УНИВЕРЗИТЕТ У НИШУ
ЕЛЕКТРОНСКИ ФАКУЛТЕТ
Катедра за рачунарство

ПРАЋЕЊЕ РЕГУЛАРНОСТИ ПОЛАГАЊА ТЕСТОВА ПРИМЕНОМ ТЕХНИКА РАЧУНАРСКОГ ВИДА

- ДИПЛОМСКИ РАД -

Задатак:

Упознати се са постојећим софтверским решењима која се користе за праћење регуларности онлајн полагања тестова. Идентификовати технике рачунарског вида које се могу искористити за реализацију оваквих система. У практичном делу имплементирати прототип система који употребом камере обезбеђује препознавање особе која полаже тест, врши детекцију нерегуларности у виду окретања и разговора, те генерише извештај са видео доказом спорних ситуација.

Ментор: проф. др Александар Милосављевић **Кандидат:** Ивана Миливојевић 16704

Комисија:

1. _____ Датум пријаве: _____

2. _____ Датум предаје: _____

3. _____ Датум одбране: _____

Ниш, 2022.

Садржај

1. Увод

Пандемија настала услед ширења вируса Ковид-19 утицала је на све аспекте живота људи, широм света. Када је у питању образовање, било је неопходно прилагодити начин извођења наставе како би се школска година одвијала несметано. Образовне институције су морале да пређу на модел учења на даљину, те је било потребно пронаћи и прави метод оцењивања ученика.

Онлајн тестирање је данас у великој мери заступљено у пракси, пошто олакшава и убрзава процес креирања тестова, полагања, као и анализу резултата. Флексибилност и могућност рада од куће јесу још неки од разлога због којих велики број наставника бира овакве тестове као меру оцене. Наравно, тиме се отвара простор за различите начине „варања“. Неке од могућности које се притом могу применити јесу: ограничење да сви морају полагати тест истовремено, генерисање већег броја различитих питања, приказ питања и понуђених одговора у различитом редоследу, увођење временског ограничења за одговор за свако питање, онемогућавање враћања на претходно питање... Међутим, ове мере некада нису довољне, јер ученици могу да пронађу начин да комуницирају са другом особом или потраже одговоре у својој литератури и на интернету. Из тог разлога направљени су софтвери за надгледање полагања онлајн тестирања. Ови софтвери, поред примене у образовним институцијама, имају примену и у различитим организацијама (за тестирање током интервјуа за посао, реализацију семинара, добијање сертификата[1]...)

У оквиру система за надгледање полагања најчешће се прате микрофон, камера и екран ученика. Једна од могућности је да дежурна особа током полагања надгледа ученике уживо; друго решење подразумева снимање полагања и касније прегледање тих снимака ради контроле регуларности. Предност оваквог решења је у томе што ученици и дежурна особа не морају бити присутни у заказано време, али, у овом случају у реалном времену не постоји могућност реаговања на неправилности због којих је можда требало санкционисати неког ученика или онемогућити му наставак полагања теста. Треће решење јесте аутоматско нагледање полагања коришћењем алгоритама вештачке интелигенције, које у великој мери мења дежурну особу и не захтева њену константну присутност. Систем приказује упозорења ученику током полагања (при чему се може алармирати дежурна особа да се прикључи полагању) и бележи све спорне ситуације које се касније могу прегледати како би се потврдиле неправилности [1].

Рад се састоји из пет поглавља. Након уводног дела, у другом поглављу је дат преглед постојећих софтверских решења за проблем праћења регуларности онлајн тестирања, са фокусом на аутоматско надгледање. У трећем поглављу дат је преглед техника рачунарског вида које се у овим софтверима могу применити за обраду података са камере. Прво су обрађене технике и алгоритми који се могу применити за детекцију објеката, на које се надовезују методе за детекцију и препознавање лица. Након тога су описани начини за одређивање положаја главе, праћење погледа и детекцију причања на видео-снимку. У наредном поглављу описана је конкретна имплементација једноставног прототипа система који се може користити за аутоматско надгледање полагања тестова употребом камере. Систем врши препознавање лица и детектује нерегуларности које се могу јавити током полагања, а то су најчешће окретање и причање. Поред тога, детектује се одсуство особе у кадру или присуство више особа, коришћење мобилног телефона, и гледање са стране. Притом, за време теста приказују се упозорења ученику и све детектоване спорне ситуације бележе у видео-фајл. У закључном поглављу осврћемо се на претходна поглавља и сумирамо теоријске и практичне аспекте описаног проблема.

[1] У наставку ће се, за потребе овог рада, особа која полаже тест ословљавати као: ученик.

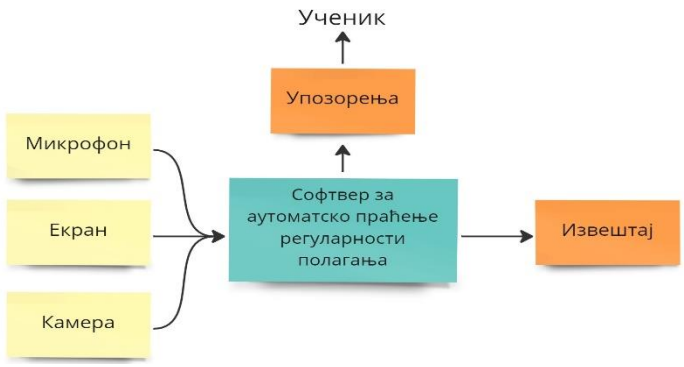
2. Софтвери за праћење регуларности полагања тестова

Постоји велики број софтвера који су развијени у циљу омогућавања учења на даљину (при чему неки од њих укључују полагање тестова), а доступни су и софтвери намењени само реализацији тестирања. Често је реч о апликацијама којима се приступа преко веб-претраживача, али има и апликација које је потребно инсталирати на свом уређају. Као најважнији захтеви које је потребно да ти софтвери испуне могу се издвојити: једноставност коришћења, сигурност осетљивих података (као што су сами тестови и лични подаци корисника), скалабилност, прилагодљивост и једноставна интеграција са постојећим платформама за учење. Скалабилност има важну улогу када је у питању употребљивост система, зато што може бити потребно да и више хиљада корисника истовремено полаже тестове. Систем треба да буде прилагодљив зато што корисници апликације имају различите потребе, и треба им обезбедити могућност избора скупа функционалности за реализацију конкретног тестирања [2].

Када су у питању озбиљнији тестови, чији интегритет треба очувати, треба одабрати неки од напреднијих софтвера за тестирање који нуде и праћење регуларности полагања (у зависности од потреба и буџета организације). Софтвери који врше праћење регуларности полагања тестова проширују могућности платформи за учење (и тестирање). У уводном поглављу је напоменуто да можемо разликовати три групе ових софтвера (за надгледање уживо, снимање полагања и аутоматско надгледање). Софтвери који врше аутоматско праћење регуларности не захтевају ангажовање дежурне особе током тестирања, а омогућавају и детаљнију анализу резултата (који се потом могу искористити за унапређење процеса тестирања), па се све више користе у пракси.

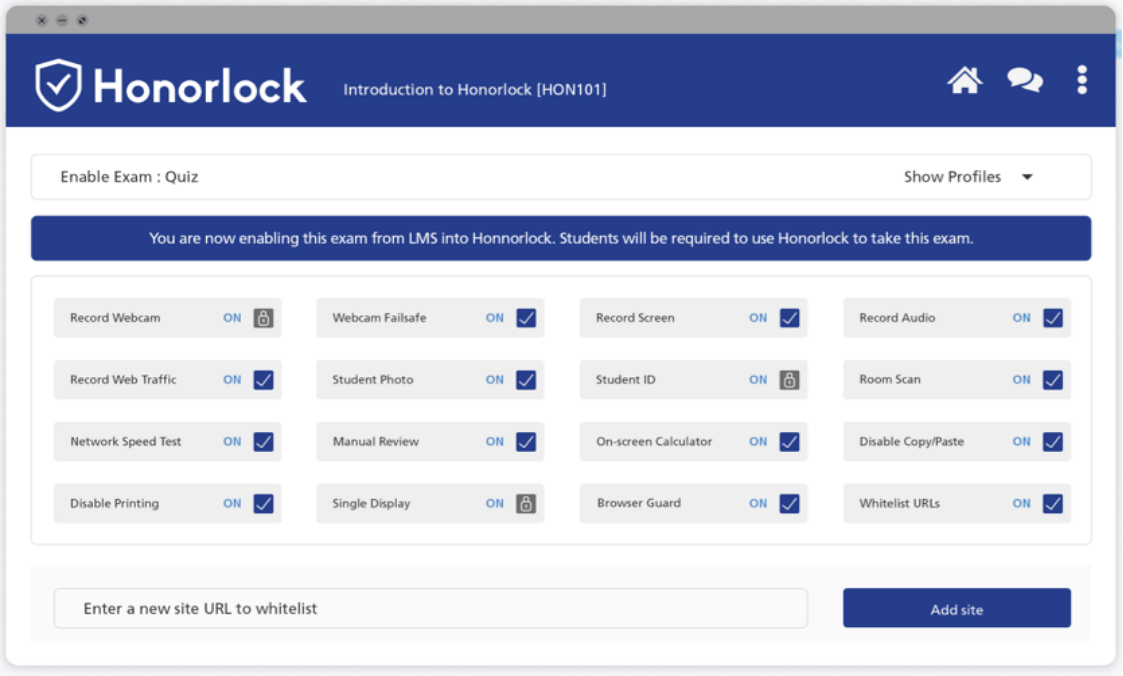
2.1. Аутоматско праћење

У оквиру софтвера за аутоматско надгледање полагања најчешће постоји више модула задужених за обраду података из различитих извора, као што су микрофон, камера и екран ученика (слика 1). Модул за праћење аудио података захтева да ученик има повезан микрофон и обезбеђену тишину у просторији у којој ће се радити тест. Може се испитати да ли ученик разговара са неким, а уколико тест укључује гласовну активност, може се проверити да ли је ученików глас исти током трајања теста, као и да ли се поклапа са његовим гласом из базе података. Неки системи притом детектују одређене кључне речи за које се процењује да би их ученици могли користити ако желе да потраже помоћ. Модул задужен за праћење екрана најчешће онемогућава сликање и снимање прозора у коме је отворен тест, забрањује напуштање теста пре предаје, као и отварање других прозора, апликација или фајлова. Неки од система омогућавају да наставници означе веб-сајтове којима ученици могу приступити током полагања. Модул за обраду података са камере захтева да камера буде укључена током полагања теста како би се контролисало да ли у кадру има других особа или недозвољених предмета (попут мобилних телефона, књига и папира). Овај модул врши и препознавање лица, прати покрете ученика и детектује када се он окреће или гледа са стране. Софтвер за аутоматско надгледање полагања, уз помоћ вештачке интелигенције, може додатно вршити онлајн претрагу, како би утврдио да ли је садржај теста „процурео“, при чему може предузети и кораке за његово уклањање. Систем на крају полагања генерише одговарајући извештај, који се касније може прегледати и анализирати. Извештај најчешће садржи листу спорних ситуација, заједно са пратећим снимцима [3].



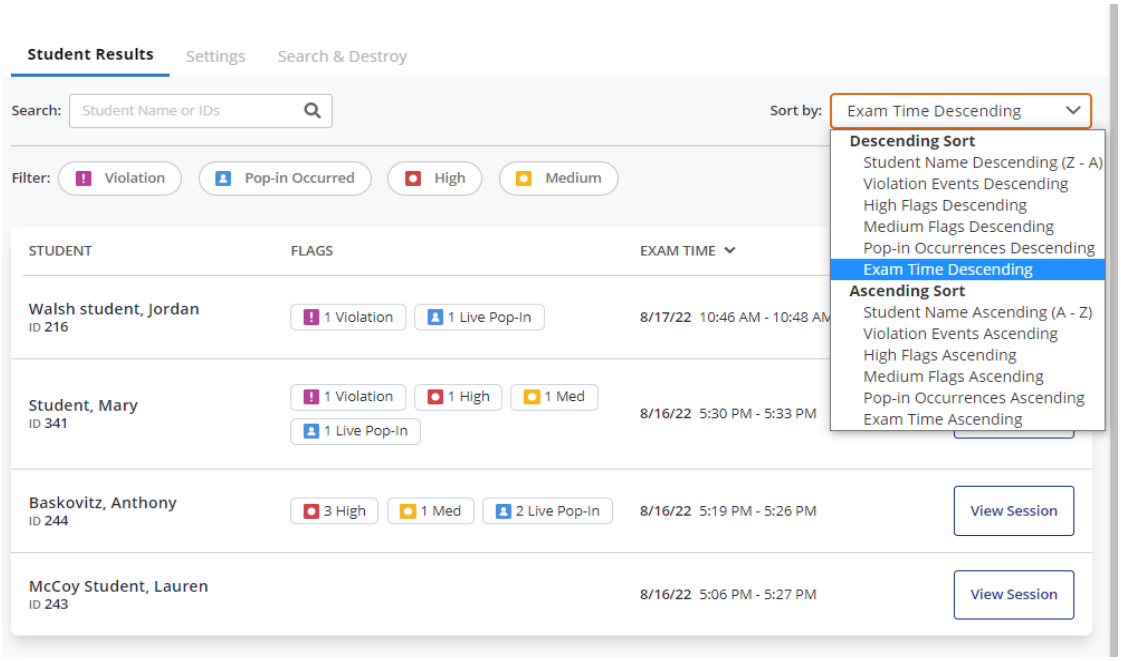
Слика 1: Аутоматско надгледање полагања

Honorlock, један од првих сервиса за праћење онлајн тестирања, комбинује аутоматско праћење регуларности са људским надгледањем – по потреби се може алармирати дежурна особа да се прикључи полагању [4]. Предуслов за коришћење Honorlock-a је преузимање екстензије за веб-претраживач (подржан је само Google Chrome), а предуслови за полагање теста су да особа буде сама у просторији, да рачунар на коме се ради тест има повезан само један монитор, и да корисник поседује 360° камеру како би се скенирала просторија у којој се налази. Овај систем користи софтвер за закључавање веб-претраживача који онемогућава приступ другим веб-сајтовима, при чему закључава и одређене пречице на тастатури – нпр. за копирање/лепљење (енг. copy/paste) и сликање екрана, и онемогућава минимизирање прозора и напуштање теста. Такође, систем идентификује процурели садржај теста на интернету и предузима кораке за његово уклањање. Још једна функционалност која је овде уведена јесте да систем детектује ако ученик покуша да приступи материјалу за учење током теста путем неког другог уређаја и бележи снимак екрана током трајања приступа. Овај систем не користи биометријске методе током полагања, већ пре почетка теста (коришћењем камере) услика лице ученика и његову идентификациону картицу. Ученицима је омогућено да путем чета контактирају надлежне уколико имају неких проблема или питања, а током трајања теста приказују им се упозорења ако су прекршили неко од дефинисаних правила. На слици 2 је приказано како се бирају функционалности за одређени тест.



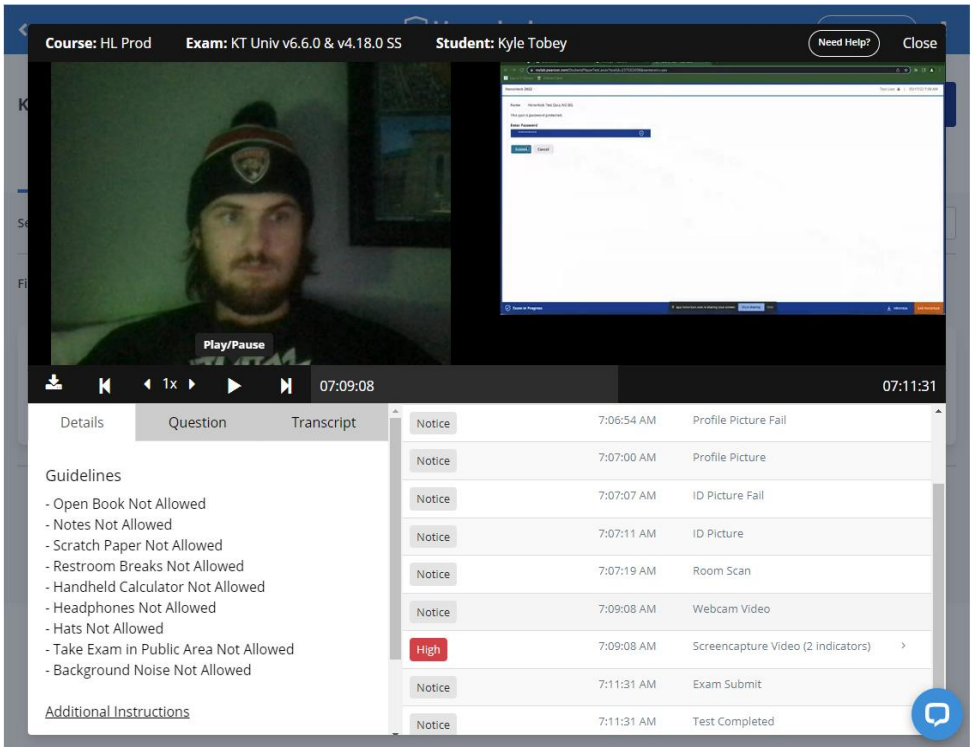
Слика 2: Одабир функционалности система за надгледање полагања [4]

На слици 3 се може видети како изгледа извештај који се односи на један тест. Приказује се списак свих ученика који су полагали тај тест, заједно са подацима о томе колико је спорних ситуација детектовано и који је ниво значаја тих ситуација. Такође, приказује се и информација о томе када је полагање обављено и колико је трајало.



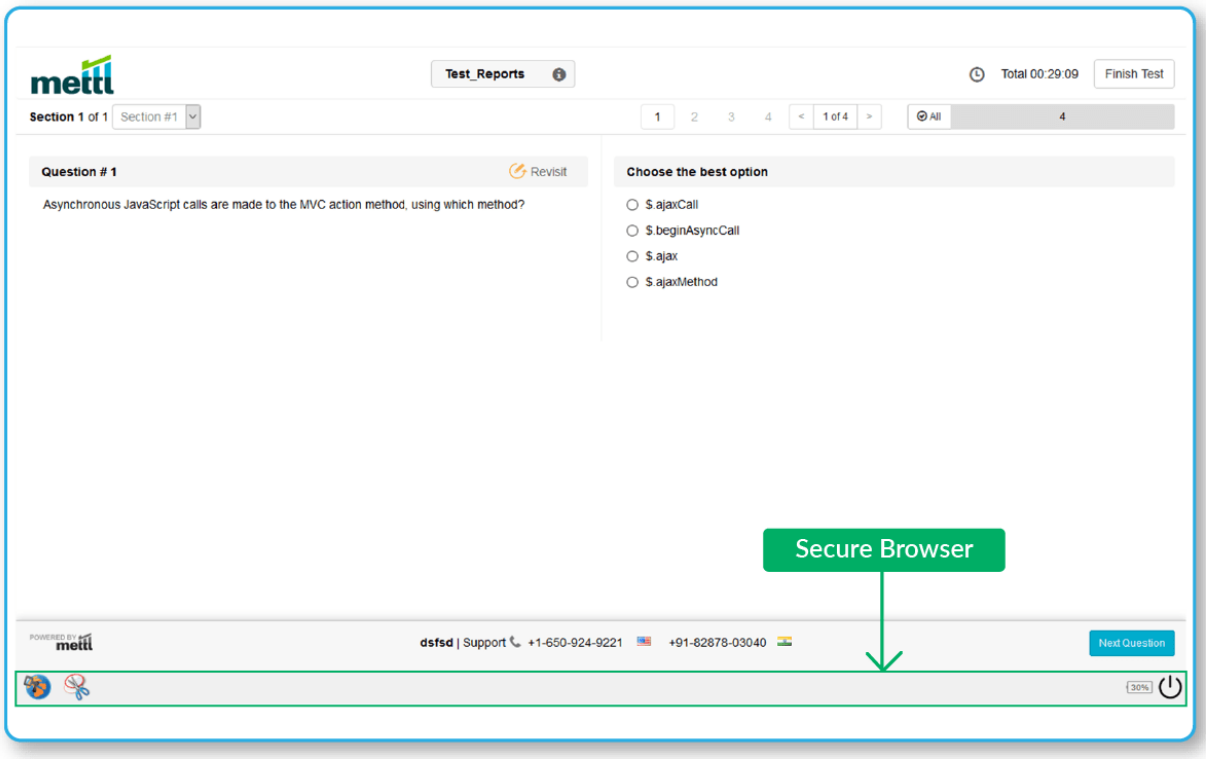
Слика 3: Извештај о резултатима тестирања [4]

Кликом на дугме „View Session“ отвара се нови прозор, који приказује детаље о спорним ситуацијама за конкретног ученика (слика 4). У левом делу се приказује снимак са камере ученика, у десном снимак екрана, а испод ових снимака налази се листа значајних догађаја који су се десили током полагања.



Слика 4: Детаљи везани за полагање конкретног ученика [4]

Mercer | Mettl нуди још једно од решења које омогућава безбедно тестирање, а може се користити као комплетна платформа за тестирање, или као сервис за удаљено надгледање којим се проширује нека платформа за тестирање [5]. Систем детектује уколико кандидат није присутан или је присутна особа која није она којом се представља (током трајања теста се у одређеним временским интервалима врши препознавање лица), детектује присуство мобилних телефона, као и разговор са другим особама. Ненадгледани тестови се могу полагати и на уређајима попут мобилних телефона и таблета, али тестове који захтевају већу сигурност неопходно је полагати на лаптоп или десктоп рачунарима. Подржани су претраживачи: Chrome, Firefox, Safari и Microsoft Edge. Још једна од функционалности овог система јесте могућност праћења локације са које се полаже тест. Систем је обучен да детектује до 18 типова нерегуларних ситуација. У понуди је и безбедни претраживач Mettl Secure Browser (приказан на слици 5), који искључује све екстерне портове и не дозвољава повезивање са другим уређајима, блокира недозвољене акције попут копирања и штампања садржаја теста, онемогућава напуштање теста, као и приступање веб-претраживачима и недозвољеним апликацијама.



Слика 5: Полагање теста уз коришћење безбедног претраживача [5]

3. Технике рачунарског вида за праћење регуларности полагања тестова

Рачунарски вид (енг. computer vision) је област рачунарства чији је основни циљ обука рачунара да постигну висок ниво разумевања дигиталних слика и видео-снимака, при чему се тежи аутоматизацији задатака које може да обави човеков визуелни систем. Као научна дисциплина, рачунарски вид се бави теоријом вештачких система који издвајају корисне информације из слика [6].

Када је реч о апликацијама за праћење полагања онлајн тестова, технике рачунарског вида можемо применити у оквиру модула који обрађује податке са камере. Неке од техника које се могу применити су: детекција објеката, детекција и препознавање лица, одређивање положаја главе, праћење погледа и детекција причања особе која полаже тест.

3.1. Детекција објеката

Детекција објеката је рачунарска техника која спада у област рачунарског вида и бави се проналажењем објеката који припадају одређеним класама на дигиталним сликама и видео-снимцима. Свака класа објеката има специфичне особине – фичере (енг. features), који одређују припадност објекта тој класи. Поред информације о класи објекта, као резултат се добија и информација о његовој локацији на слици [7]. Детекција објеката има широку примену у свакодневном животу (нпр. у саобраћају: за проналажење пешака, аутомобила и саобраћајних знакова), а један од најпознатијих специфичних случајева детекције објеката јесте детекција лица.

Методе за детекцију објеката се могу поделити на оне које се заснивају на традиционалним техникама обраде слика и на оне које су базиране на дубоком учењу. Код метода које су засноване на традиционалним техникама издвајају се фичери објеката са слике коришћењем неког од детектора фичера, а након тога се користе технике за класификацију. Технике базиране на дубоком учењу се ослањају на конволуционе неуронске мреже. Поступак детекције код њих се може реализовати у оквиру једне фазе, или кроз две фазе. Код двофазних детектора прва фаза јесте проналажење произвољног броја региона слике у којима се потенцијално налазе објекти, а друга је класификација сваког од њих, уз одређивање коначног правоугаоника којим се објекат може уоквирити. Једнофазни детектори предвиђају правоугаонике за уоквиравање објеката без претходно издвојених региона од интереса, третирајући детекцију објеката као регресиони проблем. Регресија оквирних правоугаоника (енг. bounding box regression) је кључни концепт у предвиђању њихових координата. Ови детектори су бржи и структурно једноставнији, али су и нешто мање поуздани од двофазних. Раније (традиционалне) методе за детекцију могу бити ограничене услед постојања комплексне позадине, делимично прекривених објеката и лошег осветљења, док су технике дубоког учења значајно отпорније на наведене проблеме. Дубоке конволуционе неуронске мреже у комбинацији са убрзањем које пружа графички процесор дају веома добре резултате и омогућавају детекцију објеката у приближно реалном времену. Недостатком ових метода може се сматрати потреба за значајним хардверским ресурсима и великим бројем тренинг слика које треба ручно означити (односно на којима треба обележити објекте) [8].

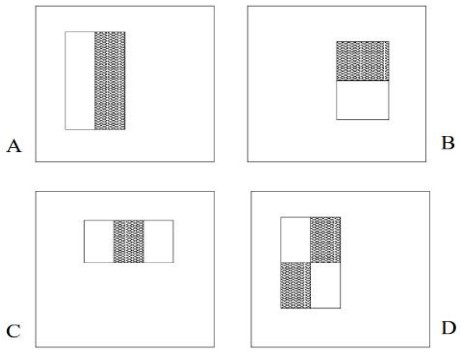
Када је у питању евалуација детектора објеката, најпопуларнији референтни скупови података су COCO и PASCAL VOC. Метрике које се најчешће користе су Frames Per Second (FPS) и mean Average Precision (mAP). FPS представља број обрађених фрејмова у секунди, а mAP средњу просечну прецизност, чије се рачунање мало разликује за COCO и PASCAL VOC, тако да при упоређивању детектора треба узети у обзир над којим референтним скупом је ова мера рачуната. У општем случају, просечна прецизност (AP) се рачуна за сваку класу објеката на основу графика зависности прецизности (енг. precision) од комплетности (енг. recall). Прецизност се рачуна као: $\frac{TP}{TP+FP}$, а комплетност као: $\frac{TP}{TP+FN}$, при чему TP представља број тачних позитивних предикција, FP број нетачних позитивних предикција, а FN број нетачних негативних предикција. После тога се налази mAP као средња вредност израчунатих AP [9]. Тежи се што већој mAP мери и што већем FPS.

У наставку ћемо дати преглед неколико детектора објеката који су били важни за развој ове области, почевши од Viola-Jones и HOG детектора, који се заснивају на традиционалним техникама, до двофазних (различите верзије R-CNN-а) и једнофазних (YOLO и SSD) детектора, који користе конволуционе неуронске мреже.

3.1.1. Viola-Jones детектор

Viola и Jones (у раду *Rapid Object Detection using a Boosted Cascade of Simple Features* [10] из 2001. године) представљају метод за детекцију објеката у реалном времену реализован кроз примену неколико нових техника. Овај метод је првобитно био намењен детекцији лица, али се може користити и за детекцију других типова објеката. Viola-Jones детектор се често назива и „Haar cascade“, зато што се за детекцију објеката користе касакада класификатора и скуп фичера који подсећају на Haar-ове таласе.

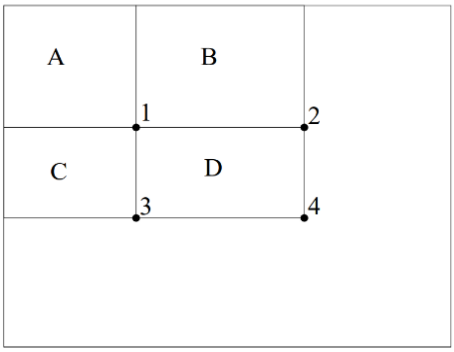
Аутори користе скуп фичера приказан на слици 6. Притом, дефинисане су три врсте фичера: са два (A и B), три (C) и четири (D) суседна правоугаоника. Вредност фичера се рачуна тако што се сума пиксела у белим регионима одузима од суме пиксела у сивим. Ови фичери омогућавају проналажење ивица и линија на слици.



Слика 6: Haar фичери [10]

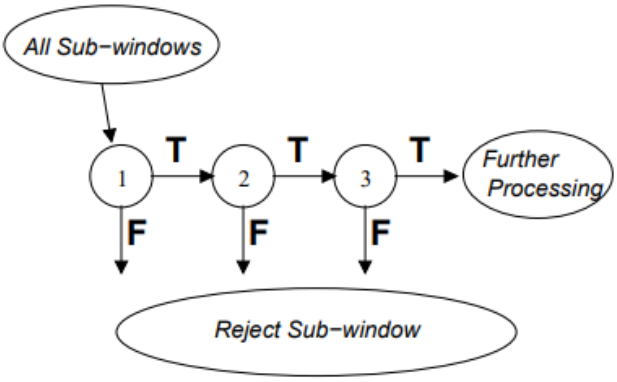
У раду је приказан нов начин за репрезентацију слике, назван „интегрална слика“, који омогућава брзо рачунање фичера. Интегрална слика се рачуна једним проласком кроз оригиналну слику и може се посматрати као матрица истих димензија као оригинална слика, с тим што на локацији (x, y) садржи суму пиксела оригиналне слике изнад и лево од тачке (x, y). Оваква репрезентација омогућава израчунавање суме пиксела произвољног правоугаоног региона коришћењем вредности у само четири тачке, односно применом само три операције сабирања/одузимања. На слици 7 се може приметити да сума пиксела у оквиру правоугаоника D може бити израчуната на основу

вредности интегралне слике у тачкама 1, 2, 3 и 4. Вредност интегралне слике у тачки 1 је сума пиксела у правоугаонику A. Вредност у тачки 2 је: A + B, у тачки 3 је: A + C, а у тачки 4 је: A + B + C + D. Сума пиксела у оквиру правоугаоника D би била: 4 + 1 - (2 + 3). За два правоугаоника сума пиксела може бити израчуната на основу вредности у 6 тачака, за три правоугаоника на основу 8, а за случај четири правоугаоника на основу 9 тачака. Када се интегрална слика једном израчуна, фичери било које величине на било којој локацији могу се израчунати за константно време.



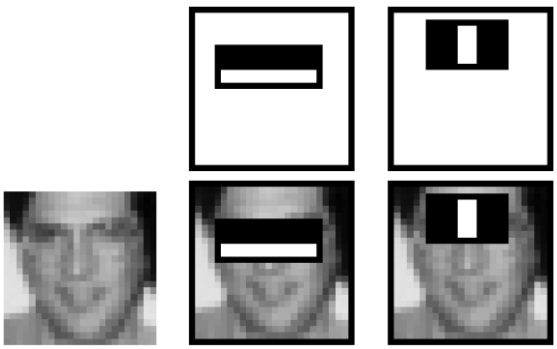
Слика 7: Рачунање суме пиксела у оквиру правоугаоног региона [10]

За прозор димензија 24 × 24 се израчунавају фичери (око 160 000 фичера различитих величина и позиција). Иако се појединачни фичери добијају прилично једноставно и брзо, рачунање комплетног скупа фичера је захтевно и скупо. Да би издвојили мањи скуп најважнијих фичера из великог скупа фичера и тренирали класификатор, Viola и Jones су користили алгоритам заснован на AdaBoost (Adaptive Boosting) техници. Свака фаза „бустинг“ (енг. boosting) процеса даје нови слаби класификатор, који зависи само од једног фичера. Овим се број фичера смањује на око 6000. AdaBoost алгоритам комбинује више слабих класификатора у јак (добар) класификатор. За детекцију објеката је коришћена каскада јаких класификатора који се примењују један за другим, при чему је сваки класификатор у каскади комплекснији од претходног. На слици 8 је илустрован процес класификације. Ако прозор не прође први класификатор (не садржи објекат), одбацује се и не обрађује даље. Ако прође, пропушта се кроз наредни класификатор и процес се понавља. Прозор који прође све класификаторе сматра се прозором који садржи објекат. Циљ је брзо одбацити регионе без објеката, а даљу обраду вршити само над регионима који су обећавајући, односно онима који потенцијално садрже објекат.



Слика 8: Каскада класификатора [10]

Детекција објеката је демонстрирана на примеру детекције лица. На слици 9 приказана су прва два фичера селектована AdaBoost алгоритмом. Први фичер наглашава особину лица да је регион око очију углавном тамнији од региона горњег дела образа, док други фичер осликава особину да је регион очију тамнији од региона носа. Сваки од класификатора у каскади од 38 слојева трениран је фронталним сликама лица, које су скалиране на резолуцију 24 × 24 (како би се поклопиле са прозором) и сликама на којима нема лица, са којих су издвајани подпрозори 24 × 24. Број фичера у првих пет слојева је 1, 10, 25, 25 и 50, респективно. У процесу детекције детектор обилази слику која се тестира у више величина методом клизајућег прозора, при чему се детектор (прозор) скалира, а не слика, како би се искористиле предности интегралне слике. С обзиром на то да ће бити генерисано више детекција за исто лице, потребно је на крају те детекције свести на једну. Све детекције се раздвајају у непреклапајуће подскупове, при чему ће се две детекције наћи у истом подскупу ако им се оквирни региони преклапају. Сваки подскуп даће једну коначну детекцију, чије су границе просек граница свих детекција у том подскупу [10].



Слика 9: Прва два фичера селектована AdaBoost техником [10]

Viola-Jones детектор ради са сивим сликама, детектује објекте на сликама независно од њихове локације и величине, и инваријантан је на промене у осветљењу. Једна од највећих предности овог детектора јесте велика брзина детекције, а недостатак представља склоност лажним позитивним детекцијама и детектовање једино фронтално постављених објеката. Иако је објављен пре много година (2001), он се и даље често користи за детекцију лица.

3.1.2. HOG детектор

У раду *Histograms of Oriented Gradients for Human Detection*, објављеном 2005. године [11], Dalal и Triggs су показали да се за детекцију објеката на слици могу успешно користити Histogram of Oriented Gradients (HOG), односно хистограм оријентисаних градијената, и линеарни SVM класификатор. Аутори су се фокусирали на детекцију пешака на слици, али се детектор може применити и за детекцију објеката других класа.

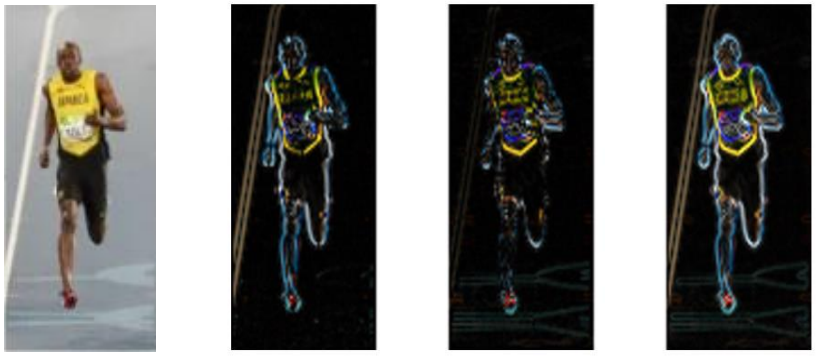
Фичер дескриптор је репрезентација слике у којој су издвојене корисне информације из ње, а одбачене ирелевантне. HOG фичер дескриптор је вектор дужине 3780, при чему се као фичери користе хистограми смерова оријентисаних градијената. Градијенти слике (x и y изводи) су корисни, зато што је њихова магнитуда велика око ивица и ћошкова (региони наглих промена интензитета) – који садрже много више информација о облику објекта него равни региони. Аутори су дескриптор фичера рачунали над регионима слике величине 64 × 128. Хоризонтални и вертикални

градијенти слике се могу наћи филтрирањем слике Собел филтером, а магнитуда и угао финалног градијента се рачунају по следећим формулама:

$$G = \sqrt{{G_x}^2 + {G_y}^2}, \quad (1)$$

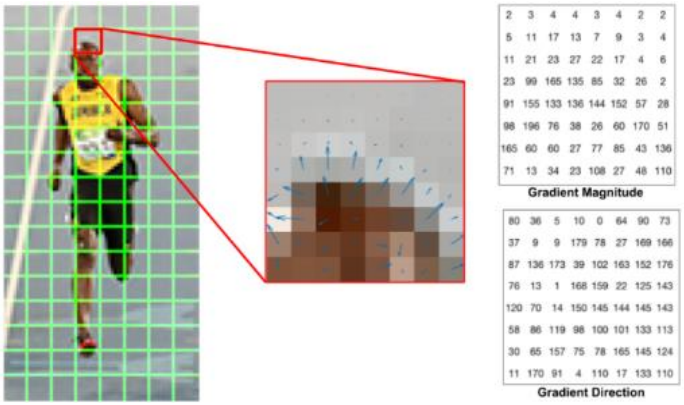
$$\theta = \tan^{-1}\frac{G_y}{G_x}, \quad (2)$$

где су G_x и G_y градијенти по *x* и *y* осама. У левом делу слике 10 приказана је слика особе која трчи, у средишњем делу су приказани *x* градијент (који издваја вертикалне линије) и *y* градијент (који издваја хоризонталне линије), док је финална слика приказана десно. Особа је у првом плану, а већина неважних информација је уклоњена. Код слика у боји рачунају се градијенти за сва три канала, па се за магнитуду у сваком пикселу узима максимална вредност магнитуда канала, а за угао онај који одговара тој магнитуди.



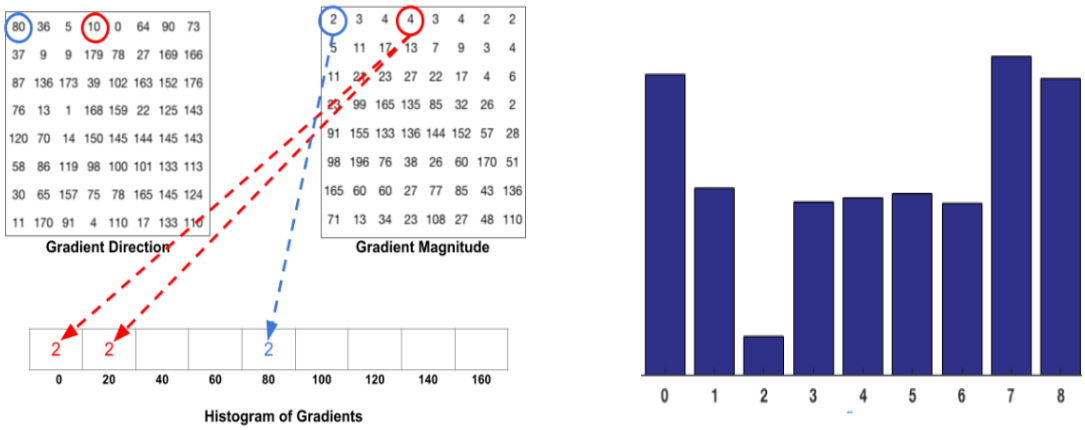
Слика 10: Део улазне слике (лево), његови *x* и *y* градијенти (средина) и финална магнитуда градијената (десно) [12]

Слика се затим дели на ћелије од 8 × 8 пиксела, при чему за сваки пиксел постоје две вредности (магнитуда и смер градијента), што даје: 8 * 8 * 2 = 128 вредности. У левом делу слике 11 приказана је подела слике на ћелије, а у средини је издвојена једна ћелија (чији су смерови градијената илустровани смером стрелица, а магнитуде дужинама стрелица). Смерови стрелица показују смер промене интензитета, а њихова дужина колико је велика разлика. У десном делу слике 11 су бројевима представљене вредности магнитуда и смерова градијената за пикселе издвојене ћелије. Углови се посматрају у односу на *y*-осу и у опсегу од 0° до 180° уместо од 0° до 360° (зато што су посматрани „неозначени“ градијенти).



Слика 11: Подела слике на ћелије (лево), градијенти ћелије представљени стрелицама (средина), градијенти ћелије представљени бројевима (десно) [12]

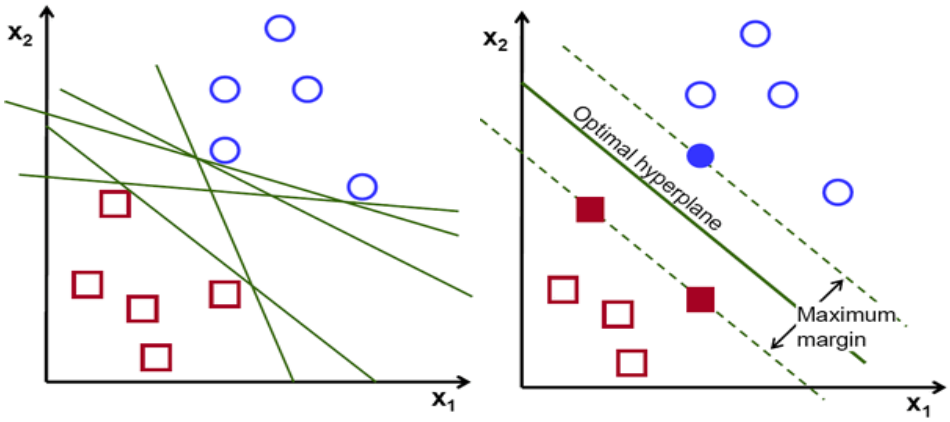
Индивидуални градијенти пиксела могу садржати шум, али хистограм над ћелијом пиксела је много отпорнији на шум и притом даје компактнију репрезентацију слике. Хистограм је низ од 9 елемената (приказан на слици 12 у левом делу), који одговарају угловима од 0°, 20°, 40°, …, 160°. При рачунању хистограма, елемент низа се бира на основу смера (угла) градијента. Градијент означен плавом бојом има угао од 80° и магнитуду 2, тако да се вредност 2 додаје елементу на петој позицији. Градијент означен црвеном бојом има угао од 10°, који је на средини између 0° и 20°, тако да се његова вредност 4 додаје пропорционално првом и другом елементу (по 2). Ако је угао већи од 160°, он је заправо између 160° и 180°, а како су углови од 0° и 180° еквивалентни, његова вредност би била пропорционално придружена елементима који одговарају угловима од 0° и 160°. У десном делу слике 12 је приказано како изгледа добијени хистограм. Како у хистограму доминирају вредности близу 0° тј. 180°, може се закључити да ова ћелија вероватно садржи ивицу (градијенти су усмерени на горе или доле). Да би дескриптор био инваријантан на промене у осветљењу, треба извршити нормализацију хистограма. Посматра се прозор од 16 × 16 пиксела, који садржи четири хистограма – нормализација се врши над вектором од 36 елемената који се формира надовезивањем ових хистограма. Прозор се затим помера за 8 пиксела и процес се понавља све док се не обиђе цела слика. Да би се израчунао коначан вектор фичера, нормализовани вектори дужине 36 се надовезују и креирају вектор од 3780 елемената (36 * 105 = 3780, где је 7 * 15 = 105 број позиција на којима може да се нађе прозор). Добијени вектор се даље користи за тренирање SVM класификатора [12].



Слика 12: Процес рачунања хистограма градијената (лево),
график хистограма (десно) [12]

Support Vector Machine (SVM), односно машина са векторима подршке, има за циљ проналажење оптималне хиперравни (границе одлуке) у вишедимензионалном простору, која раздваја податке различитих класа. Маргина представља најкраће растојање од хиперравни до најближег тренинг податка који припада било којој класи. Уколико није могуће у потпуности поделити податке, треба наћи хипер раван која максимизује маргину, односно минимизује грешке у класификацији. На слици 13 је у левом делу приказано више могућих хиперравни у 2D простору (димензионалност простора је одређена бројем фичера), а у десном оптимална хипер раван. Очекује се да хипер раван са већом маргином буде прецизнија приликом класификације непознатих података у односу на хипер раван са мањом маргином [13]. Подаци који се налазе на самим маргинама називају се „вектори подршке“ (енг. support vectors) – они се најтеже класификују, али дају највише података о самој класификацији. У случају скупа података који није линеарно сепарабилан, може се извршити трансформација тог скупа у вишедимензионални простор, где ће подаци бити линеарно сепарабилни. Како се не би

експлицитно примењивала функција трансформације над сваким податком, користе се кернел функције („кернел трик“) – које обезбеђују нелинеарну границу у оригиналном простору.



Слика 13: Могуће хиперравни (лево), оптимална хиперраван (десно) [13]

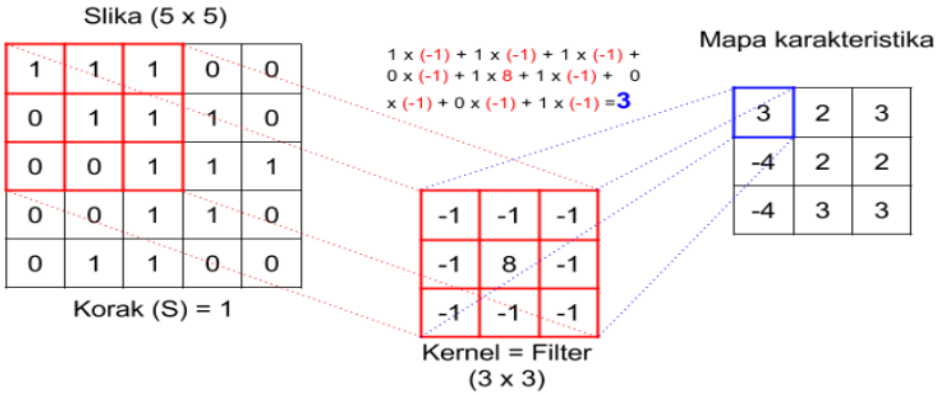
HOG фичер дескриптор у комбинацији са SVM класификатором може се користити за детекцију објеката у реалном времену, па налази примену и данас: даје добре резултате и инваријантан је на промене у осветљењу, али није инваријантан на промене у ротацији и углу гледања (због чега може детектовати само објекте који су постављени приближно фронтално). Како би се детектовали објекти различитих величина, примењују се пирамида слика и метода клизајућег прозора.

3.1.3. R-CNN детектор

Region-based Convolutional Neural Network (R-CNN) детектор, објављен 2014. године (Girshick и сар. у раду *Rich feature hierarchies for accurate object detection and semantic segmentation* [14]), један је од првих модела који за детекцију објеката примењује дубоко учење, а може се користити и за семантичку сегментацију. Заснива се на конволуционим неуронским мрежама, које се по својој архитектури могу сврстати у дубоке неуронске мреже. Ове мреже имају велику примену у домену рачунарског вида, зато што су пројектоване за рад са 2D структурама (као што су слике).

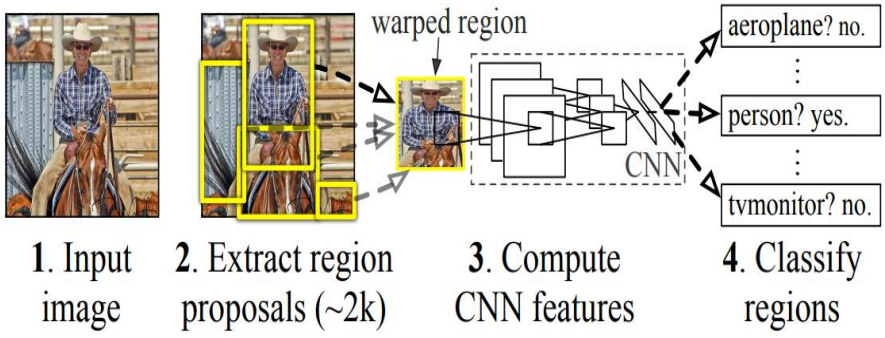
Convolutional Neural Networks (CNNs), односно конволуционе неуронске мреже, јесу вештачке неуронске мреже које су добиле име по конволуцији (оператору који се примењује у обради слика за изоштравање и замућење, као и за детекцију ивица). Конволуциони филтери (кернели) се примењују на слике како би се издвојили фичери и креирале њихове мапе, при чему се може вршити и редукција резолуције слика. Филтер се представља дводимензионалном матрицом малих димензија (најчешће 3 × 3), која се примењује у конволуционом слоју мреже на слику која се обрађује у том слоју. На слици 14 може се видети пример примене 3 × 3 филтера са кораком померања 1 на монохроматску слику димензија 5 × 5. У сваком кораку се вредности на истој позицији множе и сабирају са производима других парова у прозору, чиме се добија елемент мапе фичера. Треба имати у виду да се RGB слика представља као 3D структура (ширина × висина × дубина), при чему трећа димензија одговара броју канала, тако да ће и филтер имати трећу димензију. Архитектура мреже је таква да се на почетку налази улазни слој којим се слика уводи у мрежу; затим следи један или више конволуционих слојева, између којих се могу наћи слојеви сажимања (енг. pooling layer) – са циљем прогресивног смањења слике. Постоје и слојеви активационе функције (најчешће ReLU), који могу значајно побољшати перформансе мреже – често се убацују након сваког конволуционог

слоја. На крају мреже се налази један или више потпуно повезаних слојева (енг. fully connected layer) – неурони из потпуно повезаног слоја су повезани са свим неуронима из претходног слоја [15].

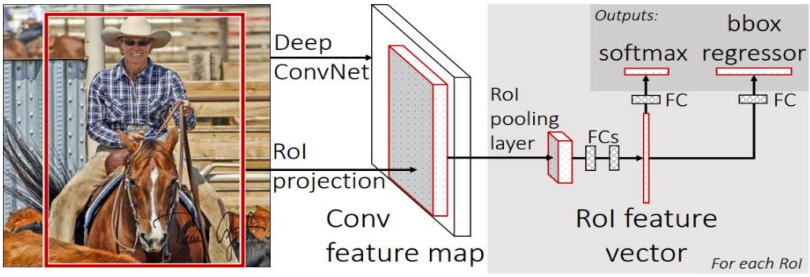


Слика 14: Пример примене конволуционог филтера [15]

На слици 15 је приказано како R-CNN модел функционише. Аутори овог детектора су направили систем за детекцију објеката који се састоји из три модула. Први модул применом селективног тражења издваја из улазне слике око 2000 региона, који представљају кандидате за детекцију објеката (алгоритам селективног тражења врши сегментацију слике на основу интензитета пиксела и генерише регионе груписањем мањих сегмената сличних по боји, текстури, величини и облику). Издвојени региони су правоугаоници за које постоји вероватноћа да садрже објекат (без информације о томе којој класи припада тај објекат). Други модул је конволуциона неуронска мрежа (позната као „AlexNet“) са пет конволуционих слојева и два потпуно повезана слоја, која издваја фичере (векторе фичера фиксне дужине од 4096 елемената) из сваког предложеног региона. Величина региона је пре пропуштања кроз мрежу промењена, тако да одговара улазу који мрежа очекује. Трећи модул је скуп SVM-а за појединачне класе објеката, уз помоћ којих се класификују региони. Да би се побољшале перформансе локализације, коришћена је регресија оквирних правоугаоника. Након тога, примењује се пост-процесирање, да се пречисте оквирни правоугаоници и елиминише вишак. Ово се постиже коришћењем технике потискивања немаксимума (енг. *non-maximum suppression*) за сваку класу посебно: правоугаоник се одбацује ако са правоугаоником који има већу поузданост детекције има већи IOU од одређеног прага („Intersection Over Union“ – IOU представља однос између пресека и уније два правоугаоника) [14]. Највећи недостатак R-CNN-а је у томе што процес тренирања и тестирања трају веома дуго.



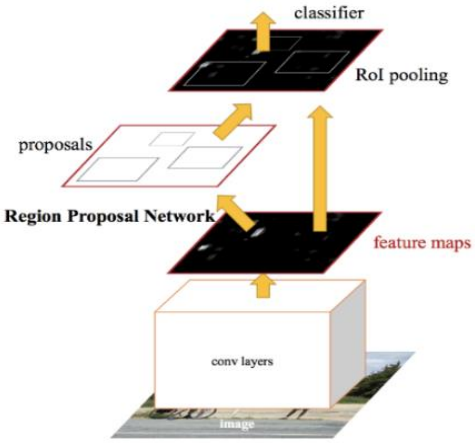
Слика 15: Начин функционисања R-CNN модела [14]



Слика 16: Архитектура Fast R-CNN-а [16]

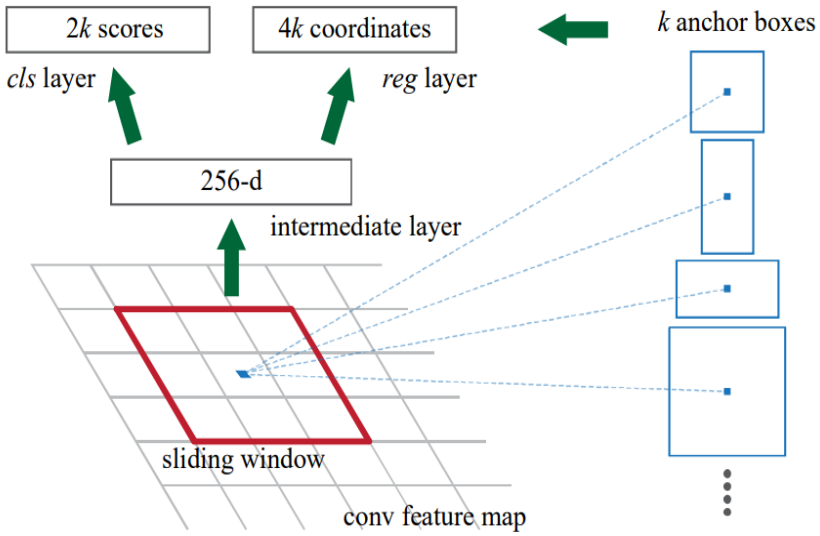
Girshick, један од аутора R-CNN-а, објављује 2015. године његово побољшање – Fast R-CNN [16], код кога тестирање (уз претходно издвојене регионе) захтева 0,32 секунде по слици уместо 47 секунди. Коришћена је притом VGG-16 неуронска мрежа, која има 13 конволуционих слојева и 3 потпуно повезана слоја. На слици 16 приказан је начин рада Fast R-CNN-а. Мрежа, која као улаз има комплетну слику и регионе генерисане селективним тражењем, на основу слике генерише конволуциону мапу фичера (генерише се једна мапа фичера, а не 2000). Затим, за сваки регион, RoI (Region of Interest) слој сажимања примењује сажимање максимумом и „пакује“ регион у малу мапу фичера фиксних димензија – она се прослеђује потпуно повезаним слојевима (FCs), који дају вектор фичера. Потпуно повезани слојеви се гранају у два излазна слоја: један који генерише *softmax* вероватноћу над K класа објеката (+1 за позадинску класу) и други, који генерише одступања од локација и димензија предложених региона за сваку од класа. Softmax функцијом рачуна се расподела вероватноћа припадања објекта класама, тако да свака вероватноћа буде из опсега [0, 1], а сума свих вероватноћа буде 1. Овај модел достиже mAP од 70% (док R-CNN има 66%) на PASCAL VOC 2007, али и даље троши 2 секунде на фазу предлагања региона.

Селективна претрага за проналажење региона је веома спора, односно представља уско грло у систему. Уместо тога, Ren и сар. су представили Faster R-CNN, у раду *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* [17], који користи дубоку конволуциону мрежу за предлагање региона и обавља комплетну детекцију за свега 0,2 секунде. На слици 17 се може видети како изгледа архитектура комплетног Faster R-CNN система. Један модул је Region Proposal Network (RPN) – мрежа која на основу конволуционе мапе фичера издваја предложене регионе. Ради се о потпуно повезаној конволуционој мрежи, која симултано предвиђа границе објеката и информацију о вероватноћи да тај део слике садржи објекат. Други модул је Fast R-CNN детектор, који користи предложене регионе. Ова два модула деле конволуционе слојеве – систем је јединствена неуронска мрежа за детекцију објеката.



Слика 17: Faster CNN детектор [17]

Принцип рада RPN је приказан на слици 18. Мрежа као улаз има n × n клизећи прозор над мапом фичера (у раду је узето n = 3). Сваки прозор се мапира у 256-димензиони фичер и затим прослеђује слоју за регресију (reg) и слоју за класификацију (cls). На свакој локацији клизећег прозора предвиђа се максимално k региона, тако да регресиони слој који даје координате региона има 4k излаза, а класификациони, који даје вероватноћу за присуство или одсуство објекта у региону, има 2k излаза. Аутори ове регионе посматрају релативно у односу на k референтних правоугаоника, названих „anchors“, који су центрирани у прозору и имају придружен фактор скалирања и однос ширина:висина (енг. aspect ratio). Подразумевано се генерише девет оваквих правоугаоника (користећи три фактора скалирања и три *aspect ratio*-а).



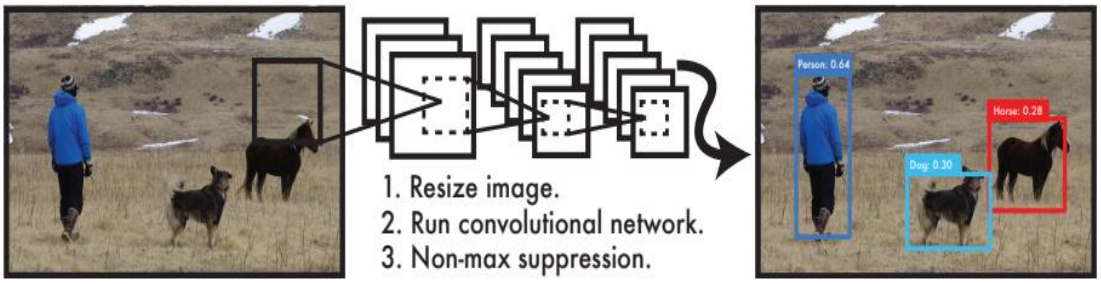
Слика 18: Region Proposal Network (RPN) [17]

Mask R-CNN [18] детектор из 2017. године је унапређење Faster R-CNN-а. Разлика између њих је у томе што Mask R-CNN додаје грану за предикцију маске објекта – маска која се рачуна на нивоу пиксела омогућава раздвајање објекта од позадине, примењујући сегментацију инстанци (енг. instance segmentation).

3.1.4. YOLO детектор

You Only Look Once (YOLO) детектор, презентован у раду Redmon-а и сар. *You Only Look Once: Unified, Real-Time Object Detection*, објављеном 2016. године [19], представљао је нови приступ проблему детекције објеката. Неуронска мрежа, инспирисана GoogLeNet моделом, предвиђа оквирне правоугаонике и одговарајуће вероватноће директно из комплетних слика – отуд и потиче назив детектора: You Only Look Once („погледај само једном“).

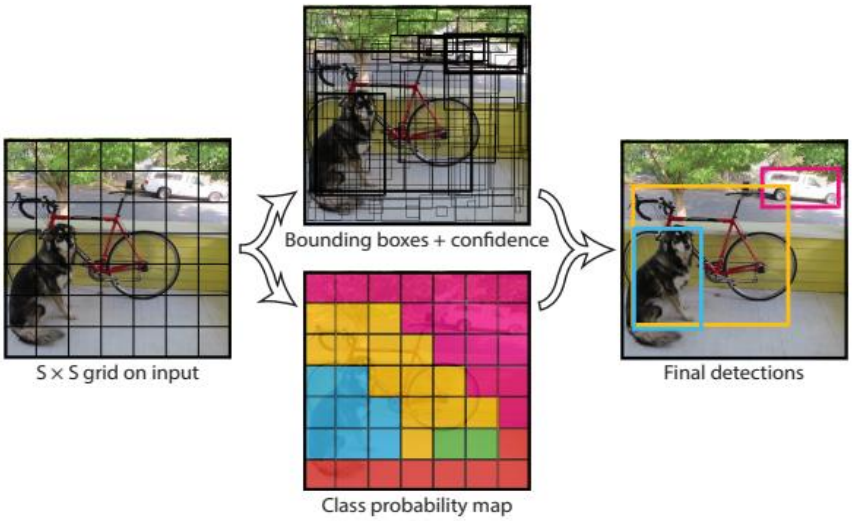
На слици 19 приказан је процес детекције YOLO детектором. Први корак је прилагођавање величине улазне слике, затим се таква слика пропушта кроз конволуциону неуронску мрежу, након чега се врши потискивање немаксимума, односно одбацивање оквирних правоугаоника који су вишак. Прво се елиминишу они који имају поузданост мању од задате границе, након тога се бира правоугаоник са највећом поузданошћу, па се одбацују они који имају преклапање (IOU) са њим веће од граничног. Процес се понавља све док сви правоугаоници не буду обрађени.



Слика 19: Процес детекције YOLO детектором [19]

Илустрација модела овог детектора приказана је на слици 20. Аутори су поделили улазну слику на грид димензија S × S: ако центар објекта упада у ћелију грида, та ћелија ће бити задужена за његову детекцију. Свака ћелија предвиђа B обухватајућих правоугаоника, при чему сваки обухватајући правоугаоник има пет предикција: x, y, w, h, и confidence. Координате (x, y) представљају центар правоугаоника релативно у односу на границе ћелије. Ширина и висина (w и h) представљене су релативно у односу на целу слику. Поузданост (confidence) се рачуна као $\mathrm{P(Object)} * \mathrm{IOU}$, где је $\mathrm{P(Object)}$ вероватноћа да правоугаоник садржи објекат, а IOU осликава тачност граница правоугаоника (рачуна се за предиктовани правоугаоник и стварни правоугаоник који означава где се објекат налази – „ground truth“). Свака ћелија грида предвиђа и C условних вероватноћа $\mathrm{P(Class_i|Object)}$. Ово су вероватноће појављивања објеката сваке од класа у грид ћелији – предвиђа се само један скуп ових вероватноћа по ћелији, независно од броја обухватајућих правоугаоника B. Њиховим множењем са претходно израчунатом поузданошћу за појединачни правоугаоник, добија се вероватноћа да се објекат класе *i* појављује у том правоугаонику, уз информацију о тачности граница правоугаоника:

$$\mathrm{P(Class_i|Object)} * \mathrm{P(Object)} * \mathrm{IOU} = \mathrm{P(Class_i)} * \mathrm{IOU}. \tag{3}$$



Слика 20: Модел YOLO детектора [19]

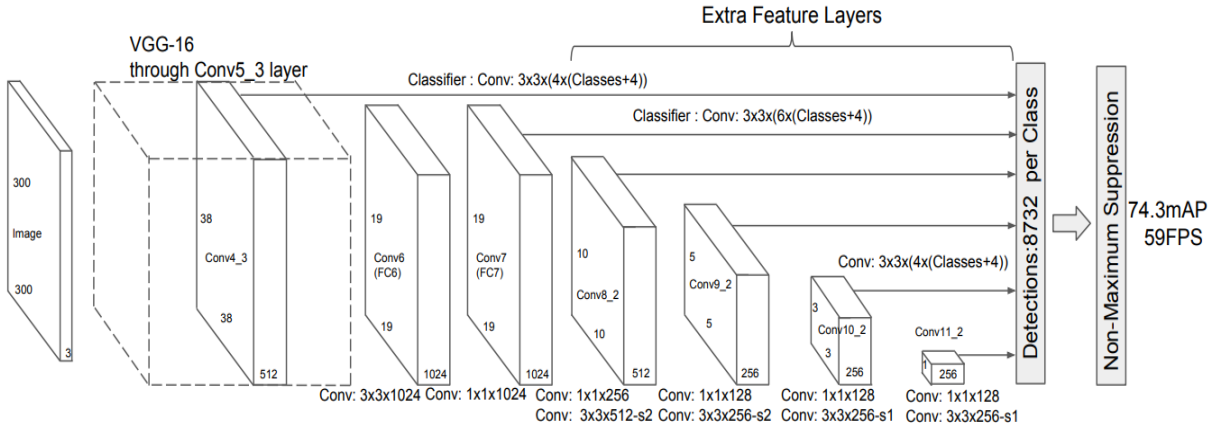
Предикције се кодирају као S × S × (B ∗ 5 + C) тензор, при чему су аутори користили S = 7, B = 2 и C = 20. Модел је имплементиран као конволуциона неуронска мрежа која има 24 конволуциона слоја и два потпуно повезана слоја, док бржа варијанта мреже има мање конволуционих слојева (девет) и мање филтера у тим слојевима. Иницијални конволуциони слојеви издвајају фичере из слике, док потпуно повезани слојеви предвиђају излазне вероватноће и координате правоугаоника. Модел обрађује 45 фрејмова у секунди достижући mAP од 63,4% на PASCAL VOC 2007, док Faster R-CNN достиже mAP од 73,2% уз обраду 7 фрејмова у секунди. Бржа варијанта YOLO модела (Fast YOLO) обрађује и до 155 фрејмова у секунди, уз mAP од 52,7% [19].

Архитектура овог детектора је била знатно бржа од оних које су нудили остали детектори који раде у реалном времену (као што је Faster R-CNN). Још неке од његових предности су то што генерише мало лажно позитивних предикција када нема ничега на слици и то што даје добре резултате при детекцији објеката и на уметничким сликама. Главни недостатак овог алгоритма је у томе што не детектује добро мале објекте и објекте који су блиско груписани (зато што дели слику на грид у коме је свака ћелија задужена за детекцију само једног објекта). У наредним годинама су објављиване новије верзије овог детектора, од којих је последња YOLOv7 (објављена ове године). YOLOv7 се сматра најбољим детектором у овом тренутку.

3.1.5. SSD детектор

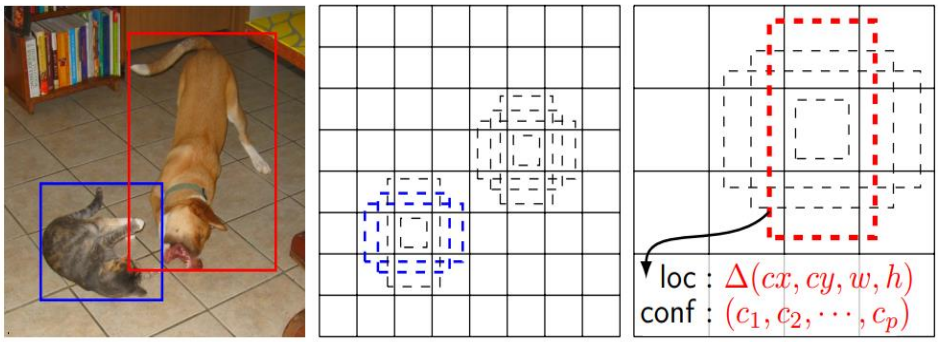
SSD детектор представили су Liu и сар. у раду *SSD: Single Shot MultiBox Detector* [20] 2016. године. Детекција објеката обавља се коришћењем дубоке конволуционе неуронске мреже, а модел је дизајниран тако да ради у реалном времену. Овај детектор уводи неке новине, као што су коришћење мапа фичера различитих резолуција и дефинисање подразумеваних правоугаоника (енг. default boxes) различитих величина и *aspect ratio*-а.

Коришћена архитектура детектора, приказана на слици 21, састоји се из два дела: први је заснован на VGG-16 мрежи, а други обухвата додатне конволуционе слојеве. Процес детекције подразумева издвајање мапа фичера и примену малих конволуционих филтера, како би се детектовали објекти. VGG-16 мрежа је одсечена пре класификационог слоја, а њени потпуно повезани слојеви 6 и 7 су конвертовани у конволуционе (Conv6 и Conv7). Затим следи скуп од шест додатних конволуционих слојева (од којих се пет користи за генерисање предикција), при чему се њихова резолуција прогресивно смањује (почетни слојеви доприносе детекцији мањих објеката, а крајњи детекцији већих објеката).



Слика 21: Архитектура SSD детектора [20]

На пример, у конволуционом слоју Conv4_3 мапа фичера се дели на 38 × 38 ћелија, при чему се за сваку од њих генеришу 4 подразумевана правоугаоника, а потом се за сваки од њих генерише по једна предикција, тако да се добија укупно: 38 * 38 * 4 предикција [21]. Предикција се састоји од предиктованог оквирног правоугаоника и 21 вредности, које означавају припадност свакој од класа (20 класа за објекте и једна класа за позадину). Предиктован оквирни правоугаоник је дефинисан релативно у односу на подразумевани правоугаоник, коришћењем четири вредности: (Δcx, Δcy, Δw, Δh), које представљају одступања у односу на центар (cx, cy), ширину (w) и висину (h) подразумеваног правоугаоника. Рачунање предикција се обавља применом 3 × 3 конволуционих филтера над сваком ћелијом. Сличан поступак се обавља у сваком конволуционом слоју, при чему се у неким слојевима генерише 4, а у неким 6 предикција. Са смањењем величине слојева смањује се и резолуција мапа фичера, што омогућава детекцију објеката различитих величина и значајно побољшава тачност детектора.



Слика 22: Слика са ручно означеним правоугаоницима (лево), 8 × 8 мапа фичера (средина), 4 × 4 мапа фичера (десно) [20]

SSD дефинише фактор скалирања за сваки од слојева. Комбиновањем ових фактора скалирања и жељеног *aspect ratio*-а, израчунавају се ширина и висина подразумеваних правоугаоника. На нивоу једног слоја користе се исти подразумевани правоуганици за сваку ћелију и они су центрирани у тој ћелији. Предикције се могу класификовати као позитивна и негативна поклапања. Ако одговарајући подразумевани правоугаоник има IOU већи од 0,5 са стварним правоугаоником, поклапање се сматра позитивним, а у супротном негативним. На слици 22 у средини може се приметити да се два подразумевана правоугаоника поклапају са правоугаоником којим је означена мачка, а десно да се један поклапа са правоугаоником у коме је пас. Ова три правоугаоника се сматрају позитивним преклапањима приликом тренирања, а остали негативним. Овде је притом приказан и пример како SSD детектује објекте различитих величина и облика. Пас је детектован у једном подразумеваном правоугаонику (црвеном) у слоју са 4 × 4 мапом фичера, али ни у једном у мапи фичера веће резолуције (8 × 8), док је мачка мања и детектована је коришћењем слоја са 8 × 8 мапом фичера (плави правоугаоници). На крају процеса детекције се користи потискивање немаксимума како би се уклониле предикције дупликати. Предикције се сортирају по поузданости и задржава се 200 најбољих [21].

Овај модел по перформансама надмашује Faster R-CNN модел, који је до тог тренутка (2016) сматран најбољим, достижући 74.3% mAP на PASCAL VOC 2007 тесту и 59 FPS. Један од недостатака SSD-а је у томе што даје слабије резултате за мале објекте (они могу бити детектовани само у слојевима највеће резолуције, који садрже мање значајне фичере). Тачност детекције модела расте са повећањем броја подразумеваних правоугаоника, али по цену брзине.

3.2. Препознавање лица

Када је реч о обради слика лица, прво треба дефинисати појам детекције лица, зато што је то процес који најчешће претходи било којој обради лица. Детекција лица је рачунарска техника која се користи за проналажење људског лица на дигиталним сликама, а односи се на испитивање да ли се на слици налази неко лице, где је оно лоцирано и које је величине [22]. Као најпознатији алгоритми који се користе за детекцију лица могу се издвојити Viola-Jones и HOG заједно са линеарним SVM (описани у претходном поглављу), Eigenfaces, као и методе засноване на неуронским мрежама. Алгоритми се разликују по понашању при екстремним условима – лоше осветљење, различити положаји лица, изрази лица, лица веома мале или велике резолуције, прекривеност лица (нпр. наочаре, коса, брада). Након детекције лица могу се издвојити карактеристичне тачке лица (енг. facial landmarks), које означавају истакнуте делове попут очију, обрва, носа, усана и образа. На пример, карактеристичне тачке се успешно примењују за поравнање лица, препознавање лица, одређивање положаја главе...

Препознавање (или идентификација) лица је процес утврђивања коме одређено лице са слике припада, односно упоређивање неког лица са познатим лицима из базе података у циљу проналажења поклапања [23]. Ово је процес који је у великој мери заступљен у свакодневном животу, при чему се не мора увек вршити упоређивање са свим лицима из базе података, јер је у неким применама потребно само дозволити приступ одређеној особи, а осталима забранити (као на пример код откључавања телефона). Поред 2D система, постоје и 3D системи за препознавање лица, који укључују додатне информације о облику лица у простору – они могу дати боље резултате него 2D системи. У циљу побољшања резултата препознавања лица често се врши поравнање лица, при чему неке методе посматрају 3D модел лица, а једноставније методе се ослањају само на карактеристичне тачке лица (конкретно, на тачке које одређују очи).

Код система за препознавање лица може бити проблема уколико неко покуша злонамерно да приступи систему представљајући се као неко други. Може се десити да неко подметне слику или видео друге особе и, уколико нема провере да ли је лице на камери реално („живо“), систем подметнута лица може препознати као валидна. Постоје различити приступи овом проблему, а неки од њих су: анализа текстура, анализа фреквенција (Фуријеов домен лица), анализа променљивих величина између узастопних фрејмова, алгоритми базирани на хеуристикама (покрети очију, усана, трептање), алгоритми оптичког тока (разлике у оптичким токовима генерисаним од стране 3D и 2D равни), посматрање 3D облика лица или комбинација наведених приступа [24].

Временом, најпре су били развијени системи који користе геометрију лица за идентификацију (позиција и величина очију, носа, образа и браде), затим системи који користе алгоритме машинског учења (издвајање фичера и тренирање класификатора), а у скорије време за препознавање лица све више се користе алгоритми дубоког учења. Један од историјски најзначајнијих алгоритама је Eigenfaces. Након њега су представљене методе Fisherfaces и Local Binary Patterns (LBP), које се и данас користе у многим апликацијама. Када је у питању дубоко учење, имплементиране су специјалне архитектуре које се називају „сијамске мреже“ (енг. siamese networks) – ове архитектуре садрже две или више идентичних неуронских мрежа, којима се прослеђују улази чију сличност треба одредити. FaceNet је један од најуспешнијих система за препознавање лица који примењује дубоко учење [25].

3.2.1. Eigenfaces метод

У раду *Face Recognition Using Eigenfaces* [26] из 1991. године, Turk и Pentland представили су Eigenfaces метод за детекцију и препознавање лица. Овај метод се заснива на техници линеарне алгебре за редукцију димензија простора, која се назива Principal Component Analysis (PCA), односно анализа главних компоненти.

Алгоритам најпре обрађује скуп од M тренинг слика лица које су сиве, исте величине и поравнате тако да се кључни делови лица што више поклапају. Посматране [27] су слике димензија N × N (у општем случају могу бити произвољне величине), које се линеаризују и представљају као вектор димензија N^2, односно као тачка у простору са N^2 димензија. Израчунава се средње лице, које је представљено вектором дужине N^2 (чији су елементи просечна вредност одговарајућих елемената вектора тренинг слика), након чега се свака тренинг слика представља као разлика оригиналне слике и слике средњег лица. Овим се заправо средња тачка (средње лице) помера у координатни почетак, како би се лакше израчунала коваријанса (варијанса представља одступање вредности тачака од средње вредности и рачуна се за сваку од димензија, а коваријанса представља меру повезаности димензија). Матрица коваријансе има димензије које одговарају броју димензија простора ($N^2 \times N^2$) и може се израчунати као $C = AA^T$, при чему је матрица A добијена смештањем вектора тренинг слика у колоне. У пресеку неке врсте и колоне матрице коваријансе налазиће се вредност коваријансе димензија које су везане за ту врсту и колону. Циљ примене PCA јесте пронаћи компоненте које најбоље описују дистрибуцију слика лица у тренинг скупу, односно *eigenvector*-е матрице коваријансе која одговара скупу тренинг слика. Eigenvector је вектор димензија N^2 и може се посматрати као замућено лице које називамо „eigenface“ или „својствено лице“. Рачунање матрице коваријансе може се ефикасније обавити множењем A^T и A, чиме се добија матрица C' димензија M × M, при чему важе следеће релације:

$$A^T A v_i = \lambda_i v_i \tag{4}$$

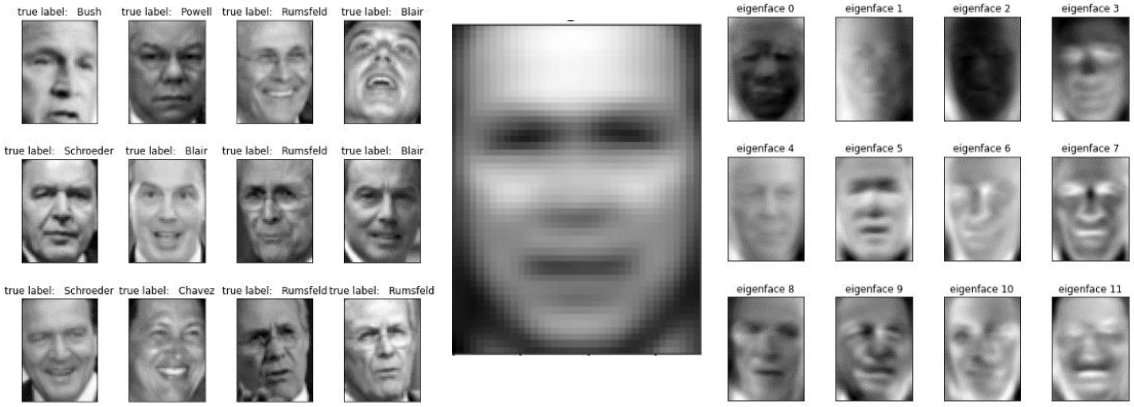
$$AA^T A v_i = \lambda_i A v_i \tag{5}$$

$$C u_i = \lambda_i u_i \tag{6}$$

где су λ_i својствене вредности (eigenvalues) обе матрице, v_i својствени вектори матрице C', а u_i својствени вектори матрице C. На основу матрице C' могу се израчунати вектори v_i, а затим вектори u_i (коришћењем једнакости $u_i = Av_i$).

Свако лице из тренинг скупа може бити представљено као линеарна комбинација K својствених лица, која најбоље репрезентују варијације међу лицима и описују K-димензиони простор – „простор лица“, при чему је K много мање од иницијалне димензионалности простора слика лица. Тренинг слике се могу представити вектором тежина које се јављају у линеарној комбинацији. На слици 23 лево су приказана тренинг лица, у средини је представљено средње лице, а десно је приказано првих 12 eigenface репрезентација – светлији региони одговарају већим варијацијама, а тамнији мањим. Када се анализира нова слика лица, она се модификује слично као тренинг слике, тако што се од ње се одузима средње лице, а затим се пројектује на K-димензиони простор као линеарна комбинација својствених лица. За потребе детекције лица, из K-димензионог простора се реконструише слика (тако што се средњем лицу дода сума производа својствених лица и одговарајућих тежина придружених пројектованој слици) и пореди са оригиналном сликом. У случају да је та разлика већа од граничне вредности, на слици нема лица, а у супротном је лице детектовано. Уколико је циљ препознавање

лица, онда се добијени K-димензиони вектор пореди са векторима добијеним пројекцијом тренинг слика (рачуна се еуклидско растојање – ближа лица су сличнија). Лице се на крају идентификује као оно лице са којим има најмање растојање [27].

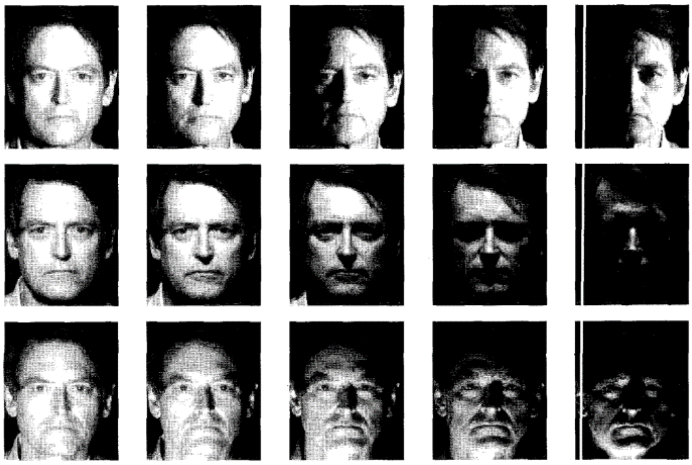


Слика 23: Тренинг лица (лево), средње лице (средина), eigenfaces (десно) [27]

Eigenfaces је био један од првих значајнијих алгоритама намењених препознавању лица. Предности овог алгоритма су у томе што није рачунски захтеван, брзо се извршава, и не захтева знање о карактеристикама лица. Један од недостатака овог алгоритма је то што захтева поравнање слика лица приликом тренирања и препознавања (ради се на нивоу пиксела, тако да је потребно да се лица скоро савршено поклапају). Такође, захтева фронтална лица, осетљив је на промене у положају и осветљењу, а на ефикасност препознавања може утицати и експресија лица.

3.2.2. Fisherfaces метод

Fisherfaces метод представљен је 1997. године у раду Belhumeur-a и сар. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection* [28]. Уз Eigenfaces, ово је један од најпознатијих алгоритама за препознавање лица – и он се заснива на редукцији димензија, али користи Linear Discriminant Analysis (LDA), односно линеарну дискриминантну анализу. Сваки пиксел на слици посматран је као тачка у високодимензионом простору и слика је линеарно пројектована у нискодимензиони потпростор који није осетљив на промене у осветљењу и експресији лица.



Слика 24: Слике лица при различитом осветљењу [28]

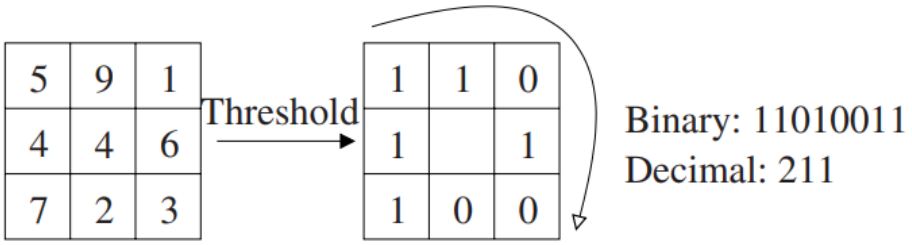
Eigenfaces користи пројекције које максимизују разлике између лица у целом скупу слика, што не даје добре резултате при препознавању када постоје промене у осветљењу, јер су веће разлике између слика истог лица при различитом осветљењу него између слика лица различитих особа (што се може уочити на слици 24). Fisherfaces користи пројекцију која ће максимизовати однос расподеле између класа и расподеле унутар класа. Тежи се максимизацији удаљености лица различитих класа, а минимизацији удаљености унутар сваке класе. Рачунају се матрице коваријансе између класа и унутар класа, а затим својствени вектори и својствене вредности – бирају се вектори са највећом својственом вредношћу. Као код Eigenfaces алгоритма, тренинг слике се пројектују у потпростор слика, улазна слика се такође пројектује у овај потпростор и упоређују се растојања [28].

Fisherfaces метод је дао боље резултате од Eigenfaces метода и није осетљив на велике варијације у осветљењу (које се односе на промену у интензитету, али и смеру и броју светлосних извора) и експресији лица. Један од недостатака овог алгоритма је што не подржава промене у пози и, слично Eigenfaces алгоритму, захтева да слике лица буду поравнате. Налажење пројекција је рачунски захтевније него код Eigenfaces, али је Fisherfaces метод поузданији.

3.2.3. LBP метод

Алгоритам који за препознавање лица користи Local Binary Patterns (LBP), односно локалне бинарне обрасце, приказан је у раду Ahonen-a и сар. *Face Recognition with Local Binary Patterns* [29], објављеном 2006. године. Овај метод узима у обзир информације о облику и текстури приликом представљања слике.

LBP оператор сваки пиксел на слици пореди са околним пикселима и резултат посматра као бинарну вредност. Пример примене овог оператора дат је на слици 25. Вредностима већим од 4 придружена је јединица, а мањим од 4 придружена је нула, након чега се добијена осмобитна вредност преводи у децимални број. Затим се хистограм ових вредности користи као дескриптор текстуре.



Слика 25: LBP оператор [29]

Први корак у алгоритму су превођење слике у сиву и подела на матрицу од 7 × 7 једнаких ћелија. Затим се за сваку од ћелија израчунава LBP хистограм фичера. Ако се посматра 8 суседних пиксела, биће: $2^8 = 256$ могућих вредности, те је хистограм дужине 256. Рачунањем хистограма за сваку ћелију заправо енкодирамо просторне информације као што су очи, нос и уста. Неки региони слике носе више информација, неки мање, тако да се уводе тежине које се придружују ћелијама. На слици 26 лево је приказано лице подељено на ћелије, а десно тежинска шема. Бели квадрати (очи) имају тежину 4 (њихови хистограми су помножени са 4), светлосиви тежину 2 (уста и уши), тамносиви 1 (унутрашњи образи и чело), док црни имају тежину 0 (нос и спољашњи део образа) – вредности тежина су експериментално утврђене. Тежински хистограми се надовезују и

формирају јединствени хистограм фичера који представља слику лица. Препознавање лица се врши упоређивањем растојања: улазно лице се обрађује као и тренинг лица (издвајају се LBP, додају им се тежине, надовезују се), а затим се примењује k-Nearest Neighbors (k-NN) алгоритам (k = 1), како би се нашло најближе лице из тренинг скупа [29].

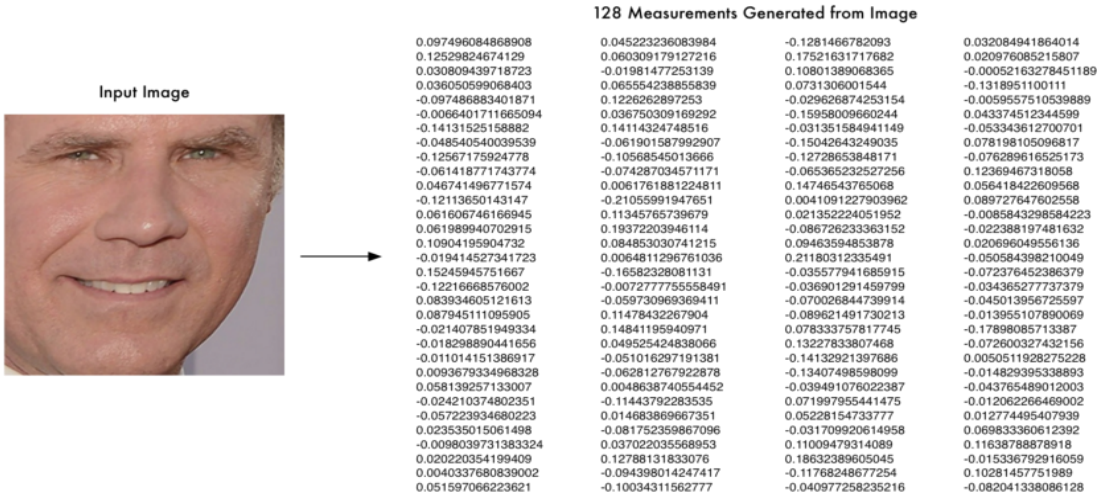
Слика 26: Слика лица подељена на ћелије (лево), тежинска шема за одговарајуће ћелије (десно) [29]

LBP представља локалну репрезентацију текстуре. Овај алгоритам може да се ажурира приликом додавања новог лица у систем, док многи алгоритми захтевају да сва лица која ће се идентификовати буду присутна у време тренирања. LBP алгоритам је отпорнији на шумове, јер не ради директно са интензитетима пиксела и даје боље резултате него Eigenfaces – ефикасан је и омогућава веома брзо издвајање фичера.

3.2.4. Метод дубоког учења

Дубоке конволуционе неуронске мреже могу се користити за препознавање лица са великом тачношћу. Један од најпознатијих система, назван FaceNet, представљен је у раду Schroff-a и сар. *FaceNet: A Unified Embedding for Face Recognition and Clustering* [30]. Мрежа је тренирана да генерише вектор од 128 елемената за свако лице, који се може користити при упоређивању лица. Мери сличности два лица директно одговарају растојања између њихових вектора. На широко коришћеном скупу слика Labeled Faces in the Wild (LFW), овај систем постиже тачност од 99,63%.

При препознавању лица, на почетку треба детектовати лице, издвојити карактеристичне тачке и поравнати га тако да очи и уста буду на истим позицијама на сликама, како би било олакшано упоређивање. За разлику од неуронских мрежа за детекцију објеката које се тренирају да би препознале објекат на слици, ове неуронске мреже се тренирају да генеришу 128 мера (низ реалних бројева) које карактеришу свако лице (слика 27). Идеја да се компликовани подаци (попут оних које носи једна слика) представе преко 128 бројева била је веома значајна за машинско учење – ови бројеви називају се „embeddings“. Тренирање мреже се обавља тако што се користе три различите слике (енг. triplets), при чему две припадају истој особи, а трећа некој другој особи. Тежине мреже се подешавају тако да слике исте особе имају врло сличне векторе, а слике те особе и неке друге што различитије. Овај поступак се понавља милионима пута за милионе слика, да би мрежа научила да поуздано генерише мере за сваку особу. Када се мрежа једном истренира, може се користити за генерисање вектора за било које лице, па и потпуно непознато. Да би се непознато лице препознало, треба пронаћи у бази познатих лица оно које има најближе мере [31].

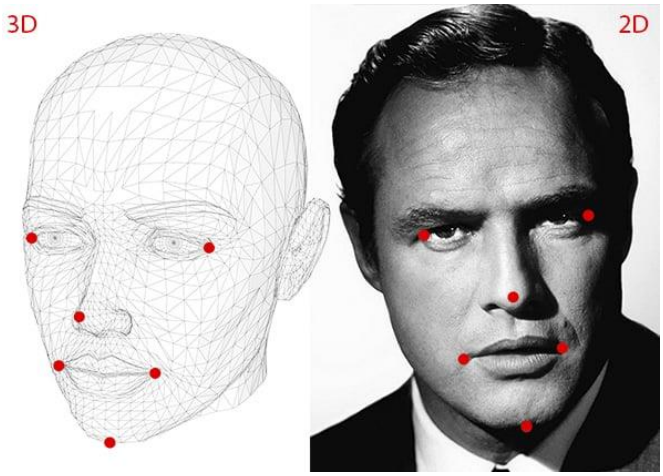


Слика 27: Пример 128 мера лица које генерише неуронска мрежа [31]

Дубоко учење се данас увелико примењује у домену препознавања лица; када је у питању поузданост препознавања, може се рећи да је надмашило традиционалне алгоритме. Недостатком овог метода може се сматрати потреба за тренирањем мреже великим бројем слика лица (које садрже довољно варијација).

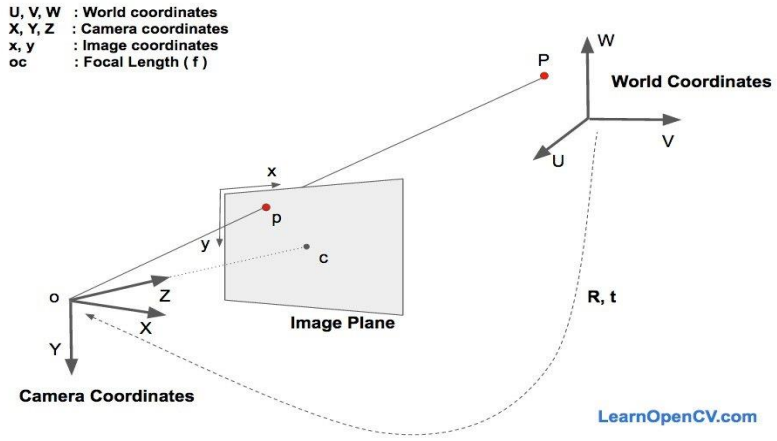
3.3. Одређивање положаја главе

Поза неког објекта се посматра као његова релативна оријентација у односу на камеру – проблем естимације позе се често назива Perspective-n-Point проблем или PnP. Положај објекта се може одредити када постоји калибрисана камера и знају се локације n 3D тачака које одговарају 2D пројекцијама на слици. На слици 28 је дат пример шест тачака у 3D простору и одговарајућих 2D тачака, које се могу користити за одређивање положаја главе [32].



Слика 28: Шест карактеристичних 3D тачака лица и одговарајућих 2D тачака [32]

Објекат се може кретати у односу на камеру применом транслације или ротације, а то кретање можемо посматрати и као кретање камере у односу на објекат. Транслацијом се назива померање камере са њене тренутне 3D позиције (X, Y, Z) на нову 3D локацију (X', Y', Z'), и она се може представити вектором t = (X' – X, Y' – Y, Z' – Z). Ротацијом се назива ротација камере око X, Y и Z осе, а може се представити коришћењем Ојлерових углова, 3 × 3 ротационе матрице или смером ротације и углом. За одређивање позе главе потребно је знати 2D координате (x, y) неколико тачака на слици (врх носа, врх браде, углови очију и углови усана), 3D координате истих тачака, као и унутрашње параметре камере (фокалну дужину камере, оптички центар слике и параметре радијалне дисторзије). Оптички центар слике се може апроксимирати центром слике, фокална дужина ширином слике, а може се претпоставити да радијална дисторзија не постоји.



Слика 29: Координатни системи [32]

Посматрају се три координатна система, приказана на слици 29: светски координатни систем, координатни систем камере и координатни систем слике. 3D координате тачака представљене у светским координатама, уз познату ротацију и транслацију, могу се трансформисати у 3D тачке у координатама камере. Ове тачке могу бити пројектоване на раван слике коришћењем унутрашњих параметара камере, чиме се добијају тачке у координатном систему слике. Разматрају се једначине које дају пројекцију p 3D тачке P, која је на позицији (U, V, W) у светском координатном систему, на раван слике. Ако су познати ротациона матрица R (3 × 3) и транслациони вектор t (3 × 1), могу се израчунати (X, Y, Z) координате тачке P у координатном систему камере:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} U \\ V \\ W \end{bmatrix} + t = [R \mid t] \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}, \tag{7}$$

односно:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}. \tag{8}$$

Ако постоји довољан број парова (X, Y, Z) и (U, V, W), може се решити систем линеарних једначина који ће дати непознате r_{ij} и (t_x, t_y, t_z). При одређивању положаја главе познате су тачке у 3D моделу (U, V, W) и 2D тачке (x, y), али не и (X, Y, Z). У одсуству радијалне дисторзије координате (x, y) тачке p дате су једначином:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = s \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{9}$$

где су f_x и f_y фокалне дужине у *x* и *y* смеровима, а (c_x, c_y) је оптички центар – непознати фактор скалирања *s* постоји у једначини услед чињенице да није позната дубина слике. Спајањем било које 3D тачке *P* са центром камере добиће се пројекција тачке *P* – тачка *p* у којој одговарајући зрак пресеца раван слике. Све тачке дуж једног зрака имаће исту пројекцију на раван слике, тако да је коришћењем претходне једначине могуће добити координате (X, Y, Z) само са скалирањем *s*. Сада једначина (8) изгледа овако:

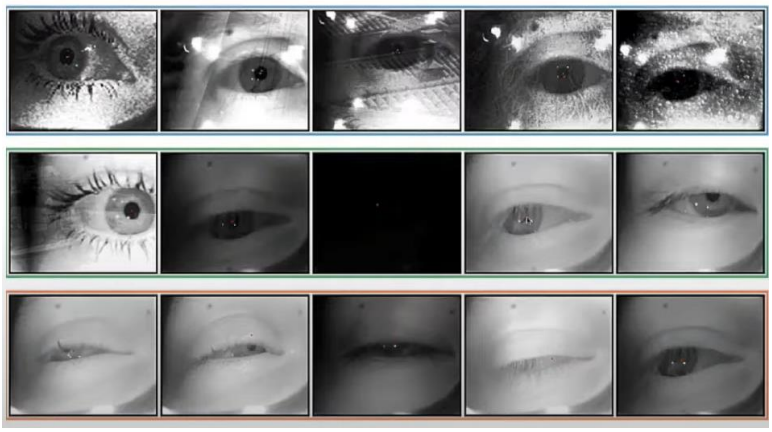
$$s \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix}. \tag{10}$$

Она се може решити коришћењем метода директне линеарне трансформације (DLT), који се може користити кад год постоји проблем где је једначина скоро линеарна, али је помножена непознатим фактором. Из једначина (8) и (9) се види да се (уз познавање *R* и *t*) може наћи тачка *p* на слици за сваку 3D тачку *P* – пројектовањем 3D тачака на 2D слику. *R* и *t* се могу добити DLT методом, а пошто су познате 2D карактеристичне тачке лица, може се посматрати растојање између пројектованих тачака и тих тачака, па израчунати грешка пројекције. Резултат се може побољшати коришћењем метода Levenberg-Marquardt оптимизације (итеративном променом вредности *R* и *t*, тако да се грешка пројекције смањује) [32].

3.4. Праћење погледа

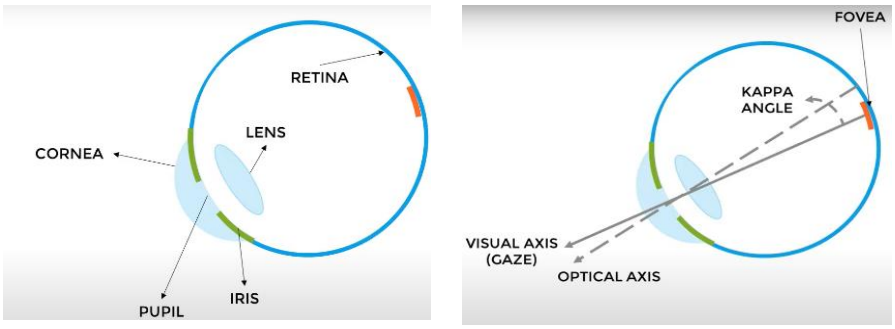
Праћење покрета очију је добро познат проблем у рачунарском виду. Развијена су различита решења за праћење погледа, која се успешно примењују у свакодневном животу. Неки од примера примене су: управљање показивачем на екрану очима, помоћ људима са одређеним видовима инвалидитета, анализа пажње.

Постоје два случаја праћења погледа: када треба одредити тачку на екрану у коју особа гледа, и када треба одредити тачку у сцени у коју особа гледа. У оба случаја користи се систем са две камере [32]. У првом случају једна камера заправо представља екран, а друга снима лице особе и најчешће је закачена за екран. У другом случају особа носи специјалне наочаре које имају камеру у средини (усмерену ка сцени) и другу камеру која је закачена за наочаре и усмерена ка очима (најчешће је ка очима усмерено и инфрацрвено светло које се користи да би се добио бољи контраст, а при томе не утиче на очи, јер га људско око не може спознати). Један од најважнијих корака у одређивању смера гледања јесте детекција зенице, а проблеми који се притом могу јавити су следећи: око може бити снимано из различитих углова; особа може носити наочаре; очи јој могу бити делимично затворене; трепавице могу делимично прекрити око; могу постојати варијације у облику ока код људи; квалитет слике и осветљење могу бити лоши. Примери различитих слика очију су приказани на слици 30.



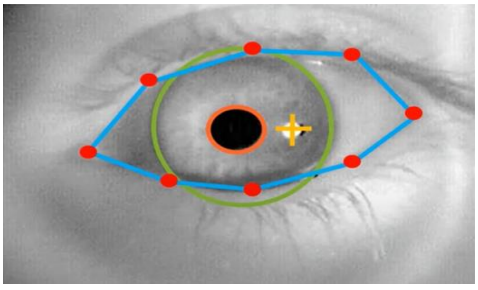
Слика 30: Варијације у сликама очију [33]

На слици 31 приказана је грађа људског ока. Светлост пролази кроз отвор који се зове зеница (енг. pupil), коју контролише дужица (енг. iris), па се прелама кроз сочиво (енг. lens), после чега се слика пројектује на мрежњачу (енг. retina). Око зенице налази се рожњача (енг. cornea). Линија која повезује центар зенице и центар сочива назива се оптичка оса. Место на мрежњачи где је слика најоштрија зове се фовеа (енг. *fovea*) и налази се у центру жуте мрље. Линија која спаја ову тачку и центар сочива назива се визуелна оса – она представља правац погледа. Угао између ове две осе се назива *kappa* угао и најчешће је око пет степени. Визуелну осу је могуће одредити на основу оптичке, а пре тога је потребно урадити калибрацију – корисник се фокусира на одређену тачку, а потом се одређује тачка у коју гледа и налази разлика између локације те тачке и стварне тачке. Та информација се користи да се побољша наредна процена.



Слика 31: Грађа ока (лево), оптичка и визуелна оса (десно) [33]

Неке од карактеристика ока које могу бити значајне за одређивање тачке погледа су означене на слици 32 – ради се о центру и полупречнику зенице, локацији одсјаја рожњаче, центру и полупречнику дужице и контури (полигону) ока.



Слика 32: Значајне карактеристике ока [33]

Најпрецизнији алгоритми за праћење погледа су засновани на 3D моделу главе и ока, при чему се могу користити RGBD камере за одређивање 3D параметара, око и рожњача се могу моделовати као сфере, а зеница и дужица као концентрични кругови. Овакви модели се добро понашају при покретима главе и различитим светлосним условима, али су комплексни, па захтевају калибрацију и доста хардвера. Неки од мање захтевних метода раде без експлицитног моделовања геометрије ока и посматрају одређивање тачке погледа као регресиони проблем [33].

У апликацијама за надгледање полагања тестова може се детектовати трептање и одређивати смер гледања особе која ради тест. Тачка екрана у коју особа гледа се може пратити уколико је потребно одрадити анализу пажње, али за утврђивање регуларности полагања то није неопходно, већ се може проверити само да ли гледа са стране, тј. ван екрана. Праћење треба обављати у реалном времену на видеу са стандардне веб-камере, јер је претпоставка да ученици у кућним условима немају никакву додатну хардверску опрему и не могу задовољити захтеве напреднијих система.

3.5. Детекција говора

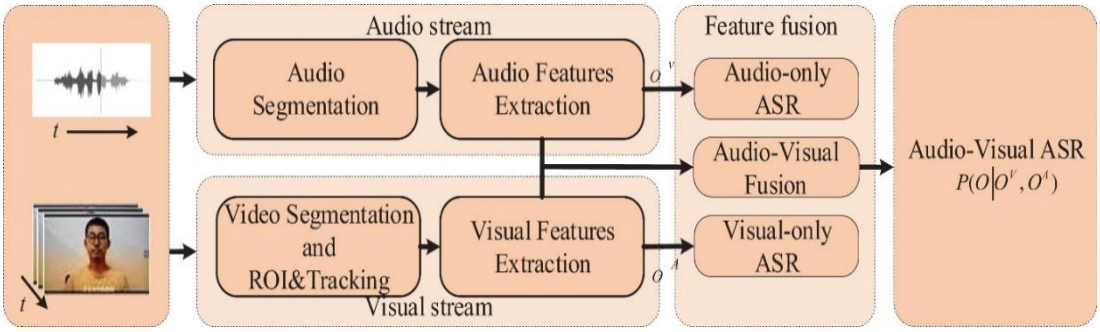
Већина онлајн тестова не захтева усмене одговоре, а причање током теста је забрањено, тако да у системима за надгледање полагања може бити довољно проверити да ли особа прича или не (енг. voice activity detection). Додатно, може се вршити препознавање говора, уколико желимо да у извештају наведемо шта је особа изговарала (енг. speech recognition).

Алгоритми за детекцију говора у аудио сигналу најчешће прво редукују шум, затим издвајају одређене карактеристике сигнала из сваког сегмента сигнала и након тога примењују класификационо правило којим се сегмент класификује као сегмент са говором или као сегмент без говора. Неке од карактеристика које се могу користити јесу: спектрални нагиб, коефицијенти корелације и кепстрални коефицијенти (већина мера се заснива на информацијама о фреквенцијама присутним у сигналу). Детекција говора је изазован задатак због постојања позадинске буке, која може значајно утицати на звучни сигнал [34].

Методе за препознавање говора углавном као основу користе Hidden Markov Model (HMM), односно скривени Марковљев модел. У овом моделу претпоставка је да говорни сигнал, када се посматра у кратком временском интервалу (10 ms), може бити апроксимиран стационарним процесом (у коме се статистичке особине не мењају током времена). Други приступ јесте коришћење неуронских мрежа које се најчешће употребљавају када постоји лош квалитет сигнала, бука или више говорника. У многим модерним системима се користи хибридни модел: неуронске мреже се користе да поједноставе говорни сигнал пре него што се изврши HMM препознавање. Посебно су значајне Recurrent Neural Networks (RNNs), односно рекурентне неуронске мреже, које су широко коришћене у обради природних језика [35].

Када је реч о видео-снимцима, могуће је добити прихватљиве резултате и посматрањем само видео-података. Технике рачунарског вида се могу употребити за детекцију причања и читање речи са усана (овај процес се назива „визуелно препознавање говора“). Традиционалне методе могу посматрати карактеристичне тачке лица или карактеристике усана на нивоу пиксела, а у последње време се све више примењују методе засноване на 3D конволуционим мрежама. Међутим, најбољи начин за детекцију и препознавање говора на видео-снимцима јесте комбиновање визуелних и аудио података у процесу који се назива Audio Visual Speech Recognition (AVSR), односно аудио-визуелно препознавање говора. Визуелне информације (покрети усана) су веома значајне када је заступљена бука у звучном сигналу. На слици 33 је приказано

како изгледа модел једног AVSR система: постоје два улазна канала (аудио и видео). Паралелно се врше сегментација аудија и издвајање аудио фичера, и сегментација видеа, издвајање региона од интереса (регион уста), праћење покрета усана и издвајање визуелних фичера. На крају следи фаза спајања аудио и видео вектора фичера [36].

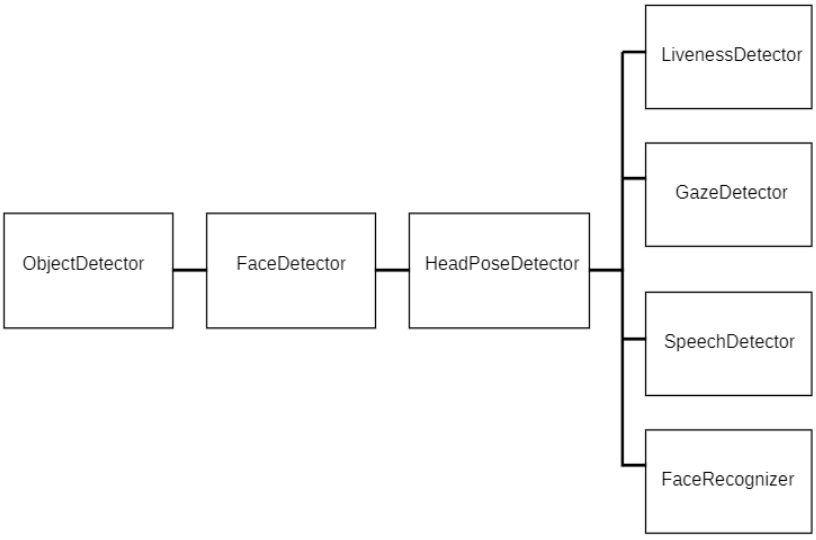


Слика 33: Аудио-визуелни систем за препознавање говора [36]

4. Систем за праћење регуларности полагања тестова употребом камере

За реализацију прототипа система који врши надгледање онлајн полагања тестова употребом камере искоришћен је програмски језик Python, а пројекат је имплементиран у развојном окружењу PyCharm. Систем не представља платформу преко које се могу полагати тестови, већ само симулира полагање теста укључивањем тајмера, а надгледање се обавља обрадом учитаних фрејмова. За рад апликације потребне су основне информације о ученицима и тестовима, а како ти подаци не би били хардкодирани или чувани у локалном фајл систему, коришћене су Firebase услуге[2]. За обраду фрејмова (слика) највише су коришћене библиотеке OpenCV и Dlib. Ове две библиотеке су веома популарне и користе се за потребе машинског учења и рачунарског вида. Кориснички интерфејс апликације је једноставан – интеракција са системом је имплементирана преко конзоле. Омогућен је избор између неколико акција: додавање/брисање ученика, додавање/брисање теста и покретање теста.

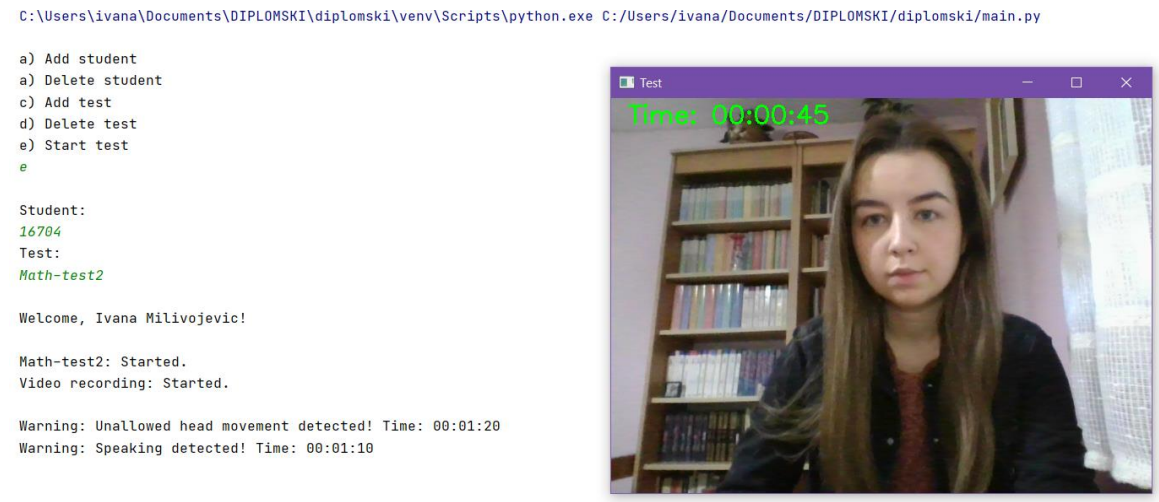
На слици 34 је приказан дијаграм који илуструје компоненте система за надгледање полагања: *ObjectDetector* – детектује присуство више особа у кадру и присуство мобилних телефона; *FaceDetector* – детектује лице и издваја карактеристичне тачке лица; *LivenessDetector* – проверава да ли је детектовано лице „живо“ или је камери подметнута слика; *FaceRecognizer* – проверава да ли детектовано лице одговара лицу ученика који је покренуо тест; *HeadPoseDetector* и *GazeDetector* – проверавају да ли су ученикова глава и поглед усмерени ка екрану; *SpeechDetector* – проверава да ли ученик прича. Свака од компоненти имплементирана је као посебан модул, а из главног фајла (који је улазна тачка апликације) позивају се њихове функције за детекцију и валидацију. Оваква организација кода омогућава да функционалности буду раздвојене и да касније измене у обради фрејмова буду локализоване у оквиру сваког модула.



Слика 34: Компоненте система за надгледање полагања

[2] Firebase је Гуглова платформа за развој апликација – она нуди различите сервисе, попут база података, аутентификације и аналитике.

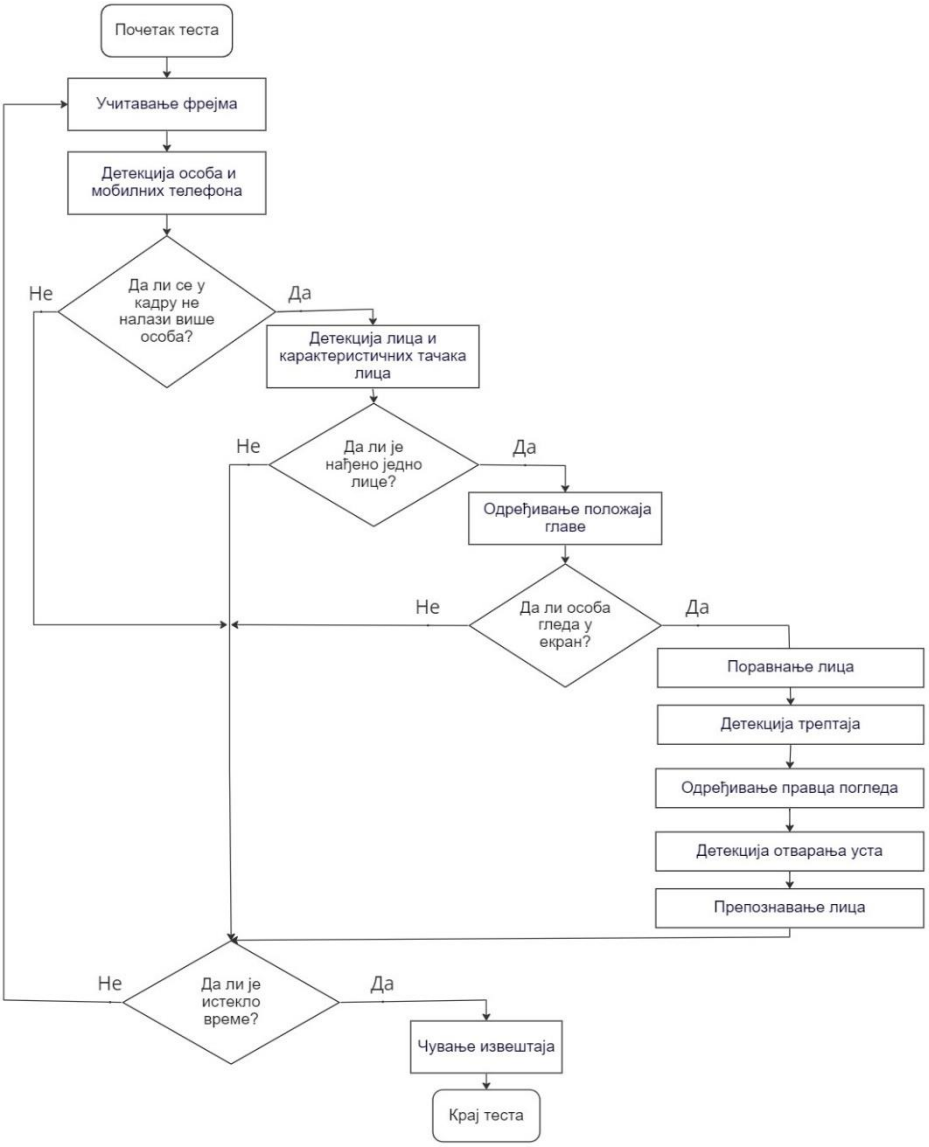
Са базом података се комуницира преко посебног модула, названог *Database*. Модел је максимално поједностављен – подаци који се чувају о ученицима су: идентификациони број, име, презиме, једна слика[3] и видео-снимци полагања тестова, а о тестовима се чувају информације које се односе на идентификациони број и дужину трајања. Слике и снимци се чувају у Firebase Storage-у, а остали подаци у Firestore Database. За комуникацију са овим сервисима коришћена је библиотека Firebase Admin. Пре покретања теста се на основу идентификационог броја ученика учитавају његово име и презиме (како би му се приказала одговарајућа поздравна порука) и његова слика (која се касније користи за препознавање лица), а на основу идентификационог броја теста учитава се информација о трајању теста. Затим се проверава да ли учитана слика садржи лице, лице се поравнава и укључују се тајмер и камера. Док је тест покренут, на снимку са камере који је приказан кориснику исписује се колико је времена остало до краја теста, а у конзоли му се приказују упозорења и обавештења (слика 35).



Слика 35: Кориснички интерфејс

Након покретања камере, систем обрађује један по један фрејм (на слици 36 је приказан ток њихове обраде). На учитаном фрејму прво се детектују особе и мобилни телефони. Уколико није детектовано више особа у кадру, прелази се на детекцију лица и карактеристичних тачака лица. Уколико је ова детекција успешна, проверава се да ли је глава особе усмерена ка екрану. Ако јесте, следи поравнање лица, након чега се даља обрада врши над поравнатим лицем. Затим се утврђује да ли се десио трептај, пошто се бројањем трептаја врши провера да ли је камери подметнута слика. Након тога се одређује да ли особа гледа право у екран (или са стране) и да ли су јој уста отворена, јер се праћењем покрета усана закључује да ли особа прича. На крају се врши препознавање лица. Обрада фрејмова се прекида када прође онолико времена на колико је подешено трајање теста (или притиском на тастер „Q“ на тастатури, у случају ранијег завршетка теста), и тада се бележи извештај у форми видео-фајла, који се касније може прегледати ради провере спорних ситуација. Генеришу се два фајла: један који садржи комплетан снимак полагања са означеним спорним ситуацијама, и други, који садржи само спорне ситуације.

[3] Претпоставка је да сваки ученик који је убачен у систем има слику попут слика за личне документе (на којој му се лице јасно види и гледа у камеру).



Слика 36: Дијаграм тока обраде фрејмова

Ради лакше обраде, направљена је посебна класа која представља један фрејм – ова класа има четири атрибута: слику, вредност тајмера у тренутку учитавања фрејма, информацију о валидности и поруку о нерегуларностима, која ће бити исписана на фрејму. Како у једном фрејму може истовремено бити више нерегуларности, сваки од детектора ажурира атрибут поруке дописивањем специфичног текста и ажурира атрибут слике доцртавањем додатних информација о детектованој нерегуларности. Сви учитани фрејмови се смештају у бафер главног програма, на основу кога се на крају генерише видео-запис полагања. Да би се у посебном фајлу издвојиле само спорне ситуације, атрибут валидности сваког фрејма у коме је детектована нека спорна ситуација се поставља на вредност „нетачно“, а на крају теста се бафер фрејмова филтрира по овом атрибуту. Поред тога, сваки од појединачних детектора има свој бафер, у коме чува „прозоре“ фрејмова (прозором се сматра одређени број сукцесивних фрејмова). Прозори се посматрају како би се избегле случајне (лажне) детекције – спорна ситуација неће трајати само један фрејм, већ више узастопних фрејмова. Сви фрејмови који припадају прозору који се сматра нерегуларним биће означени као нерегуларни. У тренутку завршетка теста прекидају се текући прозори свих детектора, и позивају се њихове функције за ресетовање.

4.1. Имплементација компоненти система

Прва компонента система, задужена за детекцију објеката, имплементирана је у оквиру модула *ObjectDetector*. Један од најпознатијих детектора је YOLO, међутим, оптимизован је за рад на GPU. Ова апликација се извршава на CPU и, како би имала солидне перформансе у реалном времену, изабран је MobileSSD модел. У пројекту је коришћена TensorFlow имплементација [37], која комбинује MobileNet архитектуру и SSD. MobileNet архитектура је ефикаснија у односу на друге мрежне архитектуре (попут VGG или ResNet) када су у питању уређаји са ограниченим ресурсима (зато што дели конволуцију на две фазе), али има мало мању тачност [38]. Овај модел је трениран COCO [39] скупом слика 80 класа објеката који се могу срести у свакодневном животу. Скуп садржи око 330 000 слика, са 2,5 милиона означених инстанци објеката. Како је циљ овог модула детекција особа и мобилних телефона у кадру, од значаја су биле класе објеката „person“ и „cellphone“. Коришћене су функције из модула за дубоко учење OpenCV библиотеке, који је доступан од верзије 3.1 (и значајно унапређен у верзији 3.3).

На почетку је учитан и парсиран текстуални фајл са лабелама за сваку класу објеката, тако да се од њега направи низ који садржи називе класа, након чега је учитана мрежа позивом функције *readNetFromTensorflow* (слика 37).

```
with open('detectors/object_detector/COCO_labels.txt', 'r') as f:
    self.classes = f.read().split('\n')

self.neural_network = cv2.dnn.readNetFromTensorflow('detectors/object_detector/frozen_inference_graph.pb',
                                                    'detectors/object_detector/ssd_mobilenet_v2_coco_2018_03_29.pbtxt')
```

Слика 37: Учитавање назива класа и мреже за детекцију објеката

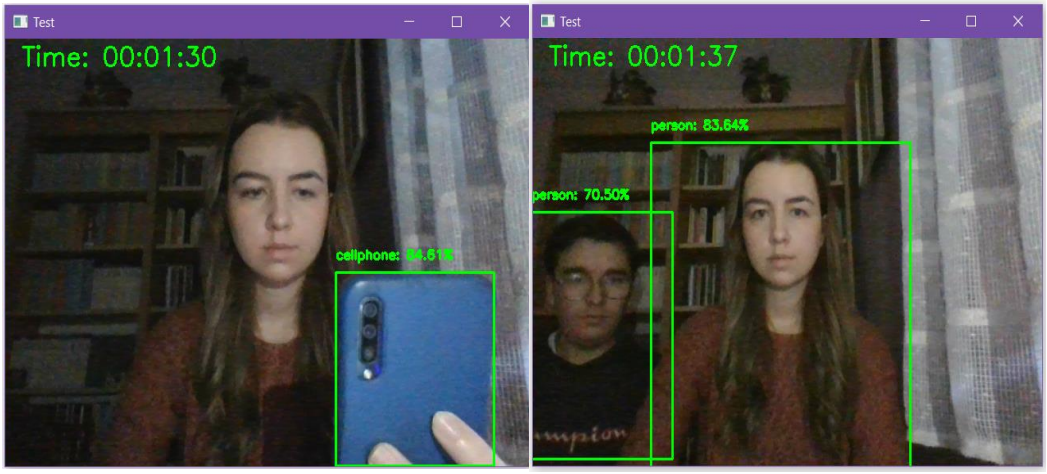
Величина улазне слике (фрејма) је коришћењем функције *blobFromImage* промењена на величину од 300 × 300 пиксела – коју мрежа очекује; слика је нормализована (од сваког пиксела је одузета средња вредност пиксела тренинг слика за сваки од канала), а затим је прослеђена мрежи. Мрежа за сваки детектовани објекат враћа информацију о класи, о координатама правоугаоника којим се објекат може обухватити и о поузданости детекције. Координате правоугаоника су у опсегу [0, 1], те се множењем тих бројева са оригиналном ширином и висином слике добија предикција за оригиналну слику. Разматране су само детекције које имају поузданост већу од 0,6 и међу њима издвајане оне које припадају класама од интереса (слика 38).

```
def detect(self, frame, h, w):
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), size=(300, 300), mean=(104, 117, 123), swapRB=True)
    self.neural_network.setInput(blob)
    detections = self.neural_network.forward()

    person_cnt = 0
    cellphone_cnt = 0
    self.result = []

    for i in np.arange(0, detections.shape[2]):
        score = float(detections[0, 0, i, 2])
        if score > 0.6:
            class_id = int(detections[0, 0, i, 1])
            class_name = self.classes[class_id]
            if class_name == "person" or class_name == "cellphone":
                self.result.append((detections[0, 0, i, 3:7] * np.array([w, h, w, h]), score, class_name))
                if class_name == "person":
                    person_cnt = person_cnt + 1
                else:
                    cellphone_cnt = cellphone_cnt + 1
    return person_cnt < 2, cellphone_cnt == 0
```

Слика 38: Функција за детекцију објеката

За обе класе од интереса направљен је по један бафер прозора. Тестирањем различитих величина прозора установљено је да је 30 фрејмова довољно да се детектују спорне ситуације. На нивоу сваког прозора броји се колико је било нерегуларних фрејмова, и ако тај број прелази једну трећину укупног броја фрејмова у прозору, он се сматра нерегуларним. Направљен је и помоћни бафер, који чува узастопне нерегуларне фрејмове – овај бафер има улогу да контролише прелаз између прозора. Може се десити да се један део нерегуларне секвенце налази у текућем, а други у следећем прозору, и да она не буде урачуната у потпуности. Из тог разлога се на прелазу из једног у други прозор испитује да ли бафер који садржи узастопне нерегуларне фрејмове није празан. Ако није празан, текући прозор се „скраћује“ за број елемената тог бафера и не отвара се нови прозор све док има сукцесивних нерегуларних фрејмова. Када се наиђе на први валидни фрејм, испитује се да ли у овом баферу има више елемената од половине величине једног прозора, и, ако има, ти фрејмови се означавају као нерегуларни.

У овом модулу је обезбеђена и функција која ће у извештају око детектованог објекта исцртати оквирни правоугаоник и исписати колика је поузданост детекције. На слици 39 у левом делу приказан је пример детекције коришћења мобилног телефона. Треба напоменути да детекцију мобилних телефона „омета“ чињеница да ће најчешће бити држани испод стола, бити прекривени руком или прислоњени уз главу ако особа са неким разговара. У тим случајевима велика је вероватноћа да телефон неће бити детектован, али хоће ако га особа држи као на слици 39 и покушава да нешто прочита или слика питања са теста. У десном делу слике приказана је ситуација када су детектоване две особе у кадру. Овакви фрејмови се неће прослеђивати даље, већ ће узроковати ресетовање осталих детектора. Може се десити да модел не детектује особу уколико је мало ближа камери, те из тог разлога у овом детектору ситуација када нема особе у кадру није сматрана проблематичном (да не би било лажних детекција спорне ситуације). Овај случај је обухваћен у наредном модулу (који детектује лице). Функција за ресетовање проверава да ли је до тог тренутка у баферу сукцесивних нерегуларних фрејмова било довољно елемената да би се обухваћени период сматрао спорним. Ако то није случај, проверава се да ли је обрађено више од две трећине прекинутог прозора, и испитује се да ли у њему има довољно нерегуларних фрејмова.



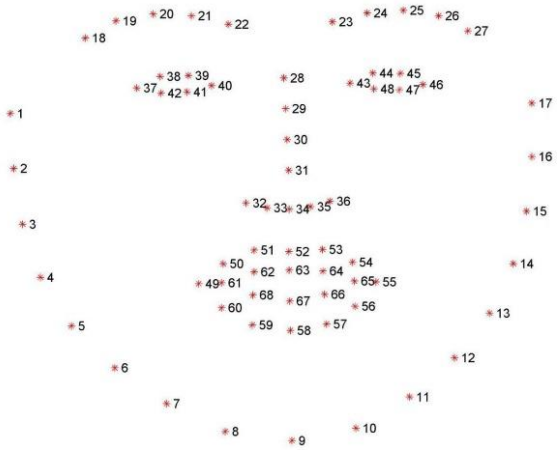
Слика 39: Детекција мобилног телефона (лево) и
детекција две особе (десно)

Следећа компонента, задужена за детекцију лица и карактеристичних тачака лица, имплементирана је у оквиру модула *FaceDetector*. При одабиру детектора лица треба размотрити неколико најзначајнијих детектора: Viola-Jones и DNN детектор из OpenCV библиотеке, као и HOG у комбинацији са линеарним SVM и CNN детектор из Dlib

библиотеке. Viola-Jones је најпознатији детектор лица; веома је брз и погодан за извршавање у реалном времену. Са друге стране, подложан је лажно позитивним детекцијама, захтева ручно подешавање параметара, детектује само фронтална лица и неће радити ако је лице нечим прекривено. HOG алгоритам је поузданији и има стабилнију детекцију. Његови недостаци су у томе што ради само са фронталним (и благо окренутим) лицима, није толико поуздан као детектори засновани на дубоком учењу, не може детектовати веома мала лица, и оквирни правоугаоник некада може да одсече део чела или браде. CNN детектор има велику тачност, може детектовати делимично прекривена лица, ради добро при различитим оријентацијама лица и светлосним условима, али није погодан за коришћење у реалном времену без GPU убрзања. DNN детектор лица из OpenCV библиотеке је заснован на SSD и ResNet мрежи. Може се извршавати у реалном времену на просечним рачунарима и поузданији је од Viola-Jones и HOG алгоритма, али је мање поуздан од CNN из Dlib библиотеке. Модел ради добро када је лице делимично прекривено, када постоје брзи покрети главе, и може детектовати бочна лица – али некада не детектује веома мала лица [40].

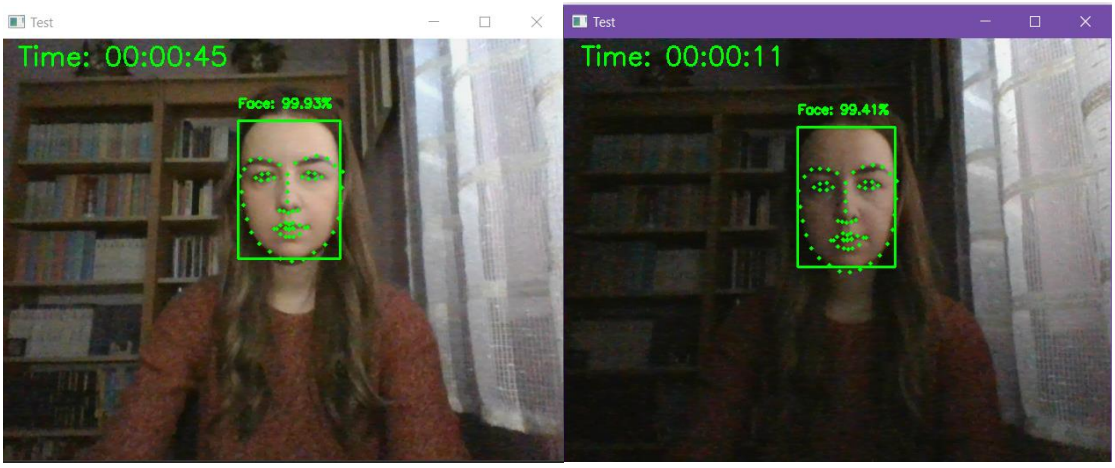
За детекцију лица одабран је DNN детектор, због брзине извршавања и добрих резултата при екстремним условима. Коришћена је квантована[4] TensorFlow верзија [41] модела, зато што се извршава за нијансу брже од оригиналне. Поступак детекције лица је сличан као код детекције објеката, а нерегуларним фрејмовима се сматрају они на којима се не налази само једно лице, тако да ће се у том случају ресетовати детектори који очекују лице као улаз.

Након детекције лица, детектују се карактеристичне тачке, потребне за даљу обраду лица. Dlib нуди детектор 68 тачака и детектор 5 значајних тачака – који је бржи, али препорука је да се користи у случајевима када су потребне само локације носа и очију. У овом пројекту је коришћен модел који детектује 68 тачака [42]. Детектују се координате тачака које се мапирају на структуру лица, приказане на слици 40. Имплементиране су и помоћне функције које враћају специфичне тачке лица: тачке левог и десног ока (на слици означене бројевима 37-40 и 43-48), горње и доње усне (на слици означене бројевима 49-68) и шест карактеристичних тачака за естимацију позе главе (на слици означене бројевима 31, 9, 46, 37, 55 и 49). На слици 41 приказано је лице заједно са кључним тачкама при различитом осветљењу, где се може приметити да детектор лица ради са веома великом поузданошћу.



Слика 40: Локације 68 карактеристичних тачака лица [43]

[4] Квантизација је процес којим се смањује број битова којима се представљају бројеви у неуронској мрежи.



Слика 41: Детекција лица заједно са карактеристичним тачкама, при различитом осветљењу

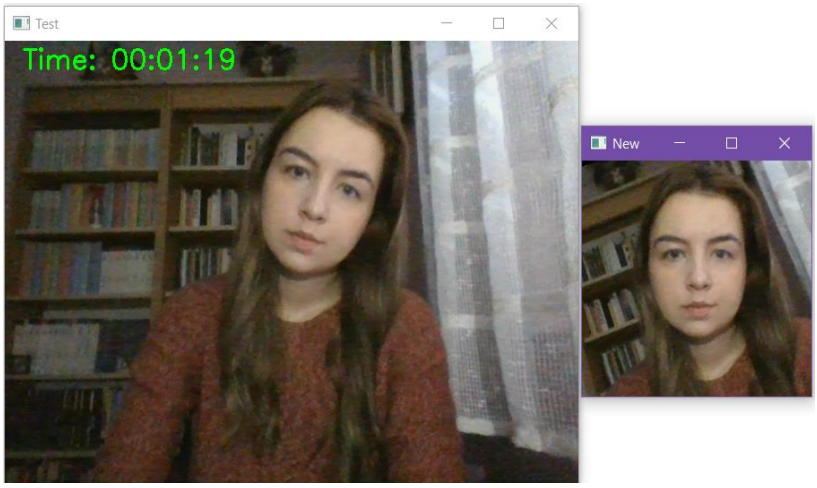
У овом модулу имплементирана је и функција за поравнање лица – поравнање лица се ради како би се побољшали алгоритми који ће се примењивати над лицем. Овде су то: препознавање лица, праћење покрета очију и детекција говора. Циљ је да лице буде центрирано на слици, заротирано (тако да очи буду на хоризонталној линији), и да буде скалирано (тако да величина свих лица буде приближно иста). Све слике су смањене на величину од 256 × 256 пиксела.

На почетку се задаје жељена (x, y) позиција центра левог ока [44]. Координате су изражене у процентима ширине и висине оригиналне слике, који су најчешће у опсегу 20-40% (у овом пројекту узето је 40%). Што је мањи проценат, лице ће више бити зумирано. Затим се израчунава жељена *x* позиција центра десног ока (од 1 се одузме жељена *x* позиција левог ока, због симетрије), а жељена *y* позиција десног ока је иста као за лево. На основу карактеристичних тачака очију израчунавају се координате тренутних центара једног и другог ока, а након тога и угао који линија која их спаја заклапа са хоризонталом – овај угао је кључан за поравнање слике. Затим је потребно наћи централну тачку која се налази између очију, јер ће се око ње вршити ротација. Сада се може израчунати тренутна дистанца између два ока. Дељењем жељене дистанце са тренутном дистанцом између центара очију може се израчунати фактор скалирања. За рачунање ротационе матрице M коришћена је OpenCV функција *getRotationMatrix2D*, којој су као параметри прослеђени претходно израчунати центар ротације, угао ротације и фактор скалирања (слика 42).

```
desiredRightEyeX = 1.0 - self.desired_left_eye[0]

left_eye_center = left_eye_pts.mean(axis=0).astype("int")
right_eye_center = right_eye_pts.mean(axis=0).astype("int")

dY = right_eye_center[1] - left_eye_center[1]
dX = right_eye_center[0] - left_eye_center[0]
angle = np.degrees(np.arctan2(dY, dX))

eyes_center = ((left_eye_center[0] + right_eye_center[0]) // 2, (left_eye_center[1] + right_eye_center[1]) // 2)
center = (int(eyes_center[0]), int(eyes_center[1]))

dist = np.sqrt((dX ** 2) + (dY ** 2))
desiredDist = (desiredRightEyeX - self.desired_left_eye[0])
desiredDist *= self.desired_face_width
scale = desiredDist / dist

M = cv2.getRotationMatrix2D(center, angle, scale)
```

Слика 42: Рачунање ротационе матрице

Након ажурирања транслационе компоненте ове матрице може се применити функција *warpAffine* из OpenCV библиотеке, којој се као параметри прослеђују слика, матрица M, жељена ширина и висина излазне слике и параметар којим се специфицира интерполациони алгоритам. На слици 43 у левом делу приказана је слика где очи особе нису у хоризонтали, а у десном делу обрађена слика, где је издвојено поравнато лице.



Слика 43: Оригинални фрејм (лево) и поравнато лице (десно)

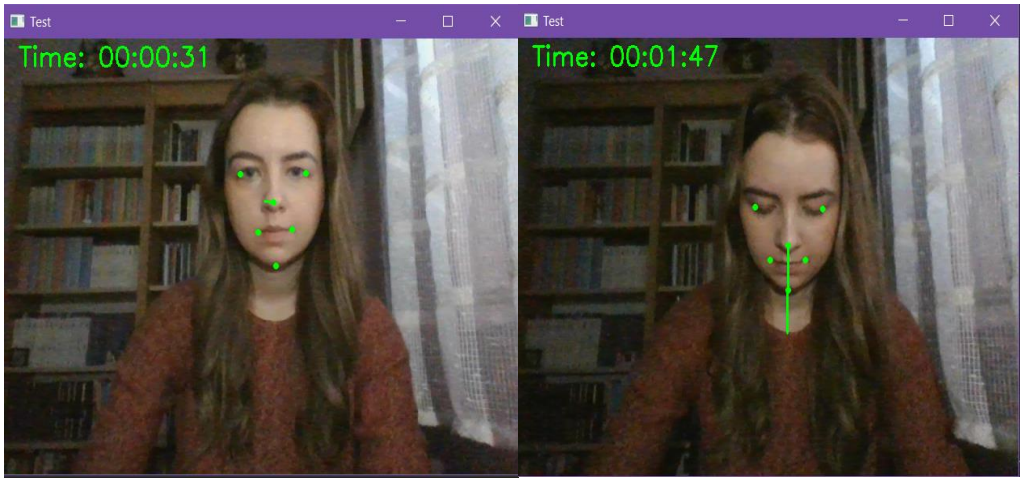
Следећа компонента, намењена детекцији окретања главе, је имплементирана у оквиру модула *HeadPoseDetector*. Одређивање положаја главе ученика је реализовано коришћењем карактеристичних тачака лица и неколико функција из OpenCV библиотеке. Издвајање спорних ситуација и ресетовање су (и у овом случају) имплементирани као у детектору објеката.

Функција коришћена за решавање PNP проблема је *solvePnP*. Она као параметре има низ 3D тачака лица, низ одговарајућих тачака на слици (2D тачке лица), унутрашњу матрицу камере, низ коефицијената дисторзије, и, опционо, метод за решавање PNP проблема (подразумевани је SOLVEPNP_ITERATIVE, који одговара оптимизованом DLT решењу). Уместо 3D модела лица може се користити генерички 3D модел, са тачкама у произвољном (светском) координатном систему: врх носа (0, 0, 0), брада (0, -330, -65), леви угао левог ока (-225, 170, -135), десни угао десног ока (225, 170, -135), леви угао уста (-150, -150, -125), десни угао уста (150, -150, -125). Оптички центар слике се може апроксимирати центром слике, фокална дужина ширином слике, а може се претпоставити да радијална дисторзија не постоји. Функција враћа ротациони и транслациони вектор, који трансформишу 3D тачке објекта у координате камере [32]. Затим се могу издвојити информације о угловима ротације око координатних оса коришћењем функције *RQDecomp3x3* (слика 44). Ова функција захтева ротациону матрицу, а не ротациони вектор (чију конверзију нам омогућава функција *Rodrigues*).

```
(success, rotation_vector, translation_vector) = cv2.solvePnP(self.face_3d_model, image_points,
                                                              self.camera_matrix,
                                                              self.dist_coeffs)

rmat, jac = cv2.Rodrigues(rotation_vector)
angles, mtxR, mtxQ, Qx, Qy, Qz = cv2.RQDecomp3x3(rmat)

self.x = angles[0]
self.y = angles[1]
self.z = angles[2]
```

Слика 44: Налажење углова ротације главе

Као координатни почетак се посматра врх носа; *x*-оса је усмерена на десно (из перспективе особе на слици), *y*-оса на горе, а *z*-оса право ка екрану. Ротација главе лево и десно одговара ротацији око *y*-осе, а ротација горе и доле врши се око *x*-осе. На слици 45 лево је приказана ситуација када особа гледа право у екран, а десно ситуација када је њен поглед усмерен на доле. Фрејмови у којима ученик не гледа право у екран се не пропуштају наредним детекторима, већ узрокују њихово ресетовање.

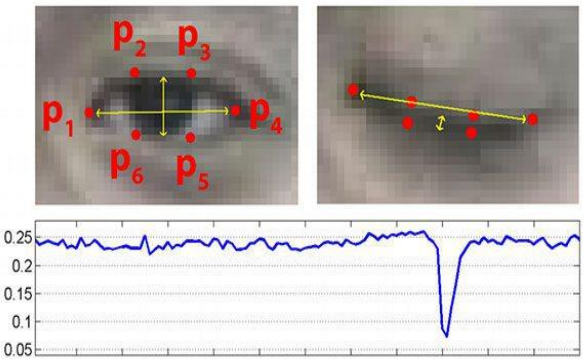


Слика 45: Пример ситуације када је глава особе усмерена право ка екрану (лево) и на доле (десно)

У оквиру модула *LivenessDetector*, за проверу да ли је лице „живо“, односно да ли је камери подметнута слика лица, бројани су трептаји ока у одређеном временском интервалу. За детекцију трептаја ока коришћена је метрика EAR (Eye Aspect Ratio), предложена у раду *Real-Time Eye Blink Detection using Facial Landmarks* [45]. EAR је решење које врши једноставну калкулацију засновану на односу растојања између одговарајућих карактеристичних тачака очију. Свако око представљено је помоћу шест тачака, почевши од левог угла ока и крећући се у смеру казаљке на сату. Постоји релација између ширине и висине региона ока:

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_4||}, \tag{7}$$

где су p_1, …, p_6 тачке које одређују око. Ова величина је приближно константна (око 0,25) док је око отворено, а нагло опада ка нули када се деси трептај. На графику са слике 46 се може видети EAR кроз време на видео-снимку. Коришћењем ове методе избегавају се технике процесирања слике.



Слика 46: Детекција трептаја [45]

Трептај ока ће трајати минимум два фрејма, тако да се два и више узастопних фрејмова који имају EAR мањи од граничне вредности (0.22) могу сматрати трептајем. Људи у просеку трепну између 12 и 22 пута у току једног минута, а како је прозор од једног минута сувише дуг (јер неки тестови могу кратко трајати, па систем можда неће стићи да детектује неправилност), посматран је прозор од 30 секунди, пошто се претпоставља да ниједан тест неће трајати краће од тог времена. Очекује се да ће у том периоду особа трепнути између 6 и 11 пута, али посматран је опсег мало шири од овог због потенцијалних грешака при детекцији. Прозори који имају бројач трептаја ван овог опсега сматрани су нерегуларним, а ресет функција овог детектора враћа на нулу бројаче и празни бафер. Ово је један од једноставнијих начина провере; систем се може преварити подметањем видео-снимка. Ученик може припремити снимак на коме ради тест и користити га као виртуелну камеру. Систем би овакав начин варања могао да детектује тако што би повремено захтевао од особе која полаже тест да направи одређени покрет, који подметнути снимак неће моћи да прикаже.

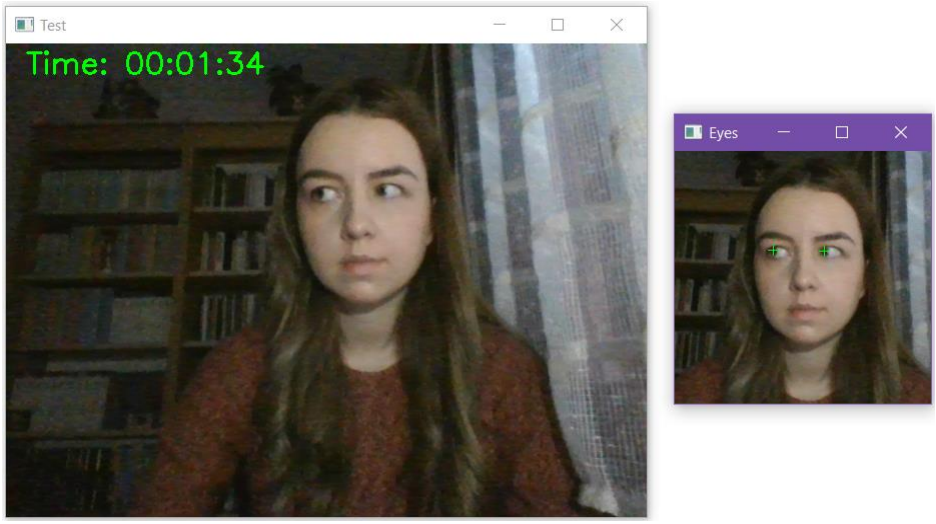
Ученик би током полагања теста требало да гледа право у екран и да не скреће поглед лево, десно, горе или доле. У модулу *GazeDetector* детектују се периоди када постоји гледање ван екрана. Овај детектор посматра прозоре фрејмова на сличан начин као претходно описани детектори. Оба ока су праћена истовремено и мерено је растојање зенице од централног положаја по хоризонтали (чиме се детектује гледање у лево или десно) и по вертикали (чиме се детектује гледање на горе). Поглед на доле није детектован на овај начин, јер су тада очи практично затворене и зеница се не може издвојити, али је искоришћена информација о EAR (која је претходно израчуната). Детектору се прослеђују поравната слика и карактеристичне тачке једног и другог ока, прати се центар зенице (уз помоћ функција из OpenCV библиотеке) и рачуна однос између растојања центра зенице од десне (горње) границе ока и ширине (висине) ока. Овај однос је опсегу од 0 до 1, а док особа гледа право вредност је 0,5. Ако је однос по хоризонтали мањи од 0,35 – сматра се да особа гледа десно, а ако је већи од 0,65 – лево. Ако је однос по вертикали мањи од 0,35 – сматра се да особа гледа на горе, а ако је EAR мањи од граничног, сматра се да гледа на доле (слика 47). Карактеристичне тачке очију се користе како би се изоловали са слике само региони од интереса.

```
def check_frame(self, frame, left_eye_landmarks, right_eye_landmarks, closed):
    if closed:
        return False, "Down"

    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    self.left_eye = Eye(frame, left_eye_landmarks)
    self.right_eye = Eye(frame, right_eye_landmarks)

    self.vertical_ratio = (self.left_eye.get_vertical_ratio() + self.right_eye.get_vertical_ratio())/2
    self.horizontal_ratio = (self.left_eye.get_horizontal_ratio() + self.right_eye.get_horizontal_ratio()) / 2

    if self.horizontal_ratio <= 0.35:
        return False, "Right"
    elif self.horizontal_ratio >= 0.65:
        return False, "Left"
    elif self.vertical_ratio <= 0.35:
        return False, "Up"
    else:
        return True, "Center"
```

Слика 47: Функција за одређивање правца погледа

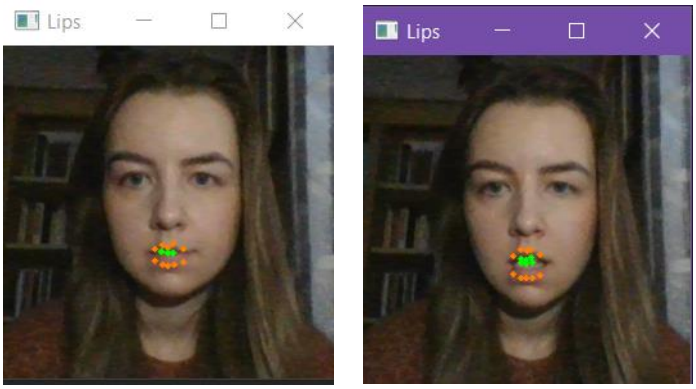
Око се са слике издваја по угледу на имплементацију у библиотеци Gaze Tracking [46] – тако што се направи једна црна слика истих димензија као фрејм, затим слика која служи као маска (бела слика на којој се црним пикселима попуни полигон који представља око); а након тога се примени функција *bitwise_not*. Добија се бела слика на

којој је само око у боји, након чега се издваја део те слике на коме је око. Над издвојеним оком се примењује функција *bilateralFilter* како би се уклонио нежељени шум и задржале ивице, а затим операција ерозије, како би се уклониле избочине и танке везе. Након тога се проналази праг којим ће функција *threshold* најбоље издвојити дужицу ока. Биће враћено око које је беле боје, са дужицом црне боје. Након тога се користи функција *findContours* за проналажење контуре дужице и *moments*, како би се нашле координате центра те контуре, односно зенице. На слици 48 приказан је пример гледања са стране, са означеним центрима зеница.



Слика 48: Праћење зеница

Детекција причања је реализована у оквиру модула *SpeechDetector* праћењем покрета усана. Коришћене су карактеристичне тачке горње и доње усне и мерена растојања између одговарајућих тачака. Детектор је иницијализован сликом из базе података на којој су ученику уста затворена. Вредности које представљају растојања по *y*-оси карактеристичних тачака усана на фрејму (приказаних на слици 49 лево) су унете у два низа: један за спољашње (означене наранџастом бојом), један за унутрашње (означене зеленом бојом) тачке. Ови низови су третирани као вектори, који су након нормализације упоређивани са истим векторима добијеним са слике из базе. Ако је еуклидско растојање између њих веће од унапред задате вредности (експериментално је одабрана вредност 0,2) – уста се сматрају отвореним (слика 50).



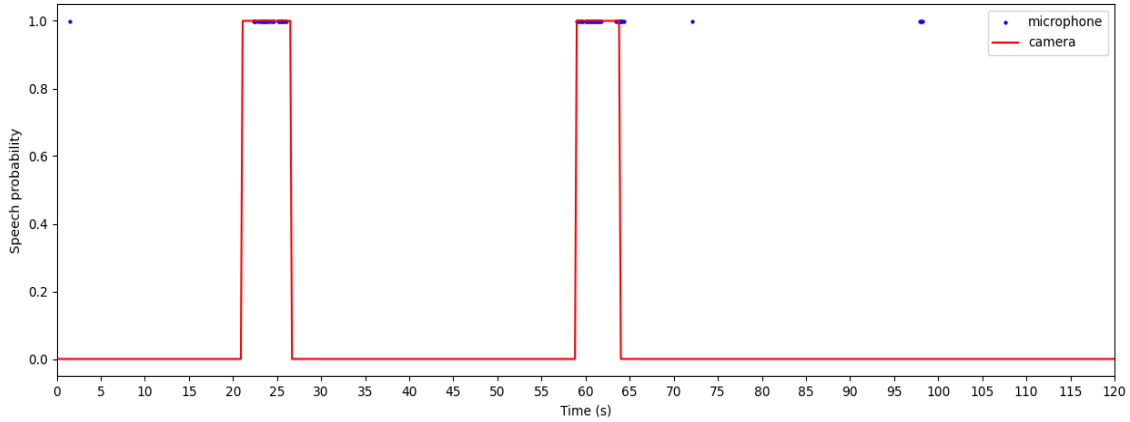
Слика 49: Спољашње и унутрашње тачке усана када су уста затворена (лево) и отворена (десно)

```python
def is_open(self, input_image, top_lip, bottom_lip):
    self.set_image(input_image, top_lip, bottom_lip, False)
    if self.initial_image_set() and self.input_image_set():
        dist1 = np.linalg.norm(self.initial_dist_outer - self.input_dist_outer)
        dist2 = np.linalg.norm(self.initial_dist_inner - self.input_dist_inner)
        if dist1 > 0.2 or dist2 > 0.2:
            return True
        else:
            return False
```

Слика 50: Фукција за утврђивање да ли су уста отворена

У оквиру прозора фрејмова броје се фрејмови на којима су уста ученика отворена, и уколико је при преласку на наредни прозор тај број већи од четвртине обрађених фрејмова, пријављује се причање. Како се може десити да се ученик прозева, смеје или из неког другог разлога држи уста отвореним дуже време, те ситуације се толеришу и сматрају валидним. Експериментално је утврђено да приликом причања неће бити више од четири узастопна фрејма када су уста отворена, тако да се низови дужи од 4 сматрају валидним – уколико се налазе у оквиру прозора неће се урачунавати у бројање. Постоји бафер који чува узастопне фрејмове у којима су уста отворена – уколико на преласку између прозора овај бафер има елемената, отварање новог прозора се одлаже све док су уста отворена. При ресетовању детектора се испитује да ли је обрађено више од две трећине текућег прозора и спорном ситуацијом се сматра случај када је бројач фрејмова са отвореним устима већи од половине броја обрађених фрејмова.

Детекција би била поузданија ако би се користили подаци са микрофона: у оквиру модула који би обрађивао аудио податке и детектовао интервале у којима има гласовне активности. У овом пројекту је фокус био на праћењу података са камере, али је, поређења ради, покренута још једна нит у оквиру које је симултано детектован људски глас. Дакле, једна нит прихвата и обрађује податке са камере, а друга са микрофона. Уз помоћ PyAudio библиотеке отворен је аудио стрим и током трајања теста учитаване су секвенце аудио података у трајању од 20 ms, које су прослеђиване детектору гласовне активности коме се приступа преко WebRTC VAD интерфејса [47]. Ове секвенце се називају фрејмови, и најчешће су у трајању од 20 ms зато што је то оптималан интервал у оквиру кога се звучни сигнал може сматрати стациоанарним процесом. На слици 50 је приказан график вероватноће присутности људског гласа у времену (за цртање графика искоришћена је Matplotlib библиотека).



Слика 50: График зависности вероватноће присутности гласовне активности од времена

Посматран је период од 120 секунди: вероватноћа има вредност 0 ако се сматра да није детектован глас, а 1 у супротном. Црвеном линијом је означен резултат добијен на основу обраде фрејмова са камере, док је плавим тачкама означен резултат добијен обрадом аудио фрејмова. Може се приметити да се добијени резултати у великој мери поклапају, а изоловане плаве тачке су ситуације у којима је глас трајао веома кратко, или које су грешком означене као позитивне, тако да их не треба посматрати као спорне. Једна плава тачка представља вероватноћу за постојање гласовне активности у десетом делу секунде. Ова вредност је 1 уколико је више од 50% фрејмова из тог дела секунде детектор гласовне активности означио позитивно. На слици 51 се може видети на који начин је креирана структура речника која чува податке потребне за цртање дијаграма.

```
while True:
    time_passed = round(time.time() - start_time, 1)
    if time_passed > time_limit:
        break
    data = stream.read(frames_per_buffer)
    is_speech = vad.is_speech(data, sample_rate)
    if time_passed not in self.voice_dict:
        self.voice_dict[time_passed] = 0
    else:
        if is_speech:
            self.voice_dict[time_passed] = self.voice_dict[time_passed] + 1
        else:
            self.voice_dict[time_passed] = self.voice_dict[time_passed] - 1
```

Слика 51: Програмска петља за обраду аудио података

На крају, у оквиру модула *FaceRecognizer*, за препознавање лица коришћен је модел мреже ResNet (са 29 конволуционих слојева) из Dlib библиотеке [48]. Овај модел има тачност од 99,38% на LFW скупу слика, а трениран је коришћењем 3 милиона слика лица. Посматран је прозор од 150 фрејмова, и у оквиру сваког прозора је вршено препознавање по три пута (са једнаким размацима), с обзиром на то да је ово захтевнија операција и проузроковала би знатно слабије перформансе када би била извршавана над сваким фрејмом. Уколико су у прозору била бар два неуспешна препознавања, тај прозор се означава као спорaн. При ресетовању овог детектора, уколико је до момента ресета бар два пута у текућем прозору препознавање било неуспешно, фрејмови из тог прозора који су обрађени означавају се као нерегуларни.

Дескриптори за иницијалну слику (из базе) и улазну слику (са камере) рачунају се коришћењем функције *compute_face_descriptor* (слика 52), којој се као параметри дају поравната слика лица и карактеристичне тачке лица. Мрежа генерише 128 јединствених реалних вредности које описују лице. Ове две слике су упоређиване, тачније, упоређивани су њихови дескриптори – нема потребе упоређивати улазну слику са свим сликама из базе. Разлика између њих је рачуната као еуклидско растојање њихових дескриптора (вектора). Уколико је разлика између њих већа од 0,5 сматра се да лице није препознато (различита лица углавном имају удаљеност од преко 0,8, а врло слична лица између 0,6 и 0,7 – те је зато као граница одабрана вредност 0,5).

```
def set_image(self, image, marks, initial):
    encodings = np.array(self.net.compute_face_descriptor(image, marks, 1))
    if initial:
        self.initial_image_encodings = encodings
    else:
        self.input_image_encodings = encodings
```

Слика 52: Рачунање дескриптора слика лица

5. Закључак

Развој веб технологија омогућио је да онлајн учење буде све заступљеније у пракси, а платформе за учење посебно су добиле на значају у периоду пандемије. Предности оваквог начина учења су бројне, почевши од могућности рада од куће, до брзе и прегледне анализе резултата ученика. Са друге стране, многи сматрају да знање стечено кроз онлајн наставу никада неће моћи да буде једнако добро као знање добијено у учионицама.

Софтвер за полагање тестова је развијан заједно са софтверима за учење. Надгледање полагања, уобичајено, може вршити дежурно особље, али вештачка интелигенција и машинско учење значајно су унапредили и аутоматизовали овај процес. Уз коришћење оваквих софтвера губи се потреба за увођењем брзих тестова (када нема довољно времена за консултацију са неким), који су често стресни за ученике. Приликом тестирања могу се надгледати микрофон, камера и екран особе која полаже тест – најпрецизнија анализа добија се комбинацијом сва три извора информација. Многи софтвери омогућавају полагање тестова и на уређајима попут мобилних телефона, али надгледање најбоље функционише на десктоп или лаптоп рачунарима.

У овом раду фокус је био на обради података са камере приликом надгледаља тестирања – дат је преглед техника рачунарског вида које се могу приметити у ту сврху и имплементиран је прототип система који примењује неке од њих. Алгоритми као што су Viola-Jones, HOG и Eigenfaces имали су велики значај за развој области детекције објеката и детекције лица. Ови алгоритми се извршавају веома брзо, али се не понашају довољно добро у условима слабијег осветљења и оријентације објеката (лица) која није фронтална. Ове недостатке превазилазе модернији методи, засновани на дубоким конволуционим неуронским мрежама, који су данас и најзаступљенији. За детекцију објеката најчешће се користе детектори као што су YOLO и SSD. Детекција и препознавање лица такође дају најбоље резултате када се користе неуронске мреже. За праћење покрета главе и очију требало би користити 3D моделе, али задовољавајући резултати се могу добити и уз одређене апроксимације. Детекција разговора може се вршити праћењем покрета усана на снимку са камере (у комбинацији са детекцијом гласовне активности путем микрофона).

При одабиру алгоритама за реализацију прототипа система за надгледање полагања тестова узето је у обзир окружење у коме систем треба да се извршава, и добијени су задовољавајући резултати. Камера има увид у ограничен део простора, тако да поред особе која ради тест може седети неко и помагати јој, а да то не буде примећено. Овај случај би могао бити превазиђен уколико би се захтевао и снимак просторије у којој се полаже тест коришћењем додатне камере, међутим, скенирање собе ученика може се сматрати нарушавањем њихове приватности (због чега је било протеста). Провера да ли је лице „живо“ вршена је на основу бројања трептаја, што може довести до погрешних закључака уколико су некоме очи природно затвореније него просечној особи. Такође, ако особа дужи период гледа на доле, то ће бити исправно детектовано као спорна ситуација, али ће највероватније бити и погрешно детектовано да у том периоду лице није било „живо“ јер неће бити довољно детектованих трептаја – тако да би било боље за ову намену одабрати неку од поузданијих метода.

Системи за аутоматско надгледање полагања тестова дају прилично добре резултате, али се могу наћи начини да се систем превари. Такође, алгоритми који се примењују раде са одређеном грешком, тако да се још увек не можемо у потпуности ослонити на њих (и искључити људски фактор), али чињеница је да су ови системи све напреднији и применљивији у пракси.

Литература

[1] S. Ryan, „Why Live Proctoring of Online Exams Is Becoming Mainstream,“ јун 2022. [Онлајн]. Доступно: https://www.testreach.com/blog-post/proctoring-online-exams.html?fbclid=IwAR0r8C4Cg4BLoVv-VJdbadDFxZk8qAiA_HHEkq7mLrEGkpa-U_7iah8jG10. [Преузето: 01.10.2022].

[2] A. Sharma, „12 Best Proctoring Software for Cheating-Free Online Exams,“ август 2022. [Онлајн]. Доступно: https://www.techjockey.com/blog/proctoring-software-for-online-exam. [Преузето: 01.10.2022].

[3] G. Brown, „Can An Online Exam Detect Cheating?,“ 2022. [Онлајн]. Доступно: https://www.masterteachingonline.com/can-an-online-exam-detect-cheating/?fbclid=IwAR0CTcD3-2ixdQQaBXkrii_eVHSyffVz2AZ4P08q0tc4cSRFCm0-VVLCOPs. [Преузето: 01.10.2022]

[4] Honorlock, „Exclusive Remote Proctoring Features“. [Онлајн]. Доступно: https://honorlock.com/exclusive/. [Преузето: 06.10.2022].

[5] Mercer | Mettl, „Proctored Exams: Secure Online Proctoring With AI-Powered Tools“. [Онлајн]. Доступно: https://mettl.com/online-remote-proctoring. [Преузето: 06.10.2022].

[6] „Computer vision,“ у *Wikipedia, the Free Encyclopedia*. Доступно: https://en.wikipedia.org/wiki/Computer_vision. [Преузето: 23.10.2022].

[7] „Object detection,“ у *Wikipedia, the Free Encyclopedia*. Доступно: https://en.wikipedia.org/wiki/Object_detection. [Преузето: 24.10.2022].

[8] G. Boesch, „Object Detection in 2022: The Definitive Guide,“ 2022. [Онлајн]. Доступно: https://viso.ai/deep-learning/object-detection/. [Преузето: 24.10.2022].

[9] J. Hui, „mAP (mean Average Precision) for Object Detection,“ март 2018. [Онлајн]. Доступно: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173. [Преузето: 04.12.2022].

[10] P. Viola, M. Jones, „Rapid Object Detection using a Boosted Cascade of Simple Features,“ у *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001. Доступно: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf. [Преузето: 26.10.2022].

[11] N. Dalal, B. Triggs, „Histograms of Oriented Gradients for Human Detection,“ у *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005. Доступно: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf. [Преузето: 28.10.2022].

[12] S. Mallick, „Histogram of Oriented Gradients explained using OpenCV,“ децембар 2016. [Онлајн]. Доступно: https://learnopencv.com/histogram-of-oriented-gradients/. [Преузето: 31.10.2022].

[13] R. Gandhi, „Support Vector Machine - Introduction to Machine Learning Algorithms,“ јун 2018. [Онлајн]. Доступно: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Преузето: 31.10.2022].

[14] R. Girshick, J. Donahue, T. Darrell, J. Malik, „Rich feature hierarchies for accurate object detection and semantic segmentation,“ у *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014. Доступно: https://arxiv.org/pdf/1311.2524.pdf. [Преузето: 01.11.2022].

[15] М. Дабовић, И. Тартаља, „Дубоке конволуцијске неуронске мреже – концепти и актуелна истраживања,“ у *Зборник 61. Конференција за електронику, телекомуникације, рачунарство, аутокатику и нуклеарну технику, ЕТРАН 2017*, 2017. Доступно: https://www.researchgate.net/publication/321709294_Duboke_konvolucijske_neuronske_mreze_-_koncepti_i_aktuelna_istrazivanja. [Преузето: 01.11.2022].

[16] R. Girshick, „Fast R-CNN,“ у *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015. Доступно: https://arxiv.org/pdf/1504.08083.pdf. [Преузето: 04.11.2022].

[17] S. Ren, K. He, R. Girshick, J. Sun, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,“ у *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, 2015. Доступно: https://arxiv.org/pdf/1506.01497.pdf. [Преузето: 04.11.2022].

[18] K. He, G. Gkioxarim, P. Dollár, R. Girshick, „Mask R-CNN,“ у *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. Доступно: https://arxiv.org/pdf/1703.06870.pdf. [Преузето: 04.11.2022].

[19] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection,“ у *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. Доступно: https://arxiv.org/pdf/1506.02640.pdf. [Преузето: 09.11.2022].

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A. C. Berg, „SSD: Single Shot MultiBox Detector,“ у *European Conference on Computer Vision (ECCV) (2016)*, 2016. Доступно: https://arxiv.org/pdf/1512.02325.pdf. [Преузето: 10.11.2022].

[21] J. Hui, „SSD object detection: Single Shot MultiBox Detector for real-time processing,“ март 2018. [Онлајн]. Доступно: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06. [Преузето: 10.11.2022].

[22] „Face detection,“ у *Wikipedia, the Free Encyclopedia*. Доступно: https://en.wikipedia.org/wiki/Face_detection. [Преузето: 10.11.2022].

[23] „Facial recognition system,“ у *Wikipedia, the Free Encyclopedia*. Доступно: https://en.wikipedia.org/wiki/Facial_recognition_system. [Преузето: 10.11.2022].

[24] A. Rosebrock, „Liveness Detection with OpenCV,“ март 2019. [Онлајн]. Доступно: https://pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/. [Преузето: 09.11.2022].

[25] A. Rosebrock, „What is face recognition?,“ мај 2021. [Онлајн]. Доступно: https://pyimagesearch.com/2021/05/01/what-is-face-recognition/. [Преузето: 09.11.2022].

[26] M. A. Turk, A. P. Pentland, „Face Recognition Using Eigenfaces,“ у *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1991. Доступно: https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf. [Преузето: 10.11.2022].

[27] „ML | Face Recognition Using Eigenfaces (PCA Algorithm),“ септембар 2021. [Онлајн]. Доступно: https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/. [Преузето: 10.11.2022].

[28] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman, „Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,“ у *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 19, Issue: 7, July 1997)*, 1997. Доступно: https://web.ece.ucsb.edu/~hespanha/published/faces.pdf. [Преузето: 10.11.2022].

[29] T. Ahonen, A. Hadid, M. Pietikäinen, „Face Recognition with Local Binary Patterns,“ у *Computer Vision - ECCV 2004, 8th European Conference on Computer*, 2004. Доступно: https://link.springer.com/content/pdf/10.1007/978-3-540-24670-1_36.pdf. [Преузето: 10.11.2022].
[30] F. Schroff, D. Kalenichenko, J. Philbin, „FaceNet: A Unified Embedding for Face Recognition and Clustering,“ у *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Доступно: https://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf. [Преузето: 01.12.2022].
[31] A. Geitgey, „Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning,“ јул 2016. [Онлајн]. Доступно: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78. [Преузето: 10.11.2022].
[32] S. Mallick, „Head Pose Estimation using OpenCV and Dlib,“ септембар 2016. [Онлајн]. Доступно: https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib. [Преузето: 11.11.2022].
[33] S. Mallick, „Gaze Tracking and Estimation,“ *YouTube*, новембар 2019. [Видео датотека]. Доступно: https://www.youtube.com/watch?v=-lmc2-podgQ. [Преузето: 11.11.2022].
[34] M. Fabien, „Voice Activity Detection“. [Онлајн]. Доступно: https://maelfabien.github.io/machinelearning/Speech4/?fbclid=IwAR2V3yKH5REzQ_LgUDfCkN1X52fkQ10io7H_4Q-iKCrtOcjyhupWxlB4wwQ#. [Преузето: 12.11.2022].
[35] „Speech recognition,“ у *Wikipedia, the Free Encyclopedia*. Доступно: https://en.wikipedia.org/wiki/Speech_recognition. [Преузето: 12.11.2022].
[36] L. Xia, G. Chen, X. Xu, J. Cui, Y. Gao, „Audiovisual speech recognition: A review and forecast,“ децембар 2020. [Онлајн]. Доступно: https://journals.sagepub.com/doi/full/10.1177/1729881420976082#fig3-1729881420976082. [Преузето: 04.12.2022].
[37] OpenCV, „TensorFlow Object Detection API“. [Онлајн]. Доступно: https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API. [Преузето: 28.10.2022].
[38] A. Rosebrock, „Object detection with deep learning and OpenCV,“ септембар 2017. [Онлајн]. Доступно: https://pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/. [Преузето: 20.11.2022].
[39] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollár, „Microsoft COCO: Common Objects in Context,“ у *European Conference on Computer Vision*, 2014. Доступно: https://arxiv.org/pdf/1405.0312.pdf. [Преузето 28.10.2022].
[40] A. Rosebrock, „Face detection tips, suggestions, and best practices,“ април 2021. [Онлајн]. Доступно: https://pyimagesearch.com/2021/04/26/face-detection-tips-suggestions-and-best-practices/. [Преузето: 04.10.2022].
[41] OpenCV, „face_detector“, [Онлајн]. Доступно: https://github.com/opencv/opencv/tree/4.x/samples/dnn/face_detector. [Преузето: 04.10.2022].
[42] Dlib, Face Landmark Detection, [Онлајн]. Доступно: http://dlib.net/face_landmark_detection.py.html. [Преузето: 01.10.2022].
[43] C. Sagonas, S. Zafeiriou, „Facial point annotations“. [Слика]. Доступно: https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/. [Преузето: 01.10.2022].

[44] A. Rosebrock, „Face Alignment with OpenCV and Python,“ мај 2017. [Онлајн]. Доступно: https://pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/. [Преузето: 13.11.2022].
[45] T. Soukupová, J. Čech, „Real-Time Eye Blink Detection using Facial Landmarks,“ у *21st Computer Vision Winter Workshop*, 2016. Доступно: http://vision.fe.uni-lj.si/cvww2016/proceedings/papers/05.pdf. [Преузето: 02.11.2022].
[46] A. Lamé, „Gaze Tracking,“ 2019. [Онлајн]. Доступно: https://github.com/antoinelame/GazeTracking. [Преузето: 14.11.2022].
[47] J. Wiseman, „py-webrtcvad,“ 2016. [Онлајн]. Доступно: https://github.com/wiseman/py-webrtcvad. [Преузето: 24.11.2022].
[48] D. King, „High Quality Face Recognition with Deep Metric Learning,“ фебруар 2017. [Онлајн]. Доступно: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html. [Преузето: 14.11.2022].