Факултет за информатички науки и компјутерско инжинерство

ДИГИТАЛНО ПРОЦЕСИРАЊЕ НА СЛИКА

СЕМИНАРСКА РАБОТА

Детекција на ирис, визуелни карактеристики за ирис и споредба

Георгина Михаилова 201224

Дарко Крстевски 201203

Содржина

Апстракт

Биометриката може да се опише како технологија која се користи за идентификување или автентификација на поединнци врз основа на нивните уникатни биолошки карактеристики како што се ирис, потпис и глас. Овој труд во детали ги опишува различните алгоритми за детекција на ирис, нивните силни и слаби страни, како и нивна примена во реалниот свет. Детекцијата на ирисот се смета за еден од најпрецизните биометриски системи за идентификација на поединци.

Клучни зборови: процесирање на слика, ирис, биометрика, алгоритми, препознавање на ирис.

Вовед

Ирисот е еден од најуникатните и препознатливи карактеристики на човекот. Во овој проект, нашата главна цел е да ја истражеме сложеноста на препознавањето на ирисот и да ги истражиме различните визуелни аспекти што ја прават идентификацијата на ирисот можна. Тука, ние ќе анализираме различни слики на ирисот за да стекнеме подлабоко разбирање на уникатните обрасци и карактеристики што го прават секој ирис уникатен. Нашиот фокус ќе биде на користење на најсовремени алгоритми и техники за прецизно откривање и класификација на ирисот со цел безбедна биометриска идентификација. Ние ќе ги споредуваме ирисите користејќи дескриптори како SIFT (Scale-Invariant Feature Transform) и ORB (Oriented FAST and Rotated BRIEF). SIFT и ORB се најсовремени алгоритми за откривање карактеристики кои можат да идентификуваат најразлични обрасци во слики, што ги прави идеални за препознавање на ирисот. Нашата цел е да ги имплементираме истите и да ја оцениме ефективноста и ефикасноста на овие алгоритми во откривањето и споредувањето на обрасците на ирисот и да одредиме кој од нив дава подобри резултати за препознавање на ирисот.

ИМПЛЕМЕНТАЦИЈА

Нашата имплементација започнува со тоа што првично се вчитуваат библиотеките кои се потребни за правилно функционирање на кодот, ние за оваа имплементација ги користиме библиотеките OpenCV и numpy и се читаат две слики кои треба да се обработуваат. Потоа, двете слики индивидуално ја повикуваат функцијата detect_eyes чија главна цел е да детектира око на соодветната слика.

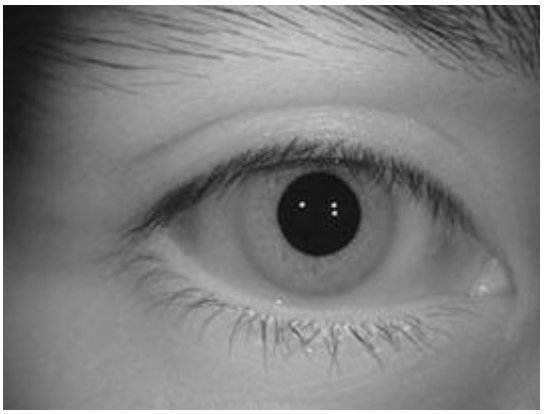
```
def detect_eyes(img):

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    eye_cascade = cv2.CascadeClassifier("haarcascade_eye.xml")

    eyes = eye_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))


    if len(eyes) > 0:

        x, y, w, h = eyes[0]
        cv2.imshow("eyes detected", img[y:y + h, x:x + w])
        cv2.waitKey(0)
        cv2.destroyAllWindows()
        return img[y:y + h, x:x + w]

    else:
        return ""
```

Во овој дел тука, функцијата зема слика како влезен аргумент и ја враќа областа на првото откриено око на сликата. Функцијата ја користи библиотеката OpenCV за обработка на слики. Влезната слика прво се трансформира од нејзината оригинална претстава на боја во црно-бела скала со помош на функцијата cv2.cvtColor(). Ова е затоа што каскадните класификатори на Haar, кои се користат за откривање на објекти во OpenCV, најдобро функционираат со црно-бели слики. Haar каскадниот класификатор за откривање на очи се вчитува со помош на функцијата cv2.CascadeClassifier().

haarcascade_eye.xml е XML-датотека која содржи претходно обучен Haar каскаден класификатор. Каскадните класификатори на Haar се еден вид алгоритам за откривање објекти кој се користи во компјутерската визија и обработката на слики. Тие работат со пребарување на одредени обрасци (patterns) во слика што одговараат на карактеристиките на предметот на интерес, во овој случај, очите. Каскадниот класификатор Haar во haarcascade_eye.xml е обучен на голема база на податоци од слики што содржат очи и може да детектира очи на нови, непознати слики со висок степен на точност. XML-датотеката ги содржи параметрите на класификаторот, како што се вредностите на прагот, карактеристиките на Haar што се користат за откривање и тежините поврзани со секоја карактеристика. Функцијата detectMultiScale() се повикува на црно-белата слика и на каскадниот класификатор. Аргументот scaleFactor го одредува факторот со кој функцијата detectMultiScale() треба да ја зголеми големината на прозорецот за пребарување очи во секоја итерација, аргументот minNeighbors го одредува минималниот број на соседи што

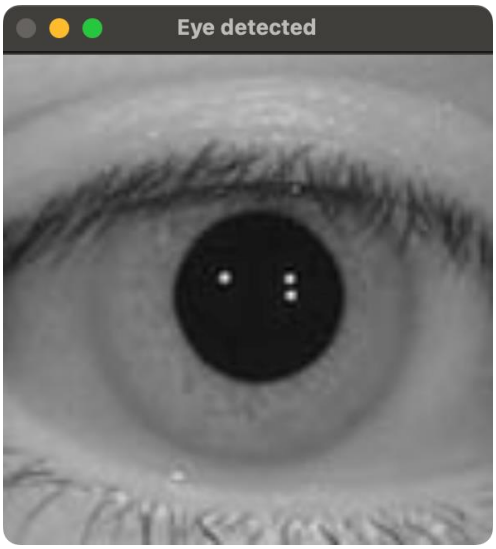
еден регион мора да го има за да се смета за око, а аргументот minSize ја одредува минималната големина на регионот што треба да се смета како око.Кодот потоа проверува дали се откриени очи на сликата со проверка на должината на променливата *'eyes'*. Ако должината на *'eyes'* е поголема од 0, што значи дека се откриени едно или повеќе очи, тогаш функцијата враќа сечена слика која го содржи првиот регион кој содржи око. Сликата се прикажува со помош на функцијата cv2.imshow(). Функцијата cv2.waitKey(0) се користи за да се чека клучен настан, а функцијата cv2.destroyAllWindows() се користи за затворање на прозорецот што ја прикажува сликата. Ако должината на *'eyes'* не е поголема од 0, што значи дека не се откриени очи, тогаш се враќа празен стринг.

За визуелно да ја прикажеме улогата на оваа функција, ги користиме сликите (Слика 1. и Слика 2.) кои детално се обработуваат со дадените фунцкионалности, а како резултат од истата се добиваат сликите (Слика 3. и Слика 4.).

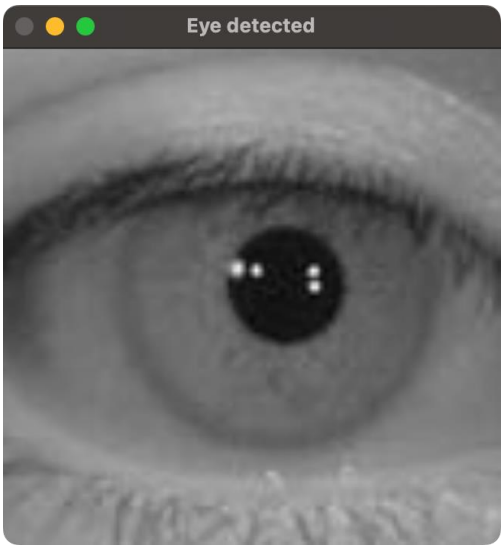


Слика 1. Ирис 1

Слика 2. Ирис 2



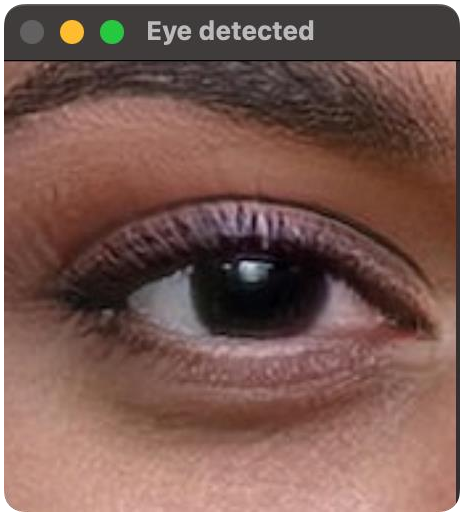
Слика 3. Детектирано око 1



Слика 4. Детектирано око 2

За целите на оваа функција, би додале уште еден примерок каде се дектира око кој е прикажан на (Слика 5. и Слика 6.). Во овој примерок, на фунцкијата како влез му се дава целосно лице, а како излез се добива само окото од тоа лице.

Слика 5. Жена



Слика 6. Детектирано око 3

ORB дескриптор

ORB (Oriented FAST and Rotated BRIEF) е метод за екстракција на карактеристики што вообичаено се користи во дигиталната обработка на ирисот. Овој дексриптор е комбинација од методот за откривање агол FAST и методот за дескриптори BRIEF (Бинарни робусни независни елементарни карактеристики). Во дигиталната обработка на ирисот, дескрипторот ORB се користи за извлекување уникатни карактеристики од сликата на ирисот. Овие карактеристики потоа се користат за прикажување на ирисот на компактен и описен начин, што овозможува споредување на сликите на ирисот и препознавање на истиот. Дескрипторот ORB е особено корисен во дигиталната обработка на ирисот бидејќи е пресметковно ефикасен, робустен на шум и непроменлив за размерот и ротацијата на сликата. Ова го прави добро прилагоден за обработка на слики од ирис, кои можат да бидат бучни и подложни на значителни варијации во обемот и ориентацијата. Генерално, ORB е важна алатка во дигиталното процесирање на ирисот, обезбедувајќи компактна и карактеристична претстава на сликите на ирисот што може да се користи за споредба и препознавање.

За целите на овој проект, нашата имплементација е следна:

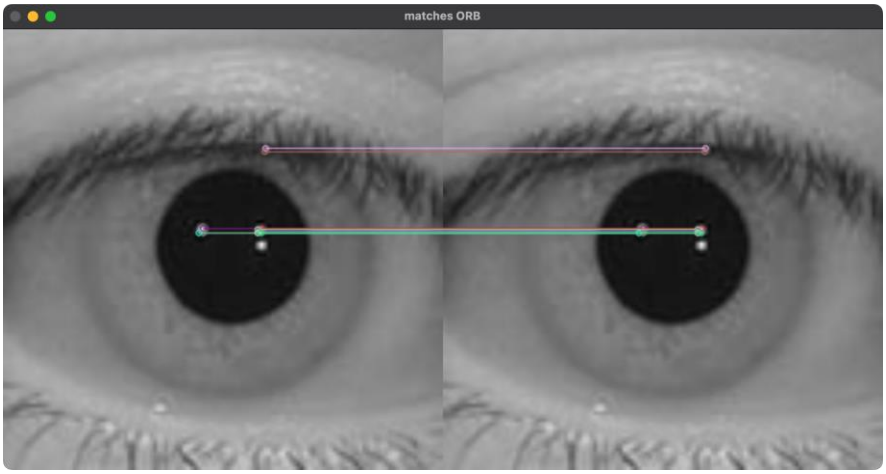
```
def detect_and_compare_irises_orb(iris1, iris2):
    gray_iris1 = prepare_images(iris1)
    gray_iris2 = prepare_images(iris2)

    orb = cv2.ORB_create()

    kp1, des1 = orb.detectAndCompute(gray_iris1, None)
    kp2, des2 = orb.detectAndCompute(gray_iris2, None)

    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

    matches = bf.match(des1, des2)
```

```
    matches = sorted(matches, key=lambda x: x.distance)

    img_matches = cv2.drawMatches(gray_iris1, kp1, gray_iris2, kp2, matches[:10], None,
                                  flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
    num_matches = len(matches)
    cv2.imshow("matches ORB", img_matches)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    print("ORB number of matches: " + str(len(matches)))

    if num_matches >= 100:
        print('ORB: The irises match.')
    elif num_matches >= 0:
        print('ORB: The irises do not match.')
    return len(matches)
```

Кодот најпрво ги подготвува сликите со ирис преку повикување на функцијата prepare_images чија главна цел е да ги конвертира истите во слики со сиви тонови и да направи прилагодување во единечна големина. Следно, објектот ORB се креира со помош на функцијата cv2.ORB_create() од библиотеката OpenCV. Потоа, точките на интерес и дескрипторите се детектираат и се пресметуваат од двете слики користејќи го методот detectAndCompute на објектот ORB. Се креира brute-force matccher со помош на cv2.BFMatcher користејќи метрика за растојание од Хамин и crossCheck поставен на True, што значи дека функцијата ќе врати совпаднување само доколку истиот дескриптор се појавува на двете слики. Потоа, совпаднатите делови, се подредуваат врз основа на нивните растојанија во растечки редослед. Конечно, совпаднатите делови се извлекуваат и се прикажуваат со помош на методот cv2.drawMatches на OpenCV. Бројот на совпаѓања се брои и се печати на конзола. Ако бројот на совпаѓања на двете слики е еднаков или поголем од 100, ирисите се сметаат за исти, во спротивно, се смета дека не се совпаѓаат. Бројот на совпаѓања, во случајот 100, добиен од процесот на препознавање на ирисот е показател за тоа колку се слични двете слики на ирисот. Ако бројот на совпаѓања е висок, тоа значи дека има многу слични карактеристики меѓу двете ириси и најверојатно тие се исти. Од друга страна, ако бројот на совпаѓања е мал, тоа значи дека има многу малку сличности меѓу двата ириси и веројатно не станува збор за ист ирис. Вредноста 100 се користи како праг за да се донесе бинарна одлука за статусот на совпаѓање или несовпаѓање на ирисите. Причината поради која ние ја одбравме вредноста 100 да биде праг на совпаѓање е поради тоа што индивидуално сметаме дека е погоден за нашето податочно множество. Со извршување на овој код, се добиваат следните резултати:

```
ORB number of matches: 120
ORB: The irises match.
```

Слика 7. Број на совпаѓања за првиот сет примерок кај ОРБ

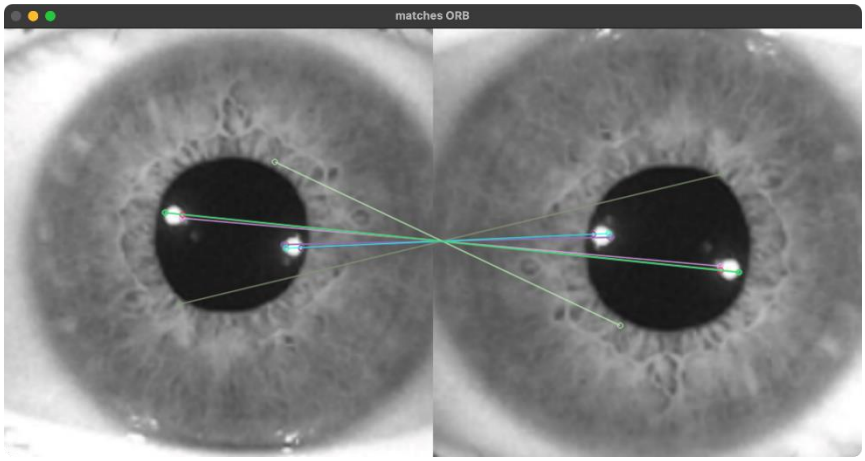


Слика 8. ORB исцртување на совпаѓања за првиот сет примерок

За овој примерок, се добиваат 120 совпаднувања и според претходно наведениот праг, ова значи дека ирисите се исти. За вториот примерок, се добиваат 119 совпаднувања. Со овие два примероци може да се заклучи дека алгоритамот работи коректно и детектира исти ириси. Иако на вториот примерок размерот на сликите е различен и едната од сликите е ротирана, кај ORB тоа не претставува проблем бидејќи е непроменлив и во тој случај.

```
ORB number of matches: 203
ORB: The irises match.
```

Слика 9. Број на совпаѓања на вториот сет примерок кај ОРБ

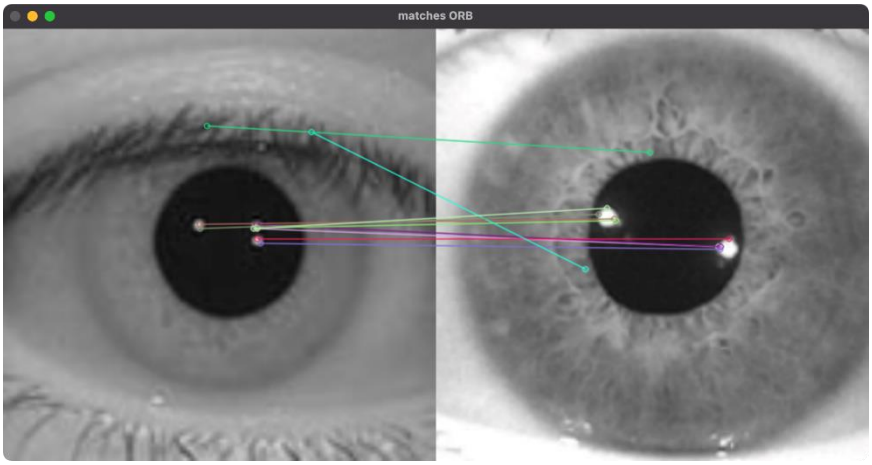


Слика 10. ORB исцртување на совпаѓања за првиот сет примерок

Следните сет примероци прикажуваат резултати за ириси кои се различни и не се совпаѓаат.

```
ORB number of matches: 39
ORB: The irises do not match.
```

Слика 11. Број на совпаѓања на третиот сет примерок кај ОРБ



Слика 12. ORB исцртување на совпаѓања за третиот сет примерок

SIFT дескриптор

Scale-Invariant Feature Transform (SIFT) е широко користен алгоритам во компјутерската визија и обработката на слики. Се користи за извлекување уникатни и робустни карактеристики од слика, кои потоа може да се користат за препознавање објекти, совпаѓање на слики и 3D реконструкција. SIFT дескрипторот работи така што открива точки на интерес на сликата, кои се точки кои се сметаат за карактеристични и информативни за содржината на сликата. Откако ќе се детектираат овие точки, алгоритмот SIFT создава збир на локални дескриптори околу секоја точка. Овие дескриптори се создаваат со пресметување на хистограмите за ориентација на градиент во збир од соседни пиксели околу точката на интерес. Една од клучните предности на SIFT е неговата способност цврсто да ги извлекува карактеристиките од сликите дури и при варијации како што се скалирање, ротација и промени во осветлувањето. Ова се постигнува со користење на разликата од Гаусовата (DoG) пирамида за откривање на точки на интерес, што овозможува откривање на карактеристики кои не се менуваат во обем. Дескрипторот SIFT исто така ги користи хистограмите за ориентација на градиент за да ја претстави локалната структура на сликата, што помага да се обезбеди стабилна репрезентација дури и кога сликата претрпува мали промени во погледот. Друга важна карактеристика на SIFT е неговата способност да се справува со афини трансформации, кои се промени во сликата предизвикани од промени во гледиштето. Алгоритмот SIFT може да се справи со афините трансформации користејќи адаптивен регион за поддршка околу секоја точка на интерес. Во продолжение ќе биде прикажана нашата имплементација на SIFT дескрипторот.

```python
def detect_and_compare_irises_sift(iris1, iris2):
    gray_iris1 = prepare_images(iris1)
    gray_iris2 = prepare_images(iris2)

    sift = cv2.SIFT_create()
    kp1, des1 = sift.detectAndCompute(gray_iris1, None)
    kp2, des2 = sift.detectAndCompute(gray_iris2, None)

    des1 = des1.astype(np.float32)
    des2 = des2.astype(np.float32)

    bf = cv2.BFMatcher()
    matches = bf.knnMatch(des1, des2, k=2)

    good_matches = []
    for m, n in matches:
        if m.distance < 0.75 * n.distance:
            good_matches.append([m])

    num_matches = len(good_matches)

    print("SIFT number of matches: " + str(num_matches))

    if num_matches >= 40:
        print('SIFT: The irises match.')
    elif num_matches >= 0:
        print('SIFT: The irises do not match.')
```

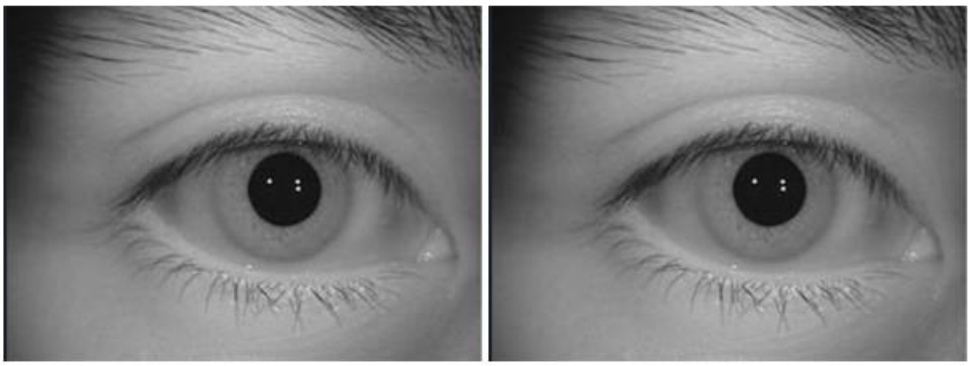
Оваа функција прима две слики од ирис како влез и ги извршува следните чекори за да утврди дали ирисите се совпаѓаат или не:

1. **Промена на големината на сликите на ирисот:**
   Големината на сликите се менува до заедничка големина со помош на функцијата prepare_images. Овој чекор е неопходен за да се осигураме дека двете слики се анализираат на иста скала.
2. **Откривање на клучни точки и пресметување на дескриптори SIFT:**
   Алгоритмот SIFT се користи за откривање важни карактеристики на двете слики и за пресметување на дескриптор за секоја карактеристика. Функцијата cv2.SIFT_create создава SIFT објект, а detectAndCompute се користи за откривање на клучните точки и пресметување на дескрипторите.
3. **Усогласување на дескрипторите SIFT:**
   Дескрипторите SIFT на двете слики се усогласуваат со помош на совпаѓач со brute force. Функцијата cv2.BFMatcher создава brute force совпаѓач , а knnMatch се користи за пронаоѓање на k најблиските соседи за секој дескриптор.
4. **Филтрирање на совпаднатите точки:**
   Совпаднатите точки се филтрираат со помош на тестот за сооднос на Лоу, кој ги елиминира совпаѓањата кои не се робусни. Тестот го споредува растојанието помеѓу најблискиот и вториот најблизок сосед и само совпаѓањата со растојание до

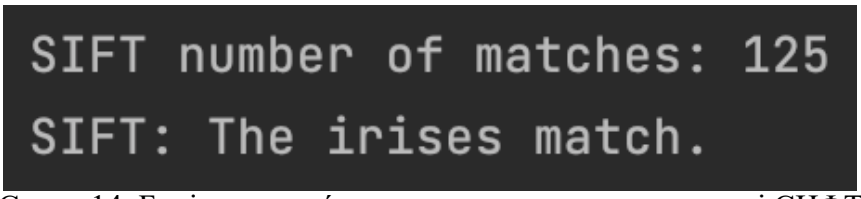
најблискиот сосед што е помало од 0,75 пати од растојанието до вториот најблизок сосед се сметаат за добри совпаѓања.

5. **Пресметување на резултатот за сличност:**
   Бројот на совпаѓања се користи како резултат на сличност помеѓу двете слики од ирисот. Ако бројот на совпаѓања е поголем или еднаков на 40, се смета дека ирисите се совпаѓаат. Ако бројот на совпаѓања е помал од 40, се смета дека ирисите не се совпаѓаат.

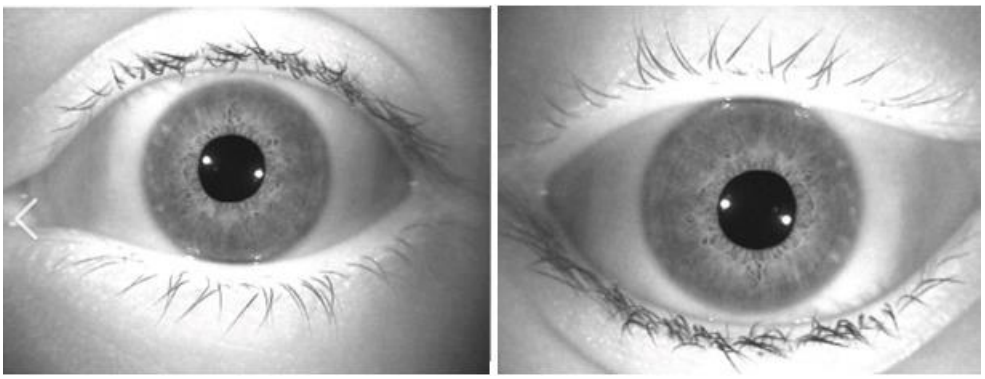
Во прилог визуелно, преку слики, ќе ја објаснеме логиката зад овој алгоритам.



Слика 13. Прв сет примерок СИФТ.



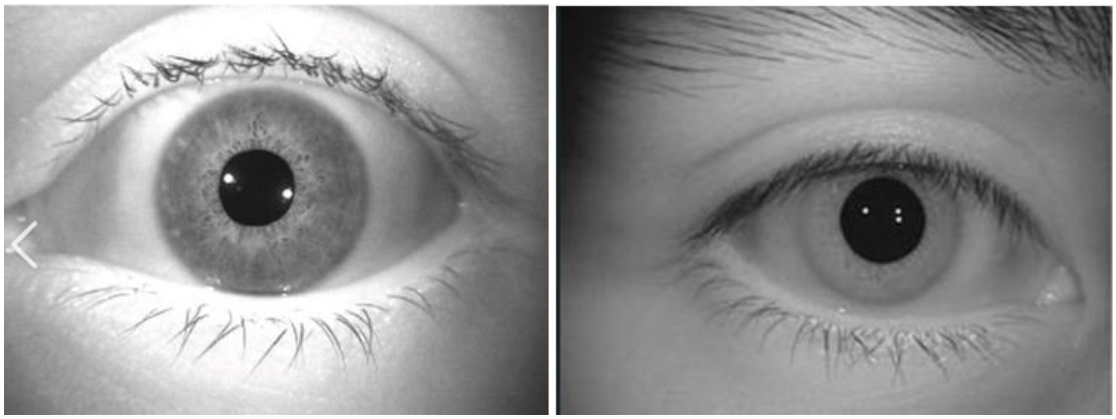
Слика 14. Број на совпаѓања на првиот сет примерок кај СИФТ.



Слика 15. Втор сет примерок СИФТ.

```
SIFT number of matches: 94
SIFT: The irises match.
```

Слика 16. Број на совпаѓања на вториот сет примерок кај СИФТ.



Слика 17. Трет сет примерок СИФТ.

```
SIFT number of matches: 1
SIFT: The irises do not match.
```

Слика 18. Број на совпаѓања на третиот сет примерок кај СИФТ.

Според ова, може да се заклучи дека SIFT дескрипторот е робустен и ефективен дескриптор на карактеристики за совпаѓање на слики. Во првиот сет на слики, каде што и двете слики се сосема исти, дескрипторот SIFT откри голем број на совпаѓања (125). Ова покажува дека SIFT може прецизно да детектира и усогласува слични карактеристики помеѓу две слики.

Во вториот сет на слики, каде што една слика е малку изменета, бројот на совпаѓања е помал (94) отколку во првиот сет. Сепак, сè уште се добива значителен број на совпаѓања, што сугерира дека SIFT е способен да открие и усогласи слични карактеристики дури и на слики кои се малку различни.

Во третиот сет на слики, каде што ирисите се сосема различни, бројот на совпаѓања е многу помал (19). Овој резултат покажува дека SIFT може да разликува слики кои се значително различни и може да се користи за да се утврди дали две слики се различни.

Како заклучок, дескрипторот SIFT е робустен и ефективен дескриптор на карактеристики што може да се користи за совпаѓање и споредба на слики.

Споредба помеѓу SIFT и ORB

Споредувањето на резултатите од откривањето на карактеристиките на сликите и алгоритмите за совпаѓање е важно за да се одреди кој метод најдобро одговара за одредена задача. Во овој случај, перформансите на SIFT и ORB ќе се споредат врз основа на резултатите од совпаѓањето на сликите на три групи слики. Првиот сет се состои од две идентични слики, вториот сет се состои од две слични слики со разлики во ротација и размер, а третиот сет се состои од две сосема различни слики. Бројот на совпаѓања откриени од SIFT и ORB ќе се користи како метрика за одредување на ефективноста на секој метод.

Според резултатите добиени погоре, се открива дека ORB го надминува SIFT во однос на точноста на совпаѓање на сликите. И во првиот и во вториот сет на слики, каде што сликите беа или идентични или малку изменети, ORB беше во можност да открие поголем број на совпаѓања од SIFT. Ова сугерира дека ORB е способен да детектира и да одговара на слични карактеристики поефикасно од SIFT во овие сценарија.Во третиот сет на слики, каде што сликите беа значително различни, ORB сепак беше подобар од SIFT, откривајќи 39 совпаѓања во споредба со 1 совпаѓање на SIFT. Ова покажува дека ORB е подобро способен да прави разлика помеѓу слики кои се значително различни. Вреди да се напомене дека резултатите зависат од специфичните слики и специфичните услови под кои се врши совпаѓањето на сликата. Сепак, врз основа на овие резултати, очигледно е дека ORB е поефективен дескриптор на карактеристики од SIFT за совпаѓање на слики.

Една од можните причини за супериорните перформанси на ORB е тоа што тој претставува комбинација од техники за откривање и опис на карактеристики, вклучувајќи го алгоритмот FAST за откривање на агол и BRIEF дескрипторот. Оваа комбинација овозможува ORB да биде и брз и прецизен во откривањето и усогласувањето на карактеристиките, што го прави добро прилагоден за апликации во реално време.

Визуелни карактеристики на ирис

Следниве се некои од визуелните карактеристики на ирисот што може да се детектираат или анализираат:

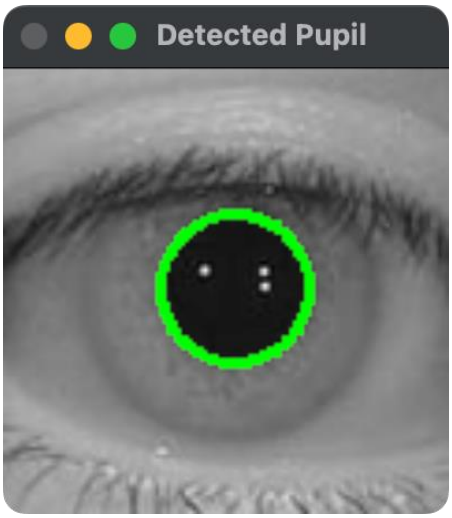
1. **Боја на ирис:**
   Бојата на ирисот може да варира многу, почнувајќи од сина до кафеава, зелена или лешник.
2. **Текстура на ирис:**
   Текстурата на ирисот може да се опише како мазна или груба, а може да се забележи и присуство на пеги или други пигментирани дамки.
3. **Дијаметар на ирис:**
   Дијаметарот на ирисот е мерење од едната до другата страна на ирисот.
4. **Дијаметар на зеница:**
   Големината на зеницата може да се користи за да се одреди количината на светлина што влегува во окото и може да се менува како одговор на различни стимули.

5. **Облик на зеница:**
   Обликот на зеницата може да биде кружен или овален, а варијациите во формата може да укажат на одредени состојби на очите.

Овие визуелни карактеристики може да се откријат со помош на техники за обработка на слики, како што се откривање на рабови, прагови и анализа на контури, заедно со алгоритми за машинско учење. Во нашата имплементација за визуелните карактеристики на ирисот, одлучивме да вклучиме алгоритми за пронаоѓање на дијаметар на зеницата и облик на ирис.

Големина на зеница

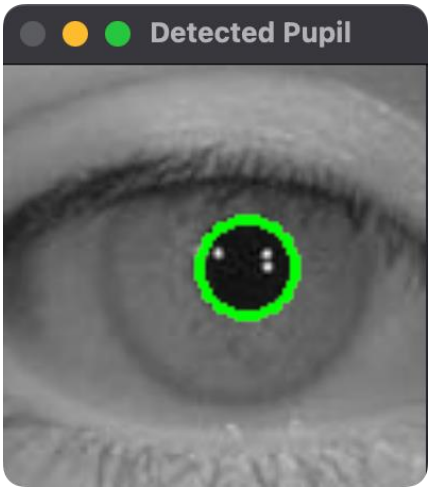
Нашиот код ја имплементира функцијата find_pupil_size(img) чија главна цел е да ја пронајде зеницата и да го прикаже дијаметарот на истата. Кодот зема слика како влез и ја обработува сликата за да ја открие големината на зеницата. Прво ја претвора сликата во црно-бела, ја замаглува за да го намали шумот, а потоа создава бинарна слика со прагови. Потоа ги наоѓа контурите во бинарната слика, ги подредува по површина и ја избира најголемата. Ако соодносот на граничниот правоаголник околу најголемата контура е приближно 1, тој го пресметува центарот и дијаметарот на зеницата, црта круг околу неа и го печати дијаметарот. Конечно, ја прикажува сликата со откриената зеница. Ако соодносот не е приближно 1, отпечатува дека зеницата не е пронајдена. По извршување на алгоритамот, визуелната репрезентација за тоа како фунцкионира е следна:



Слика 19. Детектирана зеница 1

```
Diameter of the pupil: 41
```

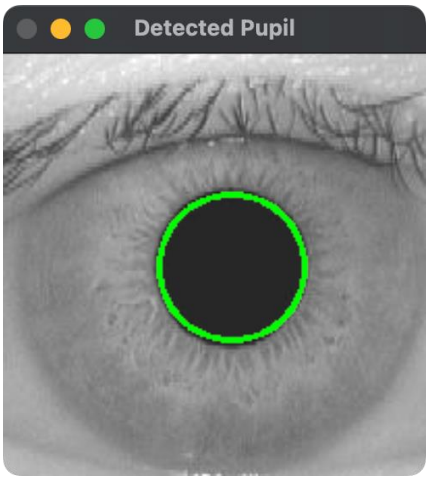
Слика 20. Дијаметар на зеница 1



Слика 21. Детектирана зеница 2

```
Diameter of the pupil: 26
```

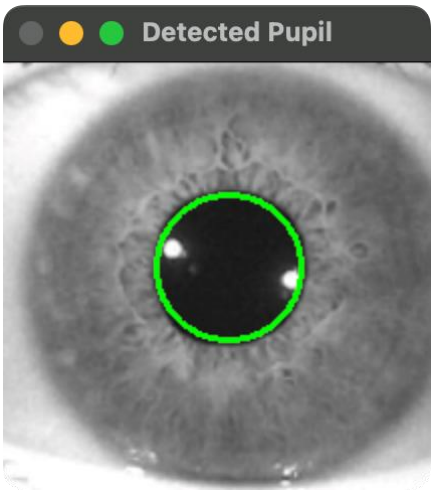
Слика 22. Дијаметар на зеница 2



Слика 23. Детектирана зеница 3

```
Diameter of the pupil: 71
```

Слика 24. Дијаметар на зеница 3



Слика 25. Детектирана зеница 4

```
Diameter of the pupil: 73
```

Слика 26. Дијаметар на зеница 4

Како заклучок, овој алгоритам е едноставна имплементација за откривање на големината на зеницата во сликата. Користи различни техники за обработка на слики, како што се конверзија во сиви тонови, заматување, праг и откривање на контури за да се постигне целта. Конечниот излез на кодот е дијаметарот на откриената зеница, која е нацртана на оригиналната слика.

Облик на ирис

Нашата имплементација на овој алгоритамот претставува код за компјутерска визија што го открива обликот на ирисот на сликата. Ја користи библиотеката OpenCV во Python за извршување на различни задачи за обработка на слики. Кодот прво применува Гаусово замаглување за да го намали шумот на сликата, а потоа го користи детекторот за рабови Canny Edge Detector за да ги пронајде рабовите на заматената слика. Кодот потоа ги наоѓа контурите на сликата, ја избира контурата со најголема површина, одговара на елипса на контурата и ја црта елипсата на сликата. Кодот го пресметува соодносот на елипсата, што е односот на главната оска со помалата оска, и го одредува обликот на ирисот врз основа на односот на изгледот. Конечниот резултат, кој е или „Circular“, „Spatulate“ или „Oval“, се печати на конзолата.

Излезот од оваа функција, во зависност од обликот е следен:

```
Iris shape is: Oval
```

Слика 27. Детектиран овален ирис

Според нашите примероци, оваа имплементација се докажа како доста ефективна затоа што користи добро воспоставени техники во компјутерската визија и обработката на слики за прецизно одредување на обликот на ирисот на сликата.

ЗАКЛУЧОК

Овој проект имаше за цел да спореди два дескриптори на слики, SIFT и ORB, за да ја одреди нивната ефикасност во препознавањето на моделите на ирисот. Резултатите покажаа дека ORB е подобро прилагоден за оваа задача, бидејќи дава попрецизни резултати во споредба со SIFT. Ова се должи на способноста на ORB да се справува со ротации, промени на скалата на сликата и варијации на осветленоста, кои се вообичаени проблеми во препознавањето на ирисот. SIFT, од друга страна, е добро воспоставен дескриптор и наоѓа апликации во области како што се препознавање објекти, совпаѓање на слики и 3D реконструкција. SIFT е ефикасен во откривањето карактеристики и кај слики кои се различно скалирани, што го прави робустен во однос на промените на скалата. Сепак, можеби не е најдобриот избор за препознавање на ирисот, бидејќи шаблоните на ирисот може значително да се променат помеѓу две слики направени во различно време или под различни услови на осветлување.

Покрај споредувањето на дескрипторите, проектот се фокусираше и на визуелните карактеристики на ирисот, како што се големината на зеницата и обликот на ирисот. Алгоритмите користени за извлекување на овие карактеристики беа ефективни и дадоа значителен придонес во целокупната точност на препознавање. Комбинацијата на дескриптори и визуелни карактеристики овозможи посеопфатна анализа на ирисот, што доведе до попрецизно препознавање на моделите на ирисот.

За крај, употребата на ORB и SIFT дескриптори, како и алгоритми за откривање на обликот на ирисот и големината на зеницата може значително да ја подобри точноста на системите за препознавање на ирисот. Овој проект ја нагласува важноста од изборот на соодветни алгоритми и техники за препознавање на ирисот и ја демонстрира нивната ефикасност во реалните апликации.

Користена литература

- Iris Recognition using Python OpenCV Script [Extract Iris Feature] (phdservices.org)
- What is Iris Recognition and how does it work? - NEC New Zealand
- New Methods in Iris Recognition | IEEE Journals & Magazine | IEEE Xplore
- How Iris Recognition Works - ScienceDirect