Wstęp do multimediów - Laboratorium 7
Analiza obrazu – detekcja twarzy

Celem laboratorium jest zapoznanie studentów z wybranymi algorytmami umożliwiającymi wykrywanie twarzy w obrazach. W zadaniach zostaną wykorzystane algorytmy zaimplementowane w bibliotekach OpenCV [1], Dlib [2] oraz InsightFace [3].

W ramach przygotowania do zajęć laboratoryjnych należy zapoznać się z przykładowym skryptem, w którym zademonstrowane zostały operacje wykrywania twarzy z wykorzystaniem różnych detektorów.

Wprowadzenie

Jednym z pierwszych algorytmów wykrywania twarzy w obrazach był algorytm zaproponowany przez P. Viola i M. Jones [4]. Algorytm ten, mimo iż obecnie znanych jest wiele znacznie bardziej efektywnych rozwiązań, nadal jest wykorzystywany, gdyż umożliwia detekcję twarzy w czasie rzeczywistym przy niewielkich nakładach obliczeniowych, co jest istotne np. w przypadku implementacji w systemach wbudowanych. Działanie algorytmu polega na przeszukiwaniu obrazu i wykrywaniu regionów o określonych cechach reprezentowanych falkami Haara. Cechy te zawierają informacje o zmianie wartości kontrastu pomiędzy prostokątnymi grupami pikseli. Wykryte cechy są następnie analizowane przez kaskadę klasyfikatorów, które wybierają regiony zawierające cechy zgodne z modelem twarzy wyuczonym na zbiorze obrazów trenujących.

Wśród klasycznych (tzn. nie używających sieci neuronowych) algorytmów detekcji twarzy za jeden z najbardziej efektywnych uważany jest algorytm, w którym jako wektor cech używany jest histogram zorientowanych gradientów (*Histogram of Oriented Gradients*, HOG) [5], a jako klasyfikator maszyna wektorów nośnych (*Support Vector Machine*, SVM) [6]. Wektor cech (HOG) wyznaczany jest na podstawie gradientów wyznaczonych w kierunku poziomym i pionowym dla rozłącznych bloków składowej luminancji obrazu. Klasyfikator (SVM) jest trenowany na reprezentatywnym zbiorze wektorów cech zawierającym zarówno przykłady pozytywne (tzn. zawierające twarze) jak i negatywne.

Współczesne algorytmy detekcji twarzy wykorzystują splotowe sieci neuronowe (*Convlutional Neural Networks*, CNN). W algorytmach tych stosowane są różnorodne architektury sieci neuronowych, do trenowania których wykorzystywane są bazy zawierające tysiące obrazów. W ćwiczeniu zostanie wykorzystana sieć Max-Margin Object Detection (MMOD) [8], której implementacja dostępna jest w bibliotece Dlib oraz sieć neuronowa wykorzystywana w bibliotece InsightFace.

Środowisko do wykonania zadań, przykładowy skrypt

Do realizacji zadań wykorzystywany jest język Python wraz z dodatkowymi pakietami i bibliotekami: OpenCV, Dlib oraz InsightFace.

Zalecaną formą wykonania zadań i przygotowania sprawozdania są notatniki *Jupyter Notebook* (pliki z rozszerzeniem `.ipynb`). Do uruchamiania notatników można wykorzystać aplikację Jupyter Lab (wymagane zainstalowanie pakietu `jupeterlab` w środowisku Pythona).

Na komputerach dostępnych w laboratorium utworzone jest środowisko wirtualne, w którym zainstalowane są już wszystkie niezbędne pakiety oraz Jupyter Lab. Środowisko wirtualne należy aktywować z konsoli (linii komand) poleceniem (uwaga: kropka na początku polecenia jest istotna) :

```
. /opt/python/pmut-venv/bin/activate
```

a następnie można uruchomić Jupyter Lab (również z konsoli, najlepiej w tym samym katalogu, w którym znajdują się notatniki do uruchomienia, albo w jednym z katalogów nadrzędnych):

```
jupyter lab
```

W rezultacie powinno wyświetlić się okno przeglądarki internetowej ze stroną główną Jupyter Lab; w przypadku gdy przeglądarka nie uruchomi się automatycznie, należy skorzystać z linku wypisanego w konsoli podczas uruchamiania serwera.

Zadania można realizować również na własnych komputerach – odpowiednie środowisko należy przygotować we własnym zakresie, można do tego wykorzystać udostępniony plik ‘requirements.txt’ do instalacji wymaganych pakietów. Wygodne może być pobranie i zainstalowanie dystrybucji Pythona zawierającej już najpopularniejsze pakiety, np. Anaconda, która w podstawowej wersji dostępna jest za darmo (https://www.anaconda.com/download).

Notatniki można też wykonywać na platformie Google Colab, dostępnej z poziomu przeglądarki internetowej (https://colab.research.google.com/). W środowisku Colab zainstalowanych jest już wiele popularnych pakietów języka Python, w tym te wykorzystywane do zadań (oprócz InsightFace). Korzystanie z tego środowiska wymaga posiadania konta Google – w przypadku wybrania tego środowiska, należy we własnym zakresie zadbać o możliwość korzystania z niego.

Kod źródłowy do realizacji zadań (w formacie notatnika .ipynb oraz skryptu .py) wraz z potrzebnymi modelami i przykładowymi obrazami testowymi udostępnione są na platformie Teams. W założeniu, przykładowy obraz oraz modele dla detektorów zamieszczone są w tym samym katalogu co notatnik/skrypt – w przypadku innej ich lokalizacji konieczna może być modyfikacja kodu programu, aby podać właściwą ścieżkę do plików. W udostępnionym skrypcie przedstawione są: wczytywanie obrazu, inicjalizacja detektora i wykrywania twarzy dla poszczególnych detektorów.

Udostępnione jest również archiwum z kilkudziesięcioma innymi obrazami testowymi z bazy WIDER [9]. Do realizacji zadań można również wykorzystywać własne obrazy.

Sprawozdanie:

Sprawozdanie powinno być pojedynczym plikiem, którego nazwa powinna zawierać imię i nazwisko wykonawcy (oprócz innych ewentualnych elementów).

Sprawozdanie powinno zawierać: **kod źródłowy**, **uzyskane wyniki** (w tym **obrazy z zaznaczonymi wynikami detekcji** oraz **tabele z wartościami miar efektywności**) oraz **komentarze i obserwacje**.

Preferowaną formą sprawozdania jest notatnik .ipynb z widocznymi wynikami działania (pola z danymi wyjściowymi powinny być ‘rozwinięte’) oraz dodatkowymi polami tekstowymi zawierającymi zebrane wyniki i komentarze. Sprawozdanie może być również dokumentem w formacie pdf/docx.

Zadania

1. Wykonaj detekcję czterema detektorami dla zadanego obrazu testowego.
   Zaproponuj miary efektywności algorytmów detekcji twarzy i wyznacz wartości tych miar dla obrazu testowego, przy czym zawsze podaj (oprócz własnych propozycji):
   - całkowitą liczbę twarzy w obrazie (*ground truth*, GT),
   - liczbę wszystkich detekcji (wyników zwróconych przez detektor),
   - liczbę poprawnie wykrytych twarzy (*true positives*, TP),
   - liczbę nie wykrytych twarzy (*false negatives*, FN),
   - liczbę obiektów niepoprawnie rozpoznanych jako twarze (*false positives*, FP).

   Zastanów się, jakie relacje istnieją pomiędzy powyższymi wartościami.
   Wyniki zbierz w tabeli ułatwiającej ich porównanie.
   Zastanów się, jak potraktować wyniki detekcji, które obejmują jedynie część twarzy, albo takie, które oprócz twarzy zawierają również spory obszar *tła* (nie-twarzy).
   Uwaga: zliczając twarze w obrazie uwzględnij również twarze częściowo przysłonięte, obrócone, rozmyte, widoczne z profilu i od tyłu, występujące w tle, na rysunkach, itp.
   Uwaga: wraz z przykładowymi obrazami testowymi z bazy WIDER udostępniony jest również ich opis (tzw. *ground truth*) – można go wykorzystać do sprawdzenia i porównania jakie *referencyjne* twarze zostały oznaczone na tych obrazach (w tym na danym obrazie testowym).
   Porównaj i skomentuj uzyskane wyniki.
2. Przygotuj dodatkowe obrazy testowe i wyznacz dla każdego z nich miary zaproponowane w pkt. 1 (również w każdym przypadku podając wymienione tam wielkości).
   Wyniki zbierz w tabelach ułatwiających porównanie skuteczności detektorów – dla każdego obrazu przygotuj tabelę jak w pkt. 1.
   Sprawdź działanie detektorów na co najmniej trzech innych obrazach testowych (przykładowe obrazy testowe dostępne są na platformie Teams, można wykorzystać również własne obrazy), w tym przynajmniej jednego „trudnego” (dużo twarzy, twarze różnych rozmiarów, mocno przysłonięte, rozmyte, obrócone, itp. – dla prostych przypadków bez problemu można uzyskać detekcję poprawną w 100%).
   Porównaj i skomentuj uzyskane wyniki.
3. Wyznacz miary zaproponowane w pkt. 1 (włącznie z tymi już tam wymienionymi) **łącznie dla wszystkich obrazów** testowanych w pkt. 1 i pkt. 2.

Wyniki zbierz w tabeli

Uwaga: nie chodzi o powtórzenie tabel z pkt.1 i pkt.2!

Porównaj i skomentuj uzyskane wyniki.

4. Detektory z biblioteki Dlib (hog_svm_detector i cnn1_detector w kodzie) mogą dokonać *k*-krotnego skalowania obrazu wejściowego (drugi parametr wywołania detektora). Również rozmiar obrazu dla detektora InsightFace można zmieniać (parametr det_size metody prepare).
   Sprawdź, jak na efektywność detekcji twarzy wpłynie zmiana rozmiaru analizowanego obrazu: dla detektorów z biblioteki Dlib zmień wartość parametru z 1 na 0 oraz na 2, dla detektora InsightFace dwukrotnie zmniejsz i zwiększ wartości w det_size.
   Jakie nowe twarze są znajdowane lub przestają być znajdowane? Wyznacz wartości miar zaproponowane w pkt. 1.
   Wyniki zbierz w tabeli.
   Porównaj i skomentuj uzyskane wyniki.

Bibliografia

1. https://opencv.org/
2. http://dlib.net/
3. https://insightface.ai/
4. Paul Viola , Michael Jones, “Rapid object detection using a boosted cascade of simple features”, *Conference on Computer Vision and Pattern Recognition* (2001)
5. https://learnopencv.com/histogram-of-oriented-gradients/
6. https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html
7. https://learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/
8. Davis E. King, Max-Margin Object Detection, *Computer Vision and Pattern Recognition*, 2015, http://arxiv.org/abs/1502.00046
9. http://shuoyang1213.me/WIDERFACE/