Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

Rozpoznawanie twarzy z obrazu kamery, przy wykorzystaniu biblioteki EmguCV

Wiktor Borowski Paweł Furmaniak Dawid Zborowski

Podstawy Teleinformatyki

Vi Semestr Informatyki specjalność: WTI

Wydział Elektryczny

Rok akademicki 2015/2016

Spis Treści

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

1. Opis projektu

Tematem projektu jest stworzenie programu służącego rozpoznawaniu twarzy. System składa się z modułu wykrywania twarzy, który ma za zadanie

przechwycić obraz z kamery a następnie wykryć na niej twarz, by następnie porównać go z nauczonymi już twarzami.

Drugi moduł, to moduł treningowy, który ma za zadanie „uczyć się", czyli po wykryciu twarzy możliwa jest opcja zrobienia serii zdjęć w celu nauczenia się twarzy osoby.

2. Wykorzystane technologie

Realizacja projektu została wykonana w platformie .NET. Do celów przetwarzania obrazu została użyta biblioteka EmguCV która jest modyfikacją biblioteki OpenCV przeznaczoną dla języka C#. Wizualizacja wykorzystuje techonolgię Windows Forms przeznaczoną do tworzenia aplikacji okienkowych na platformę Windows.

3. Wymagania systemowe oraz sprzętowe

W celu poprawnego uruchomienia aplikacji wymagany jest 32 (lub 64) bitowy system Windows z zainstalowanym .NET Framework 4.5. Aplikację zaleca się uruchamiać na dedykowanej karcie 2|17 graficznej która zapewni płynniejszą i szybszą realizcję modułu przetwarzającego obraz. Minimalna rozdzielczość kamery wykorzystywanej do przechwytywania obrazu to 640x480 (wymagania biblioteki)

4. Przetwarzanie obrazu

Na potrzeby projektu została stworzona zakupiona kamerka internetowa firmy Logitech. Obraz z kamery przechwytywany jest przez naszą aplikację. Gdy aplikacja wykryje twarz rysuje wokół niej prostokąt, jeśli twarz zostanie zidentyfikowania (znaleziona korelacja z istniejącymi już rekordami w naszej „bazie danych" prostokąt zostaje podpisany imieniem, wraz z prawdopodobieństwem pewności. W celu poprawnego działania algorytmów służących do rozpoznawania, kamera musi

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

znajdować się bezpośrednio przed twarzą, oraz warunki oświetleniowe muszą być dobre (nie może być za jasno, oraz twarz musi kontrastować z tłem, jasne tło pogarsza bądź wręcz uniemożliwia wykrycie twarzy)

4.1 Klasa Capture odpowiedzialna jest za przechwytywanie i przetwarzanie obrazu z kamery.

Wbudowana ona jest w bibliotekę EmguCV. Umożliwia przechwytywanie obrazu z kamery

podłączonej do komputera. Klasa pozwala przetwarzać przechwycony

obraz pod kątem wyszukiwania na nim obiektów, markerów czy identyfikacji kolorów.

4.2 Opis obiektu haar – obiekt haar jest typu CascadeClasifier, jest to typ z biblioteki EmguCV, który po załadowaniu odpowiedniego pliku XMLowego z regułami służy detekcji obiektów. W naszym przypadku jest to plik dołączony do biblioteki - frontface_default, czyli jak sama nazwa wskazuje, służy do detekcji twarzy zwróconej prosto do kamery. Biblioteka pozwala na definiowanie własnych XMLi z regułami.

4.3 Opis obiektu frame – obiekt frame jest obiektem typu Image<Bgr, Byte> - jest to klatka przechwycona z kamery w postaci tablicy bitów. Do tego obiektu przesyłany jest obraz z kamery z częstotliwością 30 FPS.

4.4 Opis obiektu grayFrame – obiekt jest tego samego typu, co frame, jednak jest rzutowany w skali szarości, jest to wymóg biblioteki EmguCV. Każdy obraz przetwarzany przez EmguCV musi być przetworzony do skali szarości, stąd pojawia się problem z detekcją przy jasnym tle i mocnym świetle.

4.5 Opis obiektów CurrentFace i CurrentFaceGray – obiekty takiego samego typu jak frame – trafiają do nich odpowiednio przycięte (tak by zmieściła się w nich wykryta twarz + kilka jednostek więcej dla pewności) twarze wykryte na przechwyconej klatce z kamery (dla CurrentFaceGray, przekształcone do skali szarości)

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

5 Obiekty służące rozpoznawaniu:

5.1. Struktura RecognizeResults – w tej strukturze składowane są dwa pola – nieobiektowy string name, służący przechowywaniu imienia wykrytej osoby, oraz wartość zmiennoprzecinkowa distance określające prawdopodobieństwo prawidłowego rozpoznania.

5.2 Klasa Recognizer, dziedzicząca interfejs IDisposable, w celu zwolnienia wykorzystywanych zasobów po skończeniu przetwarzania.

5.2.1 TRAINED_FACES_PATH - Zawiera ścieżke do nauczonych już twarzy.

5.2.2 LBPHFaceRecognizer recognizer – służy do zapisywania w odpowiednim formacie rekordów w xmlowym pliku haar.

5.2.3 Listy faces i names – przechowują bitową reprezentację twarzy oraz ich podpisy.

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

6. Listingi Kodu – umieściliśmy tutaj definicje najważniejszych metod

6.1 Przetwarzanie ramki,dla wykrycia twarzy:

```
1.  private void ProcessFrame(object sender, EventArgs e)
2.          {
3.              frame = capture.QueryFrame().ToImage<Bgr, Byte>().Resize(320, 240, Emgu.CV.CvEnum.Inter.Cubic); // przetworzenie klatki do
    obiektu Image EmguCV
4.
5.              grayFrame = frame.Convert<Gray, Byte>(); // przetworzenie do skali szarości
6.
7.              Rectangle[] facesDetected = haar.DetectMultiScale(grayFrame, 1.2, 10, new Size(50, 50), Size.Empty); // do rysowania konturów w
    okół twarzy
8.
9.              if (facesDetected == null || facesDetected.Length == 0)
10.             {
11.                 return;
12.             }
13.
14.             Rectangle face = facesDetected[0];
15.             // zmniejszenie obrazu, aby zmiescil sie na pictureBox2
16.             face.X += (int)(face.Height * 0.15);
17.             face.Y += (int)(face.Width * 0.22);
18.             face.Height -= (int)(face.Height * 0.3);
19.             face.Width -= (int)(face.Width * 0.35);
20.
21.             if (currentState != RecordState.Stop)
22.             {
23.                 currentFaceGray = grayFrame.Copy(face).Resize(100, 100, Emgu.CV.CvEnum.Inter.Cubic);
24.                 currentFaceGray._EqualizeHist();
25.                 pictureBox2.Image = currentFaceGray.ToBitmap(); // przesłanie przetworzonej twarzy do pictureBox'a podglądowego.
26.             }
27.
28.             frame.Draw(face, new Bgr(Color.Red), 2); // rysuje kontury twarzy
29.
30.             switch (currentState)
31.             {
32.                 case RecordState.Rec: // gdy wykryta jest jedna twarz
33.                     faces.Add(currentFaceGray);
34.                     currentState = RecordState.Stop;
35.                     break;
36.
37.                 case RecordState.RecMulty: // gdy wiele
38.                     faces.Add(currentFaceGray);
39.                     if (faces.Count == MULTY_FACE_COUNT)
40.                     {
41.                         currentState = RecordState.Stop;
42.                     }
43.                     break;
44.                 default:
45.                 case RecordState.Stop:  // gdy nic nie wykryte
46.                     break;
47.             }
48.
49.             pictureBox1.Image = frame.ToBitmap(); // przesłanie na głównego pictureBoxa twarzy, wraz z obrysowaniem
50.         }
```

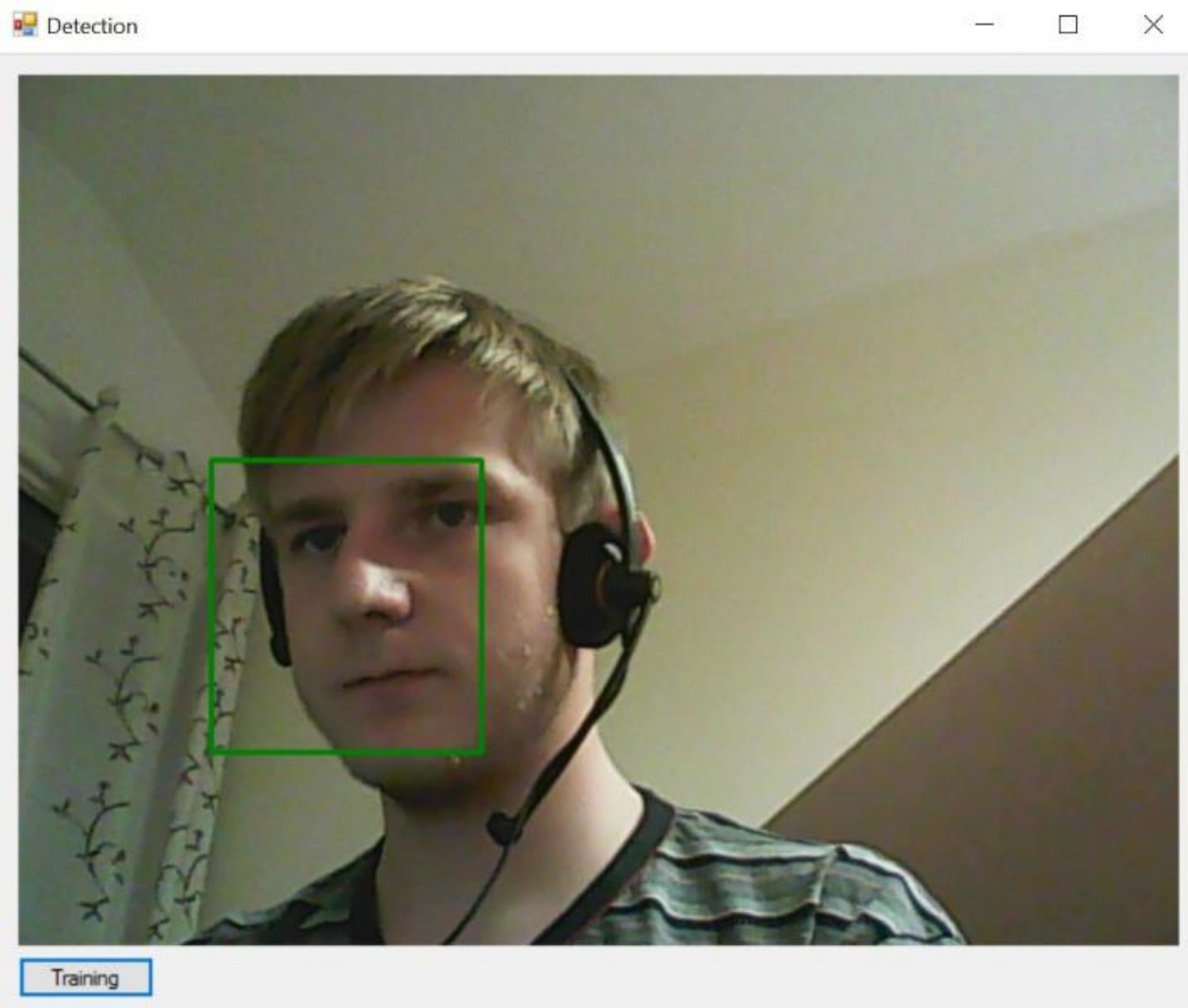
Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

6.2 Przetwarzanie ramki dla rozpoznania twarzy:

```
1.  private void ProcessFrame(object sender, EventArgs arg)
2.          {
3.              frame = capture.QueryFrame().ToImage<Bgr, Byte>(); // przechwycona klatka
4.              grayFrame = frame.Convert<Gray, Byte>(); // konwersja dla odcieni szarości
5.
6.              Rectangle[] facesDetected = haar.DetectMultiScale(grayFrame, 1.2, 10, new Size(50, 50), Size.Empty); // kontur prostokąta
7.
8.              for (int i = 0; i < facesDetected.Length; i++) // dla wszystkich wykrytych na ramce twarzy
9.              {
10.                 // obramowania
11.                 facesDetected[i].X += (int)(facesDetected[i].Height * 0.1); //0.15
12.                 facesDetected[i].Y += (int)(facesDetected[i].Width * 0.22);
13.                 facesDetected[i].Height -= (int)(facesDetected[i].Height * 0.3);
14.                 facesDetected[i].Width -= (int)(facesDetected[i].Width * 0.35);
15.
16.                 currentFace = frame.Copy(facesDetected[i]); // przypisanie do zmiennej tymczasowej twarzy w rgb i odcieniach szarości
17.                 currentFaceGray = grayFrame.Copy(facesDetected[i]).Resize(100, 100, Emgu.CV.CvEnum.Inter.Cubic); // jak wyżej
18.                 currentFaceGray._EqualizeHist(); // histogram
19.
20.                 frame.Draw(facesDetected[i], new Bgr(Color.Green), 2);
21.
22.                 if (recognizer.IsTrained) // jeżeli jest dopasowanie
23.                 {
24.                     var result = recognizer.Recognize(currentFaceGray);
25.                     frame.Draw(result.Name + ", Na : " + (int)result.Distance+"%", // podpisanie konturów (prostokata)
26.                         new Point(facesDetected[i].X , facesDetected[i].Y ),
27.                         Emgu.CV.CvEnum.FontFace.HersheyComplex,
28.                         1d,
29.                         new Bgr(0, 255, 0)
30.                         );
31.                 }
32.             }
33.
34.             pictureBox1.Image = frame.ToBitmap();
35.         }
```

7. Interfejs użytkownika

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny

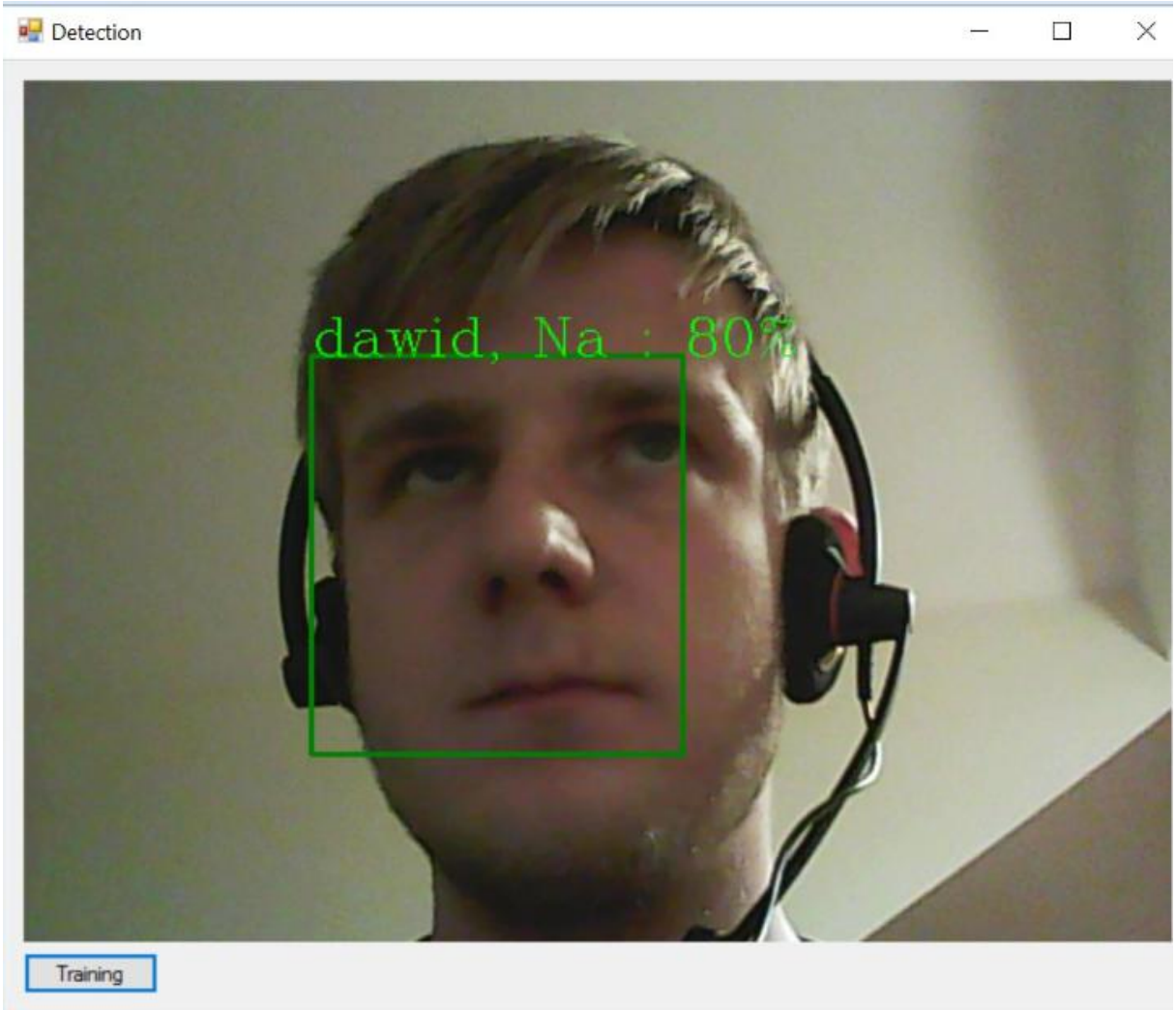


Jeżeli wykryta twarz nie znajduje się w bazie twarzy, obramowana zostaje prostokątem. Poniżej przechwyconego obrazu z kamery znajduje się przycisk Training, który przekierowuje do Form'a odpowiedzialnego za dodawanie nowych rekordów do bazy danych.

W przypadku wykrycia twarz zostaje podpisana, wraz z podaniem dokładności wykrycia.

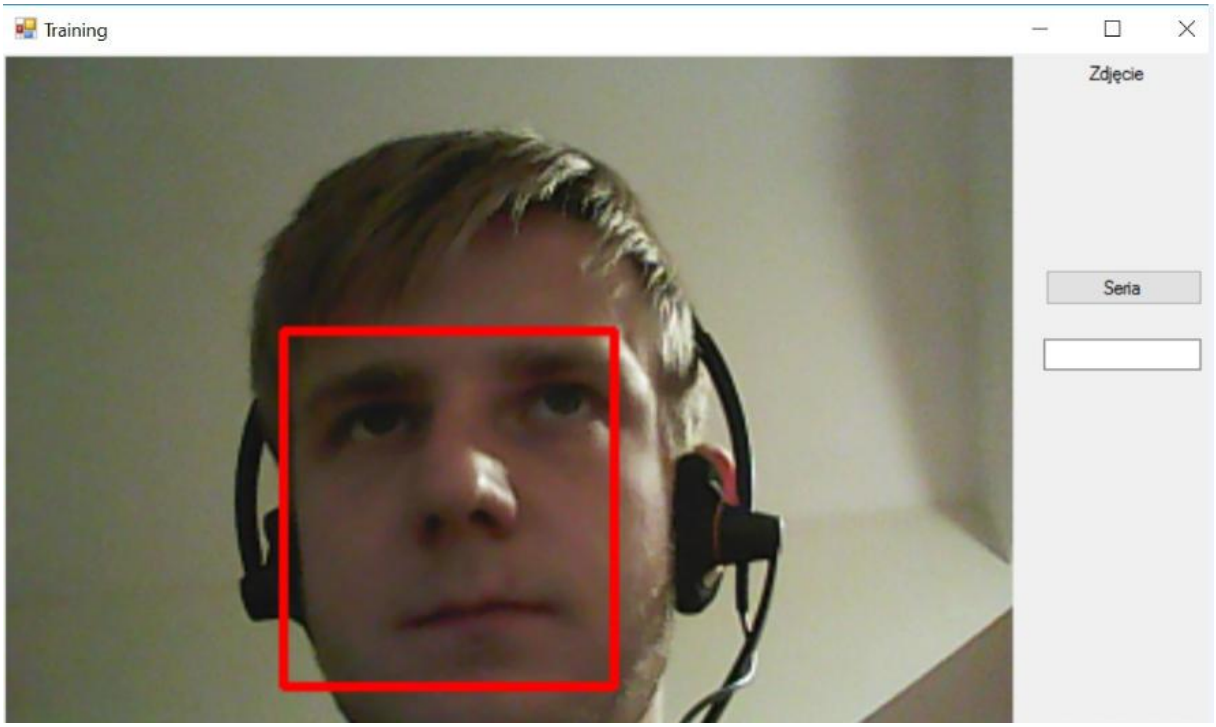
Oczywistym jest, że im więcej rekordów w bazie, tym większa dokładność wykrycia. Poniżej przykład, w którym w bazie znajdowało się 5 zdjęć moich, oraz 5 zdjęć kolegi z grupy Wiktora Borowskiego. Jak widać Emgu jest dość pewne swoich przewidywań, jednak im więcej różnych twarzy pojawiało się w bazie, tym mniejsza pewność się pojawiała, mało tego czasami biblioteka potrafiła się nieźle pomylić.

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny



Poniżej interfejs dla uczenia się twarzy:

Wiktor Borowski
Paweł Furmaniak
Dawid Zborowski
VI Semestr Informatyki
Podstawy Teleinformatyki
Politechnika Poznańska
Wydział Elektryczny



Przycisk Seria służy do zrobienia serii pięciu zdjęć, zapisania ich w lokalizacji startowej aplikacji, oraz skojarzeniu ich z imieniem wpisanym z pola tekstowego w xmlowym pliku Faces.

8.0 Uwagi.

Do zrealizowania projektu posłużyliśmy się dokumentacją dostępną na stronie twórcy biblioteki. W początkowych fazach projektu (wykrycie twarzy) to wystarczyło, jednak później nasz projekt staną w miejscu. Mieliśmy problemy z implementacją modułu rozpoznającego twarz, a konkretniej to, w jaki sposób zapisać lub odczytać dane z lub do XMLa, tak aby były zgodne z EmguCV. Pomocne okazały się rozwiązania znalezione na git hubie.