VILNIAUS UNIVERSITETAS
FIZIKOS FAKULTETAS
RADIOFIZIKOS KATEDRA

Albert Cesiul

DIRBTINIŲ NEURONINIŲ TINKLŲ TAIKYMAS VAIZDO ATPAŽINIMUI IR MECHANIZMŲ VALDYMUI ĮTERPTINĖSE LINUX SISTEMOSE

Magistrantūros studijų baigiamasis darbas

(studijų programa – TELEKOMUNIKACIJŲ FIZIKA IR ELEKTRONIKA)

Studentas	Albert Cesiul
Darbo vadovas	doc. Vytautas Jonkus
Recenzentas	doc. Česlovas Pavasaris
Katedros vedėjas	habil. dr. Jūras Banys

Vilnius 2017

Turinys

1. Įvadas 4
2. Kas tai yra dirbtinis intelektas 7
3. Neuroniniai tinklai 9
4. Raspberry Pi mini kompiuteris 12
4.1. Python programavimo aplinka ir globalus interpretatoriaus užraktas 13
4.2. Raspberry Pi kompiuterio pagrindinio procesoriaus spartos didinimas 15
4.3. Naudojamos sąsajos 15
5. OpenCV bibliotekų apžvalga 17
5.1. OpenCV bibliotekų struktūra ir kintamųjų tipai, statinės ir dinaminės struktūros 18
6. OpenCV bibliotekos vaizdo apdorojimo funkcijos 20
6.1. Vaizdo blukinimas 20
6.2. Objekto slenkstinis „Threshold“ atpažinimas 20
6.3. RGB vaizdo konvertavimas į HSV formatą 21
6.4. Krašto atpažinimo filtrai 22
6.5. Kraštų detekcija ir Freemano algoritmas 24
6.6. Tiesinė Hough‘o transformacija 25
6.7. Apskritiminė Hough’o transformacija 26
6.8. Vaizdo spalvų histogramos 28
6.9. Koreliacijos vaizdo atpažinimas 29
6.10. Judėjimo analizavimas ir objektų sekimas 29
6.11. Veido atpažinimas ir Haar‘o bruožai 30
6.12. SURF algoritmas 32
7. Dirbtinio intelekto įterpimas – neuroniniai tinklai ir mašininis mokymasis 33
7.1. Integruotos OpenCV „Machine Learning“ mašininio mokymosi bibliotekos 33
7.2. Neuroninis tinklas vaizdo atpažinimo programos kraštinių sąlygų koregavimui 37
8. Mechaninis tyrimo maketas 38
8.1. Ultragarsinis atstumo matavimo įrenginys HC-SR04 40
8.2. Vaizdo atpažinimo algoritmo su mechanine valdymo dalimi sąsaja 40
8.3. Naktinio matymo sistema 41
9. Rezultatai 43
10. Išvados 56
11. Santrauka 57

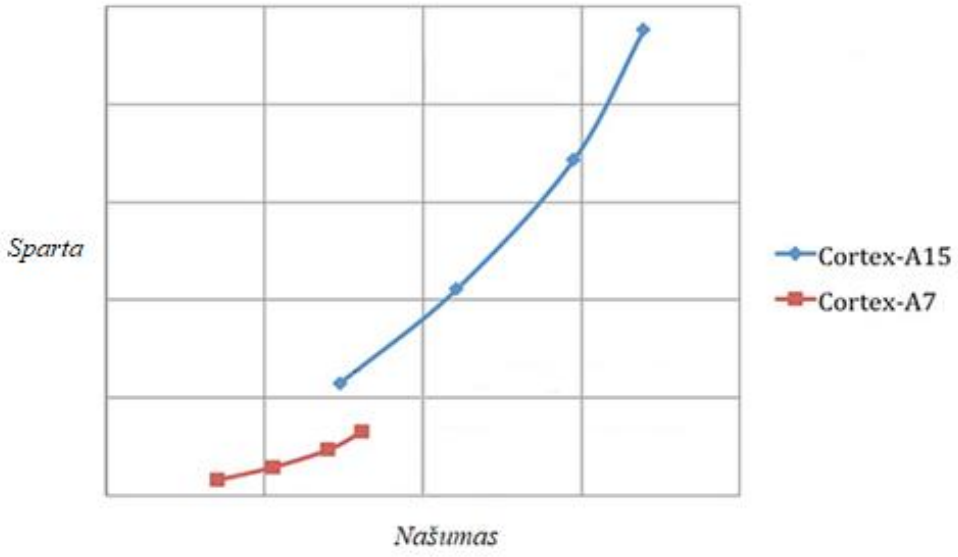
12. Summary 58
13. Informacijos šaltiniai.....59

1. Įvadas

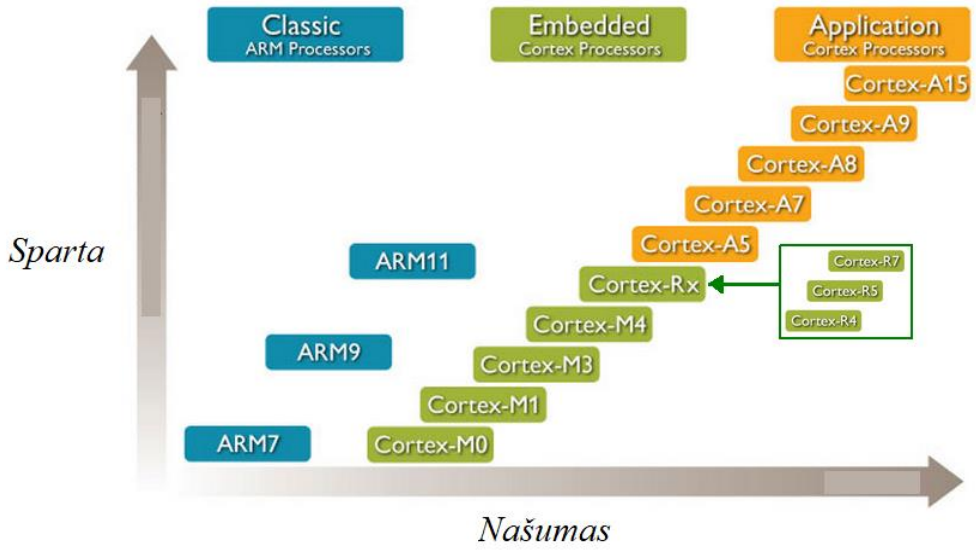
Naujų technologijų kūrime gana didelę dalį užima vaizdo atpažinimo, apdorojimo technologijos. Apie 90% visos gaunamos informacijos iš išorės žmogus gauna regos pagalba, todėl akivaizdu, kad vaizdo atpažinimo integravimas elektronikoje bus labai plačiai vystomas. Naudojimo sritis yra gana didelė: pradedant saugumo įranga, baigiant autonominiais automobiliais, kuriais pradeda domėtis, kuo toliau, tuo daugiau, automobilių gamybos kompanijų. Vis dėl to, užduotis nėra pati paprasčiausia. Vaizdo apdorojimas ir atpažinimas reikalauja gana didelės skaičiavimo spartos procesorių bei didelės darbinės atminties pralaidos, nes vaizdo nesuspausta informacija yra saugoma kai atskiri kadrai – matricos.

Vis dėl to, dirbant su vaizdo atpažinimu ne visada yra geriausia naudoti galingiausią įrangą, nes nei ištekliai, nei galimybės to neleidžia. Pavyzdžiui, konstruojant mobilų įrenginį reikia taip pat atkreipti dėmesį į kompiuterio svorį ir gabaritus, be to, labai svarbu taip pat atkreipti dėmesį į energijos suvartojimą. Kaip pagalbinės programavimo bibliotekos tokiam darbui yra tinkamos OpenCV atviro kodo vaizdo apdorojimo bibliotekos, kurios yra parašytos daugumai populiarių operacinių sistemų: Microsoft Windows, Android, Linux, Apple Macintosh ir iOS. Šitos bibliotekos yra pradėtos kurti Intel kompanijos ir pirmą kartą buvo išleistos 2000 metais. Iki dabar jos išlieka pirmaujančios pasaulyje savo srityje ir yra toliau atnaujinamos, bei tobulinamos. [1] [2]

Kaip valdymo kompiuteris buvo pasirinktas Raspberry Pi 2 mini kompiuteris, kuris turi ant vienos plokštės visus reikalingus komponentus pilnaverčiam veikimui, turi gana gerą suvartojamos energijos ir spartos santykį ir turi daug programiškai valdomų išvadų ir sąsajų, kuo labai primena mikrovaldiklius, o taip pat naudoja Cortex-A serijos procesorių, kurie plačiai yra paplitę išmaniuose telefonuose bei daugumoje kitų mobilių įrenginių. Raspberry Pi 2 naudoja keturių branduolių Cortex-A7 serijos procesorių su grafinę posisteme Broadcom VideoCore IV. Kaip matome 1 paveikslėlyje, nors A7 nėra labai spartus dėl ne naujausios technologijos, bet jo energijos suvartojimo ir spartos santykis yra pakankamai geras, o taip pat bet kokia Cortex-A serija yra žymiai spartesnė už mikrovaldiklius, kurie yra žymimi kaip Cortex-M serija, ką galime pamatyti antrame paveikslėlyje. Be to, Raspberry kompiuterio operacinė sistema yra sukurta Linux atviro kodo operacine sistema, kuri yra plačiai paplitusi visame pasaulyje dėl jos paprastumo, patikimumo ir universalumo, o taip pat yra mėgstama programavimo entuziastų. Nieko keista, kad nuo gamybos pradžios 2012 metais užkariavo gana didelę mini kompiuterių rinkos dalį, kaip naujų programavimo ir elektronikos inovacijų kūrimo pagalbininkas. [3]



1 pav. Cortex-A šeimos procesorių spartos ir energijos suvartojimo palyginimas [4]



2 pav. Cortex procesorių šeimų spartos palyginimas [5]

Kuo toliau, tuo labiau šiuolaikinės inovacijų technologijos linksta prie dirbtinio intelekto vystymo. Bet, kas tai yra dirbtinis intelektas, kuo jis skiriasi nuo „paprasto“ intelekto ir kur jį galima panaudoti? Ar bus iš to nauda? Daugelis medicininės įrangos gamintojų, saugumo ir žvalgybos tarnybos, bei komercinės gamyklos yra labai susidomėjusios šios srities vystymu.

Dirbtinis intelektas, trumpai apibrėžus, yra mašinos bruožas priimti sprendimus remiantis gautais duomenimis ir galimybe „mokytis“ ateities problemų sprendimui, tai yra ne tik vykdyti iš anksto programuotojo parašytas komandas. Vis dėl to tie sprendimų ir mokymosi algoritmai yra jau

apibrėžti žmogaus ir patys negali evoliucionuoti. Jau gana ilgą laiko tarpą vyksta debatai tarp mokslininkų, ar mašinos galės ateityje, tikra to žodžio prasme, galvoti už save. [6]

Mano darbo tikslas yra parašyti programą vaizdo apdorojimo ir atpažinimo mini kompiuteriui Raspberry Pi 2, naudojantis OpenCV atviro kodo bibliotekomis, kuri galės atpažinti objektus, atpažinti kliūtį, sekti norimo objekto padėtį ir kontroliuoti kitų įrenginių darbą, vadovaujantis vaizdo atpažinimo rezultatais. Apibendrinus, tikslas yra sukonstruoti ir užprogramuoti savarankišką įrenginį, kuris galės judėti be jokios žmogaus įtakos ir atlikti užprogramuotus iš anksto veiksmus. Pvz.: Nuvažiuoti į iš anksto nustatytą padėtį, išvengiant atsitiktinai atsiradusių kliūčių arba ieškoti ir sekti norimą objektą. Taigi, vaizdo atpažinimo testavimui reikia sukonstruoti mechaninį važiuoklės maketą bei parašyti atpažinimo algoritmą testavimui. Programos universalumui ir adaptyvumui padidinti skirtingose situacijose galima panaudoti dirbtinius neuroninius tinklus ir išbandyti skirtingus vaizdo atpažinimo modelius.

2. Kas tai yra dirbtinis intelektas

Teoriškai dirbtinis intelektas yra apibrėžiamas tuo: kai žmogus bendrauja su kompiuteriu to nežinodamas, žmogus negalėtų atskirti ar šnekasi su tikru žmogumi, ar su mašina. Tai yra vienas iš pagrindinių Turingo testo bruožų. Testas testuoja logiką, minčių išdėstymą, atmintį, sprendimų priėmimą, trūkstamos informacijos nuspėjimą ir visus kitus žmogaus protinius sugebėjimus. Kol kas tokio sudėtingo dirbtinio intelekto niekas dar nesukūrė, nors jau buvo keli labai priartėję prie Turingo testo išlaikymo projektai. Todėl dažniausiai yra apsiribojama supaprastintu modeliu ir jo pritaikyme kažkokio veiksmo vykdyme. [6]

3 pav. Alan Turing fotografija [7]

Ieškant sprendimo anksčiau nurodytai problemai yra atliekama be galo daug tyrimų, kad sužinoti ir suprasti, kaip pačio žmogaus protas veikia, bet tikslaus atsakymo dar niekas nerado. Taigi, visi bandymai atkurti smegenų veiklą yra aproksimuoti ir supaprastinti. Kaip pavyzdys medicinoje medikai tiriant smegenų aktyvumą gali pasakyti ar smegenis yra sveikos, ar gerai veikia, kurios sritys yra už ką atsakingos, kokios cheminės reakcijos vyksta ir kokie elektriniai signalai juda neuronais, bet jeigu visą tai reikėtų apjungti į vieną pilną veikiantį modelį, tai dar kol kas būtų neįmanoma. Tai galima labai lengvai paaiškinti tuo, kad nors dabar smegenų veiklos supratimas yra gana didelis, bet dar nėra pilnas. Be to, yra gana didelė tikimybė, kad žmonijos progresas dar nepasiekė tos ribos, kai technologiškai yra įmanoma pilnai atkartoti smegenis. Kol kas yra sukurtos gana primityvios sistemos, kurios atkartoja tikrų smegenų veiklą. 2012 metais mokslininkams su vadovu Chris Eliasmith iš University of Waterlook, kuris yra Kanadoje, Ontario mieste, pavyko sukurti dirbtines smegenis iš 2,3 milijono neuronų. Palyginus su vidutinio žmogaus smegenimis – virš 100 milijardų,

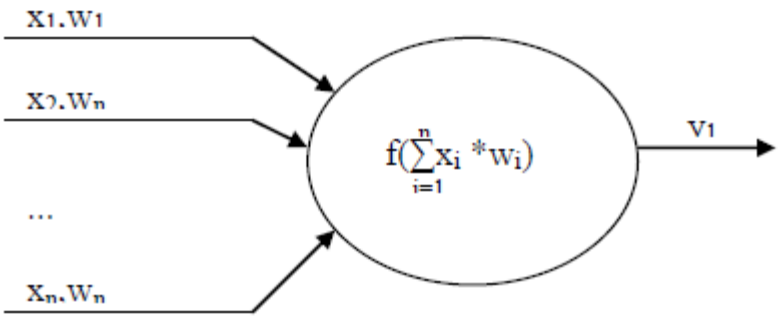
tai dar nėra gana geras pasiekimas ir be to, dirbtinio modelio neuronų sujungimai tarpusavyje taip pat yra supaprastinti ir ne tokie sudėtingi. Visgi, einama gera kryptimi. [8]

Kiti mokslininkai neaugina dirbtinių neuronų, o pasinaudoja silicio technologijomis ir bando atkartoti protinę veiklą, modeliuojant ją kompiuterinėse sistemose. Bet tada atsiranda kita problema – spartos trūkumas. Jeigu vienos architektūros procesorius bando atkartoti visiškai kitokios architektūros procesoriaus darbą: savo galimų komandų vykdymu bando atkartoti kito procesoriaus gaunamus rezultatus, turint tuos pačius duomenis, tai sklando „gandas“, kad jeigu norime išgauti panašią spartą, kaip atkartojamojo procesoriaus, reikia bent 10 kartų didesnės spartos įrenginio. Bet tai dar priklauso nuo atkartojimo programos (emuliatoriaus) optimizavimo. Mūsų atveju, mes iš viso bandome atkartoti sistemą, kuri fundamentaliai yra visiškai skirtinga nuo žmogaus sukurtų procesorių veikimo. Be to, biologai teigia, kad žmogaus smegenis gali vykdyti 10000000 miliardų slankaus kablelio operacijų per sekundę (10 PFLOPS), ir nors galingiausias šiuo metu superkompiuteris pavadinimu Sunway TaihuLight Kinijoje, Wuxi mieste 2016 metais pasiekė (93 PFLOPS), bet jis yra tik vienas toks visame pasaulyje ir kaip anksčiau minėta, negalime pilnai aprašyti to, ko dar pilnai nesuprantame. Be to, smegenis gali adaptuotis, mokytis bei vykdyti daug skirtingų veiksmų lygiagrečiai. Standartiniai superkompiuteriai ir paprastam vartotojui prieinami kompiuteriai gali nebent sumodeliuoti žiurkės smegenų dydžio modelį. Bet, net ir jo užtenka kartais ir labai sudėtingiems veiksmams atlikti.

Mokslininkai sukūrė labai daug skirtingų dirbtinio intelekto modelių: neuroninių tinklų modeliavimas, chaoso inžinerija, mašininio mokymosi modelis, „fuzzy“ logika ir t.t. Taip pat yra sukurta daug dirbtinio intelekto bibliotekų, kurios gali būti pritaikytos daugumai populiariausių programavimo kalbų. Galima sakyti, kad populiariausias yra neuroninių tinklų modelis, kurį aptarsime kituose skyriuose.

3. Neuroniniai tinklai

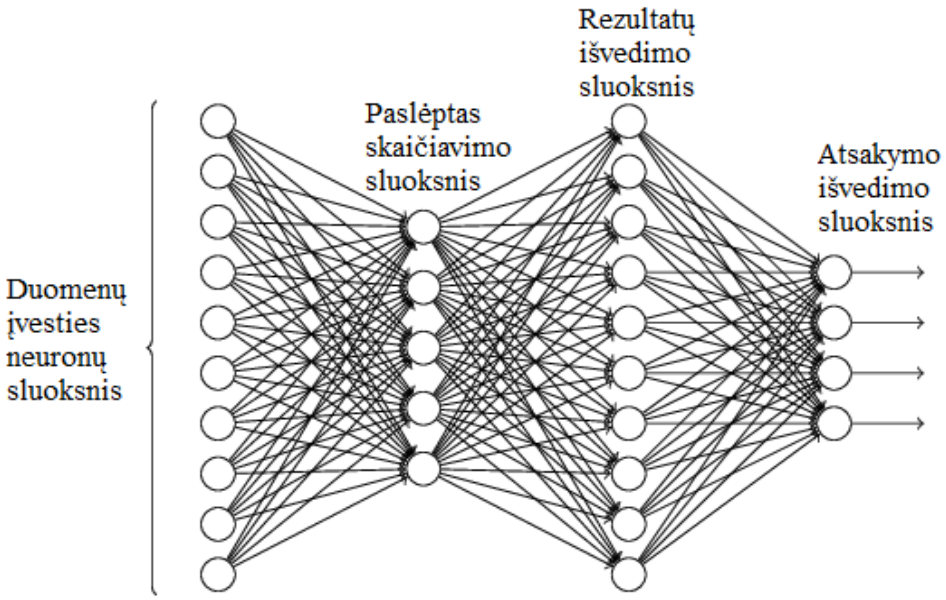
Neuroniniai tinklai atkartoja tikrų neuronų elgesį. Virtualūs neuronai, angliškai vadinami „neurodes“. Dirbtinį neuroną galima laikyti įrenginiu, kuris turi keletą įvesčių ir vieną išvestį, yra jungiami tarpusavyje kaip ir tikri neuronai, o informacija yra saugoma sujungimų tarpusavyje kombinacijose ir kaip svoriniai koeficientai. Skaičiavimai vyksta keičiant šias vertes. Dirbtinis neuronas turi daug įėjimų su svoriniais koeficientais ir vieną išėjimą su savo ribine verte, t.y. išėjimo vertė pasikeičia, priklausomai ar ribinė vertė buvo viršyta, ar ne. Kai vienas virtualus neuronas pakeičia savo srityje esančio sujungimo vertę, kitas šalia esantis virtualus neuronas sureaguoja į tai, arba net keli, jeigu tas sujungimas yra tarp daugiau nei dviejų neuronų. Neuroniniai tinklai pasižymi keliomis svarbiomis charakteristikomis: adaptyvumu ir saviorganizacija. Panaudodami prisitaikantį apsimokymą ir saviorganizaciją, tinklai pasižymi galingomis ir efektyviomis apdorojimo galimybėmis. Netiesiniu tinklo duomenų apdorojimu jie padidina tinklo apibendrinimo laipsnį, klasifikacijos ir triukšmo pašalinimo galimybes. Neuroniniai tinklai pasižymi lygiagrečiais skaičiavimais – juos sudaro daugybė skaičiavimus atliekančių tarpusavyje susietų ląstelių. Geriausias ir sudėtingiausias variantas yra toks, kai neuroninis tinklas sugeba pats projektuoti sau naujus virtualius neuronus ir jų sujungimus, tarsi pati programa modifikuoja ir perrašo savo algoritmą - adaptuojasi. [9]



4 pav. Dirbtinio neurono struktūra [9]

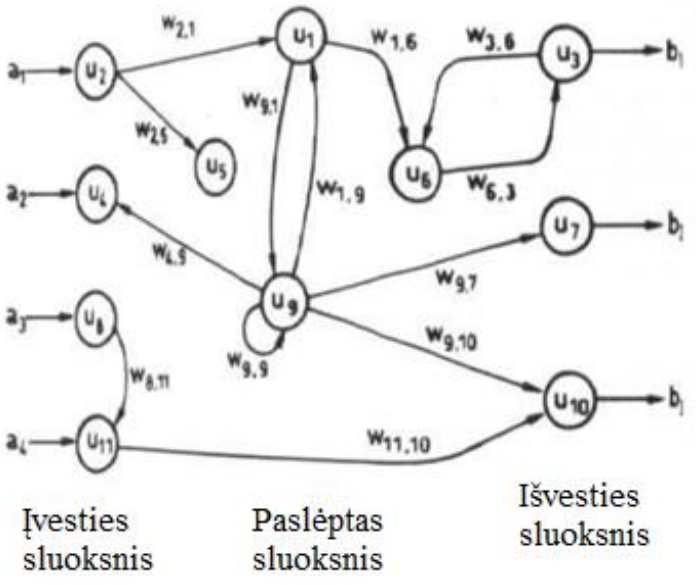
Virtualus neuronas veikia keliais režimais: apmokymo ir skaičiavimo. Apmokymo režime neuronas sukuria ir koreguoja su įvestimis susietų svorių reikšmes taip, kad būtų gauta pageidaujama išėjimo reikšmė. Naudojimo režime neuronas grąžina su įėjimuose esančiomis reikšmėmis susijusią reikšmę. Su kiekvienu neuronu yra susieta perdavimo funkcija, kuri nurodo kaip to neurono reikšmė yra perduodama kitam neuronui. Įprastai perdavimo funkcija padaugina kiekvieną svorį iš atitinkamai susijusių įėjimo verčių ir šias sandaugas susumavus yra gaunama suminė įėjimo vertė. Tada gautai suminei įėjimo vertei yra pritaikoma funkcija – tai, ką neuronas siunčia kaip rezultatą kitam neuronui.

Gana didelis neuroninių tinklų privalumas yra atsparumas jo pažeidimams arba klaidų tolerancija – sugedus daliai tinklo nukentės našumas bei tikslumas, tačiau tam tikros tinklo funkcijos vis tiek gali būti vykdomos patikimai. Taip yra jeigu neuroninis tinklas veikia ant daugelio serverių, tai vieno sugedimas nedarys didelės įtakos visos sistemos darbui. Kitos neuroninio tinklo dalys prisitaikys situacijai ir pasidalys naujai atsiradusia apkrova, bet skaičiavimo našumas bei tikslumas mažės proporcionaliai virtualių neuronų sumažėjimui. Kitas taip pat privalumas yra galimybė naudoti ir veikti su iškraipytais arba dalinai sugadintais duomenimis. Pavyzdžiui, dalinai sugadintos fotografijos atkūrimui.



5 pav. Neuroninio tinklo principinis atvaizdavimas [10]

Atgalinio grįžtamojo ryšio neuroniniuose tinkluose signalai gali sklisti visomis kryptimis: į priekį, atgal ir tarp to paties sluoksnio neutronų. Dėl to atgalinį ryšį turintys neuroniniai tinklai yra žymiai sudėtingesni už tiesioginio sklidimo neuroninius tinklus, tačiau taip pat ir daug spartesni. Tokio tipo tinklai yra dinaminiai - jų būsena nuolat kinta. Rezultato skaičiavimas yra vykdomas iki to laiko, kol yra pasiekiamas pusiausvyros taškas ir tinklas jame lieka iki to, kai pasikeičia įvedimo duomenys, tada skaičiavimas prasideda iš naujo – tai yra: pasikeitus duomenims yra ieškomas naujas pusiausvyros taškas.



6 pav. Atgalinio grįžtamojo ryšio neuroninis tinklas [9]

Galima būtų išskirti vieno sluoksnio ir daugiasluoksnius neuroninių tinklų modelius. Vieno sluoksnio architektūros atveju visi neuronai yra tarpusavyje susiję, kai tuo tarpu daugiasluoksnės architektūros atveju neuronai yra hierarchiškai suskirstyti į sluoksnius, tarsi atskiros mąstymo sistemos, kur kiekvieno sluoksnio neuronų įėjimai yra susiję tik su prieš tai buvusio sluoksnio neuronais.

Dažniausiai sutinkami neuroniniai tinklai yra sudaryti iš trijų sluoksnių grupių: įvesties sluoksnio (angliškai „input“), paslėptų neuronų sluoksnių (angliškai „hidden“), išvesties sluoksnio (angliškai „output “). Įvesties sluoksnio neuronai yra atsakingi už duomenų įvedimą į tinklą. Kartais jie taip pat vykdo tam tikrą pradinį įvesties duomenų apdorojimą, triukšmo filtravimą arba dekodavimą. Paslėptųjų sluoksnių neuronai, remdamiesi įvedimo reikšmėmis ir apmokymo metu nustatytais svoriniais koeficientais, vykdo su uždavinio sprendimu susijusius skaičiavimus. Išvesties sluoksnio neuronų skaičiavimo rezultatas priklauso nuo paslėptųjų sluoksnių neuronų išvedimo reikšmių ir su jais susijusių svorinių koeficientų.

Dabar truputį plačiau apie neuroninio tinklo apmokymą. Neuroninių tinklų apmokymo metodai gali būti suskirstyti į tris didesnes kategorijas: mokymą su mokytoju, bandymų ir klaidų, bei savarankišką mokymąsi. Mokymo su mokytoju metu yra panaudojamas išorinis mokytojas, kuris nurodo, koks turėtų būti reikiamas atsakymas. Įprastai tinklui apsimokyti yra pateikiami įvedimo ir išvedimo duomenų rinkiniai. Tinklo svoriai koreguojami taip, kad tinklas ateityje pateiktų reikiamą atsakymą. Bandymų ir klaidų metodas yra panašus į praeitą būdą, bet tik nurodo, ar buvo apskaičiuota teisingai ar ne. Paskutinis iš nagrinėjamų mokymosi metodų tinklas negauna informacijos nei apie tai, kokį tipą jis turėjo priskirti, nei atsakymo ar jo atliktas priskyrimas buvo geras ar blogas. Vietoj

to mokymosi proceso metu jis remiasi vidiniais kriterijais. Pats rikiuoja panašius duomenis, vienus šalia kitų ir taip kaupia „žinias“. Pavyzdžiui: panašius paveikslus kaupia vienoje atminties erdvės dalyje, o kitus kitoje, ir kai gauna naują paveikslėlį, tai gali iš karto nurodyti, kuriai grupei jis priklauso.

Vienas iš svorinių koeficientų radimo būdų yra naudoti klaidos sklidimo atgal algoritmą. Klaidos sklidimo atgal algoritmas pasinaudodamas gradientinio nusileidimo metodu, ieško klaidos funkcijos minimumo svorių erdvėje. Svorių kombinacija, kuriai esant klaidos funkcija yra minimali, laikoma uždavinio sprendiniu. Kadangi naudojant šį metodą, reikia skaičiuoti klaidos funkcijos gradientą, tai klaidos funkcija privalo būti be trūkio taškų ir diferencijuojama.

Pirmiausia, atsitiktinai parenkame pradinius svorius, tada paduodame tinklui pradinius duomenis ir gauname atsakymą. Toliau nustatome išėjimo sluoksnio neuronų skaičiavimo paklaidas, po to sudauginame paklaidas su įvesties duomenimis ir sudedame jas su svoriniais koeficientais. Suskaičiuojame paslėpto sluoksnio neuronų paklaidas, pasiremdami toliau esančio sluoksnio neuronų rezultatais, kuriuos suskaičiavome ankstesniame žingsnyje. Jei dar neviršijome mokymo ciklų limito, tada vėl nustatome išėjimo sluoksnio neuronų paklaidas ir perskaičiuojame svorius ir taip iki tol, kol svoriniai koeficientai nepriartės prie mums reikalingų verčių – tai yra, kai dirbtinis neuroninis tinklas skaičiuos teisingai.

Viena iš problemų, kurios gali iškilti neuroninio tinklo apmokymo metu, yra ta, jog paklaidos funkcijoje gali pasitaikyti lokalių minimumų. Tokiu atveju, pradėjus apmokyti neuroninį tinklą su svoriais, prie kurių klaidos funkcija yra netoli tokio regiono algoritmas gali konverguoti į lokalų minimumą, kai tuo tarpu norint surasti globalų minimumą gali tekti svorius pakoreguoti taip, jog klaida šiek tiek padidės, kol bus įveiktas lokalus maksimumas.

4. Raspberry Pi mini kompiuteris

Rinkoje yra daug skirtingų mini kompiuterių, nes jie yra populiarūs ir pageidaujami visokiausių projektų kūrime, todėl dauguma didelių korporacijų bando atsipjauti ir sau dalį šito pyrago, gamindami savo šito norimo gaminio variantus. Viena iš pirmaujančių kompanijų yra Raspberry Corporation Jungtinėje Karalystėje, kuri yra gana jauna kompanija, bet susilaukė didelės sėkmės pristačius Raspberry Pi mini kompiuterį, kurio pradinė užduotis buvo skatinti programavimo mokymą mokyklose. Laikui bėgant Raspberry Pi labai išpopuliarėjo ir pradėjo juo domėtis radiomegėjai, ir net profesionalai.

Labai didelis Raspberry kompiuterio privalumas yra tas, kad yra sukurtas kūrėjo kompanijos serveris, kur yra talpinamos sukurtos jam programos, valdikliai bei operacinės sistemos atnaujinimai. Be to, pagrindinė mini kompiuterio operacinė sistema Raspbian yra parašyta Linux sistemos pagrindu

ir taip pat yra atviro kodo, taigi, laikui bėgant atsirado daug kitų oficialių ir mėgėjiškų operacinės sistemos variantų.

Kaip vaizdo atpažinimo platforma buvo pasirinktas Raspberry Pi 2 B modelio mini kompiuteris. Šitas modelis turi 4 USB sąsajos išvadus, HDMI, LAN, Audio-jack, bei specialias jungtis Raspberry kamerai ir ekranui. Gana didelis privalumas yra programuojama 40 kontaktų juosta su GPIO kojelių funkcijomis, SPI, UART ir I2C sąsajomis, PWM signalo išvadais ir 3.3V, bei 5V maitinimu, o taip pat virtualios žemės išvadais. Pagrindinio procesoriaus (kodinis pavadinimas BCM2836) taktinis dažnis yra 900 MHz, jis yra keturių branduolių, 32 bitų Cortex-A7 šeimos mikroprocesorius, palaikantis komandų rinkinį ARMv7. Be to Raspberry Pi 2 turi 1 GB darbinės atminties, o kaip pagrindinę atmintį duomenų saugojimui, naudoja SD kortelę (palaiko iki 32 GB). [11]

7 pav. Raspberry Pi 2 mini kompiuteris [12]

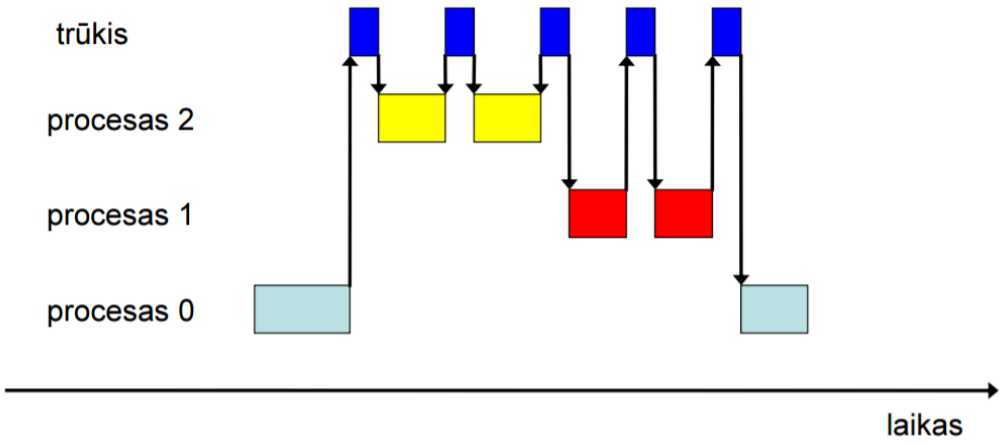
4.1. Python programavimo aplinka ir globalus interpretatoriaus užraktas

Raspberry Pi 2 operacinė sistema yra paremta Linux operacinės sistemos pagrindu ir turi integruotą Python programavimo kalbos supaprastintą aplinką. Vaizdo apdorojimo ir atpažinimo OpenCV bibliotekos taip pat yra pritaikytos Python kalbai, taigi buvo logiška pasirinkti šį variantą. Be to, PCA9685 praplėtimui taip pat buvo gamintojo parašytos bibliotekos, kas gana stipriai palengvino visų komponentų sujungimą.

Vis dėl to, darbo eigoje buvo pastebėta, kad Raspberry Pi mini kompiuterio Raspbian operacinėje sistemoje, Python kalboje pagal nutylėjimą yra išjungtas aparatinis interpretatorius, taigi, nors ir turime keturis branduolius, vienas programos procesas, nors gali turėti daug gijų, bus vykdomas ant vieno branduolio – nebus optimizuojamas. Aparatinis interpretatorius reiškia tai, kad ten, kur įmanoma, automatiškai bus dalinama veiksmus į kelis lygiagrečius procesus.

Operacinė sistema veikia gijų ir trūkių pagrindu. Realiai vienu laiko momentu viename procesoriuje yra vykdomas tik 1 procesas - programa, kadangi procesai keičiami procesoriuje labai

greitai (kas 0,1-0,01 ms) – sudaromas lygiagretumo įspūdis – pseudolygiagretumas. Procesai gali turėti skirtingus prioritetus ir būti vykdomi skirtingus laiko intervalus. Operacinės sistemos veikimo principas pavaizduotas 8 paveikslėlyje. Tikrasis lygiagretumas skaitomas tada, kai yra bent 2 veikiantys procesoriai tuo pačiu momentu ir kurie dalijasi ta pačia atmintimi. Gijos (angliškai „threads“)– tai atskirai valdomi elementai, kurie dirba su procesoriumi. Gijos dar vadinamos lengvaisiais procesais – kiekvienas procesas gali turėti kelias gijas. Terminas daugiagijis nusako, kad viename procese veikia kelios gijos pseudolygiagrečiai, kurios naudojasi bendra adresų erdve, atvirais failais ir kitais bendrais resursais. Gijos yra labiau susietos tarpusavyje nei procesai, nes vienu metu geba dalintis tais pačiais resursais ir adresų erdve. Kaip matome 8 paveikslėlyje, operacinė sistema veikia trūkių principu ir paeiliui perjunginėja procesus – nustato, kuris šiuo metu turi veikti, kontroliuoja bendrų resursų panaudojimą, procesų prioritetus ir kaip dažnai jie yra paleidžiami. [13] [14]



8 pav. Operacinės sistemos veikimo principas [13]

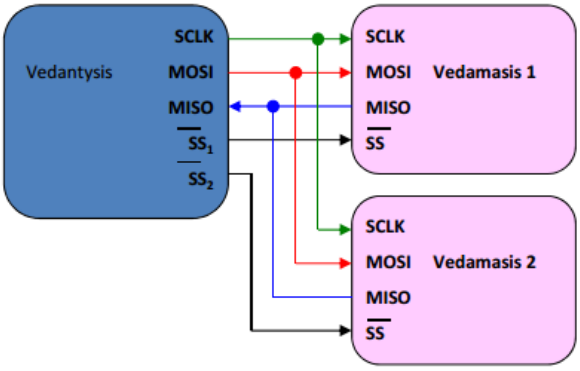
Globalus Interpretatoriaus Užraktas (angliškai „Global Interpreter Lock“) yra resursų blokavimo sistema, kuri neleidžia paleisti kelių lygiagrečių gijų tos pačios programos ant skirtingų procesoriaus branduolių. Taip yra padaryta dėl to, kad python programavimo kalbos programų atminties valdymas nėra saugus lygiagrečiu gijų atžvilgiu (užtikrinama prieiga prie bendros atminties vienai gijai vienu laiko momentu) ir apsaugo nuo nesuderinamumo su C/C++ kalbų bibliotekomis, kurios nepalaiko daugiagijinių skaičiavimų. Kad apeiti GIL, reikia naudoti ne kelias gijas, o vietoj jų reikia kurti nepriklausomus procesus, kurie gali tarpusavyje mainytis duomenimis per specialiai sukurtus atmintyje perstumiamus duomenų masyvus, kuriuos vienas priimantis procesas nuskaito, o iš kito galo kitas procesas siunčia duomenis. Todėl Python kalboje ši funkcija ir buvo pavadinta „Queue“ – eilė. [15]

4.2. Raspberry Pi kompiuterio pagrindinio procesoriaus spartos didinimas

Kai atsiranda skaičiavimo spartos trūkumas ir nėra galimybės pakeisti procesoriaus į galingesnį, lieka tik viena išeitis: procesoriaus taktinio dažnio didinimas virš leistinos gamintojo nurodytos vertės (angliškai „overclocking“). Didinant dažnį reikia būti labai atsargiems, nes per daug padidinus dažnį procesorius pradeda veikti nestabiliai, o be to išskiriamas šilumos kiekis taip pat didėja ir gali pažeisti procesorių. Be to, didinant dažnį, reikia didinti taip pat ir branduolių veikimo įtampą, kad signalų frontai spėtų pasiekti reikalingas ribines vertes ir procesorius veiktų stabiliai. Jeigu loginių signalų frontai nespėja pasiekti reikiamų verčių, procesorius gali pradėti veikti, kaip anksčiau paminėta - nestabiliai. Vis dėl to, padidinus maitinimo įtampą, frontai gali pradėti kilti greičiau ir veikimo stabilumas nenukentės. Pats paprasčiausias pavyzdys gali būti Arduino mikrovaldikliai su Atmega328 mikroprocesoriais, kurie prie 3,3 V veikia stabiliai su 8 MHz taktiniu dažniu, bet jeigu norime stabilaus veikimo su 16 MHz dažniu, tada maitinimas turi būti 5 V. Aišku, kad įtampą didinti reikia atsakingai, nes gali įvykti puslaidininkio pramušimas ir prietaisas bus sugadintas. Maitinimo įtampos didinimas taip pat lemia išskiriamos šilumos padidėjimą, kas taip pat gali nulemti procesoriaus sugadinimą. Todėl, jeigu didiname procesoriaus taktinį dažnį, geriausia yra panaudoti daug efektyvesnį aušinimą. Šiuo atveju buvo panaudotas radiatorius iš nešiojamo kompiuterio vietoje tiesioginio aušinimo oru, nes standartiškai Raspberry Pi 2 mini kompiuterio procesorius neturi jokio papildomo šilumos nuvedimo. Skirtumas buvo akivaizdus, kai pilnai apkrovus procesorių, darbinė temperatūra nukrito nuo 87 ºC su gamintojo nustatytu taktiniu dažniu, iki 58 ºC su apie 30 % pakeltu taktiniu dažniu. Akivaizdu, kad taip pat reikia atsižvelgti ir į tai, kad taktinio dažnio ir įtampos kėlimas taip pat nulemia procesoriaus suvartojamos energijos kiekio augimą.

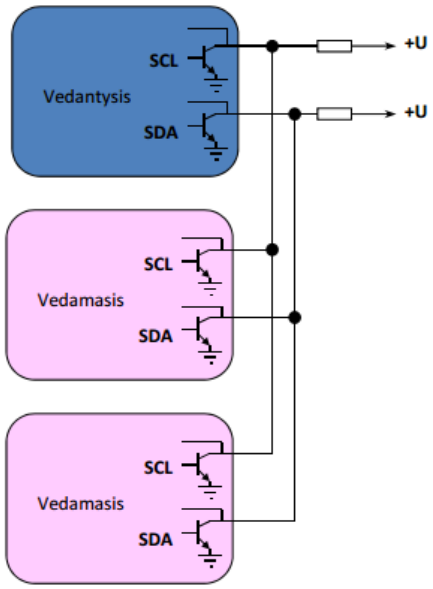
4.3. Naudojamos sąsajos

SPI sąsaja yra naudojama liečiamam ekranui valdyti, kuris labai palengvina pačio įrenginio darbą. SSP tai nuosekli sinchroninė sąsaja, o tai reiškia, kad duomenys yra paduodami dvejomis linijomis: MOSI (duomenų išvesties linija) ir MISO (duomenų įvesties linija), o jų sinchronizacijos impulsai kita linija: SCLK. Kai norima nusiųsti nauja duomenų srautą, iš pradžių reikia pakeisti iš aukšto lygio į žemą SS koją (kiekvienam įrenginiui yra priskirta skirtinga valdymo kojelė), nes tik tada vedamasis įrenginys pradeda laukti duomenų signalo iš SSP sąsajos. Dažniausiai iš pradžių eina komandos bitai, o po to yra siunčiama informacija. [16]



9 pav. SPI sąsajos sujungimo principas [16]

I^2C ir UART sąsajos yra naudojamos valdyti išorinius įrenginius. I^2C sąsaja turi du valdomus išvadus: duomenų siuntimo ir sinchronizacijos (SDA ir SCL). I^2C sąsajoje visada turi būti valdantysis įrenginys ir valdomieji įrenginiai su priskirtais adresais. Valdantysis įrenginys inicializuoja duomenų perdavimą ir gali siųsti, ir priimti duomenis iš vedamų įrenginių. Sąsajos privalumas yra tas, kad priklausomai nuo adresavimo tipo galima prijungti 127, 255 arba 1027 įrenginius prie I^2C sąsajos. Yra naudojamas įrenginių aparatinis adresavimas: 7, 8 arba 10 bitų ilgio adresas yra priskiriamas atitinkamai sulituojant vedamojo įrenginio lusto adreso kojeles.



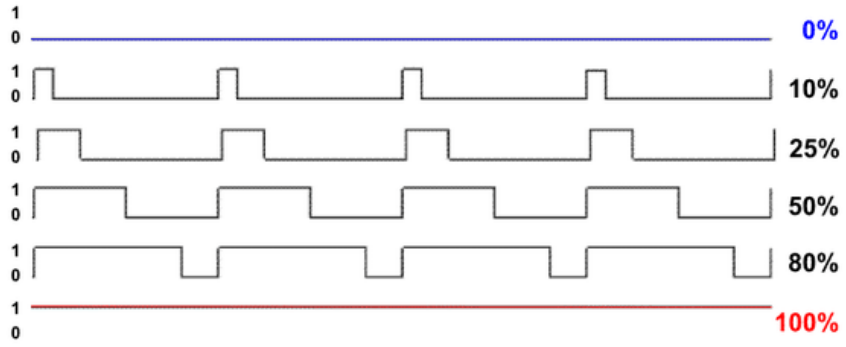
10 pav. I^2C sąsajos sujungimo principas [16]

UART (universal asynchronous receiver/transmitter) sąsaja yra elektroninis įrenginys, kuris perduoda informacijos bitus po vieną (nuosekliai). Vienas iš svarbiausių UART sąsajos požymių yra tai, kad joje nėra sinchronizacijos impulsų, tik dvi nuoseklios magistralės (siuntimo ir priėmimo).

Todėl, kad sąsajos veikimas būtų tvarkingas, du prietaisai, kurie yra sujungti tarpusavyje turi bendrauti vienoda sparta.

Sąsajos veikimo principas yra ganėtinai paprastas. Siuntimo įrenginys iš pradžių siunčia vieną bitą kaip įspėjamąjį, kad pradėta siuntimo operaciją. Tada siunčia aštuonis duomenų bitus ir du baigiamuosius, kurie nurodo, kad siuntimas baigėsi. Nebūtinai reikia siųsti aštuonis bitus, galima ir penkis, priklausomai nuo sąsajos tipo. Imtuvas dažniausiai veikia aštuonis kartus greičiau už duomenų perdavimą. Per kiekvieną imtuvo taimerio sutiksėjimą yra nuskaitoma įėjimo loginė būsena. Jeigu užfiksuojama, kad signalo bitas trunka daugiau nei pusę numatyto ilgio, reiškia galima nuskaityti duomenis. Jeigu trunka trumpiau, UART sąsaja ignoruoja šį signalą kaip triukšmą. Kaip žinoma, pagaminti du tobulai vienodus osciliatorius neįmanoma, todėl įrenginiai bando sinchronizuotis pagal pirmo bito frontą. Tai žymiai pagerina sąsajos veikimą ir neleidžia įrenginiams išsinchronizuotis. Siuntimo įrenginys turi indentifikavimo vėliavėlę, kad dar siunčia duomenis, nes dažnai taip būna, kad UART sąsaja veikia lėčiau nei pagrindinis procesorius.

Impulso pločio moduliacija (angliškai „Pulse-width moduliation“) yra naudojama šio projekto visos mechanikos valdyme. Pats moduliacijos principas yra gana paprastas. Skaitmeninis signalas turi savo pasikartojimo dažnį, o pati moduliacija, tai laikas, kiek per vieną periodą yra signalo aukštas lygis. Jeigu užpildymas yra 25%, tai per vieną periodą pirmus 25% laiko signalas turės aukštą lygį, o likusį laiką žemą, tokiu būdu yra formuojami impulsai su kintamu pločiu. Pavyzdžiui, servo mechanizmai pagal impulso plotį nustato savo pasukimo kampą, taip pat galima reguliuoti įrenginių išėjimo galią arba net perduoti informaciją.



11 pav. Impulso pločio moduliacijos pavyzdys, kai periodo užpildas yra 0%, 25%, 50%, 80% ir 100% [17]

5. OpenCV bibliotekų apžvalga

OpenCV yra atviro kodo bibliotekos skirtos komerciniams ir edukaciniams tikslams. Buvo parašyta C ir C++ kalbomis, bet yra perkonvertuota naudojimui su C#, Python, Ruby, Matlab ir Java

kalbomis. Bibliotekų sukūrimui didelę įtaką turėjo poreikis apdoroti vaizdo informaciją realiu laiku, kur programų našumas turi labai didelę įtaką. Naujos bibliotekų versijos palaiko kelių branduolių procesorius. [18]

Vienu iš pagrindinių OpenCV bibliotekos tikslų yra būti gana galingu įrankiu skaitmeninio vaizdo atpažinimo srityje ir tausoti laiką pažengusiems programuotojams, kad kiekvieną kartą neaprašinėti bazinių funkcijų ir algoritmų, bet tuo pat metu nebūti sunkiai perprantama paprastam naudotojui.

Kaip jau įvade buvo minėta, OpenCV biblioteka buvo pradėta kurti 90 metais Intel kompanijos. Tikslas buvo labai paprastas: sukurti įrankį, kuris bus naudojamas plačiai besivystančioje srityje ir kuris skatins vartotojus pirkti greitesnius procesorius, o tai, savo ruoštu, skatins naujų procesorių projektavimą ir gamybą. Patiems sukurti pilnai veikiančią programinę įrangą yra neekonomiška laiko ir finansų atžvilgiu. Intel pastebėjo, kad daugelio universitetų darbuotojai su studentais kūrė savo vaizdo apdorojimo įrankius, Intel tuo pasinaudojo ir dėl to, kad daug studentų įnešdavo savo darbo dalį į bendrą galutinį produktą – vystymas vyko gana sparčiai. Taip ir atsirado OpenCV bibliotekos branduolys, o dėl to, kad projektas yra atviras ir bet kam prieinamas, kiekvienas vartotojas gali įnešti savo indelį į tolimesnį programos vystymą. Nuo 2000 metų buvo nuolat išleidžiamos beta testinės versijos ir 2006 metai buvo išleista jau pirma oficiali OpenCV bibliotekų stabili versija.

OpenCV bibliotekos susideda iš pagrindinių penkių komponentų:

- CV ir CVaux – bibliotekos komponentai, kurių paskirtis filtruoti, transformuoti, konvertuoti, analizuoti vaizdo informaciją. Be to atlikti morfologinius skaičiavimus, ieškoti objektų kontūrų bei analizuoti vaizdo histogramas.
- HighGUI – grafinė aplinka, kuri sukuria peržiūros langus, valdymo langus bei rūpinasi vartotojo duomenų įvestimi iš klaviatūros ir t.t.
- MLL – „Machine Learning Library“ – statistinė biblioteka, kuri naudojasi statistikos metodais ir gali klasifikuoti vaizdus bei „apsimokyti“ juos atpažinti.
- CxCore – tai branduolys visų bibliotekų, kurio pagrindas yra matematinės funkcijos, matricų skaičiavimai, Furie transformacija ir daugelis kitų matematinių veiksmų. Taip pat turi XML palaikymo paketus bei įrankius dvimačių objektų paišymui ir t.t.

5.1. OpenCV bibliotekų struktūra ir kintamųjų tipai, statinės ir dinaminės struktūros

Vaizdas kompiuterio atmintyje yra atvaizduojamas matricomis – skaičių sekomis kompiuterio atmintyje. OpenCV prisitaikė prie to ir duomenis apdorojimui siunčia konvertuotus į pilką vaizdą ir

rezultato matricoje vertės būna nuo 0 iki 255 (pilko atspalvio intensyvumas) arba nuo 0 iki 1 (binarinė skalė). Tai nereiškia, kad vaizdas netenka spalvos, tiesiog vaizdas yra išardomas į atskiras spalvas ir po to kiekviena sparva atskirai yra apdorojama kaip pilka spalva, nes kompiuteris nesupranta spalvų kaip žmogus. [18]

Verta paminėti, kad funkcijų, kintamųjų struktūrų ir kintamųjų klasių pavadinimai bibliotekoje visada prasideda priešdėliu „cv“, kas yra trumpinys nuo „computer vision“ (išvertus iš anglų kalbos: „kompiuterinis matymas“).

OpenCV turi tris pagrindines statines struktūras, kurios vadinasi IplImage, CvMat ir CvArr.

- CvMat struktūra aprašo dvimačias matricas, bet praktiškai tai yra gana abstraktus pavadinimas, nes šios matricos yra sudėtingesnės už algebroje naudojamas. OpenCV matricos yra aprašomos per du parametrus, kurie aprašo vaizdo gylį ir vaizdo spalvų erdvę (Spalvotam vaizdui gali būti priskirta daugiau nei viena vertė, tai yra kiekvienai spalvai yra priskiriama raudonos, žalios ir mėlynos spalvos intensyvumo vertė).
- IplImage struktūra yra naudojama vaizdo apdorojimo procese kaip vaizdo saugojimo vieta ir turi savyje visus kintamuosius nurodančius vaizdo formatą ir tipą. Gali saugoti 8, 16 ir 32 bitų spalvos vaizdus.
- CvArr yra abstrakti klasė, nes naudojama dažniausiai yra kaip tarpininkas darbe su kitomis matricomis ir perduodant duomenis tarp funkcijų.

Dinaminės struktūros ir kintamieji yra neatsiejami apdorojant vaizdą realiu laiku. Dažnai yra naudojami kaip buferiai, kurie talpina savyje vaizdo kadrus, kuriuos paima iš išorinių šaltinių, pavyzdžiui kamerų. Statinė atmintis paskiriama programos kompiliavimo metu ir lieka paskirta visam programos veikimo laikui. Tai gali būti labai neefektyvu, kai iš anksto nežinome, kiek programos vykdymo metu iš tikrųjų reiks atminties, pavyzdžiui, masyvui ar matricai saugoti. Galimas problemos sprendimo būdas – paskirti masyvui atminties tiek, kad bet kokiu atveju jos pakaktų. Vis tik panašiais atvejais žymiai racionaliau masyvui skirti dinaminę atmintį programos vykdymo metu, kai jau bus žinoma, kiek konkrečiai atminties prireiks. Kai masyvas nebereikalingas, jam skirtą atmintį iškart galima atlaisvinti ir naudoti kitiems programos tikslams.

ROI (angliškai „Region of interest“) algoritmas buvo sukurtas tam, kad paspartinti vaizdo apdorojimą. Principas remiasi tuo, kad vietoje viso vaizdo apdorojimo, kad padidinti spartą yra pasirenkama tik stačiakampė dalis viso kadro, kuri yra naudojama tolimesniems skaičiavimams. Iš to ir kilo jo pavadinimas – domėjimosi sritis. Pavyzdžiui, kai sekamas objektas yra surastas vaizdo kadre, tada yra paimama vaizdo kadro dalis su sekamu objektu ir toliau yra dirbama su juo. Jeigu

objekto pozicija pasikeičia, tai ir pasikeičia ROI stačiakampio pozicija. T.y. ROI stačiakampio centras koreguojamas taip, kad sutaptų su objekto centru.

COI (angliškai „Color of interest“) algoritmas veikia vienodai kaip ROI, tik kad vietoje išskirtos srities kadre, yra siunčiama apdorojimui tik mus dominanti spalva, o visos kitos spalvos yra interpretuojamos kaip juoda arba kaip nuliai matricoje. COI algoritmas labiau veikia kaip filtras ir todėl jo sparta palyginus su ROI yra mažesnė.

6. OpenCV bibliotekos vaizdo apdorojimo funkcijos

6.1. Vaizdo blukinimas

Dažniausiai funkcija yra naudojama, kaip triukšmo ir vaizdo iškraipymo šalinimo įrankis arba, kaip įrankis smulkių detalių šalinimui, kurios gali trukdyti efektyviai apdoroti vaizdą tolimesniuose veiksmuose. Filtras gali būti Gauso, abipusis, medianinis ir daug kitų. [18]

Medianinis filtras yra naudojamas atsitiktinių trukdžių šalinimui, kurių intensyvumas daugiau nei dvigubai skiriasi nuo šalia esančių pikselių intensyvumo vidurkio.

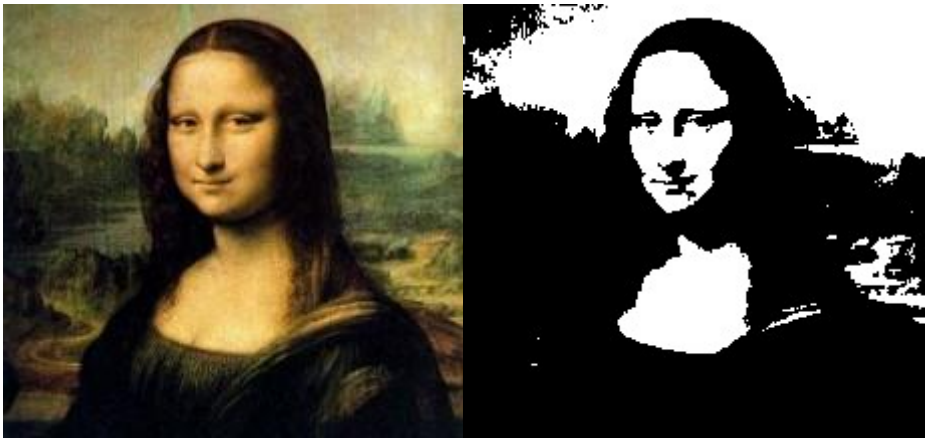
Gauso filtras yra naudojamas vaizdo kontrasto mažinimui – vaizdo blukinimui. Gali panaikinti smulkias detales iš vaizdo kadro, kurios nėra mums svarbios. Algoritmas primena medianinį filtrą, bet naudoja kitą kriterijų: Gauso kaukę – tai matrica, kuri atvaizduoja Gauso funkciją.

Abipusio filtro (angliškai „Bilateral“) privalumas yra tas, kad jis neblukina objektų kraštų. Veikia kaip Gauso filtras, bet filtro parametrai ir Gauso kaukės vertės priklauso ne tik nuo atstumo tarp pikselių, bet ir nuo jų šviesumo skirtumo. Tai veikia tarsi grįžtamąsis ryšis ir ten kur yra kraštas, blukinimas bus mažesnis, o kur pokytis šviesumo yra mažesnis – blukinimas bus intensyvesnis.

6.2. Objekto slenkstinis „Threshold“ atpažinimas

Slenkstinis atvaizdo atpažinimas yra vienas iš paprasčiausių atpažinimo būdų, kur reikia išskirti vienalyčio objekto kontūrą, remiantis šio objekto spalva. Atpažinimas yra vykdomas palyginant kiekvieno pikselio vertę su užduotomis iš anksto kraštinėmis sąlygomis ieškomos spalvos. Jeigu kraštinių sąlygų ieškomos spalvos atspalvio intervalas yra užduotas per mažas, tai dėl kritusio šešėlio arba netolygaus apšvietimo vaizdo atpažinimas gali būti nesėkmingas, tas pats galioja, jeigu intervalas bus per didelis, nes tada bus surasti ir tie objektai, kurių spalva neatitinka ieškomo objekto spalvos. Yra keli slenkstinio atpažinimo būdai. Binarinis būdas – tai kada visas fonas yra perdaromas į nulius, o vienalytis objektas yra pažymimas kaip vienetai vaizdo matricoje, kitas variantas yra

pažymėti tik objekto kraštus kaip vienetus. Taip pat galima vietoje kraštinių sąlygų užduoti tik slenkstį, virš kurio vertė bus visada viena, o nepasiekus jos – kita. [18]

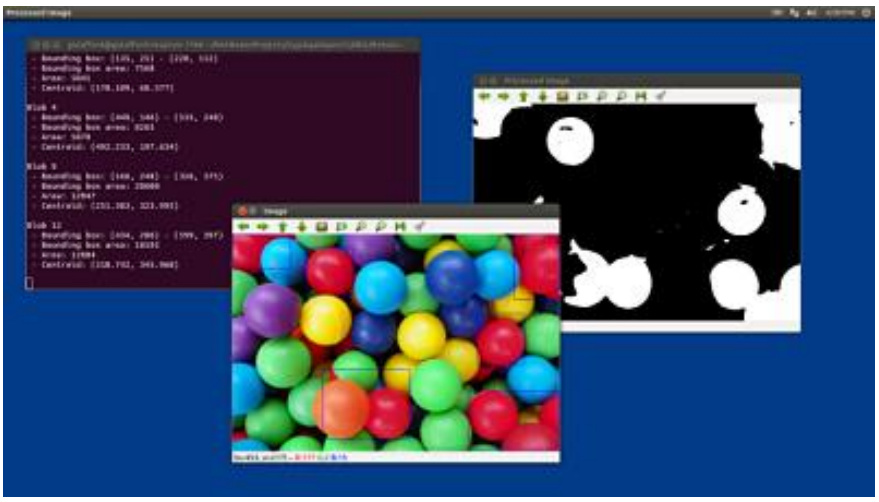


12 pav. Mona Lizos paveikslo pavyzdys pritaikius slenkstinio apdorojimo funkciją [19]

6.3. RGB vaizdo konvertavimas į HSV formatą

ASŠ (atspalvis, sodrumas ir šviesumas) spalvos atpažinimo būdų (angliškai „HSV (Hue, Saturation, Value) color detection“).

HSV vaizdo atpažinimas yra panašus į 6.2 skyriuje aptartą būdą, tik vietoje šviesumo, algoritmas remiasi RGB spalvų vaizdo konvertavimų į HSV (atspalvis, sodrumas ir šviesumas) vaizdą. Tada bet kokią spalvą gali aprašyti vienas kintamasis. Tada tik lieka sudaryti matricą, kur visur visos kitos spalvos turi vieną vertę, o ieškoma spalva – kitą vertę. Be to, kai spalva yra aprašoma vienu kintamuoju, tai galima gana lengvai išskirti ne tik tiksliai vieną spalvą, o visą intervalą spalvų arba kelis atspalvius. Turint tokią matricą, kaip pavaizduota 13 pav., telieka surasti išsiskiriančias vietas ir pažymėti jas, kaip surastą objektą. Neieškomos spalvos yra pažymimos juoda spalva – nuliai matricoje, o ieškoma balta spalva – vienetai matricoje. Algoritmas atrenka ieškomą spalvą pagal užduotas kraštines sąlygas. Dažnai yra papildomai naudojami aproksimacijos arba kiti algoritmai, kad surasti tik mums reikalingą formą, bet apie tai vėliau. [18]

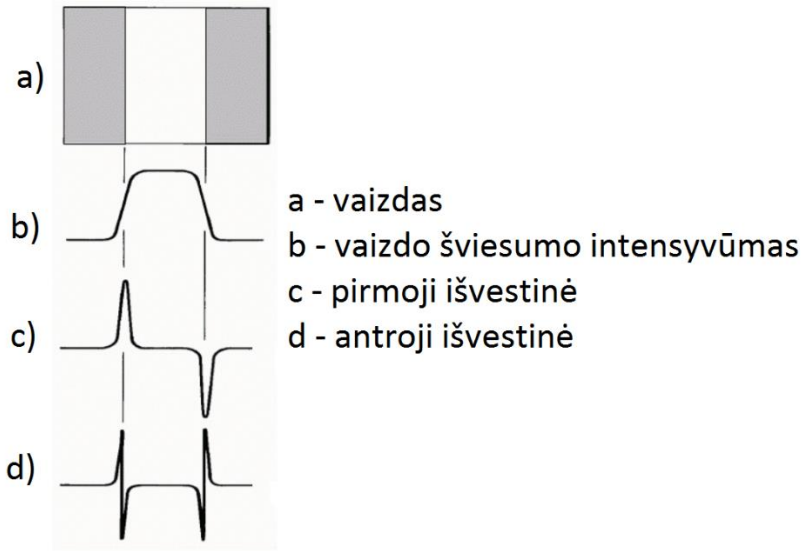


13 pav. HSV vaizdo atpažinimo ir filtravimo pavyzdys [20]

6.4. Krašto atpažinimo filtrai

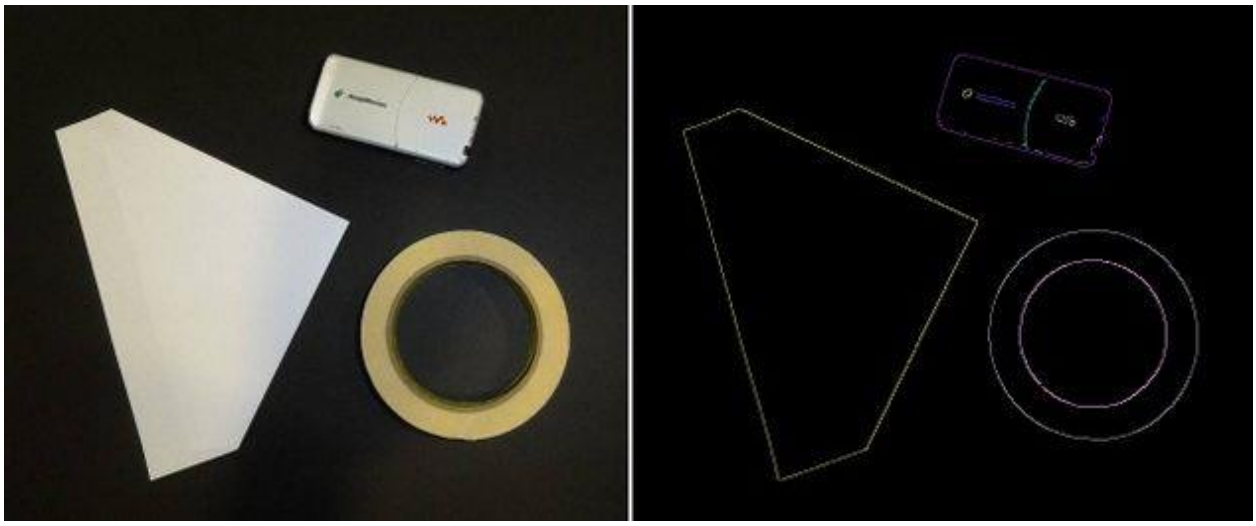
Objektų kraštų (angliškai „edge detection“) atpažinimas vyksta skaičiuojant pirmą ir antrą išvestinę funkcijos, kuri aprašo pikselių šviesumą vaizde. Diskretiniame vaizde pirmos išvestinės skaičiavimas prasideda nuo diskretinio vaizdo operatoriaus gradiento radimo, o antros išvestinės – operatoriaus gradiento krypties pokyčio radimas. Filtrai, kurie remiasi pirmąja išvestine geriausiai tinka vaizdo atpažinimui, kurio ieškomi objektai gana stipriai išsiskiria nuo fono. Tai yra: spalvos ir šviesumo pokytis yra gana lengvai pastebimas. Antros išvestinės filtrai yra lėtesni, bet naudingesni tada, kai objektai nėra labai lengvai išskiriami iš vaizdo fono. [18]

Krašto detekcijos filtrus reikia naudoti atsargiai, nes jie taip pat padidina vaizdo foninį triukšmą, nes išryškina neryškius kontūrus, kurie gali sukelti klaidingus skaičiavimo rezultatus.



14 pav. Vaizdo išvestinių vaizdas

Vaizdo matricoje pokyčio gradientas tai skirtumas tarp kaimyninių pikselių šviesumo vertės. Kad padidinti efektyvumą ir supaprastinti skaičiavimus ateityje, galima sudaryti naujas vaizdo matricas, kur visur yra 0 „juoda spalva“ ir 1 („balta“) – kur išvestinių vertė viršija mūsų užduotą ribinę vertę. Tokiu būdu pašaliname iš vaizdo viską, išskyrus mus dominančius objektų kraštus juodai baltame paveikslėlyje.



15 pav. Objektų kraštų radimo filtru apdorotas vaizdo kadras [21]

Galimi filtrų variantai [18]:

- Sobel‘o filtras yra tiesiog vaizdo matricos sudauginimas su Sobel‘o operatorium – filtro kaukė (matrica 3x3, 5x5 arba 7x7 matmenų). Kuo didesnė kaukė yra naudojama, tuo su didesniu tikslumu yra ieškomi kraštai, bet sparta sąlyginai nėra tada didelė. Didinant skaičiavimo spartą galime pasirinkti mažesnę kaukę, bet atpažinimo tikslumas nukentės;
- Canny‘o filtras yra gana stipriai išvystytas ir naudojamas ten, kur reikalaujama:
    - Didelio tikslumo – atpažintas kraštas turi būti tiksliai ten, kur yra tikrasis;
    - Detekcijos neturi dubliuotis – vienas kraštas detekuojamas tik po vieną kartą;
    - Turi apdoroti didelius duomenų kiekius.

  Kad sumažinti triukšmą vaizde, algoritmas iš pradžių naudoja Gauso filtrą, tada pritaiko Sobel‘o filtrą vertikaliai, horizontaliai ir įstrižai vaizdo kadrui. Surinkus visus duomenis surastos vietos yra sumuojamos ir pagal užduotas ribines vertes yra nusprendžiama, ar pažymėti esamą vietą kaip kraštą, ar ne. Todėl, kad rezultatai yra skaičiuojami iš visų galimų pusių, tai suvidurkinus rezultatus, kraštas yra detektuojamas ten, kur ir yra tikras objekto kraštas. Be to, kad išvengti trūkių objekto surastame krašte dėl kameros iškraipymo arba dėl atsiradusios vietos, kur nėra surastas kraštas (trūksta intensyvumo, kad viršyti ribinę vertę)

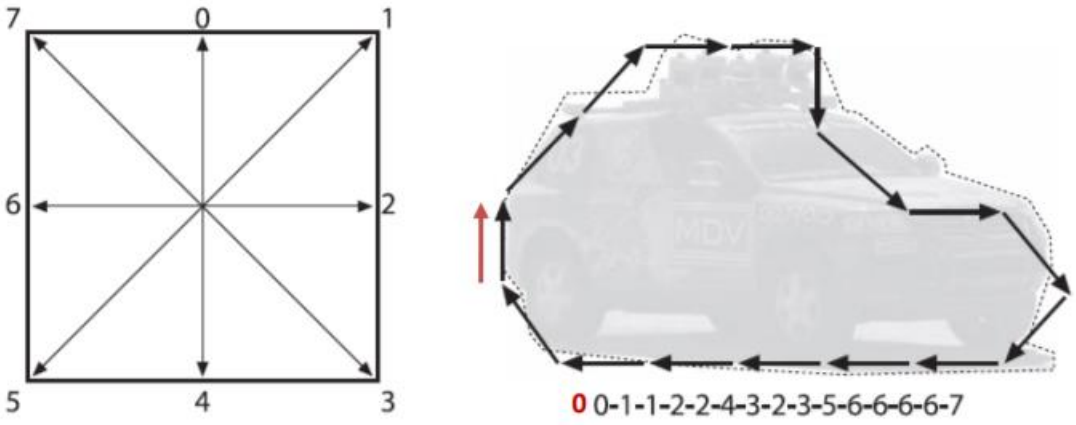
yra įvedamas papildomas algoritmas. Šis algoritmas įveda pataisą, atsižvelgiant į tai, ar gretimuose pikseliuose buvo surastas objekto kraštas;

- Laplaso filtras pasižymi tuo, kad galima aptikti beveik visus kraštus vaizde ir yra labai tikslus. Veikimas yra paremtas Laplasjano skaičiavimu, vertikaliai dalinant vaizdus į skaičių eilutes ir horizontaliai dalinant į skaičių eilutes, ir tada sumuojant jų antrojo laipsnio išvestines. Antros išvestinės skaičiavimui filtras naudoja Sobel‘o operatorių, kuris yra sudauginamas su įvesties matrica.

6.5. Kraštų detekcija ir Freemano algoritmas

Panaudojus kraštų atpažinimo filtrus turime matricas, kur kraštai yra atvaizduoti skaičiais, bet tarpusavyje jie nėra sujungti ir yra nepriskirti ieškomiems objektams programoje. OpenCV turi integruotas savyje funkcijas, kurios gali mums su tuo padėti. Pradedame nuo to, kad visi tiesūs kraštų komponentai yra sujungiami į tieses. Tada jeigu jų galai kerta viena kitą arba persikloja, tada tos vietos yra sujungiamos – formuojama grandinė iš tiesių ir darbinėje atmintyje lieka saugomi taškai tiesių sujungimų ir kurios tiesės tarpusavyje jungiasi. [18]

Kitas būdas aprašyti visą kontūrą yra naudoti Freeman‘o grandinės algoritmą. Visos atkarpos yra naudojamos vienodo atstumo. Freeman‘o algoritmas naudoja savo žvaigždyną, pavaizduotą 16 pav. Žvaigždyne yra nurodyti skaičiai ir jiems priskirti vektoriai, kurie turi savo nurodytą kryptį. Taigi atmintyje turint tik pradinio taško koordinates ir skaičių seką, nurodančią vektorių kryptį, galime pilnai aprašyti turimą kontūrą. Jeigu turime santykinai mažą objektą arba objekto kraštų pokyčiai yra labai maži, dažnai prireikia vidurkinti ir aproksimuoti jo kontūrą, nes neįmanoma tiksliai atkartoti jo vektoriais, kaip parodyta 16 pav.



16 pav. Freeman‘o algoritmo veikimo pavaizdavimas [18]

6.6. *Tiesinė Hough‘o transformacija*

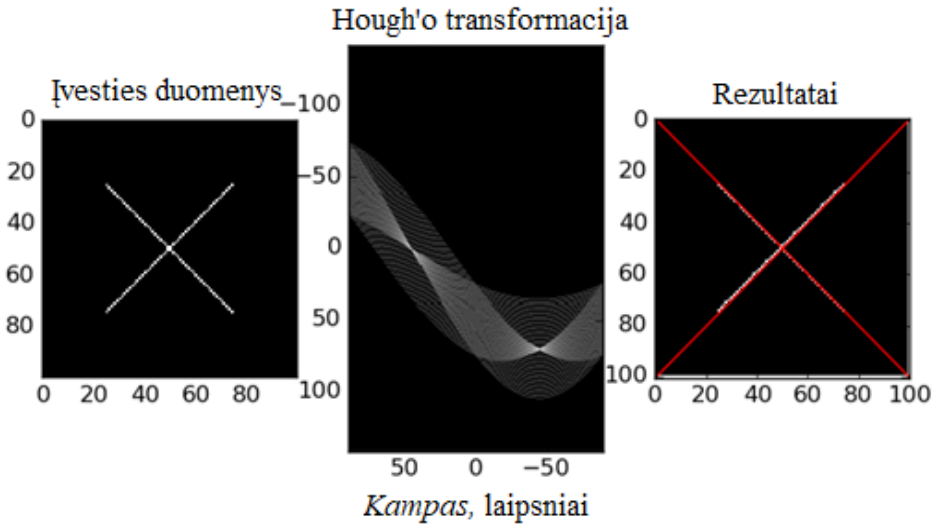
Kai jau turime identifikuotus objekto kraštus galime pradėti atpažinimą, ar tai mūsų ieškomas objektas. Labai dažnai dėl aplinkos sąlygų surandama yra tik dalis objekto krašto. Jeigu mums reikia surasti objektą, kurio forma galima aprašyti tiesėmis, apskritimais arba elipsėmis, patogu yra pasinaudoti Hough‘o transformacijomis. [18]

Klasikinė Hough‘o transformacija tiesėms teigia, kad kiekviena vaizdo tiesė gali būti atvaizduota sinusoide. Tai yra, tiesė atvaizduojama formule (1) gali būti aprašyta, kaip nurodyta formulėje (2), kur r yra atstumas nuo nulinės koordinatės, o kampas θ yra polinkis normalės, statmenos tiesei į x ašį:

$$y = a\,x + b \tag{1}$$

$$r = x \cos\theta + y \sin \tag{2}$$

Iš pradžių yra paimamas vienas detektuotas krašto pikselis ir per jį yra brėžiama viena tiesė nustatyto iš anksto „protingo“ taškų skaičiaus ilgio (Jeigu mes apdorojame matricą 500x500, tai pasirinkimas būtų apie 500, o ne 5000 taškų, nes veltui eikvotume procesoriaus resursus), tada kita tiesė brėžiama per tą patį tašką kitu minimaliai pakeistu kampu (ant kiek leidžia matricos matmenys) ir taip iki tol, kol visi galimi variantai nebus nupaišyti. Tada tie patys veiksmai yra kartojami su kitais detektuotais taškais. Kaip rezultatą paimame tų tiesių susikirtimo vietas ir sumuojame susikirtimų skaičių kiekviename taške – gauname jų pasiskirstymą erdvėje ir atvaizduojame juos kaip priklausomybę nuo susikertančių tiesių θ kampų. Galima pastebėti, kad kai kuriuose vietose susikirtimų bus daugiau – lokalūs maksimumai. Tie maksimumai atsiranda dėl to, kad piešiamos kreivės iš detektuotų pikselių susikerta tame taške Hough‘o plokštumoje (r, θ) dėl to, kad guli ant tos pačios tiesės mumis apdorojamo vaizdo kadro erdvėje. Kaip matome, Hough‘o plokštumoje tiesė yra atvaizduojama kaip tik vienas taškas, iš kurio parametrų galime apskaičiuoti tiesę vaizdo kadro plokštumoje. Minusas yra toks, kad tokia atpažinta tiesė yra begalinė – negalime nustatyti jos pradžios ir pabaigos.



17 pav. Hough‘o tiesinė transformacija su dviem maksimumais – dvi surastos tiesės [22]

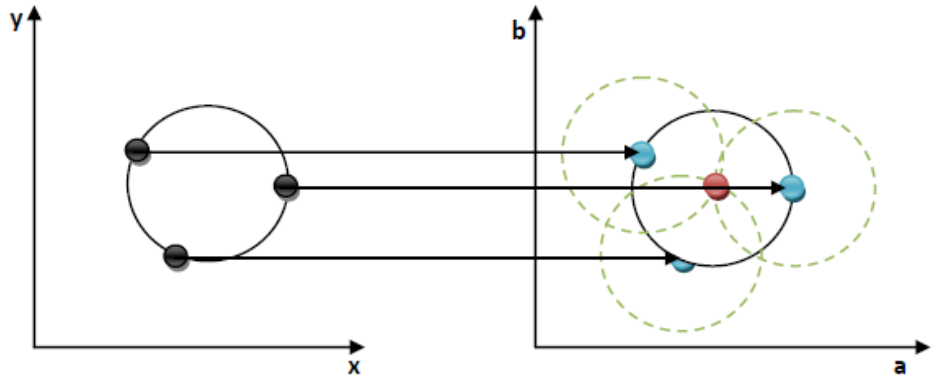
6.7. Apskritiminė Hough’o transformacija

Apskritiminė Hough’o transformacija (angliškai „Circle Hough Transform”), sutrumpintai CHT algoritmas, veikia bruožų ekstrakcijos (angliškai „Feature Extraction“) principu, kuris yra naudojamas mašinų mokymuose, struktūrų ieškojime ir vaizdo atpažinime. [18]

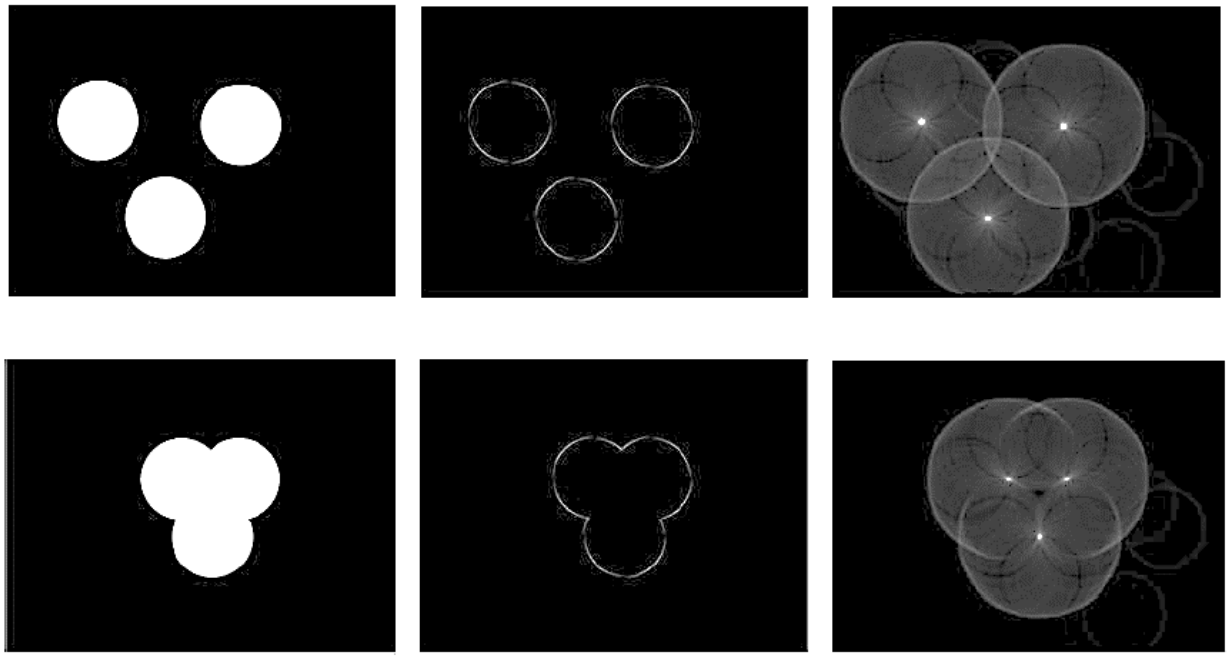
Apskritiminė Hough‘o transformacijos tikslas yra rasti apskritimo formos objektą netobuluose vaizduose. Apskritimo formos „kandidatai“ gaunami „balsavimo" principu Hough‘o parametrų erdvėje, tada pasirenkami vietiniai maksimumai akumuliatorinėje matricoje. Apskritimas gali būti aprašytas formulę:

$$(x - a)^2 + (y - b)^2 = r^2 \tag{3}$$

Kur a ir b yra apskritimo centro koordinatės, o r yra jo spindulys. Turime tris ieškomus parametrus, todėl ir parametrų erdvė yra trimatė. Trimatėje erdvėje, apskritimo parametrai gali būti identifikuojami pagal daugelio kūgio paviršių sankirtas, kurios apibrėžia dvimačio apskritimo taškus. Bėda yra tame, kad turime tris nežinomuosius, todėl dėl uždavinio supaprastinimo vieną iš kintamųjų paimame kaip konstantą (dažniausiai tai būna apskritimo spindulys) ir skaičiuojame kitus du parametrus, po to pakartojame skaičiavimus su pakeista konstanta ir taip per visą galimą verčių ruožą, todėl reikia iš anksto nustatyti protingas ribas, kad nešvaistyti laiko beverčiams skaičiavimams.



18 pav. CHT algoritmo apskritimo centro radimas parinkus teisingą apskritimo spindulį [18]



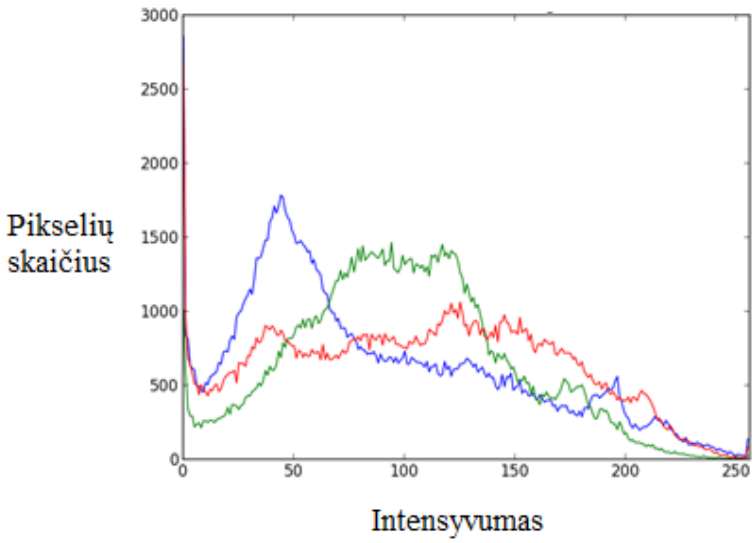
19 pav. Apskritimo centro ieškojimo tarpinių rezultatų vaizdai [18]

Iš pradžių filtravimo dėka yra surandami kraštai matomų figūrų. Tada yra paišomi apskritimai pasirinkto spindulio ant kiekvieno surasto krašto pikselio – 18 ir 19 pav. ir ten, kur paišomų apskritimų kraštai susikerta, akumuliacinėje matricoje yra pridedama po vienetą – „balsuojama“. Po to iš akumuliacinės matricos maksimumų yra išrenkamos daugiausiai „balsų“ gavusios vertės (yra nustatyta slenkstinė vertė „balsų“, kad nebūtų išrenkami klaidingi rezultatai) ir jų koordinatės yra pažymimos kaip apskritimų centras su ieškomo apskritimo spinduliu. Tada viskas yra kartojama su kitu apskritimo spinduliu iš nustatyto ieškomo rėžio.

6.8. Vaizdo spalvų histogramos

Histograma – stulpelinė diagrama, kuri grafiškai atvaizduoja statistinių duomenų pasiskirstymą arba jų tankį. Vaizdo spalvų histograma (23 pav.) nurodo, kiek pikselių kiekvienos spalvos ir intensyvumo yra vaizdo kadre. Histogramos gali nurodyti spalvų kiekius, šviesumo pasiskirstymą ir spalvų pasiskirstymą kadre. Pagal šiuos duomenis po to galima automatiškai koreguoti vaizdo atpažinimo programos kraštines sąlygas. [23]

Paprastesnis variantas yra tiesiog ieškoti dominuojančių spalvų vaizdo kadre. Dominuojančių spalvų algoritmas padalina vaizdą į mažas, vienos spalvos atkarpas ir apjungia jas į vieną stulpelį, jungiant vienodos spalvos atkarpas šalia, tada surikiuoja stulpelio dalis nuo didžiausios atkarpos vienos spalvos iki mažiausios atkarpos. Tokiu būdu gauname stulpelį, kuris mums nurodo, kokios spalvos dominuoja vaizdo kadre. Tokiais būdais yra labai patogu ieškoti fotografijų kopijų, nes jos turės vienodas histogramas ir dominuojančias spalvas, o kad dvi skirtingos fotografijos turėtų visiškai vienodas histogramas, tikimybė yra be galo maža. Toks paieškos būdas gali net ieškoti panašių fotografijų, kurių raiškos nesutampa. Tai yra, dvigubai didesnės raiškos vaizdo kadro histograma bus vienodos formos kaip mažesnio vaizdo kadro. Visų spalvų pikselių skaičius skirsis du kartus, bet santykis dominuojančių spalvų bus vienodas.



20 pav. Vaizdo kadro histograma nurodanti RGB spalvų pikselių kiekius su jų šviesumo intensyvumu [24]

6.9. Koreliacijos vaizdo atpažinimas

Šabloninis vaizdo atpažinimo koreliacijos būdas, kaip iš pavadinimo galima spręsti, yra paremtas koreliacijos principu. Duomenų bazėje yra saugomas objekto vaizdas, kuris po to yra naudojamas koreliacijoje su kiekvienu kadru. Kai koreliacijos koeficientas viršija prieš tai parinktą slenkstinę vertę, objektas yra randamas. Jeigu slenkstinė vertė yra parinkta per didelė, tai bus rastas tik visiškai vienodas objektas, o sumažinus slenkstį, galima bus atpažinti ir panašius objektus. Vis dėl to, nereikia parinkti per mažos slenkstinės vertės, nes tada visur bus randamas objektas, net ten, kur jo nėra. [25]

21 pav. Koreliacijos vaizdo atpažinimo būdo skaičiavimo rezultatas [26]

Konvoliucija (4) matematikoje yra matematinis operatorius, kuris kaip argumentus paima dvi funkcijas „*x*“ ir „*h*“ ir grąžina trečią, kuri, tam tikra prasme, parodo „*x*“ ir „*h*“ persidengimo lygį. Ji apibrėžiama kaip funkcijų sandaugos integralas, po to, kai viena jų buvo perstumta, ir jos argumentas padaugintas iš -1. Vaizdo atpažinime vietoje funkcijų yra naudojamos matricos.

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau, \tag{4}$$

6.10. Judėjimo analizavimas ir objektų sekimas

Optinio srauto (angliškai „Optic Flow“) atpažinimas yra naudojamas kliūčių atpažinime arba objektų sekime. Pats veikimo principas yra gana paprastas. Visas apdorojamas vaizdas yra dalinamas į mažesnius sektorius ir tada skaičiuojamas, ant kiek vaizdas pasikeitė lyginant su praeitu kadru, o

kad neskaičiuoti kiekvieno pikselio, išrenkami keli geriausiai tinkantys sekimui pikseliai arba jų sankaupa, ir skaičiuojamas vektorius, tarp praeito kadro ir dabartinio. Judant kamerai, kuo kliūtis yra arčiau, tuo vaizdo pokyčio vektorius bus didesnis. Taigi, robotas, judant į priekį, visą laiką pasuka ta kryptimi, kur vaizdo vektorių pokytis yra mažiausias. T.y. galimos kliūtys yra kuo toliau. Problema yra tame, kad jeigu kamera nejuda, tai galima detektuoti tik judančius objektus, bet tai taip pat praverčia saugumo įrangoje ir optinio srauto pagalba galime sekti judančius objektus. [18]

22 pav. Optinio srauto atpažinimo pavyzdys [27][28]

Vienodai galima daryti ir su detektuotu kontūru. Žinodami praeitame kadre to kontūro centrą ir naujai apskaičiuoto kadro objekto centrą, tai galime nusipiešti vektorių tarp jų. Tada žinodami kadrų nuskaitymo ir skaičiavimo spartą, galime apskaičiuoti santykinį greitį objekto, judančio kadre. Jeigu dar galime išmatuoti atstumą iki objekto ir kameros padėtį, tai galime tada nustatyti ir tikslius objekto judėjimo greičius.

6.11. Veido atpažinimas ir Haar‘o bruožai

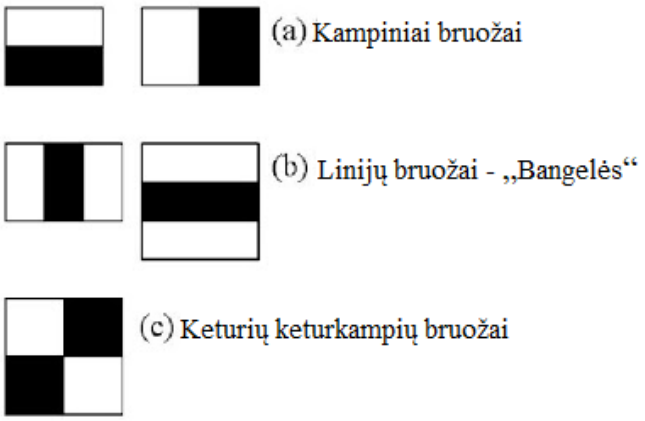
Veido atpažinimas vaizdo kadre yra paremtas algoritmu Haar-Like. Papildomai yra naudojami anksčiau aptarti filtrai kaip papildomi įrankiai pagerinti vaizdo atpažinimo tikslumą. Kad tiksliau veiktų algoritmas, reikia nustatyti eksperimentiškai kraštines sąlygas odos atspalvio radimui, tokiu būdu galima atmesti daug klaidingų veido formos objektų, esančių vaizdo kadre. Sunkumai iškyla tada, kai odos atspalvis gali skirtis nuo apšvietimo intensyvumo. Taip pat veido klasifikavimo parametrai, kurie yra naudojami vaizdo atpažinime yra gana sudėtingai aprašomi, taigi OpenCV biblioteka turi jau veikiantį bazinį bruožų aprašą. Klasifikavimo parametrai yra tarsi šablonas, kurio dėka yra ieškomas veidas vaizdo kadre. [29]

Haar-Like algoritmo veikimo principas yra paremtas dirbtinio intelekto panaudojimu ir Haar kaskadiniu klasifikatoriumi. Haar kaskadinis klasifikatorius, tai matrica nustatyto iš anksto dydžio, kuri turi savyje surinktus veido bruožus, kurie bus naudojami veido ieškojime vaizdo kadre. Dirbtinis intelektas labai praverčia veido atpažinime, nes kiekvieno žmogaus veidas skiriasi ir todėl mums reikia adaptyvaus ir lankstaus algoritmo, kuris galės atpažinti skirtingus, bet tuo pačiu ir panašius objektus.

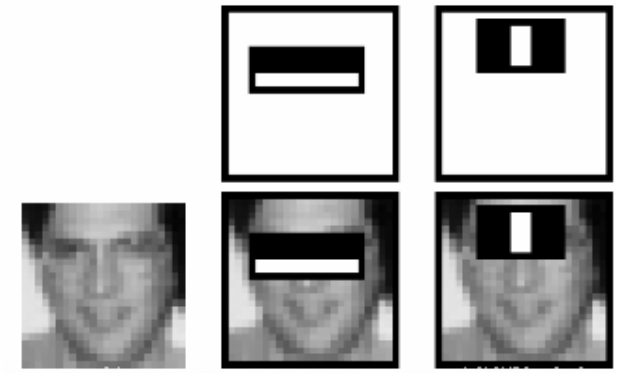
Algoritmui apmokyti reikia turėti daug „teigiamų“ fotografijų su veidais juose ir „neigiamų“ fotografijų be veidų. Algoritmo šablonas kaupia savyje duomenis iš neuroninio tinklo apmokymo – surenka svorinius koeficientus. Algoritmas gali analizuoti tokius parametrus, kaip poziciją, dydį, formą ieškomų objektų, kaip nosies, akių, skruostų ir žandikaulio iš jau žinomų vaizdo kadrų. Kai apmokymas yra pabaigtas, algoritmo surinktus veido bruožus reikia pasiimti iš šablono ir pritaikyti naujų fotografijų apdorojimui su vaizdo atpažinimo funkcijomis.

Algoritmas paima bruožams aptikti parengtas matricas (Haar‘o bangelės - angliškai „wavelets“) pavaizduotas 23 ir 24 pav. Klasifikatoriaus dydis yra nustatomas ne kaip viena vertė, o dažniausiai nuo 24x24 matmenų dydžio iki maksimalios vaizdo kadro raiškos. Taigi, veido paieška bus vykdoma daug kartų ir turi būti vykdoma su visais galimais klasifikatoriaus dydžiais, nes veidų dydis vaizdo kadre ne visada yra vienodas. Klasifikatoriaus ieškomi bruožai yra išrenkami iš anksčiau aprašyto šablono.

Haar‘o bangelės (23 pav.) yra matricinės funkcijos, kurios kartu sudaro bangą (bangos kaip virpesiai), iš kurių galite sukurti kvadratą po atitinkamos transformacijos. Dvimačiu atveju, stačiakampės bangos yra gretimų stačiakampių pora, kai viena pusė yra šviesesnė, o kita tamsesnė. Bruožo ieškojimas vyksta uždėjimo tokios matricos ant vaizdo kadro dalies ir yra stebima vidutinė vertė pikselių šviesesnės ir tamsesnės pusės. Kai pusių pikelių skirtumas viršys nustatytą kritinę vertę, tada galima teigti, kad bruožas buvo aptiktas. Kad netikrinti visų pikselių atskirai, algoritmas pradeda integruoti vaizdo kadro dalis. Mažus vaizdo kadro plotus yra integruojama į vieną vertę, arba, jeigu integruojamas didesnis plotas, tai jis yra konvertuojamas į mažesnės raiškos plotą. Tokiu būdu gana žymiai sumažėja apdorojamos informacijos kiekis.



23 pav. Haar‘o algoritmo bruožų pavyzdys [29]



24 pav. Atpažinimo bruožų pritaikymas ieškant naujo veido vaizdo kadre [29]

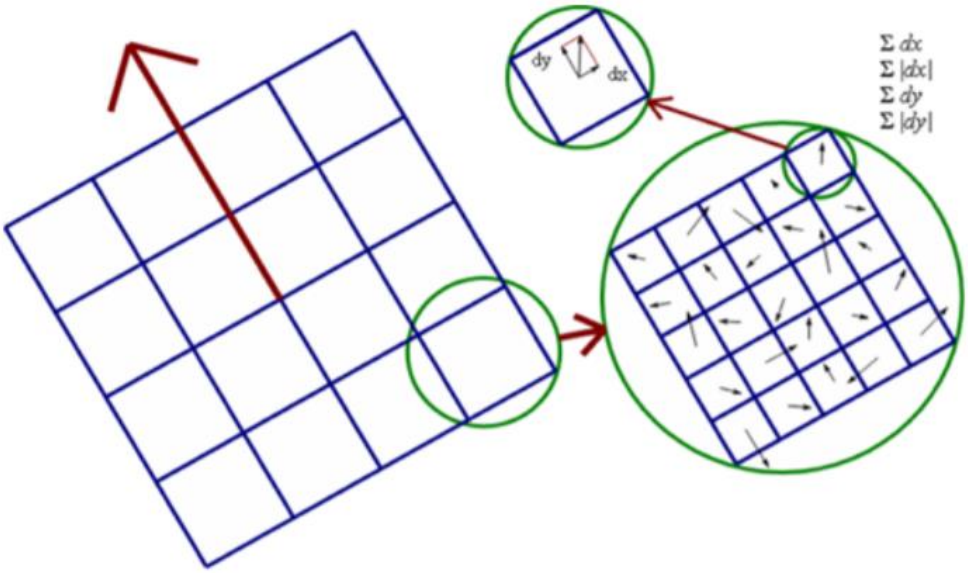
6.12. SURF algoritmas

Pilnas algoritmo pavadinimas skamba „Speeded Up Robust Features“, kas išvertus skamba „Paspartinti ryškūs/žymūs bruožai“. Algoritmas išrenka iš mums žinomo paveikslėlio objekto ryškius, lengvai aptinkamus bruožus ir ieško jų naujuose vaizduose. Vaizdo detektorius remiasi Hesiano matrica (5) (angliškai „Hessian matrix“). *L* – tai konvoliucija Gauso funkcijos antrosios išvestinės apdorojamam vaizdui. Detektoriaus branduoliui yra naudojama interpoliacija vaizdo erdvės, kad jo ieškomi bruožai nepriklausytų nuo vaizdo matmenų ir rezoliucijos. [30]

$$\mathcal{H}(\mathrm{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathrm{x}, \sigma) & L_{xy}(\mathrm{x}, \sigma) \\ L_{xy}(\mathrm{x}, \sigma) & L_{yy}(\mathrm{x}, \sigma) \end{bmatrix} \quad (5)$$

Dešifratorius iš pradžių vaizde pasirinktam taškui priskiria orientacijos vektorių – dažniausiai 60 laipsnių kampo vektorių. Po to aplinkui šį tašką yra formuojamas kvadratinis plotas. Tada šis plotas yra pasukamas vienodai, kaip pasirinktas orientacijos vektorius. Pasirinktas plotas yra

padalintas į mažesnius plotus (gauname 4x4 matricą). Tada kiekvienam iš 16 kvadratinių plotų ieškome bruožų, padalinus juos į dar po 25 kvadratus. Kiekvienam iš šito kvadratinio ploto pritaikome anksčiau paminėtą vieną iš Haar bruožų - „Horizontali bangelė“. Dėka to gauname duoto ploto d*x* ir d*y* vektorius. Priklausomai nuo rezultato gauname horizontalų arba vertikalų vektorių, kurio ilgis yra proporcingas bruožo atitikimo arba neatitikimo koeficientui priklausomai ar tai buvo horizontalus ar vertikalus bruožas. Šis vektorius gali būti aprašomas kaip d*x* ir d*y* vektorių suma. Toliau sumuojame vektorius visų mažesnių plotų, tada didesnių ir gauname galutinį vieną vektorių – vaizdo bruožą. Taip pat sumuojame modulius |d*x*| ir |d*y*|. Tai yra pavaizduota 25 paveiksle. Rezultate gauname keturmatį dešifratorių su d*x*, d*y*, |d*x*| ir |d*y*| vektoriais – bruožais, kurių dėka vaizde galime aptikti ieškomus objektus nepriklausomai nuo jų pasukimo kampo arba neryškių kontūrų.



25 pav. Pasirinkto ploto vektorių formavimo pavyzdys [30]

7. Dirbtinio intelekto įterpimas – neuroniniai tinklai ir mašininis mokymasis

7.1. Integruotos OpenCV „Machine Learning“ mašininio mokymosi bibliotekos

Mašininio mokymosi algoritmai (angliškai „machine learning“) yra atšaka dirbtinio intelekto mokslo srities, kurios pagrindinė užduotis yra sumodeliuoti programą, kurios dėka, mašinos galės pačios priiminėti sprendimus remiantis prieš tai surinkta informacija. Mašininio mokymosi algoritmai dažnai reikalauja tarpdisciplininį žinių ir dažnai susikerta su statistikos, matematikos, fizikos ir vaizdo atpažinimo sritimis. [31]

OpenCV vaizdo atpažinimo bibliotekos turi savyje integruotas mašininio mokymosi bibliotekas, skirtas kurti sprendimų medžius, neuroninius tinklus, spartinimo algoritmus bei palaiko vektorines sistemas.

Duomenų apmokymas (angliškai „Training Data") – tai yra apmokymo duomenų rinkinys, naudojamas dirbtinio intelekto sistemai apmokyti – bazinis žinių paketas. Kartais dar yra vadinamas kaip bruožų vektoriai.

Sprendimų medžiai (angliškai „Decision Trees" arba „Prediction Trees") – tai binarinis medis. Yra galimi tik du išsišakojimai, todėl sprendimų medžiai gali ne visada teisingai skaičiuoti, kai yra daugiau nei du atsakymai. Kaip 26, 27 ir 28 pav. pirmame lange matome, kad remiantis „Training Data" kuriama laipteliuota atsakymų erdvė, kuri ne visada sutampa su teisingų atsakymų erdve. Laiptelių skaičius priklauso tiesiogiai nuo neuroninio tinklo skaičiavimo sluoksnių kiekio. Sprendimų medžiai gali būti naudojami įvesties duomenų klasifikavimui. Dėl to, kad yra visada parenkamas labiausiai tinkantis atsakymas iš dviejų variantų, tai algoritmas gali veikti su nepilnais arba pažeistais duomenimis. Tokius išsišakojimus, jungiant nuosekliai ir lygiagrečiai, gauname medžio šakų vaizdą – iš to ir kilo pavadinimas. Taip pat gali būti naudojami keli sprendimų medžiai, kurie yra vadinami „mišku". Įvesties duomenys yra apdorojami visais sprendimų medžiais ir tada yra išrenkamas dažniausiai atsikartojantis atsakymas arba kelių geriausių atsakymų vidurkis. Reikia pabrėžti, kad kiekvienas medis buna apmokomas su tais pačiais tinklo parametras, bet apmokymo duomenys yra skirtingi.

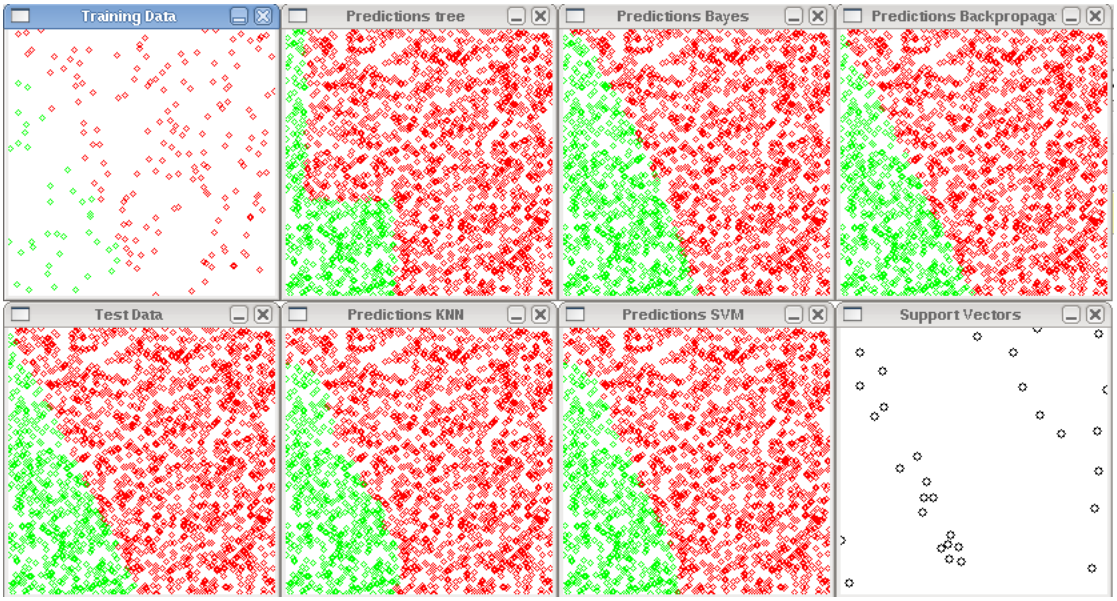
Normalus Bayer'o klasifikatorius (angliškai „Normal Bayes Classifier") nustato naujų įvesties verčių rezultatus pagal duomenų apmokymo duomenis. Klasifikatorius numato, kad naujų verčių išsidėstymas yra tolygus erdvėje ir padalina erdvę į teisingus ir neteisingus plotus. Yra gan spartus algoritmas, bet netinka naudoti, kai atsakymai erdvėje yra pasiskirstę netolygiai.

Arčiausi K nariai (angliškai „K-Nearest Neighbors") algoritmas veikia panašiai į Bayer'o, bet vietoje tiesinio pasiskirstymo jis naudoja nustatytą iš anksto K skaičių, pagal kurį nustatoma, kiek kaimyninių verčių bus naudojama ieškomos vertės apskaičiavimui. Arčiausiai ieškomo atsakymo vertės iš duomenų apmokymo rinkinio yra surenkamos ir balsavimo principu yra apskaičiuojamas rezultatas – arčiausiai esanti apmokymo duomenų vertė turi didžiausią įtaką atsakymui.

Vektorinio palaikymo mašinos (angliškai „Support Vector Machines") – tai yra taip pat vienas iš apmokymo duomenų klasifikavimo būdų. Vektoriškai apjungia apmokymo duomenų rezultatus į atskiras erdvės dalis ir tik palieka ribas tarp jų kaip vektorius – palaikymo vektorius (angliškai „Support Vectors"). Tokiu būdu, jeigu turime labai daug apmokymo duomenų, tai galime sutaupyti atminties vietos.

Atgalinio mokymosi algoritmas (angliškai „Backprogation learning") yra vienas iš dažniausiai naudojamų neuroninio tinklo apmokymo variantų. Apmokymas vyksta koreguojant

neuronų svorinius koeficientus: parinkus pirmus atsitiktinai svorinius koeficientus yra skaičiuojamas atsakymas įvesties apmokymo duomenų. Tada gautas atsakymas yra palyginamas su apmokymo duomenų teisingu atsakymu ir apskaičiuojama paklaida. Galiausiai gauta paklaida yra sudauginama su įvesties verte ir pridedama prie svorinio koeficiento. Po daugelio tokių ciklų neuroninio tinklo koeficientai nusistovi tokie, kad tinklas pradeda skaičiuoti teisingai – apmokymas buvo sėkmingas.



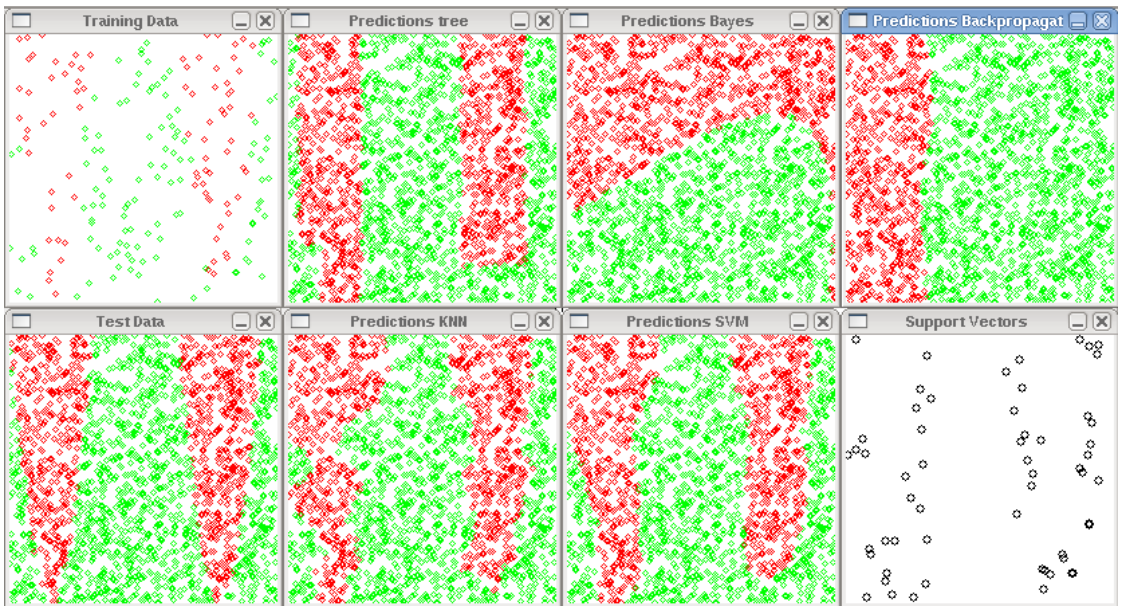
26 pav. Apmokymo, nuspėjimo ir sprendimo priėmimo rezultatai funkcijai $y=2x$ [31]

Vektorinės palaikymo mašinos skaičiavimo tikslumas yra 0.99;

Arčiausių K narių (k=3) skaičiavimo tikslumas yra 0.9825;

Normalaus Bayer‘o klasifikatoriaus skaičiavimo tikslumas yra 0.9425;

Sprendimų medžių skaičiavimo tikslumas yra 0.923;



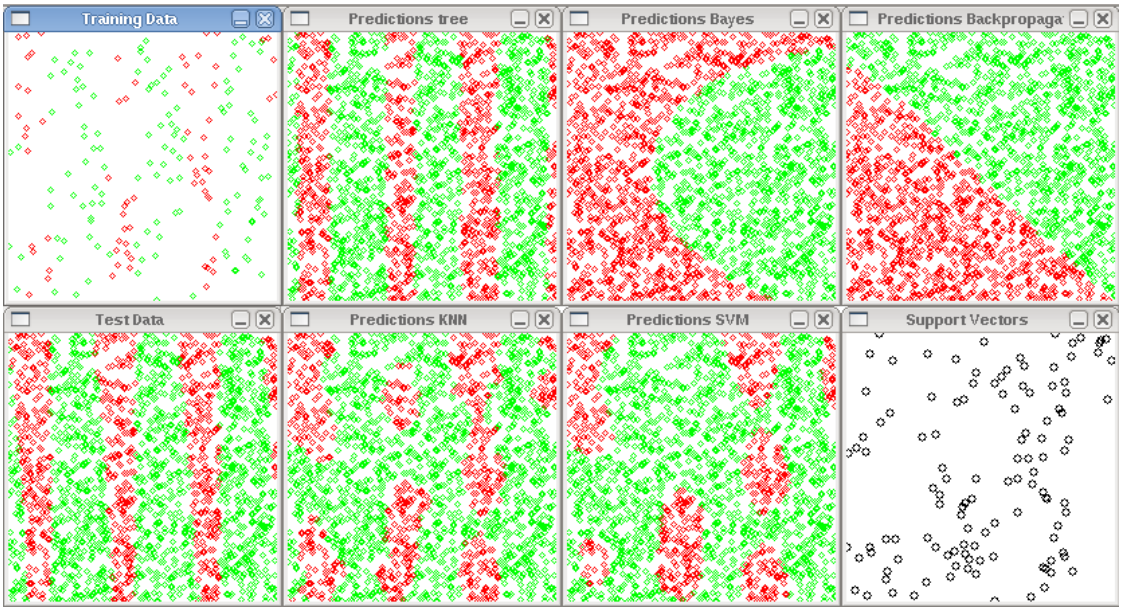
27 pav. Apmokymo, nuspėjimo ir sprendimo priėmimo rezultatai funkcijai *y*=sin(10*x*) [31]

Vektorinės palaikymo mašinos skaičiavimo tikslumas yra 0.913;

Arčiausių K narių (k=3) skaičiavimo tikslumas yra 0.9;

Normalaus Bayer‘o klasifikatoriaus skaičiavimo tikslumas yra 0.632;

Sprendimų medžių skaičiavimo tikslumas yra 0.886;



28 pav. Apmokymo, nuspėjimo ir sprendimo priėmimo rezultatai funkcijai *y*=tan(10*x*) [31]

Vektorinės palaikymo mašinos skaičiavimo tikslumas yra 0.7815;

Arčiausių K narių (k=3) skaičiavimo tikslumas yra 0.819;.

Normalaus Bayer‘o klasifikatoriaus skaičiavimo tikslumas yra 0.542;

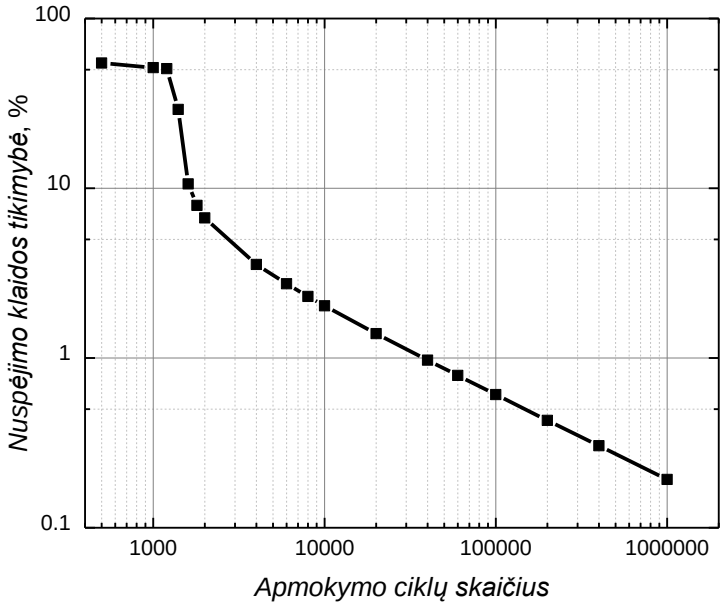
Sprendimų medžių skaičiavimo tikslumas yra 0.9155;

7.2. Neuroninis tinklas vaizdo atpažinimo programos kraštinių sąlygų koregavimui

Dėl Raspberry Pi mini kompiuterio spartos trukumo nebuvo naudotas integruotas į OpenCV bibliotekas neuroninis tinklas (visi galimi procesoriaus resursai buvo skirti vaizdo matricų filtravimui ir matricinių transformacijų skaičiavimams), o buvo sumodeliuotas „rankiniu būdu“ savadarbis dviejų sluoksnių neuroninis tinklas iš 5 neuronų vaizdo atpažinimo algoritmo kraštinėms sąlygoms nustatinėti.

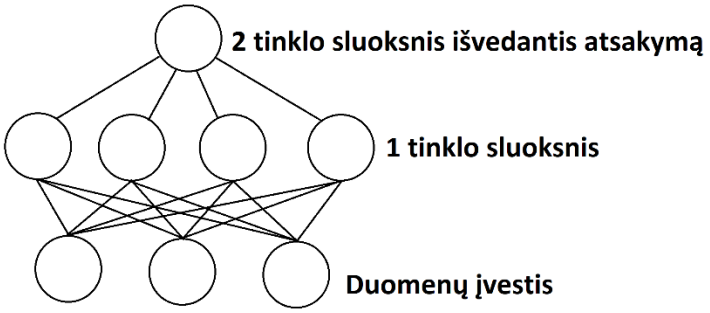
Iš pradžių neuroninis tinklas paima atsitiktinius svertinius įėjimų koeficientus ir palygindamas su apmokymo duomenimis koreguoja juos, kol nepasiekia patenkinamai mažos paklaidos. Problema yra tame, kad nuspėjimai darosi pakankamai tikslūs tik po kelių tūkstančių apmokymo ciklų – o tai taip pat sunaudoja nemažai resursų, nes jeigu žinomi duomenis pasikeičia arba pradedama veikti su kitų duomenų srautu, tai apmokymas vėl pradedamas iš naujo.

Svorinių koeficientų koregavimas – apmokymas vyksta, kai yra paimami apmokymo duomenis ir paduodami į neuroninio tinklo įėjimą ir tikrinama, koks yra gaunamas rezultatas su iš pradžių nustatytais atsitiktiniais koeficientais. Apmokymo duomenyse yra taip pat jau žinomi teisingi atsakymai. Po pirmo skaičiavimo yra nustatoma paklaida tarp apskaičiuoto ir teisingo rezultato. Tada paklaida su neurono įėjimo verte yra sudauginamos ir pridedamos prie neurono svorinio koeficiento. Taip padaroma paeiliui su kiekvienu neuronu ir jo visais įėjimų svoriniais koeficientais. Po daugelio tokių apmokymo ciklų neuroninio tinklo koeficientai priartėja prie reikiamų verčių ir atsakymai į apmokymo duomenis pasidaro „teisingi“ – paklaidų ribose. Tada galime tinklui padavinėti naujus duomenis ir stebėti jo veikimo tikslumą.



29 pav. Sumodeliuoto neuroninio tinklo klaidingo atsakymo tikimybės priklausomybė nuo apmokymo ciklų skaičiaus

Iš 29 pav. eksperimento rezultatų galime pasirinkti mus tenkinantį nuspėjimo tašką ir bandyti subalansuoti veikimo spartą ir tikslumą. Tai yra vienas iš kelių neuroninio tinklo privalumų. Galime paaukoti spartą ir padidinti tikslumą arba mažinti tikslumą, bet turėti didelę spartą. Be to, gerai apmokytas neuroninis tinklas gali gan gerai nuspėti naujų situacijų rezultatus, jeigu jos yra panašaus pobūdžio, kaip ir apmokymo duomenys.



30 pav. Sumodeliuoto neuroninio tinklo struktūra

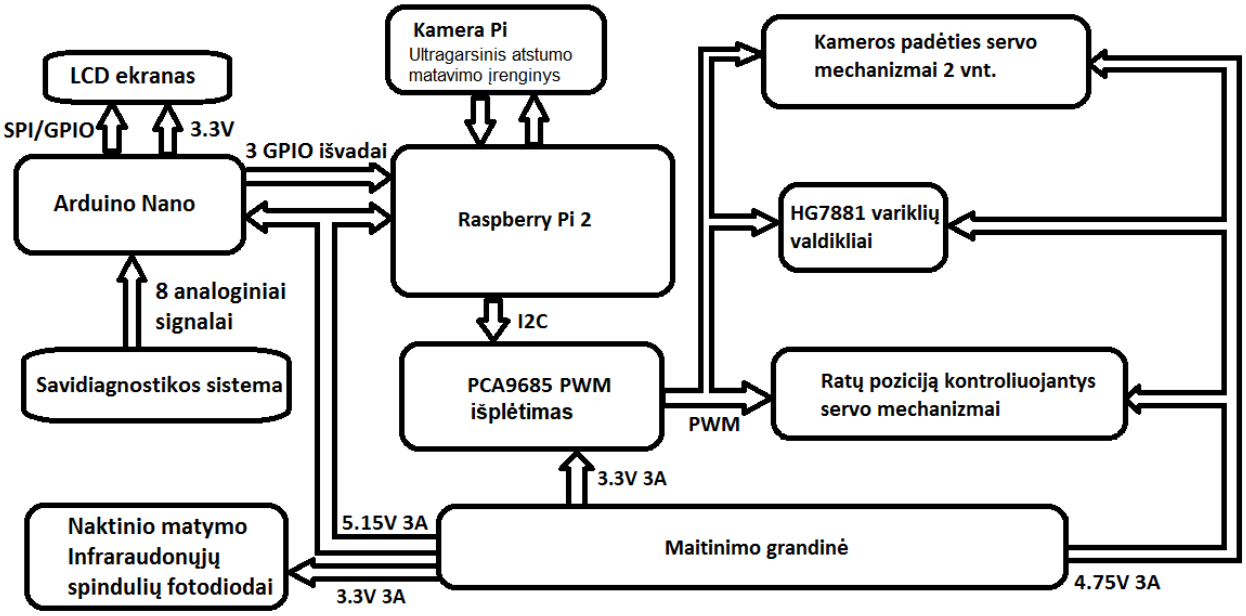
8. Mechaninis tyrimo maketas

Vaizdo atpažinimo testavimui buvo sukonstruota keturratė platforma. Dėl mobilumo kiekvieno rato kampas yra valdomas servo mechanizmo. Kamera taip pat yra valdoma servo

mechanizmais, kas duoda galimybę matyti 180 laipsnių kampu kaip horizontaliai, taip ir vertikaliai. Servo mechanizmu dažniausiai (bet ne visada) yra vadinamas riboto sukimosi kampo variklis su reduktoriumi, kurio pasukimo kampą galima valdyti su impulsine kodine moduliacija, bet pasitaiko modelių, kurių sukimosi kampas yra neribotas, o galima valdyti tik sukimosi kryptį.

Variklių valdymui buvo pasirinkti HG7881 valdikliai dėl savo mažų matmenų ir paprastumo naudojime. Variklių valdikliai ir servo mechanizmai yra valdomi impulsine kodine moduliacija, kas sukelia nepatogumus, nes Raspberry Pi 2 mini kompiuteris turi tik vieną impulsinės kodinės moduliacijos išvadą. Todėl papildomai buvo panaudotas PCA9685 išlėtimas su 16 impulsinės kodinės moduliacijos išvadais. Pats išlėtimas buvo prijungtas per I^2C sąsają. Šita sąsaja yra adresuojama, todėl, esant būtinybei, galima prijungti kitus I^2C sąsajos įrenginius.

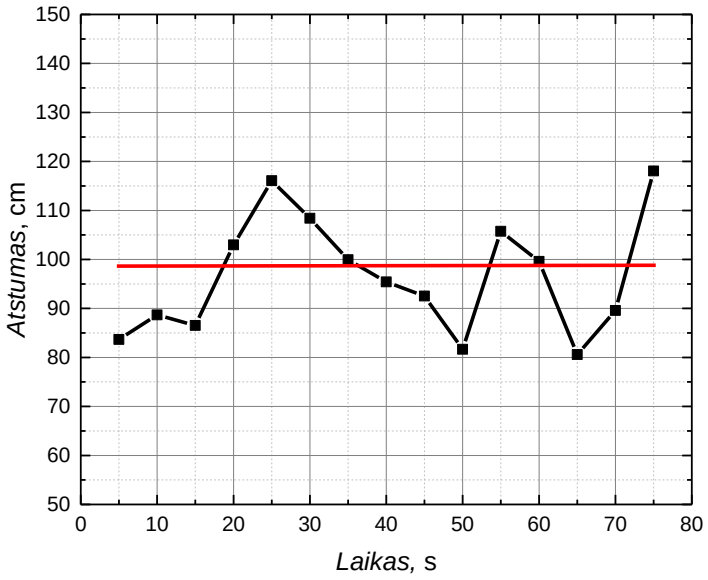
Kaip maitinimo šaltinis yra naudojami trys reguliuojamos įtampos impulsiniai maitinimo šaltiniai LM2596, kurių maksimali išėjimo srovė yra 3 A. Kaip energijos šaltinis buvo paimtas Ličio jonų akumuliatorius dėl savo labai gero talpos – masės santykio. Akumuliatoriaus parametrai: 10.8V, 5Ah ir maksimali išėjimo momentinė srovė 7,5 A. Akumuliatorius turi savadarbę gilaus iškrovimo apsaugą ir perkrovimo apsaugą. Akumuliatoriaus apsauga – tai darbo eigoje užprogramuotas Arduino Nano mikrovaldiklis, kuris veikia kaip diagnostikos sistema, kuri matuoja akumuliatorių įtampas ir maitinimo grandinių įtampas. Matuojamus parametrus išveda į monochromatinį LCD ekraną ir siunčia maitinimo grandinių statusą bei akumuliatoriaus įkrovimo lygį į pagrindinį Raspberry Pi mini kompiuterį per bendro panaudojimo kojeles (GPIO).



31 pav. Važiuoklės maketo konstrukcijos blokinė diagrama

8.1. Ultragarsinis atstumo matavimo įrenginys HC-SR04

Ultragarsinis atstumo matavimo įrenginys yra sumontuotas priekyje virš kameros ir yra skirtas atstumui iki kliūčių matavimui, nes yra gan sudėtinga matuoti atstūmus naudojantis tik vienos kameros vaizdais – nėra stereoskopinio matymo. Todėl, kad mini kompiuteris naudoja operacinę sistemą: veikia daug procesų ir subprocesų vienu metu ir jie dalinasi procesoriaus resursus, galime pastebėti, kad atstumo matavimas gali būti tikslus tik tada, kad yra gan stipriai vidurkinami jo rezultatai. Tai yra dėl to, kad ultragarsinis atstumo matavimas remiasi laiko matavimu nuo ultragarsinio signalo pasiuntimo iki jo grįžimo, atsispindėjus nuo matuojamo atstumo objekto. Jeigu laiko matavimo intervale operacinė sistema perduoda procesoriaus darbą kitam procesui, tai gali atsirasti papildomas užlaikymas. Taip pat ultragarsinė banga gali atsispindėti nuo pašalinių objektų ir įnešti papildomų paklaidų, o ir pats atstumo matavimo įrenginys nėra tobulas ir įneša savo paklaidas.

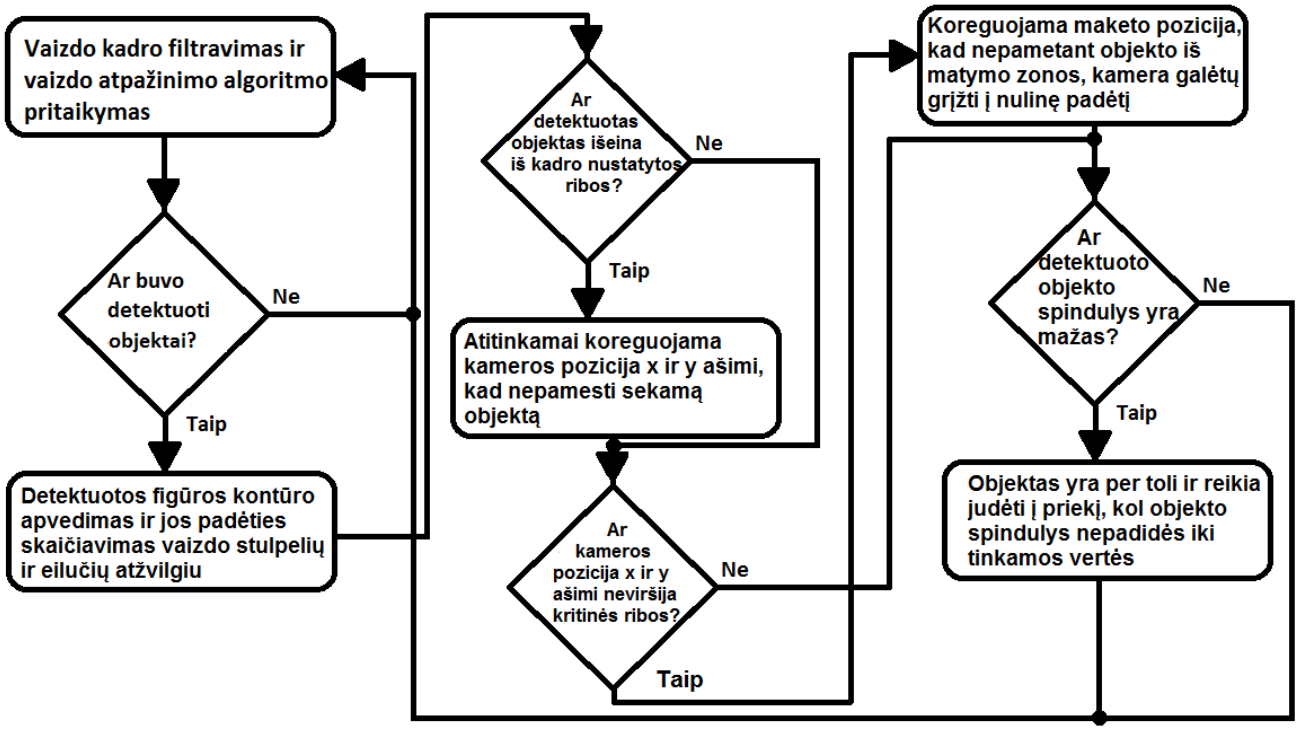


32 pav. Ultragarsinio atstumo matavimo rezultatai, kai kliūtis yra 100 cm atstumu

8.2. Vaizdo atpažinimo algoritmo su mechanine valdymo dalimi sąsaja

Nors vaizdo apdorojimo algoritmas ir atpažįsta objektus vaizdo kadruose, pats kompiuteris nesupranta nei kas tai yra, nei kur tai yra, nei ką su tuo daryti. Tam reikia papildomo algoritmo, kuris galėtų valdyti mechaniką ir tuo pačiu metu sekti objektą, bei suprasti, kur tai yra. Kad nuosekliai visada nevykdyti valdymo algoritmo dalies, per globalius kintamuosius yra nustatomos vertės atitinkamai ar vaizde yra surastas kažkoks objektas, ar ne. Jeigu nieko nebuvo detektuota, tada visa

dalis algoritmo yra praleidžiama ir grįžtama prie naujo kadro apdorojimo. Jeigu kažkas buvo surasta, tada įsijungia sekimo ir valdymo algoritmas ir pradedama sekti objektą bei koreguoti mechaninės važiuoklės ir kameros padėtį, kad nepamesti judantį objektą iš matymo zonos. Visos komandos apie padėties keitimą yra siunčiamos per I^2C sąsają į PCA9685 lustą, kuris impulsinės kodinės moduliacijos dėka valdo visą važiuoklės mechaniką: servo mechanizmų padėties kampą ir variklių sukimosi greitį. Kaip pavaizduota 33 paveikslėlyje, kai yra surastas objektas, pasileidžia ciklas, kuris tikrina kameros padėtį ir objekto koordinates vaizdo kadre, ir kai yra išeinama iš leistinų ruožų, yra koreguojama kameros ir pačio maketo pozicija, kad nepamesti sekamo objekto.



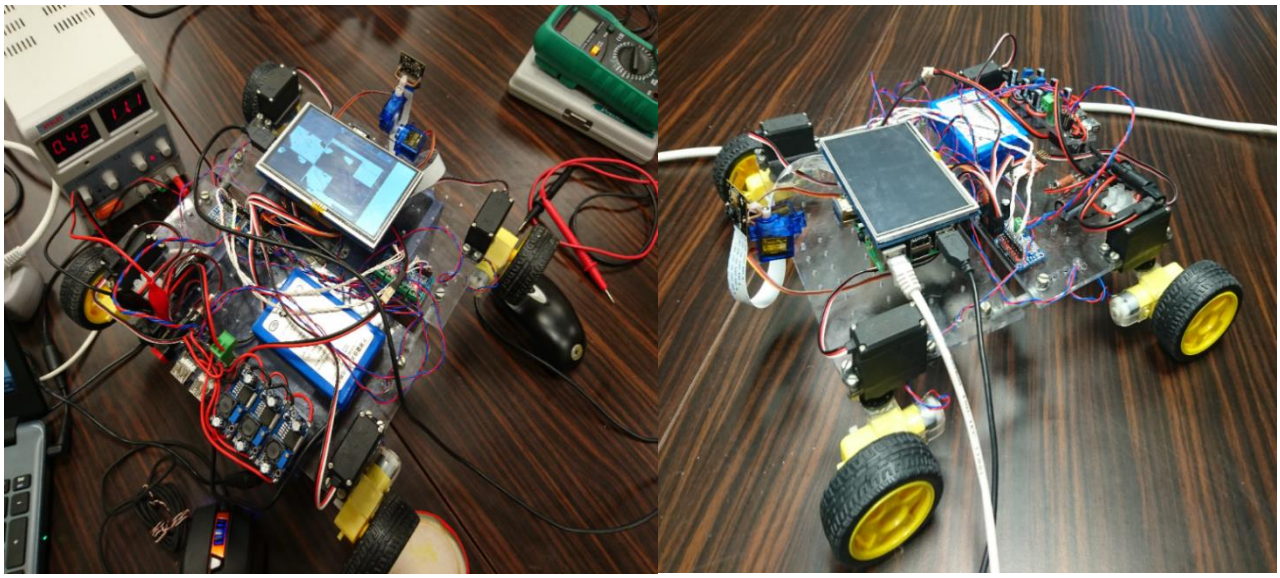
33 pav. Sekimo ir mechaninės dalies valdymo blokinė diagrama

8.3. Naktinio matymo sistema

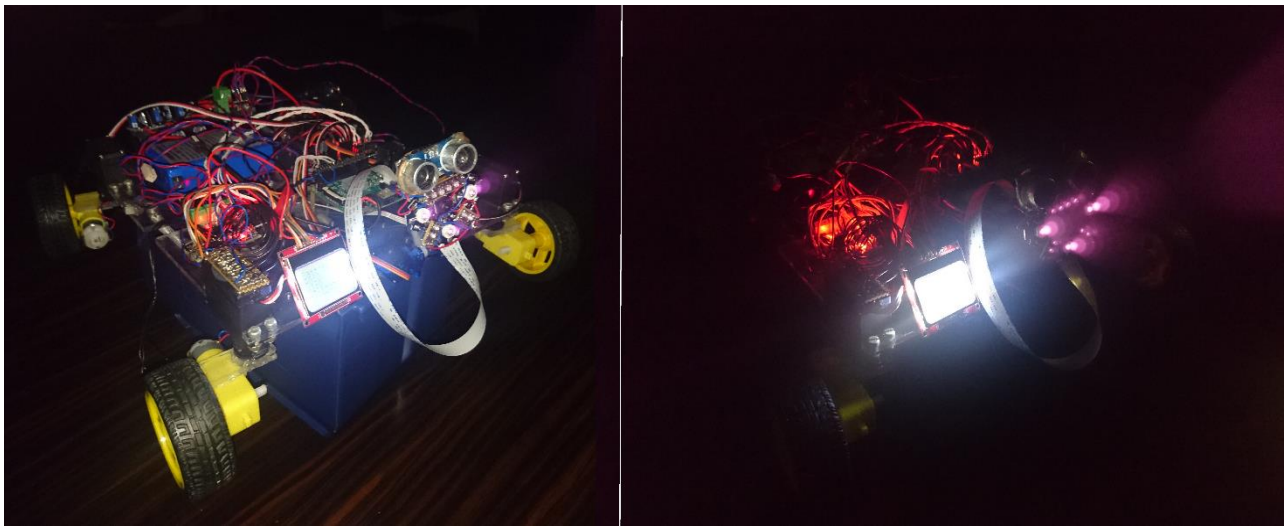
Todėl, kad vaizdo kamerų jutikliai reaguoja į didesnį elektromagnetinių bangų diapazoną nei žmogaus akis, tai galime matyti infraraudonų spindulių vaizdą ekrane. Komercinės video kameros turi uždėtus filtrus, kurie nepraleidžia pašalinės spinduliuotės nei iš infraraudonų bangų diapazono, nei iš ultravioletinio diapazono ir dėl to turime vaizdo kadrą, kuris atrodo labai artimas žmogui matomam vaizdui. Aišku, jeigu pavyzdžiui nukreipsime infraraudonos šviesos šaltinį tiesiai į objektyvą, tai filtras gali nesusitvarkyti su užduotimi ir praleisti dalį spinduliuotės į kameros matricą – tada matysime nuotraukoje, paprastai žmogui nematomą, sklindančią šviesą. Kamera negali mums

perteikti šios spalvos ir teoriškai tai bus atvaizduojama tarsi šviesumo intensyvumo padidėjimas – balta spalva, kuri darysis tuo tamsesnė (pilka), kuo intensyvumas krintančios į objektyvą spinduliuotės bus mažesnis. Kartais galima pastebėti, kad šviesioje aplinkoje filmuojant infraraudoną šviesą, objektyve matysime ne baltą spalvą, o violetinę. Toks reiškinys atsiranda dėl to, kad šviesa įgauna šį atspalvį praėjusi pro filtrus (infraraudonų bangų ir ultravioletinių bangų), taigi, matome pačio filtro spalvą. Naktinio matymo įrenginiuose nėra infraraudonos šviesos filtro ir todėl, jeigu prie kameros yra infraraudonos šviesos šaltinis, kuris apšviečia viską prieš save, o kamera filmuoja šios bangos ilgio atspindžius – tai ekrane galime matyti vienspalvį vaizdą, nors akla akimi nieko nematysime. Dažniausiai šis vaizdas yra pakeičiamas į žalio atspalvio vaizdą, nes jis mažiausiai vargina akis. Tokiu pačiu principu buvo padarytas naktinio vaizdo atpažinimo būdas tyrimų makete, nes buvo pasirinkta Raspberry Pi NoIR kamera, kuri neturi infraraudonų spindulių filtro ir yra labai jautri 850 nm bangos ilgio spinduliams. Kamera taip pat turi automatinę baltos spalvos balanso koregavimo funkciją ir gali visiškai tamsioje aplinkoje kalibruoti vaizdo baltą atspalvį ir „violetinis“ naktinio matymo vaizdas tampa juodai-baltu vaizdu – panaikina pašalinę kameros objektyvo įtaką vaizdui. [9]

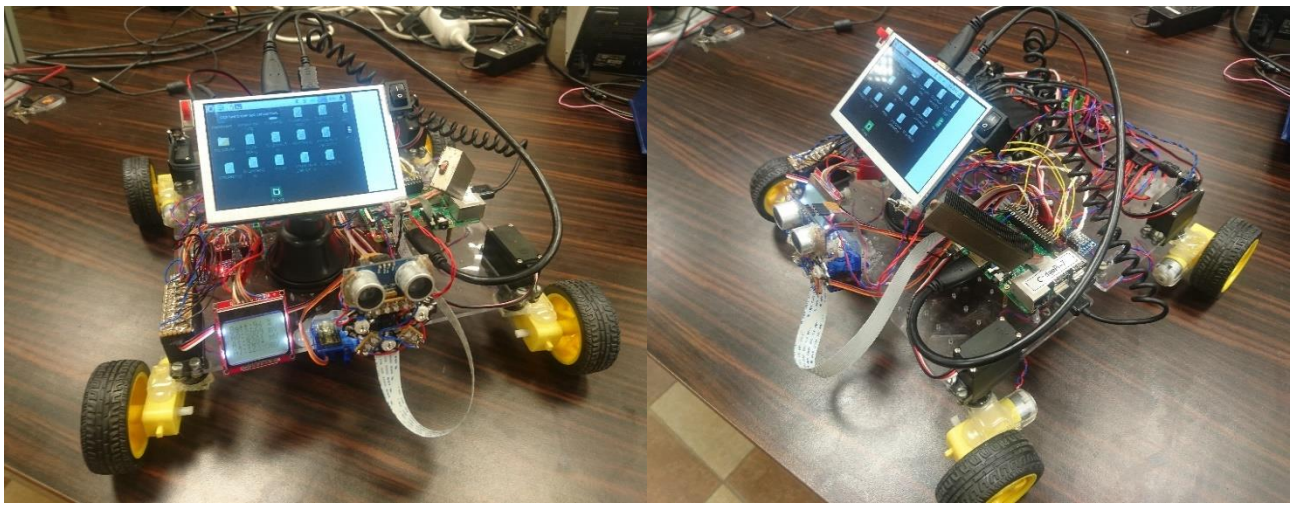
9. Rezultatai



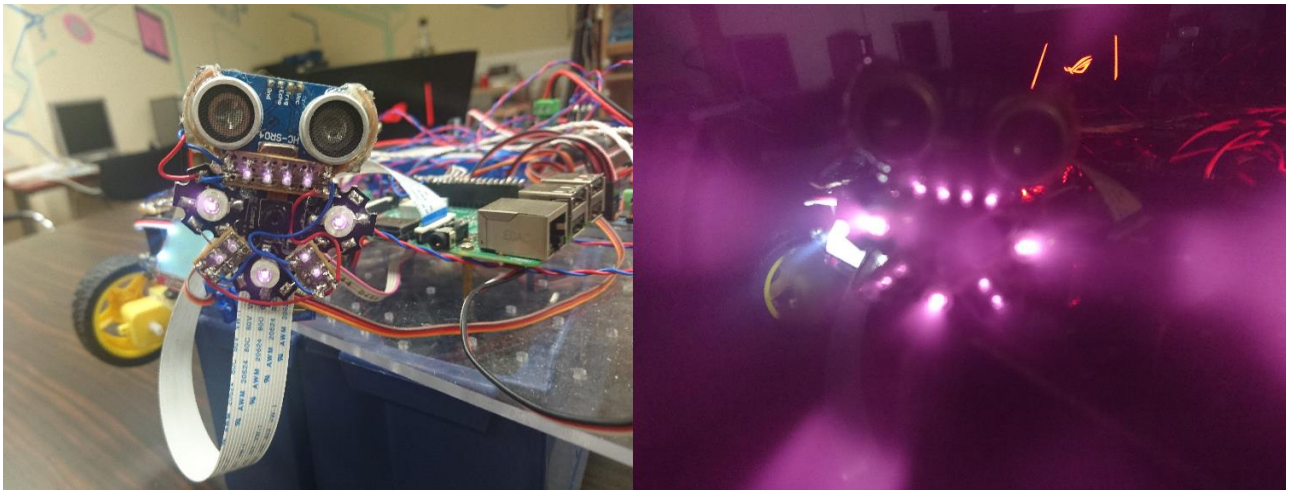
34 pav. Sukonstruotas maketas



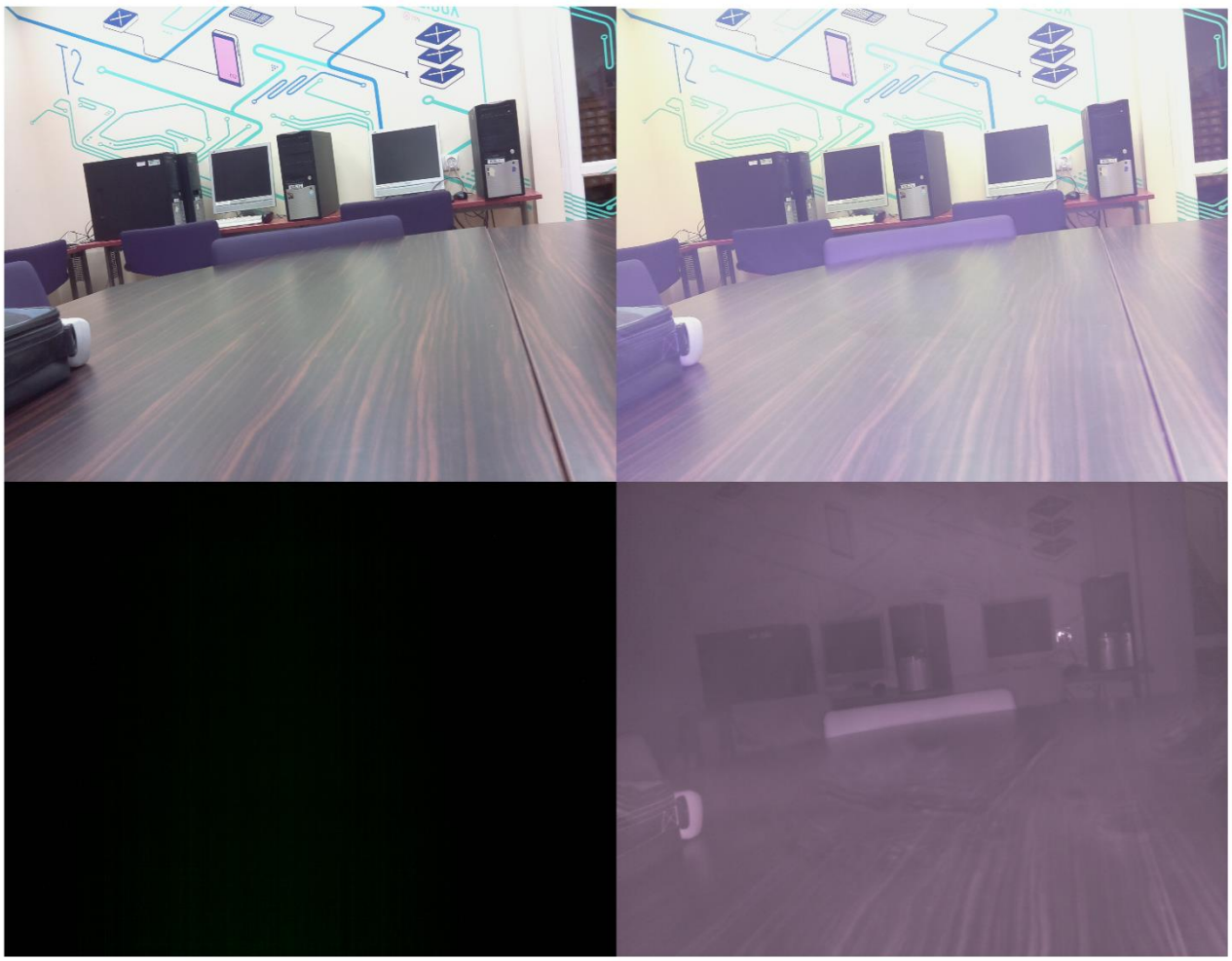
35 pav. Modifikuotas maketas su naktinio matymo sistema



36 pav. Galutinė tyrimų maketo versija



37 pav. Naktinio matymo infraraudonų spindulių šviesos diodai (bangos ilgis 850 nm ir bendra infraraudonų spindulių šviesos diodų galia 5 W)

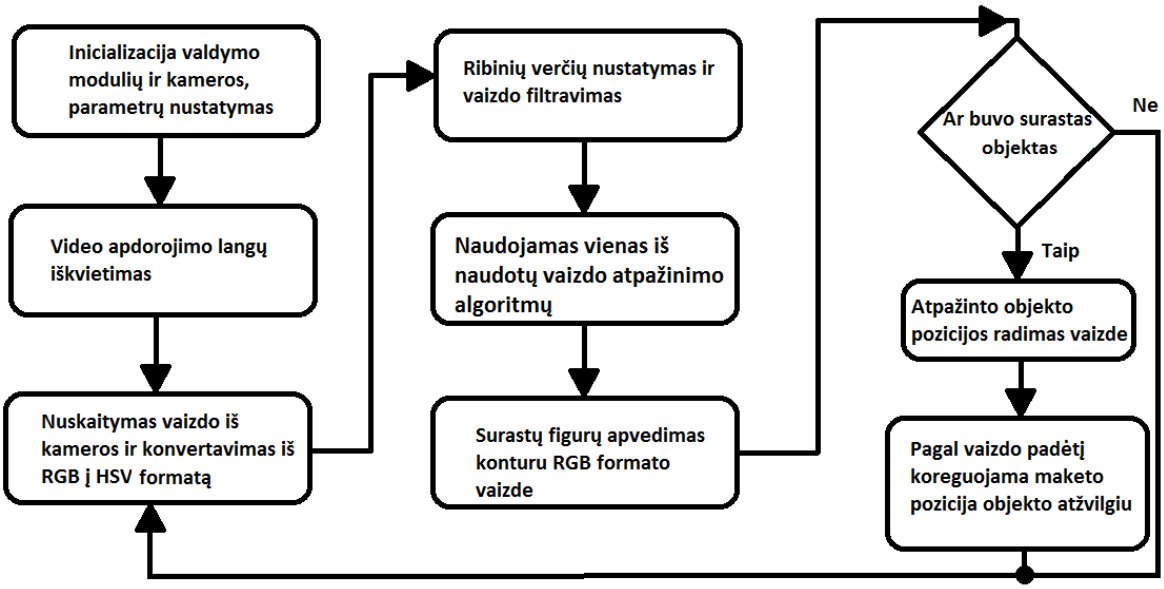


38 pav. Naktinio matymo testavimas:

- Iš viršutinio kairiojo krašto - kameros filmuotas vaizdas su išjungta naktinio matymo sistema, esant įjungtam aplinkos apšvietimui;
- Iš viršutinio dešiniojo krašto - kameros filmuotas vaizdas su įjungta naktinio matymo sistema, esant įjungtam aplinkos apšvietimui;
- Iš apatiniojo kairiojo krašto - kameros filmuotas vaizdas su išjungta naktinio matymo sistema, esant išjungtam aplinkos apšvietimui;
- Iš apatinio dešiniojo krašto - kameros filmuotas vaizdas su įjungta naktinio matymo sistema, esant išjungtam aplinkos apšvietimui.

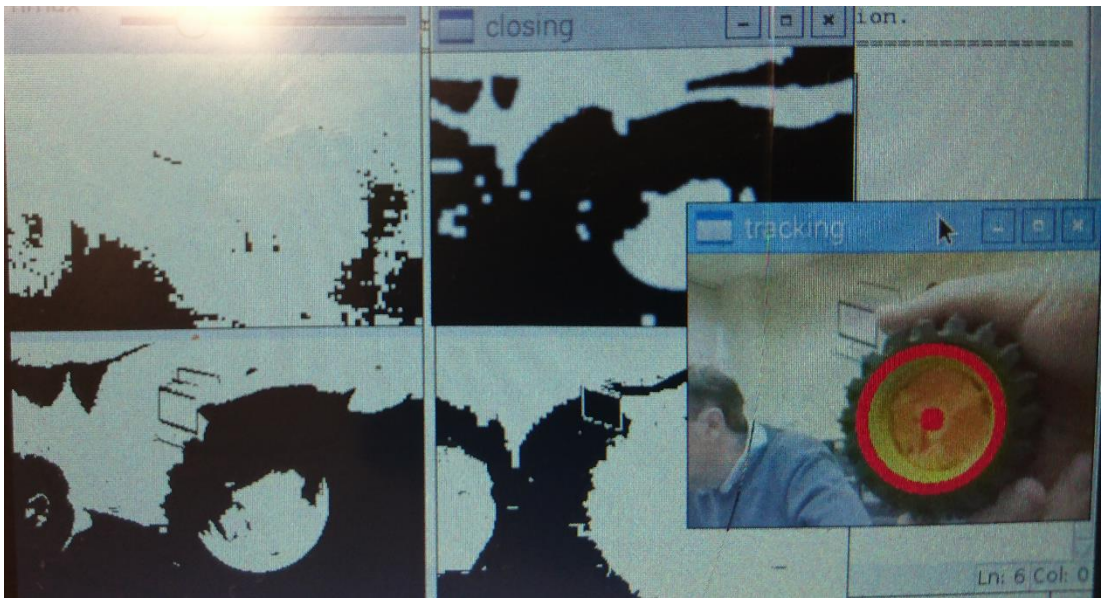
Pagaminus tyrimo maketą buvo išbandytas vaizdo atpažinimo algoritmas, kaip pasirodo, gana didelę įtaką turi aplinkos apšvietimas, nes HSV vaizdo filtravimo ir atpažinimo būdas gali klaidingai interpretuoti spalvas skirtingame apšvietime, bet naktinio matymo sistema padeda tai ištaisyti. Taip pat buvo išmatuotas kadrų apdorojimo užvėlinimo laikas ir vaizdo kadrų apdorojimo spartos priklausomybė nuo vaizdo rezoliucijos ir parinktas optimalus variantas – 240 x 160. Taip pat

buvo išmatuotas Raspberry Pi 2 mini kompiuterio energijos suvartojimas, nes tai yra aktualu mobiliems įrenginiams, besimaitinantiems nuo akumuliatorių.

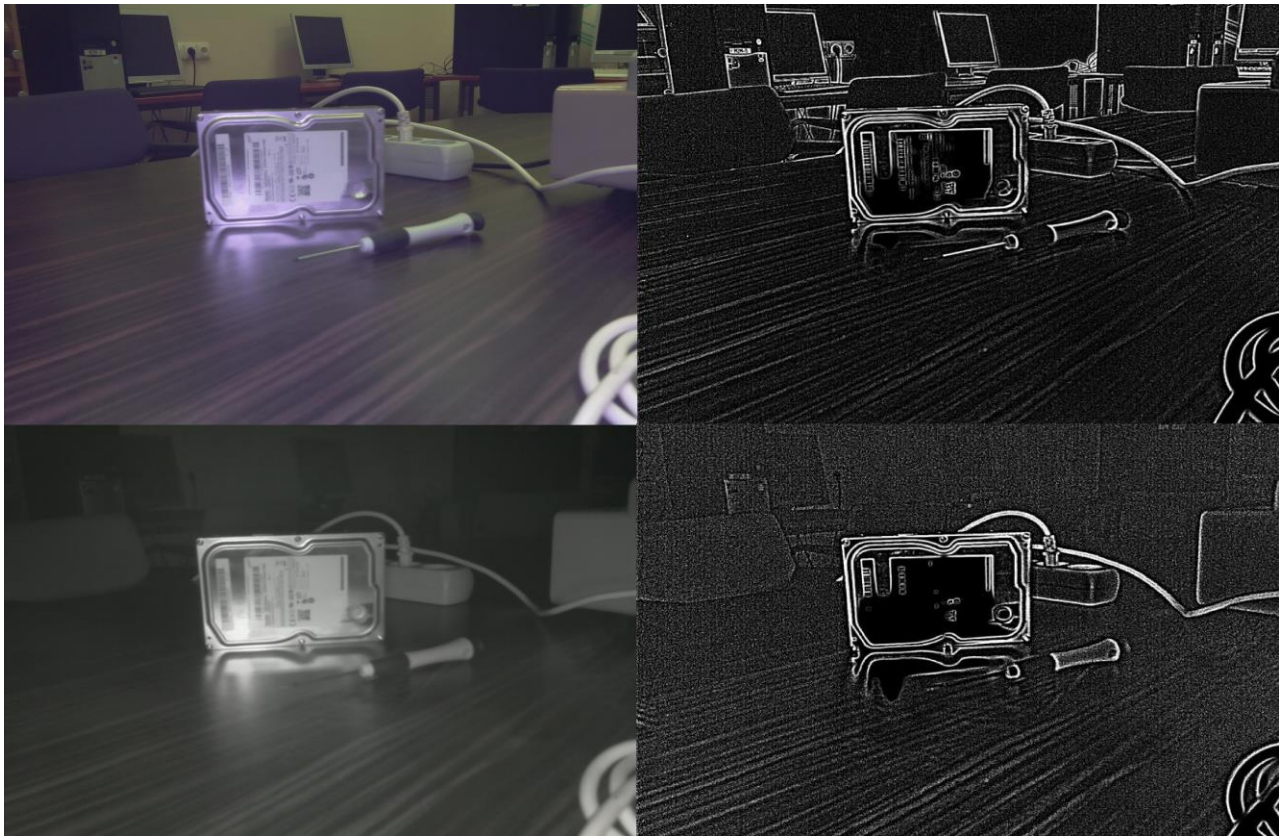


39 pav. Supaprastinta blokinė diagrama vaizdo atpažinimo programos

40 pav. Pirmi nesėkmingi vaizdo atpažinimo rezultatai (Raudona linija seka atpažinto objekto judėjimo padėtį)



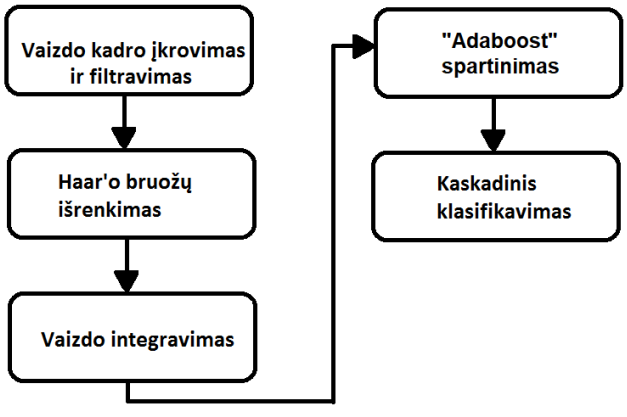
41 pav. Galutinis apskritiminės Hough‘o transformacijos vaizdo atpažinimo rezultatas su tarpiniais skaičiavimo vaizdais



42 pav. Vaizdo krašto atpažinimo algoritmo rezultatas su įjungtu apšvietimu ir įjungta naktinio matymo sistema (viršuje) ir vaizdo atpažinimo krašto ieškojimo algoritmo rezultatas su išjungtu apšvietimu ir įjungta naktinio matymo sistema (apačioje). Naudota maksimali Raspberry kameros raiška: 5 MP

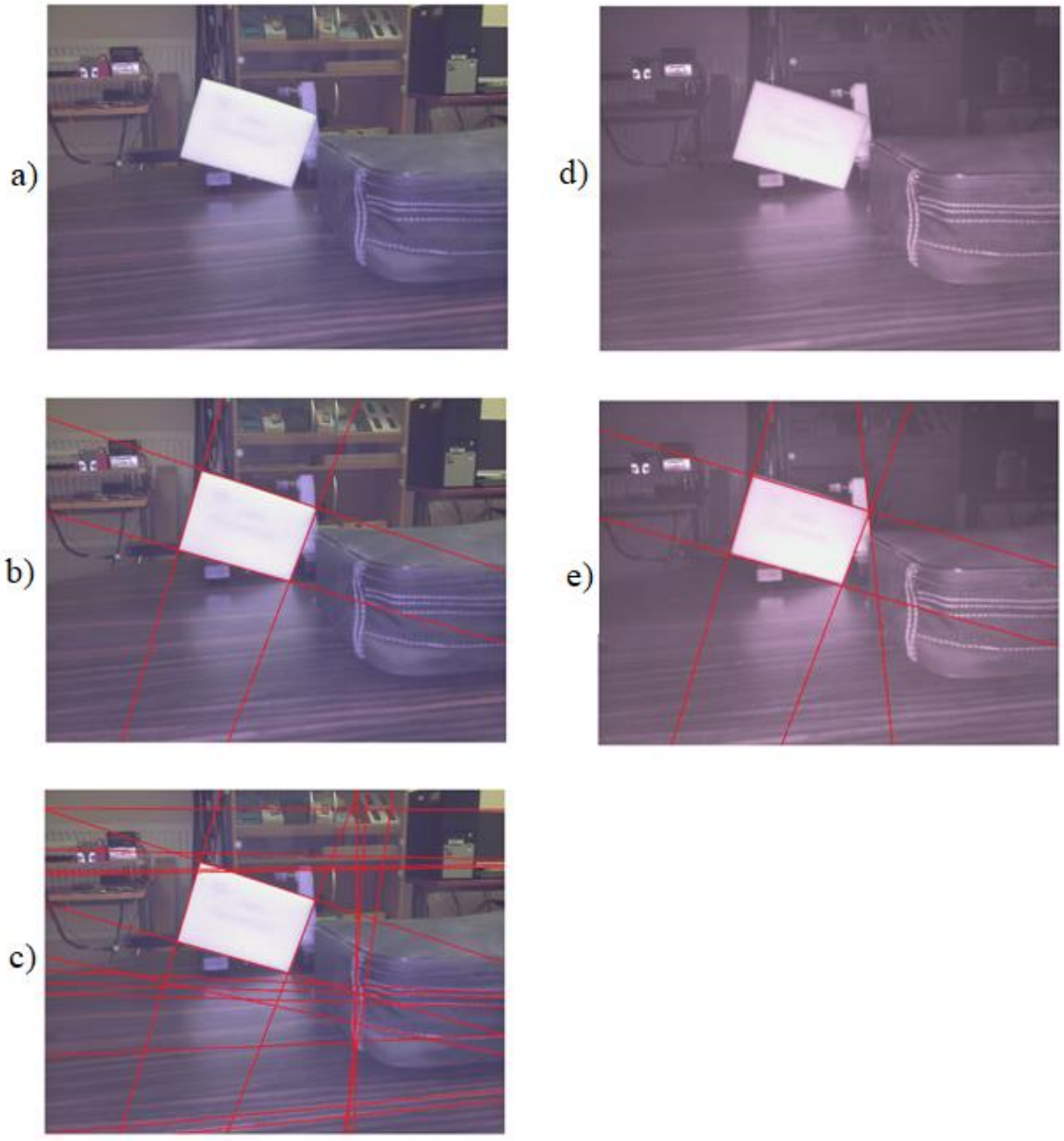
43 pav. Hough‘o apkritiminės transformacijos rezultatas su įjungtu apšvietimu ir įjungta naktinio matymo sistema (iš kairės) ir Hough‘o apkritiminės transformacijos rezultatas su išjungtu apšvietimu ir įjungta naktinio matymo sistema (iš kairės). Naudota maksimali Raspberry kameros raiška: 5 MP

44 pav. Veido atpažinimo neuroninio tinklo rezultatai: rezultatas su išjungtu apšvietimu ir įjungta naktinio matymo sistema (iš kairės veidas be akinių, iš dešinės - su akiniais). Naudota maksimali Raspberry kameros raiška: 5 MP



45 pav. Veido bruožų aptikimo algoritmo principinė diagrama

Kaip matome, todėl kad buvo naudotas bruožų šablonas, kuris buvo paremtas akių atpažinimu, vaizdo kadre su akiniais vaizdo atpažinimo algoritmas davė klaidingus rezultatus – ieškojo kito objekto labiausiai panašaus į akis. „AdaBoost“- adaptyvaus spartinimo algoritmas dar papildomai paspartina vaizdo apdorojimą, nes kiekvienoje apdorojamoje vaizdo iteracijoje su klasifikatoriais papildomai yra matuojama paklaidas, ir pagal ją yra nustatoma, kurių klasifikatorių įtakos galima nepaisyti ir juos atmesti, o taip pat, jeigu paklaida nurodo, kad klasifikatorius beveik aptiko bruožą, tai kitoje iteracijoje šis klasifikatorius „nuodugniau“ perskanuos vaizdo kadrą.



46 pav. Rezultatai krašto detekcijos algoritmo su Hough‘o transformacija tiesių linijų ieškojimui:
a) Hough‘o transformacijos pradinis įvesties vaizdas, kai yra įjungtas apšvietimas ir įjungta naktinio matymo sistema;

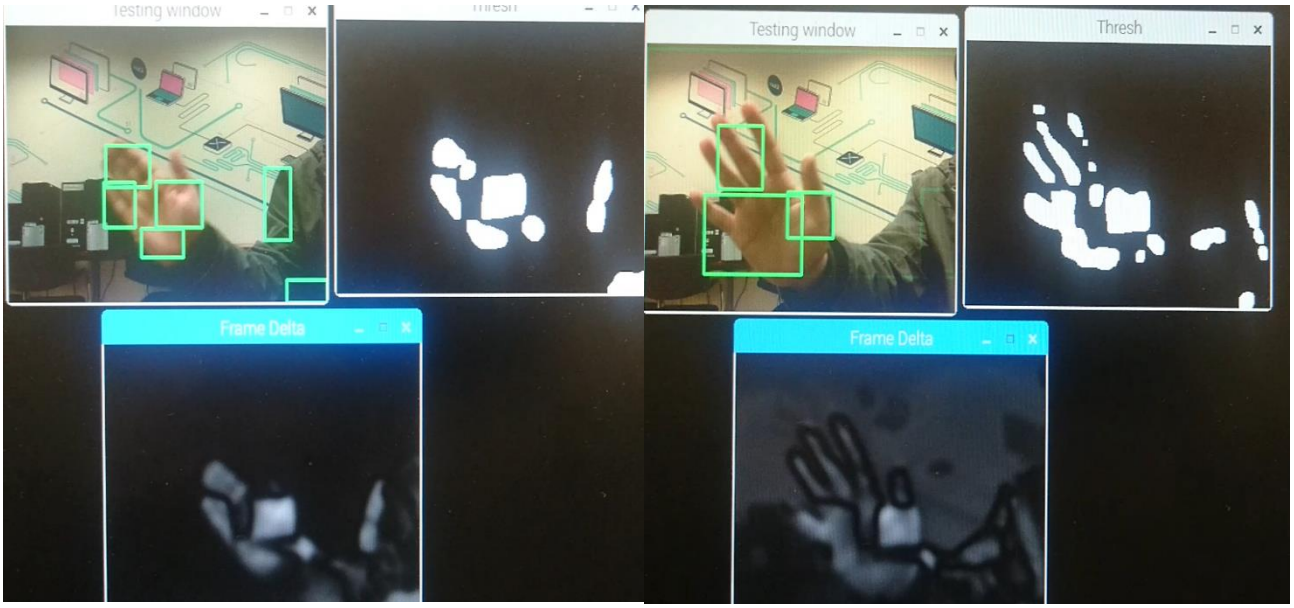
b) Hough‘o transformacijos rezultatas su tinkamai parinktais krašto detekcijos filtro koeficientais, kai yra įjungtas apšvietimas ir įjungta naktinio matymo sistema;
c) Hough‘o transformacijos rezultatas su netinkamai parinktais krašto detekcijos filtro koeficientais, kai yra įjungtas apšvietimas ir įjungta naktinio matymo sistema;
d) Hough‘o transformacijos pradinis įvesties vaizdas, kai yra išjungtas apšvietimas ir įjungta naktinio matymo sistema;
e) Hough‘o transformacijos rezultatas su tinkamai parinktais krašto detekcijos filtro koeficientais, kai yra išjungtas apšvietimas ir įjungta naktinio matymo sistema;

Taigi, jeigu parinksime geras kraštines sąlygas, tai ieškojimas vyksta gana sėkmingai, jeigu tik nebus šalia esančio kito objekto, kurio charakteristikos bus panašios, bet jeigu parametrai yra parinkti blogai, tai programa nedetektuos ieškomų objektų arba vietoje pusiau tinkamų objektų atmetimo, algoritmas gali juos pripažinti ieškomais objektais. Tai yra matoma 46 pav. „c“ punkte.

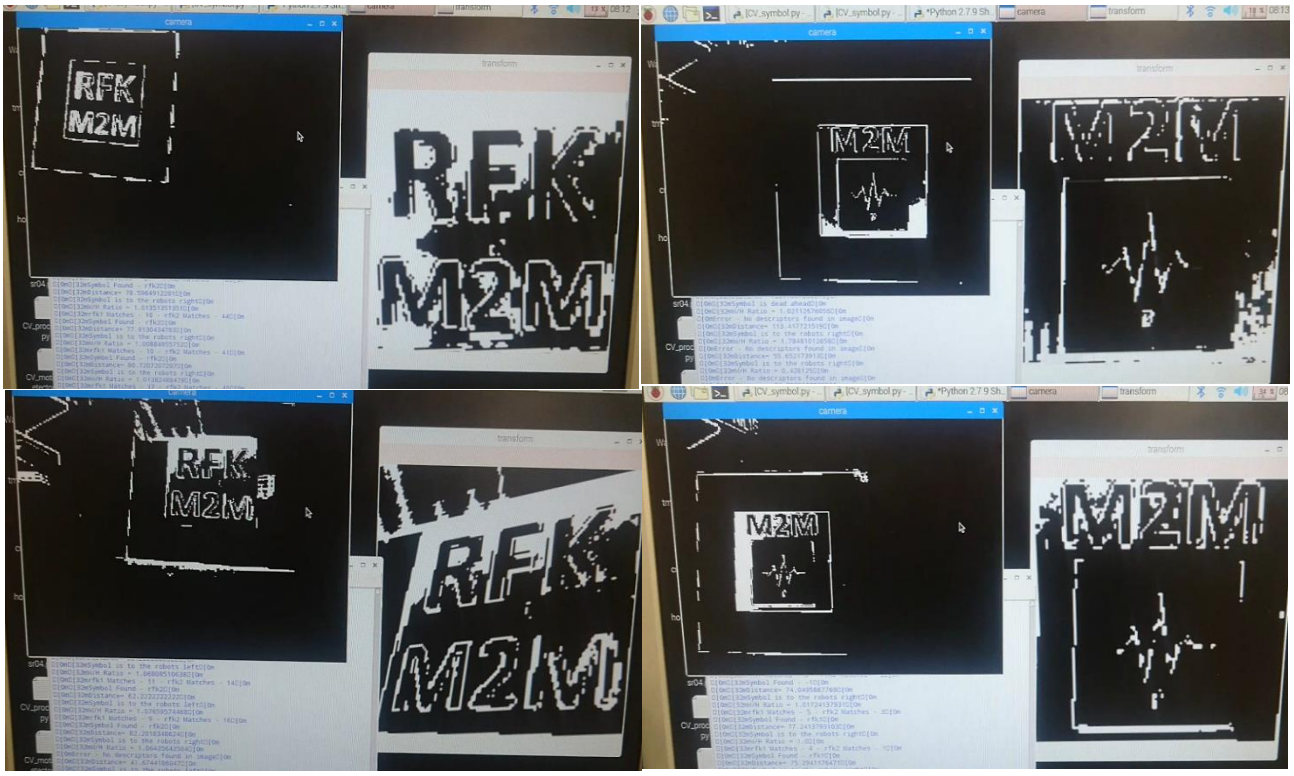
47 pav. Kliūčių išvengimo ir kelio atpažinimo programos rezultatai

Kliūčių išvengimo ir kelio ieškojimo programa daro prielaidą, kad vaizdo apačia visada yra paviršius, ant kurio stovi maketas. Tada nufiltravus vaizdą ir pakeitus iš RGB į HSV formatą yra ieškoma pirmo „kritinio“ pokyčio nuo apačios į viršų. Taip yra skanuojama kas kelis pikselius ir gauname tarsi stulpelius, kurių aukštis nurodo santikinį atstumą iki paviršiaus pabaigos ar kliūties

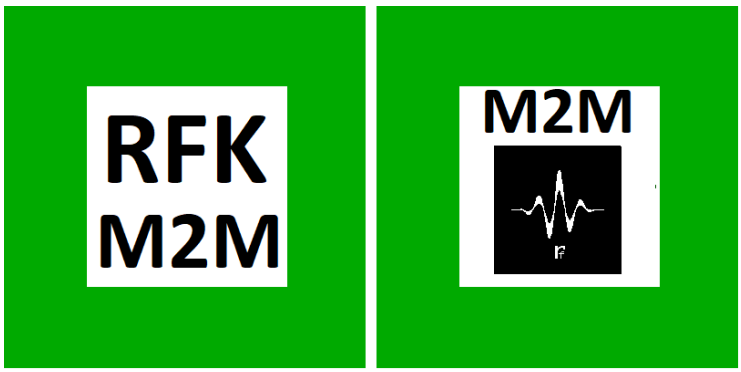
esančios ant jo. „Kritinės“ pokyčio sąlygos buvo nustatomos eksperimentiškai, kad programa atsižvelgtų į kelio paviršiaus spalvos ir apšvietimo netolygumus, ir tuo pačiu momentu atskirtų kitus objektus. Po to šios kraštinės sąlygos buvo koreguotos neuroninio tinklo.



48 pav. Judančių objektų ieškojimo programos rezultatai (Programa veikia optinio srauto principu)

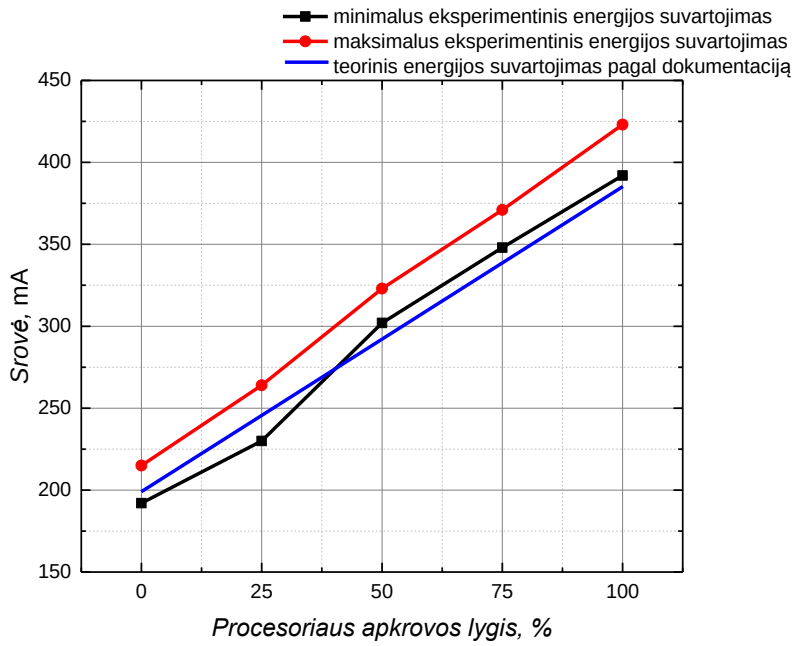


49 pav. SURF principu paremto vaizdo atpažinimo rezultatai



50 pav. SURF principu paremto vaizdo atpažinimo ieškomi objektai

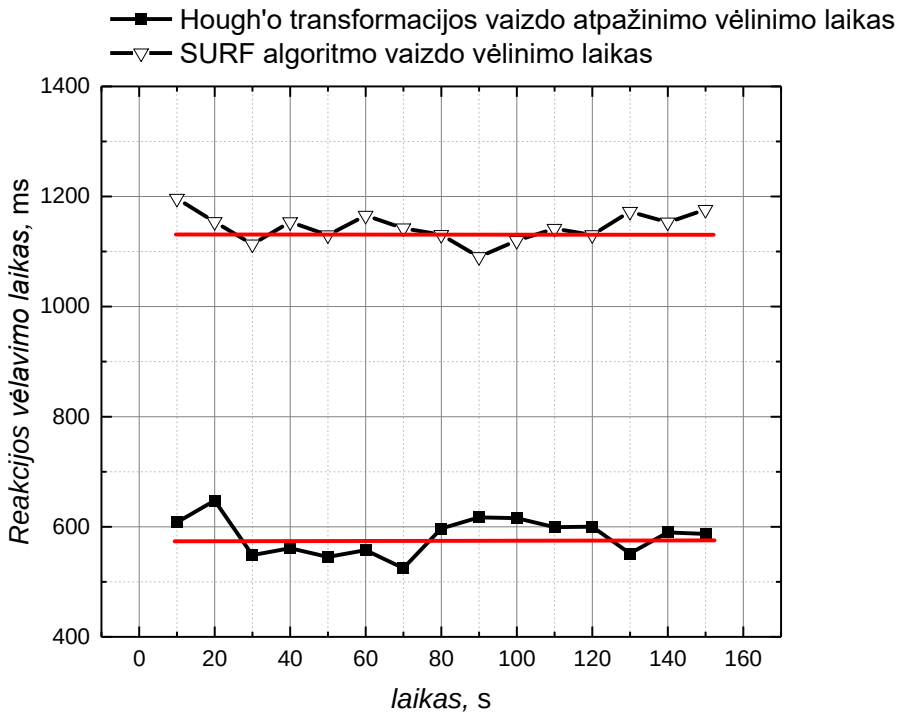
Surf pagrindu parašyta programa iš pradžių išrenka ryškius bruožus iš jau žinomų paveikslėlių ir naudoja juos jau ieškant atitikmens naujame video sraute. Kaip matome iš 49 pav., SURF algoritmas gali surasti ne tik neiškraipytą ieškomą objektą, bet ir palinkusį, blogai apšviestą arba net pasuktą kampu objektą. Nors atpažinimo būdas net savo pavadinime turi „Speeded-Up“ arba lietuviškai „paspartintą“ žodį, net su mažiausia įmanoma raiška, Raspberry Pi 2 mini kompiuteris negali apdoroti vaizdo realiu laiku.



51 pav. Raspberry Pi 2 energijos suvartojimo priklausomybė nuo procesoriaus apkrovos [32]

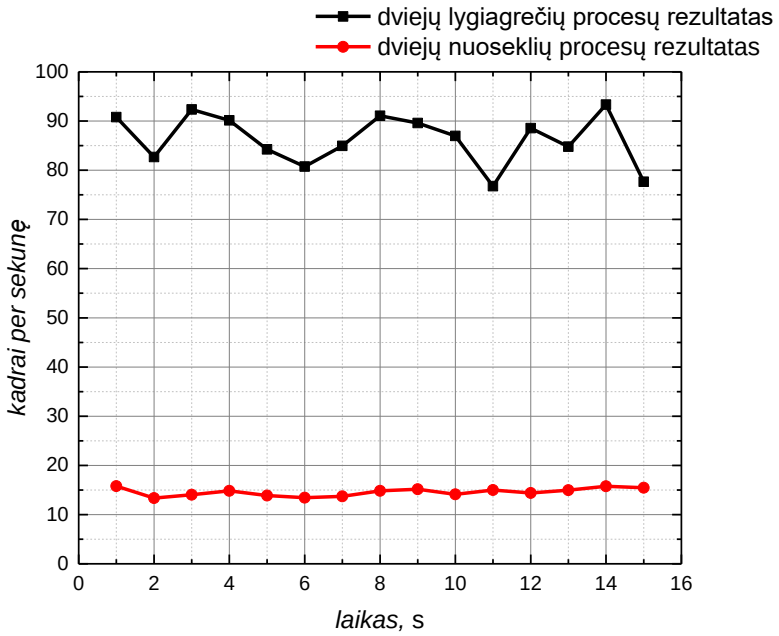
Energijos suvartojimas yra labai svarbus parametras mobiliuosiuose įrenginiuose, nes dažniausiai yra maitinami nuo elementų arba akumuliatorių. Todėl ne visada geriausia pasirinkti galingą įrenginį, o optimalų variantą su geru galios ir energijos suvartojimo koeficientu.

Vaizdo apdorojimo algoritmo reakcijos laikas yra išmatuotas ir pavaizduotas 52 paveikslėlyje ir jo vidurkis siekia beveik 570 ms. Žmogaus vidutinis reakcijos laikas yra apie 250 ms ir galima teigti, kad to pilnai užteks, bet sekant greitai judančius objektus tai yra per daug ir kartais objektas yra pametamas iš matymo zonos arba nerandamas, nes užfiksavus objektą pradedamas jo sekimas, bet realiu laiku jis spėjo pakeisti savo poziciją per 0,6 sekundės ir algoritmas bando ieškoti objekto klaidingoje vietoje. Pritaikant papildomai SURF algoritmo skaičiavimus ir pritaikant papildomus vaizdo apdorojimo filtrus vaizdo apdorojimo kokybei pagerinti, šis laikas beveik padvigubėja ir yra virš 1100 ms, taigi, Raspberry Pi 2 mini kompiuteris nelabai tinka vaizdo atpažinimui realiu laiku.

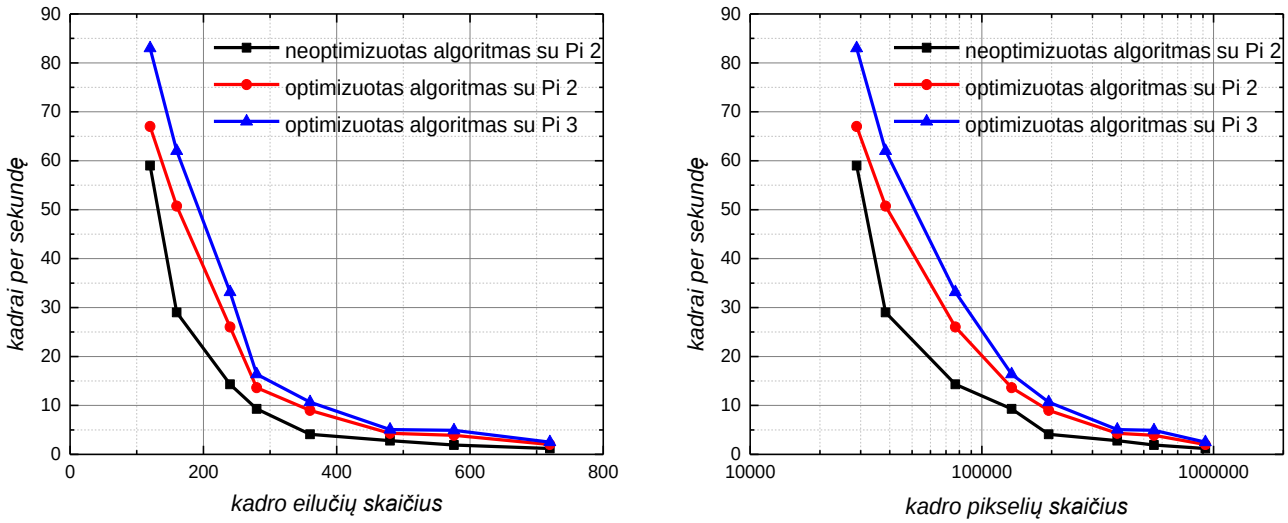


52 pav. Reakcijos laiko matavimas kas 10 sekundžių intervalais esant 240 x 160 raiškai

Iš gautų rezultatų (53 pav.) matome, kad nors du procesai pradėjo veikti lygiagrečiai, sparta padidėjo daug daugiau, nei du kartus. Šitą reiškinį galima paaiškinti tuo, kad procesai buvo papildomai stabdomi, kai veikė nuosekliai. Veikiant nuosekliai, laikas buvo gaištamas kiekvieną kartą inicializuojant vaizdo kamerą ir procesas turėjo laukti, kol prieš tai veikiantis procesas perduodavo jam duomenis. Lygiagrečiai veikiantys procesai per tam skirtas funkcijas ir kintamuosius visada gali prieiti prie kitų procesų rezultatų. Aišku, būna ir tokių atvejų, kai skaičiavimų eiliškumas yra svarbus ir tada nebus galima tikėtis tokio didelio spartos padidėjimo.



53 pav. Nuoseklaus ir lygiagretaus nuskaitymo iš kameros spartos priklausomybė nuo veikimo laiko (raiška 5 MP)



54 pav. Optimizuoto ir neoptimizuoto Hough‘o transformacijos vaizdo atpažinimo algoritmo spartos palyginimas naudojant Raspberry Pi 2 mini kompiuterį ir Raspberry Pi 3 mini kompiuterį

Kai programa veikia, yra sukuriami interaktyvūs langai su tarpiniais vaizdo apdorojimo langais: po atskirą langą atspalviui, sodrumui ir intensyvumo vertei. Tada veikimo metu galima koreguoti HSV užduotas vertes, kad kompensuoti apšvietimo iškraipymus arba tiesiog pakeisti ieškomo objekto spalvą. Kad paspartinti apdorojimo spartą ir efektyvumą reikia optimizuoti kodą ir atsisakyti testavimo tarpinių rezultatų langų, kuriuos aptarnauti yra eikvojami procesoriaus resursai.

Darbo pabaigoje buvo galimybė išmatuoti apdorojimo spartą ant neseniai išleisto naujo Raspberry Pi 3 mini kompiuterio. Sparta pakilo vidutiniškai apie 35%, kas yra gan arti gamintojo teigimo, kad Raspberry Pi 3 yra iki 50% spartesnis už Raspberry Pi 2. Bendrai Pi 2 ir Pi 3 modeliai yra labai panašūs ir veikia ant tos pačios Raspbian operacinės sistemos, o vieninteliai jų skirtumai yra [9]:

- Raspberry Pi 3 turi integruotą Bluetooth ir Wifi 802.11n standarto modulį;
- Vietoje keturių branduolių Cortex-A7 architektūros 32 bitų procesoriaus, turi keturių branduolių Cortex-A53 architektūros 64 bitų procesorių, kurių taktiniai dažniai yra atitinkamai 900 MHz ir 1200 MHz;
- Raspberry Pi 3 mini kompiuterio darbinė atmintis veikia didesniu dažniu.

Darbo eigoje buvo:

- Sukonstruotas maketas vaizdo atpažinimo testavimui;
- Išmatuota maksimali maketo suvartojama energija, kuri yra apie 38W;
- Susipažinta su Raspberry Pi veikimo principu ir Linux operacinės sistemos programavimu;
- Išmokta programuoti Python kalba;
- Susipažinta su OpenCV bibliotekų tūriniu;
- Susipažinta su neuroninio tinklo struktūra ir jo modeliavimu;
- Išbandyti ir ištirti skirtingi OpenCV bibliotekos vaizdo apdorojimo, filtravimo ir atpažinimo algoritmai;
- Ištirta Raspberry Pi 2 vaizdo apdorojimo sparta, užlaikymo parametrai bei energijos suvartojimas ir palyginta apdorojimo sparta su naujo Raspberry Pi 3 mini kompiuterio sparta;
- Išmatuota lygiagretaus ir nuoseklaus nuskaitymo iš kameros sparta;
- Sumodeliuotas neuroninis tinklas vaizdo atpažinimo programos kraštinių sąlygų nustatymui;
- Išmatuota neuroninio tinklo klaidingo rezultato gavimo tikimybės priklausomybė nuo apmokymo ciklų skaičiaus;
- Parašytos vaizdo apdorojimo ir atpažinimo programos su testavimo langais:
    - Optinio srauto judančių objektų ieškojimas;
    - Kliūčių ieškojimo ir kelio paviršiaus atpažinimo programa;
    - Veido atpažinimo programa;
    - SURF algoritmu besiremiantis vaizdo atpažinimas;
    - Hough‘o transformacijomis besiremiančios vaizdo atpažinimo programos.
- Parašyta programa lygiagrečiam veikimui su vaizdo atpažinimo programa, kuri yra skirta maketo mechaninių komponentų valdymui.

10. Išvados

Ištyrus vaizdo atpažinimą naudojant OpenCV bibliotekas ir neuroninius tinklus, Linux įterptinėse sistemose, buvo nustatyta, kad:

- Nuoseklaus vaizdo apdorojimo proceso reakcijos laikas yra apie 600 ms, o tai reiškia, kad greitai judantiems objektams, vaizdo apdorojimas ims trūkčioti ir veiks nestabiliai;
- HSV filtravimo vaizdo atpažinimo principas tinka vietose, kur yra tolygus apšvietimas;
- Neuroninio tinklo adaptyvumas ir galimybė mokytis labai praverčia vaizdo atpažinime, kai reikia ieškoti nevienodų objektų – pavyzdžiui: veidų paieška. Be to, tinklo veikimo tikslumas priklauso nuo apmokymo duomenų kiekio ir apmokymo ciklų skaičiaus - gali veikti sparčiai, bet ne taip tiksliai, arba lėčiau, bet tiksliau;
- Naktinio matymo sistema pašalina spalvų nuokrypius dėl šešėlių ir apšvietimo netolygumo, taip pat padeda išryškinti arti esamus objektus ir sumažina pašalinių, toli esančių objektų įtaką skaičiavimams;
- SURF algoritmas tinka sudėtingų objektų atpažinimui ir yra atsparus apšvietimo netolygumui, ieškomo objekto iškraipymams, bet jis netinka vaizdo atpažinimui realiu laiku dėl Raspberry Pi mini kompiuterio spartos trūkumo.

11. Santrauka

Albert Cesiul

Dirbtinių neuroninių tinklų taikymas vaizdo atpažinimui ir mechanizmų valdymui

Magistrantūros studijų baigiamasis darbas

Naujų technologijų kūrime gana didelę dalį užima vaizdo atpažinimo, apdorojimo technologijos. Apie 90% visos gaunamos informacijos iš išorės žmogus gauna regos pagalba, todėl akivaizdu, kad vaizdo atpažinimo integravimas elektronikoje bus labai plačiai vystomas. Naudojimo sritis yra gana didelė: pradedant saugumo įranga, baigiant autonominiais automobiliais, kuriais pradeda domėtis, kuo toliau, tuo daugiau, automobilių gamybos kompanijų. Mano darbo tikslas buvo parašyti vaizdo apdorojimo ir atpažinimo programą mini kompiuteriui Raspberry Pi 2, naudojantis OpenCV atviro kodo bibliotekomis, kuri gali atpažinti objektus, atpažinti kliūtį, sekti norimo objekto padėtį ir kontroliuoti kitų įrenginių darbą, vadovaujantis vaizdo atpažinimo rezultatais. Programos universalumui ir adaptyvumui padidinti skirtingose situacijose panaudotas dirbtinis neuroninis tinklas ir išbandyti skirtingi vaizdo atpažinimo modeliai: optinio srauto judančių objektų ieškojimas, kliūčių ieškojimo ir kelio paviršiaus atpažinimo programa, veido atpažinimo programa, SURF algoritmu besiremiantis vaizdo atpažinimas, Hough‘o transformacijomis besiremiančios vaizdo atpažinimo programos.

Darbo eigoje buvo nustatyta, kad nuoseklaus proceso reakcijos laikas yra apie 600 ms, kas reiškia, kad greitai judant objektams, vaizdo apdorojimas ims trūkčioti ir veiks nestabiliai. HSV filtravimo vaizdo atpažinimo principas tinka vietose, kur yra tolygus apšvietimas. Neuroninio tinklo adaptyvumas ir galimybė mokytis labai praverčia vaizdo atpažinime, kai reikia ieškoti nevienodų objektų – pavyzdžiui: veidų. Naktinio matymo sistema pašalina spalvų nuokrypius dėl šešėlių ir apšvietimo netolygumo. Naktinio matymo sistema padeda išryškinti arti esamus objektus ir sumažina pašalinių, toli esančių objektų įtaką skaičiavimams. Neuroninio tinklo vaizdo atpažinimo tikslumas priklauso nuo apmokymo duomenų kiekio ir apmokymo ciklų skaičiaus - gali veikti sparčiai, bet ne taip tiksliai, arba lėčiau, bet tiksliau. SURF algoritmas tinka sudėtingų objektų atpažinimui, bet jis Raspberry Pi 2 mini kompiuteriui netinka dėl spartos trūkumo. SURF algoritmas yra atsparus apšvietimo netolygumui ir ieškomi objektai gali būti atpažįstami, net kai yra iškraipyta jų formą.

12. Summary

Albert Cesiul

Application of Artificial Neural Networks for Image Recognition and Mechanisms Management in Embedded Linux Systems

Master's thesis

Image recognition and processing technology development takes a relatively large part of all new technology development processes. About 90% of all information received from surrounding environment a person receives as a visual information, so it is obvious that the integration of video recognition technology will be widely developed. The range of use is quite large, ranging from security equipment to the autonomous vehicles, which are becoming a huge interest of the car manufacturing companies. My aim was to write a program for image recognition using Raspberry Pi 2 mini-computer, which can recognize objects, detect an obstacle, to go to the desired position and to control mechanical devices according to visual recognition results. Artificial neural networks were used to increase the versatility of the program in different situations. Different image recognition models were tested: like moving objects detecting algorithm which uses optic flow principle, obstacle and road surface detection program, facial recognition software, SURF visual recognition program and image recognition programs with Hough transformation algorithms.

During the time of video recognition experiments it was found, that the response time of video recognition program with Hough transformation was in average 0,6 of a second, which means that fast moving objects would not be detected correctly and image processing will falter and will be unstable. Neural network adaptability and the ability to learn is very useful in a visual object recognition. For example, when you need to look for similar objects - such as faces. Night vision system eliminates colour deviations due to elimination of shadows and uneven lighting. Neural network image recognition accuracy depends on the amount of training data and the number of training cycles – it means it can run fast, but not as precise, or more slowly but more accurately. SURF algorithm is suited for complex object recognition, it is resistant to uneven lighting and can recognise even deformed and rotated objects, but the Raspberry Pi minicomputer is too slow to use it for a real-time video recognition.

13. Informacijos šaltiniai

[1] Nazmul Hossain, Mohammad Tanzir Kabir, Tarif Riyad Rahman, Mohamed Sajjad Hossen, Fahim Salauddin – „A Real-time Surveillance Mini-rover Based on OpenCV-Python-JAVA Using Raspberry Pi 2", Department of Electrical and Computer Engineering North South University Dhaka, Bangladesh 2012
Internetinė prieiga IEEE straipsnių duomenų bazėje:
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=07482232; paskutinio prisijungimo laikas: 2017-05-20.

[2] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, Mario Cifrek – „A brief introduction to OpenCV", Faculty of electrical engineering and computing, University of Zagreb, Zagreb, Croatia 2012
Internetinė prieiga IEEE straipsnių duomenų bazėje:
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber= 06240859; paskutinio prisijungimo laikas: 2017-05-20.

[3] Raspberry Pi mini kompiuterio dokumentacija:
https://www.raspberrypi.org/documentation/; paskutinio prisijungimo laikas: 2017-05-20.

[4] http://cdn03.androidauthority.net/wp-content/uploads/2013/02/Cortex-A7-vs-A15-power-performance.jpg; paskutinio prisijungimo laikas: 2017-05-20.

[5] http://i.stack.imgur.com/CeWPf.png; paskutinio prisijungimo laikas: 2017-05-20.

[6] Šarūnas Stanskis - "Neuroniniai Tinklai", Kauno Technologijos Universitetas, 2013

[7] https://tommygirard.files.wordpress.com/2016/12/alan-turing.jpg; paskutinio prisijungimo laikas: 2017-05-20.

[8] Neuroninių tinklų aprašymas ir vystymas :
http://singularityhub.com/2012/12/10/scientists-create-artificial-brain-with-2-3-million-simulated-neurons/; paskutinio prisijungimo laikas: 2017-05-20.

[9] A. Verikas, A. Gelžinis – „ Neuroniniai tinklai ir neuroniniai skaičiavimai", Kaunas 2008

[10] http://neuralnetworksanddeeplearning.com/images/tikz13.png; paskutinio prisijungimo laikas: 2017-05-20.

[11] Raspberry Pi įrangos parametrų aprašymas: https://www.raspberrypi.org/documentation/hardware/raspberrypi/; paskutinio prisijungimo laikas: 2017-05-20.

[12] http://cdni.wired.co.uk/1920x1280/a_c/25255898046_bf32f5aa51_k.jpg; paskutinio prisijungimo laikas: 2017-05-20.

[13] Vytautas Jonkus - „Mikrovaldikliai elektroninėse grandinėse“ 5 dalis – mokymo priemonė: http://rfk.ff.vu.lt/mikrovaldikliai/arm_ppt5.pdf; paskutinio prisijungimo laikas: 2017-05-20.

[14] Patricijus Cvizonas – „Procesai ir Gijos“ pristatymas: http://klevas.mif.vu.lt/~tomukas/Knygos/winapi/PG.ppt; paskutinio prisijungimo laikas: 2017-05-20.

[15] Python kalbos procesų aprašymas: https://docs.python.org/2/library/multiprocessing.html; paskutinio prisijungimo laikas: 2017-05-20.

[16] Vytautas Jonkus - „Mikrovaldikliai elektroninėse grandinėse“ 4 dalis – mokymo priemonė: http://rfk.ff.vu.lt/mikrovaldikliai/arm_ppt4.pdf; paskutinio prisijungimo laikas: 2017-05-20.

[17] http://www.pickey.es/images/Ejemplos-de-Duty-Cycle.gif; paskutinio prisijungimo laikas: 2017-05-20.

[18] Sebastian Wojas - „Metody przetwarzania obrazów z wykorzystaniem biblioteki OpenCV“, Krokuva 2010

[19] OpenCV bibliotekos naudojimosi mokymo priemonė: http://www.societyofrobots.com/programming_computer_vision_tutorial_pt2.shtml; paskutinio prisijungimo laikas: 2017-05-20.

[20] https://www.intorobotics.com/wp-content/uploads/2015/08/test-3-cvblob-with-static-image_opt.png; paskutinio prisijungimo laikas: 2017-05-20.

[21] http://boofcv.org/images/thumb/9/99/Object_contours.jpg/631px-Object_contours.jpg; paskutinio prisijungimo laikas: 2017-05-20.

[22] Hough‘o tiesių transformacijos aprašymas: http://scikit-image.org/docs/0.11.x/auto_examples/plot_line_hough_transform.html; paskutinio prisijungimo laikas: 2017-05-20.

[23] Vaizdo histogramų pavyzdžiai: http://www.pyimagesearch.com/2014/01/22/clever-girl-a-guide-to-utilizing-color-histograms-for-computer-vision-and-image-search-engines/; paskutinio prisijungimo laikas: 2017-05-20.

[24] http://www.pyimagesearch.com/wp-content/uploads/2014/01/flattened_color.png; paskutinio prisijungimo laikas: 2017-05-20.

[25] OpenCV bibliotekos aprašymas ir mokymo priemonė: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html; paskutinio prisijungimo laikas: 2017-05-20.

[26] http://opencv-python-tutroals.readthedocs.io/en/latest/_images/template_ccoeff_1.jpg; paskutinio prisijungimo laikas: 2017-05-20.

[27] http://www.imaco.pl/images/products/software/CVB/CVB-Optical-Flow-App1-I0.jpg; paskutinio prisijungimo laikas: 2017-05-20.

[28] https://i.ytimg.com/vi/V4r2HXGA8jw/hqdefault.jpg; paskutinio prisijungimo laikas: 2017-05-20.

[29] OpenCV bibliotekos mokymo priemonė veido atpažinimui: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html; paskutinio prisijungimo laikas: 2017-05-20.

[30] Piotr Guzik - „Metody wyszukiwania punktów charakterystycznych i wyznaczania ich cech”, Varšuva 2013/2014

[31] OpenCV integruotų neuroninio tinklo bibliotekų apžvalga: http://bytefish.de/blog/machine_learning_opencv/; paskutinio prisijungimo laikas: 2017-05-20.

[32] Raspberry Pi mini kompiuterių gamintojo energijos suvartojimo parametrai : http://raspberrypi.stackexchange.com/questions/43285/raspberry-pi-3-vs-pi-2-power-consumption-and-heat-dissipation; paskutinio prisijungimo laikas: 2017-05-20.