**УНИВЕРЗИТЕТ „ СВ. КИРИЛ И МЕТОДИЈ” – СКОПЈЕ  
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО**

****

**СЕМИНАРСКА РАБОТА ПО ПРЕДМЕТОТ  
ДИГИТАЛНО ПРОЦЕСИРАЊЕ НА СЛИКА  
TEMA:  
ДЕТЕКЦИЈА НА ЛИЦА И ЗАМЕНА НА ЛИЦЕ**

Ментор:   
д-р Ивица Димитровски

Изработиле:  
Живче Глигоров 216139  
Михаил Ѓорѓиев 216027

Скопје, Јуни 2023

1. Вовед

- Замената на лица е техника која овозможува замена на лицето на едно лице со друго во слика или видео, притоа зачувувајќи го оригиналниот контекст и изглед.   
Замената на лица има различни апликации во забавата, игрите и уметноста, но може да се злоупотреби и за злонамерни цели, како што се создавање лажни вести, имитирање познати личности или нарушување на приватноста.  
Постојат различни видови на алгоритми за замена на лица, во зависност од тоа како тие го синтетизираат заменетиот лик. Според , тие можат да се класифицираат во три категории: засновани на замена, базирани на модел и базирани на учење.  
  
  
➤Засновани на замена:

- Методите засновани на замена се базират на копирање и залепување на изворниот регион на лицетото со целиот регион на лицето, а потоа се применуваат некои техники за обработка на слики како што се мешање, искривување или прилагодување на бојата за да го направите резултатот пореален. Овие методи се брзи и лесни за имплементација, но тие често создаваат забележителни артефакти и слики со низок квалитет.

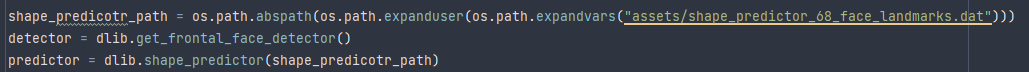
➤Базирани на модел:

- Методите базирани на модели користат 3D модел на лице за да ја пренесат формата и текстурата на лицето од изворот до целта. Овие методи можат подобро да се справат со различни пози и изрази од методите засновани на замена, но бараат прецизна 3Д реконструкција и регистрација на лицето, што може да биде предизвикувачко и пресметковно скапо.  
  
  
➤Базирани на учење:

- Методите засновани на учење користат длабоки невронски мрежи за да научат функција за мапирање од изворното лице до целното лице. Овие методи можат да генерираат висококвалитетни и фотореалистични слики, но им треба голема количина на податоци за обука и пресметковни ресурси. Некои примери на методи засновани на учење се DeepFakes, FaceShifter и Face X-ray.

2. Методи базирани на замена

- Методите базирани на замена во Замената на лице се техники кои го заменуваат пределот на лицето на една слика со друго лице, додека го зачувуваат остатокот од контекстот на сликата.   
Овие методи обично се состојат од три чекори: откривање лице, усогласување на лицето и мешање на лице.  
  
  
➤Откривањето лице е процес на лоцирање и извлекување лица од слики. Постојат различни алгоритми за откривање лица, како што се Haar Cascade, HOG, MTCNN и RetinaFace. Откривањето лице обично е првиот чекор во Замената на лице, бидејќи ги обезбедува кандидатите за замена на лица.  
  
  
➤Порамнувањето на лицето е процес на трансформирање и усогласување на лицата до заеднички координатен систем, како што е средната форма на лице. Ова осигурува дека заменетите лица имаат конзистентна поза, размер и ориентација. Порамнувањето на лицето може да се направи со користење на обележја на лицето, кои се клучни точки на лицето што укажуваат на локацијата на очите, носот, устата и другите карактеристики. Некои примери на детектори за обележје на лицето се Dlib, FAN и 3DDFA.  
  
  
➤Мешањето на лица е процес на беспрекорно спојување на изворното лице со целната слика, притоа зачувувајќи ги осветлувањето, бојата и текстурата на двете слики. Мешањето на лицето може да се направи со користење на различни техники, како што се мешање со Поасон, пренос на бои, совпаѓање на хистограм.

2.1. Откривање на лицето  
  
dlib.get\_frontal\_face\_detector() и dlib.shape\_predictor() се две функции од библиотеката dlib кои се корисни за откривање лица и препознавање на обележја на лицето.   
  
  
Првата функција враќа објект што може да детектира лица на слика користејќи хистограм на ориентирани градиенти (HOG) извлекувач на карактеристики и линеарен класификатор.  
Хистограм на ориентирани градиенти (HOG) е техника за екстракција на карактеристики која може да се користи за откривање и препознавање на објекти во компјутерската визија. HOG работи така што ја дели сликата на мали области наречени ќелии и пресметува хистограм на насоките на градиентот во секоја ќелија. Хистограмите потоа се нормализираат и се спојуваат за да формираат вектор на карактеристика што ја претставува формата и изгледот на регионот на сликата. Функциите на HOG се цврсти за промени во осветлувањето, размерот и позата и можат да доловат локални и глобални информации за сликата.   
  
  
Втората функција зема слика и рамка за ограничување на лице како влез и враќа објект што може да ја предвиди локацијата на 68 обележја на лицето, како што се очите, носот, устата итн. shape\_predictor\_68\_face\_landmarks.dat е претходно обучен модел кој може да открие 68 обележја на лицето на дадена слика на човечко лице. Знаците на лицето се точки кои ја претставуваат формата и контурата на лицето, како што се аглите на очите, врвот на носот и рабовите на усните. Моделот shape\_predictor\_68\_face\_landmarks се заснова на алгоритам за машинско учење кој беше обучен на голема база на податоци од етикетирани слики на лица. Моделот може да се користи за различни апликации, како што се усогласување на лица, препознавање лица, замена на лица и анализа на изразот на лицето.  
A screen shot of a computer

Description automatically generated with low confidence

*Слика 1. Обележја на лице*

A picture containing text, screenshot, font

Description automatically generated2.2. Вчитување на слики и добивање на landmarks

Овој код е функција која чита датотека со слика и враќа објект на сликата со променета големина. Функцијата зема еден параметар, file\_name, што е името на датотеката со слика што треба да се прочита. Функцијата ги прави следните чекори:  
  
  
- Ја користи функцијата cv2.imread за читање на датотеката со слика во режим на боја и складирање во променлива наречена слика.  
  
  
- Ја користи функцијата cv2.resize за да ја промени големината на сликата до нејзините оригинални димензии и да ја складира во истата променлива слика.  
  
  
- Го враќа објектот на сликата како излез од функцијата.  
  
  
Целта на овој код е да чита и да ја промени големината на датотеката со слика користејќи ја библиотеката OpenCV, која е популарна библиотека за апликации за компјутерска визија.

A screen shot of a computer

Description automatically generated with medium confidence  
  
Овој код е функција која зема слика како влез и враќа матрица од координати кои ги претставуваат обележјата на лицето на лицето на сликата. Знаците на лицето се точки на лицето што може да се користат за препознавање, усогласување или манипулација на лица.  
  
  
-Првата линија од кодот повикува детекторски објект кој ги наоѓа и ги враќа сите лица на сликата како правоаголници. Објектот на детекторот е иницијализиран на друго место со претходно обучен модел. Вториот аргумент на функцијата на детекторот е 1, што значи дека сликата се зголемува еднаш за да се подобри точноста на откривање.  
  
  
-Втората и третата линија од кодот проверуваат дали детекторот вратил точно еден правоаголник. Ако има повеќе од еден или нула правоаголници, тоа значи дека детекторот не успеал да најде ниту едно лице на сликата и се прави исклучок.  
  
  
-Последната линија од кодот враќа матрица од координати кои ги претставуваат фацијалните обележја на лицето во правоаголникот. Предвидувачкиот објект е иницијализиран на друго место со претходно обучен модел кој може да процени 68 обележја на лицето од слика и правоаголник. Функцијата предвидувач ја зема сликата и правоаголникот како аргументи и враќа облик на објект кој содржи 68 делови, од кои секој одговара на обележје на лицето. Јамката for се повторува над овие делови и ги извлекува нивните x и y координати, кои потоа се складираат во матрица со нула.

2.3. Порамнувањето на лицето

A picture containing text, screenshot, software, font

Description automatically generatedПорамнувањето на лицето е компјутерска визија, технологија за идентификување на геометриската структура на човечките лица во дигитални слики. Дадена локација и големина на лице, тоа автоматски го одредува обликот на деловите од лицето како што се очите и носот. Ова се прави со автоматско откривање на фацијални фидуцијални точки, исто така наречени фацијални знаменки или краеви, а потоа користејќи стандардни методи за трансформација како што се афини/слични трансформации. Овие фидуцијални точки се предефинирани точки на сликата на лицето кои се главно лоцирани или центрирани околу деловите од лицето како што се очите, носот, брадата и аглите на устата .  
  
  
Целта на порамнувањето на лицето е да се обезбеди соодветност помеѓу различни лица, така што следните задачи за анализа на слики можат да се изведуваат на заедничка основа. Порамнувањето на лицето може да се реализира како процес на трансформирање различни множества точки од влезни слики во еден координатен систем, кој е дефиниран како излезен координатен систем и служи како стационарна референтна рамка . Порамнувањето на лицето е суштински посреден чекор за решавање задачи како препознавање на изрази, препознавање на лица, моделирање на лица итн.

Овој код е за порамнување на лицето, што е техника за промена на изгледот на лицето со користење на две слики: една слика со оригиналното лице и друга слика со целниот изглед.  
Порамнувањето на лицето се прави со наоѓање на трансформација која ги пресликува точките од оригиналното лице во точките од целниот изглед.  
  
  
  
  
Функцијата transformation\_from\_points(points\_of\_img1\_landmarks, points\_of\_img2\_landmarks) прима два аргументи: points\_of\_img1\_landmarks и points\_of\_img2\_landmarks. Овие аргументи се листи од точки кои претставуваат карактеристични обележја на лицето, како очи, нос, уста итн., за секоја од двете слики. Функцијата враќа матрица која ја опишува трансформацијата.  
  
  
Во функцијата, прво се претвораат точките во numpy масиви од тип float64, за да можат да се извршат математички операции со нив. Потоа, се пресметува центарот на тежеста на точките за секоја слика со користење на numpy.mean функцијата, и се одзема од точките за да се центрираат околу нулта. Ова е потребно за да се избегне ротација или транслација при трансформацијата.  
  
  
Следно, се пресметува стандардната девијација на точките за секоја слика со користење на numpy.std функцијата, и се дели со точките за да се нормализираат. Ова е потребно за да се избегне растегнување или стеснување при трансформацијата.  
  
  
Потоа, се користи numpy.linalg.svd функцијата за да се направи SVD (singular value decomposition) на производот од транспонираните точки од првата слика и точките од втората слика. Ова ќе врати три матрици: U, S и Vt. Со помош на овие матрици, можеме да ја пресметаме ротационата матрица R како производот на U и Vt, транспонирано.  
  
  
Конечно, ја враќаме трансформационата матрица како вертикална стапка (numpy.vstack) од хоризонтална стапка (numpy.hstack) од два дела: првиот дел е производот од (s2 / s1) \* R, што ја дава ротационата компонента на трансформацијата, а вториот дел е c2.T - (s2 / s1) \* R \* c1.T, што ја дава транслационата компонента на трансформацијата. Додаваме и трет ред од [0., 0., 1.] за да ја комплетираме матрицата до 3 x 3 димензии.  
  




*Слика 2. Пред трансформација*

*Слика 3. После трансформација*

2.4. Мешање на лицата

A picture containing text, screenshot, font, software

Description automatically generated  
  
  
Овој код е функција која создава маска за лице од слика и збир на обележја. Маската за лице е бинарна слика која ја покрива целата површина на лицето и има мазни рабови. Функцијата зема два аргументи: img и обележја. Img е низа со слики што го содржи лицето што треба да се маскира. Landmarks е список на обележја на лицето откриени со модел за усогласување на лицето.  
  
  
Функцијата ги прави следните чекори:  
  
  
- Создава нова низа со слики со иста форма како img, но исполнета со нули. Ова ќе биде сликата за маска за лице.  
  
  
- Се for-loop врти низ whole\_face\_points\_ordered, кој ги содржи индексите на обележјата што ја дефинираат целата контура на лицето. За секоја група на индекси, ги извлекува соодветните точки од списокот со обележја и создава конвексен труп околу нив. Конвексен труп е многуаголник кој ги опфаќа сите точки во множеството. Потоа го пополнува конвексниот труп со оние на сликата на маската, користејќи ја функцијата cv2.fillConvexPoly.  
  
  
- Ја транспонира сликата на маската да има три канали, што одговараат на RGB боите. Ова се прави со креирање листа од три копии од сликата на маската и потоа користење numpy.transpose за замена на оските.  
  
  
  
  
- Применува Гаусовиот филтер за замаглување на сликата на маската, користејќи ја функцијата cv2.GaussianBlur. Ова ги измазнува рабовите на маската и создава ефект на градиент. Филтерот има големина на јадрото од 11 на 11 пиксели и стандардна девијација од нула. Филтерот се применува двапати: еднаш за сите канали и еднаш за само еден канал. Ова создава пореален ефект на маска.  
  
  
- Ја враќа сликата на маската како излез од функцијата.  
  
  
-Целта на овој код е да создаде маска за лице што може да се користи за различни апликации, како што се замена на лица, препознавање лице, усогласување на лица итн. Маската за лице помага да се изолира областа на лицето од позадината и другите предмети на сликата . Исто така, помага да се зачуваат цртите на лицето и изразите на оригиналното лице.

A person's face in a black background

Description automatically generated with low confidenceA white triangle on a black background

Description automatically generated with medium confidence

*Слика 4. Резултат од користење на функцијата*

*Слика 5. Резултат од користење на функцијата со комбинирање на слика*

A screen shot of a computer

Description automatically generated with medium confidence  
  
Овој код е функција која зема слика, матрица за порамнување и посакувана форма на слика како влезови и враќа нова слика која е искривена според матрицата за порамнување и ја има саканата форма. Функцијата ја користи библиотеката OpenCV за обработка на слики.  
  
  
-Првата линија од кодот дефинира нова променлива наречена new\_img, која е иницијализирана како нули низа од нули со ист тип на податоци како влезната слика и обликот наведен со параметарот img\_shape.  
  
  
-Втората линија од кодот ја повикува функцијата cv2.warpAffine, која применува афина трансформација на влезната слика користејќи ја матрицата за порамнување. Афината трансформација може да изврши скалирање, ротација, преведување и стрижење на сликата. Функцијата зема шест аргументи: влезната слика, матрицата за порамнување (се користат само првите два реда), големината на излезната слика (широчина и висина), дестинационата слика (која е new\_img во овој случај), режимот на границата и знамињата.  
  
  
Режимот на граница одредува како да се пополнат пикселите надвор од трансформираната слика. Во овој случај, cv2.BORDER\_TRANSPARENT значи дека пикселите не се пополнети и остануваат транспарентни.  
  
  
Знамињата одредуваат како да се интерполираат пикселите во трансформираната слика. Во овој случај, cv2.WARP\_INVERSE\_MAP значи дека инверзната матрица за порамнување се користи за мапирање на дестинациските пиксели на изворните пиксели, што е побрзо и попрецизно од мапирањето на изворните пиксели на одредишните пиксели.  
  
  
-Третата линија од кодот ја враќа променливата new\_img како излез од функцијата.

2.5. Корекција на боја

A screenshot of a computer program

Description automatically generated with medium confidenceСо применување на барањата 2.1, 2.2, 2.3 и 2.4, резултатот би бил следен:

Функцијата swap\_images зема две патеки за слика како влез и враќа слика на која лицето на img1 е заменето со лицето на img2. Функцијата ги користи следните чекори:  
  
  
- Ги чита img1 и img2 со помош на функцијата read\_image, која враќа нумпирана низа што ја претставува сликата.  
  
  
- Ги добива обележјата на img1 и img2 со помош на функцијата get\_landmarks, која користи библиотека dlib за откривање 68 обележја на лицето на слика и враќа список со нивните координати.  
  
  
- Ја пресметува матрицата за порамнување со користење на функцијата transformation\_from\_points со листата whole\_face\_points, која ги содржи сите 68 индекси на обележје.  
  
  
- Ја добива маската за лице на img1 и img2 со користење на функцијата get\_face\_mask со нивните соодветни обележја.  
- Ја искривува маската на лицето на img2 користејќи ја функцијата warp\_image со матрицата за порамнување и обликот на img1 како целна форма.  
  
  
- Ги комбинира маската за лице на img1 и искривената маска за лице на img2 со користење на функцијата numpy.max, која враќа низа со максимална вредност по секоја оска. Така се создава маска која ги покрива двете лица.  
  
  
- Го искривува img2 користејќи ја функцијата warp\_image со иста матрица за порамнување и целна форма како претходно.  
  
  
- Ги спојува img1 и искривениот img2 користејќи аритметички операции со нула. Го множи img1 со (1.0-combined\_mask\_of\_img1) за да ги задржи само позадинските пиксели на img1. Го множи искривениот img2 со комбиниран\_маска\_на\_img1 за да ги задржи само пикселите на лицето на искривениот img2. Ги додава овие два резултати заедно за да ја A screenshot of a person

Description automatically generated with medium confidenceсоздаде конечната слика.

*Слика 6. Резултат без користење на корекција на боја*

A picture containing text, screenshot, font, software

Description automatically generatedЗа да биде замената попрецизна се применува функцијата correct\_colors\_of\_images() која ќе ги подобри боите и светлините на двете слики.

Потребни се три параметри: img1, img2 и landmarks\_of\_image1. Еве што прави секоја линија на код:  
  
  
- Првата линија ја пресметува количината на заматување врз основа на растојанието помеѓу левото и десното око обележје на сликата1. Ова се прави со земање на средната вредност на координатите на обележјата за секое око и наоѓање на Евклидовата норма (должина) на векторот меѓу нив. Количината на заматување потоа се множи со 0,7 за да се намали малку.  
  
  
- Втората линија ја претвора количината на замаглување во цел број, бидејќи Гаусовата функција за замаглување бара цел број како влез. Исто така, осигурува дека количината на заматување е непарна, бидејќи Гаусовата функција за замаглување бара непарен број како влез. Ако количината на заматување е рамномерна, таа додава една на неа.  
  
  
- Третата и четвртата линија го применуваат Гаусовиот филтер за заматување на img1 и img2 соодветно, користејќи ја количината на заматување како големина на јадрото. Големината на јадрото одредува колку соседните пиксели се во просек заедно за да се создаде заматен пиксел.  
  
  
  
  
- Петтата линија додава 128 на која било вредност на пиксели во img2\_blur што е помала или еднаква на 1. Ова е направено за да се избегне делење со нула или многу мали броеви во следната линија, што би резултирало со многу големи или бесконечни вредности.  
  
  
- Шестата линија враќа нова слика која се добива со множење img2 со img1\_blur и делење со img2\_blur. Оваа операција ги прилагодува боите на img2 за да одговараат на оние на img1, додека ги зачувува деталите за img2.  
A screen shot of a computer

Description automatically generated with medium confidence  
  
  
Доколку се примени следнава функција резулатот би бил следен:  
  
A picture containing human face, person, smile, person

Description automatically generated

*Слика 6. Резултат со користење на корекција на боја*

3. Користени алгоритми низ сите функции

-**Откривање лице**: Функцијата get\_landmarks го користи објектот на детекторот за откривање лица на сликата. Специфичниот алгоритам за откривање лице што се користи зависи од тоа како е дефиниран или увезен објектот на детекторот.  
  
  
-**Откривање знаменитости на лицето**: Функцијата get\_landmarks исто така го користи објектот за предвидување за откривање обележја на лицето на сликата. Специфичниот алгоритам за откривање обележје на лицето што се користи зависи од тоа како е дефиниран или увезен објектот за предвидување.  
  
  
-**Разложување на еднина вредност (SVD**): Функцијата transformation\_from\_points користи SVD за да ја пресмета матрицата на ротација која порамнува две групи на ориентирни точки.  
  
  
-**Делонеј триангулација**: Овој алгоритам не се користи експлицитно во ниту една од дадените функции, но може да се користи од функцијата cv2.convexHull повикана во функцијата get\_face\_mask за пресметување на конвексното тело на множество точки.  
  
  
-**Гаусово замаглување**: Функциите get\_face\_mask и correct\_colors\_of\_images користат Гаусово заматување за замаглување на слики користејќи ја функцијата cv2.GaussianBlur од библиотеката OpenCV.  
  
  
-**Афино искривување**: Функцијата warp\_image користи афино искривување за искривување на слика според матрицата за трансформација користејќи ја функцијата cv2.warpAffine од библиотеката OpenCV.  
  
  
-**Корекција на боја**: Функцијата correct\_colors\_of\_images ги приспособува боите на една слика за да одговараат на оние на друга користејќи едноставен алгоритам за корекција на боја.

A screenshot of a cellphone

Description automatically generated with low confidence4. Графички кориснички интерфејс  
  
-GUI е кратенка за графички кориснички интерфејс, кој е начин на интеракција со компјутерска програма користејќи графички елементи како што се прозорци, копчиња, менија, икони итн. Tkinter е модул во Python кој обезбедува едноставен и лесен начин за креирање GUI апликации. Tkinter се заснова на Tk, кој е вкрстен пакет алатки што работи на различни оперативни системи како што се Windows, Linux и Mac OS.   
  
  
-Tkinter обезбедува различни графички контроли, како што се етикети, записи, текстуални кутии, рамки, платна итн., кои може да се користат за прикажување и манипулирање со податоци, како и за справување со внесување и настани од корисникот. Tkinter, исто така, поддржува различни менаџери за распоред, како што се пакет, мрежа и место, што може да се користат за уредување на графичките контроли на екранот.

*Слика 7. Изглед на демо апликацијата*

5. Користена Литература (References)

1. [Face Swapping: Realistic Image Synthesis Based on Facial Landmarks Alignment (hindawi.com)](https://www.hindawi.com/journals/mpe/2019/8902701/)
2. [Swapped face detection using deep learning and subjective assessment | EURASIP Journal on Information Security | Full Text (springeropen.com)](https://jis-eurasipjournals.springeropen.com/articles/10.1186/s13635-020-00109-8)
3. [Face Swapping: Realistic Image Synthesis Based on Facial Landmarks Alignment (hindawi.com)](https://www.hindawi.com/journals/mpe/2019/8902701/)
4. [[Guide] Extraction - A Workflow - Faceswap Forum - The place to discuss Faceswap and Deepfakes](https://forum.faceswap.dev/viewtopic.php?t=27)
5. [What is Face Detection? Ultimate Guide 2023 + Model Comparison (learnopencv.com)](https://learnopencv.com/what-is-face-detection-the-ultimate-guide/)
6. [Face detection with dlib (HOG and CNN) - PyImageSearch](https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/)
7. [Training a custom dlib shape predictor - PyImageSearch](https://pyimagesearch.com/2019/12/16/training-a-custom-dlib-shape-predictor/)
8. [How to Apply HOG Feature Extraction in Python - Python Code (thepythoncode.com)](https://www.thepythoncode.com/article/hog-feature-extraction-in-python)
9. [numpy.linalg.svd — NumPy v1.25 Manual](https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html)
10. [Face Alignment with OpenCV and Python - PyImageSearch](https://pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/)
11. [Orthogonal Procrustes problem - Wikipedia](https://en.wikipedia.org/wiki/Orthogonal_Procrustes_problem)
12. [cv2.fillConvexPoly() vs. cv2.fillPoly() fill polygon\_cv2.fillpoly function\_rice ear blog-CSDN blog](https://blog.csdn.net/u012135425/article/details/84983265)
13. [Draw Polylines, fillPoly, fillConvexPoly and ArrowedLine in OpenCV - Python Geeks](https://pythongeeks.org/draw-polylines-fillpoly-fillconvexpoly-and-arrowedline-in-opencv/)
14. [OpenCV Smoothing and Blurring - PyImageSearch](https://pyimagesearch.com/2021/04/28/opencv-smoothing-and-blurring/)
15. [Python OpenCV - Affine Transformation - GeeksforGeeks](https://www.geeksforgeeks.org/python-opencv-affine-transformation/)
16. [OpenCV - How to do affine conversion in warpAffine - pystyle](https://pystyle.info/opencv-wrap-affine/)
17. [Automatic color correction with OpenCV and Python - PyImageSearch](https://pyimagesearch.com/2021/02/15/automatic-color-correction-with-opencv-and-python/)
18. [Face detection - Wikipedia](https://en.wikipedia.org/wiki/Face_detection)
19. [Face Detection Algorithms: The Ultimate Guide [2023] (banuba.com)](https://www.banuba.com/blog/face-detection-algorithms-guide/)
20. [Facial Landmarks: Use Cases, Datasets, and a Quick Tutorial (datagen.tech)](https://datagen.tech/guides/face-recognition/facial-landmarks/)
21. [Singular value decoџmposition - Wikipedia](https://en.wikipedia.org/wiki/Singular_value_decomposition)
22. [Singular Value Decomposition (SVD) - GeeksforGeeks](https://www.geeksforgeeks.org/singular-value-decomposition-svd/)
23. [Delaunay triangulation - Wikipedia](https://en.wikipedia.org/wiki/Delaunay_triangulation)
24. [Gaussian blur - Wikipedia](https://en.wikipedia.org/wiki/Gaussian_blur)
25. [Gaussian Blur: What Is It And Its Difference To Lens Blur (aiseesoft.com)](https://www.aiseesoft.com/resource/gaussian-blur.html)
26. [Python GUI Programming With Tkinter – Real Python](https://realpython.com/python-gui-tkinter/)
27. [Introduction to GUI With Tkinter in Python | DataCamp](https://www.datacamp.com/tutorial/gui-tkinter-python)
28. [Tkinter Tutorial (pythontutorial.net)](https://www.pythontutorial.net/tkinter/)