УДК 004.93.1

СИСТЕМА РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

Пронін С.В., Кньовець В.С.

Харківський національний автомобільно-дорожній університет, Харків

Застосування систем розпізнавання обличчя. Рівень, поширення систем розпізнавання обличчя залежить від технологічного розвитку країни. Але ця технологія поступово приходить в країни з меншим розвитком.

Ця технологія на даний час з найбільш затребуваних напрямків в сучасному світі IT. Кожна компанія яка є технологічною, використовує систему розпізнавання обличчя як ідентифікація робочого персоналу. Також цією системою користуються банки, аеропорти та державні установи такі як поліція, школи, університети та інші. Система може запобігти магазинним крадіжкам, терористичним актам або допомогти в пошуку злочинців.

Аеропорти застосовують систему розпізнавання для ідентифікації обличчя на міжнародних рейсах. Замість перевірки документів і посадкового талона Вас просять подивитися в камеру під час реєстрації на рейс під час вильоту чи прильоту.

З недавнього часу цю систему розпізнавання обличчя почали використовувати найбільші банки країни. Вони використовують систему для підтримки проведення платежів з Face ID. Таким чином в додатку приват-банку з’явився новий метод оплати FacePay.

Середовище розробки системи розпізнавання обличчя.

Найпопулярнішою програмою для програмування на даний час є продукт від Microsoft це Visual Studio. Visual Studio Community - повнофункціональна, розширюється і безкоштовна інтегроване середовище розробки для створення сучасних додатків Android, iOS і Windows, а також веб-додатків і хмарних служб. Вона підтримує популярні мови програмування, серед яких C, C++, VB.NET, C#, F#, JavaScript, Python.

Функціональність Visual Studio охоплює всі етапи розробки програмного

забезпечення, надаючи сучасні інструменти для написання коду, проектування графічних інтерфейсів, збірки, налагодження і тестування програм. Можливості Visual Studio можуть бути доповнені шляхом підключення необхідних розширень.

Мова та бібліотеки розробки системи розпізнавання обличчя. Мовою програмування системи можна вибрати Python. Python – динамічна інтерпретована об’єктно-орієнтована скриптова мова програмування із строгою динамічною типізацією. Він підходить для машинного навчання і обробки зображень. В даний момент Python має безліч чудових пакетів для обробки зображень. Основні пакети для обробки зображень перераховані нижче[1].

Бібліотеки машинного зору:

- NumPy.
- Matplotlib.
- IPython Jupyter.
- Pillow.
- OpenCV.
- SciPy.
- Scikit-learn.

Метод для знаходження і розпізнавання обличчя на фото та відео

Одним з методів обробки зображень є розпізнавання осіб. Виявлення осіб - це область обробки зображень, яка використовує машинне навчання для пошуку осіб на зображеннях. Цей метод називається Каскади Хаара.

Метод був запропонований Полом Віолою і Майклом Джонсом в їхній статті «Швидке виявлення об'єктів з використанням посиленого каскаду простих функцій» в 2001 році та використовує.

У методі Віоли-Джонса цю основу складають примітиви Хаара, що представляють собою розбивку заданої прямокутної області на набори різнотипних прямокутних під областей (рис.1).

Каскади Хаара (Haar Cascade) - це метод виявлення об'єктів, що

використовується для визначення місця розташування об'єктів на зображеннях. Алгоритм навчається на великій кількості позитивних і негативних зразків: позитивні зразки - це зображення, які містять об'єкт, що цікавить, а негативні зразки - це зображення, які містять що завгодно, крім шуканого об'єкта. Після навчання класифікатор може знайти об'єкт, що цікавить на нових зображеннях [2].

Спочатку алгоритму потрібно багато позитивних зображень (зображень осіб) і негативних зображень (зображень без осіб) для навчання класифікатора. Потім потрібно витягти з нього функції. Для цього використовуються особливості Хаара, показані на зображенні нижче. Вони схожі на наше сверточное ядро. Кожна функція є окремим значення, отримане шляхом віднімання суми пікселів під білим прямокутником з суми пікселів під чорним прямокутником.

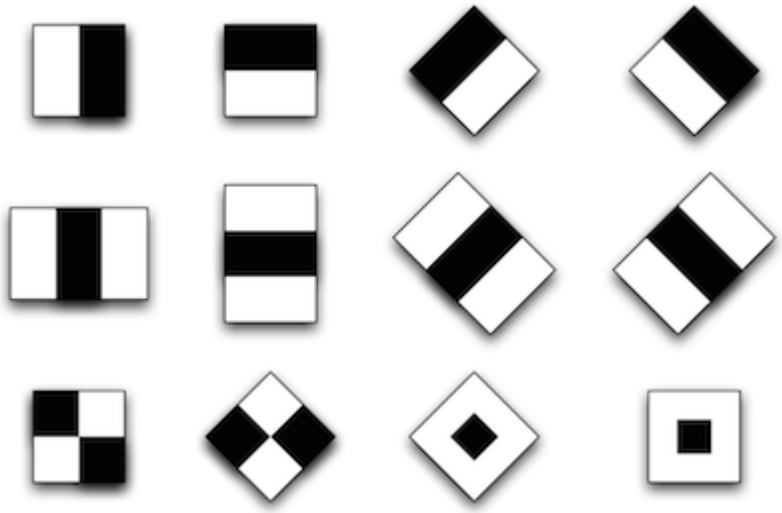

Рисунок 1 - Примітиви Хаара

Для цього застосовуємо кожну функцію до всіх навчальних зображень. Для кожної функції алгоритм знаходить кращий поріг, який класифікує особи на позитивні і негативні. Очевидно, будуть помилки або неправильні класифікації. Вибираємо риси з мінімальною частотою помилок, що означає, що саме вони найбільш точно класифікують зображення особи і зображення без обличчя.

Спочатку кожного зображення присвоюється однакову вагу. Після кожної класифікації ваги неправильно класифікованих зображень збільшуються. Потім виконується той же процес. Розраховуються нові коефіцієнти помилок. Також нові ваги. Процес триває до тих пір, поки не буде досягнута необхідна точність або частота помилок, чи не буде знайдено необхідну кількість функцій.

Останній класифікатор - це зважена сума цих слабких класифікаторів. Він називається слабким, тому що сам по собі не може класифікувати зображення, але разом з іншими утворює сильний класифікатор [3].

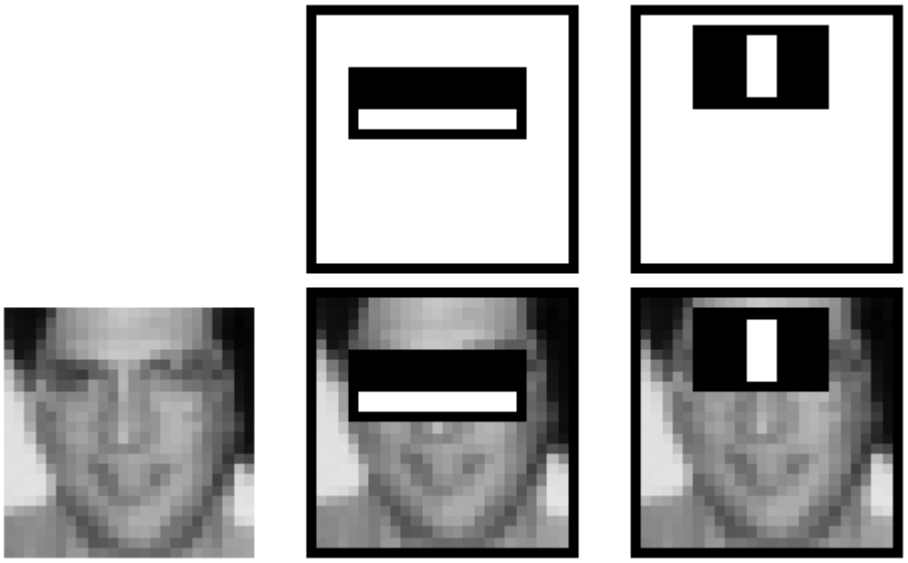

Рисунок 2 – Принцип накладення примітивів Хаара

На зображенні велика частина зображення - це не лицьова область. Тому краще мати простий спосіб перевірити, чи не є вікно областю особи. Якщо ні, то алгоритм зосередиться на регіонах, де може бути особа. Таким чином, витрачається більше часу на перевірку саме можливих областей особи. Для цього і була введена концепція каскаду класифікаторів. Замість того, щоб застосовувати всі функцій до вікна, вони згруповані по різним етапам класифікаторів і застосовуються один за іншим.

Результат роботи програма

Для створення програми була обрана бібліотека OpenCV.

OpenCV надає метод навчання або попередньо навчені моделі, які можна задати за допомогою методу cv :: CascadeClassifier [4]

На першому етапі подамо відео потік у вигляди послідовного потоку зображень:

```
cap = cv2.VideoCapture(0)
while True:
    success, img = cap.read()
    #img = cv2.imread("IMG_20191012_145410_3.jpg")
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

На наступному етапі подамо зображення до класифікатора з налаштованими параметрами:

```
faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0),2)
```

Результати роботи програми можемо побачити на рис. 3

Висновок. Система побудована на основі оптимальної бібліотеки комп’ютерного зору. Підтримка великої кількість операційних систем та хороша оптимізація, дозволяє запускати систему на багатьох комп’ютерах і засобах, що дає змогу набагато більшому колу користувачів користуватися розробкою.

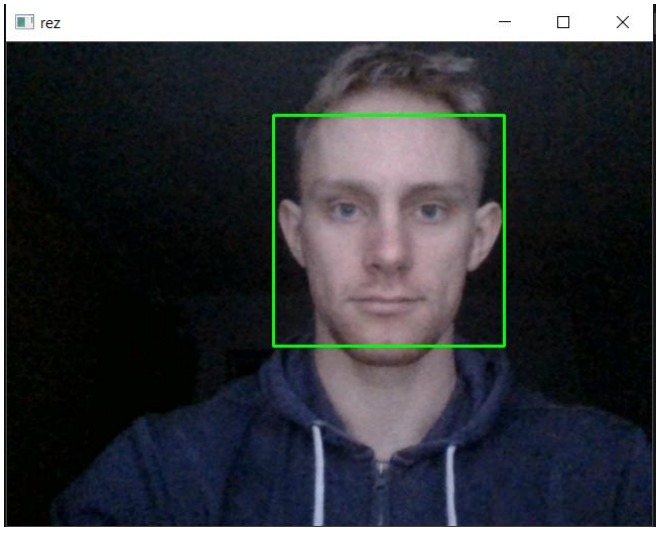

Рисунок 3 - Виявлення обличчя

Список використаних джерел

[1] Топ-7 библиотек для Python для компьютерного зрения [Он-лайн]. Доступно: https://arboook.com/kompyuternoe-zrenie/moj-top-7-bibliotek-dlya-python-dlya-kompyuternogo-zreniya/.

[2] P. Viola and M. Jones. Robust real-time face detection. IJCV 57(2), 2004.

[3] Lienhart R., Kuranov E., Pisarevsky V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In: PRS 2003, pp. 297-304 (2003).

[4] Cascade Classifier Training [Он-лайн]. Доступно: https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html.

УДК 004.93.1

ВИЯВЛЕННЯ РУХОМИХ ОБ’ЄКТІВ ЗА ДОПОМОГОЮ МЕТОДІВ КОМП’ЮТЕРНОГО ЗОРУ

Васильков Д. Р., Пронін С.В.

Харківський національний автомобільно-дорожній університет, Харків

Задачі комп’ютерного зору у транспортних системах

Сучасні засоби відеоспостереження дозволяють вирішувати завдання моніторингу транспортних потоків шляхом аналізу відеоданих, що надходять з камер відеоспостереження. Результатом аналізу є ідентифікація транспортного засобу на дорозі, збережені зображення і параметри транспортних засобів у внутрішній базі даних. При необхідності система може сигналізувати про різні події [1].

Завдання, які вирішуються:

- розпізнавання державного реєстраційного знака;
- вимірювання швидкості руху транспортного засобу;
- детектування ДТП і «затор»;
- детектування порушень ПДР;
- визначення та класифікація транспортних засобів;