ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS FAKULTETAS
ELEKTRONIKOS KATEDRA

Ellanas Rokas Biliūnas

Lazerio atšvaito sekimo sistema, panaudojant lauku programuojamą loginę matricą

Magistro darbas

Vadovas
prof. dr. G. Daunys

ŠIAULIAI, 2013

ŠIAULIŲ UNIVERSITETAS
TECHNOLOGIJOS FAKULTETAS
ELEKTRONIKOS KATEDRA

TVIRTINU

Katedros vedėjas

prof. dr. G. Daunys

Lazerio atšvaito sekimo sistema, panaudojant lauku programuojamą loginę matricą

Magistro darbas

Vadovas

prof. dr. G. Daunys

2013-06

Atliko

RM-11 gr. stud. E. R. Biliūnas

2013-06

Recenzentas

doc. dr. N. Ramanauskas

2013-06

ŠIAULIAI, 2013

Biliūnas E.R. Laser trace tracking system using FPGA: Bachelor thesis of electronic engineering. Research advisor prof. dr. G. Daunys; Šiauliai University, Faculty of Technology, Department of Electronics. – Šiauliai, 2013. 53 p.

Summary

A first chapter deals with FPGA (field programmable gate arrays) structure. There are examined FPGA gates, input and output devices, an routing inside devices.

Further an analysis of Altera's NIOS II soft processor, which is obtained by uploading the NIOS II code into the FPGA, is done. Altera's NIOS II is a general-purpose programmable, 32-bit RISC processor optimized for programmable logic. To ensure high flexibility in design, it is available in three different processor cores, as: Nios II / f (fast); Nios II / s (standard); Nios II / e (economy). Also there are analyzed in more detail the inner NIOS II architecture: the registers of the arithmetic logic unit, interrupts, command bus, cache.

This is followed by an overview of image processing methods as gradient calculation, Roberts, Prewitt, Sobell, Canny operators for edge detection. The examples of how looks the images processed by different edge detection operators are provided.

In the second chapter an overview of similar studies, such as face detection and target position tracking by strip light projection in real-time for advanced automatic parking system are examined. Also there are analysed methods as application of small image processing pipelines, contrast and brightness adjustment approach, the contour finding the image.

The third section contains the description of the reflection tracking method. There was used an image processing by company Altera Cyclone II FPGA family. The images were processed by two ways: hardware and software. The pipelines for image processing by hardware were designed by "Quartus" software package tool "Qsys". For the image processing by software it was used NIOS II processor, which is implemented in FPGA. It was obtained that 640x480 pixel image processing takes 638ms using software method and 6.2 ms using hardware method.

Turinys

Paveikslai

Įvadas

Regėjimas žmogui yra vienas iš stipriausių ir informatyviausių pojūčių lyginant su kitais. Žmogaus akis sugeba pajusti elektromagnetinių bangų spektrą , kurių bangos ilgis prasideda nuo 390nm iki 780nm[1]. Paminėtasis elektromagnetinių bandų ruožas dar vadinamas regimoji šviesa. Nors regimosios šviesos ruožas ir yra labai siauras, lyginant su visu elektromagnetinių bangų spektru, kuris apima nuo 0.0001 nm iki 100 m[2], žmogui tokios regos visiškai pakanka, kad jis galėtu kaupti informaciją apie jį supančią aplinką ir tai suteikia galimybe mokytis iš to ką matome[3].

Mašininės regos sistemos yra pritaikomos visur kur reikia apdoroti esamą vaizdą išskiriant iš vaizdo būdingus bruožus: foto akies tašką, spalvą, objekto kraštus ir t.t. Tokios sistemos gali netik atpažinti įvairius vaizdo bruožus, tačiau ir sekti vaizdo taškų trajektorijas.

Paprastai vaizdų apdorojimo metodai atliekami su centriniu procesoriumi (*angl. CPU – Central Processing Unit*), tačiau rinkoje pasirodžius grafinams procesoriams (*angl. – GPU Graphics Processor Unit*), vaizdo apdorojimas tapo greitesnis lyginant su CPU. Esminis skirtumas, kodėl greičiau apdorojamas su GPU nei su CPU, tai lygiagreti GPU architektūra, su kuria galima vykdyti net iki kelių tūkstančių operacijų vienu metu, lyginant su dešimtimis ar vienetais tokių pat operacijų su CPU[4].

Bet visiems gyvenimo atvejams per brangu gaminti specializuotus lustus. Dėlto buvo sukurtos LPLM – lauku programuojamos loginės matricos (angl. FPGA – *Field programmable gate array*).

Tai lustas, kuriame suformuoti bendros paskirties loginiai elementai – galinčios atlikti elementarias logines operacijas, tokios kaip ARBA, IR , NE ir pan.[5]. Dėl šios priežasties vartotojas LPLM gali konfigūruoti pagal uždavinio sudėtingumą, taip pat yra galimybė tobulinti anksčiau kitų vartotojų sukurtas sistemas.

Darbo tikslas: Ištirti šviesos atspindžio koordinačių kameros matymo lauke skaičiavimo, panaudojant lauku programuojamas logines matricas, algoritmų spartą.

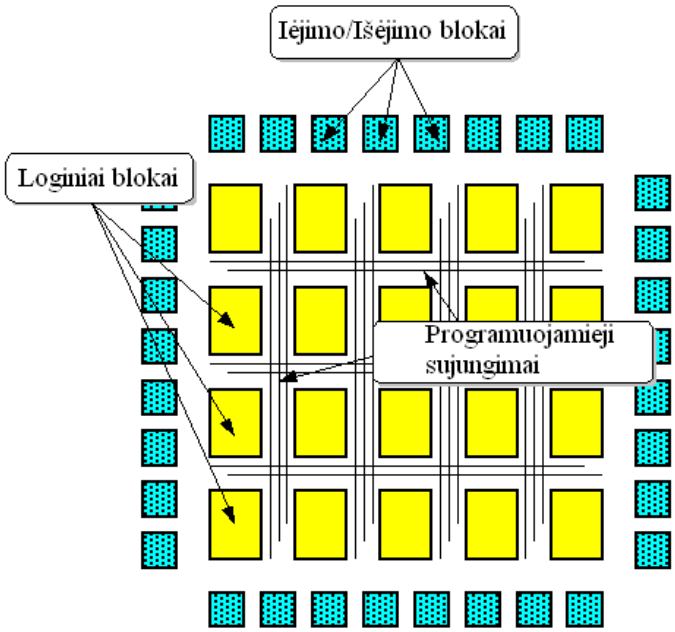
Darbo uždaviniai:

1. Susipažinti su „Altera“ lauku programuojamomis loginėmis matricomis (LPLM, angl. FPGA);
2. Išanalizuoti vaizdų apdorojimo metodus (kraštų atpažinimą, formos atpažinimą, segmentaciją);
3. Parašyti ir ištestuoti programinę įrangą LPLM (angl. FPGA) lustui;
4. Atlikti vykdymo trukmės tyrimus.

1. Mašininės regos technologijų taikomų metodų apžvalga

1.1 Lauku programuojamos loginės matricos (LPLM, angl. FPGA)

Lauku programuojamos loginės matricos LPLM (angl. Field Programable Gate Array – FPGA) – tai viena iš įterptinių sistemų rūšių. LPLM pranašumas lyginant su dauguma kitokios struktūros elektrinių grandynų yra jų pritaikomumas ir universalumas kuris gaunamas išskirtine realizuojama architektūra. LPLM turi programuojamą struktūra . Šios struktūros branduolį sudaro: programuojami sujungimai, loginiai blokai, įvesties/išvesties blokai (1.1 paveikslas)[6].

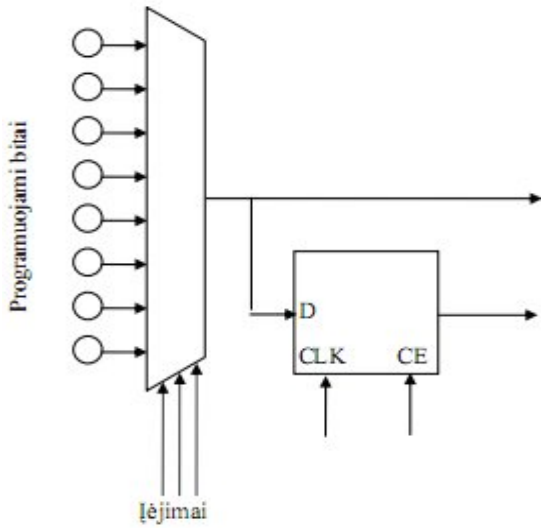


1.1 pav. Lauku programuojamos loginės matricos struktūra [6]

Programuojant specialia programavimo kalba, blokų sujungiami yra realizuojami fiziniais sujungimais. Toks programavimo būdas realizuojamas aparatinės įrangos aprašymo kalba (angl. hardware description language (HDL)). Pagrindinės tokio programavimo kalbos yra VHDL ir Verilog[7].

Mažiausias LPLM bendras vienetas yra loginė ląstelė, turinti vieno bito išėjimą. Kurią sudaro maža peržvalgos lentelė (angl. look-up table arba LUT) su išėjimo trigeriu. Loginės ląstelės struktūra pavaizduota 1.2 paveiksle. Peržvalgos lentelės įėjimų skaičius nulemia maksimalų skirtingų loginių funkcijų kiekį, kurias galima realizuoti šia peržvalgos lentele. Peržvalgos lentelė gali būti pažymėta trumpiniu n-LUT, čia n yra įėjimų skaičius. Pvz., 4-LUT žymi peržvalgos lentelę su keturiais įėjimais. Trigeris išėjime reikalingas išėjimo reikšmei įsiminti[13].

Įvesties ir išvesties (trumpai Į/I) sąsajų blokai leidžia sujungti LPLM vidinius blokus su išoriniais išvadais. Be šių pagrindinių komponentų, dauguma LPLM turi valdomą taktinių impulsų paskirstymo tinklą, kuriuo išorinio generatoriaus taktiniai impulsai perduodami LPLM vidiniams įtaisams. Taktinių impulsų paskirstymo tinklas turi sinchroniškai tiekti impulsus visoms LPLM dalims.



1.2 pav. LPLM tipinė loginė ląstelė[13]

Programuojamų sujungimų tikslas yra sukurti loginių blokų lankstaus sujungimo galimybę, kuri leistų pasiekti norimą funkcionalumą. Akivaizdu, kad netikslinga turėti tiesiogines jungtis iš visų įmanomų išėjimų į visus įmanomas įėjimus. Kiekvienu konkrečiu atveju tik labai nedidelė šių jungčių dalis būtų panaudota. Tačiau, kiekviena iš šių jungčių esant tam tikroms aplinkybėms gali būti reikalinga. Sprendimas – optimaliai sumažinti jungčių (trasavimo linijų) skaičių, kurias būtų galima efektyviau panaudoti, kuriant reikalingas jungtis. Nes trasavimo linijos užima silicio plotą, todėl reikia ieškoti kompromiso. Sumažinus trasavimo linijų skaičių, atsiranda galimybė daugiau silicio ploto skirti loginiams blokams, tačiau sudėtingi įtaisai negali būti sukurti dėl trasavimo

išteklių trūkumo. Jeigu LPLM įtaise yra per daug trasavimo linijų, jos nepanaudojamos, o loginių blokų skaičius sumažėja. Sprendimas yra maždaug ties šių kraštutinumų viduriu[13].

Daugelio LPLM šeimų sujungimų tinklas turi tinklelio struktūrą, panašią į parodytą 1.1 paveiksle. Kiekviename tinklelio mazge galima užprogramuoti ryšį tarp horizontaliosios ir vertikaliosios trasavimo linijų. Kitas veiksnys, į kurį reikia atsižvelgti, yra sujungimų tinklo delsa. Kiekvienas jungiklis, per kurį sklinda signalas, sukuria papildomą delsą. Todėl buvo prieita prie segmentinės struktūros naudojimo, kai trasavimo linija sujungiama kiekvieno susikirtimo vietoje. Taip pat tarp gretimų loginių blokų gali būti padarytos specialios tiesioginės jungtys, kad būtų sumažintas skaičius signalų, kuriuos reikia perduoti per sujungimų matricą. Projektuojant sujungimus svarbu užtikrinti, kad tik vienas išėjimas būtų prijungtas prie trasavimo linijos. Jei prijungti du išėjimai, iš kurių vienas turi aukštą, o kitas – žemą loginį lygį, per trasavimo liniją tekės trumpojo jungimo elektros srovė ir LPLM lustas gali būti sugadintas. Todėl LPLM programavimo priemonės turi užtikrinti, kad to niekada neįvyktų[14].

„Altera“ šiuo metu gamina LPLM integrinius grandynus, kurie priklauso trims šeimoms: Cyclone, Arria, Stratix. Cyclone šeimai priklauso pigūs, Arria – vidutinės klasės, Stratix – aukštos kokybės LPLM integriniai grandynai. Pradžioje Stratix turėjo tą pačią loginę architektūrą kaip ir Cyclone šeima. Į II serijos Stratix buvo įdiegti naujos struktūros prisitaikantys loginiai moduliai. Šios šeimos neturi procesoriaus branduolio. Jį turėjo Excalibur LPLM, priklausančios jau negaminamai Apex20K šeimai. Aparatūrinio procesoriaus nebuvimas yra kompensuojamas įkeliamaisiais procesoriais NIOS.

1.2 Altera NIOS II programinis procesorius

Altera NIOS II yra programinis bendros paskirties, 32 bitų RISC procesorius optimizuotas programuojama logika. Užtikrinant dideli projektavimo lankstumą, galimi trys skirtingi procesoriaus branduoliai, tai:

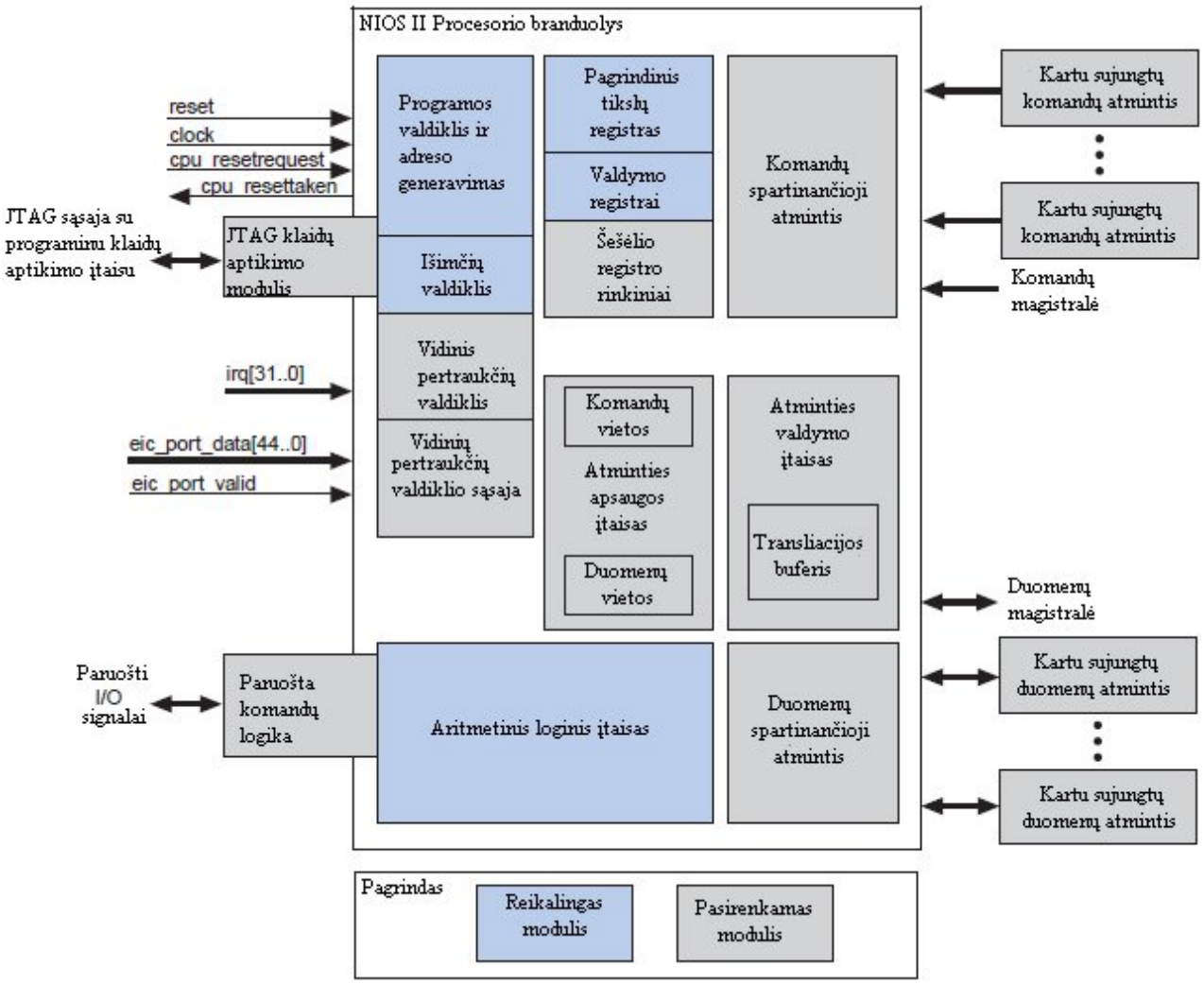
- Nios II/f(angl.fast – greitas);
- Nios II/s (angl. standard – standartinis);
- Nios II/e (angl. economy – ekonomiškas)

Suformuotas branduolių užimamas plotas svyruoja apie 3 kartus, o našumas apie 9 kartus[8].

Procesoriaus architektūra apibudina jo komandų rinkinio architektūra. Ši architektūra yra funkcinių elementų rinkinys, kuris įgyvendina komandas ir palaiko tokius įrenginius[9]:

- Registrus
- Aritmetinį loginį įtaisą
- Sąsają su paruošta komandų logika
- Išimčių valdiklį
- Vidinį ir išorinį pertraukčių valdiklį
- Komandų magistralę
- Duomenų magistralę
- Atminties valdymo įtaisą
- Atminties apsaugos įtaisą
- Komandų ir duomenų spartinančiąsias atmintines
- Kartu sujungtas atminties sąsajas komandoms ir duomenims
- JTAG klaidų aptikimo modulį

1.3 pav. pateiktas NIOS II procesorius su jo viduje esančiais įrenginiais[9].



1.3 pav. Nios II procesoriaus struktūrinė schema[9]

- **Procesoriaus registrai(angl. Register file)**

NIOS II procesoriaus architektūra palaiko 32 bitų bendrosios paskirties registrus ir 32 bitų kontrolės valdymo registrus. NIOS II procesorius pasirinktinai gali turėti vieną ar daugiau šešėlio registro rinkinių. Šešėlio registro rinkinys yra bendrosios paskirties registrų rinkinys. Kai šešėlio registrų rinkiniai realizuojami, atsiranda galimybė pamatyti kurie šiuo metu bendros paskirties registrai yra aktyvūs. Dažniausiai, šešėlio registrai naudojami, kad paspartinti perjungimus, nes yra specialių komandų, kurios gali perkelti duomenims iš vieno registrų rinkinio į kitus. NIOS II procesorius gali turėti iki 63 tokių registrų rinkinių[9].

- **Aritmetinis loginis įtaisas(angl. ALU – Arithmetic Logic Unit)**

NIOS II (ALĮ) – aritmetinis loginis įtaisas kuris dirba su bendrosios paskirties registruose saugomais duomenimis. ALĮ atlieka įvairias aritmetines operacijas, tokias kaip: (sudėtį, atimtį, dalybą, daugybą), racionaliąsias (daugiau, mažiau, nelygu ir t.t.) ir logines (NE, ARBA ir t.t), postūmio(kaire,dešinę) – tai atitinka (daugybą, dalybą), pasukimo. Sukurti kitokia operacija galima kombinuojant išvardintas operacijas[9].

- **Slankiojo kablelio komandos(angl. Floating-Point Instructions)**

NIOS II architektūra palaiko vienetinio tikslumo slankaus kablelio instrukcijas nurodytas IEEE STD 754-1985. Kad vartotojas galėtu naudotis slankiojo kablelio instrukcijomis pirmiausia jis turi į LPLM įkelti slankiojo kablelio funkcinį bloką. Tai galima padaryti su „Qsys“ arba „SOPC builder“, tai „Quoartus“ įrankiai kurie padeda projektuoti vidine LPLM struktūra. Vartotojas gali pridėti pasirinktas slankaus kablelio instrukcijas, tačiau slankaus kablelio naudojamos instrukcijos sunaudoja daugiau procesoriaus resursų, lyginant su kitomis instrukcijomis. Procesorius programuojamas C programavimo kalba, kuri bibliotekoje turi slankiojo kablelio instrukcijas. Norint naudotis slankiojo kablelio operacijomis reikia sukompiliuoti C programos kodą ir gautą kodą įkrauti į procesorių[9].

- **Išimčių ir pertraukčių valdiklis(angl. Exception and Interrupt Controllers)**

NIOS II architektūra palaiko 32 vidines aparatūrines pertrauktis. Procesoriaus branduolys turi 32 skirtingus pertraukties įėjimo identifikatorius. Pertrauktys programiškai suskirstomos pagal prioritetus. Taip pat programiškai galima įjungti arba išjungti pasirinktas pertrauktis per „ angl.PIE status control“ registrus

Vartotojo patogumui adresai yra nurodyti NIOS II procesoriaus „Qsys“ ir „SOPC builder“ redaktorių parametruose.

Pertraukčių sąsaja (angl. EIC – Exception Interface Controller) užtikrina aukštos kokybės aparatinę pertrauktį, taip sutrumpinant programos pertraukimų vėlinimą. Pertraukčių sąsaja paprastai naudojama kartu su šešėlių registrų rinkiniais.[9].

- **Komandų ir duomenų magistralės(angl. Instruction and Data Buses)**

NIOS II architektūra palaiko atskirą instrukcijų ir duomenų sąsają, kuri klasifikuojama kaip „Harvard“. Instrukcijų ir duomenų magistralės įgyvendinta kaip Avalon – MM valdantysis prievadas (angl. master).

NIOS II architektūra suteikia prieigą prie priskirtos I / O atminties.

NIOS II architektūra naudoja duomenų saugojimą "little-endian" tvarka, kai vyresnysis baitas užima aukštesnį adresa.

Procesoriaus duomenų valdantysis prievadas atlieka dvi funkcijas:

1. Skaito duomenis iš atminties arba periferinio įrenginio, kai procesorius atlieka įkėlimo komandą.
2. Rašo duomenis į atmintį arba į periferinį įrenginį, kai procesorius vykdo arba talpina komandą[9].

- **Spartinančioji atmintinė(angl. Cache Memory)**

Procesoriaus architektūra palaiko spartinančiąją atmintį (angl. Cache memory). Tai nedidelės talpos labai sparti atmintis, kurioje saugomi ypač dažnai naudojami pagrindinės atminties fragmentai. Spartinančiosios atminties pagrindinė funkcija sutrumpinti keitimosi duomenimis laiką tarp procesoriaus ir dinaminės atminties. Procesorius gali dirbti ir be spartinančiosios atminties, tačiau jai procesoriui tenka atlikinėti skaičiavimus sistemos darbas sulėtėja. [9].

- **JTAG klaidų aptikimo modulis**

Architektūra palaiko JTAG klaidų aptikimo modulį, kuris atlieka vidinę emuliaciją ir valdo procesorių atskirai nuo pagrindinio kompiuterio. Programinis procesorius NIOS II siūlo patogias derinimo galimybes. Programos derinimo metu į procesorių galima įkrauti derinimo modulį(angl. debug), atlikus programos testavimą ir koregavimą derinimo nodulį galima pašalinti iš procesoriaus.

Klaidų aptikimo modulis atlieka šias funkcijas:

- Realiu laiku fiksuoja procesoriaus įvykdytas instrukcijų sekas;
- Įrašo programą į atmintį;
- Nustato pertraukties taškus;
- Saugomi vykdymo kelio duomenys į lusto atmintinę[9].

1.3 Vaizdų apdorojimo metodikos

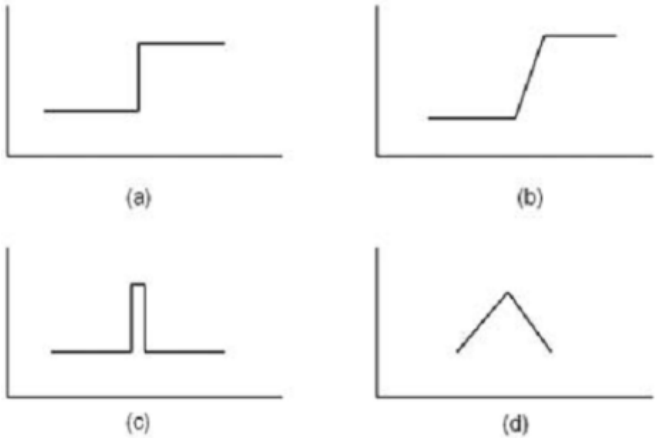
Skaitmeninių vaizdų apdorojime norint išskirti iš vaizdų dominančias detalias ar kontūrus dažniausiai naudojama kraštų radimo technika arba filtravimas pagal nustatytą slenkstinę vertę.

Segmentacija yra viena iš priemonių išskirti iš vaizdo reikalingus elementus. Yra įvairių segmentavimo būdų kurie priklauso nuo pikselių intensyvumo vertės, netolygumų. Briaunų vietos intensyvumo pokyčio atvaizde. Vaizdų segmentavimo procesas – vaizdo skaidymas į reikšmingas dalis, taip iš vaizdo gaunant dar papildomai domenų. Jeigu tiriamajame vaizde yra du ar daugiau

objektų, tai tikrai bus riba (kraštai) skirianti objektus. 1.5 paveiksle pavaizduota kaip atrodo vaizdas apdorotas skirtingais kraštų radimo operatoriais.

Yra įvairių kraštų perėjimo variantų tarp objektų kraštų[10],[11]:

- ***Žingsninis krašto modelis:*** jei vaizdo intensyvumas staigiai pasikeičia iš vienos reikšmės vienos pusės į kitą reikšmę kitos pusės, tada jis yra laikomas žingsninio krašto modeliu 1.4 paveikslas a);
- ***Nuožulnaus krašto modelis:*** tai ypatingas žingsninio krašto modelio atvejis, kai intensyvumo pokytis keičiasi ne staiga, o palaipsniui baigtiniame verčių intervale. 1.4 paveikslas b);
- ***Šuolinis krašto modelis:*** vaizdo intensyvumo vertė staiga pasikeičia, tačiau per trumpą laiką vėl grįžta į pradinio intensyvumo vertę. 1.4 paveikslas c);
- ***Pjūklinio krašto modelis:*** tai ypatingas atvejis šuolinio krašto modelio atvejis, kai šviesos intensyvumo pokytis yra ne momentinis, o įvyksta palaipsniui. 1.4 paveikslas d).



1.4pav. Kraštų rūšys: a) žingsninis, b) nuožulnus, c) šuolinis, d) pjūklinis[10]

Kraštų nustatymas yra būtinas žingsnis vaizdų apdorojime, nes kraštai vaizde yra riba tarp dviejų objektų, arba objekto ir fono. Pagal nustatytus kraštus vaizde galima išskirti reikiamus objekto segmentus. Ieškant vaizde kraštus reikia atlikti šiuos pagrindinius žingsnius[10]:

- Vaizdas yra glotninamas, taip sumažinant triukšmą vaizde;
- Kraštų išrinkimo operacija, išrenkami vaizde visi įmanomi kraštai;
- Kraštų lokalizacija, išrenkami reikalingi kraštai, pagal parinktą krašto radimo slenkstį.

Gradientas – vektorius, kurio skaitinė reikšmė ir kryptis apibūdina didžiausią skaliarinio dydžio kitimo greitį. Gradientas randamas pagal (1.1) formulę:

$$\nabla f = G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (1.1)$$

čia f(x,y) – pradinis vaizdas.

Roberts kraštų radimo operatorius:

Roberts operatorius naudojamas rasti diagonalinius kraštus ir sudėjus atstojamąsias randami kraštai.Šis operatorius pabrėžia erdvės aukščiausias vertes, kurios atitinka kraštų regionus, taip vaizdui suteikiant pilkus pustonius. Žemiau pateikiama (1.2) formulė kaip rasti gradiento dydį[10].

$$[f(x,y)] = |f(x,y) - f(x+1,y+1)| + |f(x+1,y) - f(x,y+1)| = |G_x| + |G_y| \quad (1.2)$$

čia f(x,y) – pradinis vaizdas.

Kur G_x ir G_y apskaičiuojami pagal (1.3) algoritmą:

G_x

-1	0
0	1

G_y

0	-1
1	0

(1.3)

Prewitt kraštų radimo operatorius:

Šis operatorius naudoja kaukę, kurios dydis yra 3x3 suderintom su dalinėm išvestinėmis, Prewitt operatorius yra tikslesnis lyginant su Roberts operatoriumi. Prewitt operatorius kaukė pateikta žemiau esančiose (1.4) matricose G_x ir G_y:

G_x

-1	-1	-1
0	0	0
1	1	1

G_y (1.4)

-1	0	1
-1	0	1
-1	0	1

Išvestinės x kryptis apskaičiuojama pagal skirtumą tarp trečiosios ir pirmosios 3x3 matricos eilutės, o y kryptis tarp trečiojo ir pirmojo stulpelio skirtumo (1.5) ir (1.6) formulės[10].

$$g_x = \frac{\partial f}{\partial x} = (Z_7 + Z_8 + Z_9) - (Z_1 + Z_2 + Z_3) \quad (1.5)$$

$$g_y = \frac{\partial f}{\partial y} = (Z_3 + Z_6 + Z_9) - (Z_1 + Z_4 + Z_7) \quad (1.6)$$

čia Z_n – gradientų matricų elementai.

Sobel kraštų radimo operatorius:

Tai modifikuota 3x3 Prewit kaukės versija. Prewit operatoriaus centrinis kaukės koeficientas yra 1, kai šis koeficientas pakeičiamas į 2 gaunama nauja Sobel kraštų aptikimo kaukė. Branduoliai gali būti taikomi atskirai pradiniam vaizdui, taip atrandant G_x ir G_y gradientų orientacijas. Sobel operatoriaus kaukės G_x ir G_y pateiktos (1.7) žemiau:

G_x

-1	-2	-1
0	0	0
1	2	1

G_y (1.7)

-1	0	1
-2	0	2
-1	0	1

Sobel operatorius yra grdiento dydis, kuris apskaičiuojamas pagal (1.8) formulę[10].

$$M = \sqrt{g_x^2 + g_y^2} \quad (1.8)$$

kur:

$$g_x = \frac{\partial f}{\partial x} = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3) \quad (1.9)$$

$$g_y = \frac{\partial f}{\partial y} = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7) \quad (1.10)$$

čia Z_n – gradientų matricų elementai.

Kraštai aptinkami atrankiniu būdu parenkant slenksčio reikšmę.

Canny kraštų radimo operatorius:

Canny kraštų aptikimo operatorius yra pranašesnis už aukščiau aptartus operatorius, nes šiam operatoriui būdingas žemas klaidų lygis, geriau lokalizuoti kraštai. Vaizdas yra išlyginamas naudojant apskritąjį dviejų dimensijų Gauso funkciją. Apskaičiuojamas gradientas ir tada naudojamas gradiento reikšmė ir kryptis atitinkantis apytikslius kraštų stiprumus ir kryptis kiekviename vaizdo taške.

Filtruojant vaizdą Gauso filtro sąsuka (1.11 formulė) gaunamas išlygintas duomenų masyvas f_s.

$f_s=[G(x,,y)*f(x,y)]$ (1.11)

čia f(x,y) – yra pradinis vaizdas

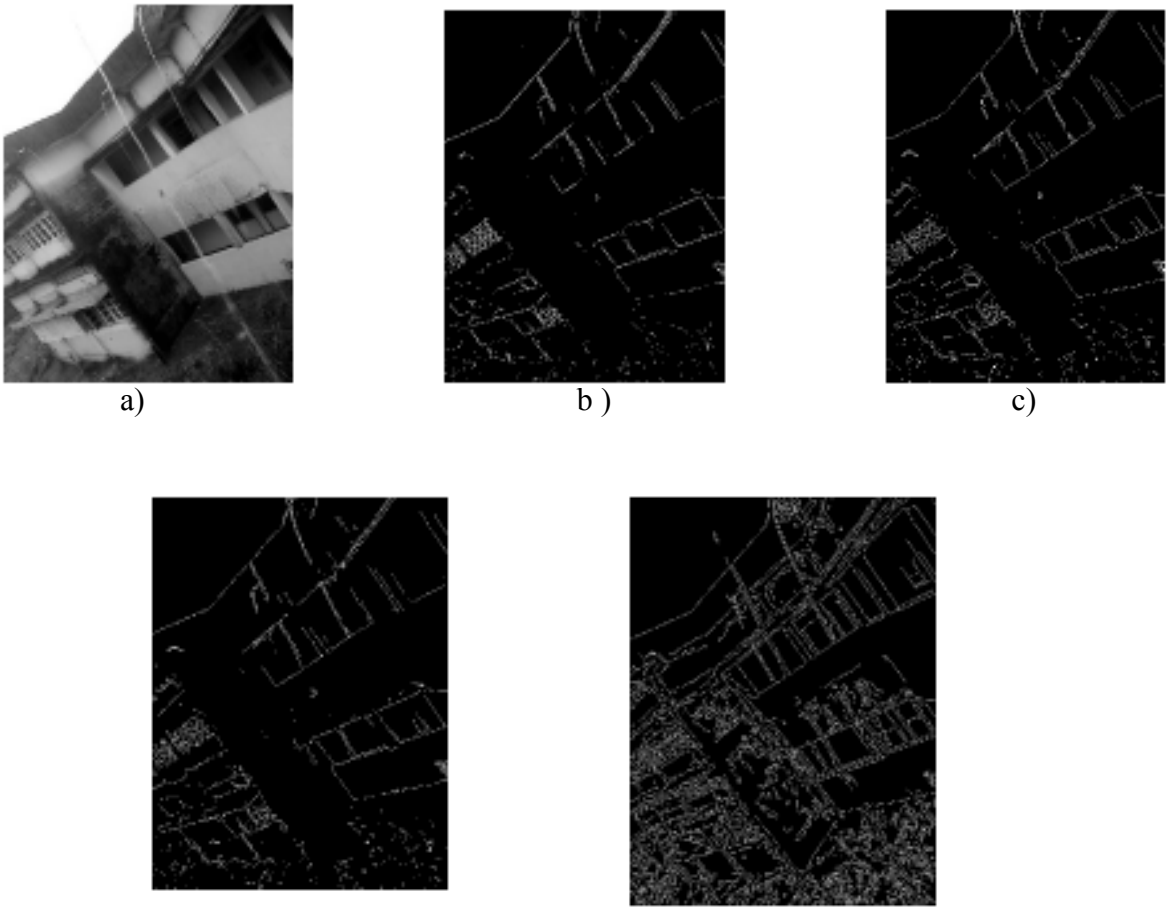
$G(x,y)=e^{\frac{x^2+y^2}{2\sigma^2}}$ (1.12)

Sekantis žingsnis yra apskaičiuoti gradiento dydį ir kampą:

$M(x,y)=\sqrt{g_x^2+g_y^2}$ (1.13)

$\alpha(x,y)=\tan^{-1}\left[\frac{g_y}{g_x}\right]$, kur $g_x=\frac{\partial f_s}{\partial x}$, ir $g_y=\frac{\partial f_s}{\partial y}$ (1.14)

Tada filtro kaukė yra naudojama gauti g_x ir g_y reikšmėms. M(x,y) masyvas turės aiškias briaunas aplink maksimumo vietas. Tai nėra pageidautinas efektas, todėl šio efekto sumažinimui naudojama maksimumų slopinimo technika. Šio žingsnio įtraukimo tikslas yra nurodyti krašto normalės diskretinių orientacijų skaičių. Klaidingi kraštų fragmentai yra atmetami naudojant dviejų slenksčių algoritmą, kuris naudoja du slenksčius T1 ir T2, kurie susiję sąryšiu T1=2*T2. Šių žingsnių įtraukimas leidžia paploninti kraštą iki vieno pikselio pločio[10].



a) b) c)

d) f)

1.5 Pav. Skirtingais kraštų radimo operatoriais apdorotas paveikslas: a) originalas, b) Roberts, c) Previt, d)Sobel, f) Cany [10]

2. Panašių tyrimų apžvalga

Toliau bus apžvelgti su vaizdų atpažinimu susiję darbai ir detaliau panagrinėtos LPLM (angl. FPGA) grandinės, kuriomis realizuojami algoritmai vaizdų apdorojime.

2.1 Veido atpažinimas ir sekimas realiuoju laiku

Viena iš sustiprintos saugos problemų yra žmogaus veido atmažinimas ir sekimas. Taip pat ši technologija yra svarbi veido gestų telekomunikacijose, robotikoje, bei žmogus – kompiuteris sąveikai (angl. human-computer interactions (HCI)).

Veido atpažinimo sistemos tikslas yra atpažinti bet kurio žmogaus veido atvaizdą. Veido atpažinime veikia įvairūs trukdžiai: veido dydis iš įvairių atstumų, veido orientacija erdvėje ir stebimi požymiai veide Pasiūlytus metodus galima suskirstyti į 4 pagrindines metodų grupes[12]:

- Žiniomis grįstus metodus;
- Invariantišku požymių metodus;
- Šablono atitikties metodus;
- Išvaizda pagrįstus metodus.

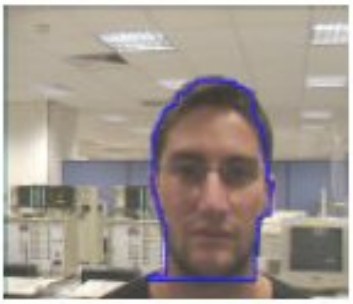
Žiniomis grįsti metodai naudoja žinias apie žmonių veidus. Invariantiškų požymių metoduose ieškoma tokių požymių, kurie tinka veidams (pavyzdžiui odos spalva). Šablono atitikties metoduose ieškoma koreliacijos tarp veido šablono ir tiriamo vaizdo. Išvaizda pagrįstuose metoduose nėra iš anksto pateikto šablono, bet jis yra randamas mašininio mokymo metodais.

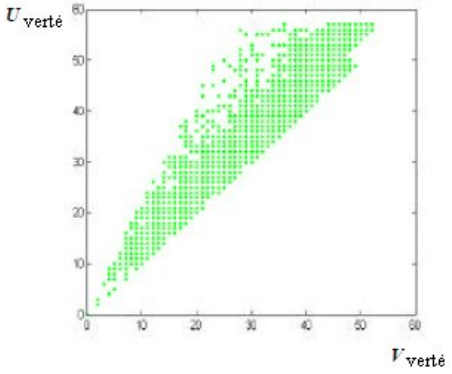
Odos spalvos naudojimas kaip bruožas veido stebėjime turi keletą privalumų. Vienas iš privalumų yra tai, kad veido odos spalvos apdorojimas yra daug greitesnis, kai neįskaitomi kiti veido bruožai. Kitas privalumas yra tai, veido odos spalvos nepriklauso nuo orientacijos esant tam tikromis apšvietimo sąlygomis. Tačiau spalva nėra fizinis reiškinys, spalva yra suvokiama kai į akies tinklainę krinta nuo daikto atsispindėjusios elektromagnetinės bangos. Tai kelia daug problemų. Pirma, kai veido spalva gauta nufilmavus kamera, gautoji odos spalva gali būti paveikta trikdžių, tokių kaip aplinkos apšvietimas, objekto judėjimas, šešėliai ir t.t. Antra, skirtingos kameros skirtingai atkuria spalvas, net filmuojant tą patį asmenį prie tokio pat apšvietimo gaunamos skirtingos spalvos reikšmės. Trečia veido odos spalvos skiriasi priklausomai nuo asmens. Spalva gali kisti nuo tamsios iki baltos. Tam, kad būtu galima naudoti odos spalvą kaip kriterijų, reikia išspręsti anksčiau paminėtąsias

problemas. Siūlomu metodu siekiama sėkmingai išspręsti visas su veido atpažinimu ir sekimu iškilusias problemas, nepriklausomai nuo veido dydžio, krypties ir odos tonų[12].

Norėdami nustatyt veido odos spalvos apibrėžimą buvo panaudotas UV žemėlapis (angl. UV mapping). Analizuojant UV žemėlapius, buvo ieškoma bendrų vektorių kurie būtų bendri visiems veidams, nepriklausomai nuo odos spalvos kitimo laipsnio. Atlikus bandymus buvo gauti trys atskiros veido tonų kategorijos, šviesus odos tonas, vidutinis odos tonas, ir tamsus odos tonas. Kiekvienai kategorijai yra atstovauja tam tikra etninė grupė. Kaukazo etninė grupė vertinama kaip šviesios odos tonų savininkai, kinų kaip vidutinių tonų, o tarp indų tamsūs tonai.

Geriausias sutapties regionas UV žemėlapio kuris fiksuoja šviesius veido tonus pavaizduotas 2.1 paveiksle.

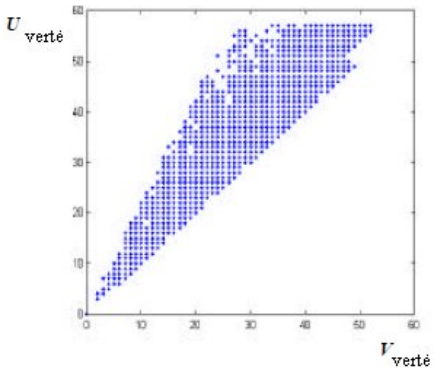




a) b)

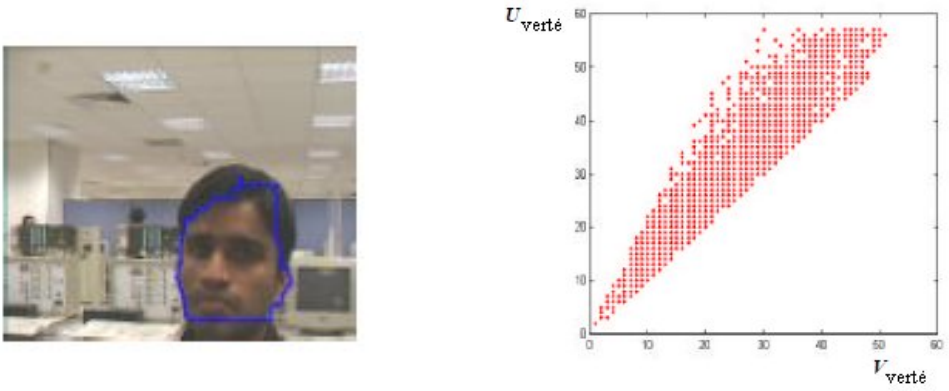
2.1 pav.a) išskirtasis šviesių tonų veidas ir b) UV žemėlapis su šviesių tonų regionais[12]

(2.2 a)) ir (2.3 a)) paveiksluose pavaizduoti vidutinės ir tamsios odos geriausi sutapties regionai. Kaip galima pastebėti (2.3 pav. a)) veido odos tonai turi bendrą UV regioną UV žemėlapyje (2.4 pav.) kuris yra galutinis 2 dimensijų veido odos spalvų modelis.



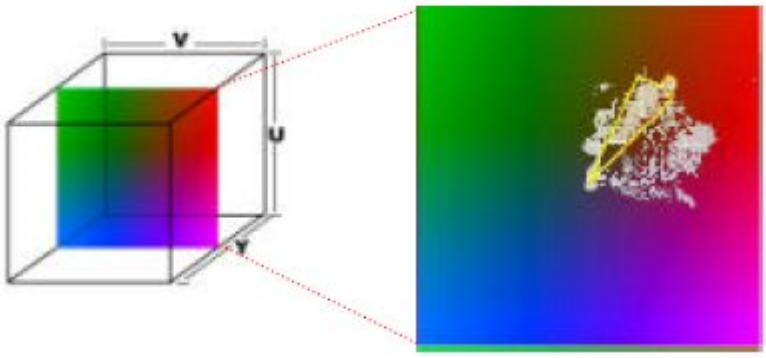
ą) b)

2.2 pav.a) išskirtasis vidutinių tonų veidas ir b) UV žemėlapis su vidutinių tonų regionais[12]



a) b)

2.3 pav.a) išskirtasis tamsių tonų veidas ir b) UV žemėlapis su tamsių tonų regionais[12]



2.4 pav. Konversija iš 3D YUV spalvų erdvės į 2D UV žemėlapį[12]

2.4 Paveiksle apibrėžtas regionas matematiškai aprašomas 2.1, 2.2, 2,3 2.4 lygtimis.

(2.1)

$$43 \leq \sqrt{u^2 + v^2} \leq 78 \tag{2.2}$$

$$0.25 \cdot \pi \leq \tan^{-1}\left(\frac{u}{v}\right) \leq 0.3611 \cdot \pi \tag{2.3}$$

$$0 \leq \sqrt{u^2 + v^2} \leq 70 \tag{2.4}$$

čia u ir v yra U ir V spalvos vertės kiekvienam atvaizdo pikseliui.

Pikselis yra priskiriamas odos spalvai, jeigu jo padėtis UV žemėlapyje tenkina vieną iš dviejų lygčių rinkinių. Tokiu atveju veido odos spalvos sritis gali būti išskirta. Tai parodyta 2.5 paveiksle.

2.5 pav. Pirminis rezultatas apdorojus odos spalvą, kairė – originalas, dešinė apdorotas vaizdas[12]

Siekiant išvengti atsitiktinių „druskos ir pipirų“ triukšmo, medianos filtravimas atliekamas po veido odos spalvos gavimo. Toliau pašalinami visi kiti pašaliniai fono taškai. „Erozija“ atliekama, kad padaryti vaizdą tolygesnį, pašalinant iš jo mažus plotelius ir skilutes, taip apibrėžiant veido ribas 2.6 pav.

2.6 pav. Gauti rezultatai po odos spalvos gavimo po apdorojimo a) prieš medianos ir morfologinio apdorojimo b) triukšmo pašalinimas ir medianos apdorojimas c) galutinis veido vaizdas po morfologinio apdorojimo [12]

Aptiktasis veido regionas analizuojamas euristinėmis taisyklėmis, kurios yra grindžiamos bendrąją žmogaus veido geometrine analize.

Panaudojant įvairius vaizdų apdorojimo metodus galima iš vaizdo išskirti netik veidą, bet ir veido sritis ar bruožus. Tai galima pritaikyti žmonių atpažinimui iš veido, ar veido specifinių bruožų.

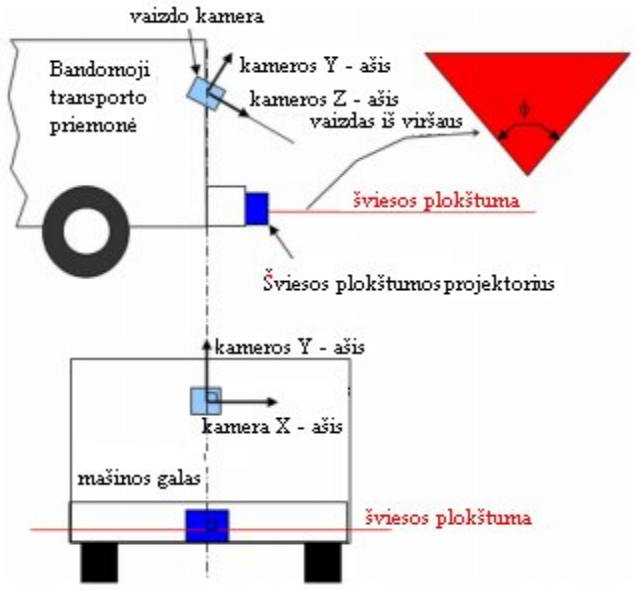
.

2.2 Šviesos juostelės projekcijos taikinio pozicijos nustatymas pažangiose automatinio parkavimo sistemose

Protingos pagalbinės parkavimo sistema (PPPS, angl. IPAS) tai sistema, kuri automatizuoja automobilio prisiparkavimą. Apskritai, IPAS susideda iš šešių komponentų: taikinio padėties nustatymas, kelio planavimas, maršruto stebėjimas, automatinis vairavimas, automatinis stabdymas, ir žmogus-mašina sąsaja (angl. HMI). Nors ultragarsinių jutiklių (angl. IPAS) sistemos jau parduodamos ir susilaukė vartotojų dėmesio, šios sistemos dar negalima vadinti automatine, kuri be žmogaus įsikišimo galėtu savarankiškai pastatyti mašiną[15].

Kaip problemos sprendimas siūlomas naujas taikinio padėties nustatymo metodas, specialiai sukurtas automobilių statymui mažai apšviestuose garažuose, ar uždaruose automobilių stovėjimo aikštelėse, pavyzdžiui, požeminėse automobilių stovėjimo aikštelėse, kurios gali būti puikus iššūkis kompiuterinės regos sistemoms.

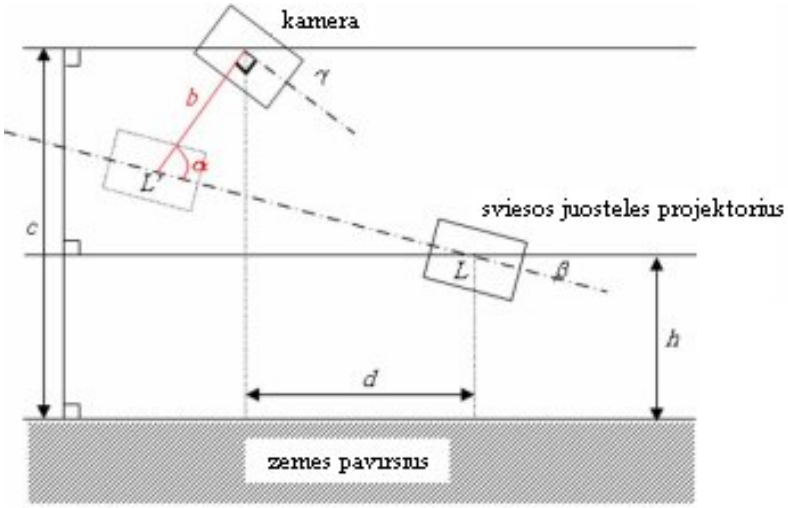
Sistema gali būti sumontuota mašinos gale 2.7 paveikslas. Sistemą sudaro vaizdo kamera ir infraraudonųjų spindulių lazeris. Infraraudonųjų spindulių lazeris yra nematomas žmogaus akims, todėl spindulys patekęs praeiviui į akis jų nevargia. O vaizdo kameros objektyvas tokiems spinduliams yra jautrus ir juos gali užfiksuoti[15].



2.7 pav. Sistemos sudėtis[15]

Kad fiksuojamas vaizdas turėtu platų matymo lauką naudojamas kameros objektyvas su „fisheye“ (žuvies akis) vaizdo iškraipymu, toliau apdorojamame vaizde pašalinamas „fisheye“ efektas, taip atkuriant neiškraipytą vaizdą.

Galima daryti prielaidą, kad LPP(liet. Šviesos plokštumos projektorius), kuris randamas taške L, yra virtualioje pozicijoje, kadangi LPP sudaro šviesos plokštumą, kaip parodyta 2.8 paveiksle.



2.8 pav. Normalizacijos konfigūravimas[15]

Normuojant konfigūraciją yra surandamas bazinis atstumas b tarp kameros ir L taško, kuris patalpintas ant kameros y ašies ir tarp α kampo. Paveiksle panaudoti pažymėjimai:

c – kameros atstumas nuo žemės;

d – horizontalus atstumas tarp kameros ir LPP;

h - LPP atstumas nuo žemės;

γ – kameros pasvyrimo horizonto atžvilgiu kampas;

β – LPP pasvyrimo pasvirimo horizonto atžvilgiu kampas.

Tokiu atveju:

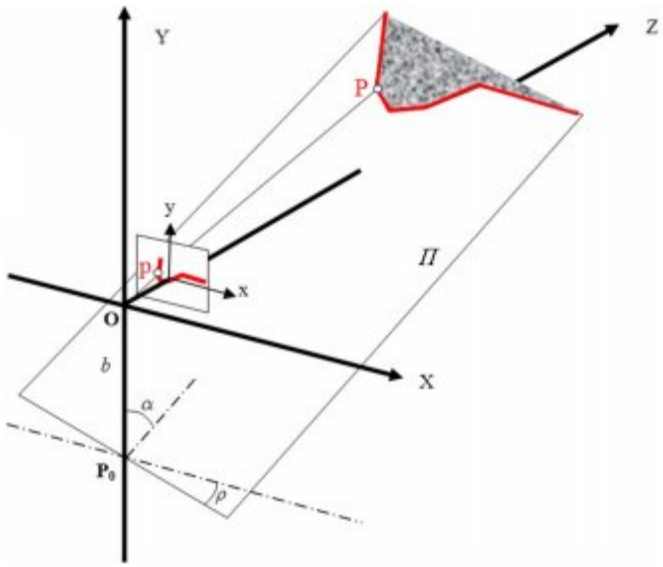
$\alpha = 90 + \beta - \gamma$

$b: c - h - d \cdot \tan(\beta) = \sin(90 - \beta) : \sin(90 - \beta - \gamma);$ (2.5)

$$b = \frac{\sin(90 - \beta)}{\sin(90 + \beta - \gamma)} (c - h - d \cdot \tan(\beta)).$$ (2.6)

Ieškant sankirtos taško tarp plokštumos Π ir tiesės OP galima apskaičiuoti skanuojančio lazerio plokštumos pikselio taško P koordinates, kaip parodyta 2.9 paveiksle. Π plokštuma yra lazerio spindulio suprojektuota plokštuma. Projektuojamasis lazerio spindulys sukuria taškus dvimatėje

plokštumoje p(x,y) ir įvertinus lazerinio projektoriaus ir kameros aukšti gaunami kliūties taškų koordinatės trimatėje erdvėje P(X,Y, Z).



2.9 pav. Šviesos plokštumos projekcija į kliūtį[15]

Π plokštumoje P0 taško koordinatės Y plokštumoje atitinka P0(0, -b, 0) koordinatės. Tarp Y ašies ir lazerio spindulio plokštumos yra kampas α, o tarp plokštumos ir X ašies kampas ρ. Atstumas tarp kameros ir P0 yra lygus b. Kampas α yra apskaičiuojamas iš konfigūracijos normalizacijos. Šviesos plokštumos normalės vektorius Π yra surandamas, sukant XY plokštumos normalės vektorių, t. y. (0, 1, 0) vektorių, kampu $\pi/2 - \alpha$ X ašies atžvilgiu bei kampu ρ Z ašies atžvilgiu.

$$n = \begin{bmatrix} \sin\alpha \cdot \sin\rho \\ \sin\alpha \cdot \cos\rho \\ -\cos\alpha \end{bmatrix} \quad (2.7)$$

Iš plokštumos lygties galima gauti normalinį vektorių n ir vieną tašką P0 plokštumoje.

$$n(X - P_0) = 0 \quad (2.8)$$

Lygties O p optinio centro O vienas taškas

$$\frac{X}{x} = \frac{Z}{f} = \frac{Y}{y} \quad (2.9)$$

Optinis centras O

$$Q = (k \cdot x, k \cdot y, k \cdot f) \tag{2.10}$$

$$k = \frac{b \cdot \tan\alpha \cdot \cos\rho}{f - \tan\alpha(x \cdot \sin\rho + y \cdot \cos\rho)} \tag{2.11}$$

Op lygtis: optinis centras O, vienas taškas P ant LSF ir atitinkamas taškas vaizdo plokštumoje pturi būti ant tos pačios tiesės erdvėje. Pagal perspektyvinės kameros modelį, visi tiesės Op taškai Q gali būti aprašyti parametru k. Čia f žymi optinės sistemos židinio nuotolį, (x,y) – taško p koordinatės. Gauname tokias lygtis:

$$X = \frac{x \cdot b \cdot \tan\alpha \cdot \cos\rho}{f - \tan\alpha(x \cdot \sin\rho + y \cdot \cos\rho)} \tag{2.12}$$

$$Y = \frac{y \cdot b \cdot \tan\alpha \cdot \cos\rho}{f - \tan\alpha(x \cdot \sin\rho + y \cdot \cos\rho)} \tag{2.13}$$

$$Z = \frac{f \cdot b \cdot \tan\alpha \cdot \cos\rho}{f - \tan\alpha(x \cdot \sin\rho + y \cdot \cos\rho)} \tag{2.14}$$

Transformaciją į automobilio koordinačių sistemą. Automobilio koordinačių sistema yra apibrėžiama taip, kad sistemos centras yra optinio centro O projekciją į žemę, XY plokštuma – lygiagreti žemei. Sukant P tašką kameros koordinačių sistemoje kampu γ X ašies atžvilgiu ir perstumiant per kameros atstumą c nuo žemės pagal Y ašį. Atitinkamas taškas Pin automobilio koordinačių sistemoje bus apskaičiuotas;

$$P' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ c \\ 0 \end{bmatrix} \tag{2.15}$$

.

Surasto pikselio i koordinatės trimatėje erdvėje apskaičiuojamos pagal (2.12)-(2.15) lygtis.

2.3 Nedideli duomenų apdorojimo konvejeriai

Apdorojant vaizdus apartūrškai, pilna vaizdo pikselio apdorojimo trukmė viršija taktinio generatoriaus periodo reikšmę. Naudojant konvejerį, vaizdo apdorojimas yra suskirstomas blokais, kurių vieno, apdorojimo trukmė gali sutapti su taktinių impulsų generatoriaus periodu, bet bendra jų visų apdorojimo trukmė yra keli ar keliolika periodų. Tačiau visumoje vaizdo apdorojimas yra paspartinamas.

Pavyzdys: tarkime, kad kvadratinė lygtis turi būti apskaičiuota *x* reikšmių diapozone kiekvienam apdorojimo ciklui[16]:

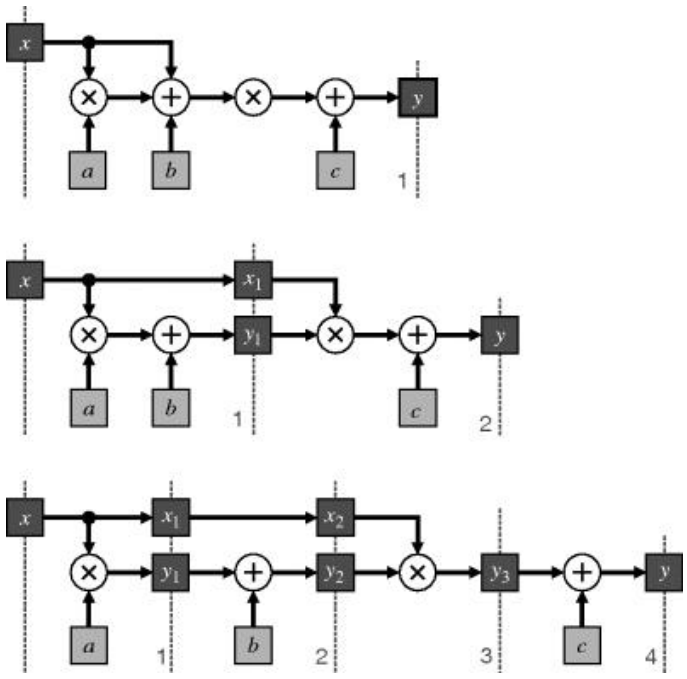
$$y = a \cdot x^2 + b \cdot x + c = (a \cdot x + b) \cdot x + c \qquad (2.16)$$

čia a, b, c konstantos;

x – įėjimo signalas;

y – išėjimo signalas.

Jeigu viename apdorojimo cikle, kaip nurodyta pav. 2.10, vėlinimas susideda iš dviejų dauginimo ir dviejų sudėties operacijų trukmių. Jei jis per ilgas, skaičiavimai gali būti išskaidytas į du iteracijos ciklus, kurie pavaizduoti viduriniame paveikslėlyje. Registre y_1 išsaugomas pimųjų skaičiavimų rezultatas, taip leidžiant antrajai sudėčiai pereiti į antrąjį apdorojimo ciklą. Reikia atkreipti dėmesį, kad registras x_1 reikalingas, kad suvėlintu įvesti į antrąjį apdorojimo ciklą. Be x_1, naujoji *x* vertė įvestyje būtų panaudota antrojoje sudėtyje, kuris pateiktų klaidingus rezultatus. Kiekvienas vėlinimas per kiekvieną iteracija, sumažėja beveik per pusę, tačiau, mažas papildomas vėlavimas atisranda su kiekvienu papildomu registru. Dviejų stadijų konvejeris padidintų apdorojimo greitį beveik dvigubai. Kiekvienas naujas skaičiavimas prasideda vis su nauju taktavimo impulsu, nors kiekvienas skaičiavimas nuo šiol reikalauja dviejų apdorojimo ciklų. Tokį įrenginio našumą sumažina tik nedidelė papildoma apdorojimo trukmė konvejeryje[16].



2.10 pav. Paprastas konvejerio pavyzdys. Viršutinysis: atliekamas skaičiavimas viename apdorojimo cikle; vidurinysis: dviejų stadijų konvejeris; apatinis: keturių stadijų konvejeris[16]

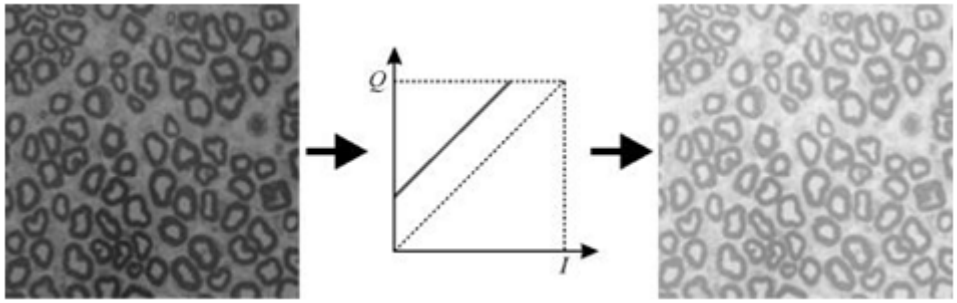
Konvejeriui gali prireikti daugiau loginių elementų. Taip atsitinka, nes reikalingi papildomi registrai saugoti stadijų rezultatus. Registrai yra padaromi iš trigerių. Gali būti panaudoti trigeriai, kurie kartu su loginiais elemtais yra LPLM ląstelėse. Pavyzdžiui, sukurti registrą $y_1$bus panaudoti trigeriai, kurie yra ląstelėse, iš kurių sukuriamas sumatorius. Tuo tarpu registras x_1 nėra susijęs su loginio įtaiso išvestimi, todėl jis gali būti padarytas iš nepanaudotų kitų ląstelių trigerių.

Pav. 5.1 apatinėje dalyje, konvejeris yra daugiau išplėstas, kuris išskaido skaičiavimus į keturis apdorojimo ciklus. Tačiau šį kartą apdorojimo periodas negali sumažėti pusiau, nes dauginimo vėlinimas yra daug ilgesnis negu sumavimo įrenginio. 1 ir 3 konvejerio pakopos, kurios turi daugiklius, ribos maksimalų apdorojimo dažnį. Taigi, nors maksimalus našumas gali būti truputėlį padidintas, tuo pačiu padidėja ir vėlinimas, nes 2 ir 4 pakopos dalį periodo neveikia.

Išbalansuotas vėlinimas skirtingose pakopose gali būti pagerintas, pasinaudojus “laiko perskirstymą” (angl. retiming). Tai reikalauja, kad dalis dauginimo logikos būtų perkelta iš 1 ir 3 pakopų per registrus y_1 ir y_3 į pakopas 2 ir 4. Priklausomai nuo logikos perkėlimo ir priklausomai nuo to kaip ji perkelta, gali iššaugti panaudojamų trigerių skaičius. Atliekant tai rankiniu būdu gali padidėti daromų klaidų[16].

2.4 Kontrasto ir šviesumo reguliavimas

Norint vaizdą padaryti šviesesniu, išėjimo taškų vertės turi būti padidintos. Tai yra pasiekiama pridedant konstantos reikšmę, kaip parodyta 2.11 paveiksle. Vaizdas gali būti ir tamsinamas, atimant iš pikselių konstantą.

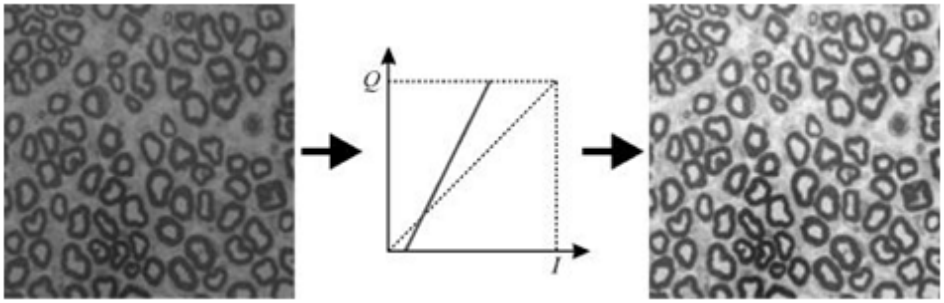


2.11 pav. Vaizdo šviesinimas pridedant konstantą[16]

Tačiau praktikoje vaizdo ryškumo keitimas yra sudėtingesnis procesas, nes žmogaus regos sistema yra netiesinė ir neįsimena kiekvieno atskiro paveikslėlio pikselio reikšmės, o lygina vaizdą bendrame kontekste. Tokia optinė iliuzija pavaizduota 2.12 paveiksle , kur per paveikslėlio vidurį einanti juosta iš kairės atrodo šviesiau, o einant į dešinę pusę juosta tamsėja, tačiau iš tikro visos juostos spalva (pikselių vertė) yra tokia pati.

2.12 pav. Ryškumo iliuzija[16]

Vaizdo kontrastas yra koreguojamas pagal kartografavimo funkcijos nuolydi. Jei nuolydis didesnis nei vienetas tai rodiklis atitinka kontrasto padidėjimą, o jei nuolydis mažesnis nei vienetas, tai atitinka kontrasto sumažėjimą 2.13 paveikslas.



2.13 pav. Kontrasto didinimas vaizde[16]

Taškinė operacija, kuri kartu reguliuoja šviesumą ir kontrastą yra:

$$Q = a \cdot I + b = a \cdot (I + b') \qquad (2.17)$$

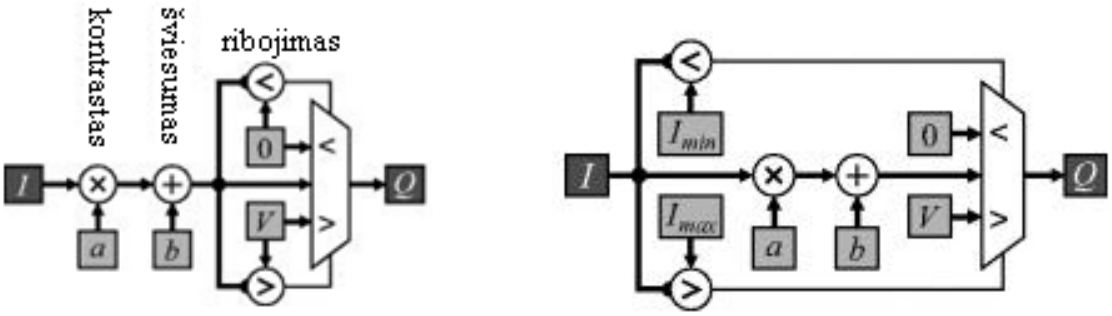
čia a ir b – konstantos kuriomis reguliuojamas ryškumas ir kontrastas;

Q – pikselio išėjimo vertė (nuo 0 iki 255);

I – pikselio įėjimo vertė (nuo 0 iki 255);

Problema atsiranda, kai rezultatas netelpa į 8 bitus. Kokia turėtų būti išvestis, jeigu *Q* būtų už 0-255 diapazono ribų? Paėmus aštuonis jauniausius bitus, sukeltų reikšmių kitimą ratu. Jeigu tikslas yra padidinti šviesumą, galima gauti priešingą efektą. Todėl labiau verta panaudoti reikšmių įsotinimą.

Ribojimui reikalingas prietaisas galintis aptikti kada yra viršijamas limitas ir būtų atitinkamai priderintas prie išvesties. Tai matoma 2.14 paveiksle a). Galima pastebėti, kad nei šviesumo , nei kontrasto operacijos negali būti atliekamos pirmiau pagal 2.17formulę[16].



a) b)

2.14 pav. Schemos atliekančios kontrasto reguliavimo operacijas a) ribojimas po apdorojimo b) ribojimas prieš apdorojimą[16]

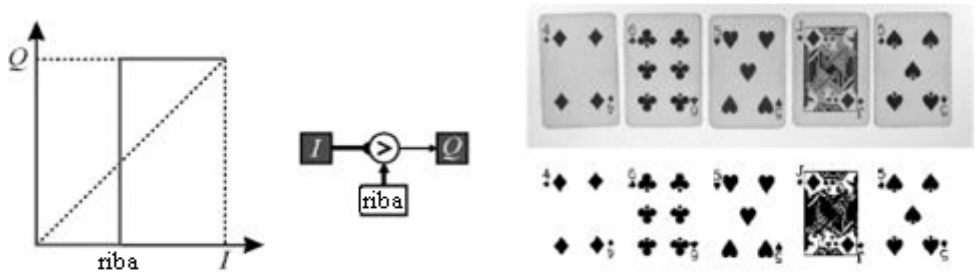
2.14 pav. Pavaizduota paprasčiausia kontrasto pritaikymo operaciją. Dešinėje pusėje pavaizduota, kaip atrodo pagerintas darbas perkeliant palyginimus į įvestį. Kairėje pusėje vaizduojama, ribojimo atlikimas po (2.17) formulės apskaičiavimo. Į visą vėlinimo trukmę įeiną ribojimo patikrinimo laikas. Tačiau, jeigu filtravimo testas butų atliktas prieš įvestį, jis būtų apdorotas lygiagrečiai su kontrasto pritaikymo operacija (kaip pavaizduota dešinėje.), kuris sumažintų vėlinimą. Tam naudojama ši formulė[16].

$$I_{min} = \frac{0 - b}{a}, \quad I_{max} = \frac{V - b}{a} \tag{2.18}$$

čia I – pikselio įėjimo vertė (nuo 0 iki 255);

2.5 Bendras slenkstis ir kontūro suradimas

Palyginimas su slenksčiu pagal jo vertę palyginamas kiekvienas pikselis vaizde su slenkstine verte ir išvedama juoda arba balta reikšmės 2.15 paveikslas



a) b) c)

2.15 pav. Bendras vieno slenksčio ribojimas vaizde a) ribos parinkimo diagrama, b) ribojimo operaciją atliekanti schema c) apdorotasis vaizdas[16]

Kadangi kiekvienas pikselis vienodai apdorojamas, palyginimas su slenksčiu yra Taškinė operacija.

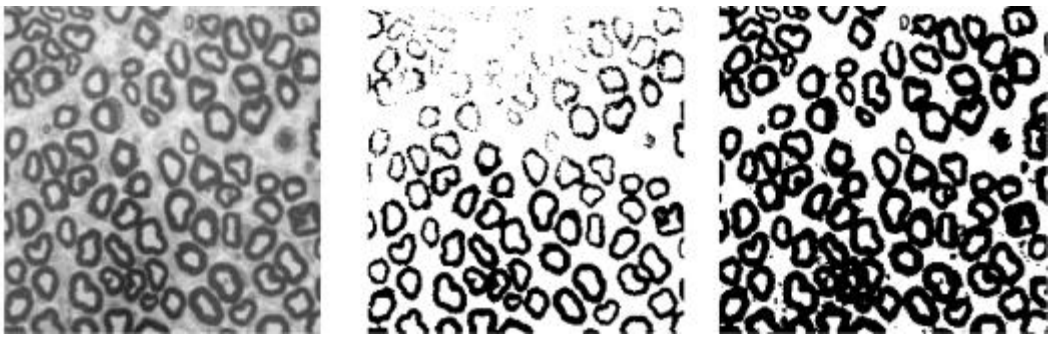
$$Q = \begin{cases} 1, & I \geq riba \\ 0, & I < riba \end{cases} \tag{2.19}$$

čia I – pikselio įėjimo vertė (nuo 0 iki 255);

Q – pikselio išėjimo vertė (nuo 0 iki 255);

Kiekvienas pikselis slenkstyje yra skirstomas į dvi klases ir bendrai naudojamas išskirti objektą iš fono[16].

Kartais vienas globalus slenkstis netinka visam vaizdui. Pavyzdys XX paveiksle:



a) b) c)

2.16 pav. Bendras vieno slenksčio ribojimas vaizde a) originalus vaizdas, b) gautas vaizdas parinkus žemą slenkstine vertę, c) gautas vaizdas parinkus aukštą slenkstine vertę[16]

Optimalus slenkstis vienai vaizdo daliai, gali būti neoptimalus kitai. Naudojant bendrą visam vaizdui slenkstį, galima sugadinti vaizdą. Taškinė operacija netinkama apdoroti tokius vaizdus, kadangi operacijos metu reikia atsižvelgti į aplinkinius pikselius. Tokie vaizdai reikalauja filtravimo su kintamu sleknsčiu.

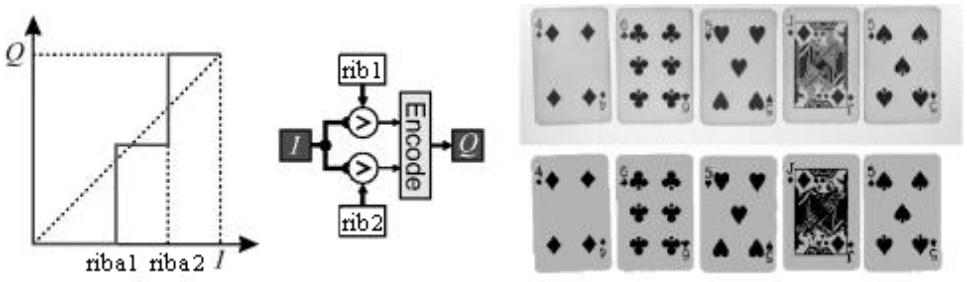
Paprastai vaidų apdorojime naudojamas daugiau nei vienas slensčio lygis 2.20 lygtis.

$$Q = \begin{cases} 2, & riba_2 \leq I \\ 1, & riba_1 \leq I < riba_2 \\ 0, & I < riba_1 \end{cases} \quad (2.20)$$

čia I – pikselio įėjimo vertė (nuo 0 iki 255);

Q – pikselio išėjimo vertė (nuo 0 iki 255);

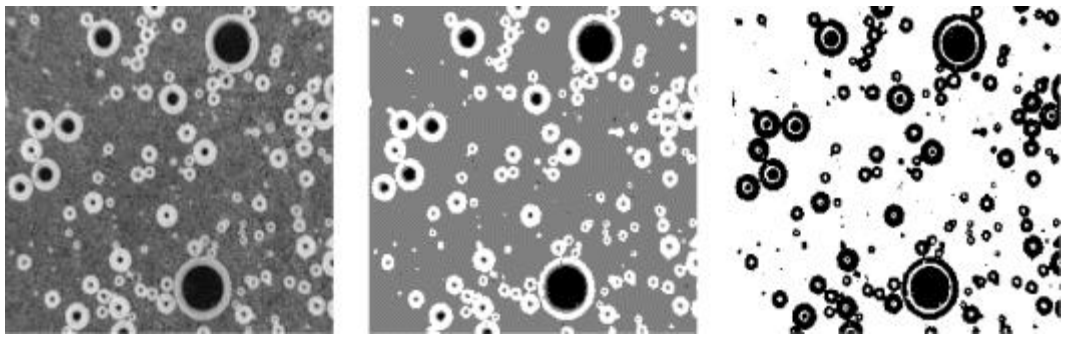
Toks pavyzdys pateiktas 2.16 paveiksle, čia panaudotos dvi ribos atskirti vaizdus į tris regionus: foną, kortas ir paveikslėlius kortuose[16].



a) b) c)

2.17 pav. Keletos slenksčių ribojimas vaizde a) ribos parinkimo diagrama, b) ribojimo operaciją atliekanti schema c) apdorotasis vaizdas[16]

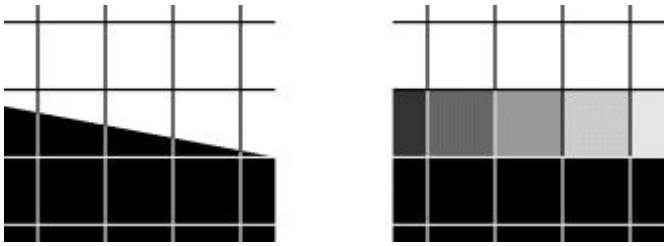
Šiame pavyzdyje tamsiausias regionas niekada nebus greta šviesiausio regiono. Kaip parodyta 2.17 paveiksle c). Tačiau toks atvejis netinka (2.18 pav.) vaizdui, kuriame burbuliukų vidus yra juodas. Be to dar yra ryškumo gradacijos tarp balto ir juodo, duodančios grupes taškų juodose figūrose surandamus kaip fono taškus. Tačiau net jai staiga pasikeitė spalva tarp burbuliukų, daugelis kraštų pikselių vis dar turi tarpinius spalvos lygius. Taip yra todėl, kad vaizdo plote imamos suvidurkintos pikselių reišmės tame plote[16].



a) b) c)

2.18 pav. Klaidngo klasifikavimo problema. Kairėje – originalus vaizdas, centre – klasifikuojami 3 lygiai, dešinėje – fonas pažymėtas baltai. Pagal foną matyti klaidos burbuliukų viduje [16]

Tačiau net jai staiga pasikeitė spalva tarp burbuliukų, daugelis kraštų pikselių vis dar turi tarpinius spalvos lygius. Taip yra todėl, kad vaizdo plote imamos suvidurkintos pikselių reikšmės tame plote. Tad, jei slenkstis patenka į pikselio ribą 2.19 pav., dalis pikselio bus juodas dalis baltas ir suvidurkinus pikselio reikšmę gaunasi tarpinės spalvos tarp vaizdo kraštų[16].



a) b)

2.19 pav. Pikseliai turintys tarpines reikšmes [16]

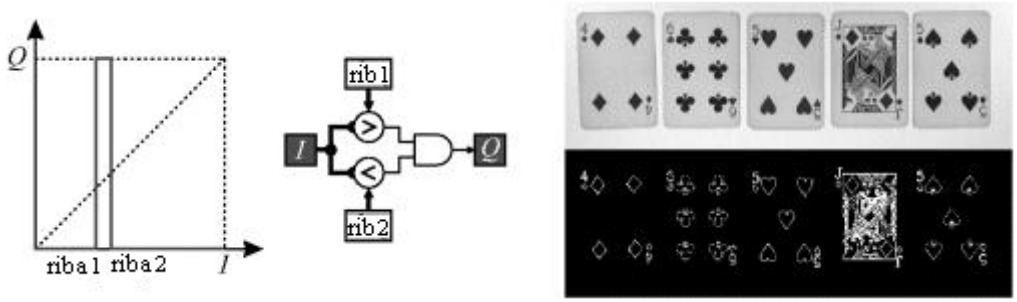
Todėl, panaudojant kelių lygių slenksčius, yra neišvengiama, kad kai kurie ribiniai pikseliai pagal savo reikšmes pateks tarp slenksčių ir bus neteisingai suklasifikuoti. Norint išvengti tokio neteisingo klasifikavimo, reikalinga konteksto informacija, kuri būtų panaudota arba prieš palyginimą su slenkstine reikšme arba po jo, bei ji padėtų perklasifikuoti neteisngai suklasifikuotus pikselius[16].

$$Q = \begin{cases} 0, & riba_2 \le I \\ 1, & riba_1 \le I < riba_2 \\ 0, & I < riba_1 \end{cases} \qquad (2.21)$$

čia I – pikselio įėjimo vertė (nuo 0 iki 255);

Q – pikselio išėjimo vertė (nuo 0 iki 255);

Kitas daugiapakopio slenksčio pritaikymas yra išskirti kontūrus paveikslėlyje. Tačiau šis metodas yra mažiau efektyvus ten, kur yra didelis kraštų kontrastas, nes jau yra išanksto parinktos tarpinės pikselių reikšmės 2.20 pav. Kraštų aptikimas priklauso nuo pikselių slenksčio intervalo parinkimo, kontūrų radimas vistiek nebūna geros kokybės, nes gauti kraštai nebeturi kontekstinio ryšio su paveikslėliu.



2.20 pav. Ieškant kontūrų, pasirenkami taškai tarp dviejų lygių [16].

Kadangi po tokio vaizdo apdorojimo gauti vaizdo pikseliai turi tik dvejetainę informaciją apie vaizdą, gautasis vaizdas gerokai mažiau užima vietos atmintyje[16].

3. Atspindžio koordinačių metodinės priemonės

Atliekant tyrimą buvo naudojama aparatinė ir programinė įranga. Programinė įranga sudarė dvi dalys: NIOS II programinis procesoriaus ir programa, kuri buvo skirta programiniam procesoriui, o techninė dalį sudarė plokštė su kuria buvo galima atlikti programinės sistemos bandymus.

3.1 Qartus II

Qartus II tai kompanijos „Altera“ produktas . Šis programinis paketas pritaikytas programuoti loginių matricų lustus (*angl. FPGA, CPLD ir t.t.*), taip pat su juo galima ir susimuliuoti lustams sukurtąsias programas.

Realizuoti kuriamą sistemą galima trim būdais:

1.) vartotojas pasitelkęs siūlomomis VHDL ar VERYLOG kalbomis pats rašo sistemos kodą.
2.) sistemą projektuojant ir sujungiant iš atskirų elementų, kur kiekvienas elementas atlieka numatytą funkciją.
3.) sistemą projektuojant ir sujungiant iš atskirų programinių bloku, kur kiekvienas programinis blokas atlieka numatytą funkciją.

Kiekvienas iš aukščiau paminėtų sistemos kūrimo būdų turi savo trūkumų ir privalumų. Sistemos realizavimas pirmuoju būdu reikalauja iš programuotojo, atsižvelgiant į sistemos sudėtingumą, daug specifinių žinių apie komponentų funkcijų veikimą, tarpusavio sąveiką, bei labai gerai mokėti specifinę aparatinės dalies aprašymo kalbą (VHDL arba VERILOG). Taip pat toks dabas labai imlus laikui, o įvėlus klaidą, sistemos kūrimas gali tapti labai ilgas ir sudėtingas uždavinys. Tačiau tokios sistemos kūrimo būdas leidžia tiksliai ir optimaliai suprojektuoti norimą sistemą, sunaudojant minimalų lusto resursų kiekį. Pirmasis būdas labiausiai paplitęs tarp labai patyrusių programuotojų.

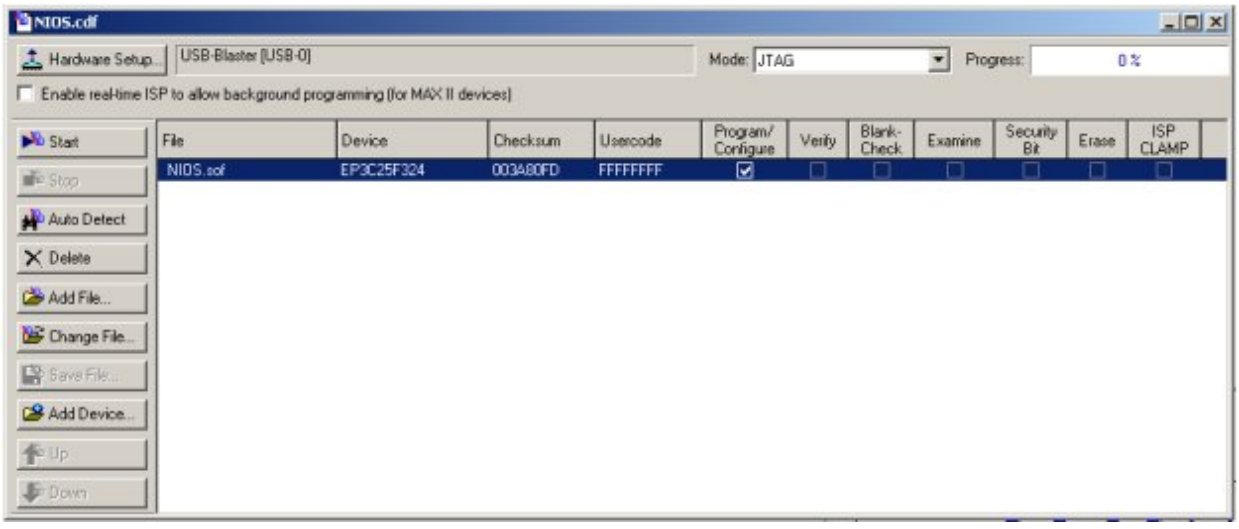
Sistemą kuriant antruoju būdu yra sukuriamas *.bdf (block diagram file) plėtinio failas, kuriame galima sistemos funkcijas ir elementus sudaryti iš atskirų loginių elementų, kurie yra aprašyti ir patalpinti į Quartus II standartinėje elementų bibliotekoje. Antrasis būdas labai patogus nesudėtinguose sistemose, nes vizuoliai matoma visa sistemos programa, kur matomi loginiai(ir kiti) elementai ir sujungimai tarp jų. Tai taip pat leidžia tiksliai ir optimaliai suprojektuoti norimą sistemą,

sunaudojant minimalų lusto resursų kiekį. Tačiau antrasis būdas buri prieš tai buvusius pirmojo būdo trūkumus, t.y. Darant sudėtingą sistemą reikia daug žinių, kad iš loginių elementų realizuoti sistemos funkcijas, taip pat didelė klaidų tikimybė.

Trečiuoju būdu kuriant sistemą galima pasitelkti specialius „Quartus II“ siūlomus įrankius „Qsys“ arba „SOPC Builder“. Šiuose įrankiuose yra bibliotekos, kuriuose aprašyti NIOS II procesorius, taip pat procesoriui reikalingos atmintys, įvairios sąsajos ir kiti elementai be kurių sistema su procesoriumi negalėtu tinkamai veikti. Pasinaudojant vieną iš minėtų įrankių galima sujungti sistemos elementus į vieną visumą. Taip kuriant sistemą sutaupoma daug laiko dėl įrankių lankstumo ir paprastumo naudotis, tereikia pasirinkti pageidaujamą komponentą, pagal turimą užduoti pakeisti jo parametrus ir išsaugoti į kuriamos sistemos failą. Sekantis žingsnis – sistemos generavimas taip gaunant aparatinės įrangos parašytas sistemos failas, kurį esant reikalui galima koreguoti priklausomai nuo tolimesnio sistemos pritaikymo.

Įvertinant visų 3-jų sistemos kūrimo būdų privalumus ir trūkumus, buvo pasirinktas trečiasis būdas – sistemą kurti su įrankių „QSYS“. Tokį pasirinkimą lėmė sistemos kūrimo aiškumas ir paprastumas, tai leido sutaupyti nepalyginamai daugiau laiko ir išvengti klaidų, lyginant su pirmuoju ir antruoju būdu, taip pat sukurtieji komponentai gali būti panaudoti kituose projektuose.

Sėkmingai sukompiliavus visą projektą, prie USB prievado galima prijungti DE2 bandymų plokštę ir įkrauti kompiliavimo rezultate gautą failą su „sof“ plėtiniu (3.1 paveikslas).



3.1 pav. Programos įkrovos lango bendras vaizdas

3.2 NIOS II BSP Eclipse

Programinė įranga susideda iš atskirų programinės įrangos komponentų rinkinių, panaudojanti „Eclipse“ aplinką ir „Eclipse“ C/C++ programinės įrangos rinkinių.

„Eclipse“ kaip atvirojo kodo programinė įranga buvo sukurta 2001 m. „IBM“ kompanijoje. Su šia aplinka kuriami projektai yra sutelkti į atviro vystymo platformos kūrimą, kuri susidedantį iš išplečiamų struktūrų ir įvairių įrankių, taip kuriant ir valdant programinę įrangą.

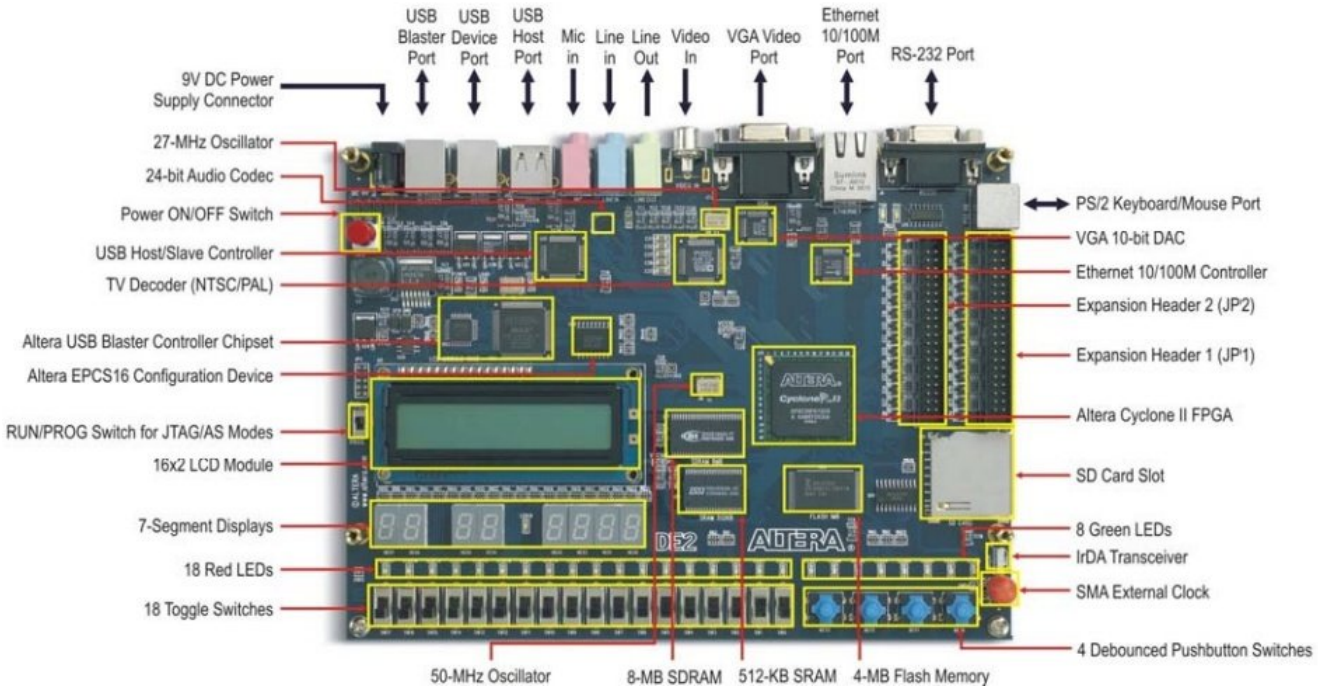
„Eclipse“ C/C++ programinės įrangos paketas visiškai užtikrina C ir C++ programavimo kalbų palaikymą, paremtą „Eclipse“ platforma. Pasinaudojant paminėta programavimo aplinka, parašytas kodas yra paverčiamas į mašininį kodą[17].

„NIOS II BSP Eclipse“ dirba su visomis NIOS II procesoriaus sistemomis taip užtikrinant nuoseklią plėtojimo platformą vartotojui.

Pradedant kurti naują „Eclipse“ projektą reikia nurodyti „.sopcinfo“ plėtinio failą, kuriame yra informacija apie sukurtą sistemą su procesoriumi. Šiame faile yra nurodytas kiekvienas panaudotas komponentas, parametrų vertės ir vardai. Projekte gali būti panaudotos visos standartinės C/C++ bibliotekos, bei specifinės funkcijos, sukurtos programuoti LPLM įrenginius. Parašius programos kodas, toliau seka programos kompiliacija, gavus patvirtinimą, kad programa sukompiliuota be klaidų, tada galima mašininį kodą įkrauti į sistemą, taip paleidžiant jos darbą.

3.3 DE2 bandymų plokštė

Sistema buvo testuojama kompanijos „Altera“ sukurtoje DE2 bandymų plokštėje, kuri pavaizduota 3.2 pav.[18]

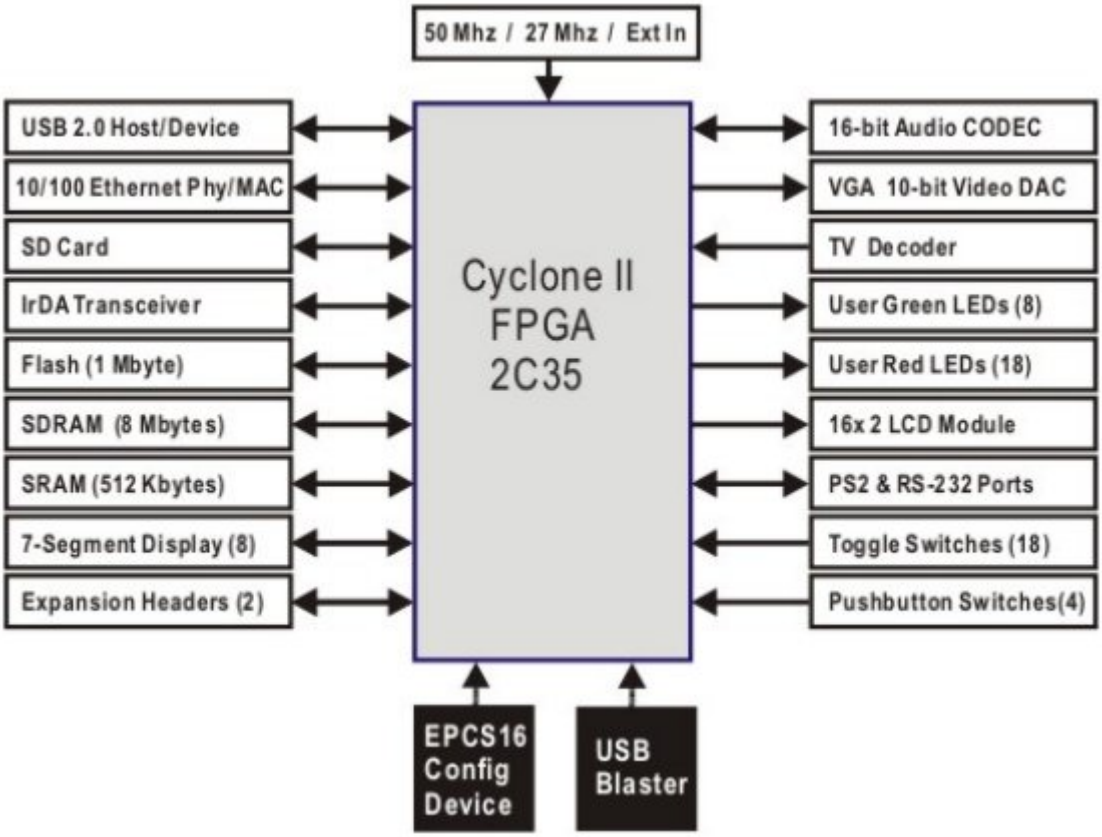


3.2 pav. „Altera“ DE2 bandymų plokštė[18]

Plokštėje integruota Cyclone II EP2C35F672C6 LPLM, tai pagrindinis plokštės komponentas, plokštėja taip pat integruoti:

• USB Blaster ir JTAG programavimo sąsajos;
• 512-K bitų SRAM atminties;
• 8-M bitai SDRAM atimties;
• 4-Mbitai Flach atminties;
• SD kortelės lizdas;
• 4 paspaudžiami nefiksuojamos padėties mygtukai;
• 18 fiksuojamos padėties jungiklių ;
• 18 raudono LED lemputės;
• 9 žalio LED lemputės;
• 50-MHz s ir 27-MHz generatoriai;
• VGA SAK (10-bitų) VGA-jungtis;
• 16x2 LCD modulis;

Plokštės blokinė diagrama pateikta 3.3 paveiksle. Iš plokštės blokinės diagramos galima spręsti, kad plokštė turi platų pritaikymą kuriant projektus ar atliekant bandymus.



3.3. pav. DE2 plokštės blokinė diagrama[18]

3.4. D5M 5 megapikselių skaitmeninė kamera

D5M skaitmeninė vaizdo kamera yra 5 megapikselių skiriamosios gebos, tai „terasic“ kompanijos produktas. Vaizdo kameros vaizdas pateiktas 3.4 paveiksle. Tokia tipo kamera panaudota tiriamajame darbe. Vaizdo kamera prie DE2 plokštės yra prijungiama per 40 –ties kontaktų jungtį. Tokia jungtis užtikrina greitą ir patogų įrenginių sujungimą[19].

3.4 pav. D5M 5 megapikselių skaitmeninė kamera[19]

D5M vaizdo kamera turi:

- CMOS 5 megapikselių foto jutiklis;
- RGB „Bayer“ spalvos filtrą;
- Kadro dydis VGA(640 x 480);
- Pikselio dydis 2.2µm x 2.2µm x.
- 40 kontaktų jungtį.

4. Koordinačių sekimo sistemos sudarymas ir bandymai

Koordinačių sekimo sistemos duomenų apdorojimas yra realizuotas dviem būdais: aparatūriškai(angl. hardware) ir programiškai(angl. software).

4.1 Aparatūrinis duomenų apdorojimas

Aparatūrins duomenų apdorojimas yra realizuotas „Quoartus“ programos „Qsys“ įrankiu. Kuriant vaizdo duomenų apdorojimo konvejerį buvo pasitelktos dvi funkcinių blokų bibliotekos. Biblioteka susideda iš „Qsys“ programos siūlomos bibliotekos ir iš „University Program“ bibliotekos. Ši biblioteka automatiškai įdiegiama į „Qsys“ įrankį, kai yra suinstaliuojamas „Altera“ laisvai platinamas „altera_upd_s“ programisnis paketas.

Bendras aparatūrinio duomenų apdorojimo konvejerio vaizdas pateiktas 4.1 paveiksle.

Use	C...	Name	Description	E...	Clock
☑		**CPU**	Nios II Processor		**sys_clk**
☑		**sysid**	System ID Peripheral		**sys_clk**
☑		**merged_resets**	Reset Bridge		
☑		**sys_clk**	Clock Bridge		**sys_clk**
☑		**vga_clk**	Clock Bridge		**Clock_Signals_vga_clk**
☑		**SDRAM**	SDRAM Controller		**sys_clk**
☑		**SRAM**	SRAM/SSRAM Controller		**sys_clk**
☑		**JTAG_UART**	JTAG UART		**sys_clk**
☑		**Interval_Timer**	Interval Timer		**sys_clk**
☑		**clk**	Clock Source		
☑		**External_Clocks**	Clock Signals for DE-series Board Peri...		**clk**
☑		**Onchip_memory**	On-Chip Memory (RAM or ROM)		**Clock_Signals_sys_clk**
☑		**Clock_Signals**	Clock Signals for DE-series Board Peri...		**clk**
☑		**AV_Config**	Audio and Video Config		**Clock_Signals_sys_clk**
☑		**video_decoder_0**	Video-In Decoder		**Clock_Signals_sys_clk**
☑		**video_bayer_resamp...**	Bayer Pattern Resampler		**Clock_Signals_sys_clk**
☑		**video_clipper_0**	Clipper		**Clock_Signals_sys_clk**
☑		**VGA_Pixel_Scaler**	Scaler		**Clock_Signals_sys_clk**
☑		**video_rgb_resample...**	RGB Resampler		**Clock_Signals_sys_clk**
☑		**Koordinaciu_selekto...**	Koordinaciu_selektorius		**Clock_Signals_sys_clk**
☑		**video_rgb_resample...**	RGB Resampler		**Clock_Signals_sys_clk**
☑		**video_dma_controlle...**	DMA Controller		**Clock_Signals_sys_clk**
☑		**sram_0**	SRAM/SSRAM Controller		**Clock_Signals_sys_clk**
☑		**Char_LCD_16x2**	16x2 Character Display		**Clock_Signals_sys_clk**
☑		**CPU_fpoint**	Floating Point Hardware		
☑		**clk_27**	Clock Source		

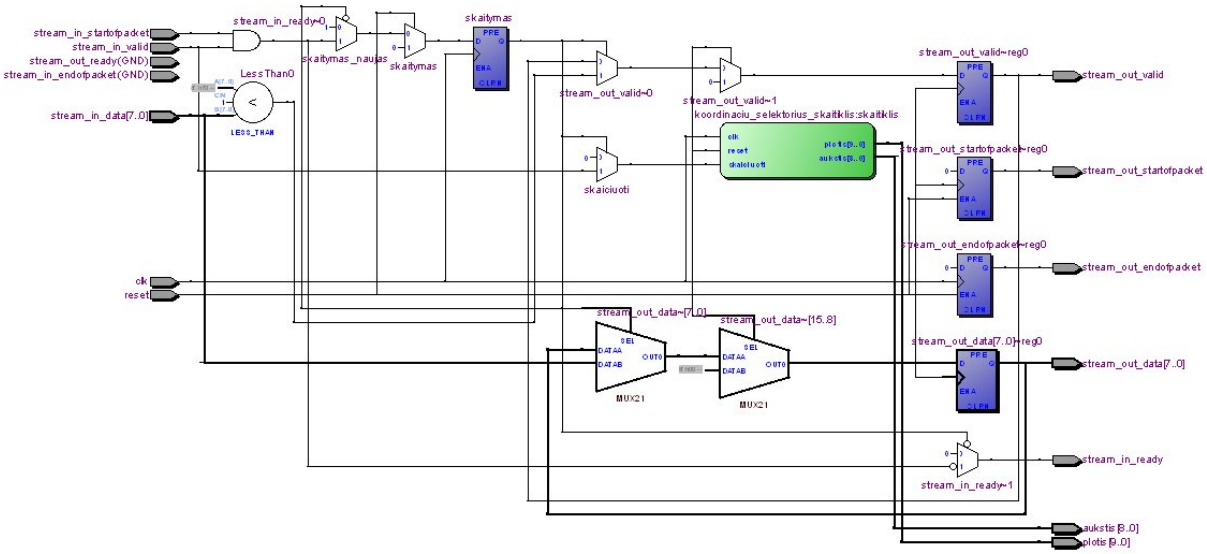
4.1 pav. vaizdo apdorojimo konvejeris

Toliau bus detaliau apžvelgti 4.1 pav. sistemos elementai, bei jų atliekamos funkcijos:

- **CPU** – tai 32 bitų NIOS II procesorius. Procesoriaus vidinis konvejeris yra parinktas „NIOS II/f“ – tai greičiausias siūlomas konvejerio variantas, toks procesoriaus branduolys LPLM sunaudoja 1800 loginių blokų. Kai vaizdas apdorojamas aparatūriniu būdu procesorius jokių skaičiavimų neatlieka. O kai vaizdas apdorojamas programiškai, visas kadro apdorojimo operacija įskaitant ir vaizdo nuskaitymu iš SRAM atminties atlieka procesorius. Taip pat procesorius dar formuoja pertrauktis.
- **sys_clk** – tai taktinis impulsų generatorius. Elemento generuojamas dažnis – 50MHz. Šiuo dažniu yra sinchronizuojamas beveik visų funkcinių blokų darbas sistemoje;
- **External_Clocks** – Tai išorinis signalų šaltinis kuris formuoja „reset“ signalus sistemos funkciniams blokams;
- **AV_Config** – tai funkcinis blokas kuris nurodo video šaltinį DE2 plokštėje;
- **Video_decoder** – blokas dekoduoja ir sinchronizuoja vaizdo šaltinio taktinį dažnį su sistemos taktiniu dažniu, taip pat gautasis vaizdas yra konvertuojamas į „Altera University“ programos duomenų paketus;
- **Video_bayer_resampler** – blokas konvertuoja duomenų srautą į „Bayer“ RGB 24 bitų spalvų modelį, taip gaunamas spalvotas vaizdas;
- **Video_clipper** – blokas leidžia modifikuoti vaizdo skiriamąją gebą taip pat apkarpyti kadro eilutes ir stulpeliu;
- **Video_scaler** – bloku parenkamas įeinantis kadro dydis ir po to kadaras yra padalinamas į 2 dalis taip taip sumažinamas duomenų kiekis kuris vienu kartu bus talpinamas į atmintį;
- **Video_rgb_resampler_0** – blokas konvertuoja video duomenų srautą tarp RGB spalvų erdvės formato, taip pat blokas konvertuoja duomenų srautus tarp skirtingų funkcinių blokų, šio atveju yra konvertuojami vaizdų srautai tarp „**Video_scaler**“ ir „**Video_dma_controller**“, atitinkamai duomenų srautai yra 24 bitai ir 8 bitai ;
- **Video_dma_controller** – blokas valdo duomenų įrašymą į SRAM atmintinę ir skaitymą iš jos.
- **Sram_0** – Tai atmintis į kurią yra talpinamas apdorotasis vaizdas;
- **Char_LCD_16x2** – tai blokas realizuojantis LCD ekranėlį esantį DE2 plokštėja, į ekranėli yra išvedinėjamos lazerio taško koordinatės.

- **Koordinačių selektorius** – tai funkcinis blokas kuriuo apdorojami vaizdo kadrai ir nustatoma lazerio taško koordinatės kadre.
- **CPU_fpoint** – tai slankaus kablelio modulis, kad procesorius galėtu atlikti matematines operacijas su kableliu.
- **Interval_Timer** – Tai vidinis procesoriaus laikmatis kurio galima matuoti operacijų laiką. Modulis valdomas programiškai, t.y. laiko reikšmės apdorojamos procesoriaus.

„Koordinačių selektorius“ (4.2 pav.) tai elementas kurį reikėjo susikurti, nes „Qsys“ bibliotekoje tokio elemento nebuvo.



4.2 pav. Koordinačių selektoriaus schema

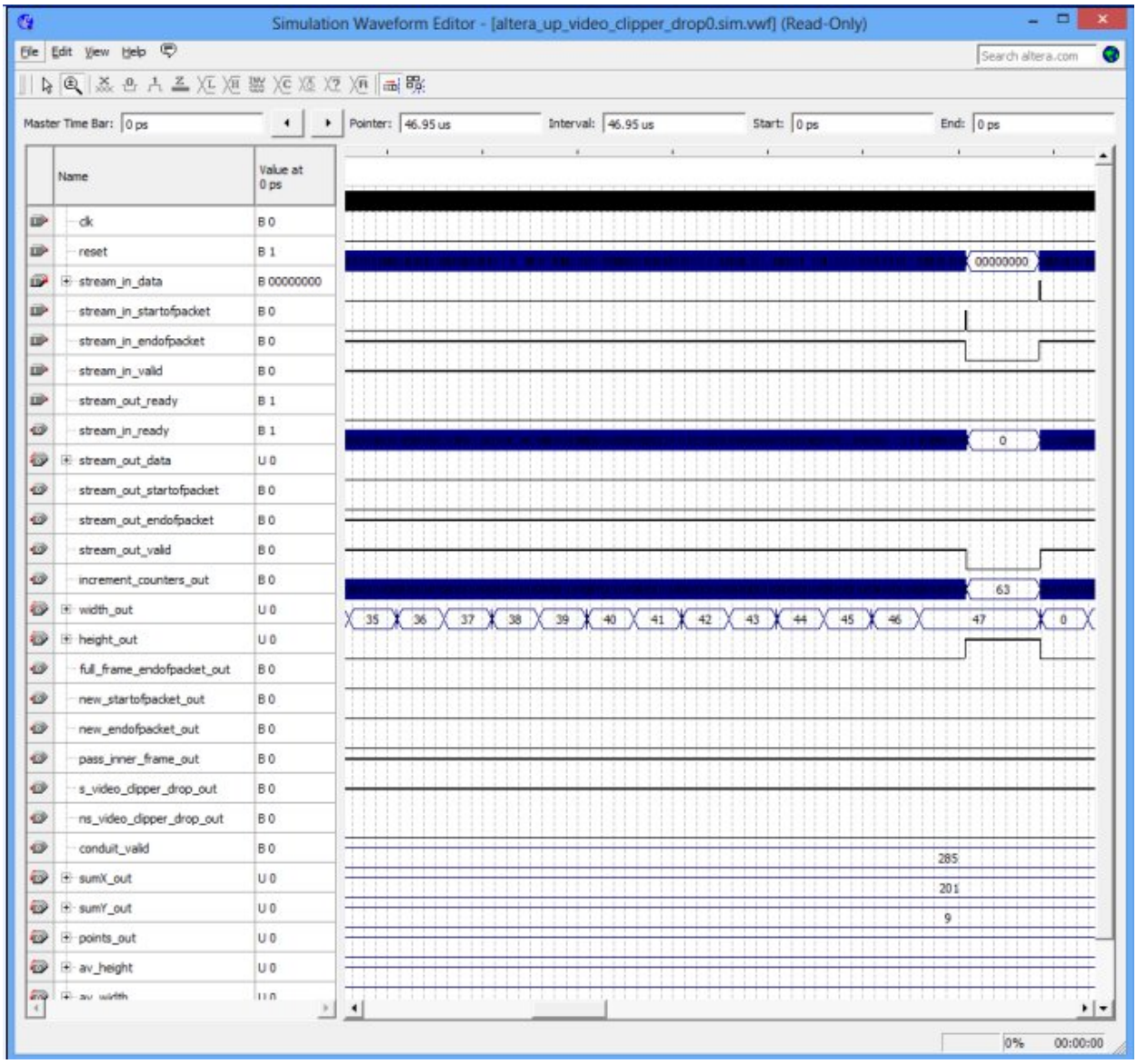
„Koordinačių selektorius“ blokas – aprašytas Verilog kalba, ir su specialiu „Qsys“ redaktoriumi integruotas į „Qsys“ įranki.

Vaizdo apdorojimo laikas buvo išmatuotas programinio modeliavimo būdu. Apdorojant 640x480 pikselių dydžio vaizdą aparatūriniu būdu, kadro apdorojimo trukmė buvo 6,2 ms

„Koordinačių selektorius“ dirba taip – skenuojamas pilko vaizdo kadras nuo pradžios iki pabaigos. Kiekvieno vaizdo kadro pikselio reikšmė yra lyginama su atramine reikšme. Pikselis kurio reikšmė yra mažesnė už atraminę yra praleidžiamas ir skenuojamas sekantis pikselis. Jeigu pikselio reikšmė yra lygi arba didesnė už atraminę, tada kintamojo *points_out* reikšmė padidinama vienetu Susumavus tokių kadro pikselių „x“ ir „y“ koordinates gaunamos koordinačių „x“ ir „y“ sumos. Norint apskaičiuoti lazerio taško „x“ ir „y“ koordinates, reikia „x“ koordinatės pikselių sumą padalinti

iš kintamojo *points_out* vertės. Analogiški veiksmai atliekami ieškant „y“ koordinatęs. Modeliavimo rezultatai pavaizduoti 4.3 paveiksle.

Tokiu algoritmu apdorojant kiekvieną kadrą gaunamos lazerio taško koordinatės, kurių reikšmė yra išvedama į LCD 16x2 ekranėli, kuris yra DE2 plokštėje.



4.3 pav. Koordinačių selektoriaus įtaiso modeliavimo rezultatai

Reikia pažymėti, kad kai koordinačių reikšmės apdorojamos aparatūriniu būdu, koordinačių reikšmės į LCD ekraną taip pat išvedamos programiškai, tai atlieka NIOS II procesorius. Koordinačių reikšmės irašomos į SRAM atmintį, o iš SRAM atminties NIOS II procesorius nuskaito koordinačių reikšmes ir nusiunčia į LCD ekraną.

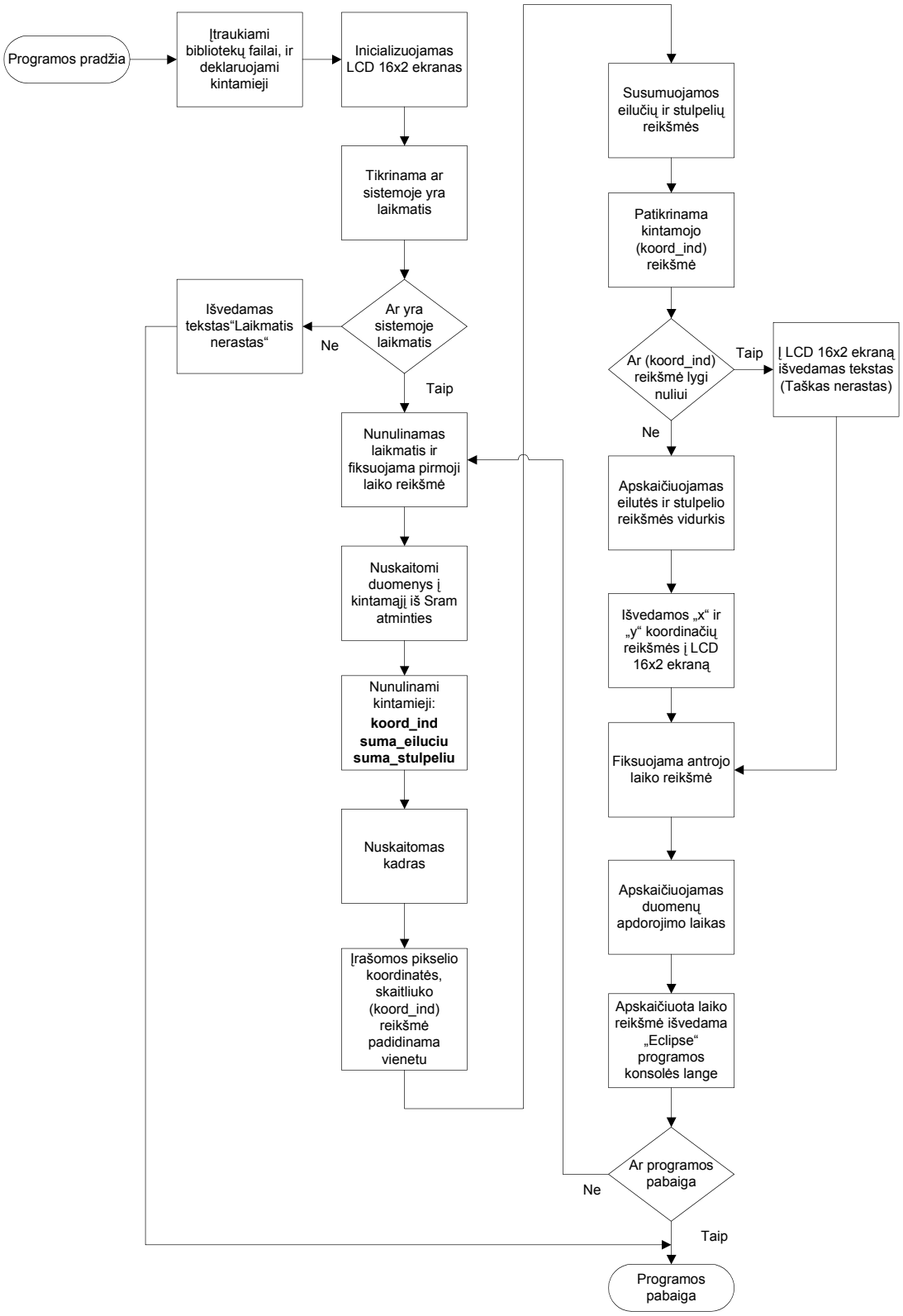
4.2 Programinis duomenų apdorojimas

Apdorojant duomenis programiniu būdu (programiškai), atliekami šie veiksmai: kadro nuskaitymas iš SRAM atminties, kadrų filtracija, koordinačių radimas kadruose, matuojamas anksčiau išvardintu operacijų laikas , koordinačių reikšmių išvedimas į LCD ekraną. Įeinančio vaizdo formavimas ir įrašymas į SRAM atmintį kaip ir anksčiau yra atliekamas aparatūriniu būdu.

Kai duomenys apdorojami programiškai, apdorojimas vykdomas NIOS II procesoriuje. Programos kodas procesoriui parašytas su „Eclipcse“ programa, panaudojant C programavimo kalbą.

Kadro filtravimo ir koordinačių radimo vaizdo kadre logika išlieka analogiška aparatūrinio apdorojimo būdui. Procesorius NIOS II nuskaito iš duomenų buferio video kadrą, jį apdoroja programiškai ir gautas koordinačių reikšmes, t.y. „x“ ir „y“ atvaizduoja LCD 16x2 ekrane.

4.4 paveiksle pateiktas programos algoritmas, vaizdą apdorojant, programinių būdu.



4.4 pav. Programos algoritmas

Pirmą kartą startavus programai (4.4 pav.) į programos vykdymą yra įtraukiami bibliotekos failai. Bibliotekos failuose aprašytos funkcijos, kurios naudojamos programoje: *printf*, *alt_up_character_lcd_write, alt_timestamp_start* ir kitos.

Toliau yra inicializuojamas LCD 16x2 ekranas. Inicializacija reikalinga, kad būtų leista naudotis LCD ekranu. Sekantis žingsnis patikrinti ar sistemoje yra laikmatis. Laikmatis realizuoja funkcija kurios pagalba yra fiksuojamas sistemos operacijų vykdymo laikas. Jai laikmačio programa neranda yra generuojamas pranešimas „Laikmatis nerastas“, pranešimas išvedamas „Eclipse“ konsoles lange ir programos vykdymas stabdomas. Laikmatis yra įgyvendintas aparatūriniu būdu „Qsys“ įrankiu.

Gavus patvirtinamą, kad sistemoje yra laikmatis, laikmačio registras yra nunulinamas ir pradedamas skaičiuoti laikas. Startavus laikmačiui toliau nuskaitomas kadras į kintamąjį iš Sram atminties, Sram pradinis atminties adresas yra 0x08000000, o jos dydis 512KB.

Kintamųjų: *koord_ind, sums_eiluciu, suma_stulpeliu*, nunulinimas reikalingas, kad kaskart nuskaitant naują kadrą būtu iš naujo perskaičiuojamos koordinačių sumos reikšmės.

Kadro nuskaitymo programos kodas fragmentas pateiktas 4.5 paveiksle.

```
//Nufiltruojamas kadras pagal parinkta slenkstine reiksme
//ir irasomos pikseliu koordinates kuriu vetre buvo lygi arba didesne uz slenkstine reiksme
for(eilutes=0;eilutes<240;eilutes++){
    for(stulpeliai=0;stulpeliai<640;stulpeliai++){
        indeksas=eilutes*640+stulpeliai;
        if(*(pSDRAM+indeksas) > 240)
        {
        kord_masyvas[koord_ind][0]=eilutes;
        kord_masyvas[koord_ind][1]=stulpeliai;
        koord_ind++;
        }
    }
}
```

4.5 pav. Kadro nuskaitymas

Kadras nuskaitomas po pikseli nuskaitant stulpelius ir eilutes, kai eilutėje nuskaitomi visi stulpeliai, pereinama į sekančią eilutę ir vėl skaitomi stulpeliai. Kadro skaitymo metu lyginama kiekvieno pikselio reikšmė su nustatyta atramine reikšme, kuri šiuo atveju yra 240. Atraminė reikšmė parinkta eksperimentiškai, kadangi apdorojamo kadro pikselio maksimali reikšmė yra 255 (t.y 8 bitai), ši maksimali reikšmė interpretuojama kaip baltas (šviesiausias) taškas.

Pikselio, kurio reikšmė didesnė už atraminę reikšmę, koordinatės yra įsimenamos, o pikselis kurio reikšmė mažesnė už atraminę reikšmę yra praleidžiamas ir nuskaitomas sekantis pikselis. Taip

nuskenavus visą kadrą gaunamos lazerio taško koordinačių reikšmių masyvas. Susumavus visas koordinačių reikšmes pagal „x“ ir „y“ koordinates (4.1 ir 4.2 formulės) ir apskaičiavus jų vidurkius (4.3 ir 4.4 formulės) gaunama absoliutinės lazerio taško koordinatės kurios išvedamos į LCD 16x2 ekraną.

$$x = \sum x_n \tag{4.1}$$

$$y = \sum y_n \tag{4.1}$$

čia

x – x koordinatės suma;

y – y koordinatės suma;

x_n – n-tasis x koordinatės elementas;

y_n – n-tasis y koordinatės elementas.

$$x_{vid} = \frac{x}{n} \tag{4.3}$$

$$y_{vid} = \frac{y}{n} \tag{4.3}$$

čia n – *koord_ind* reikšmė.

Kintamasis *koord_ind* skaičiuoja kiek gauta koordinačių reikšmių apdorojant kadrą. *koord_ind* kintamojo reikšmės yra naudojamos dviem loginėms operacijoms atlikti:

- Kai *koord_ind* reikšmė nelygi nuliui kintamojo reikšmė naudojama kaip daliklis, apskaičiuojant koordinačių vidurkius;
- Kai *koord_ind* reikšmė lygi nuliui, tada interpretuojama, kad kadre nebuvo aptiktas lazerio taškas ir į LCD ekraną išvedamas pranešimas „Taškas nerastas“.

Po koordinačių išvedimo į LCD ekraną yra nuskaitoma antroji laikmačio reikšmė. Tai interpretuojama, kaip kadro apdarojimo pabaiga. Iš gautųjų dviejų laikmačio reikšmių yra apskaičiuojamas laikas, per kurį buvo apdorojamas kadras. Apskaičiuotoji laiko reikšmė yra išvedama į „Eclipse“ konsolės langą. Po laiko reikšmės išvedimo, programos ciklas grįžta atgal į pradžią, kur nunulimanas laikmatis, taip pradedamas naujas kadro apdorojimo ciklas.

Atlikus vaizdo apdorojimo programiškai bandymą, pastebėta , kad kadro apdorojimas vidutiniškai truko 0.638099 sekundės, tai sistema praktiškai per sekundę apdoroja mažiau nei pusantro

kadro. 0.638099 sekundės kadras apdorojamas pasirinkus NIOS II/f‘ – tai greičiausias siūlomas konvejerio variantas.

Išvados

1. Sistema buvo testuojama dviem būdais. Vaizdo, kurio dydis 640x480 pikselių, apdorojimas atliktas programiniu ir aparatūriniu būdu. Vaizdą apdorojant programiniu būdu kadro apdorojimo trukmė – 0.638 sekundės arba 638 ms, o vaizdą apdorojant aparatūriniu būdu kadro apdorojimo trukmė – 6,2 ms.

2. Vaizdą apdorojant aparatūriniu būdu kadro apdorojimo greitis yra beveik 100 kartų didesnis, negu vaizdą apdorojant programiniu būdu t.y. kai vaizdą apdoroja NIOS II procesorius.

3. Palyginus gautus rezultatus matyti, kad sistema, kuri koordinačių skaičiavimus atlieka ne įkeliamame procesoriuje, o specialiai sukurtame aparatiniame įtaise, yra pranašesnė už sistemą, kurioje skaičiavimus atlieka vien įkeliamasis procesorius.

4. Sistemas ištestavus ant bandymų plokštės DE2 galima patvirtinti, kad LPLM lustas patogiai yra konfigūruojamas pagal vartotojo poreikius. Nustatyta, kad patogiausias būdas kurti sistemas – pasinaudoti kompanijos "Altera" sukurta programine įranga – „Quartus II" ir specialiu įrankiu „Qsys", panaudojant „Qsys" ir „University Programme" bibliotekas. Nesant reikiamo komponento „Qsys" bibliotekoje, reikalingą elementą galima susikurti, aprašant jį VHDL arba Verilog kalba. Tokiu būdu sistema pasižymi universalumu ir lankstumu.

Literatūra

1. Human Eye [interaktyvus], [Žiūrėta 2013-01-23]. Prieiga per internetą: <http://www.wiley-vch.de/books/sample/3527403809_c01.pdf>.
2. Electro Magnetic Spectrum and light[interaktyvus] , [Žiūrėta 2013-01-23]. Prieiga per internetą: <http://9-4fordham.wikispaces.com/Electro+Magnetic+Spectrum+and+light>.
3. Vitkauskienė R. Žmoguas akies dažninės skiriamosios gebos tyrimas[interaktyvus].Šiauliai, Šiaulių Universitetas,2011, 57 psl.[Žiūrėta 2013-02-25]. Prieiga per internetą: <http://vddb.laba.lt/fedora/get/LT-eLABa-0001:E.02~2011~D_20111103_091502-40831/DS.005.0.01.ETD>.
4. Kazakevičius T. Greitas ir tikslus objekto parametrų nustatymas mašininės regos sistemose [interaktyvus].Vilnius, Vilniaus Technikos Universitetas,2011, 53 psl.[Žiūrėta 2013-02-15]. Prieiga per internetą: < http://vddb.laba.lt/fedora/get/LT-eLABa-0001:E.02~2011~D_20110610_160456-77470/DS.005.0.02.ETD>.
5. Grigaitis D. Programuojamų loginių matricų pradžiamokslis[interaktyvus]. Vilnius, 2009, 102 psl. [Žiūrėta 2013-02-25]. Prieiga per internetą: <http://www.grigaitis.eu/zip/pradziamokslis.pdf>.
6. FPGA – 911[interaktyvus], [Žiūrėta 2013-02-25]. Prieiga per internetą: <http://www.vlsiegypt.com/home/?p=226>.
7. FPGA Fundamentals[interaktyvus], [Žiūrėta 2013-02-25]. Prieiga per internetą: <http://www.ni.com/white-paper/6983/en>.
8. Embedded Soft Processors // Altera Corporation [interaktyvus]. San Jose (California), USA, 2011, gegužė [žiūrėta 2012-03-19]. Prieiga per internetą: <http://www.altera.com/devices/processor/faq/emb-faq-soft-processor.html#niosii >.
9. Nios II Processor Reference // Corporation [interaktyvus]. San Jose (California), USA, 2011, gegužė [žiūrėta 2013-04-10]. Prieiga per internetą:< http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf>.
10. Ravi S. Operators used in edge detecion computation [interaktyvus]. Mangalore University,2012, 6 psl.[Žiūrėta 2013-04-15]. Prieiga per internetą: < http://gimt.edu.in/clientFiles/FILE_REPO/2012/NOV/24/1353740646160/180.pdf >. ISSN 0973-4562.
11. Trost A.,Baldomir Z. Desing of real-time edge detection cicrcuit on multi-FPGA prototyping system [interaktyvus]. Ljubljana, Slovenija: University of Ljubljana - Department of Electrical Engineering, [žiūrėta 2013-05-23]. Prieiga per internetą: < http://www.emo.org.tr/ekler/f20fe72886d5849_ek.pdf >.
12. R.C.K Hua, L.C. De Silva and P. Vadakkepat. Detection and Tracking of Faces in Real-Time Environments [interaktyvus]. Singapore: University of Singapore- Department of Electrical and Computer Engineering, [žiūrėta 2013-05-23]. Prieiga per internetą: < http://www.ece.nus.edu.sg/showcase/detection%20and%20tracking%20of%20faces%20in%20real-time%20environments.pdf>.

13. Daunys G. Įterptinės mašininės regos sistemos. TEV, 2012, 113 p. e-ISBN 978-609-433-098-8

14. Kuon I. FPGA architecture: Survey and Challenges/R.Tessier, J.Rose//Foundations and Trands in Electronic Design Automation [interaktyvus]. Hanover (Massachusetts), USA, 2007, nr. 2, p.135-253 [žiūrėta 2012-03-18]. Prieiga per internetą: <http://inst.eecs.berkeley.edu/~cs294-59/fa10/resources/kuon-book.pdf>. ISSN-1551-3939.
15. Ho Gi Jung. Light Stripe Projection based Parking Space Detection for Intelligent Parking Assist System: Dong Suk Kim, Pal Joo Yoon and Jaihle Kim[interaktyvus]. Turkey, birželis 13-15, 2007 [žiūrėta 2013 -05-20]. Prieiga per internetą: < https://www.google.lt/url?sa=t&rct=j&q=&esrc=s&source=web&cd=4&sqi=2&ved=0CE0QFjAD&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.159.8665%26rep%3Drep1%26type%3Dpdf&ei=ciuNUeb6NMXJswbe4oCQCQ&usg=AFQjCNFsa8x6FeveVBwKfojFYLJ1GNHLmg&sig2=G7Vhm74Vf5FUum32WaUPXA>.
16. Bailey D. G. Design for embedded image processing on FPGAs. New Zealand: Massey University, John Wiley & Sons. 2011, p.482. ISBN: 978-0-470-82849-6.
17. Eclipse Fundation [interaktyvus], [žiūrėta 2013 -05-10]. Prieiga per internetą:< http://www.eclipse.org/org/>.
18. DE2 Development and Education Buord// // Altera Corporation [interaktyvus]. 2006, [žiūrėta 2013-04-10]. Prieiga per internetą:< ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf>.
19. TRDB_D5M// // Altera Corporation [interaktyvus]. 2008, [žiūrėta 2013-04-10]. Prieiga per internetą:< http://www.cs.washington.edu/education/courses/cse467/08au/labs/Resources/TRDB_D5M_UserGuide.pdf>.

Priedai

1 Priedas. Kompaktinis diskas.

Kompaktinio disko turinys

1. Baigiamasis magistro darbas;
2. Eclipse programos kodas;
3. Qsys biblioteka (koordinačių selektorius).