**☆ Uparivanje obeležja ☆**

Polaznik: Sara Milosavljević

Mentor: Srđan Radović

**Uvod**

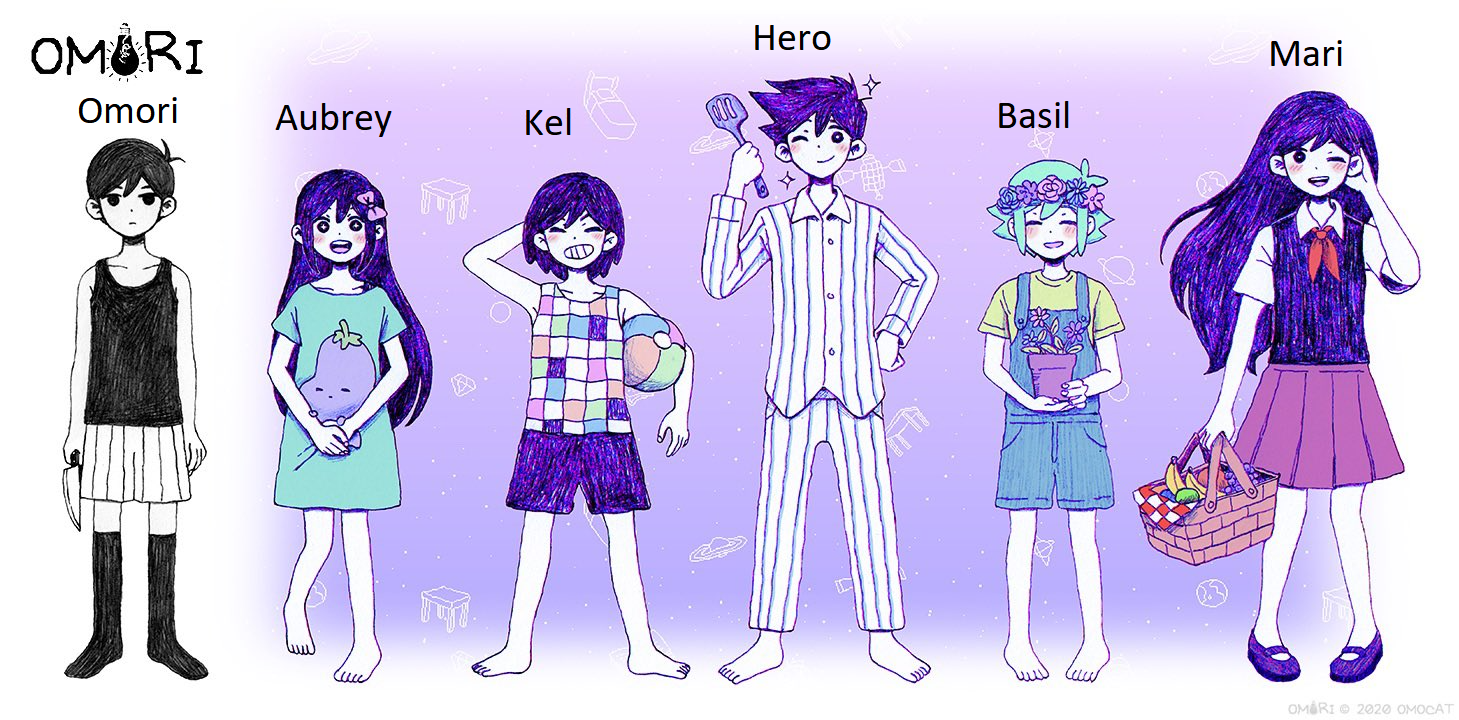
Uparivanje obeležja (engl. feature matching) je proces pronalaženja i povezivanja istih ili sličnih karakterističnih tačaka ili obeležja između dve ili više slika.

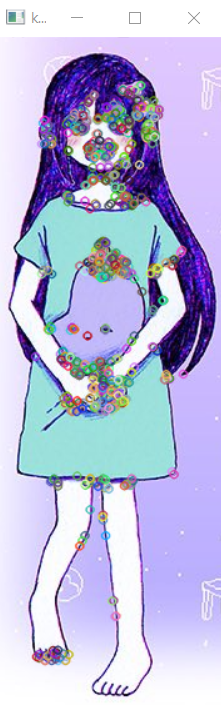
Ima puno primena, kao na primer:

* Sporazumevanje između slika: kako se slike odnose jedna prema drugoj, kao što su panorame i slike pre i posle nekog događaja
* Praćenje objekata u video snimcima
* Medicinska slikovna dijagnostika: za analizu medecinskih slika, poput rendgenskih snimaka
* Prepoznavanje objekata: u ovom izveštaju ću detaljnije predstaviti takvu primenu

U mom projektu se nalaze četiri koda:

* projekat date slike.py - kod koji povezuje slike zadate od saradnika
* projekat omori.py - kod koji povezuje sliku lika i sliku sličice lika iz igrice Omori. Na početku se velikim slovom upisuje ime lika i ime lika na sličici koje želimo da uporedimo. To služi da možemo da vidimo kako izgleda i kad program pokušava da upari dve slike koje nisu slične
* omori webcam character recognizer NEO 3000.py - veoma sofisticiran i revolucionaran program koji preko web kamere može da prepozna lika na sličici i da ispiše njegovo ime
* cv2 test.py - program koji povezuje dve slike koristeći gotovu OpenCV funkciju, testiranja radi.



**Kako radi?**

Učitamo dve slike. Koristeći ORB, na njima pronalazimo ključne tačke. Ključne tačke imaju deskriptore. To su vektori brojeva koji opisuju tu tačku. Za svaku sliku se čuvaju deskriptori njenih ključnih tačaka.

#ucitaj i nadji

Qimg = cv2.imread('dateSlike/pano2.jpg')

Timg = cv2.imread('dateSlike/pano3.jpg')

orb = cv2.ORB\_create(nfeatures=1000)

kpQimg, desQimg = orb.detectAndCompute(Qimg, None)

kpTimg, desTimg = orb.detectAndCompute(Timg, None)

Ti deskriptori se onda stavljaju u algoritam koji će da ih spoji po sličnosti. Svaka ključna tačka iz prve slike se upoređuje sa svakom ključnom tačkom iz druge slike. Zato se zove brute force match.

U svakom upoređivanju se računa distanca između tačaka, to jest koliko su različite. Za to se koristi Hamming distance. Hamming distance između dva niza iste dužine predstavlja broj pozicija na kojima se odgovarajući simboli razlikuju: distance = np.count\_nonzero(desc != desc2)

Sva upoređivanja za jednu tačku se onda stavljaju u niz i sortiraju se. Samo dve najbliže tačke se pamte.

def bfmatch(des1, des2):

matches = []

for i, desc in enumerate(des1):

matches2=[]

distances = []

for j, desc2 in enumerate(des2):

distance = np.count\_nonzero(desc != desc2)

distances.append(distance)

sortiraniidx = np.argsort(distances)

for j in range(2):

matches2.append(cv2.DMatch(i, sortiraniidx[j], distances[sortiraniidx[j]]))

matches.append(matches2)

return matches



Lista svih parova se nakon toga stavlja u filter. On odvaja parove koje smatra pouzdanim. Gleda da li je distanca prvog najboljeg para tačke dovoljno manja od distance drugog najboljeg. Tako filtrira parove koji bi mogli da se pomešaju.

def filter(matches):

good = []

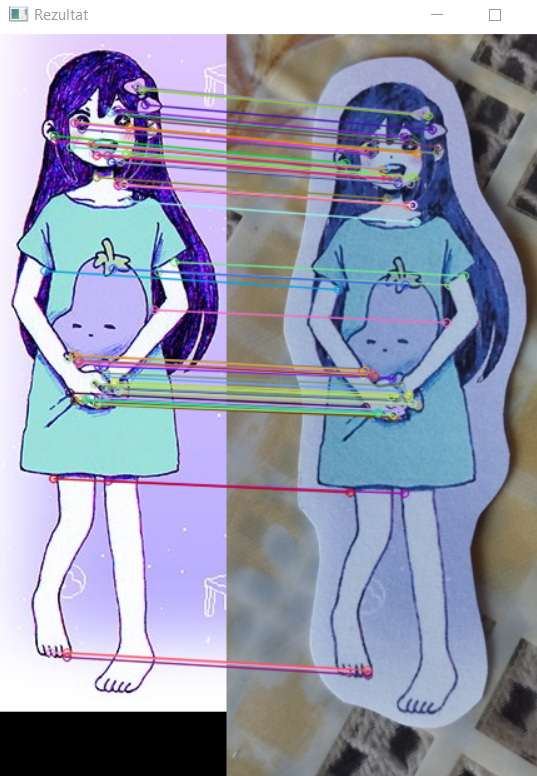
for m, n in matches:

if m.distance < 0.75 \* n.distance:

good.append([m])

return good

Krajnji rezultat:



**Prepoznavanje objekata**

Da bi program bio univerzalan i mogao da prepozna koliko god sličica odlučimo da dodamo, napravljen je tako da se iz foldera ImagesQuery učitavaju slike i imena slika. Onda se odbija file extension da bi ostalo samo ime lika.

slike = []

imena = []

lista = os.listdir('ImagesQuery')

for cl in lista:

imgCur = cv2.imread(f'ImagesQuery/{cl}', 0)

slike.append(imgCur)

imena.append(os.path.splitext(cl)[0])

Onda se nalazi lista deskriptora za svakog lika

def findDes(slike):

desList = []

for slika in slike:

kp, des = orb.detectAndCompute(slika, None)

desList.append(des)

return desList

Učitava se slika sa web kamere i zajedno sa listom deskriptora stavlja u funkciju findID, koja traži ID broj (redni broj po alfabetu) lika na slici. Ako nađe ID, onda se ime lika ispisuje na ekran.

while True:

uspeh, snimak = cap.read()

snimakboja = snimak.copy()

slika = cv2.cvtColor(snimak, cv2.COLOR\_BGR2GRAY)

rezidx = findID(snimak, desList)

if rezidx != -1:

cv2.putText(snimakboja, imena[rezidx], (50, 50), cv2.FONT\_HERSHEY\_COMPLEX, 1, (153, 51, 255), 2)

cv2.imshow('Rezultat', snimakboja)

key = cv2.waitKey(1)

if key == 27 or cv2.getWindowProperty('Rezultat', cv2.WND\_PROP\_VISIBLE) < 1:

break

ID se traži sledećom funkcijom. Ona upoređuje sliku sa web kamere sa svakom sličicom i pamti broj dobrih parova. Ako lik sa najviše parova ima makar 5 parova, ispisuje se ime tog lika.

def findID(slika, desList):

kp2, des2 = orb.detectAndCompute(slika, None)

matchevi = []

rezidx = -1

try:

for des in desList:

matches = bfmatch(des, des2)

good = filter(matches)

matchevi.append(len(good))

except:

pass

if len(matchevi) != 0:

if max(matchevi) > 4:

rezidx = matchevi.index(max(matchevi))

return rezidx

Rezultat:



**☆ Hvala na pažnji☆**