Проектна задача по предметот Дигитално процесирање на слика

Давид Гичев
196034

# Едноставен кориснички интерфејс базиран на препознавање на гестови преку камера

## Апстракт

Интеракцијата помеѓу човекот и компјутерот (HCI – Human Computer Interaction) е сè поприсутна. Нешто што пред неколку децении било гледано како научна фантастика, денес е секојдневно користено. Новите технологии кои се развиваат вклучуваат иновативни начини на интеракција со корисникот, меѓу нив се оние кои се базираат на гестови. Во овој труд прво е дадено вовед во ваквите технологии, нивниот развој, примена и потенцијал. Потоа, приложена е сумаризација на проблемите кои се појавуваат кај системите кои користат камера (RGB) за детекција на гестови и нивните решенија. Во овој труд се предложени два методи. Првиот користи класично процесирање на влезната слика. Најмногу внимание се посветува на овој метод поради тесната повзаност која ја има со техниките изучени на овој предмет. Вториот метод се основа на софтвер базиран на машинско учење кој дава информации за скелетот на дланката. Истите потоа се користат во склоп на систем за контрола на кориснички интерфејс. На крај е дадена дискусија околу двата методи, која ги вклучува нивните предности и недостатоци.

## Содржина

**1. Вовед** **2**
**2. Предизвици и поврзана литература** **3**
**3. Предложен метод** **4**
3.1. Детекција на движење 5
3.2. Сегментација на кожа 6
3.3 Препознавање на гестови 8
3.3.1 Изолирање на дланката од останатиот дел од раката 8
3.3.2 Броење колку прсти се кренати 10
3.4 Дефинирање акции и контроли 10
**4. Алтернативен метод** **11**
4.1 Mediapipe Hands 11
4.2 Event handling 12
**5. Дискусија** **14**
**6. Заклучок** **15**

# 1. Вовед

Од уреди кои целосно се контролираат со говор, како сите персонални асистенти, до технологија без допир, новите иновации ги надминуваат сите претходно поставени очекувања. Во периодот на кризата со COVID-19 оваа технологија стана поприсутна и потребна – најчест пример за ова е плаќањето со бесконтактни картички. Системи кои се потполно автоматски и очекуваат од корисникот интеракција со одреден интерфејс ползуваат од ваков тип технологии. Технологија кон која најмногу се гледа во последно време е виртуелната реалност, заедно со аугментираната реалност. Имерзијата на корисникот во тој тип технологија бара поставување на контроли кои се интуитивни и близу до движењата кои самиот ги прави во реалноста. За луѓето најприроден начин за комуникација се гестовите и говорот. Двете се сврзани и ни помагаат да се изразиме најдобро што можеме. Гестовите даваат осет и се алатка со која го опишуваме просторот околу нас. Дури и кога соговорникот не може да нè гледа (пр. при зборување на телефон) сепак гестикулираме бидејќи така комуникацијата ни е олеснета и се чуствуваме дека појасно говориме. Затоа, многу технологии сакаат да го искористат овој јазик со цел полесно управување на нивниот интерфејс. Примери за ова се Kinect системот развиен од Microsoft, Leap Motion, Tap Strap 2, системот за гестови имплементиран во Google Pixel 4/XL, итн. Настрана од уреди кои се поврзуваат со персоналните компјутери, во последната деценија има значително голем развој на софтвер наменет да работи на мобилните уреди. Истиот се користи секојдневно. На секоја од мобилните апликации на социјалните медиуми кои се базираат на споделување слики и видео има функционалност која е зависна од овој тип технологија. Најочигледен пример за ова се филтрите на Instagram, Snapchat и TikTok. Сите се основаат на алгоритми кои даваат информации за цртите на лицето со кои се аугментира сликата во реално време. Освен препознавање на лице, многу од филтрите ги користат положбите на рацете, телото и очите при аплицирање на филтерот. Надвор од примената кај системи за забава, оваа технологија е често појдовна точка за контрола на системи дизајнирани за лица со попреченост. Препознавањето на гестови има огромна примена за лицата кои користат знаковен јазик, служејќи како еден вид преведувач со кој можат да се обратат кон луѓе кои не го познаваат тој јазик. Класичните уреди за влез како глувчето и тастатурата се доволно добри за навигација во 2Д простор, но не толку кога станува збор за 3Д простор. Додека пак движењето на рацете за опис на објекти во 3Д простор е далеку поинтуитивно. Од тука може веднаш да се види примена на оваа технологија во софтвер за дизајн на модели во 3Д простор (CAD) или за навигација во 3Д простор (Google Earth).

Првиот тип решенија на овој проблем се основаат на физички ракавици кои се очекува да бидат носени од корисникот. На нив се поставуваат сензори кои комуницираат со уредот и на тој начин се добива 3Д модел од раката и зглобовите. Проблемите кај овој пристап се должат на неудобноста на ракавицата. Затоа, вториот тип решенија е бесконтактен и се спроведува преку камери. Камерите можат да бидат нормални RGB камери, но за оваа проблематика почесто се користат камери кои можат да детектираат длабочина. Често решенијата се комбинација од повеќе типови камери, бидејќи на тој начин поголемото количество информации ја зголемува прецизноста. Во овој труд единствени податоци со кои ќе работиме се слики добиени од камера (webcam).

# 2. Предизвици и поврзана литература

Можноста за препознавање на гестови е привлечна поради потенцијалната примена која ја има, но истата претставува прилично тежок проблем. Сепак, во пределот на компјутерска визија има многубројни пристапи за решавање на оваа проблематика[4]. Доколку фокусот го насочиме кон техники кои користат видео од RGB камера како влез, можеме поконкретно да ги разгледаме својствените предизвици на ова подмножество пристапи. Првиот предизвик кој произлегува е одвојувањето на објектите кои се движат на сцената. Овој проблем, детекција на движење, е екстензивно проучуван со децении наназад. Ова се должи на неговата голема практична примена – системите за надзор базирани на CCTV камери[1]. Од превеција на криминал, до активирање на аларм како резултат на необично движење, директно се гледаат потенцијалот и можностите кои би се добиле со решавање на овој проблем. Решенијата за овој проблем најчесто се делат во 3 категории и тоа:

1. Background Subtraction
2. Frame Differencing
3. Optical Flow

Решенијата кои припаѓаат во првата категорија[6,7] ги детектираат движењата на објектите преку наоѓање на разликата помеѓу состојбата на сцената без движење и моменталната сцена. Проблемите кои настануваат кај овој пристап се должат на претпоставките кои ги прави овој метод. Главната претпоставка е дека позадината сама по себе не содржи движење, нешто што не може секогаш да се претпостави (пр. дрва во позадината кои постојано се придвижени од ветрот). Промени на осветлувањето исто така претставуваат проблем. Затоа, постојат методи кои чуваат информации за тенденциите кои ги има секој пиксел од сликата – варијацијата на боја, осветленоста и слично. Во овие пристапи пак, се поставува прашањето - “Колку долго треба субјектот да биде стационарен за да стане дел од позадината?”. Доколку даден објект доволно време не покажува знаци на движење, тогаш моментот кога ќе излезе од сцената на негово место ќе остане дупка.

Frame Differencing категоријата ги опфаќа методите кои ја користат разликата помеѓу последните, најчесто 2 до 3, соседни слики за да добијат престава за кои делови од сликата се движат. Проблемот е што при движење на еднобоен објект, само неговите надворешните црти ќе бидат детектирани, со што внатрешниот дел останува необележан. Ова преставува проблем доколку се очекува целиот објект да биде обележан, вклучувајќи ја и внатрешноста.

Третата категорија, optical flow, се базира на следењето на кои делови од сликата се движат преку споредувањето на едни групи пиксели со други. Претпоставката е дека при движење, објектот во соседни позиции ќе содржи слична конфигурација на пиксели, со што може да се направи споредбена анализа и да се заклучи движењето на секој регион пиксели. Проблемот со овој пристап е во огромниот број калкулации кои ги бара. Затоа, генерално не е користен во апликации кои работат во реално време.

Секој од овие методи има свои предности и недостатоци, затоа многу често се користат пристапи кои се комбинација од некои од нив, пример првиот и вториот.

Од кога го имаме регионот од сликата каде веруваме дека има движење, наидуваме на следниот проблем – сегментација на регионот кој ни е од интерес. Конкретно, за детекција на дланка, целта ни е да го екстрактираме делот кој е кожа, а да го изоставиме останатото (пр. ракав, облека, и сл.). Главната информациска добивка доаѓа од бојата. Затоа, кај овие пристапи се користи H каналот од сликата конвертирана во HSV или YcbCr формат.

Наједноставниот и често користен пристап е дефинирањето на интервал вредности кои се својствени за кожата. Истиот потоа се користи за сегментирање на сликата во региони каде има и каде нема кожа. Има неколку проблеми со овој пристап. Првиот е тоа што се очекува бојата на кожата да биде иста, или предвреме корисникот да ја има дефинирано. Освен тоа, често се случува позадината на сцената да има делови кои се обоени во иста боја како кожата, или пак каналот за боја да има позастапно ниво на шум. Затоа простојат адаптивни методи кои го менуваат интервалот на боја кој одговара на кожата додека софтверот е користен[2, 3].

Кога регионот што ни е потребен е издвоен, следи препознавање на позата на дланката. Односно ориентација на дланката, прстите, дали и кои се кренати, итн. Повеќето предложени методи за овој проблем се базираат на машинско учење. Ова е очекувано поради комплексноста на проблемот. Наоѓање на позицијата на секој прст на сликата, заедно со информации за секој од зглобовите, е практично невозможна задача за класичните алгоритми. Дланката и прстите при движење постојано се преклопуваат, а во поголемиот дел од времето некои делови од прстите воопшто не се видливи. Затоа, тука доминираат модели кои се базирани на машинско учење, особено длабоко учење кое вклучува коволуциски слоеви. Податочните множества кои се користат за тренирање содржат RGB слики со зглобовите на прстите обележани со 3Д координати. Еден од најкористените модели за детекција на дланки е Mediapipe, софтвер развиен од Google кој е направен да биде високо перформантен, со цел да може да биде користен и на мобилни уреди. Настрана од детекција на дланки, овој модел вклучува модули за детекција на лице, очи, позиција на тело, коса, детекција и класификација на објекти итн.

# 3. Предложен метод

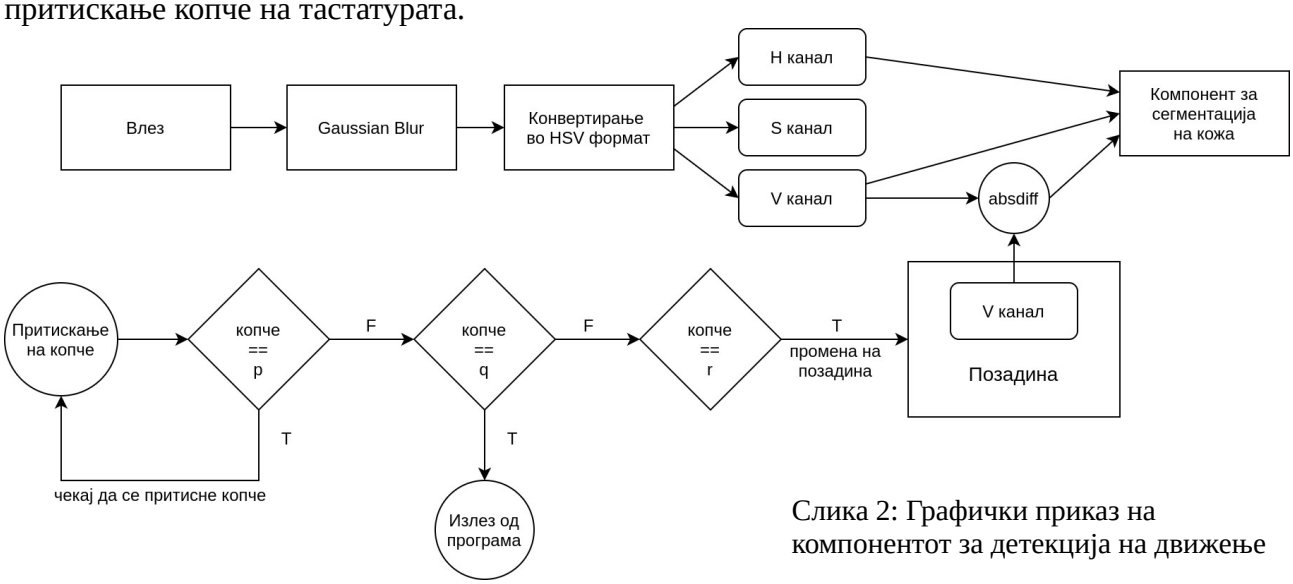
Главниот фокус на трудот претставува овој метод. Одлуката за ова е предводена од тесната поврзаност на методот со алгоритмите и техниките кои се изучени на предметот. Методот е комплетно изграден користејќи ги функционалностите на OpenCV библиотеката.


Слика 1: Графички приказ на предложениот метод

Влезот претставува слика добиена од вградена или екстерна камера поврзана со компјутерот. Во компонентот за детекција на движење се користи референтна слика од позадината која се споредува со добиената слика, со што се добива во кои делови од сликата е детектирано движење. Вториот компонент ги користи овие информации заедно со самата слика за да ја сегментира сликата во региони каде верува дека има присуство на кожа. Важно за компонентот и воопшто методот е што не очекува да му бидат дефинирани параметри кои ја определуваат бојата на кожата. Дополнително, доколку корисникот носи ракавици тогаш соодветно истите ќе бидат детектирани како “кожа”. Следниот компонент го користи овој излез за да ја препознае позата на дланката – бројот на прсти кои корисникот ги има кренато. Понатаму оваа информација е користена за контрола на корисничкиот интерфејс. Ова се прави преку поврзување на состојбата на дланката (бројот на кренати прсти) со соодветна акција. Пример, доколку сите прсти се кренати се очекува да се паузира видеото, ако се кренати 3 прсти тогаш се преминува на следно и сл. Дополнително се користи местоположбата на прстот за движење на глувчето по екранот. Секој од компонентите е детално опишан во следните 3 сегменти.

## 3.1. Детекција на движење

Во оддел бр. 2 се наведени различните пристапи кои се често користени за решавање на овој проблем. Конкретно во контекст на детектирање на гестови, соочени сме со одредени ограничувања кои треба да ги имаме во предвид при избор на метод. Нашата цел е софтверот да работи во реално време, значи третата категорија (Optical Flow) отпаѓа. Дополнително, важно е да можеме да го изолираме целосно објектот кој е во движење, па затоа втората категорија (Frame Differencing) не е задоволителна – поради дупките кои настануваат во склоп на објектот кој се движи. Затоа, како најповолна варијанта е одбрана првата категорија пристапи, односно Background Subtraction. Недостатоците на оваа категорија не се толку изразити во нашиот контекст. Конкретно, ваков систем веројатно би бил користен во внатрешни услови, каде позадината не е постојано во движење. Додатно, се очекува дека ќе има ретки промени во осветлувањето. Сепак, за да се адресираат овие проблеми се воведува опција позадината да може да биде ажурирана во текот на користењето, пр. преку притискање копче на тастатурата.



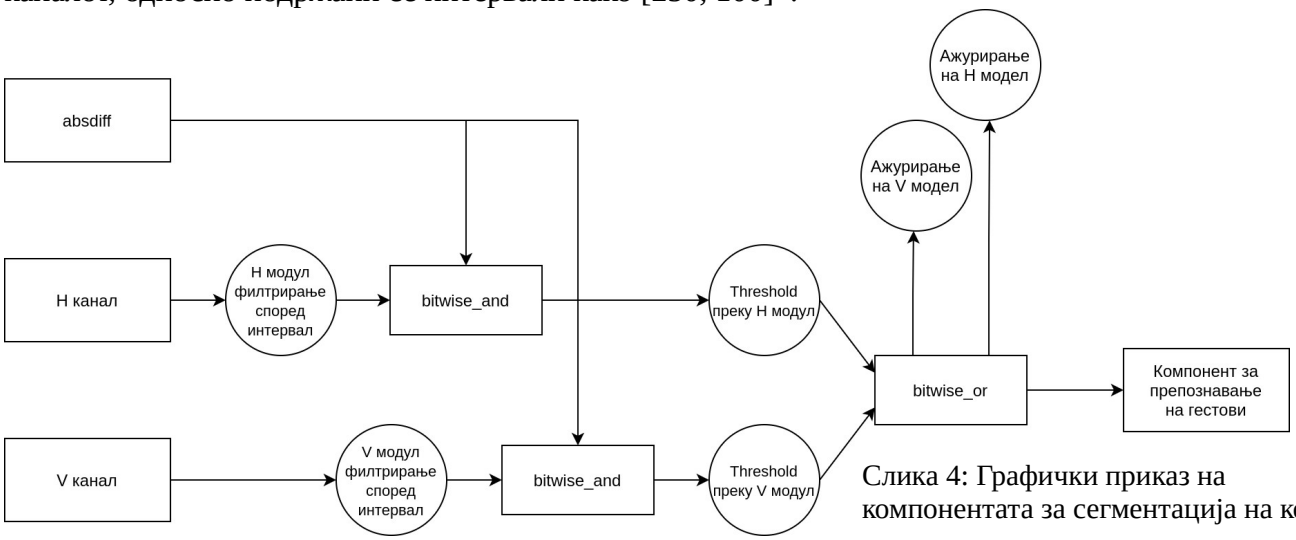

Слика 2: Графички приказ на компонентот за детекција на движење

Влезот, слика во BGR формат, поминува низ Gaussian Blur (cv2.GaussianBlur) со големина на кернел (3,3). Ова се прави за да се редуцира нивото на шум. Следно сликата е конвертирана во HSV формат (cv2.cvtColor) и веднаш потоа разделена на трите канали. За наоѓање на движењето се користи третиот, односно V (value) канал. Најповолен избор е овој канал за детектирање движење бидејќи првиот канал (H – hue) има многу големо количество шум од слика на слика, што го прави следно до бескорисен во овој поглед. Апсолутната разлика од V каналот на моменталната слика и V каналот на позадината ја користиме како индикатор за регионите од сликата каде е присутно движење. Дополнително на дијаграмот (сл. 2) се покажани неколкуте контроли кои се вклучени.

Слика 3: Изглед на absdiff, H каналот и V каналот, соодветно. Може да се забележи зголеменото ниво на шум кое е резултат на вештачкото амбиентално осветлување

## 3.2. Сегментација на кожа

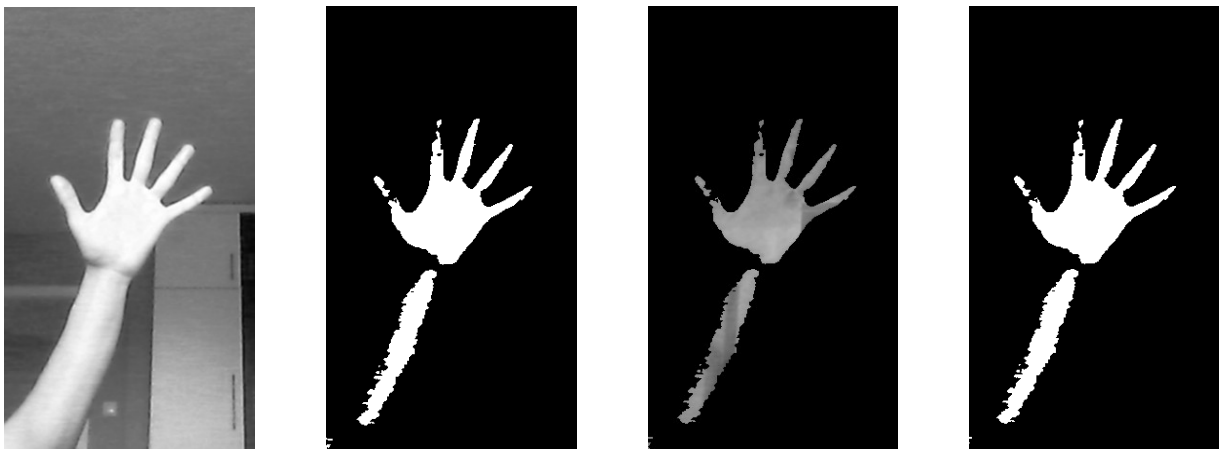
Како што е и споменато во оддел 2, за овој проблем многу пристапи дефинираат статичен интервал на боја на кожа со кој потоа се филтрираат само оние пиксели чиј вредности се во истиот. За овој проблем во предложениот метод се користи адаптивен пристап кој динамички детектира кои делови се кожа, а кои не. Ова овозможува полесно користење на софтверот и поголема робустност и флексибилност. Додатно, интервалот на боја се ажурира на секоја нова слика, со што е можно пример менување на ракавици во реално време, без да настанат проблеми. Една од одликите на H каналот е тоа што по природа е конгруентен (по mod операцијата). Односно, пиксел со вредност 250 е поблизу до пиксел со вредност 0, отколку 200. Ова својство се зема во предвид при операциите со интервали кои се прават врз H каналот, односно подржани се интервали како [250, 100]*.




Слика 4: Графички приказ на компонентата за сегментација на кожа

Како влез во оваа компонента се трите излези (сл. 3) кои ги дава компонентата за детекција на движење, односно: V и H каналите на моменталната слика, заедно со absdiff V канал кој се користи како индикатор за движење. Во следното објаснување се претпоставува дека веќе имаме интервали на боја кои одговараат на кожа, а потоа ќе биде објаснето како се добиваат истите. Оваа компонента се состои од два поткомпоненти (модули) кои се одговорни за V и H, соодветно. Идејата е кожата да ја сегментираме врз основа на осветленоста и бојата која ја има, посебно. За таа цел, делот кој е одговорен за осветленоста како карактеристика на кожата, има сопствени параметри (интервал на доверба) кои одговараат на осветленоста, истото важи и за бојата.

(1) V каналот (2) Кожа во V интервалот (3) bitwise_and со diffV (4) Threshold според V



Слика 5: Сегментација на кожа според модулот за V каналот

* доколку интервалот е од облик [x,y], x > y , тогаш при филтрирањето се спроведува со cv2.bitwise_not(cv2.inRange(img, y, x))

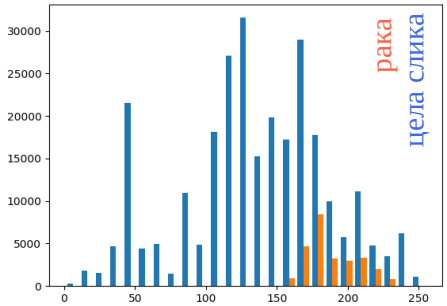
(1) H каналот (2) Кожа во H интервалот (3) bitwise_and co diffV (4) Threshold според H

Слика 6: Сегментација на кожа според модулот за H каналот

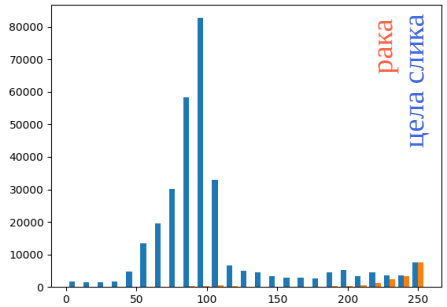
Прво, V каналот е сегментиран на делови со кожа и без, според модулот за осветленост, користејќи ја cv2.inRange функцијата (сл. 5.2). Со ова имаме бинарна слика која ќе ја користиме како маска/филтер. Следува cv2.bitwise_and на маската од претходниот чекор и влезната absdiff слика (сл. 5.3). Врз основа на ширината на интервалот на доверба кој го има модулот за осветленост, се добива колкав треба да биде threshold-от кој ке го користиме за филтрирање на делови кои сме сигурни дека се осветлени како кожата и се во движење. Интуитивно, доколку интервалот е многу голем (пример [0, 255]) тоа би земало и делови од сликата кои не се кожа. Затоа, за да сме сигурни дека деловите што ќе ги земе се сигурно кожа, треба барем да сме сигурни дека истите се во движење. Односно, ако не сме сигурни што е кожа, ќе направиме претпоставка дека деловите што сигурно се движат, веројатно се кожа. За квантификација на оваа сигурност се користи следната формула:

$$\frac{maxT - minT}{255 - minVrange} \cdot (rangeLength - minVrange) + minT$$ (истата врска важи и за H каналот)

Препорачани вредности за maxT, minT, minVrange константите се 100, 40, 40, соодветно.
Добиената вредност ја користиме како threshold кој го аплицираме на сликата добиена од претходниот чекор, со помош на cv2.threshold со параметар cv2.THRESH_BINARY (сл. 5.4). Истата постапка се прави и со модулот за боја на кожа (H каналот, исто се користи absdiff од V, сл. 6). Добиваме две бинарни слики кои ги спојуваме со cv2.bitwise_or, добивајќи слика (combined, сл. 9) која ја содржи кожата во движење. Пред да ја пратиме оваа слика на следниот компонент, прво ќе ја искористиме за ажурирање на параметрите на двата модула, со што ќе ја покриеме претпоставката поставена на почетокот. Добиената бинарна слика ја користиме како филтер кој го аплицираме на V и H каналот од моменталната слика, посебно. Со cv2.bitwise_and се спојуваат маската и H каналот од сликата, со што го добиваме делот од оргиналната слика кој одговара на кожа. Важното е дека маската не е само базирана на H каналот, туку и на V каналот, што значи дека региони кои според H интервалот не биле кожа сега ќе ги има на новодобиената слика. Следува анализа на вредностите на филтрираната слика преку нејзиниот хистограм. Се игнорираат вредности кои се 0 бидејќи тие доаѓаат од филтрирањето.




Слика 7: Хистограм на каналот H



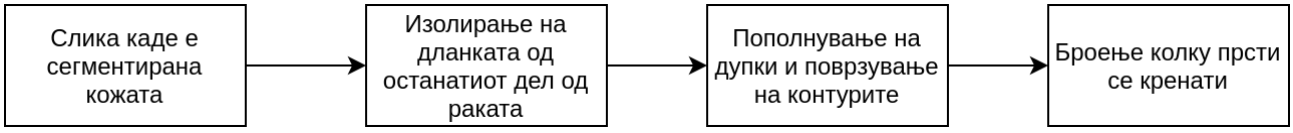

Слика 8: Хистограм на каналот V

Слика 9: Резултат од сегментација

Најчестата боја која е присутна на филтрираната слика би имала највисок врв на хистограмот (сл. 7). По лоцирање на врвот, дефинираме интервал околу истиот. Вака добиениот интервал е новиот интервал според кој ќе се води модулот за боја на кожа. Постапката за модулот за V каналот е идентична. Може да се види како доколку би почнале со интервал [0, 255] би се земале регионите каде сме сигурни дека има движење, чија најчеста боја потоа ќе биде земена како нов интервал. Овој алгоритам практично конвергира поради природата на видео снимките. Односно, бојата на регионите на сликата се менува постепено. Дополнително, користењето на два канали потпомага за одржување рамнотежа. Сликата (combined, сл. 9) ја праќаме кон следниот компонент.

## 3.3 Препознавање на гестови

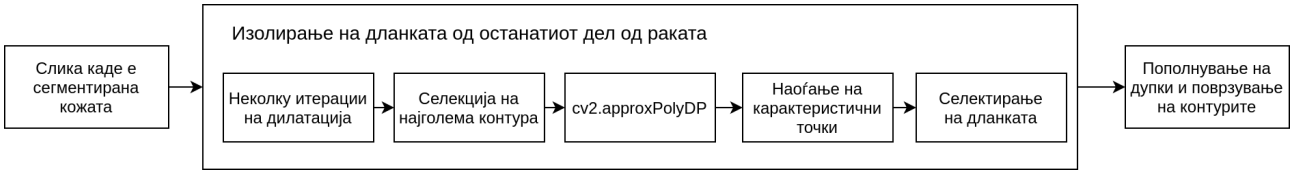
Има големи шанси при користење на софверот, корисникот да го нема покриено поголемиот дел од раката со облека (ракав пример). Резултат на ова е што не може лесно да се открие кој регион од делот што се движи е дланка, а кој е подлактица, рамо итн. Ова може да се реши доколку само одреден дел (квадрат пример) од екранот е обележан и само во него е регистрирано движење, но тоа е рестриктивно и ја лимитира потенцијалната функционалност на системот. Затоа, предложен е пристап кој дозволува движење на раката низ целиот екран.




Слика 10: Графички приказ на целата компонента за препознавање гестови

Првиот дел од овој компонент има задача да ја оддели дланката од останатиот дел од сликата. Се очекува од корисникот да има дефинирано која рака ја користи кога ја започнал програмата. Друга претпоставка е дека корисникот генерално би ја позиционирал раката (не дланката) на страната која ја има дефинирано.

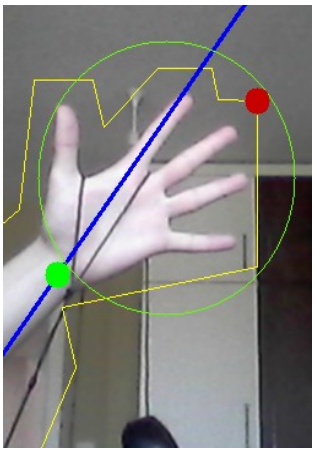
### 3.3.1 Изолирање на дланката од останатиот дел од раката




Слика 11: Графички приказ на процедурата за изолирање на дланката

Првиот чекор е неколкукратна дилатација на пикселите од влезната слика. Ова го правиме за да сме сигурни дека сите прсти се опфатени, дури и во случај каде некои прсти изгледаат визуелно одвоени од дланката (поради недостатоци кај минатите компоненти). На новата слика ги наоѓаме контурите (cv2.findContours) и ги сортираме според плоштина. Ја селектираме најголемата контура, што е всушност многуаголник. Истиот поминува низ cv2.approxPolyDP, па на добиениот поедноставен многуаголник се пресметуваат моментите со cv2.moments.

Првата карактеристична точка која ја добиваме е центроидот, односно центарот на маса на целата рака. Следно, со помош на cv2.fitLine, базирано на линеарна регресија, добиваме линија која ја покажува ориентацијата на раката. Нејзините пресеци со границите на екранот, даваат претстава за почетниот дел од раката (рамото или подлактицата). Сега, во зависност од страната која ја конфигурирал корисникот*, ја наоѓаме најдалечната точка на раката. Доколку е одбрана лева страна, тогаш се гледа Евклидивото растојание од секоја од точките на многуаголникот, до левиот пресек на правата и екранот (ако е десна страна, тогаш десен пресек). Точката со најголемото вакво растојание ја дефинираме како најдалечна точка. Оваа нова карактеристична точка очекуваме да биде во регинот на дланката, односно краевите на прстите. Средишната точка помеѓу двете карактеристични точки (централната и најдалечната) е нова карактеристична точка каде претпоставуваме дека ќе биде почетокот на дланката. Ова не би важело доколку дланката е веќе изолирана, добиената средина точка би била центар на дланката, а не почеток. Затоа, доколку почетниот апроксимиран многуаголник наликува на круг повеќе од правоаголник, постапката треба да се смени. Одлуката за ова е предводена од минималниот однос на висината и ширината на правоаголникот добиен со cv2.minAreaRect(апроксимираниот правоаголник). Доколку тој однос е помал од 1.2 тогаш претпоставувме дека на сликата е видлива само дланката и како почеток ја земаме точката која е најблизу до страната од екранот која ја има одбрано корисникот (слично како процедурата со најдалечна точка само наместо максимум, бараме минимум). Најдалечната точка се бара на ист начин како во иницијалниот алгоритам. Средишната точка во овој случај би била меѓу најблиската и најдалечната точка. Со ова ги имаме дефинирано потребните карактеристични точки (сл. 12). Дланката ја одвојуваме со тоа што го земаме делот од сликата кој е во склоп на кружница со центар во средишната точка и радиус до најдалечната точка. Модулот за сегментација на кожа може да биде ажуриран со сликата добиена во овој чекор.



Слика 12: Карактеристичните точки и фигури

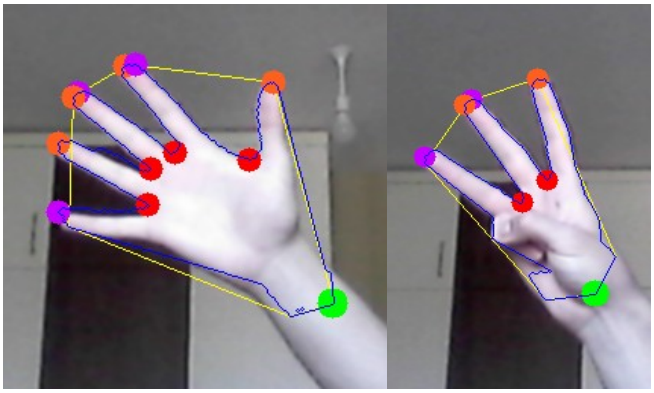
Слика 13: Пред и по поврзување на контурите и одвојување на дланката

Често се случува делови од дланката да не се обележани соодветно и на нивно место да има дупки. Ова е проблем за следниот дел, каде е очекувано дланката да ни е една поврзана целина. Затоа, ги бараме сите контури на добиената дланка. Секоја од составните точки ги поврзуваме со центроидот или најблиската точка, доколку дланката не е сама по себе во оргиналната слика или е, соодветно (сл. 13). Овој чекор е некаков еквивалент на наоѓање на concave hull, бидејќи после него имаме една поврзана целина.

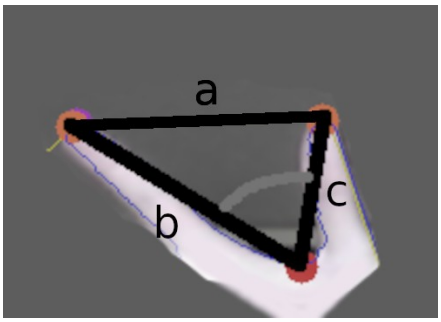
Од кога ја имаме конечната слика можеме повторно да го ажурираме компонентот за сегментација на кожа. На овој начин сегментацијата ќе се врши според бојата на дланката, а не на раката. Така, иако при движење бојата на раката ќе доминира (поради површината), ние сепак ќе го селектираме регионот на дланката (ова дозволува користење ракавици).

### 3.3.2 Броење колку прсти се кренати

Резултат на минатиот чекор е слика од дланката која не содржи дупки, заедно со најдалечната точка на дланката која понатаму е користена за контрола на интерфејсот. За проблемот на броење на прсти најпопуларно решение е користење на convex hull методот[5]. Овој алгоритам, иако не е најточен, е доволно едноставен и ефективен за нашите цели.



Слика 14: Визуелен приказ на convex hull и дефектите
црвено – дното на вдлабнатината
портокалово и виолетово – почетна и крајна точка




Слика 15: Аголот зафатен од крајните точки на даден дефект

Користејќи го cv2.convexHull методот, добиваме многуаголник кој ја обвиткува дланката со што се губат вдлабнатините. Дефектите, односно вдлабнатините на оргиналната фигура, ќе ги користиме за одредување на бројот на прсти. За секоја вдлабнатина, користејќи ги податоците кои ги враќа cv2.convexityDefects ќе го пресметаме аголот зафатен од трите точки:

$$\theta = \arccos\left(\frac{b^2+c^2-a^2}{2bc}\right)$$

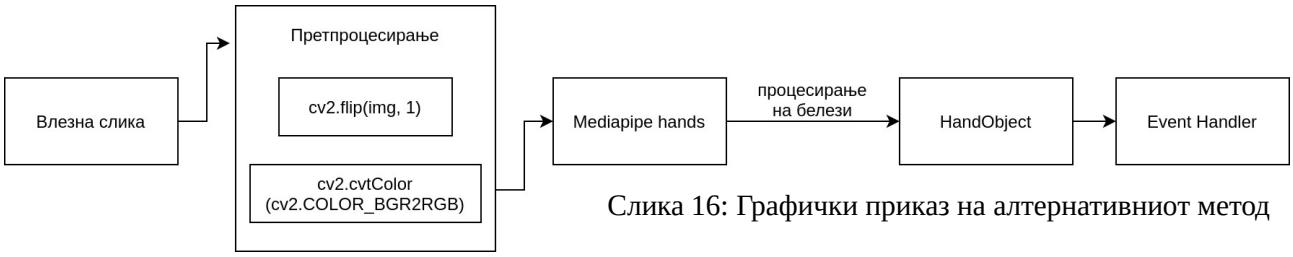
Доколку овој агол е над 90° тогаш не ја земаме таа вдлабнатина во предвид. Дополнително, доколку вдлабнатината не е доволно длабока, тогаш е отфрлена. Бројот на кренати прсти се добива со зголемување на бројот на валидни вдлабнатини за 1. Лимитација на овој метод е што не ја гледа разликата меѓу точно еден и ниеден кренат прст. Со тоа, како резултат можни се 0, 2, 3, 4, 5 кренати прсти.

## 3.4 Дефинирање акции и контроли

Врз основа на резултатите од минатиот компонент, можеме бројот на кренати прсти да го поврземе со одредена акција. Пример, доколку се кренати 3 прсти, тогаш притискаме одредено копче на тастатурата. Доколку има два кренати прсти тогаш го движиме глувчето по екранот, според скалираните координати на најдалечната точка добиена во оддел 3.3.1. За комуникација со оперативниот систем ќе ја користиме python библиотеката pyautogui која нуди контрола на глувчето, тастатурата и приказ на пораки на екранот. Поголемиот дел од акции кои ни се потребни можат индиректно да бидат спроведени преку комбинација од неколку функции. Поради малиот број опции кои ги имаме за гестови (бројот на кренати прсти) не сме во можност да направиме многу корисен систем, дополнително елементите сами по себе се прилично нестабилни и подложни на шум. Во оддел 4.2 е опишано како може да се креира систем кој е далеку пософистициран.

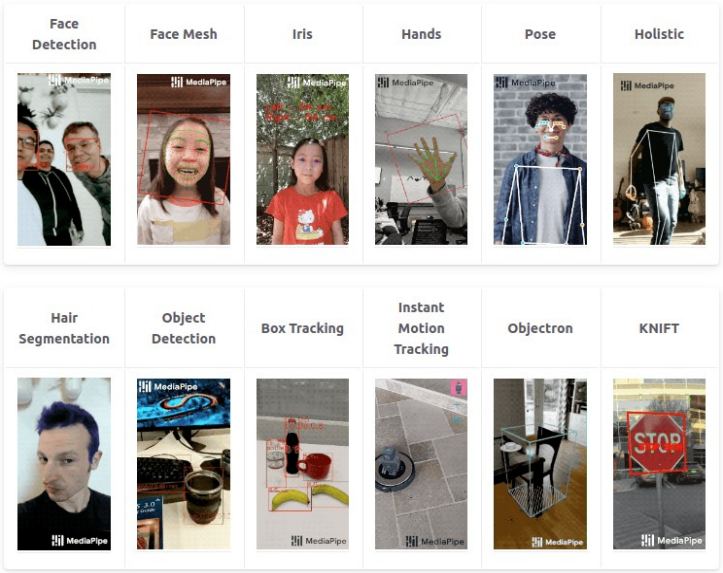
# 4. Алтернативен метод

Во овој дел е опишано како со користење на библиотеката Mediapipe може да се изгради систем за препознавање гестови кој е постабилен и покомплексен. Ова не е главниот метод кој е разгледуван во овој проект бидејќи овој метод е веќе трениран и подготвен, единствено што преостанува е да се искористи излезот за контрола на интерфејсот. Затоа, поголем фокус во овој дел има на системот за гестови (4.2) кој го воведува концептот на контекст и акција која е резултат на низа гестови.



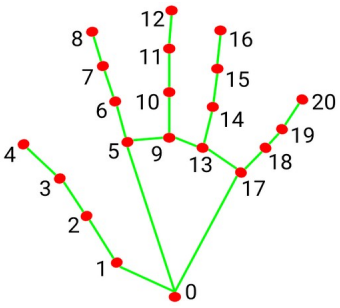

Слика 16: Графички приказ на алтернативниот метод

## 4.1 Mediapipe Hands

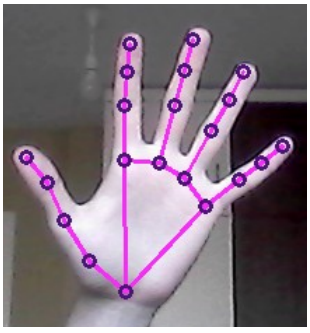



Слика 17: Модулите кои работат со Mediapipe

Mediapipe претставува софтвер со отворен код направен од Google чиј фокус е детекција и класификација на објекти во реално време. Наменет е да работи на повеќето платформи, вклучувајќи ги и мобилните уреди.
Меѓу многубројните решенија кои ги нуди, е модулот за препознавање на дланки. Добивките во перформанси доаѓаат од тоа дека белезите се лоцирани само на почетокот, па потоа се следи движењето на раката со нивна помош. Во случај кога од слика до слика белезите не можат да се поврзат, односно моделот не е сигурен дека сè уште има дланка на сцената, повторно е повикан модулот за класификација за да проба да ја лоцира дланката[8]. Како излез моделот ги дава координатите на белезите на дланката. Координатите се дадени во 3Д (2.5Д) каде x,y одговараат на сликата, а z е релативна длабочина која ја има белегот во споредба со основата на дланката. Секој член од тројката (x,y,z) има вредности кои припаѓаат во [0,1]. Системот е доволно брз за да работи во реално време без никакви проблеми. Дополнително, обележувањето на белезите е стабилно и нема многу варијација од слика на слика. Заедно со деталноста добиена од белезите (21 тројка вредности, секој зглоб од китката е обележан), системот има голем потенцијал за дефиниција на најразлични гестови.




0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP



Слика 18: Белезите кои се дадени како резултат

За да можеме да дефинираме пози/гестови на раката прво треба да ги анализираме белезите. Дефинираме HandObject класа чии инстанци се состојби на раката. Основни карактеристики кои можат да се запишат за секоја состојба се:

- ориентацијата на раката - (аголот на векторот од #0 до #9)
- координати на врвот од показалецот (#8)
- низа од пет објекти од типот FingerObject кои ќе содржат информации за секој од прстите

Притоа, секој FingerObject содржи информации добиени од трите точки кои одговараат на соодветниот прст (#1, #2, #3, #4 за првиот, #5, #6, #7, #8 за вториот, итн.). Некои од нив се:

- релативната ориентација на прстот – агол на векторот од почетниот до крајниот зглоб на прстот
- дали прстот е кренат

Втората карактеристика се одредува на следниот начин:
Нека z е низата зглобови z = [$z_0$, $z_1$, $z_2$, $z_3$] каде секој $z_i \in (x, y)$, за $x, y \in [0,1]$
Тогаш $p_0 = z_1 - z_0$ , $p_1 = z_3 - z_1$. Аголот помеѓу $p_0$ и $p_1$ може да се дознае преку нивниот скаларен производ:

$$p_0 \cdot p_1 = |p_0||p_1|\cos(\theta)$$

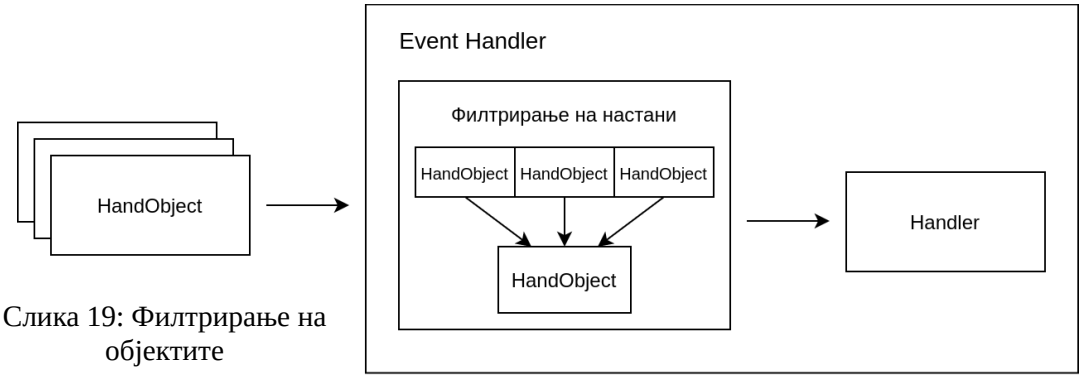
$$\cos(\theta) = \frac{p_0 \cdot p_1}{|p_0||p_1|}$$

$$\theta = \arccos\left(\frac{p_0 \cdot p_1}{|p_0||p_1|}\right)$$

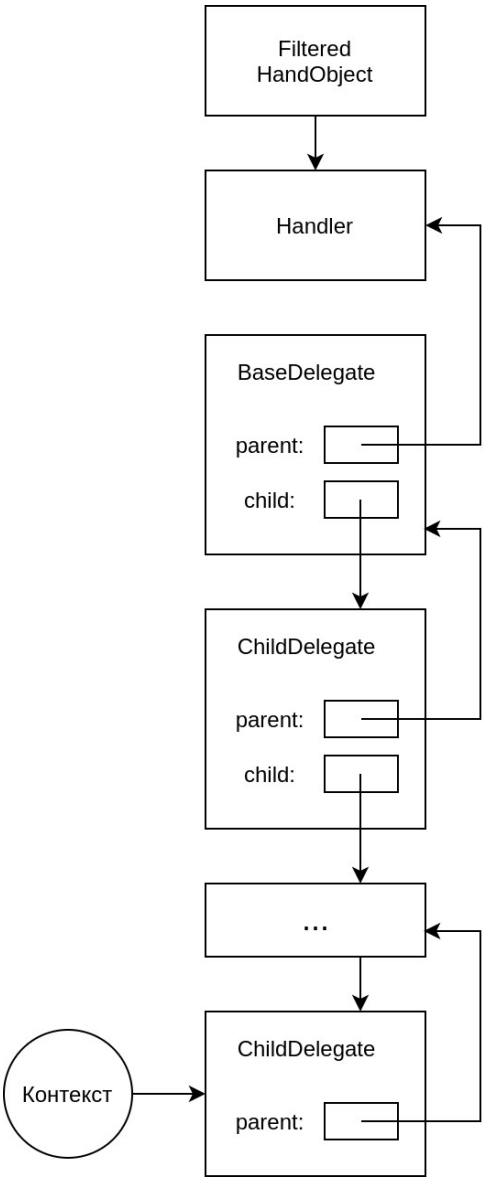
Доколку оваа вредност е помала од 0.4, тогаш велиме дека прстот е кренат. Овој алгоритам не зависи од ориентацијата на раката - што значи дека може странично, или наопаку, да биде поставена и сепак состојбата на прстите ќе биде коректно класифицирана.
Со помош на овие карактеристики можеме да ја дефинираме позата која ја има раката. Пример, доколку се кренати само вториот и третиот прст, тогаш таа поза можеме да ја дефинираме како “peaceSign”, или ако е само кренат првиот прст тогаш “thumbsUp”. Математички има $2^5 = 32$ вакви пози, но само најлесните можеме да ги имаме дефинирано. Оваа одлика ја додаваме како атрибут на инстанцата HandObject која одговара на моменталната состојба. Овој објект го праќаме на компонентата за справување со настани.

## 4.2 Event handling

На располагање имаме голем број пози кои може да ги има раката. Ова ни дозволува да направиме систем кој вклучува разновиден број на контроли. Пред HandObject (гестот) да биде пратен на справувачот со настани, прво се додава во редица. Секои три гестови од редицата се споредуваат – доколку позата на раката е иста кај сите три, тогаш се праќа последниот од нив понатаму. Ако не, тогаш се отфрла првиот од нив и се чека нов гест да биде додаден на редицата. Иако системот е точен, овој механизам е имплементиран бидејќи при транзиција од еден на друг гест се случува во движењето да бидат детектирани гестови кои не сме имале намера да ги направиме.




Слика 19: Филтрирање на објектите



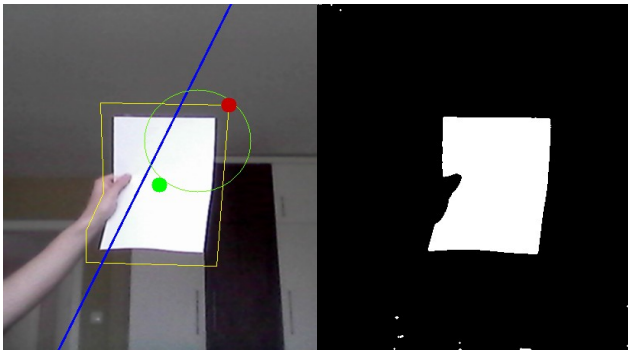

Слика 20: Графички приказ на структурата која ја користи справувачот со настани

Главниот принцип на овој систем е идејата на контекст. Односно, истиот гест може да значи различни работи во зависност од кога е направен. Пример, гестот “thumbsUp” би можел да значи паузирање доколку имаме отворено видео, а би можел и да значи започнување на презентација доколку имаме отворено Powerpoint.
За таа цел се воведува концептот на контекст. При извршување на одреден гест, контролата се префрла на друг справувач (handler) кој ќе се справува со сите понатамошни гестови – сме го сменile контекстот. Доколку од справувачот се побара да биде процесиран гест за кој нема процедура (справувачот не го очекува тој гест) тогаш справувачот заминува од сцената и контекстот се префрлува на неговиот родител (претходниот контекст). Справувачите во имплемтацијата се референцирани како делегати, бидејќи се повикани од основниот справувач. Секој нов настан е пренесен по дрвото (или листата, сл. 20) до последниот делегат каде е соодветно процесиран. Пример за вакво разгранување е TestDelegate-от кој се активира со позата “spiderman” (или “те сакам” на знаковен јазик). По правењето на оваа поза, се пренесува контекстот на делегатот и секоја наредна поза е процесирана од него. На екранот се покажува името на секоја направена поза, во посебен прозорец. За да се вратиме на основниот контекст треба да се направи истиот гест повторно (“spiderman”). Може да се забележи дека со овој начин на работа истиот гест може да има повеќе значења во зависност од контекстот. Со ова се проширува множеството на контроли без да има потреба од дефинирање на нови пози. ChainDelegate е друг тип делегат кој доколку детектира низа од пози по спефицичен редослед, како резултат ќе даде акција специфична за таа комбинација пози (пр. покренување на одредена апликација).

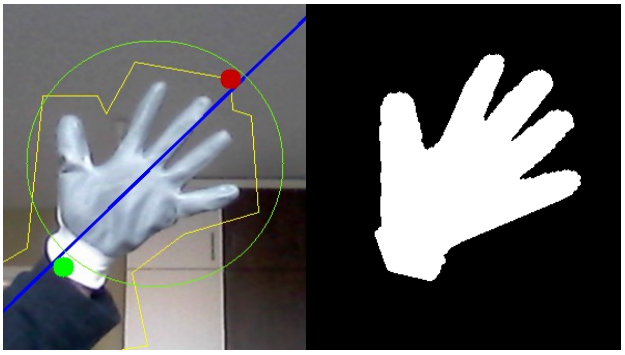
Доколку ChainDelegate види поза која не е очекувана тогаш сите направени пози ги препраќа на родителот. Сега, доколку корисникот не е свесен дека низа од пози предизвикуваат одредена акција и направи 3 од потребните 5 пози по случајност, заедничката акција нема да се исполни, но трите направени пози сепак ќе бидат пратени за процесирање на претходниот контекст. Акциите се спроведуваат преку истата библиотека спомената во 3.4 (pyautogui) заедно со subprocess.

# 5. Дискусија

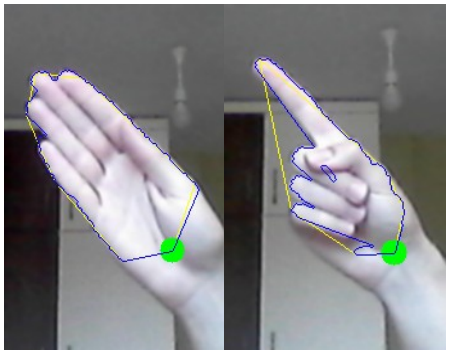
Проблемите кои се присутни во предложениот модел најчесто се резултат на еден од составните негови компоненти. Доколку во позадината има често движење или промени на амбиенталното осветлување, тогаш не се задоволени претпоставките кои ги имавме направено во 3.3.1, при вакви услови се очекува алгоритмот да даде погрешни резултати (затоа се препорачува повремено ажурирање на позадината). Доколку позадината е многу блиска со бојата на кожата тогаш одредени делови од дланката ќе бидат изоставени бидејќи absdiff (објаснето во 3.2) нема да дозволи сегментираната кожа да биде прикажана. Сепак, повеќето од времето ова е поправено во последниот чекор од 3.3.1. Во случај кога не е, може да се отвори процеп кој понатаму ќе направи нов дефект (сл. 24) во convex hull-от кој се користи за броењето прсти (3.3.2). Овие недостатоци се случуваат кога прстите се поставени пред дланката, бидејќи тогаш има најмногу сенки. Сепак, методот е направен да е флексибилен (сл. 21, 22) и да не бара многу предуслови за користење, па очекувано е да не работи перфектно поради проблемите кои доаѓаат со релаксирање на предусловите. Покрај се тоа, излезот кој го добиваме е премногу лимитирачки за да биде користен како основа за корисен систем за контрола со гестови.



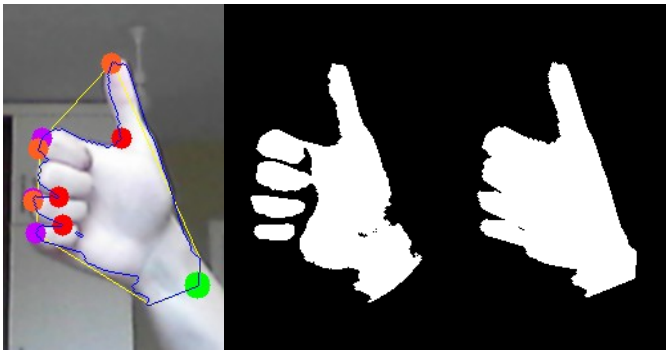
Слика 21: Листот е одбран како дланка



Слика 22: Користење на ракавица



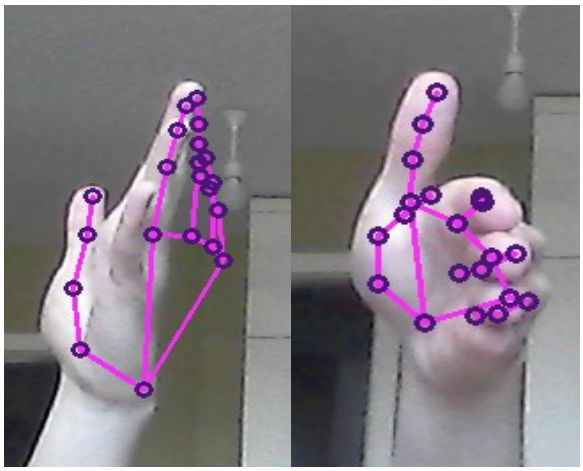
Слика 23: Помал број на прсти од очекувано



Слика 24: Поголем број на прсти од очекувано

Алтернативниот метод е неспоредливо подобар и ги нема горенаведените пропусти. Неговата стабилност и точност е на сосема друго ниво, па затоа и има голема практична употреба. Излезот е богат и претставува одлична основа за систем кој може да подржи големо множество гестови. Преку предложениот систем може да се види потенцијалот на ваков тип софтвер.

Страничното позиционирање на дланката може да направи проблеми за алтернативниот модел, како и други конфигурации каде е тешко да се оцени кој прст на која позиција е (сл. 25). Дополнително, носењето ракавица не е подржано од моделот (сл. 26).



Слика 25: Погрешно позиционирани белези

Слика 26: Користење на ракавица со Mediapipe моделот

# 6. Заклучок

Препознавањето на гестови има голема примена и претставува нов начин на интеракција со корисничките интерфејси кој е природен и интуитивен за луѓето. Постојано има обиди за решавање на овој проблем - некои од нив веќе се комерцијално достапни и се користат секојдневно. Во овој труд се разгледани тешкотиите со кои се соочуваме при имплементација на ваков систем. Како и многу други проблеми во пределот на компјутерска визија, примена наоѓа машинското учење. Со него може да се постигне многу повисока точност, но не и за цена на намалени перформанси. Во последно време се посветува повеќе внимание на софтвер базиран на машинско учење кој може да работи и на мобилни уреди, што само ја зголемува неговата користеност и примена. Дополнително овој тип софвер е предуслов за системи кои се базираат на виртуелна реалност (VR) и аугментирана реалност (AR). Овој дел од компјутерската визија ќе остане актуелен и проучуван поради потенцијалот кој го има за подобрување на интеракцијата човек-компјутер.

# Референци


[1] Kanade, Takeo & Collins, Robert & Lipton, Alan & Fujiyoshi, Hironobu & Duggins, David. (1999). A System for Video Surveillance and Monitoring CMU VSAM Final Report. 135.

[2] Stergiopoulou, Ekaterini & Sgouropoulos, Kyriakos & Nikolaou, Nikos & Papamarkos, Nikos & Mitianoudis, Nikolaos. (2014). Real Time Hand Detection in a Complex Background. Engineering Applications of Artificial Intelligence. 35. 10.1016/j.engappai.2014.06.006.

[3] Tofighi, Ghassem & Afarin, Nasser & Raahemifar, Kamraan & Venetsanopoulos, Anastasios. (2014). Hand Pointing Detection Using Live Histogram Template of Forehead Skin. International Conference on Digital Signal Processing, 10.1109/ICDSP.2014.6900691.

[4] Oudah, Munir, Ali Al-Naji, and Javaan Chahl. 2020. "Hand Gesture Recognition Based on Computer Vision: A Review of Techniques" Journal of Imaging 6, no. 8: 73. https://doi.org/10.3390/jimaging6080073

[5] Dhawan, Amiraj & Honrao, Vipul. (2013). Implementation of Hand Detection based Techniques for Human Computer Interaction. 10.5120/12632-9151.

[6] Benezeth, Yannick & Jodoin, Pierre-Marc & Emile, Bruno & Laurent, Hélène & Rosenberger, Christophe. (2008). Review and Evaluation of Commonly-Implemented Background Subtraction Algorithms. 1-4. 10.1109/ICPR.2008.4760998.

[7] Benezeth, Yannick & Jodoin, Pierre-Marc & Emile, Bruno & Laurent, Hélène & Rosenberger, Christophe. (2010). Comparative study of background subtraction algorithms. Journal of Electronic Imaging. 19. 033003-033003. 10.1117/1.3456695.

[8] Zhang, Fan & Bazarevsky, Valentin & Vakunov, Andrey & Tkachenka, Andrei & Sung, George & Chang, Chuo-Ling & Grundmann, Matthias. (2020). MediaPipe Hands: On-device Real-time Hand Tracking.