Projektni zadatak

Face recognition

Članovi tima:
-Irma Rožajac
-Selma Lekić
-Mirza Muharemović
-Arnes Rogo

Sadržaj

1. Opis teme

Face recognition je sistem prepoznavanja lica, koji može identificirati osobu sa slike. Osnovni princip rada sistema zasniva se na poređenju odabranih karakteristika lica ulazne slike, sa licima u bazi podataka. Prvi zadatak sistema jeste detekcija svih lica na ulaznoj slici, a zatim njihovo poređenje u da li odgovaraju nekoj osobi koja postoji u bazi podataka. Data set projekata rađenog u okviru ovog predmeta sastoji se od slika tri osobe i to: Bakira Izetbegovića, Milorada Dodika i Dragana Čovića. Ujedno su to i klase koja svaka ima po 20 slika, odnosno naš data set sastoji se od 60 slika.

2. DataPreparation1

2.1. Anotacija

Prvo je potrebno identifikovati lice na ulaznoj slici, odnosno kreiranja anotacija. Ovaj proces je radi funkcija *anotacije.py* u kojoj je korištena OpenCV funkcija CascadeCtulassifier, zajedno sa *haarcascade_frontalface_default.xml* file za detkovanje kontura lica. Lica su označena pravouganikom, a koordinate kao i visina i širina pravouganika su spašeni u datoteku *data.txt* u *data.json* u *json* formatu.

2.2. Maska

Kreiranje maski vrši se pomoću funkcije *maske.py*, koja koristi podatke iz *data.json* fila

3. DataPreparation2

3.1. Uklanjanje šuma

Za uklanjanje šuma korištena je funkcija *averagingFilter* koja je implementirana u fajlu *averagingFilter.py*. Ova funkcija ne daje najbolje rezultate, bolje rezultate bi davala bilateralna funkcija uklanjanja šuma.

3.2. Maskiranje neoštrina

Za maskiranje neoštrina iskorištena je matrica *[-1,-1,-1], [-1,9,-1], [-1,-1,-1]*, njena implementacija se nalazi u *izostrenaSlika.py* fajlu

4. DataPreparation3

4.1. Poboljšanje kontrasta

Za poboljšanje kontrasta je na početku urađena konverzija RGB u HLS. Dalje, vrši se manipulacija nad pikselima. U slučaju da je vrijednost piksela manja od srednje vrijednosti *cs2.mean(slika)*, radi se posvjetljenje svijetlih piksela za iznos faktora koji se može prilagođavati. Ukoliko je vrijednost piksela veća od srednje vrijednosti *cs2.mean(slika)*, onda se radi operacija u kojoj tamni pikseli postaju tamniji za iznos faktora koji se može prilagođavati. Faktor koji je korišten je *10*. Implementacija se može naći u fajlu *poboljsanjeKontrasta.py*.

4.2. Poboljšanje osvjetljenja

Kao i kod poboljšanja kontrasta prvo je potrebno izvšiti konverziju RGB u HLS. Manipulacija nad pikselima vrši se sabiranjem iznosa pojedinačnih piksela sa faktorom. Faktor koji je uzet je *15*, a implementacija se može naći u *poboljšanjeSvjetlosti.py* fajlu.

4.3. Ujednačavanje histograma

Ujednacavanje histograma se obavlja u *histogram.py*, i to metodom *CLAHE*. Sliku je prvo potrebno pretvoriti u *greyscale*. Vrijednost faktora je *3.0*.

5. DataPreparation4

Izvšena je podjela i to u skupove: *Train* i *Test.* Iz svake klase je prvih 9 slika smjesteno u skup *Test,* dok su ostale slike smjestene u skup *Train,* ovu podjelu vrši funkcija *podjelaTrainTest.py*

6. Izbor modela za prepoznavanje

Model za prepoznavanje koji je korišten u ovom projektu je OpenCV algoritam LBPH (Local Binary Pattern Histogram). Kod LBPH algoritma se sumira lokalna struktura slike poređenjem svakog piksela sa njegovim susjednim. LBPH algoritam je izabran jer je jako pogodan za preciznu obradu podataka u realnom vremenu. Njegova računska složenost je manja i efikasnija u odnosu na druge algoritme zaprepoznavanje lica.

7. Izbor deskriptora

Deskriptor koji je pogodan za prepoznavanje lica je *Haar-like feature*. Ovaj deskriptor razmatra susjedne pravougaone regije na određenoj lokaciji u prozoru za detekciju, sumira intenzitete piksela u svakom regionu i izračunava razliku između tih suma. OpenCV koristi kaskade za pronalazak lica na slici. Same kaskade su samo XML datoteke koje sadrže OpenCV podatke koji se koriste za otkrivanje objekata. Ovaj deskriptor se može koristiti i za pronalazak drugih objekata, a to ustvari zavisi od vrste XML datoteke koja se koristi *haarcascade_frontalface_default.xml* za lica. Deskriptor radi sa *grey scale* slikama, jer za pronalazk lica nije bitna boja tj. za ekstraktovanje osobina lica. Sve slike se prvo konvertuju u *greyscale*. Funkcija koja ovo obavlja je *anotacije.py* jer se koristila i za izdvajanje regiona od interesa. Slike se nakon sto se lice pronađe spase u folder *anotacije*.

8. Kreiranje i treniranje modela

Model se kreira komandom: *face_recognizer = cv2.face.LBPHFaceRecognizer_create()*. Prvo se vrši čitanje slika za svaku klasu iz foldera Train, te se poziva funkcija faceDetector koja detektuje lice i vraća greyscale sliku i koordinate gdje se nalazi lice na slici. Zatim započinje treniranje tako sto se izvrši komanda: *face_recognizer.train(faces, np.array(names).)* Gdje parametar *faces* predstavlja niz maski za svaku sliku iz foldera *Train* tj. dijelovi slika samo na kojima se nalaze lica, dok *names* predstavlja odgovarajuci niz oznaka klasa koja su 0, 1, 2. Osnovna ideja LBPH je da sumira lokalnu strukturu slike tako što će uporediti svaki piksel sa njegovim susjednim. Glavna ideja je da se LBP slika podijeli u lokalne regije i da se iz svakog ekstraktuje histogram. Histogrami se nazivaju lokalni binarni obrasci.

Koraci pri prepoznavanju:
1. Predstaviti novu sliku prepoznavatelju
2. Kreira histogram za novu sliku
3. Novi histogram se poredi sa histogramom koji već ima
4. Detektira najbolje podudaranje i vraća ime klase sa najboljim podudaranjem

Svi rezultati treniranja se spase u fajl *train.yml*.