Rozpoznawanie człowieka metodami biometrii
Projekt 3. — Rozpoznawanie na podstawie twarzy
Raport

Bartłomiej Dach

11 maja 2019

Poniższy dokument stanowi sprawozdanie z implementacji aplikacji dokonującej rozpoznwania człowieka na podstawie zdjęć twarzy w pozycji frontalnej. W dokumencie opisano porównywane metody oraz zawarto wyniki eksperymentalne dla obu metod na dostarczonym zbiorze zdjęć.

1 Wstęp

Twarz to cecha biometryczna zaliczająca się do cech biologicznych. Większość mierzalnych cech twarzy (geometria, kolor skóry) jest opartych na cechach anatomicznych lub etnicznych. Z drugiej strony aspekty takie, jak zarost czy wiek utrudniają wykorzystanie tych cech, ponieważ zmieniają się z czasem.

Rozpoznawanie na podstawie twarzy jest częścią coraz większej liczby urządzeń konsumenckich, szczególnie w branży urządzeń mobilnych. Ponieważ akwizycja obrazu twarzy nie wymaga bezpośredniego kontaktu, a coraz więcej smartfonów posiada kamery po przedniej stronie, jest to wygodna metoda potwierdzenia tożsamości użytkownika telefonu.

W ramach projektu zaimplementowano dwie metody oparte na identyfikacji punktów charakterystycznych twarzy. W obu przypadkach do znajdowania twarzy na zdjęciach i lokalizacji punktów charakterystycznych wykorzystano bibliotekę `dlib`. Metody różnią się właściwym przetwarzaniem punktów — w jednej metodzie punkty charakterystyczne wykorzystywane są bezpośrednio po pewnych korekcyjnych przekształceniach geometrycznych, zaś w drugiej w celu porównania z rozwiązaniami *state-of-the-art* wykorzystano wytrenowany klasyfikator ResNet [2, 5]. Dokładniejsze informacje znajdują się w podrozdziale 3.3.

2 Opis aplikacji

Opracowana aplikacja została zaimplmentowana w języku skryptowym Python w wersji 3.5.2. Wybór umotywowany był minimalizacją czasu implementacji poprzez wykorzystanie istniejących bibliotek *open source*. Aplikacja ma postać zbioru skryptów konsolowych — zdecydowano się na rezygnację z interfejsu graficznego ze względu na jego niewielką przydatność w stosunku do czasu implementacji. Sposób wywołania skryptów opisano w podrozdziale 2.2.

2.1 Zastosowane biblioteki

W implementacji zastosowano zbiór bibliotek *open source* ułatwiających implementację systemu. Pełna lista zastosowanych bibliotek znajduje się w tabeli 1.

Nr	Nazwa	Opis	Licencja	
1	`dlib` 19.17.0	Biblioteka wspomagająca proces rozpoznawania twarzy	Boost	[4]
2	`matplotlib` 3.0.3	Tworzenie wykresów i wizualizacji	PSF	[3]
3	`pandas` 0.24.2	Struktury do manipulacji i analizy danych	BSD	[6]
4	`seaborn` 0.9.0	Rozszerzone wizualizacje danych	BSD	[9]
5	`tqdm` 4.31.1	Biblioteka wspomagająca do pasków postępu w skryptach	MPL	[1]

Tablica 1: Lista bibliotek użytych w projekcie

2.2 Instrukcja obsługi

W celu uruchomienia skryptów do rozpoznawania konieczne jest zainstalowanie interpretera Python oraz bibliotek zawartych w tabeli 1. Aby zainstalować wymagane biblioteki, należy wywołać polecenie

```
$ pip3 install -r requirements.txt
```

gdzie plik `requirements.txt` to plik dołączony do źródeł aplikacji

2.2.1 Skrypt `recognize.py`

Skrypt `recognize.py` pozwala na sprawdzenie, czy twarze na podanym zdjęciu należą do osób z podanego zbioru treningowego. Wykonywanie skryptu powinno być zgłodne ze składnią:

```
recognize.py [-h] (-d TRAINING_SET_DIR | -f EMBEDDING_FILE)
             [-l LANDMARK_MODEL] (-n | -r RECOGNITION_MODEL)
             [-t THRESHOLD] [-o OUTPUT_EMBEDDINGS]
             TEST_IMAGE
```

gdzie poszczególne argumenty to:

- `-h` — wyświetla pomoc,
- `-d TRAINING_SET_DIR` — oznacza, że wektory cech dla zdjęć ze zbioru testowego powinny być obliczone na podstawie zdjęć z podanego folderu `TRAINING_SET_DIR`,
- `-f EMBEDDING_FILE` — oznacza, że wektory cech dla zdjęć ze zbioru testowego powinny być wczytane z istniejącego pliku `EMBEDDING_FILE`,
- `-l LANDMARK_MODEL` — zawiera ścieżkę do pliku zawierającego model używany do lokalizacji punktów charakterystycznych twarzy. W przypadku braku tego parametru domyślnie zachodzi próba użycia pliku o nazwie

  `shape_predictor_68_face_landmarks.dat`

  z katalogu roboczego.
- `-n` — oznacza, że do rozpoznawania ma być zastosowany własny algorytm normalizacji punktów charakterystycznych,
- `-r RECOGNITION_MODEL` — oznacza, że do rozpoznawania ma być zastosowany wytrenowany model ResNet załadowany z pliku `RECOGNITION_MODEL`,
- `-t` — oznacza próg liczbowy, którego przekroczenie powoduje odrzucenie zdjęcia ze zbioru testowego przy klasyfikacji twarzy,
- `-o OUTPUT_EMBEDDINGS` — pozwala na zapisanie wektorów cech po wyznaczeniu punktów charakterystycznych do pliku,
- `TEST_IMAGE` — ścieżka do zdjęcia z twarzami do identyfikacji.

2.2.2 Skrypt `confusion_verification.py`

Skrypt `confusion_verification.py` dokonuje testu wybranego klasyfikatora dla zadania weryfikacji tożsamości. Składnia wywołania powinna być zgodna z następującym wzorcem:

```
confusion_verification.py [-h]
                          (-d TRAINING_SET_DIR | -f EMBEDDING_FILE)
                          [-l LANDMARK_MODEL]
                          (-n | -r RECOGNITION_MODEL)
                          [-o OUT_FILE_PREFIX]
```

Wszystkie parametry poza `-o` mają to samo znaczenie, co w przypadku skryptu `recognize.py`. Flaga `-o` oznacza w tym przypadku prefiks dla plików tworzonych przez skrypt.

Wynikiem skryptu są dwa pliki:

1. Pierwszy plik o sufiksie `distance_matrix.csv` zawiera symetryczną macierz odległości Euklidesa między wektorami cech wyznaczonymi przez klasyfikator na podstawie punktów charakterystycznych.

2. Drugi plik o sufiksie `error_rates.csv` zawiera stosunki błędów w zadaniu weryfikacji w zależności od przyjętego progu. Dla każdej pary obrazów klasyfikator uznaje dwa obrazy za obrazy tej samej osoby, jeśli odległość Euklidesa między ich wektorami cech jest mniejsza niż przyjęty próg. Na tej podstawie obliczane są wskaźniki FAR i FRR.

2.2.3 Skrypt `confusion_identification.py`

Skrypt `confusion_identification.py` dokonuje testu wybranego klasyfikatora dla zadania identyfikacji osoby na zdjęciu. Składnia wywołania powinna być zgodna z następującym wzorcem:

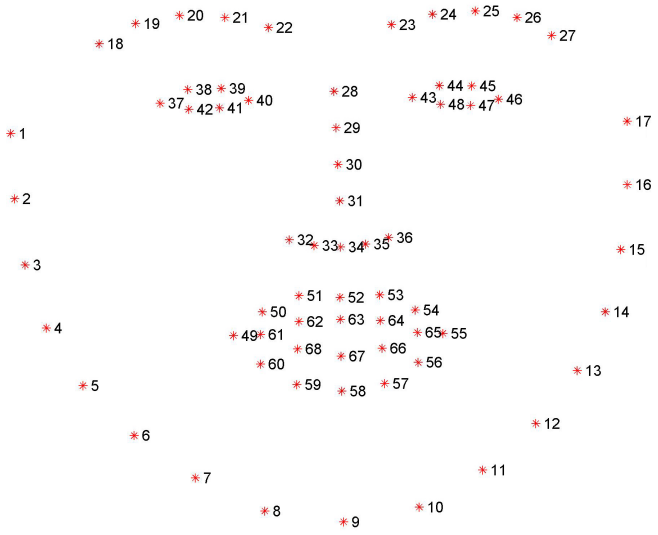
```
confusion_identification.py [-h]
                            (-d TRAINING_SET_DIR | -f EMBEDDING_FILE)
                            [-l LANDMARK_MODEL]
                            (-n | -r RECOGNITION_MODEL)
                            [-o OUT_FILE_PREFIX]
```

gdzie parametry są interpretowane tak samo, jak w skrypcie `confusion_identification.py`. Wynikiem skryptu jest pojedynczy plik o sufiksie `confusion_matrix.csv`, zawierający macierz pomyłek obliczoną dla klasyfikatora. Wiersze macierzy oznaczają etykietę osoby na zdjęciu (wzorcową), zaś kolumny — etykietę zwróconą przez klasyfikator.

3 Opis metody

3.1 Rozpoznawanie twarzy na zdjęciu

3.2 Lokalizacja punktów charakterystycznych



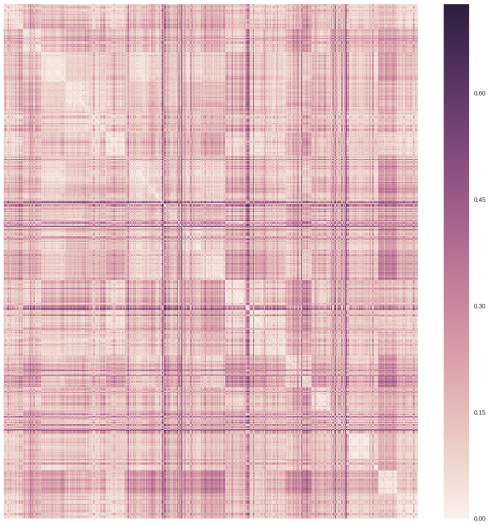
Rysunek 1: Wizualizacja 68 punktów charakterystycznych wykorzystywanych do rozpoznawania twarzy, wytypowanych przez *Intelligent Behaviour Understanding Group*. Źródło: [8]

3.3 Klasyfikacja twarzy na podstawie punktów charakterystycznych

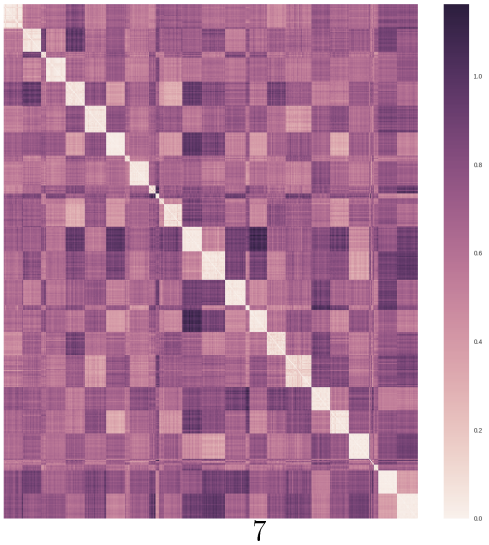
3.3.1 Podejście normalizacyjne

3.3.2 Wykorzystanie klasyfikatora ResNet

4 Wyniki eksperymentalne

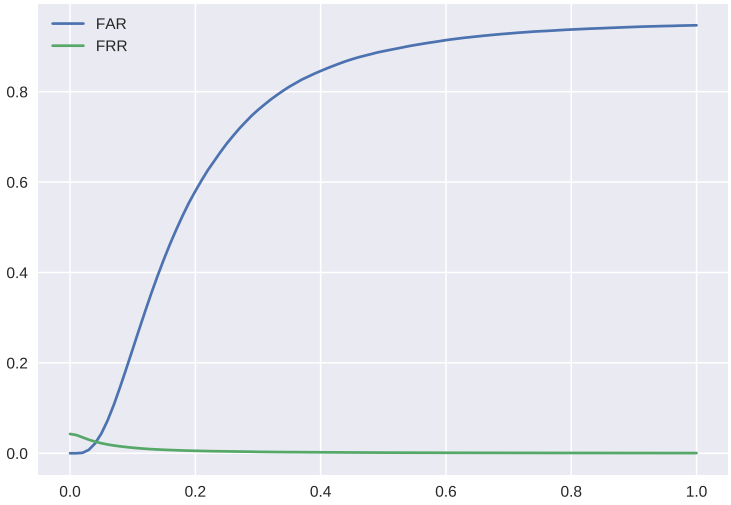


(a) Macierz dla podejścia normalizacyjnego po przetworzeniu odległości wg wzoru $d' = 1 - \frac{1}{d+1}$.

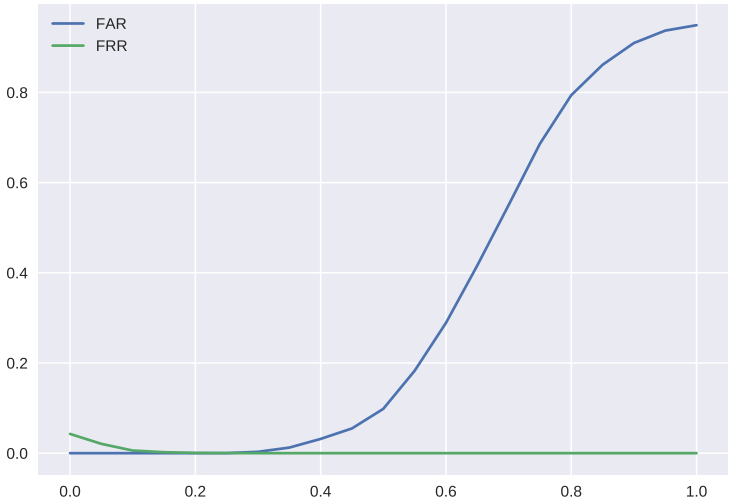


(b) Macierz dla klasyfikatora ResNet.

Rysunek 2: Macierze wzajemnych odległości dla poszczególnych par obrazów ze zbioru testowego. Zauważalne są ciemniejsze bloki wzdłuż przekątnej, obrazujące podobieństwo wielu obrazów przedstawiających jedną osobę.

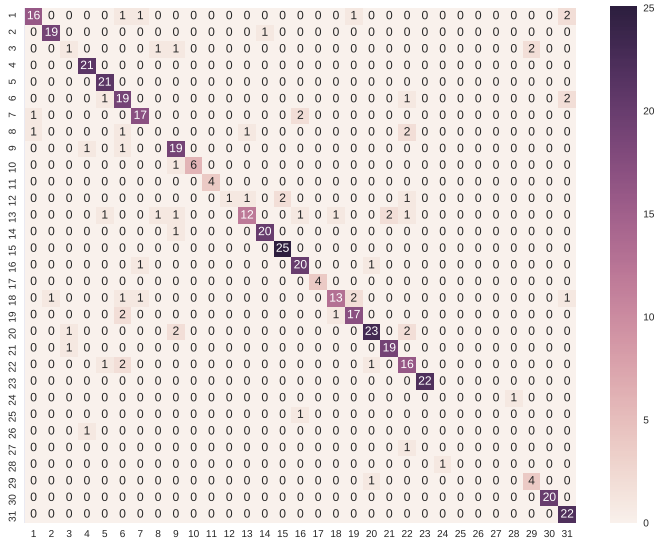


(a) Wskaźniki FAR i FRR dla podejścia normalizacyjnego.

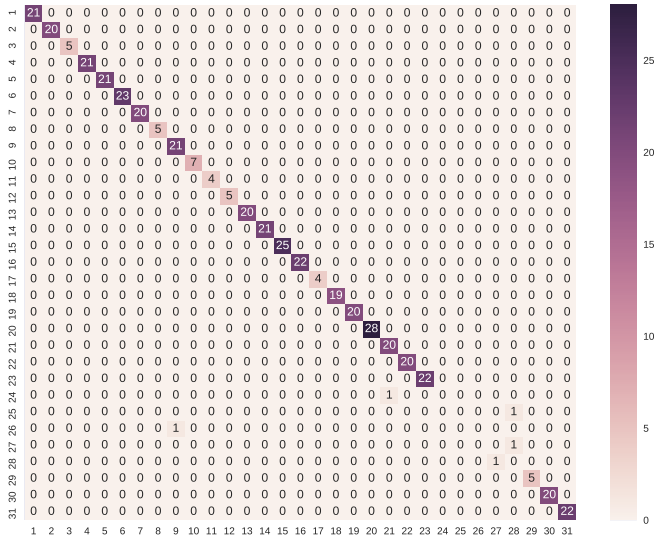


(b) Wskaźniki FAR i FRR dla klasyfikatora ResNet.

Rysunek 3: Wartości wskaźników FAR i FRR w zależności od przyjętego progu klasyfikacji w zadaniu weryfikacji tożsamości na podstawie par zdjęć.



(a) Macierz dla podejścia normalizacyjnego.



(b) Macierz dla klasyfikatora ResNet.

Rysunek 4: Macierze pomyłek dla obu podejść. Wiersze odpowiadają etykietom (identyfikatorom osób) oryginalnych zdjęć, kolumny odpowiadają etykietym przypisanym zdjęciom przez klasyfikator.

4.1 Podejście normalizacyjne

4.2 Klasyfikator ResNet

5 Podsumowanie

Literatura

[1] da Costa-Luis, C. i inni, „tqdm". [Online]
Dostępne: `https://tqdm.github.io/`. [Dostęp 11 maja 2019]

[2] He, K., Zhang, X., Ren, S., Sun, J., „Deep Residual Learning for Image Recognition", `arXiv:1512.03385`, 2015. [Online]
Dostępne: `https://arxiv.org/abs/1512.03385`. [Dostęp 11 maja 2019]

[3] „Matplotlib: A 2D graphics environment", *Computing In Science & Engineering*, tom 9, nr 3, s. 90–95, 2007.

[4] King, D. i inni, „dlib C++ Library". [Online]
Dostępne: `http://dlib.net/`. [Dostęp 11 maja 2019]

[5] King, D., „dlib-models". [Online]
Dostępne: `https://github.com/davisking/dlib-models`. [Dostęp 11 maja 2019]

[6] McKinney, W., „Data Structures for Statistical Computing in Python", *Proceedings of the 9th Python in Science Conference*, s. 51–56, 2010.

[7] Oliphant, T.E., *A Guide to NumPy*, Trelgol Publishing, Stany Zjednoczone, 2006.

[8] Sagonas, C., Zafeiriou, S., „Facial point annotations". [Online]
Dostępne: `https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/`. [Dostęp 11 maja 2019]

[9] Waskom, M. i inni, „`seaborn`: statistical data visualization". [Online]
Dostępne: `https://seaborn.pydata.org/`. [Dostęp 11 maja 2019]