Podstawy Telekomunikacji – Dokumentacja Projektu

(ćwiczenia lab. lato 2015/2016)

Imię i nazwisko	Adres e-mail	Data złożenia
Dawid Chróścielski	**dawid.chroscielski@student.put.poznan.pl**	**08.06.2016**
Jacek Przemieniecki	**jacek.przemieniecki@student.put.poznan.pl**	**08.06.2016**
Tomasz Lewandowski	**tomasz.wo.lewandowski@student.put.poznan.pl**	**08.06.2016**

1. Temat projektu:

Aplikacja uwierzytelniająca użytkownika na podstawie zdjęcia twarzy.

2. Przydział zadań członkom zespołu:

LP	Imię i nazwisko	Zadania	Rodzaj zadania (merytoryczne/organizacyjne)
1	**Dawid Chróścielski**	**Przechwycenie obrazu z kamery oraz jego preprocessing**	**merytoryczne**
2	**Jacek Przemieniecki**	**Interfejs aplikacji demonstracyjnej z wykorzystaniem biblioteki przetwarzającej** **Translacja projektu na język C++**	**merytoryczne**
3	**Tomasz Lewandowski**	**Utworzenie biblioteki z kluczowymi funkcjonalnościami systemu (wykrywanie i rozpoznawanie twarzy)**	**merytoryczne**

3. Metodologia projektowania i implementowania

Metodologia pracy, która została zastosowana, to metodologia Kanban. Wybór ten w znacznym stopniu podyktowany jest naszymi indywidualnymi przekonaniami/doświadczeniami. Spotkania między członkami projektu odbywały się w razie realnej potrzeby, natomiast komunikacja w zespole przebiegała głównie poprzez komunikator portalu społecznościowego.

4. Omówienie wykonanych dotychczas prac związanych z zapoznawaniem się z problematyką projektu, w tym uwagi dotyczące rozpoznanych programów źródłowych i ich dokumentacji:

a) temat rozpoznanego zagadnienia:

Autentykacja użytkownika za pomocą zdjęcia jego twarzy.

b) omówienie rozpoznanego zagadnienia

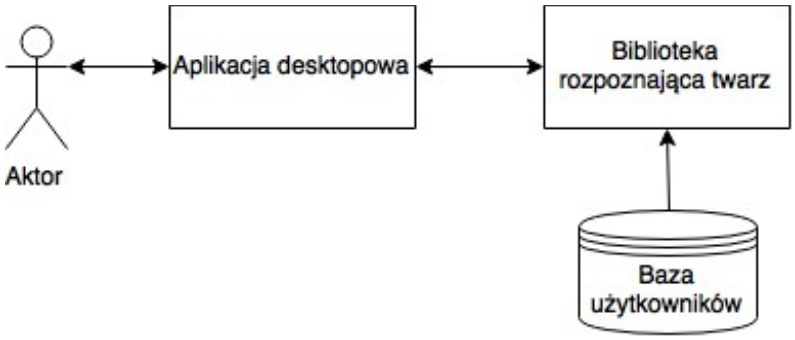
Projekt opiera się na platformie Microsoft Windows oraz technologiach .NET, EmguCV, OpenCV oraz System.Data.SQLite. Projekt początkowo zrealizowany został w języku C#, a po sugestii prowadzącego przepisany został na język C++. OpenCV umożliwia użycie trzech algorytmów rozpoznawania twarzy EigenFaces, FisherFaces i LBPH, jednakże tylko ostatni z nich pozwala na douczanie sieci. Pozostałe dwa należałoby uczyć przed każdym rozpoznaniem od początku, co wymagało by dużej mocy obliczeniowej, a także przechowywania zdjęć użytkowników czego autorzy projektu chcieli uniknąć. Algorytm LBPH polega na podzieleniu obrazu na komórki, a następnie porównywanie każdego piksela w komórce z jego sąsiadami. Jeżeli wartość piksela jest

większa od wartości piksela sąsiada rezultatem jest 0 natomiast w przeciwnym wypadku 1. Po porównaniu ze wszystkimi sąsiadami otrzymywana jest ośmiobitowa liczba. Następnie dla komórki obliczany jest histogram, w rezultacie czego otrzymywany jest 256-wymiarowy wektor cech. Po połączeniu wektorów cech poszczególnych komórek otrzymywany jest wektor cech całego obrazu.

c) wykorzystane źródła wiedzy

- Dokumentacja EmguCV
- Dokumentacja OpenCV
- Dokumentacja System.Data.SQLite

5. Rysunek poglądowy aplikacji demonstracyjnej wykonywanej przez zespół projektowy



(legenda:

- ikonka użytkownika (np. jak aktor w UMLu),
- prostokąt: moduł programowy aplikacji
- strzałka pojedyncza linią ciągłą: powiązanie danych z modułem programowym
- strzałka podwójna: przepływ sterowania pomiędzy modułami programowymi,
- walec: baza danych jako całość)

6. Specyfikacja ważniejszych algorytmów

Algorytm obróbki wstępnej zdjęcia

Wejście: Zdjęcie w przestrzeni barw BGR

Wyjście: Fragment zdjęcia zawierający twarz w skali szarości

Metoda:

1. Fotografia konwertowana jest do skali szarości (1B/piksel)
2. Za pomocą klasyfikatora kaskadowego z biblioteki OpenCV wyznaczany jest prostokąt zawierający twarz.
3. Wynikiem jest struktura ze wskaźnikiem na fotografię z informacją o podregionie zawierającym twarz.

Algorytm uczenia

Wejście: Seria zdjęć (w aplikacji desktopowej 4) po wstępnej obróbce, ID użytkownika

Wyjście: Plik z zapisaną siecią neuronową służącą do rozpoznawania twarzy.

Metoda:

1. Jeżeli plik z siecią dla danego użytkownika już istnieje, (opcjonalnie – rozpakuj go i) wczytaj zapisaną w nim sieć. Jeżeli plik nie istnieje utwórz nową sieć.

2. Doucz sieć zdjęciem z ID użytkownika jako klasyfikatorem.
3. Zapisz sieć (i opcjonalnie skompresuj, usuń nieskompresowany plik).

Algorytm rozpoznawania

Wejście: Zdjęcie po wstępnej obróbce.

Wyjście: ID rozpoznanego użytkownika lub -1 jeśli nie rozpoznano twarzy.

Metoda:

1. Zainicjalizuj BestId ← -1, BestDistance ← nieskończoność
2. Dla każdego pliku z zapisaną siecią (opcjonalnie – rozpakuj ją i):
    1. Wykonaj klasyfikację na podstawie otrzymanego zdjęcia.
    2. Jeżeli dystans klasyfikacji jest niższy niż BestDistance, zapisz BestId ← rozpoznany ID, BestDistance ← rozpoznany dystans
    3. Opcjonalnie usuń rozpakowany plik
3. Wynikiem jest BestId

7. Środowisko realizacji aplikacji

- Microsoft Windows
- C#
- Windows Forms
- EmguCV
- System.Data.SQLite
- C++
- OpenCV
- OpenGL
- Dear ingui

8. Omówienie implementacji

Detector
Interfejs Detector zawiera metody służące do znalezienia twarzy na zdjęciach (Detect) oraz wyodrębnieniu tych twarzy jako osobnych obrazów (Extract, ExtractLargest).

CascadeDetector
Klasa implementująca interfejs Detector. Do wykrywania twarzy wykorzystuje CascadeClassifier dostarczony przez EmguCV oraz OpenCV. Wykonuje dodatkowo na zdjęciu operacje takie jak konwersja kolorów do skali szarości oraz wyrównywanie histogramów.

ImageProcessor
Interfejs zawierający funkcję składową process, służącą do obróbki zdjęcia.

FaceImagePreprocessor
Klasa implementująca interfejs ImageProcessor. Służy do obróbki zdjęcia twarzy użytkownika (konwersja kolorów, wyrównywanie histogramów, zmiana rozmiaru do z góry ustalonego) przed przystąpieniem do uczenia systemu lub rozpoznawania użytkownika.

UserFaceRecognizer
Interfejs zawierający metody umożliwiające uczenie (train) oraz rozpoznawanie twarzy użytkowników spośród wszystkich, lub z pośród wybranych użytkowników na podstawie dostarczonych zdjęć (recognize). Przewiduje także zarządzanie danymi konkretnych użytkowników (removeUserData).

Uczenie polega na podaniu identyfikatora użytkownika oraz zdjęcia jego twarzy. W przypadku rozpoznawania daną wejściową jest zdjęcie użytkownika, natomiast zwracaną wartością identyfikator rozpoznanego użytkownika, lub w przypadku gdy nie odnaleziono pasującego użytkownika zwracana jest wartość -1.

LBPHUserFaceRecognizer
Klasa implementująca interfejs UserFaceRecognizer. Do rozpoznawania twarzy wykorzystuje LBPHFaceRecognizer (Local Binary Pattern Histogram) dostarczony przez EmguCV. Metoda ta opiera się o Local Binary Pattern's. Użycie powyższego algorytmu umożliwia łatwe douczanie systemu na podstawie dodatkowych danych. System umieszcza wektory cech w plikach w katalogu określonym przez użytkownika biblioteki. Dla każdego użytkownika tworzony jest jeden plik. Ma to na celu łatwą modyfikację danych i umożliwia ewentualne usunięcie danych konkretnego użytkownika. Gdyby dane systemu przechowywane były w jednym pliku zabieg taki stałby się niemożliwy bez uczenia systemu na nowo, co we wielu przypadkach jest niemożliwe. Podczas rozpoznawania klasa przeszukuje pliki użytkowników i znajduje taki, który daje najlepszy wynik uwzględniając pewne, określone podobieństwo minimalne.

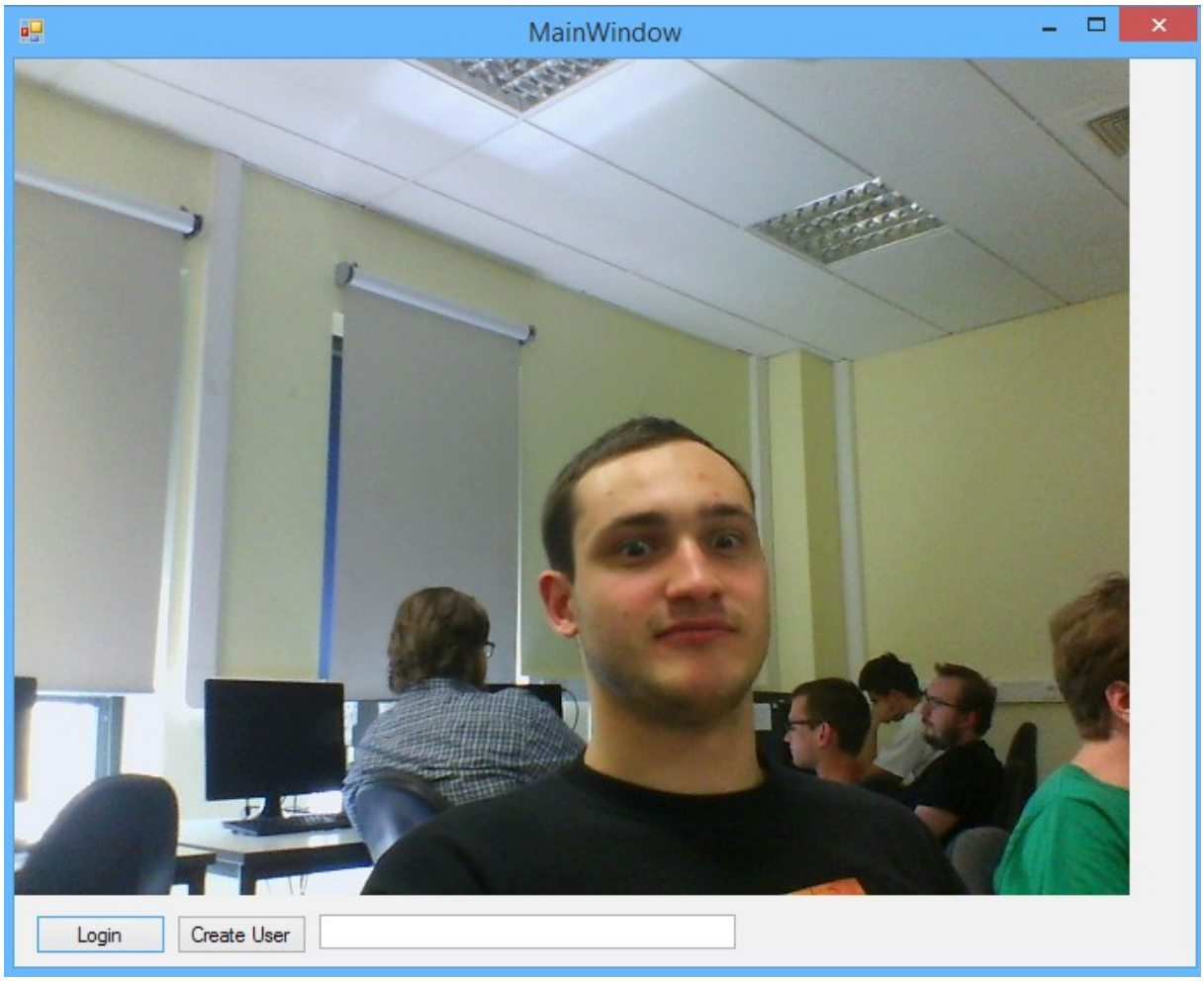
9. Opcjonalna instrukcja użytkowania aplikacji oraz przeprowadzone testy

Zrzut ekranu dla interfejsu użytkownika:

Instrukcja użytkownika aplikacji:

1. Użytkownik uruchamia aplikację.
    a. Rejstracja:
        i. Użytkownik wpisuje swoje imię w pole tekstowe.
        ii. Użytkownik naciska przycisk „Create User”.
        iii. Program wykonuje serję zdjęć użytkownika, celem przyuczenia sieci neuronowej.
    b. Logowanie:
        i. Użytkownik naciska przycisk „Login”.
        ii. Aplikacja wykonuje zdjęcie użytkownika, celem autentykacji.
        iii. Aplikacja wyświetla na ekranie komunikat powitalny z imieniem użytkownika.
2. Użytkownik kończy działanie aplikacji, klikając przycisk zamykający okno.

10. Zrzut ekranu z aplikacji



11. Podsumowanie i możliwe kierunki rozwoju aplikacji

Głównym kierunkiem rozwoju aplikacji jest wykorzystanie biblioteki przygotowanej na potrzeby projektu w realnym, użytkowym projekcie „Inteligentnego Lustra”, gdzie autentykacja użytkownika za pomocą zdjęcia twarzy i komend głosowych, będzie służyła do logowania, celem spersonalizowania interfejsu.