorytmie oraz dostarczonym plik z ciągami uczącymi pozwala na wykrycie twarzy na zdjęciu. Wyszukiwanie dla całego obrazu wejściowego realizowana jest przy pomocy koncepcji okna. Prostokąt o szerokości i wysokości mniejszej niż obraz wejściowy przesuwany jest po obrazie wejściowym. Algorytm wyszukiwania obiektu jest realizowany na tak zdefiniowanym przez prostokąt fragmencie obrazu. Klasyfikator można zakwalifikować jako algorytm o dużym obciążeniu, ale także o dużej dokładności działania.

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]



Rysunek 2.2 Szyfr Cezara

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.

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 Konfiguracja środowiska programistycznego Qt

Przed przystąpieniem do pisania projektu korzystającego z biblioteki OpenCV w technologii qt powinno się przeprowadzić kilka operacji, by kompilacja jak i linkowanie przebiegało pomyślnie. Zakładając, że w systemie są zainstalowane wyżej wspomniane biblioteki oraz oprogramowanie Qt Creator:

Po uruchomieniu programu Qt Creator należy z menu kontekstowego File wybrać opcję New file or project, a następnie Applications/Qt Widgets Application

Po wybraniu nazwy oraz lokalizacji projektu, należy przejść do katalogu z projektem i otworzyć plik nazwa_projektu.pro:

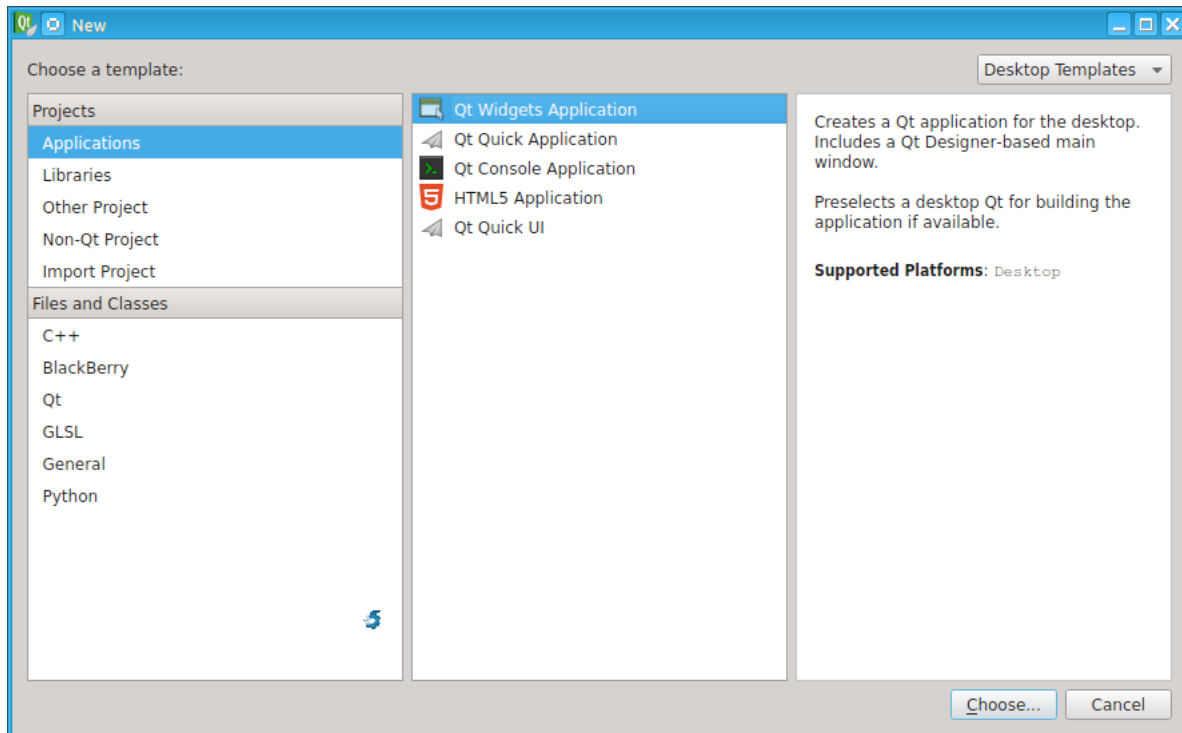
i dodać linię:

LIBS += 'pkg-config opencv --libs'

Podobną komendę stosuje się w pliku konfiguracyjnym dla innych bibliotek. Tak wygląda plik pro zrealizowanego programu.

3.3 Podręcznik użytkownika

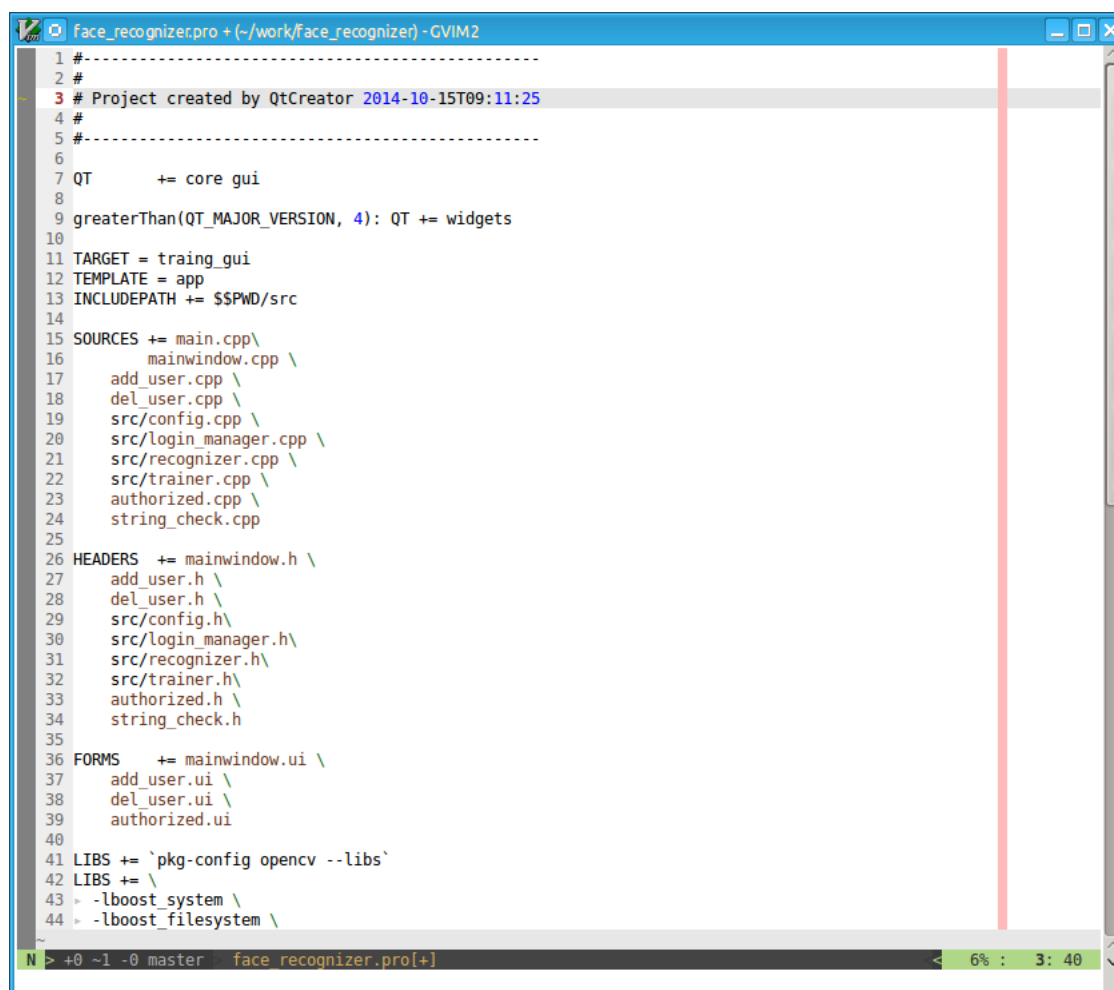
Program został napisany przez Dominik Guderskiego, jako element projektu inżynierskiego. Opracowanie zostało napisane 30.11.2014 r.



Rysunek 3.1 Nowy projekt Qt



Rysunek 3.2 Czysty plik konfiguracyjny pro

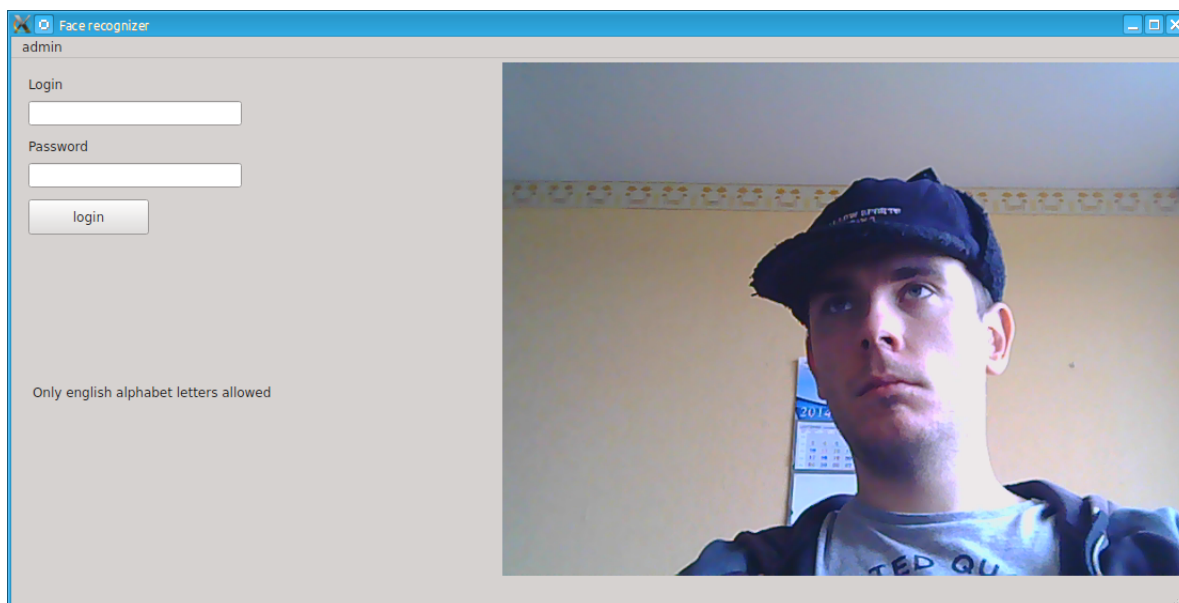


```
1 #-----
2 #
3 # Project created by QtCreator 2014-10-15T09:11:25
4 #
5 #-----
6
7 QT      += core gui
8
9 greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
10
11 TARGET = traing_gui
12 TEMPLATE = app
13 INCLUDEPATH += $$PWD/src
14
15 SOURCES += main.cpp \
16    /mainwindow.cpp \
17     add_user.cpp \
18     del_user.cpp \
19     src/config.cpp \
20     src/login_manager.cpp \
21     src/recognizer.cpp \
22     src/trainer.cpp \
23     authorized.cpp \
24     string_check.cpp
25
26 HEADERS +=/mainwindow.h \
27     add_user.h \
28     del_user.h \
29     src/config.h \
30     src/login_manager.h \
31     src/recognizer.h \
32     src/trainer.h \
33     authorized.h \
34     string_check.h
35
36 FORMS    +=mainwindow.ui \
37     add_user.ui \
38     del_user.ui \
39     authorized.ui
40
41 LIBS += `pkg-config opencv --libs`
42 LIBS += \
43     -lboost_system \
44     -lboost_filesystem \
```

N > +0 ~1 -0 master face_recognizer.pro[+] 6% : 3: 40

Rysunek 3.3 Plik pro zrealizowanego projektu

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:



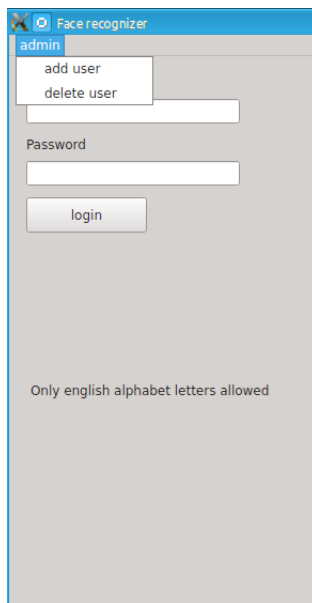
Rysunek 3.4 Główne okno programu

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.

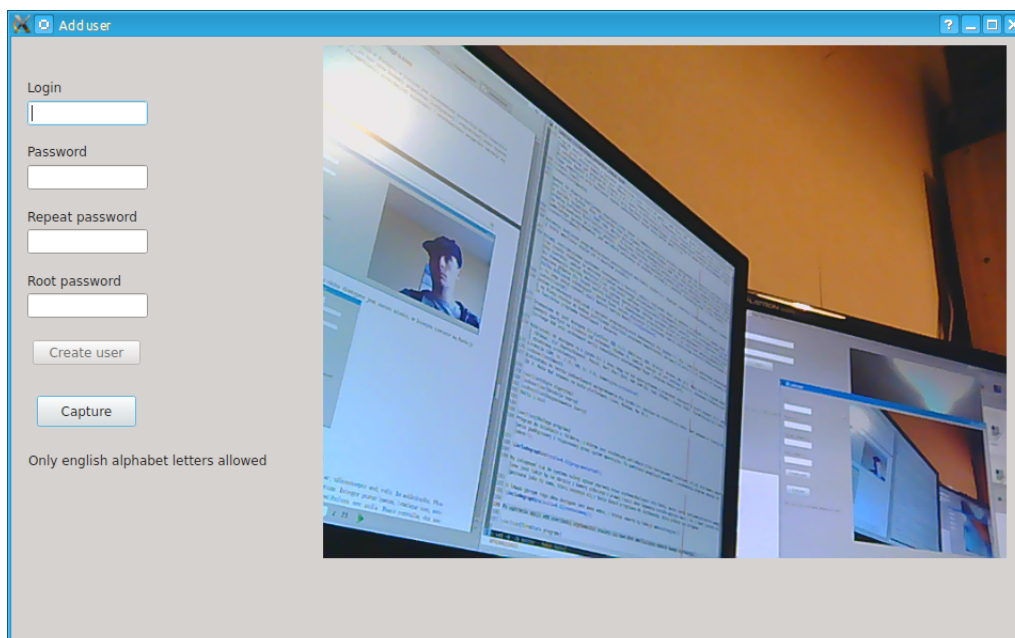
W lewym górnym rogu okna dostępne jest menu admin, w którym zawarte są funkcje administracyjne:

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



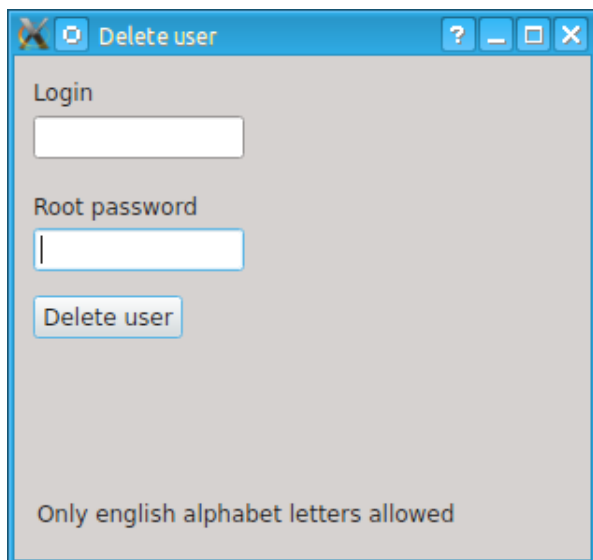
Rysunek 3.5 Menu kontekstowe



Rysunek 3.6 Okno dodania użytkownika

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.



Rysunek 3.7 Okno usunięcia użytkownika

Gdy w głównym oknie programu wpisze 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:

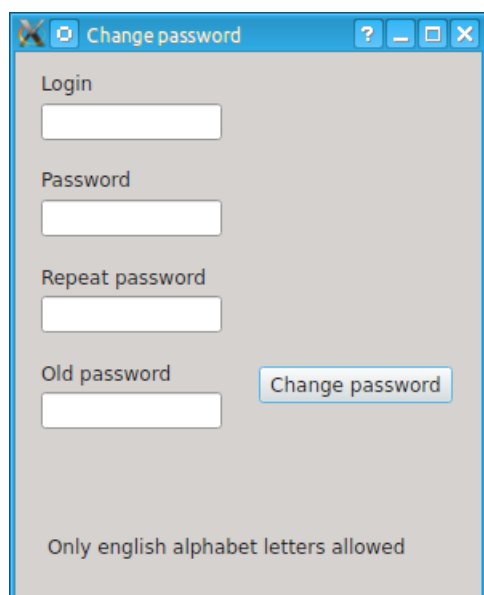
Po poprawnym wypełnieniu wszystkich pól następuje zmiana dotychczasowego hasła na nowe dla wybranego użytkownika.

3.4 Struktura programu

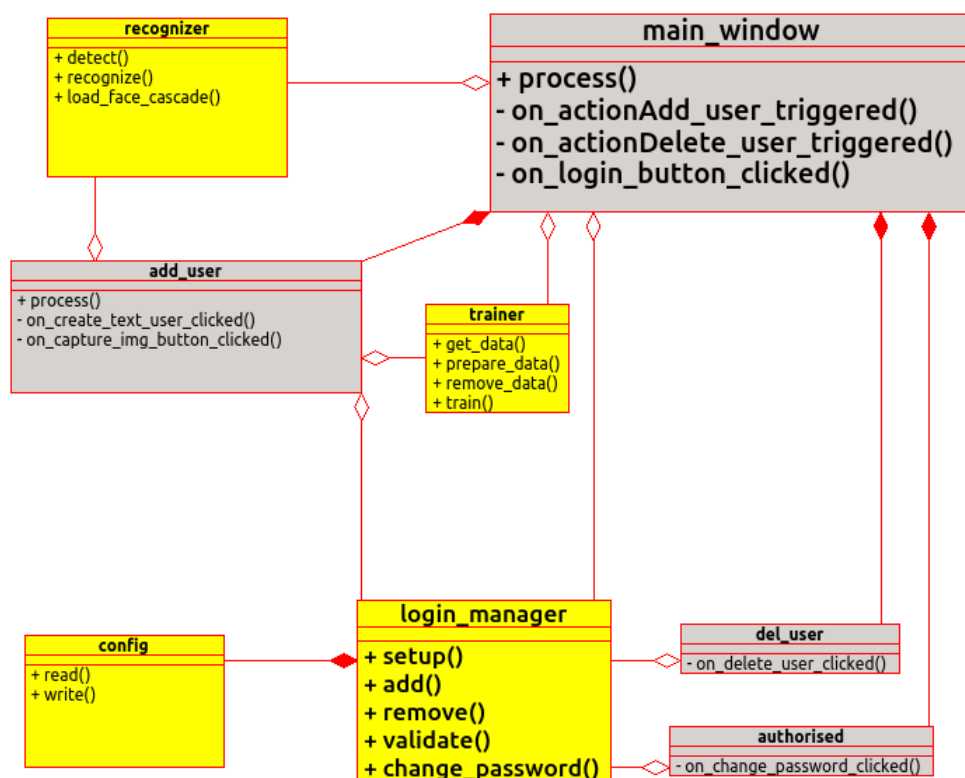
Program można podzielić koncepcyjnie na dwa elementy. Warstwę prezentacji i warstwę logiczną. Warstwę prezentacji stanowi kod odpowiedzialny za tworzenie i obsługę graficznego interfejsu użytkownika. Natomiast warstwa logiczna to klasy realizująca potrzebne algorytmy: komunikacje z bazą danych, kodowanie 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 odmiennych 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



Rysunek 3.8 Zmiana hasła



Rysunek 3.9 Diagram klas programu

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 równolegle 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

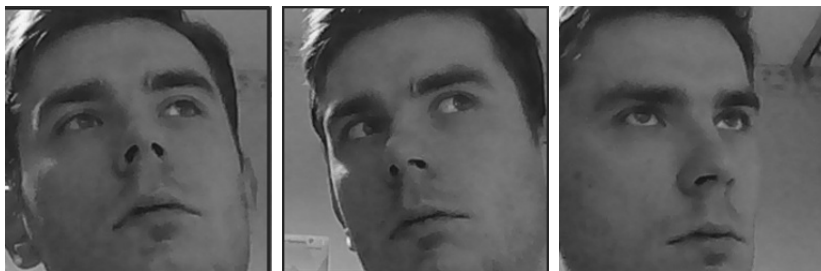
4.1 Testowanie funkcjonalności uwierzytelniania kodem tekstowym i zarządzania użytkownikami

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 otworzyć jednego z plików, który należy do bazy danych.

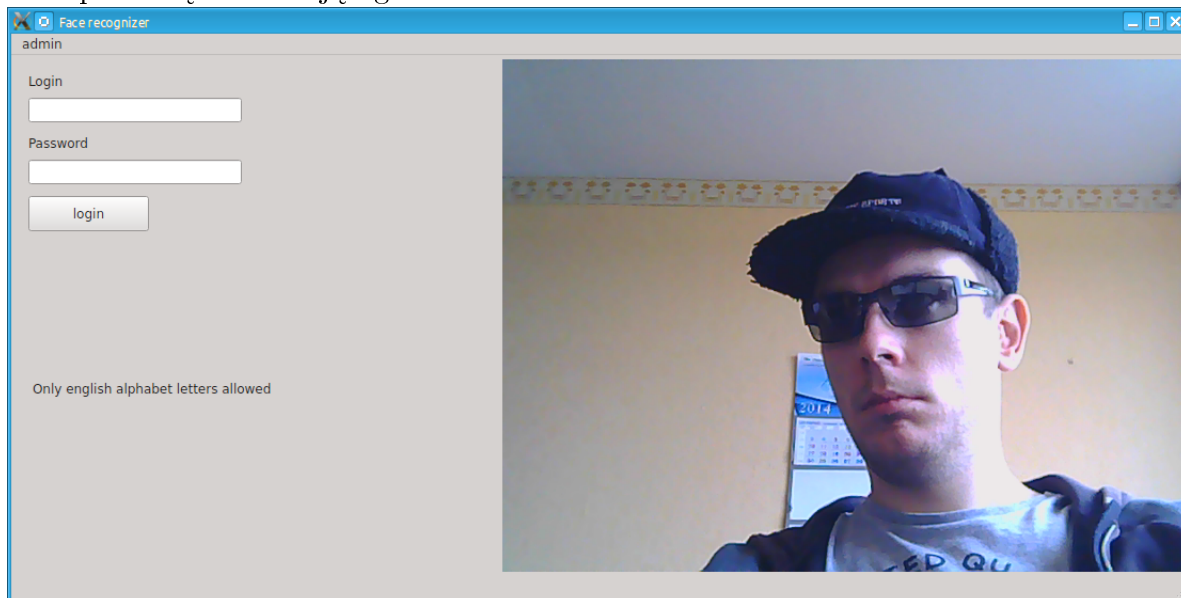
4.2 Testowanie algorytmu wykrywającego twarz

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

Efektem projektu jest aplikacja działająca w systemie graficznym zbudowana w oparciu o technologię Qt realizująca założenia i koncepcję projektu. System kontroli dostępu za pomocą hasła oraz biometrycznego zabezpieczenia- rozpoznawania twarzy został zrealizowany. Program identyfikuje twarze w czasie rzeczywistym. Program rozpoznaje twarze w czasie rzeczywistym. Przy kwestii rozpoznawania twarzy wpływ na skuteczność działania algorytmu mają warunki oświetlenia kąt nachylenia twarzy względem obiektywu kamery oraz odzież, ozdoby znajdujące się na twarzy użytkownika w czasie działania algorytmu.

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
- [14] <http://pl.wikipedia.org/wiki/RGB>

Spis rysunków

1.1	Format pliku przechowującego login i hasło	4
1.2	Format katalogu przechowującego wzorcowe zdjęcia twarzy	5
1.3	Twarz wykryta w niewłaściwym miejscu	6
1.4	Twarz wykryta na ekranie LCD	7
1.5	Twarz nie wykryta- czapka i okulary	7
2.1	Przestrzeń RGB	10
2.2	Szyfr Cezara	11
3.1	Nowy projekt Qt	15
3.2	Czysty plik konfiguracyjny pro	15
3.3	Plik pro zrealizowanego projektu	16
3.4	Główne okno programu	17
3.5	Menu kontekstowe	18
3.6	Okno dodania użytkownika	18
3.7	Okno usunięcia użytkownika	19
3.8	Zmiana hasła	20
3.9	Diagram klas programu	20