**Arcfelismerés Alapok**

*Skultéty Áron 2024.06.08*

**Miért szükséges a szürkeárnyalatos konverzió?**

A Haar Cascade kaszkád osztályozó szürkeárnyalatos képeket igényel az arcfelismeréshez. Ennek oka a következő:

* A szürkeárnyalatos képek kevesebb információt tartalmaznak, mint a színes képek, így a feldolgozásuk gyorsabb.
* A szürkeárnyalatos képek kevésbé érzékenyek a fényviszonyokra és a zajra, így az arcfelismerés pontosabb.

**Hogyan működik az arcfelismerés az OpenCV-ben a detectMultiScale függvénnyel?**

A detectMultiScale az arc kaszkád (Haar Cascade) osztályozó kulcsfontosságú függvénye. Lényegében a következőket végzi:

1. **Szürkeárnyalatos kép bemenete:** A függvény a szürkeárnyalatos képet használja bemenetként, melyet az előző lépés során alakítottunk át.
2. **Méretezi a képet:** A kaszkád különböző méretű keresőablakokat használ, hogy különböző méretű arcokat is megtaláljon. A detectMultiScale függvény a képet többször méretezi le, így képes kisebb és nagyobb arcok megtalálására is.
3. **A keresőablak mozgatása:** A keresőablak a szürkeárnyalatos képen lépésről lépésre végigmegy. Minden egyes pozíciónál a Haar-jellemzők kiszámításra kerülnek.
4. **Haar-jellemzők kiértékelése:** Minden egyes régióban kiszámításra kerülnek a Haar-jellemzők, ezek a téglalap alakú minták, amelyek sötét és világos pixelek kontrasztján alapulnak. A kaszkád ezeket a jellemzőket vizsgálja, hogy megtalálja az arcnak megfelelő részeket (pl. szem, orr, száj).
5. **A kaszkád szintjei:** A kaszkád egy többlépcsős rendszer. Minden egyes szint egy egyre összetettebb osztályozót tartalmaz. Ha egy régió eléri az egyik szintet, akkor átadásra kerül a következő, ennél is nehezebb osztályozó szintnek.
6. **Arc észlelése:** Ha a régió átjut az összes kaszkádszinten, akkor arcként kerül azonosításra. A detectMultiScale függvény egy listát ad vissza, melyben koordináták találhatók (x, y, w, h), melyek az arcok körüli téglalapot írják le.

**Paraméterek:**

A detectMultiScale függvénynek néhány fontos paramétere van:

* **scaleFactor:** Ez a paraméter meghatározza, hogy mennyire szeretnénk méretezni a képet az egyes kaszkádszintek között. Egy nagyobb scaleFactor gyorsabb felismerést, de kevésbé pontos eredményt jelent.
* **minNeighbors:** Ez a paraméter meghatározza, hányszor kell a kaszkádnak egy régiót arcként azonosítania, mielőtt ténylegesen arcként detektálja. Egy nagyobb minNeighbors szigorúbb kritériumokat állít fel, és segít csökkenteni a hamis pozitív eredményeket.

**Kódban:**

A scaleFactor 1.1 értékre van állítva, az azt jelenti, hogy minden szint között a kép 10%-kal lesz lekicsinyítve. A minNeighbors 5-re van állítva, ez azt jelenti, hogy egy régiót legalább ötször kell arcként azonosítani, mielőtt ténylegesen arcként kerül észlelésre.

ez itt a szürkeárnyalatos konverzió és az arcfelismerés


**Rajzolás:**

* **cv2.rectangle:** Ez a függvény rajzol egy téglalapot a képre. A paraméterei a következők:
  + **kep:** A bemeneti kép, amire rajzolni fogunk.
  + **(x, y):** A téglalap bal felső sarkának koordinátái.
  + **(x+w, y+h):** A téglalap jobb alsó sarkának koordinátái.
  + **(255, 0, 0):** A téglalap színe (kék).
  + **2:** A téglalap vastagsága.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

**II. verzió:  
  
Kép beolvasása a kameráról:**

* A ret, kep = cap.read() sor egy képet kap a kamerafolyamból. A ret egy logikai változó, amely azt jelzi, hogy sikeresen beolvastunk-e egy képet vagy sem a .read() művelettel. A kep változó tartalmazza magát a kép adatait.

Ablak bezárása:

* A cv2.waitKey(1) parancs 1 milliszekundumra vár billentyűleütésre. Ha ezen idő alatt nem történik billentyűleütés, a parancs -1 értéket ad vissza. Ha billentyűleütés történik, a parancs a lenyomott billentyű ASCII kódját adja vissza.
* A & 0xFF bitenkénti ÉS művelet a billentyűkód legfelső 8 bitjét 0-ra állítja. Ez biztosítja, hogy a billentyűkód ASCII érték legyen, még akkor is, ha a billentyűzet nem angol elrendezésű.
* Az ord('q') parancs a 'q' betű ASCII kódját adja vissza, ami 113.

A képen szöveg, Betűtípus, képernyőkép látható

Automatikusan generált leírás

**Kép beolvasása a kameráról:**

A cap.read() parancs beolvassa a következő képet a kamerafolyamból. A parancs két értéket ad vissza:

* ret: Egy logikai változó, amely jelzi, hogy a beolvasás sikeres volt-e. Ha True, akkor a beolvasás sikeres volt, és a kep változó tartalmazza a beolvasott képet. Ha False, akkor a beolvasás sikertelen volt.
* kep: A beolvasott kép adatait tartalmazó tömb.

A képen szöveg, Betűtípus, képernyőkép, tervezés látható

Automatikusan generált leírás

**Memóriafelszabadítás és bezárás:**

**1. cap.release():**

* Felszabadítja a kamerát a program által történő használat alól.
* Biztosítja, hogy más programok is hozzáférhessenek a kamerához.
* Memóriát szabadít fel, amit a kamera foglalt.

**2. cv2.destroyAllWindows():**

* Bezárja az összes OpenCV ablakot, amelyet a program nyitott.
* Felszabadítja a memóriát, amit az ablakok foglaltak.
* Megakadályozza, hogy a program ablakai nyitva maradjanak a háttérben, miután a program befejezte a futást.

**III. verzió**

Egy lényeges for ciklussal bővült a kód az arc adatainak kinyerése végett:  
  
A képen szöveg, képernyőkép, szoftver, Betűtípus látható

Automatikusan generált leírás

* A ciklus akkor és csak akkor fog lefutni hogyha a faces\_detect változó nem egy üres tömböt ad vissza, azaz észlelt egy arcot a kamerán.

**Arc kivágása:**

* A face\_extract = gray[y:y+h, x:x+w] kifejezésben a gray[y:y+h, x:x+w] rész egy részképet (szubszerkezetet) választ ki egy nagyobb képből.

**Arc átméretezése:**

* A resize függvény egy olyan képet hoz létre, amely az eredeti képből kivágott részképet alakítja át az arckep\_meret által meghatározott méretre. Ezt úgy teszi, hogy a képet **interpolációval** méretezi át, ami azt jelenti, hogy a program kiszámolja a kivágott részkép és az új méret közötti átmenetet, hogy az átméretezett kép minősége lehetőleg ne romoljon.

**Kép vektorizálása:**

* A flatten() metódus egy tömböt egydimenziós tömbbé alakít. Tehát ha az arckep változó egy kétdimenziós tömb (esetünkben egy kép), akkor a flatten() metódus kinyeri az összes elemet a képből, és egyetlen egydimenziós tömbben helyezi el őket.

**PCA (Principal Component Analysis):**

Az pca változó egy objektumot jelöl, amelyet a scikit-learn könyvtár PCA osztálya hoz létre. A PCA az adatok dimenziócsökkentésére és jellemzők kinyerésére szolgál. Ami itt történik:

* **pca = PCA(n\_components=1)**: Ez a sor létrehozza a PCA objektumot, és beállítja a főkomponensek számát. A főkomponensek olyan irányok a térben, amelyekre az adatokat vetítjük, hogy csökkentsük azok dimenzióját. A n\_components paraméter meghatározza, hogy hány főkomponenst tartunk meg az adatokban. Ebben az esetben csak egy főkomponenst tartunk meg, ezért a dimenziócsökkentés után az adatok csak egy dimenzióban lesznek reprezentálva.
*  **pca.fit(arckep\_vektor.reshape(-1, 1))**: Ez a sor illeszti a PCA modellt az adatokhoz. Az arckep\_vektor.reshape(-1, 1) rész egydimenziós tömböt készít az arckép pixeleiből. A reshape(-1, 1) a tömb átalakítását jelenti, hogy egy oszlopvektor legyen, amely csak egy dimenziót tartalmaz. A PCA modell illesztése során a modell megtanulja az adatokban lévő mintázatokat és sajátvektorokat keres az adatokban, amelyeket főkomponenseknek nevezünk.
*  **transzformalt\_arckep = pca.transform(arckep\_vektor.reshape(-1, 1))**: Ez a sor alkalmazza a PCA transzformációt az adatokra. A transzformáció során az eredeti adatokat a PCA által meghatározott sajátvektorokra vetítjük, és így az adatokat új koordinátarendszerben reprezentáljuk. Ebben az esetben a transzformalt\_arckep változóba mentjük az arckép PCA által transzformált változatát.

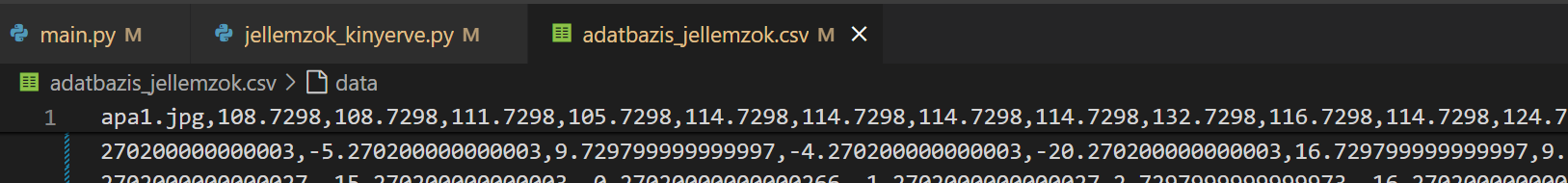
**Jellemzők kinyerése:**

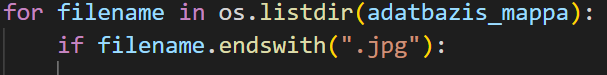
A jellemzők változó az arckép főkomponenseit tartalmazza az általunk meghatározott egydimenziós reprezentációban:

1. **jellemzők = transzformalt\_arckep.flatten()**: Ez a sor a transzformalt\_arckep változót, amely az arckép főkomponenseit tartalmazza a PCA transzformáció eredményeként, egydimenziós tömbbé alakítja (flatten() metódus). A transzformalt\_arckep változóban lévő főkomponensek általában egy többdimenziós NumPy tömb formájában vannak, így a flatten() metódus segítségével azokat egyetlen dimenziós tömbbé alakítjuk. Így könnyebben kezelhetővé válnak az adatok a további feldolgozás és analízis során.
2. **jellemzők változó**: Az eredményül kapott egydimenziós tömböt (transzformalt\_arckep főkomponenseinek a laposított változata) a jellemzők változóban tároljuk. Ez a változó tartalmazza az arckép főkomponenseinek jellemzőit az adott egydimenziós reprezentációban. **A vektor hossza 10 000 elem, mivelhogy az arckép egy 100x100 pixeles kép, és a jellemzővektor az egyes pixelek szürkeárnyalatos intenzitásértékeit tartalmazza.**

**IV. verzió**

**Új python file létrehozása amelyben a következők lettek végrehajtva:**

1. **adatbazis\_mappa:** Ez a sor hivatkozik a jelenlegi 30 arcképet (10-10-10) tartalmazó mappára
2. **adatbazis\_jellemzok\_lista = []:** létrehoztunk egy listát amelyben eltároljuk a 30 kép jellemzőit, amiket pedig később egy .csv fileba kiírunk majd.:  
   
3. a következő ciklus példányonként végigmegy az eltárolt képeken, majd lejjebb elvégzi ugyan azokat a jellemző kinyeréseket amelyeket a kamerában beolvasott képünkön végeztünk el:



1. Végül pedig with open parancscsal kiírjuk .csv file-ba az adatbazis\_jellemzok\_lista adatait sortörés nélkül (\n az alapértelmezett): A képen szöveg, képernyőkép, Betűtípus látható

   Automatikusan generált leírás

**V. verzió**

**Képmegjelenítés és euklideszi távolság számítása függvény:**

* A kepmegjelenites függvény célja, hogy megjelenítse egy adott fájlnévhez tartozó jellemzőket egy adatbázisban. Ha a fájlnév megtalálható az adatbázisban, akkor a jellemzők egy 100x100-as mátrix formájában kerülnek visszaadásra.

**A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás**

**Paraméterek**

**Bemeneti paraméterek**

* filename: A fájl neve (string), amelyhez tartozó jellemzőket keresünk az adatbázisban.
* jellemzok: Egy lista, amely a fájlhoz tartozó jellemzőket tartalmazza.

**Kimeneti értékek**

* adatbazis\_jellemzok: Egy numpy array (100x100), amely a fájlhoz tartozó jellemzőket tartalmazza, ha megtalálható az adatbázisban. Ha a fájlnév nem található az adatbázisban, az érték None.

**Működés**

1. **Adatbázis jellemzők keresése**:
   * A függvény végigiterál az adatbazis\_jellemzok\_lista nevű listán.
   * Ha az aktuális elem első eleme (elem[0]) megegyezik a megadott filename értékkel, akkor az elem többi részét numpy array-ként (np.array) átalakítja, és átméretezi egy 100x100-as mátrixra.
2. **Jellemzők visszaadása**:
   * Ha a fájlnév megtalálható az adatbázisban, a jellemzők mátrixként (adatbazis\_jellemzok) kerülnek visszaadásra.
   * Ha a fájlnév nem található az adatbázisban, az érték None.

**Az euklideszi\_tavolsag** **áttekintése**

**Euklideszi távolság előnyei**

Az euklideszi távolság egy egyszerű és hatékony módszer a különböző vektorok közötti hasonlóság mérésére. Ebben a projektben az euklideszi távolság használata több szempontból is előnyös:

1. **Egyszerűség**: Az euklideszi távolság kiszámítása matematikailag egyszerű és könnyen érthető. Ez megkönnyíti a fejlesztést és a hibakeresést.
2. **Hatékonyság**: Az euklideszi távolság kiszámítása gyors, ami különösen fontos, ha nagy mennyiségű adatot kell feldolgozni, például sok arcképet egy adatbázisban.
3. **Hasonlóság mérés**: Az arcfelismerési projektekben gyakran szükséges két kép hasonlóságának mérése. Az euklideszi távolság egy jól bevált módszer erre, mivel a vektorok közötti különbségeket pontosan méri.
4. **Robusztusság**: Az euklideszi távolság érzékeny a különbségekre, így kis eltéréseket is jól detektál, ami fontos lehet az arcfelismerésben, ahol apró részletek is számítanak.

**Euklideszi távolság alkalmazása az arcfelismerésben**

Ebben a projektben az arcképek jellemzőit vektorokként tároljuk. Az euklideszi távolság segítségével összehasonlíthatjuk ezeket a jellemző vektorokat, és meghatározhatjuk, mennyire hasonló két arc. Ez az alapja annak, hogy az arcfelismerő rendszer képes legyen azonosítani vagy megkülönböztetni az arcokat.

**Leírás**

* Az euklideszi\_tavolsag függvény két vektor közötti euklideszi távolságot számítja ki. Ha a második vektor bármely eleme string típusú, azt az elemet kihagyja a számításból.

A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

**Paraméterek**

**Bemeneti paraméterek**

* vektor1: Az első vektor (list), amelynek elemei számok.
* vektor2: A második vektor (list), amelynek elemei számok vagy stringek.

**Kimeneti értékek**

* tavolsag: Az euklideszi távolság a két vektor között (float).

**Működés**

1. **Távolság inicializálása**:
   * A tavolsag változó 0-ra inicializálása.
2. **Végigiterálás a vektor elemein**:
   * A függvény végigiterál a vektor1 összes elemén.
   * Ha a vektor2 megfelelő eleme string típusú, az iteráció folytatódik a következő elemmel.
   * Ha a vektor2 eleme nem string, a két elem közötti különbség négyzetét hozzáadja a tavolsag változóhoz.
3. **Euklideszi távolság kiszámítása**:
   * Az összegzett négyzetek gyökének kiszámítása és visszaadása.

**VI. verzió**

**Vizualizáció – kamerakép és adatbázisképek megjelenítése, valamint adatok kiíratása**

*A kamerakép (teszt2):* A képen Homlok, állkapocs, fekete, monokróm látható

Automatikusan generált leírás

Az adatbáziskép és a kiszámolt adatok:

(csak egy összehasonlításnál)

apa3.jpg

- Fekete arány: 19.56% | Fehér arány: 21.93% | Szürke arány: 58.51%

- Manhattan (sima): 637365.105 | Manhattan (előfeldolgozással): 784948.0A képen Emberi arc, portré, fekete-fehér, Homlok látható

Automatikusan generált leírás

Tesztek

A következő tesztekben a 3 személy 30 képét hasonlítottuk össze a fentebb szemléltetett módon. A képek egy egy alkalommal előfeldolgozás és előfeldolgozás nélkül kerültek kiszámításra. A konklúziók a következők:

### **Előfeldolgozással:**

### A képen szöveg, képernyőkép, képernyő, szám látható Automatikusan generált leírás**A képen szöveg, képernyőkép, Betűtípus látható Automatikusan generált leírás Színarányok és Manhattan-távolságok**

A színarányok és a Manhattan-távolságok közötti korrelációk általában alacsonyabbak:

* **A szürke arány és a Manhattan (sima) közötti pozitív korreláció (0.352417) azt sugallja, hogy a szürke szín mennyisége valamilyen mértékben összefügghet a távolsággal.**
* A fehér arány és a Manhattan (sima) közötti negatív korreláció (-0.383871) azt jelenti, hogy több fehér szín kevesebb távolsággal járhat.
* A fekete arány és a Manhattan-távolságok közötti korreláció alacsonyabb, jelezve, hogy a fekete színnek kisebb a hatása a távolságra.

**A mindenféle előfeldolgozás nélküli képadatok:**

**A képen szöveg, képernyőkép, diagram látható

Automatikusan generált leírás** A képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás  
A korrelációs mátrix alapján a színarányok közül a fekete és a fehér arányok mutatnak erősebb kapcsolatot a Manhattan-távolságokkal, míg a szürke arány gyenge kapcsolatot mutat.

* **A fekete arány általában pozitív kapcsolatot mutat a távolságokkal, ami azt sugallja, hogy több fekete szín nagyobb távolságot eredményez**.
* **A fehér arány közepes negatív kapcsolatot mutat, jelezve, hogy több fehér szín kisebb távolságot hozhat.**
* A szürke arány gyenge kapcsolatban van a Manhattan-távolságokkal, ami arra utal, hogy a szürke szín kevésbé befolyásolja a távolságokat.