KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Rytis Augustauskas

PLAŠTAKOS GESTŲ ATPAŽINIMO SISTEMOS SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas

Vadovas
Doc. Arūnas Lipnickas

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

PLAŠTAKOS GESTŲ ATPAŽINIMO SISTEMOS SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas
Valdymo technologijos (kodas 621H66001)

Vadovas
Doc. Arūnas Lipnickas

Recenzentas
Lekt. Kęstas Rimkus

Projektą atliko
Rytis Augustauskas

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos

(Fakultetas)

Rytis Augustauskas

(Studento vardas, pavardė)

Valdymo technologijos (621H66001)

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Plaštakos gestų atpažinimo sistemos sukūrimas ir tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 17 m. gegužės 26 d.

Kaunas

Patvirtinu, kad mano, **Ryčio Augustausko,** baigiamasis projektas tema „Plaštakos gestų atpažinimo sistemos sukūrimas ir tyrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Augustauskas, Rytis. Plaštakos gestų atpažinimo sistemos sukūrimas ir tyrimas. Valdymo sistemų magistro baigiamasis projektas / vadovas prof. Arūnas Lipnickas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, Automatikos katedra.

Mokslo kryptis ir sritis: Elektros ir elektronikos inžinerija, Technologiniai mokslai

Reikšminiai žodžiai: Gestai, gestų atpažinimas, rankos gestai, OpenCV, 3D duomenys, CUDA

Kaunas, 2017. 49 psl.

SANTRAUKA

Pastaruoju metu gestų komunikacija pradedama vis plačiau taikyti robotikoje bei virtualioje realybėje. Yra keletas būdų kaip atpažinti valingą rankos judesį ar atskirti gestą, naudojant skirtingus jutiklius. Ranka gali būti atpažinta tik su RGB kamera, tačiau tokį metodą riboja apšvietimas ir skirtingi odos atspalviai. Iš gylio kameros duomenų lengviau aptikti objektus, esančius prieš kamerą. Šiame darbe aprašytas žmogaus ir jo rankos gesto atpažinimo algoritmas naudojant RGB ir gylio kamerų duomenis. Minėtas būdas iš pradžių aptinka žmogaus veidą, randa atitinkamus jo taškus gylio kadre ir naudoja regioną tarp žmogaus ir kameros rankos paieškai ir gesto atpažinimui.

Augustauskas, Rytis. Development and Analysis of Hand Palm Gesture Recognition System: Master‘s thesis in Control systems / supervisor doc. Arūnas Lipnickas; Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, department of Automation.

Research area and field: Electrical and Electronics Engineering, Technological Sciences

Key words: Gestures, gestures recognition, hand gestures, OpenCV, 3D data, CUDA

Kaunas, 2017. 49 p.

SUMMARY

Nowadays there is a big interest to apply natural gesture communication language to various robotics systems and virtual reality. In fact, many techniques to detect willed hand motion or to recognize gestures using a variety of sensors already exist. RGB cameras can detect hands and hand motion, but its application becomes limited due to the lighting conditions or due the diversity of skin colours. With data provided by a depth camera, it is possible to distinguish objects in front of a camera with much more robustness. This study presents a technique to detect an active user and their correspondent hand gesture using both RGB and depth data. This method firstly detects the user's face then maps it to depth map data. Finally, the detection of human arm and the recognizance of hand palm is performed in the region between the camera and the person.

Turinys

Įvadas

Pastaruoju metu kompiuterinė rega ir vaizdo atpažinimo bei apdirbimo algoritmai vis plačiau taikomi įvairiausiose srityse: medicininių vaizdų analizėje, brokuotos produkcijos, atsiradusios gamybos metu, aptikimui, kokybės kontrolėje, apsaugos ir sekimo sistemose bei kitur. Ypatingai didelės panaudojimo galimybes kompiuterinė rega turi paskutiniaisiais metais labai išpopuliarėjusiose virtualiosios [1] (*angl*. virtual reality) ir papildytos [2] (*angl*. augmented reality) realybės projektuose. Taip pat, žmogaus rankų judesių atpažinimas ir interpretavimas yra svarbi minėtų sričių dalis. Pasitelkiant kamerų duomenis, galima išskirti žmogaus rankų gestus ir interpretuoti juos kaip konkretų valdymo signalą. Tai nereikalauja jokios sąveikos su fiziniai objektais, tokiais kaip mygtukai, vairalazdės, valdymo pultai ir kt.

Bekontaktis sistemų valdymas gali žymiai palengvinti kasdienines operacijas, atliekamas su bet kokiais išmaniaisiais prietaisais: kompiuteriu, telefonu, televizoriumi ar kitais. Tokie veiksmai, kaip teksto įvedimas klaviatūra, kompiuterio pelės rodyklės slankiojimas ar konkrečios programos valdymas gali būti išreikštas žmogaus rankų gestikuliacija. Be to, sąsaja su įvairiomis išmaniomis sistemomis, pavyzdžiui, robotais, ateityje gali būti paremta ne komunikaciniu protokolu, o žmogaus rankų gestais. Tai bene vienas natūraliausių ir interaktyviausių valdymo būdų.

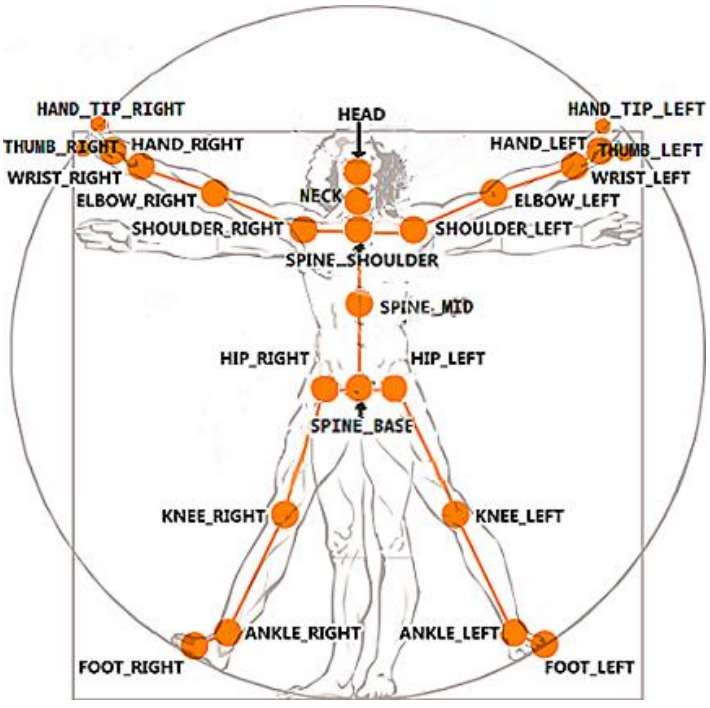
Modernūs jutikliais dar labiau palengvina šios problemos realizavimą sprendimą. Pavyzdžiui, gylio bei RGB kameros kaip Microsoft Kinect, Intel® RealSense™, DepthSenseDS325, Creative Senz3D ar kita suteikia daugiau informacijos apie prieš kamerą esančius objektus. Iš gylio kadro galima gauti kiekvieno taško atstumą iki kameros, išskirti prieš ją esančius objektus. Be to, vertinant duomenis iš RGB kameros, galima nusakyti, ar rodantis gestus žmogus žiūri į kamera, tai yra, ar dėmesys sutelktas į sistemos valdymą. Taip pat, žmogaus atpažinimas gali būti savotišku atskaitos tašku. Radus veidą RGB kameros kadre, galima nustatyti atitinkamus gylio kameros taškus. Tokiu būdu galima išskirti tik tam tikrą regioną, kur yra asmuo, atmetant kitus nereikalingus duomenis. Kombinuojant šiuos informaciją iš RGB ir gylio kamerų bei pasitelkiant atitinkamus vaizdo atpažinimo algoritmus, galima pasiekti žymiai geresnį rezultatą negu naudojant tik vieną jutiklį.

Šiame darbe kuriamas ir tiriamas statinių gestų atpažinimo algoritmas minėtuoju būdu, tai yra pasitelkiant RGB ir gylio kamerų duomenis. Taip pat, palyginti keli žmogaus veido atpažinimo būdai ir pateiktas jų greitaveikos ir tikslumo palyginimas.

1. Analitinė dalis

Didėjant kompiuteriniams resursams, kompiuterinės regos taikymas tampa įmanomas įvairiose srityse. Tai gali būti sekimo, kokybės kontrolės, medicininių vaizdų analizės ar kitokios sistemos. Taip pat, ne ką mažesnė sritis yra žmogaus-mašinos sąsaja - gestų atskyrimas, naudojant vaizdo atpažinimo algoritmus. Tai bene natūraliausia nekontaktinė valdymo forma, kuri ateityje gali pakeisti įvairius įvesties įrenginius. Šis valdymo būdas gali žymiai palengvinti kasdienines operacijas, atliekamas su bet kokiais išmaniaisiais prietaisais: kompiuteriu, telefonu, televizoriumi ar kitais. Tokie veiksmai, kaip teksto įvedimas klaviatūra, kompiuterio pelės rodyklės slankiojimas ar konkrečios programos valdymas gali būti įgalintas žmogaus gestikuliacija.

Jau galima rasti įvairios rankų gestų atpažinimą atliekančios programinės įrangos. Dažniausiai tai yra gylio kameroms pritaikyti algoritmai. Pavyzdžiui, ***Microsoft KinectV2*** su savo programinės įrangos paketu [3],[4] gali išduoti keletą skeletono taškus (1.1 pav.), tai yra, jei asmuo stovi prieš kamerą, jutiklis gali nustatyti atitinkamų kūno dalių vietas kadre. Šiuo atveju aptikti atitinkami plaštakos taškai (HAND_LEFT / HAND_RIGHT), plaštakos viršaus (HAND_TIP_LEFT / HAND_TIP_RIGHT) bei nykščių (THUMB_LEFT / THUMB_RIGHT), gali padėti nustatyti žmogaus gestą, ar bent rankos padėtį, bet negalima identifikuoti visų pirštų.



1.1 pav. *Microsoft Kinect SDK išduodami skeletono taškai* [4]

Kita, visiškai pirštų bei rankų sekimui pritaikyta programinė įranga, yra ***Leap Motion SDK*** [5], kuri yra pritaikyta tik gamintojo gylio kamerai. Šis jutiklis dažniausiai naudojama su virtualios realybės

akiniais (yra montuojama prie akinių), bet ją galima naudoti ir atskirai (prijungti prie kompiuterio kaip parodyta 1.2 pav.). ***Leap Motion*** išduoda tik gylio kadrą.

a) **b)**

1.2 pav. *Leap Motion naudojimas. a – pritaisyta prie virtualios realybės akinių*[1]*, b – prie kompiuterio prijungta kamera*[2]

Ši kamera veikia artimame diapazone, 25-600mm nuo jutiklio. Kaip parodyta 1.2 paveikslo b) dalyje, ***Leap Motion SDK*** paketas suteikia informaciją apie prieš kamerą esančio objekto (rankos bei pirštų) padėtį.

Dar viena programinė įranga - ***Close Interaction Library*** [6], kuri, panašiai kaip ir ***Leap Motion SDK***, naudodama gylio kadro duomenis, aptinka delno centrą, pirštų galų ir dilbio centro taškus. Programinė įranga veikia su ***Softkinetic*** šeimos kameromis [7]. Šio tipo jutikliai gali būti tvirtinami prie virtualios realybės akinių, taip pat, gali būti naudojami atskirai, kaip parodyta 1.3 paveiksle.

a) **b)**

1.3 pav. *a – prie virtualios realybės akinių pritvirtinta kamera* [6]*, b – pritvirtinta prie ekrano viršaus*[3]

Taip pat, pastaruoju metu atsirado programinės įrangos bibliotekos ***Caffe*** [8] ir ***Microsoft Cognitive Toolkit*** [9], galinčios realizuoti giliojo mokymo taikymą rankos sekimui ir gestų atpažinimui.

---

[1] http://www.robotshop.com/en/leap-motion-3d-motion-controller.html

[2] http://dulce.com/blog/leap-motion-orion-dulce-dotcom/

[3] http://www.electronicdesign.com/embedded/consumer-electronics-take-user-interfaces-beyond-your-fingertips

Vienas iš rankos gestų atpažinimui pritaikytų projektų, naudojant ***Caffe***, yra ***Deep-Hand*** [10]. Giliojo mokymo modelis yra parengtas, naudojant gestų kalbos vertėjų vaizdo įrašus (1.4 pav.). Mokymo bazė sudaro 1 milijonas kadrų [11]. Ši kiek aukštesnio lygio programinė įranga naudoja RGB kameros kadrą rankos aptikimui ir gesto atpažinimui.

1.4 pav. *Deep-hand projekto mokymo bazės kadras* [10]

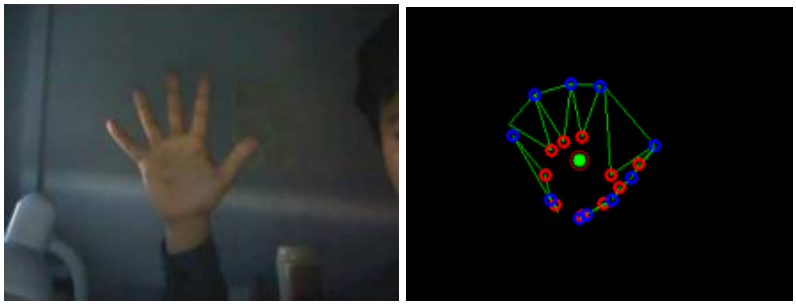
Visos išvardintos programinės įrangos sprendžia rankos aptikimo ir gesto atpažinimo problemą, tačiau turi ir trūkumų. Pavyzdžiui, visoms gylio kameroms pritaikytos išvardintos bibliotekos yra taikytinos tik to gamintojo jutikliams: ***Microsoft Kinect SDK*** V2.0 yra skirta tik antros serijos ***Kinect***, ***Leap Motion SDK*** – ***Leap Motion*** gylio kamerai, ***Close Interaction Library*** – ***Softkinetic*** įrenginiams. Taip pat, iškyla problemų tarp operacinės sistemos pasirinkimo ir architektūros. ***Microsoft Kinect SDK*** veikia tik Windows aplinkoje, ***Close Interaction Library*** yra pritaikyta tik 32 bitų architektūrai, o ***Softkinetic DS325/DS525*** kameros, anot gamintojo, nėra palaikomos ***Windows 10*** aplinkoje. Be to, pastaroji biblioteka paskutinį kartą atnaujinta 2015 metais ir yra pritaikyta tik senesniems kompiliatoriams, šiuo atveju ***Visual Studio 2013***. Šie trūkumai neleidžia naudoti kai kurių technologijų. Vienas didžiausių trūkumų – nevisiškas algoritmų pagreitinimas su ***CUDA*** [12], kas gali lemti ne pilną papildomų bibliotekų integravimą, pavyzdžiui ***DLib*** [13] be algoritmų pagreitintų su vaizdo plokšte. Paskutinioji iš minėtų rankos gestų atpažinimo programinių įrangų (***Deep-Hand***) yra visiškai kitokia technologija, kiek lankstesnė nei kitos, kadangi ji nesusieta su konkretaus gamintojo kamera. Kaip jau minėta, ji atpažinimui naudoja spalvinį RGB kadrą. Įrankis gana tiksliai gali nustatyti rodomą gestą, tačiau tai reikalauja didelių skaičiavimo resursų. Būtent dideli kompiuterinių pajėgumų reikalavimai apriboja šio giliojo mokymo metodo naudojimą programinės įrangos, skirtos paprastam vartotojui, kūrime. Be to, vertinant tik RGB informaciją, negalima tiksliai nustatyti rankos padėties kameros atžvilgiu, kaip tai daroma su gylio kamera.

Taip pat, galima rasti nemažai straipsnių apie rankos segmetavimą ir gestų atpažinimą. Dalyje tyrimų ranka išskiriama pagal odos spalvą [14], [15], [16]. Šiuose tyrimuose analizuojamas spalvinis

RGB kadras, iš visų objektų esančių nuotraukoje bandoma išskirti žmogaus plaštaką. Pavyzdžiui, viename straipsnyje [15] pateikiama keletas odos segmentavimo būdų. Keliuose iš jų naudojamos RGB ir HSV spalvų gamos, turint duomenų bazę su galimomis odos spalvų variacijomis. Kai ranka išskiriama kartu su veidu, pastarasis lokalizuojamas su ***Haar*** klasifikatoriumi [17] ir iškerpamas iš nuotraukos (1.5 pav.).

1.5 pav. *Veido pašalinimas iš analizuojamo kadro*[15]

Darbe taip pat pateikiama pirštų galų išskyrimo metodika taikant ***convex hull*** [18] funkciją ir tiriant nesutapimus (įdubas) tarp apibrėžto daugiakampio ir rankos kontūro (1.6 pav.). Mėlynieji taškai – įdubos pradžia ir pabaiga, raudonieji – didžiausios įdubos (labiausiai nuo apibrėžto daugiakampio nutolę taškai).



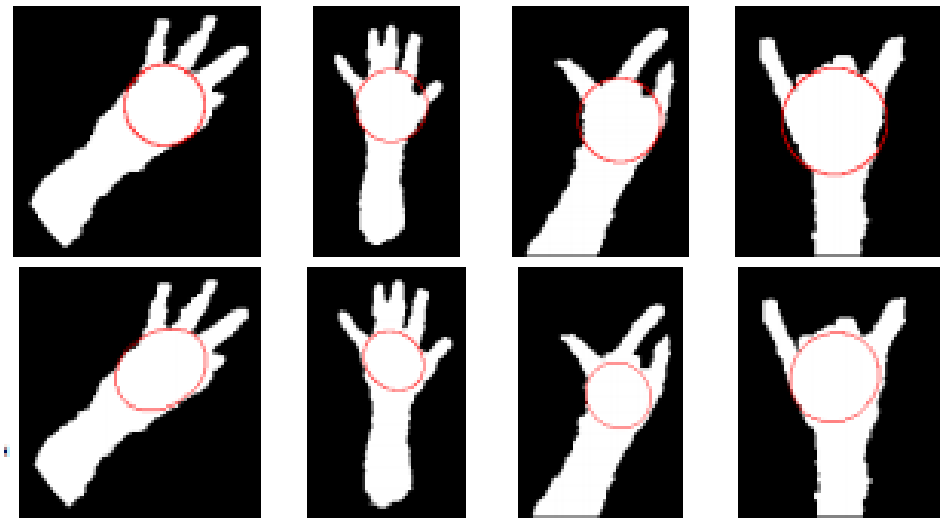
1.6 pav. *Apie rankos kontūrą apibrėžtas daugiakampis, naudojant Convex full funkciją*[15]

Kaip matome paveiksle, nesutapimų tarp kontūrų gali būti gana daug, kadangi rankos kontūras nėra idealus ir apibrėžtas daugiakampis turi daug nelygumų. Tarpo tarp pirštų išskyrimui ir nereikalingų taškų atmetimui reikalingas tolesnis filtravimas.

Tačiau, segmentuoti ranką naudojant vien tik spalvinę kamerą yra sunku. Taip yra dėl to, kad žmonių odos spalva skiriasi, netgi to paties asmens oda gali turėti įvairiausių atspalvių. Be to, ji turi išsiskirti iš aplinkos. Žmogaus odos ar aplinkos spalvos galima nepaisyti naudojant gylio kamerą. Naudojant gylio informaciją galima pasakyti, kaip objektai išsidėstę prieš kamerą. Taigi, jei rodome kokį nors gestą, galime nuspręsti, kur yra ranka ir plaštaka.

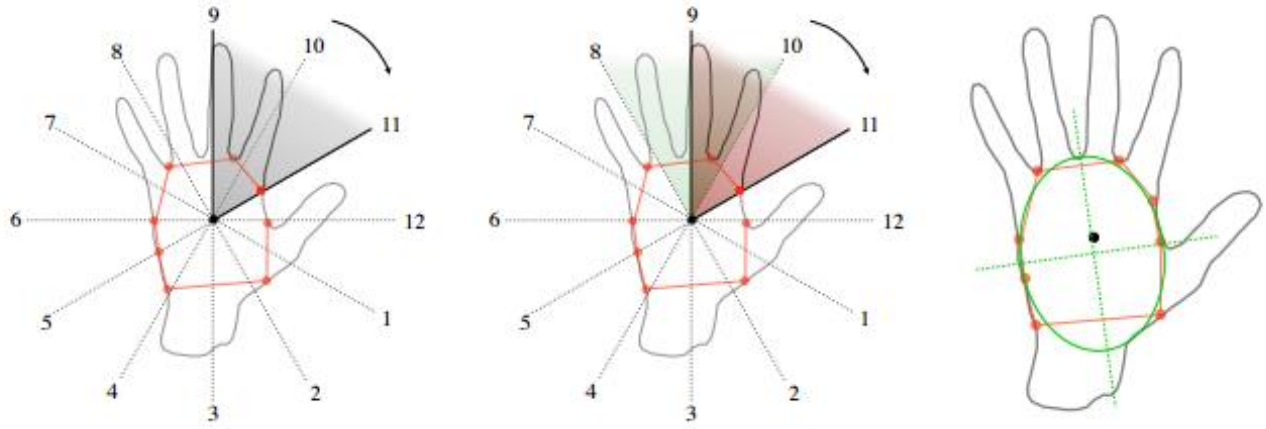
Galime rasti įvairių dilbio ir plaštakos segmentavimo būdų, naudojant gylio kadro informaciją. Beveik visi jie remiasi viso matuojamo gylio diapazono skaidymo į atskirus regionus metodu. Taigi, analizuojamos atitinkamos arba tam tikros sritys.

Vienas metodas [19] iš gylio kadro analizuoja 0.5-0.8m diapazoną. Jame vaizduojami rankomis rodomi ženklai. Į regioną patenka ranka nuo plaštakos iki dilbio vidurio (1.7 pav.). Šis metodas dažniau taikomas plaštakos centro atpažinimui.



1.7 pav. *Segmentuota rankos dalis*[19]

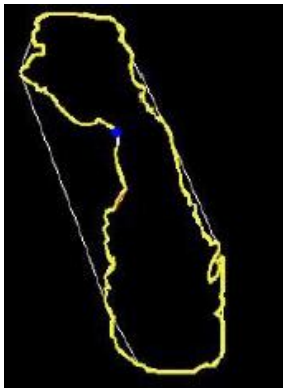
Nustatomas vieno iš septynių ***Hu momentų***[20] taškas, kuris yra suskirstytų regionų centras. Ieškoma atitinkamai diskretizuotų regionų ir kontūro susikirtimo taškų (1.8 pav.).



1.8 pav. *Skirstymas į regionus ir sankirtos taškų radimas*

Straipsnyje žingsnis yra 30° – taip aplink centro tašką išskiriama 12 regionų. Po to ieškoma galų susikirtimų su rankos kontūru. Į šiuos taškus įbrėžiama elipsė, kurios centras yra delno centras. Toks metodas nėra pats geriausias, nes sunku nustatyti pradinį diskretizavimo tašką, be to, kontūras turi būti pakankamai lygus.

Kitas apžvelgtas būdas [21] yra geresnis. Jame ranka segmentuojama taip pat (1.9 pav.), kaip ir prieš tai jau minėtame [19], tačiau plaštakos centro nustatymui naudojamas iteracinis skaičiavimas. Tyrime apžvelgiamas variantas su 25 ir 50 žingsniu. Bandoma įbrėžti didžiausio ploto apskritimą, į kurį gali patekti ne didesnis nei nustatytas taškų kiekis.

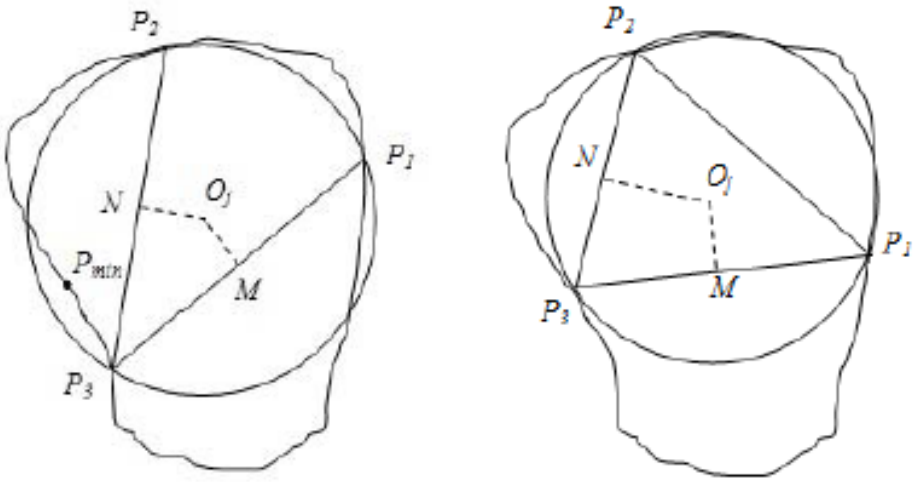


1.9 pav. *Segmentuota ranka*[21]

Tyrime parenkami 3 taškai - **P_1, P_2, P_3** (1.10 pav.) – delno centro nustatymui. Surandame vidurio taškus **N** (kraštinė **P_2P_3**) ir **M** (kraštinė **P_1P_3**). Išvedami statmenys, kurių susikirtimo taškas(X_j,Y_j) O_j. (1, 2 formulės[21]).

$$X_j = \frac{(Y_M - Y_N)(Y_3 - Y_1)(Y_3 - Y_2) + X_M(X_3 - X_1)(Y_3 - Y_2) - X_N(X_3 - X_2)(Y_3 - Y_1)}{(X_3 - X_1)(Y_3 - Y_2) - (X_3 - X_2)(Y_3 - Y_1)} \quad (1)$$

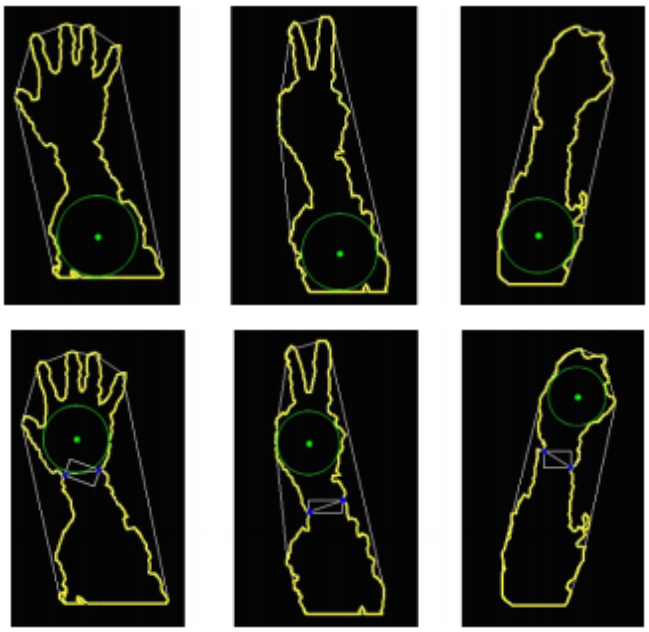
$$Y_j = \frac{(X_N - X_M)(X_3 - X_1)(X_3 - X_2) - Y_M(Y_3 - Y_1)(X_3 - X_2) + Y_N(Y_3 - Y_2)(X_3 - X_1)}{(X_3 - X_1)(Y_3 - Y_2) - (X_3 - X_2)(Y_3 - Y_1)} \quad (2)$$



1.10 pav. *3 taškų kontūre parinkimas* [21]

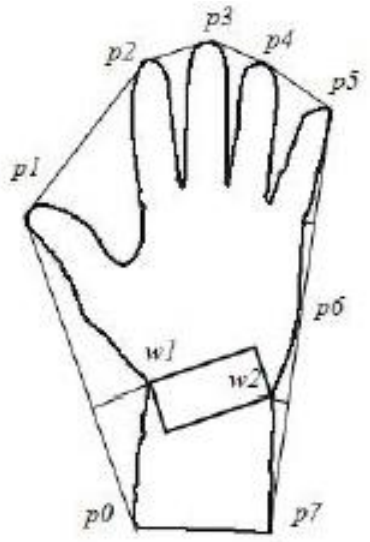
Toliau apskaičiuojamas Euklido atstumas[22] iki kiekvieno kontūro taško. Jei atstumas mažesnis už apskritimo spindulį – taškas yra apskritime (***P_{min}*** žr. 1.10 paveikslas). Jei esantis apskritimo centre

taškų kiekis viršija nustatytą ribą, parenkami kiti taškai – P_1, P_2, P_3. Šiam metodui svarbu rasti riešo taškus, kadangi didžiausias prieš tai minėtu iteraciniu būdu rastas apskritimas nebūtinai gali būti delnas. Šiuo atveju dažniausiai tai yra dilbis. Dėl šios priežasties, žinant riešo taškus, reikia tikrinti, ar rastas didžiausias apskritimas nėra žemiau riešo (1.11 pav.).



1.11 pav. *Viršuje – klaidingai aptiktas delno centras, apačioje – teisingas aptikimas*[21]

Pirštų galai randami apibrėžiant rankos kontūrą daugiakampiu (*convex hull* [18]) ir randant rankos kontūro ir daugiakampio susikirtimo taškus (1.12 pav.). Atmetami už riešo esantys susikirtimo taškai.



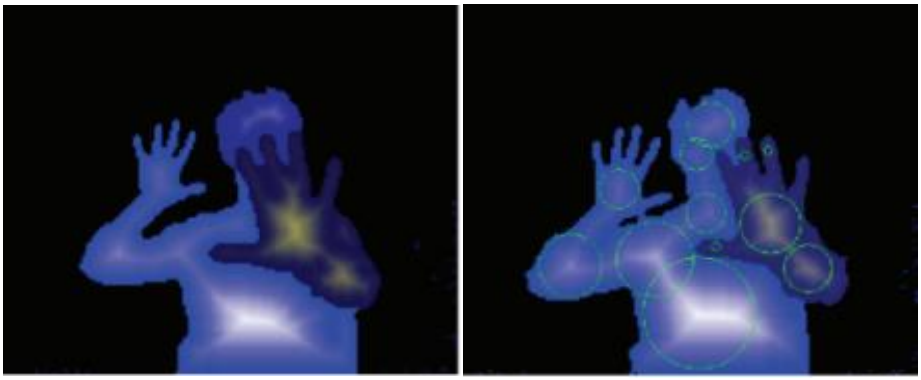
1.12 pav. *p1, p2, p3, p4, p5 – pirštų galai*

Šis metodas daug tiksliau gali nustatyti, kur yra delno centras, bet nustatymo iteracijų skaičius gali kisti dėl kontūro nelygumų ar nustatyto mažo į apskritimą patenkančių taškų skaičiaus.

Dar vienas būdas, pateiktas kaip labai greitas algoritmas plaštakos gestų atpažinimui (***FLIGHT***)[23]. Jame segmentuojama ne tik ranka, bet ir visas kūnas. Šis algoritmas, kaip ir anksčiau analizuoti, naudoja gylio kadro duomenis. Priešingai nei ankstesni metodai, pastarasis gali aptikti ir žmogaus galvos bei kūno taškus. Pirmiausiai visas žmogaus kūno regionas suskirstomas į 4 ruožus po 15cm (1.13 pav.). Kiekvienas regionas apdorojamas atskirai.

1.13 pav. *Segmentacija. 4 regionai po 15cm kairėje ir FAST algoritmo taikymas pirštų galams rasti dešinėje*[23]

Tyrime pirštų galams ir kraštams rasti naudojamas FAST [24] algoritmas (1.13 pav.). Radus atitinkamus taškus, jų filtravimui taikomas svertinis lyginimas. Didžiausių regionų plotų lokaliems maksimumams rasti yra taikoma atstumo transformacijos (***distance transform***) [25] funkcija, kuri visus binarinius taškus paverčia atstumais (skaičiais) iki fono. Taip galima įvertinti, kur yra regiono centras (1.14 pav. kairė).



1.14 pav. *Apdorotas* ***distance transform*** *funkcija kadras* [23]

Šviesiausi regionai paveikslo kairėje pusėje parodo labiausiai nuo juodo fono nutolusius taškus. Paveiksle sujungti visi segmentuoti regionai. Toliau randami labiausia nuo centro nutolusių regionų taškai, iš kurių brėžiami apskritimai (1.14 pav. dešinė). Tolesniame etape atliekamas atitinkamas klasifikavimas ir visų keturių regionų sujungimas. Šiame straipsnyje aprašytas būdas gali padėti išskirti ne tik rankos gestą, bet ir galvos bei krūtinės taškus.

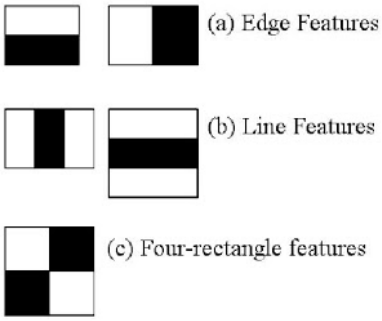
Technologijų bei tyrimų apžvalga parodo, kad nėra sukurta unifikuotos, bet kokiam jutikliui pritaikytos, technologijos rankos gestams atpažinti. Kiekviena iš minėtųjų programinių įrangų turi savų stipriųjų ir silpnųjų pusių, o tyrimų dalyje galima matyti įvairių, tiek dalinių, tiek pilnų problemos sprendimo būdų. Šiame darbe kuriama rankos gestos atpažinimo sistema, kuri yra šiek tiek universalesnė už minėtas tyrimuose. Algoritmas naudos RGB duomenis atpažinti žmogui, sėdinčiam prieš kamerą, ir gylio kadrą gesto analizavimui. Šis sprendimas kombinuos dviejų jutiklių duomenis rankos aptikimui ir gesto analizavimui.

2. Teorinė dalis

2.1 Veido aptikimas

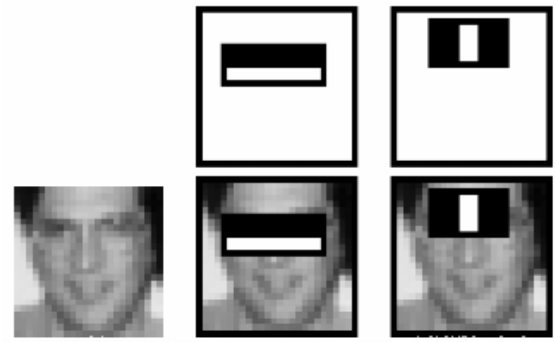
2.1.1 Haar kaskados

Haar kaskadų klasifikatorius – tai efektyvus metodas objektams atpažinti, aprašytas P.Viola ir M.Viola [17]. Šiam klasifikatoriui naudojama norimo objekto nuotraukų duomenų bazė bei daug nuotraukų, kuriose jo nėra – negatyvų. Klasifikatoriaus apmokymui reikia išskirti norimo objekto požymius. Šiam tikslui nuotrauka padengiama juodai-baltais stačiakampiais ir skaičiuojamas taškų sumos po juodu ir baltu plotu skirtumas (2.1.1.1 pav.).



2.1.1.1 pav. *Juodai-balti stačiakampiais, kuriais padengiama nuotrauka* [26]

Minėtų požymių gali būti daug. Pavyzdžiui, 24x24 taškų dydžio nuotraukai jų gali būti netgi daugiau nei 160000. Tai reikalauja daug skaičiavimo resursų, todėl ***AdaBoost*** (*angl. adaptive boosting*) [27] atliekama optimizacija – atmetami neinformatyvūs atskyrimo požymiai. Kitaip tariant, visiems požymiams suteikiama svertiniai koeficientai, o turintys mažą koeficientą – atmetami.



2.1.1.2 pav. *Stačiakampiai, su kuriais galima išskirti stipriausius požymius veido atpažinime* [26]

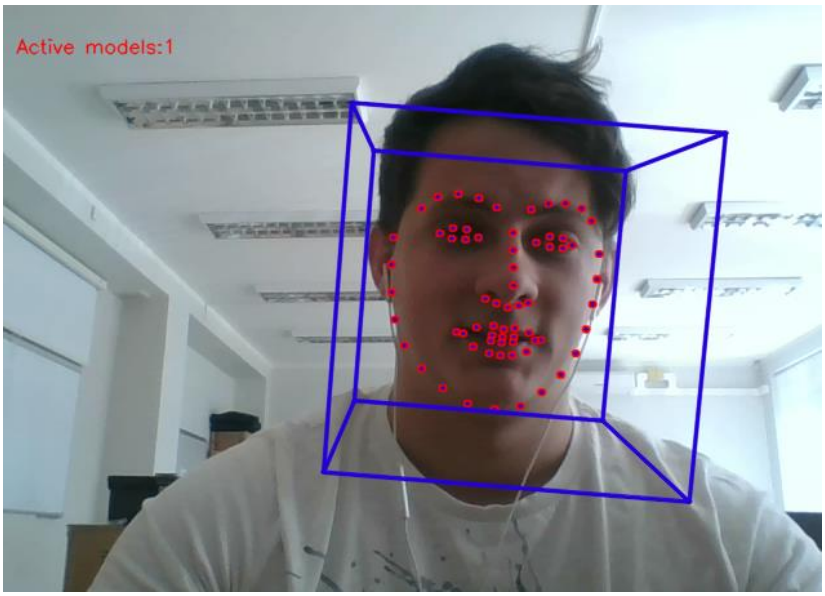
2.1.1.2 paveiksle parodyti veido atpažinime turintys didžiausius koeficientus stačiakampiai. Minėtu būdu mažinant požymių skaičių galima žymiai pagreitinti algoritmą bei išlaikyti aukštą atpažinimo lygį. 160000 sumažinus iki 6000 (netgi sumažinus iki 200), galima išlaikyti iki 95% aptikimo tikimybę [17].

Šiame darbe naudojamas ***OpenCV*** 3.2.0 bibliotekoje realizuotas Haar klasifikatorius veido atpažinimui. Algoritmas pagreitintas naudojant vaizdo plokštę (***CUDA*** technologija).

2.1.2 OpenFace

OpenFace [28] – C++ kalba parašyta programinė įranga skirta veido atpažinimui, veido bruožų aptikimui ir žvilgsnio sutelktumo nustatymui [29], [30]. Tai atviro kodo biblioteka, kuri gali būti naudojama demonstraciniuose bei komerciniuose (dėl sąlygų reikia susisiekti su autoriumi) projektuose.

Su ***OpenFace*** pateiktais programų pavyzdžiais galima analizuoti vieną nuotrauką, vaizdo klipą ar tiesioginę transliaciją iš kameros.

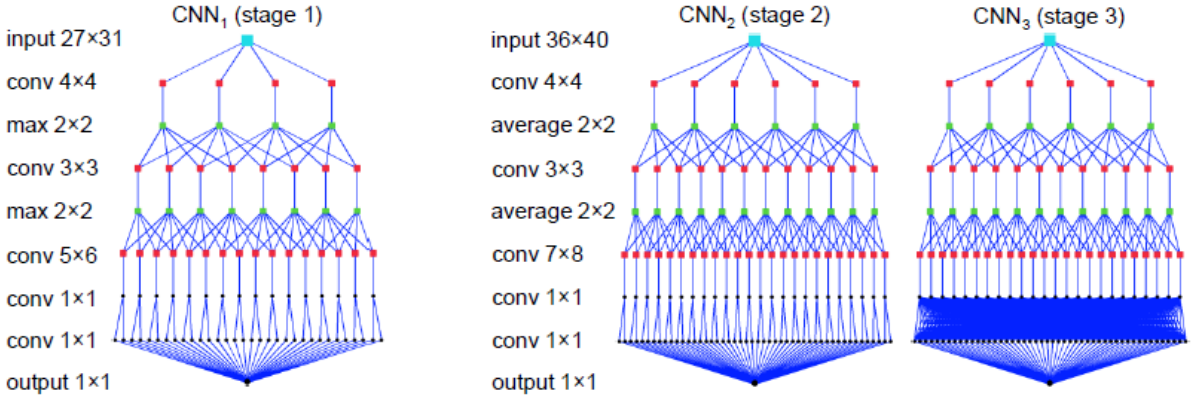


2.1.2.1 pav. *OpenFace veido bruožų aptikimas*

Ši programinė įranga naudoja kitų vaizdo atpažinimo bibliotekų funkcijas. Veido sekimui - ***Haar*** kaskadų klasifikatorius ir ***OpenCV*** [31] bei ***HOG*** [32] detektorius iš ***DLib*** [13]. ***OpenFace*** bibliotekoje pritaikytas įdomus šios funkcijos sprendimas. Jei veido atpažinimo operacija vykdoma tik kompiuterio procesoriuje, tai reikalauja daug skaičiavimo resursų. Programinės įrangos algoritmas veido ieško tik kas aštuntą (arba programuotojo užduotą) kadrą, o tarp jų veidas sekamas pagal kitą – daug kompiuterinio pajėgumo nereikalaujančią funkciją. Akių kontūro, veido formos, nosies ir antakių taškai randami naudojant ***DLib*** bibliotekos *face_landmark_detection* funkciją.

2.1.3 Compact Convulsional Neural Network Cascades

Bene moderniausias metodas veidui atpažinti, lyginant su prieš tai aprašytais, yra CNN kaskados (*Compact Convulsional Neural Network Cascades*) [33]. Šis neuroninis tinklas turi 3 lygius (2.1.3.1 pav.).



2.1.3.1 pav. *CNN struktūra*

Apmokymui panaudota ***YouTube Face duomenų bazė*** [34], kurioje yra 1595 žmonių sekami veidai iš 3425 vaizdo įrašų. Apmokymas sudarytas iš daugiau nei 1 milijono nuotraukų (433000 pozityvų ir 585000 negatyvų).

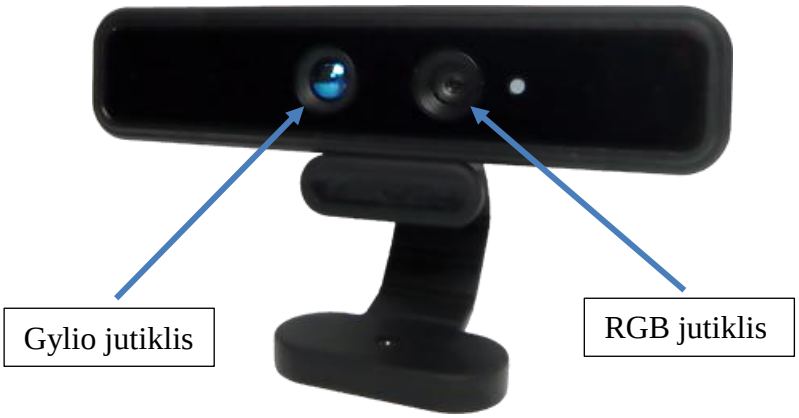
Ši biblioteka veikia tik su ***NVidia CUDA***[12]. ***CNN kaskadų*** metodas yra labai greitas net ir naudojant jį mobiliose platformose. Galima realiu laiku (25 kadrais per sekundę) išanalizuoti 4K dydžio vaizdo įrašą [33]. Tyrime naudoto kompiuterio parametrai pareikti 2.1.3.1 lentelėje.

2.1.3.1 lentelė. *Tyrime naudota aparatūra* [33].

Pavadinimas	Aparatūra/parametras
Procesorius	Intel Core i7-3610QM CPU (2.3 GHz, Turbo Boost disabled)
Vaizdo procesorius 1	Intel HD Graphics 4000 GPU1 (GT2, 16 core, 1,100 MHz)
Vaizdo procesorius 2	NVidia GeForce GT 640M GPU2 (GK107, 384 cores, 709 MHz)

2.2 Gylio ir RGB kamerų atitinkamų taškų sutapdinimas

Norint panaudoti abiejų, gylio ir RGB, kamerų duomenis, reikia rasti fizinę sąsają tarp kiekvieno taško. Kadangi jutikliai stovi skirtingose padėtyse (2.2.1 pav.), jų rezoliucijos ir optiniai parametrai skiriasi, todėl individualiai reikia apskaičiuoti kiekvieno pikselio priklausomybę iš vienos kameros kitai.



2.2.1 pav. *DepthSenseDS325 gylio ir RGB jutiklis*[4]

Pavyzdžiui, gylio taškams atitinkamos spalvos (2.2.2 pav.) randamos pagal šiuos žingsnius[35]:

- Taškų debesies (*angl. Point Cloud*) skaičiavimas (X, Y, Z koordinatės) iš gylio kameros duomenų (2, 3, 4 formulė).

$$X_D = (x_D - c_{xD}) * \frac{d}{f_{xD}} \quad (2)$$

$$Y_D = (y_D - c_{yD}) * \frac{d}{f_{yD}} \quad (3)$$

$$Z_D = d \quad (4)$$

$\boldsymbol{X_D, Y_D, Z_D}$ *– taškų debesies koordinatės,* $\boldsymbol{d}$ *– gylio kadro taško įvertis,* $\boldsymbol{x_D}$ *- gylio kadro taško x koordinatė,* $\boldsymbol{y_D}$ *- gylio kadro y koordinatė,* $\boldsymbol{c_{xD}, c_{yD}}$ *– gylio kameros optinis centras,* $\boldsymbol{f_{xD}, f_{yD}}$ *– gylio kameros optinis ilgis*

DepthSenseDS325 gylio jutiklio parametrai: $\boldsymbol{f_{xD}}$ = 224.502, $\boldsymbol{f_{yD}}$ = 230.494, $\boldsymbol{c_{xD}}$= 160, $\boldsymbol{c_{yD}}$= 120.

- Taškų debesies pasukimas ir pastūmimas pagal RGB ir gylio jutiklių padėtį (5 formulė).

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = R \begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix} + t \quad (5)$$

[4] https://www.softkinetic.com/Store/ProductID/29

$R = \begin{bmatrix} \boldsymbol{r_{11}} & \boldsymbol{r_{12}} & \boldsymbol{r_{13}} \\ \boldsymbol{r_{21}} & \boldsymbol{r_{22}} & \boldsymbol{r_{23}} \\ \boldsymbol{r_{31}} & \boldsymbol{r_{32}} & \boldsymbol{r_{33}} \end{bmatrix}$ *– posūkio matrica,* $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t_1} \\ \boldsymbol{t_2} \\ \boldsymbol{t_3} \end{bmatrix}$ *- postūmio matrica,* $\begin{bmatrix} \boldsymbol{X_D} \\ \boldsymbol{Y_D} \\ \boldsymbol{Z_D} \end{bmatrix}$ *– taškų debesies koordinatės,* $\begin{bmatrix} \boldsymbol{X_C} \\ \boldsymbol{Y_C} \\ \boldsymbol{Z_C} \end{bmatrix}$ *- apskaičiuotos koordinatės*

DepthSenseDS325 kameros posūkio R ir postūmio (pateikta metrais) T matricos:

$$R = \begin{bmatrix} 0.99999738 & 0.0012669859 & -0.0019156976 \\ 0.0012539299 & -0.99997610 & -0.0068011540 \\ 0.0019242687 & -0.0067987340 & 0.99997503 \end{bmatrix}, T = \begin{bmatrix} 0.024492104 \\ -0.00050799217 \\ -0.00086258771 \end{bmatrix}$$

- Radialinių ir tangentinių iškraipymų kompensavimas (6, 7 formulė).

$$x'' = x'(1 + k_{1C}r^2 + k_{2C}r^4 + k_{3C}r^6) + 2p_{1C}x'y' + p_{2C}(r^2 + 2x'^2) \quad (6)$$

$$y'' = y'(1 + k_{1C}r^2 + k_{2C}r^4 + k_{3C}r^6) + 2p_{2C}x'y' + p_{1C}(r^2 + 2y'^2), \quad (7)$$

čia

$$x' = \frac{X_C}{Z_C},$$

$$y' = \frac{Y_C}{Z_C},$$

$$r^2 = x'^2 + y'^2$$

$\boldsymbol{k_{1C}, k_{2C}, k_{3C}}$ *- radialiniai RGB kameros iškraipymai,* $\boldsymbol{p_{1C}, p_{2C}}$ *– tangentiniai RGB kameros iškraipymai;*

$\boldsymbol{k_{1C}}$ = 0.0225752, $\boldsymbol{k_{2C}}$ = -0.162668, $\boldsymbol{k_{3C}}$ = 0.186138, $\boldsymbol{p_{1C}}$ = 0, $\boldsymbol{p_{2C}}$ = 0

Kadangi $\boldsymbol{p_{1C}}$ ir $\boldsymbol{p_{2C}}$ lygūs 0, todėl dalis iškraipymo kompensavimo formulių (6, 7) suprastinamos (8, 9):

$$x'' = x'(1 + k_{1C}r^2 + k_{2C}r^4 + k_{3C}r^6) \quad (8)$$

$$y'' = y'(1 + k_{1C}r^2 + k_{2C}r^4 + k_{3C}r^6) \quad (9)$$

- Atitinkamų RGB kameros taškų radimas (10, 11 formulės).

$$x_C = f_{xC} * x'' + c_{xC} \quad (10)$$

$$y_C = f_{yC} * y'' + c_{yC} \quad (11)$$

$\boldsymbol{x_C, y_C}$ *– x ir y taško koordinatės RGB paveiksle,* $\boldsymbol{f_{xC}, f_{yC}}$ *– optinis x ir y ilgis,* $\boldsymbol{c_{xC}, c_{yC}}$ *– optinis centras*

DepthSenseDS325 RGB jutiklio parametrai: f_{xC} = 583.07898, f_{yC} = 596.203, c_{xC} = 320, c_{yC} = 240.

a) **b)** **c)**

2.2.2 pav. *RGB (a), gylio (b) kadrai ir ant gylio užtrauktų RGB spalvų kadras (c)*

Šiame darbe pristatomame gestų atpažinimo metode žmogaus veidas yra atskaitos taškas, nuo kurio atliekama erdvės diapazono analizė tarp kameros ir veido.

2.3 Medianos filtras

Medianos filtras – tai skaitmeninis filtravimo būdas, šalinantis nuotraukų ar signalų triukšmą [36], [37]. Šis metodas apskaičiuoja kiekvieno pikselio ir jo aplinkinių taškų vidurkį. Taikant šį filtrą galima užsiduoti kiek kaimyninių pikselių bus naudojama skaičiuojant vidutinę reikšmę. Pavyzdžiui, jei 2.3.1 paveiksle esančiam centriniam taškui, kurio reikšmė 150, pritaikysim medianos filtrą, kurio dydis 3x3, tai gausime naują reikšmę:

$$\frac{124 + 126 + 127 + 120 + 150 + 125 + 115 + 119 + 123}{9} = 124$$

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

2.3.1 pav. *5x5 paveikslo taškų vertės*[5]

Medianos filtras labai efektyvus, jei reikia šalinti „druskos ir pipirų“ triukšmą[38]. Tai juodi ar balti išsibarstę nuotraukoje taškai. Kaip matyti 2.3.2a paveiksle, aplink asmens kontūrą yra nemažai

[5] http://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm

pikselių (triukšmo), atsirandančių dėl neidealaus jutiklio veikimo. 2.3.2b paveiksle jie pašalinami pritaikius medianos filtrą (3x3).

a) **b)**

2.3.2 pav. *a – originalus gylio kadras, b – panaudojus medianos (3x3) filtrą*

Šiame darbe naudojama medianos filtras iš ***OpenCV*** 3.2.0 bibliotekos. Jis taikomas gylio kadro triukšmui pašalinti.

2.4 Atstumo transformacija

Atstumo transformacija (angl. *distance transformation*) – tai operacija, skirta skaitmeninėse nuotraukose nustatyti kiekvieno objekto taško atstumą iki jo kontūro [25][39]. Ši funkcija taikytina tik binarinėms reikšmėms kadre. Visi pikseliai turi būti lygūs 0 arba 1. Tai gali padėti nustatyti išskirto objekto didžiausias, daugiausiai pikselių nuo krašto nutolusias vietas, kaip parodyta 2.4.1 paveiksle.

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

⇒

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

a) b)

2.4.1 pav. *a – binarinis paveikslas, b – paveikslo atstumo transformacija*[6]

Atstumą galima skaičiuoti 3 skirtingais būdais[40](2.4.2 pav.):

- Euklido atstumu [22] (12 formulė)

$$L_2 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (12)$$

$\boldsymbol{L_2}$ *– Euklido atstumas;* $\boldsymbol{x_1, y_1}$ *- pirmo taško koordinatės;* $\boldsymbol{x_2, y_2}$ *- pirmo taško koordinatės*

[6] http://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm

- Miesto blokų (13 formulė)

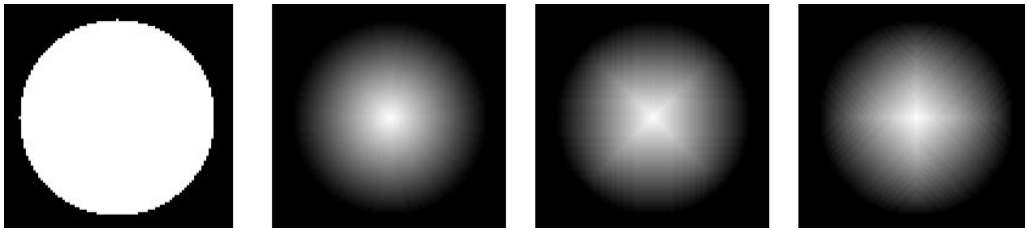
$$L_1 = |x_1 - x_2| + |y_1 - y_2| \quad (13)$$

L_1 *– atstumas, apskaičiuotas miesto blokų metodu, ;* ***x_1, y_1*** *- pirmo taško koordinatės;* ***x_2, y_2*** *- pirmo taško koordinatės*

- Šachmatų lentos (14 formulė)

$$L_\infty = \max(|x_1 - x_2|, |y_1 - y_2|) \quad (14)$$

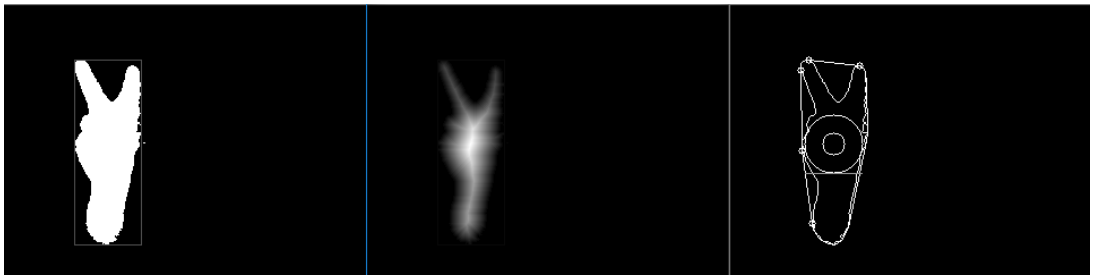
L_∞ *– atstumas, apskaičiuotas miesto blokų metodu;* ***x_1, y_1*** *- pirmo taško koordinatės;* ***x_2, y_2*** *- pirmo taško koordinatės*



a) **b)** **c)** **d)**

2.4.2 pav. *a – originalus binarinis paveikslas, b – atstumo transformacija Euklido atstumo metodu, c – atstumo transformacija miestų blokų metodu, d – atstumo transformacija šachmatų lentos metodu* [40]

Šiame darbe atstumo transformacija naudojama delno centrui nustatyti. Kadangi delnas arba kumštis analizuojamu atveju yra plačiausia objekto (plaštakos) dalis, todėl naudodami šį metodą galime identifikuoti rankos centrą, ieškodami didžiausios reikšmės tarp atstumo transformacijos taškų.



a) **b)** **c)**

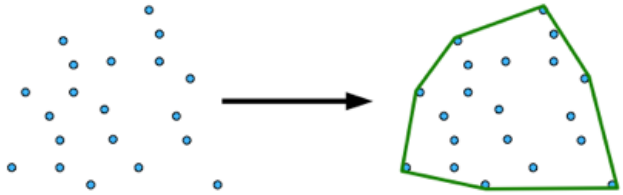
2.4.3 pav. *a - binarinis rankos paveikslas, b – tas pats paveikslas po atstumo transformacijos, c – pagal atstumo transformacijos vertę įbrėžtas apskritimas*

Be to, šis įvertis bus ir į delną ar kumštį įbrėžto apskritimo spindulys. 2.4.3 pav. ***c*** - paveikslo pusėje pateiktas lokalizuotas centras – mažas apskritimas bei rankos plotis pagal atstumo transformacijos įvertį – didžiausias apskritimas.

Kuriamame algoritme naudojama optimizuota atstumo transformacijos funkcija iš ***OpenCV*** 3.2.0 bibliotekos. Šiame darbe metodas skaičiuoja pagal Euklido atstumą.

2.5 *Convex hull* ir *Convexity defects* funkcijos

Convex hull funkcija – tai yra daugiakampis, kuriuo galima apibrėžti taškų aibę [18](2.5.1 pav.). Moksliniuose straipsniuose[41], [42], [43] galima rasti ne vieną šios matematinės problemos sprendimą.

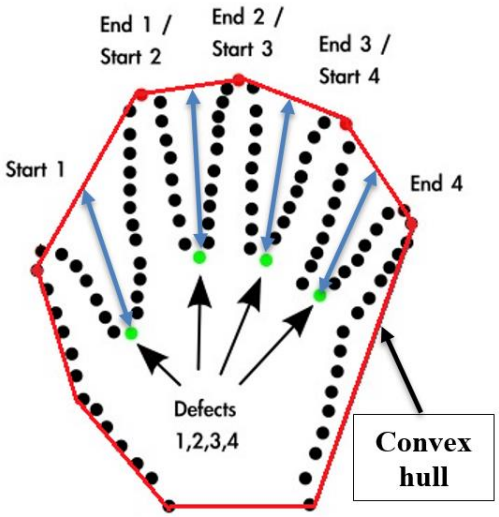


2.5.1 pav. ***Convex hull*** *daugiakampiu apibrėžta taškų aibė*[7]

Convex hull algoritmas naudojamas įvairiose srityse[44]:

- kompiuterinėse vizualizacijose (pvz.: žaidimuose);
- kelio radimo uždaviniuose (pvz.: robotų kelio radime);
- geografinėse informacijos sistemose (*angl. geographical information system (GIS));*
- vaizdo atpažinime (pvz.: objekto išskyrime);
- geometrijoje (pvz. perimetro skaičiavime).

Šiame darbe aprašomame algoritme, ***Convex hull*** funkcija naudojama apibrėžti išskirtą rankos kontūrą. Kadangi tai yra taškų masyvas, metodas gali rasti mažiausią daugiakampį apie taškus (2.5.2 pav.).



2.5.2 pav. *Rankos kontūras, apibrėžtas daugiakampiu*

Tolesniu žingsniu galime analizuoti skirtumus tarp rankos kontūro ir ***Convex hull*** daugiakampio. Tokius skirtumus galima vadinti „defektais“, o juos išskirti galime naudodami ***Convexity defects*** funkciją. 2.5.2 paveiksle parodyta rankos kontūro ir apibrėžto daugiakampio nesutapimai. Keturi įvardinti įdubų taškai (2.5.2 pav. ***Defects***) yra labiausiai nutolę nuo ***Convex hull***. Jų pradžia pažymėta

[7] http://axon.cs.byu.edu/Dan/312/projects/ConvexHull.php

Start1, Start2, Start3, Start4 ir pabaiga *End1, End2, End3, End4* taškais. ***Convexity defects*** funkcija apibrėžia įdubą („defektą“) keturiais parametrais:

- įdubos pradžios tašku;
- įdubos pabaigos tašku;
- labiausia nuo apibrėžto daugiakampio nutolusiu įdubos tašku;
- atstumu nuo apibrėžto daugiakampio iki labiausiai nutolusio įdubos taško (2.5.2 pav. mėlyna rodyklė).

Brėžiant daugiakampį apie kontūrą ir taikant ***Convexity defects*** funkciją, randama daug įdubų (2.5.3 pav.). Deja, ne visi taškai yra reikšmingi. Kadangi rankos kontūras dėl matavimo netikslumų yra nelygus, nereikalingus taškus reikia išfiltruoti pagal įdubos gylį, tai yra, pagal atstumą iki tolimiausio taško nuo apibrėžto daugiakampio.

a) **b)**

2.5.3 pav. *a – aplink kontūrą apibrėžtas daugiakampis ir pažymėti įdubos pradžios, pabaigos ir toliausiai nuo daugiakampio nutolę taškai, b – binarinis rankos gesto kadras, iš kurio gaunamas kontūras*

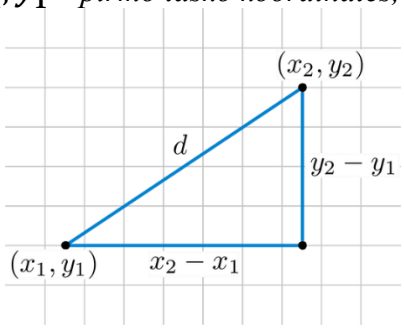
Darbe panaudotos kontūro radimo (***findCountours***), ***Convex hull*** ir ***Convexity defects*** funkcijos yra iš ***OpenCV*** 3.2.0 bibliotekos.

2.6 Euklido atstumas

Euklido atstumas – tai atstumas tarp taškų[22]. Dvimatėje erdvėje tai yra atstumas tarp dviejų taškų (2.6.1 pav.)(15 formulė).

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (15)$$

d *– Euklido atstumas;* $\boldsymbol{x_1, y_1}$ *– pirmo taško koordinatės,* $\boldsymbol{x_2, y_2}$*- antro taško koordinatės*

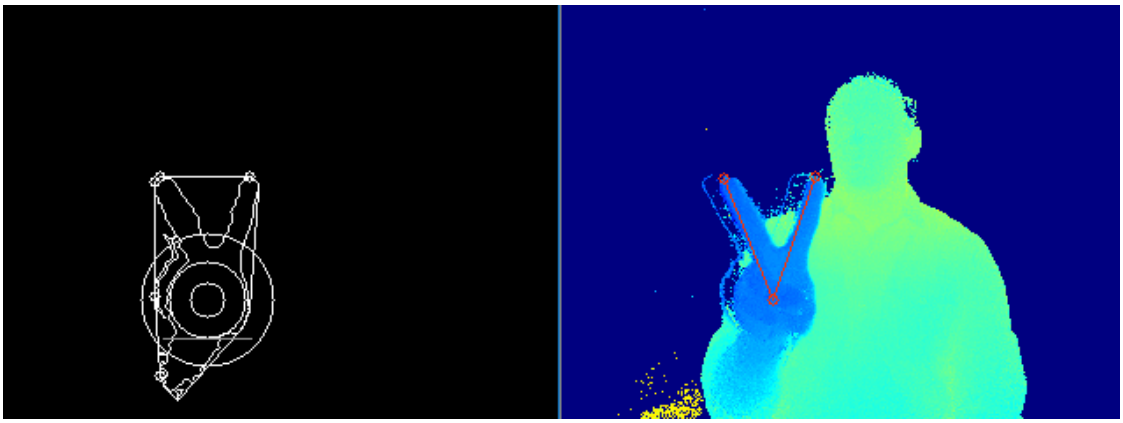


2.6.1 pav. *Euklido atstumo tarp dviejų taškų skaičiavimas*

Šiame darbe pristatomame algoritme Euklido atstumo skaičiavimas naudojamas keliose vietose. Pirmoji, jau minėta, yra atstumo transformacija. Jai apskaičiuoti naudojamas optimizuotas skaičiavimas, kadangi iteracijų skaičius yra nemažas. Antroji vieta yra pirštų galų koordinačių tikrinimas. Ši koordinatė turi nepatekti į 2.6.2 paveiksle pavaizduoto didesniojo apskritimo plotą, tai yra, turi tenkinti sąlygą (16 formulė):

$$R < \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (16)$$

R *– didesniojo apskritimo spindulys;* $\boldsymbol{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}$ *- Euklido atstumas tarp plaštakos centro* (x_1, y_1) *ir kiekvieno piršto galo* (x_2, y_2) *taškų*



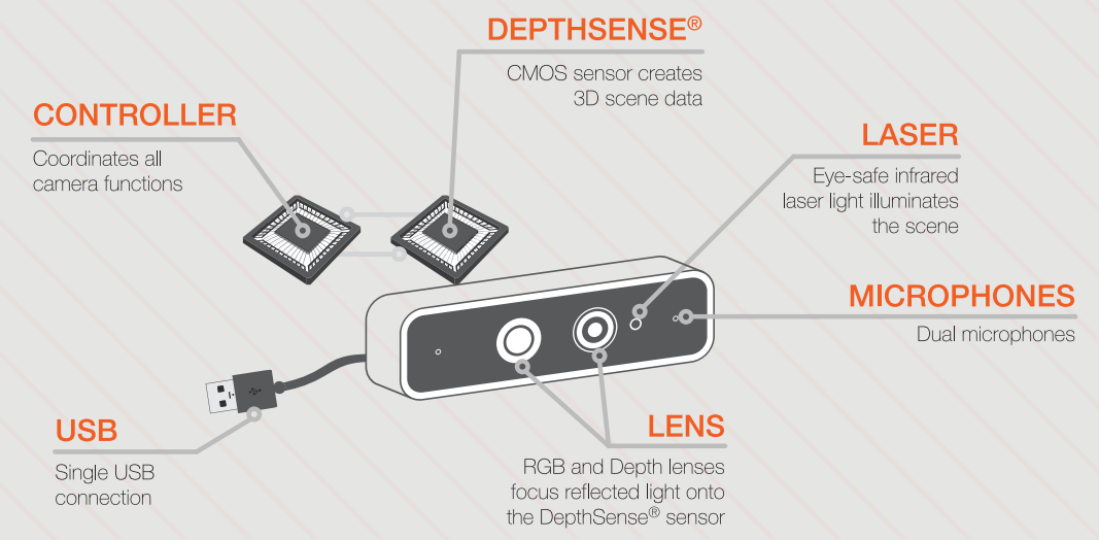
a) **b)**

3.6.2 pav. *a – rankos kontūras su įdubų pradžios ir pabaigos taškais ir apskritimais (iš delno centro), pagal kuriuos vyksta tikrinimas, b – gylio kadras su pažymėtais pirštų galų ir plaštakos centro taškais*

3. Aparatūra ir įrankiai

3.1 Kamera

Darbe naudojamas ***SoftKinetic*** kompanijos ***DepthSenseDS325*** (žr. 2.2.1 pav.) [7] jutiklis. Jį sudaro RGB (spalvinė) ir gylio kameros, mikrofonas bei akcelerometras. Dviejų paskutiniųjų jutiklių duomenys nenaudojami. Kameros sandara parodyta 3.1.1 paveiksle. Techninė specifikacija pateikta 3.1.1 lentelėje.



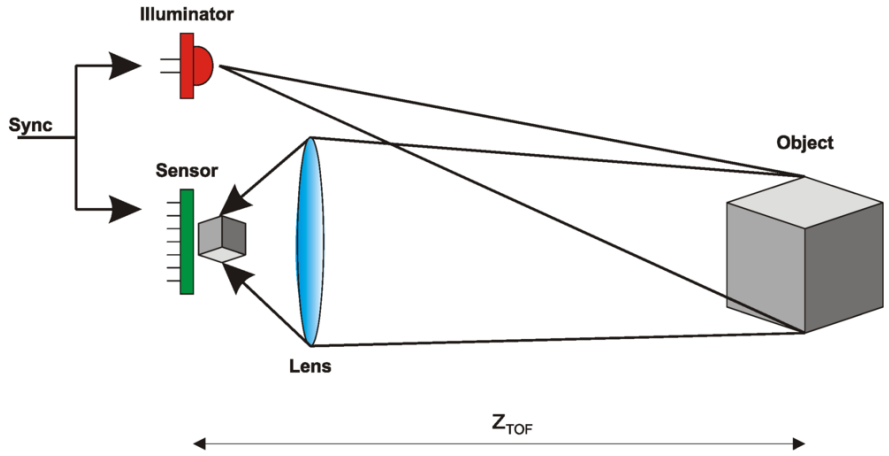
3.1 pav. *DepthSenseDS325 sudėtis* [45]

3.1.1 lentelė. *DepthSenseDS325 techninė specifikacija* [45]

Parametro pavadinimas	Aprašymas
Technologija	CAPD Time-of-flight [46]
Gylio kameros rezoliucija	320x240
Gylio kameros matymo kampas	74°x58°
Gylio kameros sparta	25-60 kadrų per sekundę
Darbinis atstumas	0.15-1.0m
Apšvietimo tipas	Sklaidytas lazeris (*angl. diffused laser*)
Geriausias aplinkos apšvietimas	Patalpų apšvietimas
RGB kameros rezoliucija	1280x720
RGB kameros sparta	30 kadrų per sekundę
RGB kameros matymo kampas	63.2°x49.3°
Mikrofonų kiekis	2
Akcelerometro tipas	3 ašių
Sąsaja	USB 2.0
Darbinė temperatūra	10°C - 40°C
Galia	<=2.5W
Dydis	10.5cm x 3.0cm x 2.3 cm

DepthSenseDS325 turi **TOF** (*time-of-flight*) gylio kamerą. Šio tipo jutiklių atstumo matavimas paremtas šviesos signalo išspinduliavimu ir atspindžio gavimu atgal. Žinant šviesos greitį ir laiką, per kurį šviesos impulsas grįžo į jutiklį, galima apskaičiuoti atstumą nuo kameros iki kiekvieno objekto

taško. Veikimo principas pavaizduotas 3.1.2 paveiksle. Nustatyto bangos ilgio šviesa atsispindėjusi nuo objekto grįžta į jutiklį. Skaičiuojamas laikas tarp išspinduliavimo ir šviesos atspindžio gavimo.



3.1.2 pav. *Time-of-flight jutiklių atstumo matavimo principas* [47]

Atstumas iki objekto nustatomas pagal formulę (17):

$$Z_{TOF} = \frac{c * t_D}{2}, \qquad (17)$$

Z_{TOF} *– atstumas tarp kameros ir objekto,* ***c*** *– šviesos greitis (3*10^8 m/s),* ***t_D*** *– laikas nuo šviesos išspinduliavimo iki atspindžio gavimo*

3.2 Kompiuteris

Šiame darbe aprašyto algoritmo tyrime naudotas nešiojamas Acer E5-575G-55JB kompiuteris, kurio parametrai pateikti 3.2.1 lentelėje.

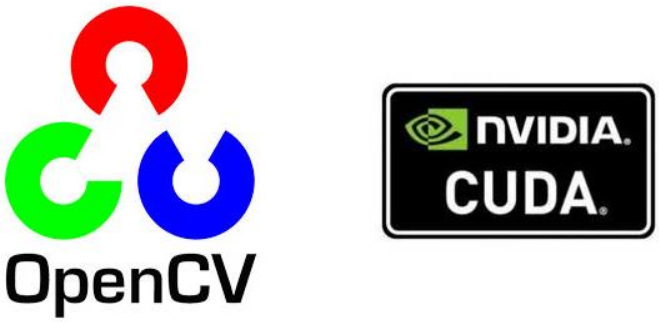
3.2.1 lentelė. *Kompiuterio specifikacija*

Dalis	**Aprašymas**
Procesorius	Intel® Core™ i5-7200U 2.5GHz (Turbo Boost 3.1GHz), 3MB cache, 2 cores, 4 threads
Operatyvioji atmintis	8GB DDR4
Kietasis diskas	256GB SSD
Vaizdo plokštė	NVidia GeForce ® 940MX 2GB GDDR5

Vienas pagrindinių šio kompiuterio privalumų yra vaizdo plokštė, leidžianti naudoti ***CUDA*** lygiagretaus skaičiavimo technologiją. Tai žymiai pagreitina kai kurių algoritmų veikimą. Daugiau apie šią technologijos naudojimą darbe bus aprašyta tolesniame skyriuje.

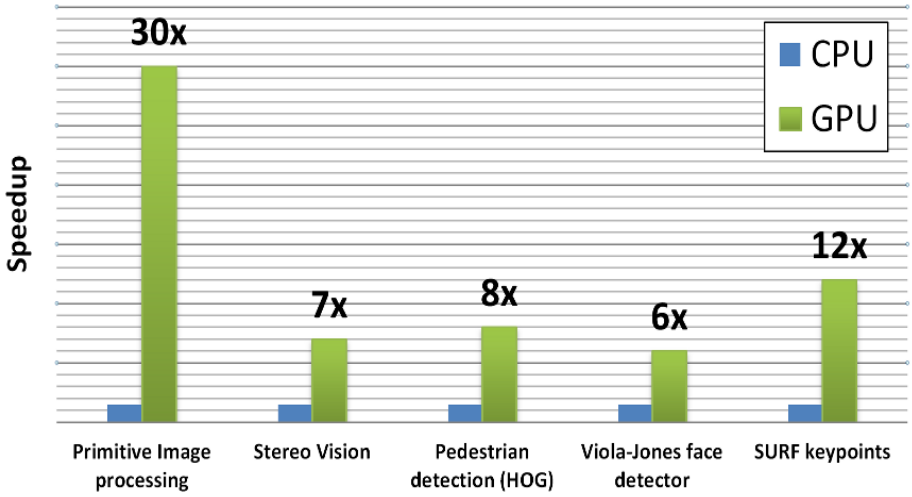
3.3 Programavimo aplinka ir programinė įranga

Darbas atliktas Windows 10 64-bitų operacinėje sistemoje. Programavimui pasirinkta Visual Studio 2013 aplinka. Senesnės versijos naudojimą lėmė ***DepthSense SDK*** kameros programinė įrangos versija, kuri yra pritaikyta minėtajai ar senesnei programavimo aplinkai. Kodas parašytas C++ kalba. ***OpenCV*** 3.2.0 [31] yra pagrindinė naudota darbe biblioteka. Ji sukompiliuota su ***CUDA SDK 8.0*** [12].



3.3.1 pav. ***OpenCV*** *ir* ***NVidia CUDA*** *logotipai*[48]

Tokia ***OpenCV*** konfigūracija įgalina ***CUDA*** paralelaus skaičiavimo resurso vaizdo plokštėje panaudojimą kai kuriems vaizdo analizavimo ir dirbtinio intelekto algoritmams [48] (3.3.2 pav.). Taip pat, algoritmų vykdymas vaizdo plokštės procesoriuje leidžia paskirstyti apkrovą, tai yra, sumažinti pagrindiniam kompiuterio procesoriui tenkančių operacijų skaičių (CPU).



3.3.2 pav. ***OpenCV*** bibliotekos algoritmų palyginimas juos vykdant procesoriuje (CPU) ir ***NVidia GeForce*** vaizdo plokštėje (GPU) su ***CUDA*** technologija [48]

Šiame darbe tokios operacijos, kaip paveikslo mažinimas, spalvų gamų keitimas, veido paieška naudojant HAAR kaskadas, vykdomos vaizdo plokštėje.

Kita naudota programinė įranga, skirta bibliotekų ir projektų konfigūravimui, yra ***CMake*** [49]. Tai pagalbinis įrankis, pagal nustatytus parametrus sugeneruojantis projektą norimai programavimo aplinkai. ***CMake*** reikalingas atitinkamas C/C++ kompiliatorius norimai darbinei aplinkai. Darbe įrankis

naudojamas Visual Studio 2013 projektams kurti. ***CMake*** turi savo kalbą (*angl. script*) konfigūracijos aprašymui (3.3.3 pav.).

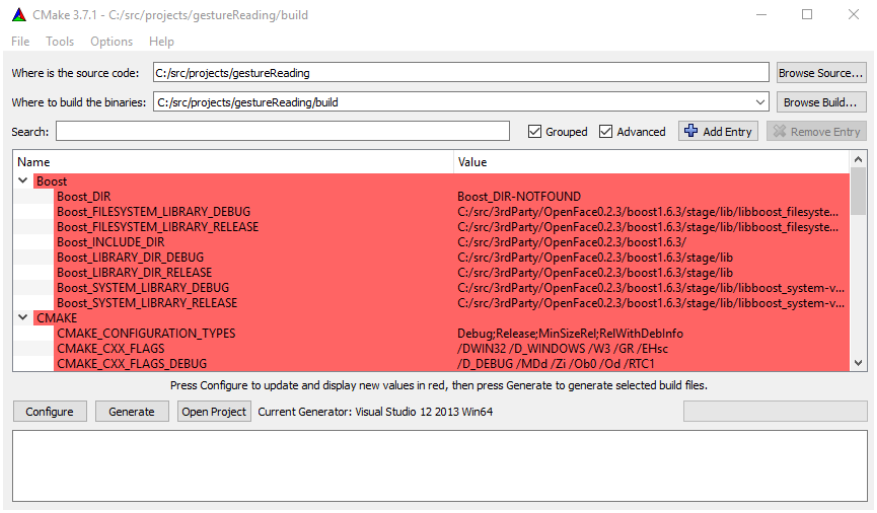
```
cmake_minimum_required (VERSION 2.8.11)
project (gestureReading)

if(CMAKE_COMPILER_IS_GNUCXX)
    add_definitions(-std=gnu++0x)
endif()

file(TO_CMAKE_PATH $ENV{Depthlib}              Depthlib_PATH)
file(TO_CMAKE_PATH $ENV{opencv_320_cuda}       opencv_320_cuda_PATH)
file(TO_CMAKE_PATH $ENV{faceCNN}               faceCNN_PATH)
file(TO_CMAKE_PATH $ENV{OpenFace}              OpenFace_PATH)
file(TO_CMAKE_PATH $ENV{VTK}                   VTK_PATH)

file(TO_CMAKE_PATH $ENV{OpenCVViz}             OpenCVViz_PATH)

find_package( Boost 1.5.9 REQUIRED COMPONENTS filesystem system)
```

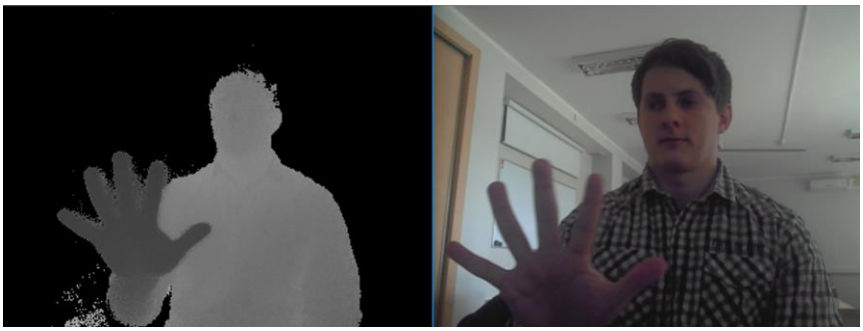


3.3.3 pav. CMake *konfigūracijos aprašymas viršuje ir programos grafinė sąsaja apačioje*

4. Algoritmo aprašymas

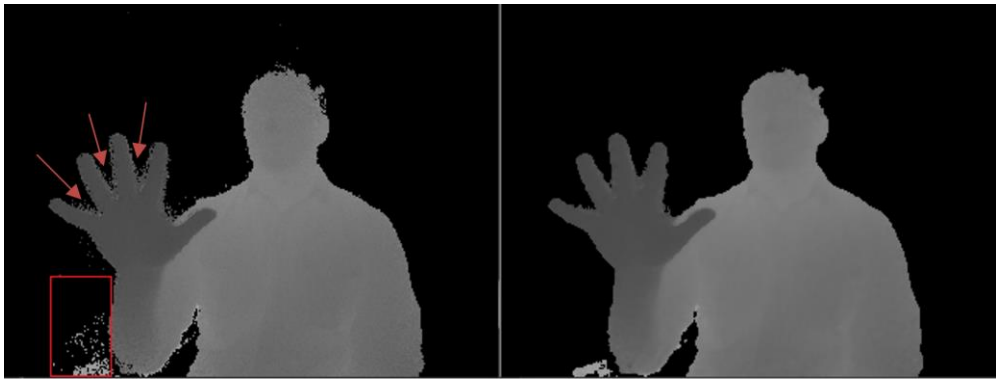
Visą programos ciklo seką galima suskirstyti į kelis etapus: duomenų paėmimas iš kameros ir jų paruošimas, RGB ir gylio kadrų atitinkamų taškų susiejimas, norimo regiono (rankos) iškyrimas ir analizavimas. Kiekviena iš išvardintų dalių vykdoma nuosekliai. Nors algoritmu galime laikyti du paskutiniuosius minėtus etapus, tačiau pirmasis žingsnis be galo svarbus visiems tolesniems skaičiavimams. Programos veikimas toliau aprašytas nuoseklia tvarka.

Pirmasis etapas yra reikalingų duomenų surinkimas iš kameros. Darbe kuriamam algoritmui reikalingi RGB ir gylio kamerų kadrai. Šie jutikliai veikia nepriklausomai ir ***DepthSense SDK*** programinė įranga, kurią naudojant valdoma kamera, neturi funkcijos, grąžinančios sinchronizuotus RGB ir gylio kadrus. Dėl šios priežasties kadrų sinchronizavimas vyksta pagal gylio kadro gavimo įvykį, tai yra, kai gaunamas nauja nuotrauka iš gylio kameros, imama naujausia nuotrauka iš RGB kameros buferio (4.1 pav.).



4.1 pav. *Sinchronizuoti RGB ir gylio kameros kadrai*

Gylio kadras dėl matavimo netikslumų ir trikdžių dažnai gaunamas su šiukšlėmis, kurios gali lemti tolesnius skaičiavimus bei rezultatus, kadangi iškraipo rankos kontūrą (4.2 pav.).

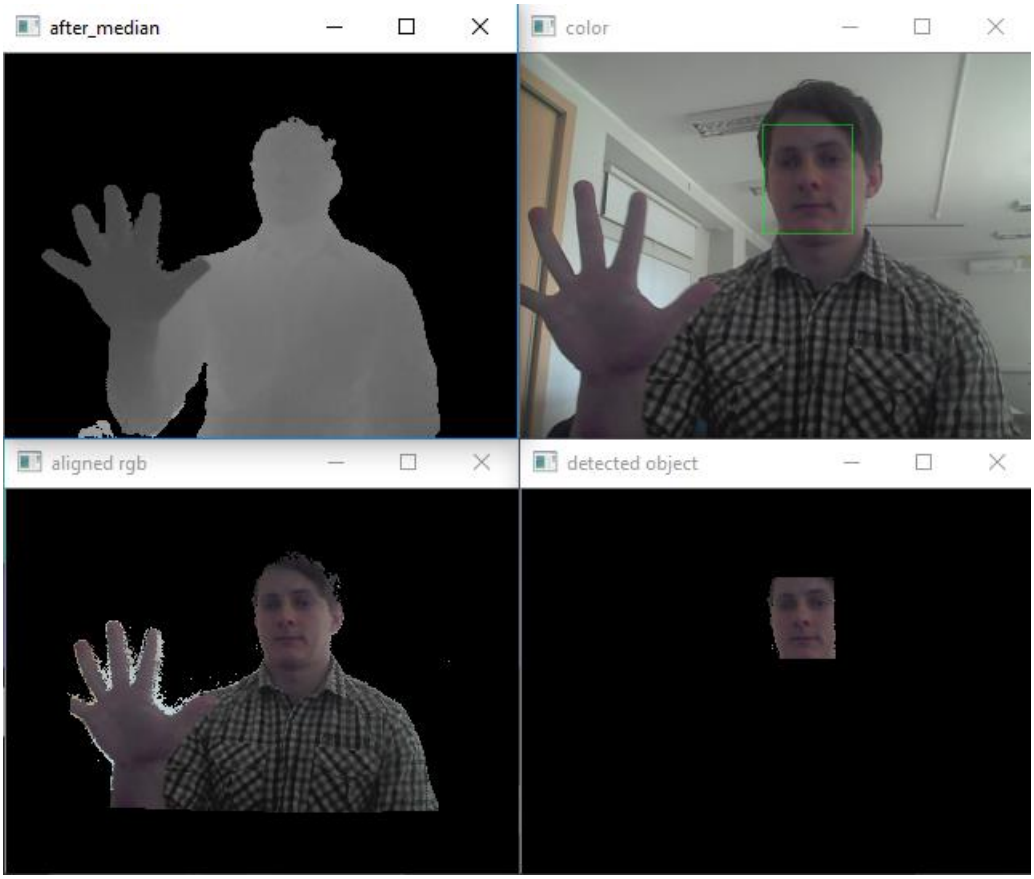


a) **b)**

4.2 pav. *a – kadras iš gylio kameros su parodytu triukšmu, b – su pritaikytu medianos filtru*

Tokio tipo triukšmas efektyviai pašalinamas medianos filtru, triukšmo taško vertei skaičiuojant vidurkį su aplinkinių taškų reikšmėmis. Taip pat, šiuo metodu šiek tiek palyginami visi kontūrai. Toliau duomenys perduodami tolesnėms funkcijoms.

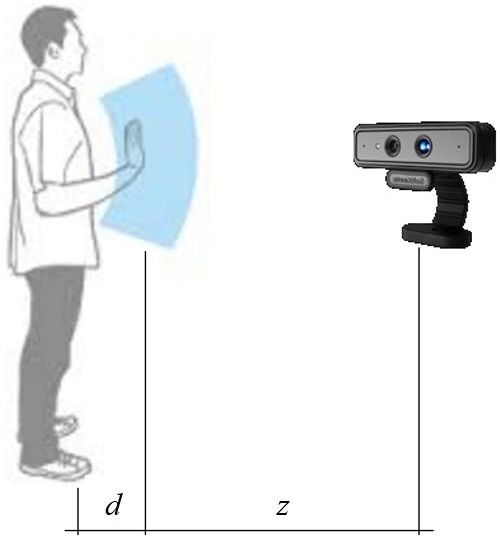
Kitu žingsniu RGB (spalvotoje) nuotraukoje ieškoma veido. Tai savotiškas atskaitos taškas, nuo kurio gylio kadre bus ieškoma ranka ir analizuojama jos forma. Veido paieškai darbe išbandyti trys įrankiai: ***OpenCV Haar kaskadų klasifikatorius*** (*haarcascade_frontalface_alt2.xml*), ***OpenFace*** [28], ***CNN kaskados*** [33]. ***OpenCV Haar klasifikatorius*** ir ***CNN kaskados*** veikia vaizdo plokštėje ir naudoja CUDA paralelaus skaičiavimo technologiją. ***OpenFace*** veido paieškai realizuota naudojant **HOG** ir **HAAR** klasifikatorius iš **OpenCV** ir **DLib** bibliotekų. Deja, šios programinės įrangos atveju negalima panaudoti **CUDA** technologijos, kadangi ***DLib*** reikalingas C++11 kalbos atributai, kurie nepilnai įdiegti Visual Studio 2013 aplinkoje [50]. Šios bibliotekos veido paieškos algoritmas vykdomas tik pagrindiniame procesoriuje (CPU). Visų minėtų įrankių tikslumo ir greitaveikos pateiktas 5 (eksperimentų) skyriuje. Naudojant bet kurį iš būdų randamas veido plotas, kuris pagal 2.2 skyriuje pateiktą skaičiavimo būdą perkeliamas į gylio kadrą (randami atitinkantys taškai) (4.3 pav.).



4.3 pav. *Viršuje kairėje – gylio kameros kadras; viršuje dešinėje – RGB kadras su aptiktu veidu; apačioje kairėje – ant gylio kadro užpiešti atitinkami RGB taškai; apačioje dešinėje – veido pikseliai gylio kadre, iš kurių skaičiuojamas vidutinis atstumas*

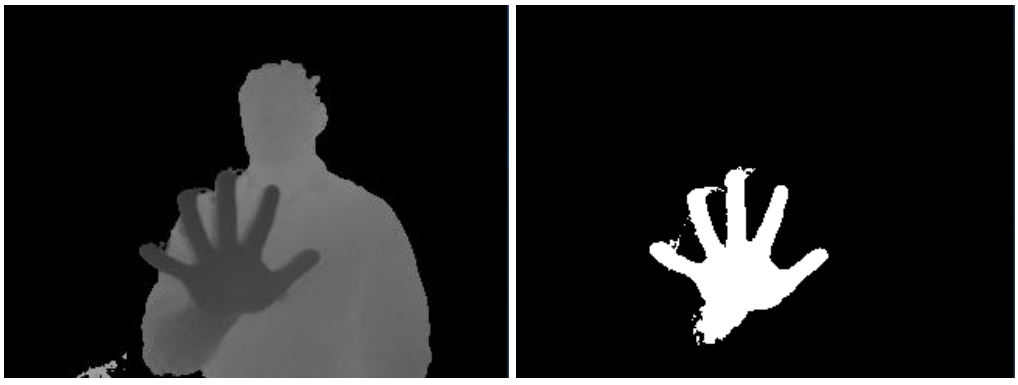
Ne visi veido ploto taškai gylio kadre turi reikšmes, kai kurie yra neišmatuoti ir lygūs 0. Pavyzdžiui, taškai esantys už operatoriaus 4.3 paveiksle yra nutolę per 4 - 9 metrus, o kameros darbinis atstumas yra 15 - 100 centimetrų (kamera išduota atstumą iki objektų, nutolusių ~140cm). Todėl neišmatuoti pikseliai atmetami ir neįtraukiami į vidutinio atstumo iki operatoriaus skaičiavimą. Kai

turime atstumą tarp žmogaus ir operatoriaus, galime atmesti visą likusią informaciją ir analizuoti tik tą, kuri patenka į šį diapazoną. Tačiau analizuojamą erdvę galima dar labiau sumažinti. Kaip parodyta 4.4 paveiksle žmogus natūraliai gestą rodo šiek tiek toliau nuo savo kūno. Dėl to sumažėja analizuojama zona *z*. Šiame darbe gesto rankos aptikimo zona sumažinama *d = 20cm*.



4.4 pav. *Žmogaus gesto rodymo zona*[8]*. z – gesto aptikimo zona, d – atstumas tarp operatoriaus ir gesto rodymo zonos pradžios*

Kai turime diapazoną (4.4 pav. *z*), kuriame galime ieškoti rankos, reikia surinkti visus į jį patenkančius gylio kadro taškus. Padaroma binarinė matrica: jei taškas patenka į diapazoną – jo reikšmė lygi 1 (255), jei ne – 0 (0). Pavyzdys parodytas 4.5 paveiksle.



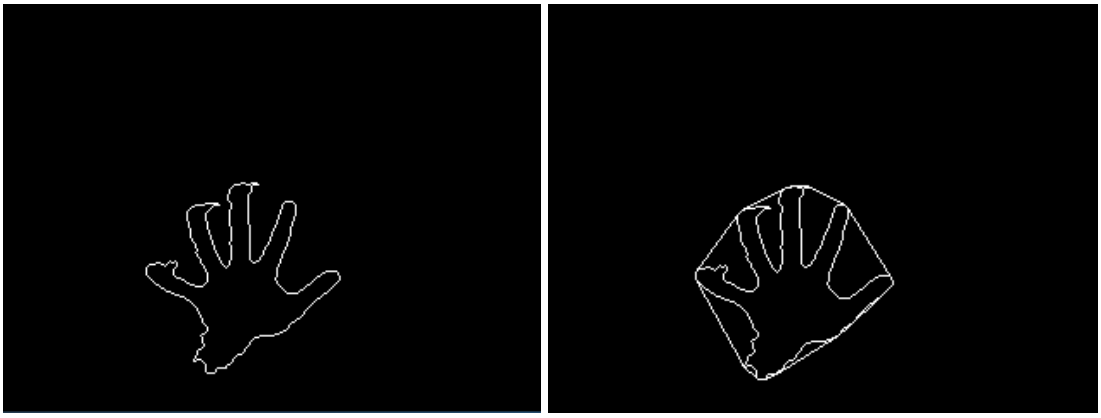
a) **b)**

4.5 pav. *a - gylio kadras, b - binarinė matrica, kurioje baltai pažymėti taškai, patenkantys į analizuojamą diapazoną*

Kita funkcija – binariniame kadre esantiems objektams ieškomi kontūrai (4.6 pav. kairėje). Dažnai tai ne tik rankos siluetas, bet ir kiti maži objektai, atspindžiai ar trikdžiai. Po kontūrų radimo operacijos

[8] http://elekslabs.com/2015/12/awaken-the-force-with-kinect-and-sphero.html

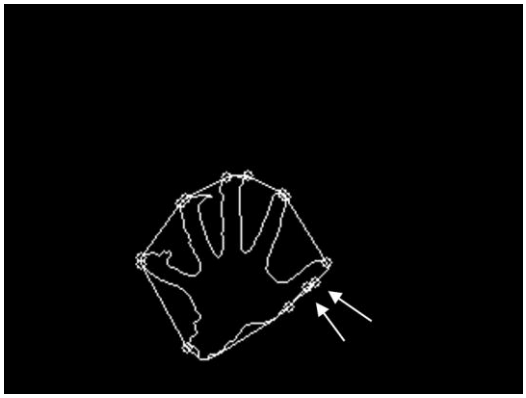
atmetami tie, kurių plotas mažesnis nei 2000 taškų. Tokia riba parinkta kelių testų metu. Ji nėra visiškai apibrėžta, galima pritaikyti pagal kamerą. ***DepthSenseDS325*** atiduoda 76800 (320x240) taškų nuotrauką, taigi 2000 lemia tai, kad aptiktas objektas kadre yra gana didelis.



a) **b)**

4.6 pav. *a – rankos kontūras, b – apibrėžtas daugiakampiu rankos kontūras (convex hull)*

Turimas rankos kontūras yra taškų masyvas, apie kurį brėžiamas daugiakampis (***convex hull***, 4.6 pav. dešinėje) (plačiau metodas aprašytas 2.5 skyriuje.). Taip galime analizuoti skirtumus tarp šių dviejų formų (***convexity defects***). Rankos kontūro atveju, tai daugiausiai tarpai tarp vienas nuo kito atitrauktų pirštų. Deja, taikant ***convexity defects*** funkciją išryškėja ir kiti nedideli skirtumai tarp rankos ir apibrėžto daugiakampio. Kadangi kontūras nėra lygus, todėl algoritmas aptinka daugiau taškų (4.7 pav.).

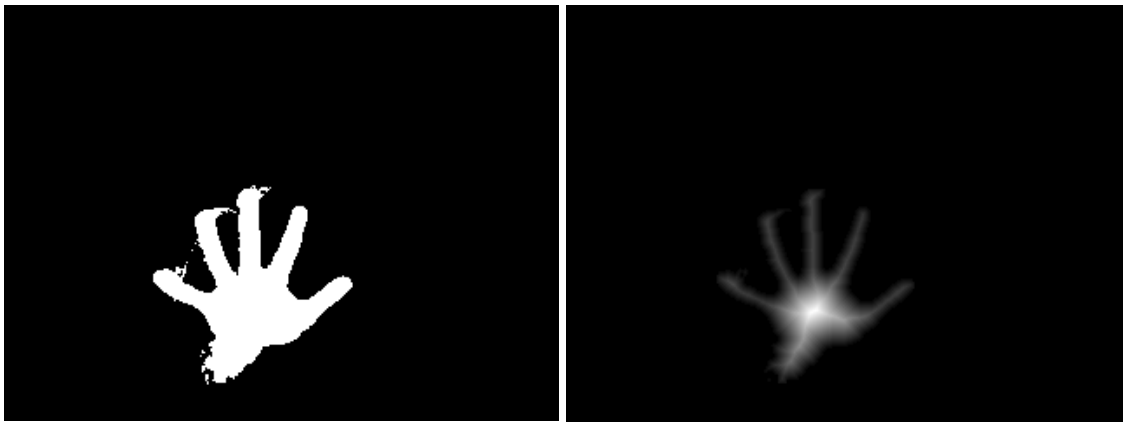


4.7 pav. *Įdubų pradžios ir pabaigos taškai tarp rankos kontūro ir daugiakampio pažymėti mažais apskritimais. Rodyklėmis pažymėti nereikšmingų (mažų) įdubų taškai*

Tokie mažų įdubų taškai yra klaidingi ir trukdo tolesniems skaičiavimas, todėl juos reikia pašalinti. Teisingų taškų atrinkimas vyksta lyginant tolimiausio įdubos taško atstumą iki apibrėžto daugiakampio (2.5.2 pav. mėlynos rodyklės). Jei atstumas didesnis nei užduotasis (šio darbo atveju 10 pikselių), įdubos ekstremumų taškai įtraukiami į masyvą tolimesniam analizavimui.

Naudojant kitą algoritmą, pagal binarinę matricą, ieškome plaštakos centro. Tai labai svarbus žingsnis, kuris leidžia nustatyti pirštais rodomo gesto erdvę bei rankos padėtį erdvėje. Centro taškas

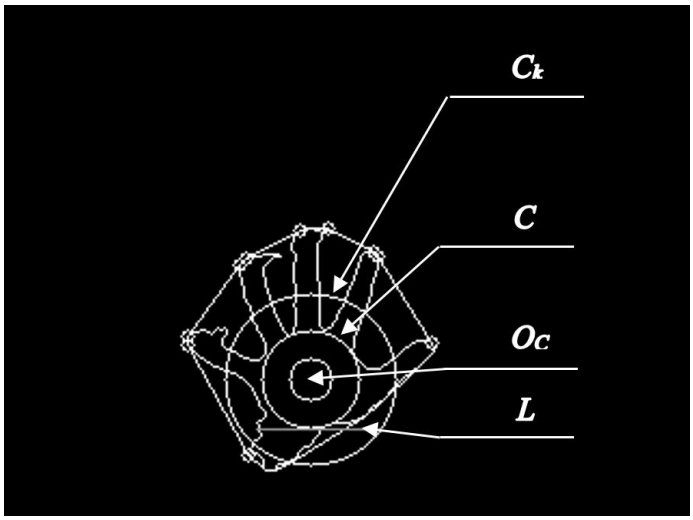
randamas remiantis tuo, kad plaštaka arba kumštis yra pati plačiausia vieta tarp riešo ir pirštų galų. Kitaip sakant, delno ar kumščio centro taškas turėtų būti labiausiai nutolęs nuo bet kurio kontūro taško. Tokią vietą galime rasti pritaikę atstumo transformaciją. Platesnis metodo aprašymas pateiktas 2.4 skyriuje. Pritaikę funkciją gauname matricą su skaičiais, kurių kiekvieno reikšmė yra nuotolis nuo kontūro. Kuo ryškesnis taškas, tuo didesnė jo vertė. Pavyzdys pateiktas 4.8 paveiksle.



a) **b)**

4.8 pav. *a - binarinis paveikslas, b – pateiktam binariniam paveikslui atlikta atstumo transformacija*

Turint atstumo transformacijos matricą, galima rasti elementą su didžiausia reikšme. Reikia išsaugoti taško (4.9 pav. O_C) x, y koordinates ir vertę. Ji yra apskritimo (*C*, įbrėžto į rankos kontūrą, spindulys *R* (4.9 pav.)).

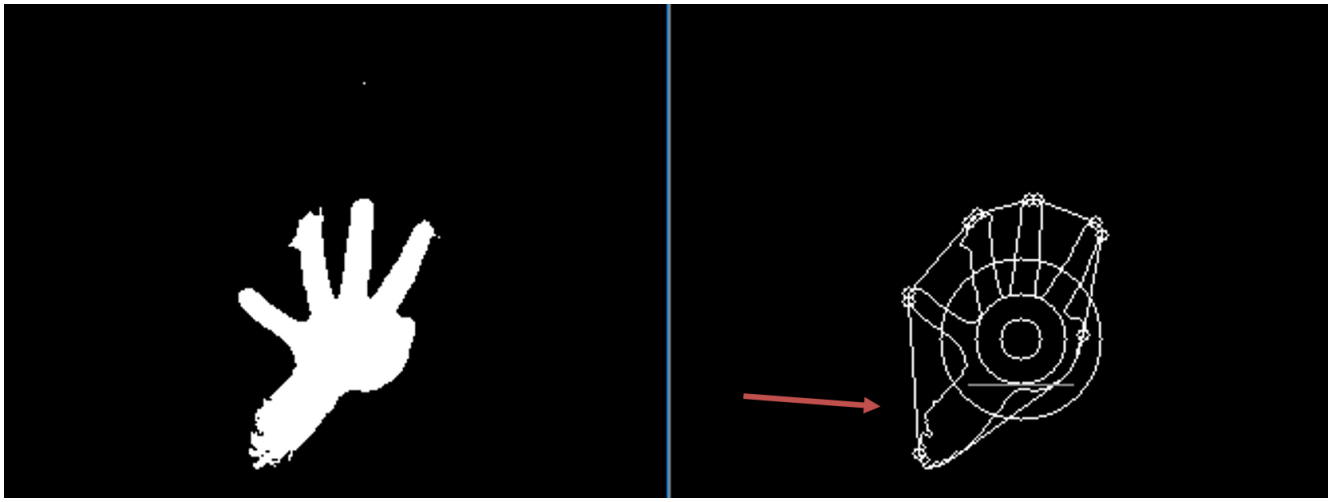


4.9 pav. *Delno centro taškas - O_C; apskritimas, įbrėžtas į rankos kontūrą - C; apskritimas, už kurio ribos yra priimama, kad tai yra pirštas - C_k; riba, virš kurios analizuojami pirštai*

Turint tik įbrėžtinį apskritimą *C*, negalime spręsti apie tai, kiek pirštų yra atlenkta, nes nors ir pirštas užlenktas, įdubos pradžios ir pabaigos taškai visada bus apskritimo išorinėje dalyje. Todėl parenkamas didesnis C_k apskritimas su tuo pačiu centru. Tik esantys apskritimo išorėje taškai yra laikomi pirštų galais. Šiame darbe didesniojo apskritimo spindulys yra **1.6** karto didesnis nei apskritimo *C*, bet tai gali

varijuoti priklausomai nuo poreikių. Prie darbe naudojamos ribos prieita bandymu keliu su 300 kadrų, kai atlenktas skirtingas pirštų skaičius, skirtingame nuotolyje nuo kameros. Labai panašus veikimo našumas gautas naudojant ***1.5*** ir ***1.7*** karto didesnį apskritimą C_k.

Toliau turime surasti kiekvieno aptikto piršto galo koordinates. Kaip matome 4.9 paveiksle, piršto galą dažniausiai apibrėžia 2 taškai: vienos įdubos pabaiga ir kito pradžia. Šių dviejų taškų vidurkis yra piršto galas. Kai turime visus galimus taškus, reikia patikrinti, ar pirštas yra atlenktas, tai yra, ar jį charakterizuojantis taškas yra už apskritimo C_k ribų. Tai daroma skaičiuojant Euklido atstumą nuo plaštakos centro iki piršto galo. Jei atstumas mažesnis už C_k spindulį, tada tas taškas atmetamas. Be to, dėl kontūro nelygumų gali atsirasti tokių įdubų, kurių pražios ar pabaigos taškai nėra pirštai. Pavyzdžiui, ties dilbiu ar riešu (4.10 pav.) gali atsirasti tokių vietų, kurios anksčiau aprašytomis funkcijomis bus interpretuojamos kaip didelė įduba, o vėliau kaip atlenktas pirštas. Todėl šiame darbe priimta prielaida, jog rodomas gestas neturi būti paverstas, visi pirštai turi būti plaštakos lygmenyje. 4.9 paveiksle pavaizduota riba *L*, žemiau kurios esantys pirštų taškai atmetami. Minėta riba yra delno centro taško O_C y koordinatės ir įbėžtojo apskritimo *C* spindulio skirtumas.



4.10 pav. *Rodomo gesto binarinė matrica ir kontūras su įdubų ekstremumų (pradžios ir pabaigos) taškais. Rodykle parodyti nelygumai, galintys lemti klaidingą pirštų atpažinimą*

Tai visas darbe kurtas ir tirtas algoritmas. Jo įėjimu galime laikyti sinchronizuotus RGB ir gylio kameros, o išėjimu – rankų vektorių, kuris sudarytas iš kiekvienai rankai priklausančių taškų: plaštakos centro ir pirštų galų taškų. Sukurtas metodas gerai atpažįsta statinius rankos gestus (kiek pirštų rodo operatorius). Tikslumo analizė pateikta eksperimentų skyriuje. Algoritme daugiausiai vykdymo laiko ir resursų reikalauja veido paieška RGB kadre. Visų testuotų įrankių veido atpažinimui tikslumas ir greitaveika bus palyginti tyrimų skyriuje.

5. Eksperimentiniai tyrimai

Atlikti keli eksperimentai: veido aptikimui naudojamų įrankių tikslumo ir greitaveikos bei sukurto rankos gestų atpažinimo algoritmo tikslumo nustatymo. Aprašymai, eiga, sąlygos ir rezultatai pateikti toliau.

5.1 Veido aptikimo įrankių greitaveikos eksperimentas

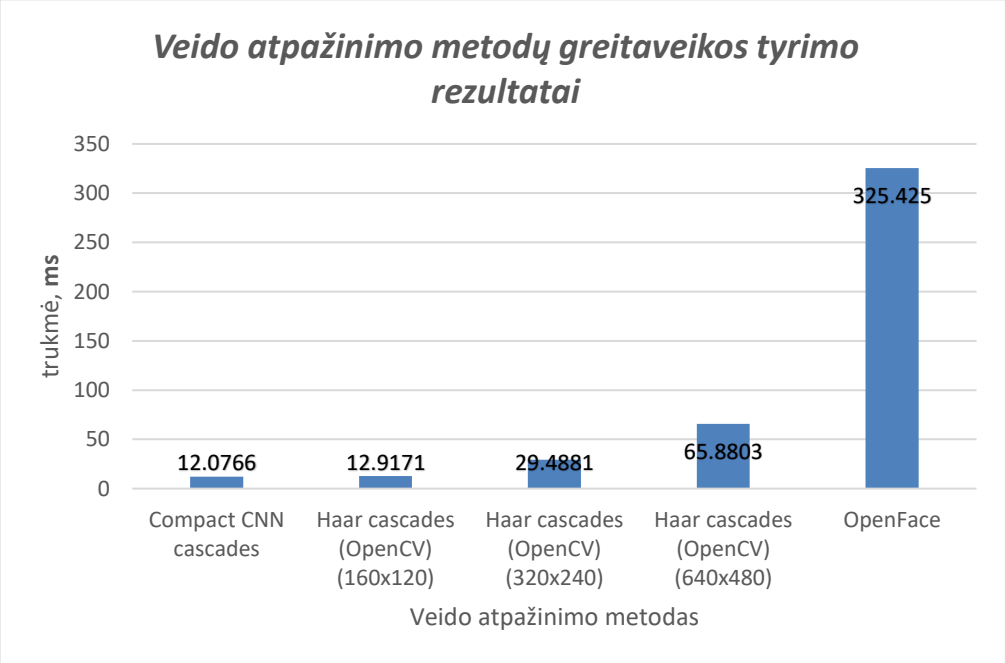
Tikslas: nustatyti kiekvieno naudojamo metodo veidui atpažinti greitaveiką.

Eksperimento aprašymas.

Iš 300 nuotraukų, gautų iš **DepthSenseDS325 kameros**, nustatyti veido atpažinimų metodų greitaveikas naudojant ***Haar kaskadų klasifikatorių*** [26] iš ***OpenCV*** (*haarcascade_frontalface_alt2.xml*) (šis klasifikatorius tiriamas su skirtingo dydžio nuotraukomis), ***OpenFace*** [28] ir **Compact CNN cascades** [33] algoritmais. Kai veidas atpažįstamas, skaičiuojama suminė algoritmo veikimo trukmė ir dalinama iš kadrų, kuriuose lokalizuojamas veidas, skaičiaus. Vidutinės trukmė skaičiavimas pateiktas 18 formulėje.

$$T_{vid} = \frac{\sum_{i=1}^{n} T_i}{N_{atpažintų}} \qquad (18)$$

$\boldsymbol{T_{vid}}$ *– vidutinė veido atpažinimo trukmė,* $\boldsymbol{T_i}$ *– i kadre atpažinto veido trukmė,* $\boldsymbol{N_{atpažintų}}$*- kadrų su atpažintu veidu skaičius,* $\boldsymbol{n}$ *– tiriamų kadrų skaičius (300)*



5.1.1 pav. *Veido atpažinimo metodų greitaveikos tyrimo rezultatų grafikas*

OpenCV Haar kaskadų klasifikatorius bei ***Compact CNN cascades*** naudoja vaizdo plokštės procesorių, ***CUDA*** technologiją, kuri žymiai paspartina veikimą. Tik ***OpenFace,*** dėl nepilno C++11

atributų palaikymo ***Visual Studio 2013*** aplinkoje, negali panaudoti minėtos technologijos, taigi veido aptikimo skaičiavimai vyksta centriniame procesoriuje (CPU). Visų aptartų metodų vidutinės veikimo trukmės pateiktos 5.1.1 paveikslo grafike. Tyrime testuota paveikslo dydžio įtaka **HAAR kaskadų klasifikatoriui.** Imti dydžiai 160x120, 320x240, 640x480. Į šio klasifikatoriaus trukmės skaičiavimą taip pat įtraukti paveikslėlio dydžio ir spalvų aibės keitimo (RGB į juodai-baltą), duomenų siuntimo/gavimo į/iš vaizdo plokštės atminties vykdymo laikai (spalvų aibių keitimo funkcijų vykdymo laikai įtraukti ir į kitų metodų vidutinės trukmės skaičiavimą).

Eksperimento išvados

Greičiausiai veikia ***Compact CNN cascades*** veido aptikimo algoritmas. Į funkciją atiduodant 640x480 dydžio paveikslą, jis veikia 5 kartus greičiau nei ***HAAR kaskadų klasifikatorius*** (naudojant ***GPU CUDA***). Pastarojo metodo greičiui įtakos turi kadro dydis. Mažinant nuotraukos aukštį ir plotį, vykdymo laikas mažėja. ***OpenFace*** vidutinę aptikimo trukmę galime laikyti nekorektiška, kadangi algoritmo veido nustatymui įtakos turi kadrų seka. Jei veidas aptiktas, metodas išduoda jo koordinates gana greitai ir keliose tolesniuose kadruose veido ieško pagal atributų taškus (greitesniu algoritmu). Bet neradus veido ***OpenFace*** metodas kelis kartus skanuoja tą patį kadrą su ***HOG*** ir ***HAAR*** klasifikatoriais, o tai užtrunka iki ~1 sekundės.

5.2 Veido atpažinimo metodų tikslumo eksperimentas

Tikslas: nustatyti kiekvieno naudojamo metodo veidui atpažinti tikslumą.

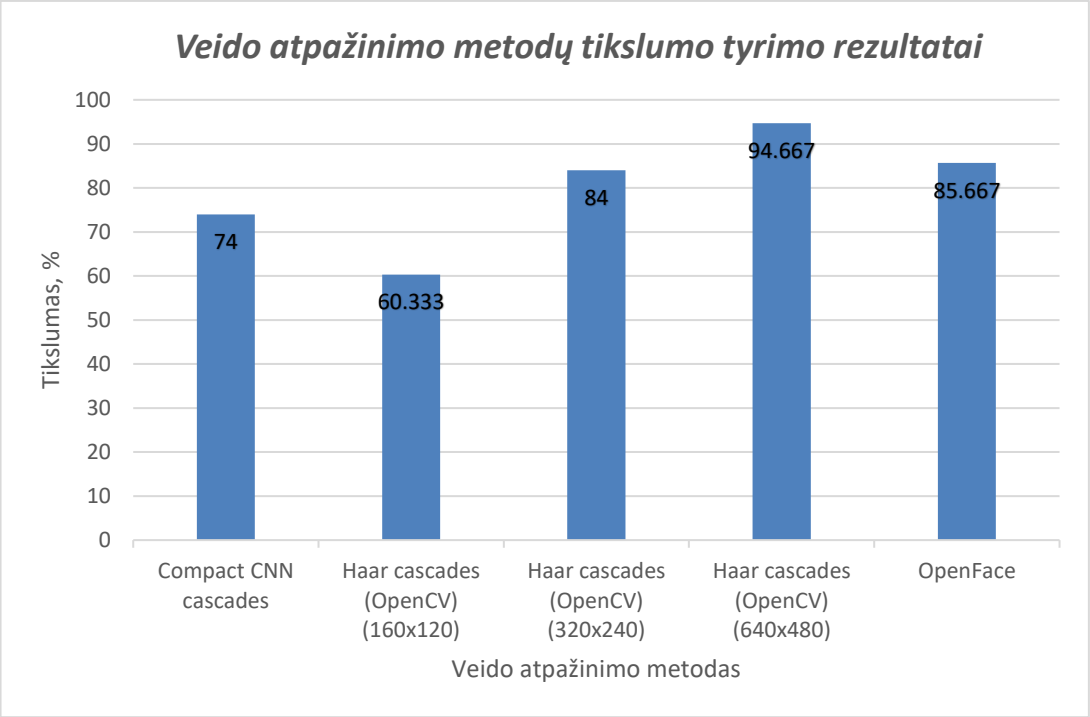
Eksperimento aprašymas

Iš 300 nuotraukų, gautų iš **DepthSenseDS325 kameros**, nustatyti veido atpažinimų metodų tikslumus naudojant ***Haar kaskadų klasifikatorių*** [26] iš ***OpenCV*** (*haarcascade_frontalface_alt2.xml*) (šis klasifikatorius tiriamas su skirtingo dydžio nuotraukomis), ***OpenFace*** [28] ir **Compact CNN cascades** [33] algoritmais. Su kiekvienu iš išvardintų metodų skaičiuojama, keliuose kadruose aptiktas veidas. Tyrimo duomenys pateikti 5.2.1 lentelėje. Visose nuotraukose yra tas pats asmuo, skirtingai savo veidą nukreipęs į kamerą.

5.2.1 lentelė. *Teisingų veido lokalizavimų skaičius*

Metodo pavadinimas	Teisingų lokalizavimų skaičius
Compact CNN cascades	222
Haar cascades (OpenCV) (160x120)	181
Haar cascades (OpenCV) (320x240)	252
Haar cascades (OpenCV) (640x480)	284
OpenFace	257

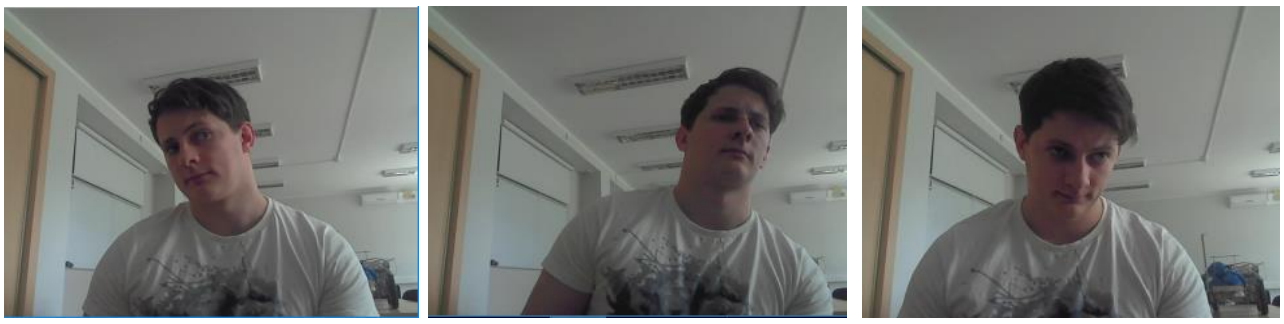
Haar kaskadų klasifikatorius iš ***OpenCV*** bibliotekos patirtas šiek tiek išsamiau. Į šį metodą buvo paduodama 3 skirtingų dydžių kadrai (žr. 5.2.1 lentelė ir paveikslą) ir tikrintas algoritmo tikslumas su kiekvienu iš jų.



5.2.1 pav. *Veido atpažinimo metodų tikslumo tyrimo rezultatų grafikas*

Eksperimento išvados

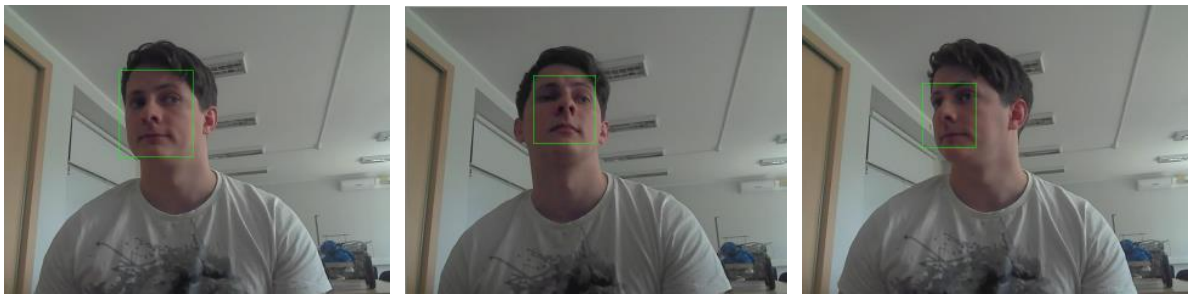
Tiksliausiai veikia ***HAAR kaskadų klasifikatorius*** iš **OpenCV**, kai į metodą kraunamas pilno dydžio kadras (640x480). Mažinant nuotrauką, tikslumas taip pat mažėja. Tačiau dalis tyrimų duomenų bazėje naudotų kadrų yra su žymiai į šoną pasuktu veidu (5.2.2 pav.). Ne visuose kadruose prieš kamerą sėdintis žmogus žiūrėjo tiesiai į ją.



5.2.2 pav. *Keli tyrimui naudoti kadrai, kuriuose naudojant* ***Compact CNN cascades*** *metodą veidas neaptinkamas*

Pavyzdžiui, veido aptikimui naudojant ***Compact CNN cascades*** įrankį, veidas ne visada aptinkamas minėtose padėtyse (5.2.2 pav.). Bet daugelyje kadrų, net ir kamerai nematant tiesiai į ją nukreipto veido,

pastarasis yra gerai lokalizuojamas (5.2.3 pav.). Šio tyrimo metu visi naudoti metodai parodė gerus rezultatus, nepaisant to, kad duomenų bazę naudota testavimui nebuvo visiškai ideali.



5.2.3 pav. *Keli tyrimui naudoti kadrai, kuriuose naudojant **Compact CNN cascades** metodą veidas aptinkamas*

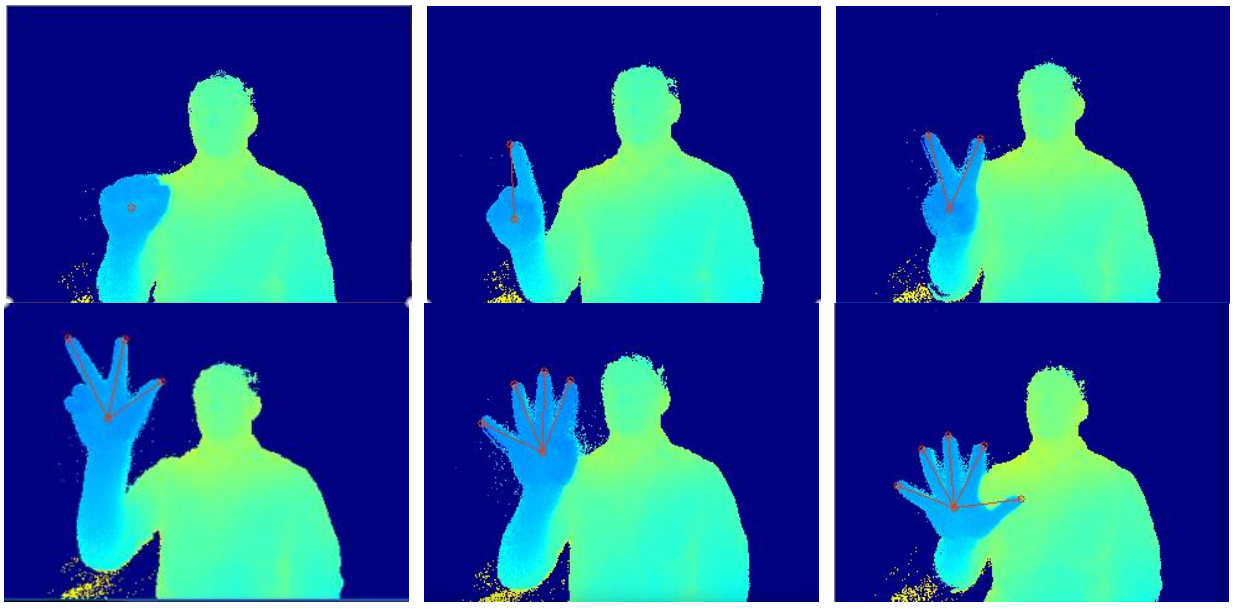
Be to, galima nenaudoti šių algoritmų veido atpažinimui kiekvienoje nuotraukoje. Tarp nustatytų kadrų žmogų galime sekti kitais algoritmais.

5.3 Statinių rankos gestų atpažinimo algoritmo tikslumo nustatymo eksperimentas

Tikslas: nustatyti kiekvieno iš rodomų gestų atpažinimo tikslumą.

Eksperimento aprašymas

Tiriama 6 statiniai gestai (5.3.1 pav.), atlenktų pirštų skaičius. Žmogaus atpažinimui kadre pasirinktas ***Compact CNN cascades*** [33] metodas. Kiekvieno gesto tyrimo metu imama 300 kadrų, kai plaštaka yra įvairiose kadro padėtyse. Gestas yra analizuojamas tada, kai veidas lokalizuojamas nuotraukoje. Jei žmogus neaptinkamas, tame kadre esantis gestas neįtraukiamas į statistiką.



5.3.1 pav. *Tirti gestai*

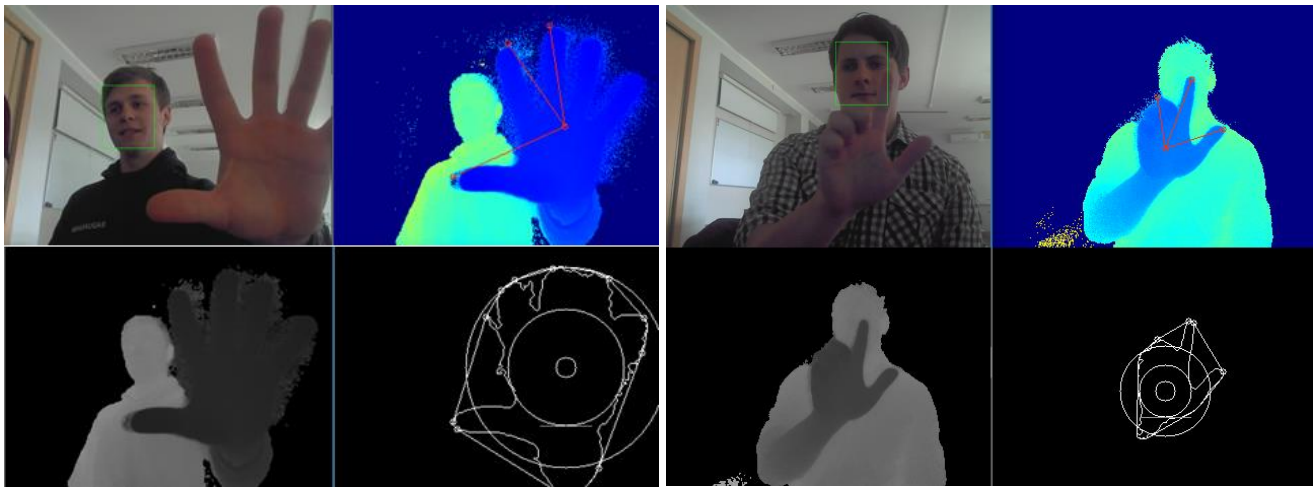
Tyrime dalyvauja 3 asmenys (2 vyrai ir 1 moteris). Kiekvieno žmogaus, dalyvavusio tyrime, rodomi gestai patalpinti ir analizuoti iš skirtingų įrašų. Tyrimo rezultatai pateikti 5.3.1 lentelėje. Joje nurodyta tyrimas su kiekvieno žmogaus rodomais gestais bei vidurkis.

5.3.1 lentelė. *Gestų atskyrimo tyrimo rezultatai*

Atlenktų pirštų skaičius	***Tesingi gesto aptikimai***			***Tikslumas, %***			
	I	**II**	**III**	**I**	**II**	**III**	**Vidurkis**
0	289	281	278	96.33	93.67	92.67	94.22
1	289	282	288	96.33	94.00	96.00	95.44
2	281	291	286	93.67	97.00	95.33	95.33
3	265	260	279	88.33	86.67	93.00	89.33
4	278	275	279	92.67	91.67	93.00	92.44
5	290	291	279	96.67	97.00	93.00	95.56

Eksperimento išvados

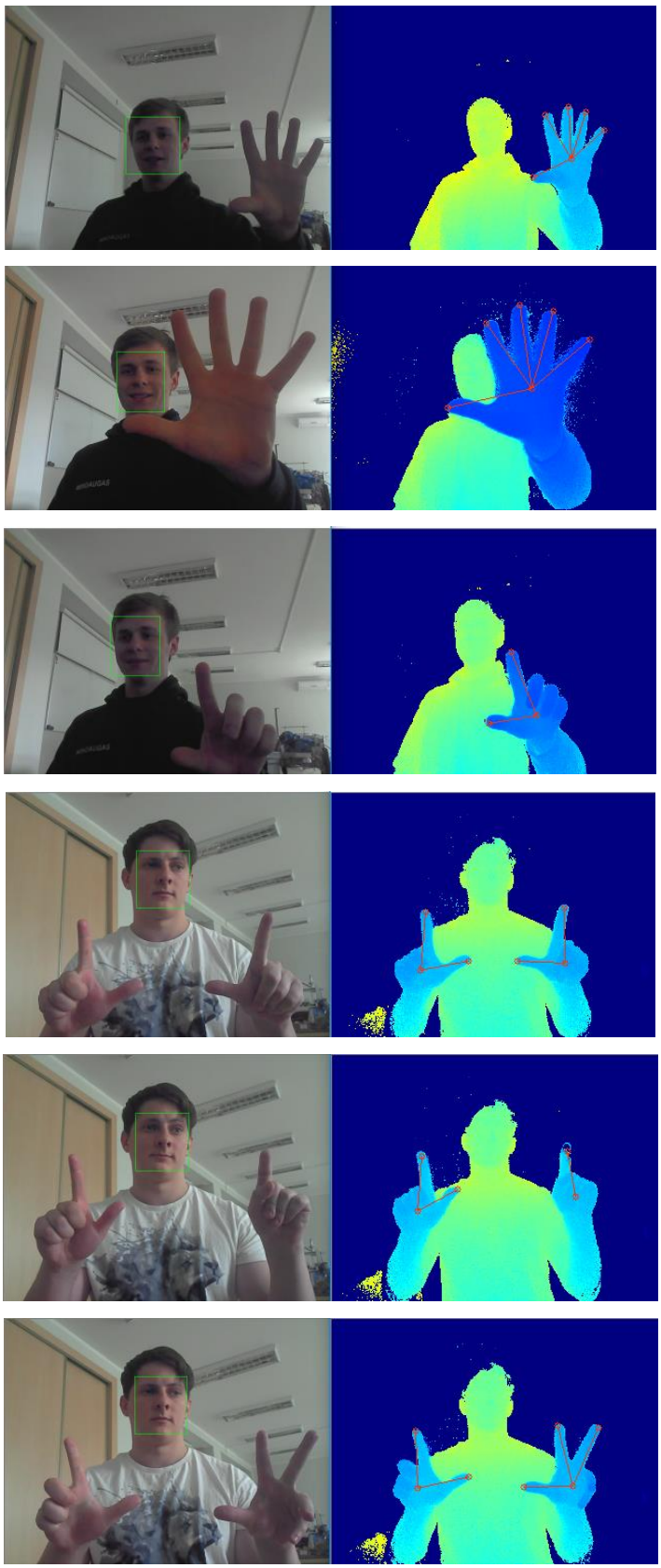
Eksperimento rezultatai rodo, kad algoritmas veikia gerai. Dinamiškai rodant gestus, tai yra, kai ranka yra nuolat judinama, metodo tikslumas daugumai gestų siekia 95%. Rodant 3 pirštus, vidutinis atpažinimo tikslumas žemiausias – 89.33%. Tačiau, kai kuriems rezultatams įtakos turėjo per arti parodyti gestai ar per arti vienas prie kito laikomi atlenkti pirštai (5.3.2 pav.).



a) **b)**

5.3.2 pav. *Blogai atpažinti gestai. a – per arti parodytas gestas, b – ne visai atlenkti piršai interpretuojami kaip atlenkti*

Nepaisant kelių kadrų, kuriuose blogai aptiktas rankos gestas, algoritmo tikslumas yra aukštas. Gerai aptikti gestai pateikti 5.3.3 paveiksle. Taikant tolimesnius filtravimus, ar gestą patvirtinant remiantis kelių kadrų informacija, galima jį dar labiau padidinti.



5.3.3 pav. *RGB kameros duomenys su atpažintu veidu ir gylio kameros duomenys su atpažintu delno centru ir pirštų galų taškais*

Rezultatai ir išvados

Rezultatai

Šiame darbe sukurtas algoritmas gestams atpažinti. Vieno kadro su rodomu gestu apdorojimas trukmė 5-15 milisekundžių. Labiausiai trukmę įtakoja aptiktos plaštakos dydis (t.y. plaštakų plotas), nes taikomos atstumo transformacijos [39] funkcijos skaičiavimas yra vienas sunkiausių ir priklauso nuo binarinėje matricoje esančių objektų dydžio (taškų kiekio). Kuo plaštaka arčiau kameros, tuo didesnį plotą ji užima. Veido atpažinimui naudojant greitą metodą, pavyzdžiui, ***Compact CNN cascades*** [33] metodą, algoritmas veikia realiu laiko, tai reiškia, kad gali apdoroti 25-30 kadrų per sekundę. Skaičiavimų trukmė su veido atpažinimu trunka apytiksliai 17-27 milisekundes. Tačiau nepriklausomus skaičiavimus paleidus lygiagrečiai, galima pasiekti dar geresnę greitaveiką. Tai galima padaryti skirtinguose branduoliuose analizuojant kontūro įdubas ir taikant atstumo transformacijos operaciją. Be to, veido aptikimas gali būti vykdomas ne kiekvienam kadre, priimant sąlygą, kad žmogus per užduotą kadrų skaičių nejuda ar juda mažai. Bet, nepaisant to, sukurtas algoritmas yra pakankamai greitas ir gali būti naudojamas net mobiliuose įrenginiuose.

Taip pat, pasiektas gana didelis algoritmo tikslumas. Daugelyje eksperimentų jis siekia net 95%, kai gestas rodomas įvairiose padėtyse ir ranka juda (5.3.3 pav.). Keli kadrai, kuriose gestas identifikuotas neteisingai, su kameros triukšmu arba pirštai dalinai atlenkti ar neužlenkti (5.3.2 pav.).

Kadrai iš tyrimo su aptiktais gestais pateikti eksperimentų skyriuje (5.3.3 pav.). Gestų atpažinimo vaizdo medžiaga pateikta pridėtame DVD diske.

Be pateiktų rezultatų, demonstraciniams tikslams sukurta aplikacija su ***OpenCV Viz*** moduliu [51]. Tai viena iš aprašyto gestų atpažinimo algoritmo panaudojimo galimybių. Šiame projekte dvejais rankų gestais, sukniaužtu kumščiu ir 2 ištiestais pirštais, 3D objektas sukamas ir slenkamas erdvėje. Keli kadrai iš demostacinio projekto pateikti 1 priede ir vaizdo įrašas DVD diske.

Išvados

1. Sukurtas rankų gestų atpažinimo algoritmas, kuris naudoja RGB ir gylio kameros duomenis.
2. Veido atpažinimo metodai pagreitinti su ***CUDA*** technologija.
3. Iš testuotų veido atpažinimo metodų greičiausia veikia ***Compact CNN cascades*** [33], kuris veidą aptinka per vidutiniškai 12 milisekundžių.
4. Rankos gesto analizė trunka 5-15 milisekundžių, priklausomai nuo objekto kadre dydžio.
5. Sukurtas rankų gestų atpažinimo algoritmas veikia realiu laiku, apdoroja 25-30 kadrus per sekundę.
6. Algoritmo tikslumas siekia 95%.

Informacijos šaltiniai

[1] “Virtual reality apibrėžimas.” [Prieiga per internetą]. Internetinis adresas: https://en.wikipedia.org/wiki/Virtual_reality. [Žiūrėta: 12-05-2017].

[2] “Augmented reality apibrėžimas.” [Prieiga per internetą]. Internetinis adresas: https://en.wikipedia.org/wiki/Augmented_reality. [Žiūrėta: 12-05-2017].

[3] “Microsoft Kinect SDK aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://developer.microsoft.com/en-us/windows/kinect. [Žiūrėta: 12-05-2017].

[4] “Microsoft Kinect SDK skeletono taškų aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx. [Žiūrėta: 12-05-2017].

[5] “Leap motion aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://developer.leapmotion.com/. [Žiūrėta: 12-05-2017].

[6] “Softkinetic Close Interaction Library aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://www.softkinetic.com/products/CIlib. [Žiūrėta: 12-05-2017].

[7] “DepthSense DS325 aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://www.softkinetic.com/products/depthsensecameras. [Žiūrėta: 12-05-2017].

[8] “Caffe giliojo mokymo bibliotekos aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://github.com/BVLC/caffe/wiki. [Žiūrėta: 14-05-2017].

[9] “Microsoft Cognitive Toolkit bibliotekos aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://github.com/Microsoft/CNTK/wiki. [Žiūrėta: 14-05-2017].

[10] “Deep Hand giliojo mokymo modelis Caffe aplinkoje.” [Prieiga per internetą]. Internetinis adresas: http://www-i6.informatik.rwth-aachen.de/~koller/1miohands/. [Žiūrėta: 14-05-2017].

[11] O. Koller, H. Ney, and R. Bowden, “Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data Is Continuous and Weakly Labelled,” *Comput. Vis. Pattern Recognit.*, 2016.

[12] “CUDA SDK aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://developer.nvidia.com/cuda-downloads. [Žiūrėta: 15-05-2017].

[13] “DLib vaizdo atpažinimo ir dirbtinio intelekto biblioteka.” [Prieiga per internetą]. Internetinis adresas: http://dlib.net/. [Žiūrėta: 15-05-2017].

[14] Wei Wang and Jing Pan, “Hand segmentation using skin color and background information,” *2012 Int. Conf. Mach. Learn. Cybern.*, vol. 4, pp. 1487–1492, 2012.

[15] D. S. Deepak Gurung, Cansen Jiang, Jeremie Deray, “Hand Gestures Recognition and Tracking,” *<hal-00903898>*, 2013. [Prieiga per internetą]. Internetinis adresas: https://hal-univ-bourgogne.archives-ouvertes.fr/hal-00903898/document. [Žiūrėta: 15-05-2017].

[16] S. N. Karishma and V. Lathasree, “Fusion of Skin Color Detection and Background Subtraction for Hand Gesture Segmentation,” vol. 3, no. 1, pp. 13–18, 2014.

[17] M. J. P. Viola, “Rapid Object Detection Using A Boosted Cascade of Simple Features,” *Proc. 2001 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, no. 1, pp. 511–518, 2001.

[18] “Convex hull algoritmo aprašas.” [Prieiga per internetą]. Internetinis adresas: https://en.wikipedia.org/wiki/Convex_hull_algorithms. [Žiūrėta: 15-05-2017].

[19] G. Marin, M. Fraccaro, and M. Donadeo, “Palm area detection for reliable hand gesture recognition” Department of Information Engineering, University of Padova, pp. 1–2, 2013.

[20] “Hu momentų aprašymas.” [Prieiga per internetą]. Internetinis adresas: http://stackoverflow.com/questions/11379947/meaning-of-the-seven-hu-invariant-moments-function-from-opencv. [Žiūrėta: 15-05-2017].

[21] Z. Yao, Z. Pan, and S. Xu, “Wrist recognition and the center of the palm estimation based on depth camera,” *Proc. - 2013 Int. Conf. Virtual Real. Vis. ICVRV 2013*, pp. 100–105, 2013.

[22] “Euklido atstumo aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://en.wikipedia.org/wiki/Euclidean_distance. [Žiūrėta: 15-05-2017].

[23] G. Hackenberg, R. McCall, and W. Broll, “Lightweight palm and finger tracking for real-time 3D gesture control,” *Proc. - IEEE Virtual Real.*, no. March 2010, pp. 19–26, 2011.

[24] E. Rosten, R. Porter, and T. Drummond, “Faster and better: A machine learning approach to corner detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, 2010.

[25] G. Borgefors, “Another comment on ‘a note on “distance transformations in digital images,”’” *CVGIP Image Underst.*, vol. 54, no. 2, pp. 301–306, 1991.

[26] “Haar klasifikatoriais aprašymas OpenCV bibliotekos dokumentacijoje.” [Prieiga per internetą]. Internetinis adresas: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html. [Žiūrėta: 16-05-2017].

[27] R. E. Schapire, T. Labs, P. Avenue, a Room, and F. Park, “A Brief Introduction to Boosting Generalization error,” *Ijcai 99*, pp. 1401–1406, 1999.

[28] “OpenFace kodas ir aprašymas.” [Prieiga per internetą]. Internetinis adresas: https://github.com/TadasBaltrusaitis/OpenFace. [Žiūrėta: 16-05-2017].

[29] T. Baltrusaitis, P. Robinson, and L. P. Morency, “OpenFace: An open source facial behavior analysis toolkit,” *2016 IEEE Winter Conf. Appl. Comput. Vision, WACV 2016*, 2016.

[30] T. Baltrusaitis, “Automatic facial expression analysis,” *Pattern Recognit.*, 2014.

[31] “OpenCV bibliotekos aprašymas internete.” [Prieiga per internetą]. Internetinis adresas: http://opencv.org/. [Žiūrėta: 16-05-2017].

[32] “HOG veido detektoriaus aprašymas DLib bibliotekoje.” [Prieiga per internetą]. Internetinis adresas: http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html. [Žiūrėta: 16-05-2017].

[33] I. A. Kalinovskii and V. G. Spitsyn, “Compact convolutional neural network cascade for face

detection," *CEUR Workshop Proc.*, vol. 1576, pp. 375–387, 2016.

[34] "YouTube Face Duombazė." [Prieiga per internetą]. Internetinis adresas: https://www.cs.tau.ac.il/~wolf/ytfaces/. [Žiūrėta: 16-05-2017].

[35] "CameraModelDocumentation.pdf kameros kalibravimo aprašymas. Dokumentas pateikiamas DepthSense SDK instaliacijos direktorijoje."

[36] "Medianos filtras OpenCV bibliotekoje." [Prieiga per internetą]. Internetinis adresas: http://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html. [Žiūrėta: 17-05-2017].

[37] "Medianos filtro aprašymas." [Prieiga per internetą]. Internetinis adresas: https://en.wikipedia.org/wiki/Median_filter. [Žiūrėta: 17-05-2017].

[38] "„Druskos ir pipirų" triukšmo aprašymas." [Prieiga per internetą]. Internetinis adresas: https://en.wikipedia.org/wiki/Salt-and-pepper_noise. [Žiūrėta: 17-05-2017].

[39] S. Krinidis, "Fast 2-D distance transformations," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 2178–2186, 2012.

[40] D. Huttenlocher, "Computer Vision 7. Distance Transforms. Cornell universiteto paskaitų medžiaga."

[41] J. Sklansky, "Finding the convex hull of a simple polygon," *Pattern Recognit. Lett.*, vol. 1, no. 2, pp. 79–83, 1982.

[42] A. M. Andrew, "Another efficient algorithm for convex hulls in two dimensions," *Inf. Process. Lett.*, vol. 9, no. 5, pp. 216–219, 1979.

[43] T. M. Chan, "Optimal output-sensitive convex hull algorithms in two and three dimensions," *Discrete Comput. Geom.*, vol. 16, no. 4, pp. 361–368, 1996.

[44] "*Introduction to Convex hull applications*. Liege universiteto paskaitų medžiaga." [Prieiga per internetą]. Internetinis adresas: http://www.montefiore.ulg.ac.be/~briquet/algo3-chull-20070206.pdf [Žiūrėta: 21-05-2017].

[45] "DepthSenseDS325 techninė specifikacija." [Prieiga per internetą]. Internetinis adresas: http://www.softkinetic.com/Portals/0/Download/WEB_20120907_SK_DS325_Datasheet_V2.1.pdf. [Žiūrėta: 21-05-2017].

[46] "Time-of-flight jutiklių veikimo aprašymas." [Prieiga per internetą]. Internetinis adresas: https://en.wikipedia.org/wiki/Time-of-flight_camera. [Žiūrėta: 21-05-2017].

[47] M. Perenzoni and D. Stoppa, "Figures of merit for indirect Time-of-Flight 3D cameras: Definition, experimental evaluation," *Remote Sens.*, vol. 3, no. 11, pp. 2461–2472, 2011.

[48] "CUDA technologijos naudojimo OpenCV bibliotekoje aprašymas." [Prieiga per internetą]. Internetinis adresas: http://opencv.org/platforms/cuda.html. [Žiūrėta: 21-05-2017].

[49] "CMake aprašymas." [Prieiga per internetą]. Internetinis adresas: https://cmake.org/. [Žiūrėta: 22-05-2017].

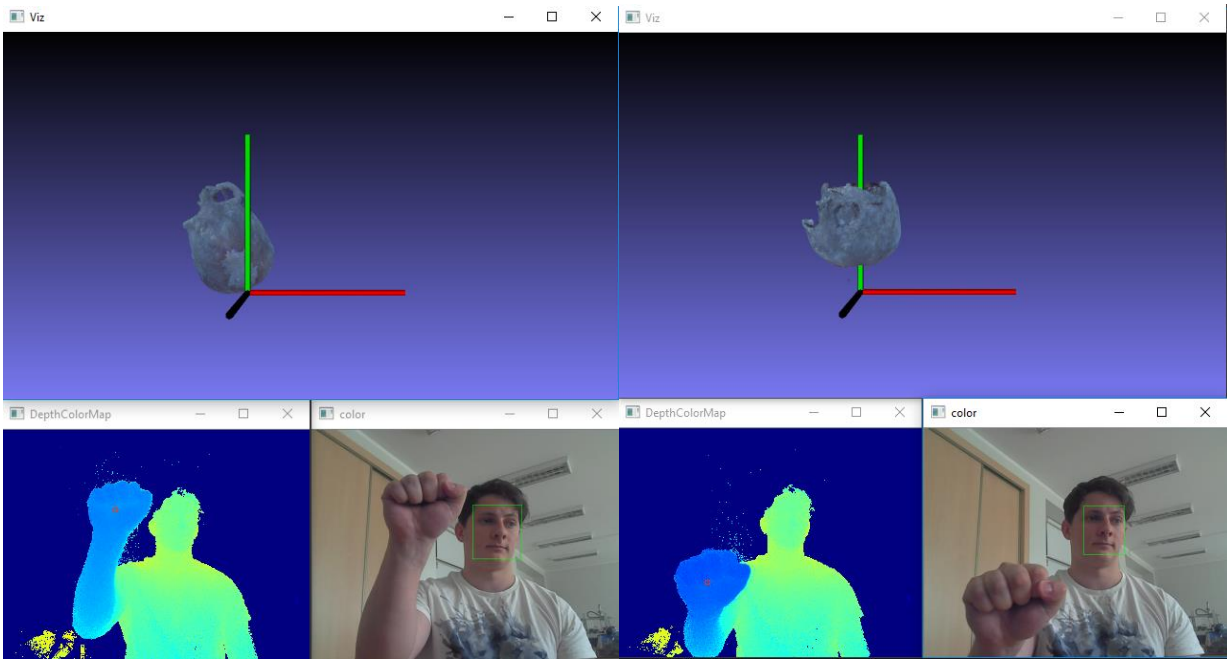
[50] “C++11 funkcijų implementacija Visual Studio 2013.” [Prieiga per internetą]. Internetinis adresas: https://msdn.microsoft.com/en-us/library/hh567368.aspx. [Žiūrėta: 22-05-2017].

[51] “OpenCV Viz modulio aprašymas.” [Prieiga per internetą]. Internetinis adresas: http://docs.opencv.org/2.4/modules/viz/doc/viz3d.html. [Žiūrėta: 26-05-2017].

Priedai

Priedas 1. Demonstracinis projektas 3D objekto valdymui gestais OpenCV Viz aplinkoje

Demonstraciniame projekte pristatoma viena iš sukurtosios rankos gestų atpažinimo sistemos panaudojimo galimybių, tai yra, 3D objekto valdymas gestų pagalba. Projekte valdymas vyksta 2 gestas: kumščiu ir 2 atlenktais pirštais. Pirmuoju gestu galima sukti apie atitinkamas ašis (priedas 1. 1.1. pav.), antruoju – stumti ašimi.



1.1 pav. *Gestais valdomo 3D objekto pasukimas **OpenCV Viz** aplinkoje*

Šiame demonstraciniame projekte fiksuojama plaštakos centro (arba kumščio) x ir y koordinačių vertės. Darant judesį ir rodant tą patį gestą skaičiuojamas x ir y koordinačių skirtumai nuo pirmo iki esamo gesto parodymo ir pagal tai objektas 3D lange yra sukamas arba slenkamas. Demonstracijos įrašas pridėtas DVD diske.