VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

Tomyslav SLEDEVIČ

OBJEKTŲ SEKIMO VAIZDE ALGORITMŲ ĮGYVENDINIMO LPLM ĮRENGINIU TYRIMAS

INVESTIGATION OF THE OBJECT TRACKING ALGORITHM BASED ON FPGA

Magistro baigiamasis darbas

Elektronikos inžinerijos studijų kryptis
Elektronikos studijų programa, valstybinis kodas 621H61003
Kompiuterizuotų elektroninių sistemų specializacija

Vilnius, 2012

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

_____
(*parašas*)
prof. habil. dr. R. Martavičius
2012 m. _____ mėn. ___ d.

Tomyslav SLEDEVIČ

OBJEKTŲ SEKIMO VAIZDE ALGORITMŲ ĮGYVENDINIMO LPLM ĮRENGINIU TYRIMAS

INVESTIGATION OF THE OBJECT TRACKING ALGORITHM BASED ON FPGA

Magistro baigiamasis darbas

Elektronikos inžinerijos studijų kryptis
Elektronikos studijų programa, valstybinis kodas 621H61003
Kompiuterizuotų elektroninių sistemų specializacija

Vadovas	dr. A. Serackis	_____	_____
	(*mokslinis vardas ir laipsnis, vardas, pavardė*)	(*parašas*)	(*data*)
Konsultantas	asist. D. Macko	_____	_____
	(*mokslinis vardas ir laipsnis, vardas, pavardė*)	(*parašas*)	(*data*)

Vilnius, 2012

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

ELEKTRONINIŲ SISTEMŲ KATEDRA

Technologijos mokslų sritis	TVIRTINU
Elektros ir elektronikos inžinerijos mokslo kryptis	Katedros vedėjas
Elektronikos inžinerijos studijų kryptis	_____
Elektronikos studijų programa, valst. kodas 621H61003	prof. habil. dr. R. Martavičius
Kompiuterizuotų elektroninių sistemų specializacija	2012 m. _____ mėn. ____ d.

MAGISTRO BAIGIAMOJO DARBO UŽDUOTIS

Studentui Tomyslav SLEDEVIČ, EKSfm-10 gr.

Baigiamojo darbo tema: Objektų sekimo vaizde algoritmų įgyvendinimo LPLM įrenginiu tyrimas

Investigation of Object Tracking Algorithms Based on FPGA

Patvirtinta 2011 m. _____ mėn. ____ d. dekano įsakymu Nr. _____.

Baigiamojo darbo užbaigimo terminas 2012 m. _____ mėn. ____ d.

Darbo tikslas – LPLM įrenginyje įgyvendinti ir ištirti objektų sekimo vaizde algoritmus.

Darbo uždaviniai

1. Apžvelgti objektų sekimo vaizde algoritmus bei jų įgyvendinimo galimybes LPLM įrenginyje.
2. LPLM įrenginyje įgyvendinti objektų sekimo vaizde algoritmus.
3. Ištirti objektų sekimo vaizde algoritmus.

Aiškinamojo rašto turinys

1. Įvadas
2. Analitinė literatūros apžvalga
3. Algoritmų objektams vaizde sekti kūrimas ir įgyvendinimas LPLM įrenginyje
4. Objektų sekimo algoritmų tyrimas
5. Apibendrinimas. Išvados

Literatūra

Priedai

A priedas. CD su magistro baigiamojo darbo aiškinamuoju raštu

B priedas. Pranešimo 15-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga

Baigiamojo darbo rengimo konsultantas: –

(*mokslinis vardas ir laipsnis, vardas, pavardė*)

Vadovas dr. Artūras Serackis

(*parašas*) (*mokslinis vardas ir laipsnis, vardas, pavardė*)

Užduotį gavau 2012 m. _____ mėn. ____ d.

(*parašas*)

Vilniaus Gedimino technikos universitetas
Elektronikos fakultetas
Elektroninių sistemų katedra

ISBN ISSN
Egz. sk. .....
Data .....-.....-.....

Antrosios pakopos studijų **Elektronikos inžinerijos** programos baigiamasis darbas

Pavadinimas **Objektų sekimo vaizde algoritmų įgyvendinimo LPLM įrenginiu tyrimas**

Autorius **Tomyslav Sledevič**

Vadovas **dr. Artūras Serackis**

Kalba: lietuvių

Anotacija

Magistro baigiamojo darbo tikslas – įgyvendinti realiuoju laiku veikiančius objektų sekimo vaizde algoritmus lauku programuojamų loginių matricų įrenginyje (LPLM) ir ištirti šių algoritmų veikimą. Iškelti uždaviniai pasiekti 3 etapais. Atlikta analitinė objektų sekimo vaizde literatūros apžvalga, išanalizuoti objektų sekimo vaizde algoritmai bei jų įgyvendinimo galimybės LPLM įrenginiuose. Sukurti algoritmai ir programos įgyvendintos viename ir keliuose LPLM įrenginiuose (sinchroniškai) taikant VHDL programavimo kalbą ir veikia realiu laiku. Atlikti sukurtų algoritmų tyrimai ir gautų rezultatų analizė. Ištirtas objektų sekimo stabilumas keičiant apšviestumo lygį, fono sudėtingumą, objekto spalvą, judesio greitį, atstumą iki kameros ir posūkio kampą. Darbo apimtis – 69 psl. teksto be priedų, 72 iliustr., 70 bibliografinių šaltinių, 3 priedai.

Prasminiai žodžiai

2D Gauso filtras, būdingųjų taškų vaizde išskyrimas, HOG, judesio aptikimas, lauku programuojamų loginių matricų įrenginiai, LPLM, LBP, objekto sekimas realiu laiku, požymių deskriptorius, SURF, vaizdo kamera, vaizdo signalų apdorojimas, VHDL.

Vilnius Gediminas Technical University
Faculty of Electronics
Department of Electronic Systems

ISBN ISSN
Copies No. .....
Date .....-.....-.....

Master Degree Studies **Electronics Engineering** study programme Final Work

Title **Investigation of Object Tracking Algorithms Based on FPGA**

Author **Tomyslav Sledevič**

Academic supervisor **Dr Artūras Serackis**

Thesis language:
Lithuanian

Annotation

The aim of master's thesis is to investigate the object tracking methods and implement the object tracking algorithms in field programmable gate array (FPGA) devices for real-time execution. The aim is achieved by performing 3 tasks. The analytical review of object tracking methods is performed, reviewing the abilities of algorithms implementation on FPGAs. The object tracking algorithms are implemented in VHDL and distributed on one and few FPGA chips in parallel and works in real-time. The implemented algorithms are investigated and results are analyzed. The stability of different object tracking is investigated by changing the illumination, background complexity, object color, moving velocity, distance to camera and rotation angle. Thesis consists of: 69 p. text without appendixes, 72 figures, 70 bibliographical entries, 3 appendixes included.

Keywords

2D Gauss filter, point of interest extraction in video, HOG, motion detection, field programmable gate array devices, FPGA, LBP, real-time object tracking, features descriptor, SURF, video camera, video signal processing, VHDL.

Vilniaus Gedimino technikos universiteto egzaminų, sesijų ir baigiamųjų darbų rengimo bei gynimo organizavimo tvarkos aprašo 2011-2012 m. m.
1 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Tomyslav Sledevič, 20063012
(Studento vardas ir pavardė, studento pažymėjimo Nr.)
Elektronikos fakultetas
(Fakultetas)
Elektronikos inžinerija, EKSfm-10
(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO) SĄŽININGUMO DEKLARACIJA

2012 m. gegužės 31 d.

Patvirtinu, kad mano baigiamasis darbas tema „Objektų sekimo vaizde algoritmų įgyvendinimo LPLM įrenginiu tyrimas“ patvirtintas 2010 m. lapkričio 5 d. dekano potvarkiu Nr. 196el, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Parenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: Daiva Macko. Mano darbo vadovas dr. Artūras Serackis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

	Tomyslav Sledevič
(Parašas)	(Vardas ir pavardė)

TURINYS

Žymenys ir santrumpos

ASEF (angl. *Average of Synthetic Exact Filter*)	Tikslus dirbtinio filtro vidurkis
BRAM (angl. *Block Random Access Memory*)	Blokinė atsitiktinės kreipties atmintis
DCM (ang. *Digital Clock Manager*)	Skaitmeninio laikrodžio tvarkytojas
DEMUX (angl. *Demultiplexer*)	Demultiplekseris
DNT	Dirbtinių neuronų tinklas
DoG (angl. *Difference of Gaussian*)	Gauso skirtumas
DDR (angl. *Double Data Rate*)	Dvigubas duomenų greitis
eof (angl. *end of frame*)	Kadro pabaiga
FAST (angl. *Features from Accelerated Segment Test*)	Požymiai iš pagreitinto segmento bandymo
FSM (angl. *Finite State Mashine*)	Baigtinių būvių mašina
HOG (angl. *Histogram of Oriented Gradients*)	Orientuotų gradientų histograma
I^2C (angl. *Inter-Integrated Circuit*)	Integrinis grandynas
LBP (angl. *Local Binary Pattern*)	Lokalusis dvejetainis modelis
LCD (angl. *Liquid Crystal Display*)	Skystųjų kristalų vaizduoklis
LPLM	Lauku programuojama loginė matrica
LUT (angl. *Look-Up Table*)	Paieškos lentelė
LVDS (angl. *Low Voltage Differential Signal*)	Žemos įtampos skirtuminis signalas
MOSSE (angl. *Minimum Output Sum of Squared Error*)	Minimali klaidų kvadratų suma išėjime
MUX (angl. *Multiplexer*)	Multiplekseris
PLL (angl. *Phase-Locked Loop*)	Fazių užrakinta kilpa
RGB (angl. *Red Green Blue*)	Raudona, žalia, mėlyna spalvos
SDRAM (angl. *Synchronous Dynamic Random Access Memory*)	Sinchroninė dinaminė atsitiktinės kreipties atmintis
SIFT (angl. *Scale-Invariant Feature Transform*)	Nekintančios skalės požymių transformacija
SURF (angl. *Speeded-Up Robust Features*)	Pagreitinti stiprūs požymiai
SSD (angl. *Sum of Squared Differences*)	Skirtumų kvadratų suma
SUSAN (angl. *Smallest Univalue Segment Assimilating Nucleus*)	Mažiausio dydžio segmento sulyginimo branduolys
SXGA (ang. *Super eXtended Graphics Array*)	Papildomai pratęsta grafinė matrica
TH (angl. *Threshold*)	Slenkstis
USB (angl. *Universal Serial Bus*)	Universalioji nuoseklioji jungtis
VGA (angl. *Video Graphics Array*)	Vaizdo grafinė matrica
VHDL (angl. *Very High Speed Integrated Circuits Hardware Description Language*)	Labai didelio greičio integrinių grandynų aparatūros aprašymo kalba

1. ĮVADAS

Magistro baigiamajame darbe nagrinėjama vaizdo analizės algoritmų įgyvendinimo lauku programuojamomis loginėmis matricomis (LPLM) grįstuose įrenginiuose tema. Algoritmai turi būti modifikuojami priklausomai nuo vaizdo analizės algoritmo taikymo srities, reikalavimų pagrindiniams vaizdo signalo parametrams, LPLM įrenginio našumą apibūdinančių parametrų – aparatūroje įgyvendintų skaičiavimams skirtų elementų skaičiaus ir pan. Pagrindinis dėmesys skiriamas algoritmų, skirtų objektų sekimui vaizde, įgyvendinimui, tobulinimui ir tyrimui.

Darbo aktualumas ir tikslas

Objektų sekimo vaizde algoritmai plačiai taikomi eismo intensyvumui stebėti keliuose [18, 27, 28, 37, 39–43, 59], transporto navigacijos sistemose [11, 19, 44], robotų regos algoritmuose [8, 53, 55], veidų sekimo vaizduose programose [8, 48–51, 65], medicinoje [13] ir, šiuo metu ypač dažnai, papildytos realybės sistemose [33, 36, 54, 62]. Norint, kad sekimo algoritmai veiktų realiuoju laiku ir apdorotų didelės raiškos (1280×720; 1920×1080 taškų) bei didelio kadrų dažnio (30, 60 kadrų per sekundę) vaizdus, reikia spartinti vaizdo filtravimo procesus. Šiam tikslui pasiekti vis dažniau taikomi LPLM, nes skaičiavimai juose gali būti išlygiagretinti ir įrenginys veikia našiau lyginant su šiuolaikiniais procesoriais net tuo atveju, kai taktinis dažnis yra tik šimtų ar dešimčių megahercų eilės.

Darbo tikslas – LPLM įrenginyje įgyvendinti objektų sekimo vaizde algoritmus, kurie veiktų realiuoju laiku ir ištirti šių algoritmų veikimą.

Darbo uždaviniai

Siekiant užsibrėžto tikslo darbe formuluojami trys uždaviniai:

- atlikti analitinę literatūros apžvalgą, išanalizuoti objektų sekimo vaizde algoritmus bei jų įgyvendinimo galimybes LPLM įrenginiuose;
- sukurti naujus algoritmus ir programas, skirtas įgyvendinti LPLM įrenginyje;
- atlikti sukurtų algoritmų tyrimus ir gautų rezultatų analizę.

Naudoti tyrimo ir analizės metodai

Suformuluotiems uždaviniams darbe yra naudojami įvairūs vaizdo pirminio apdorojimo, analizės algoritmai. Objektams vaizde sekti taikomi dvimačiai koreliaciniai, spalvos išskyrimo filtrai. Sutapdinant vaizdus skaičiuojama mažiausia skirtumų kvadratų suma tarp aibės taškų abejuose vaizduose. Taikomi tekstūros kodavimo metodai: orientuotų gradientų histograma HOG (angl. *Histogram of Oriented Gradients*) ir lokalusis dvejetainis modelis LBP (angl. *Local Binary*

Pattern). SURF (angl. *Speeded-Up Robust Features*) požymiams vaizde išskirti skaičiuojamas Hessian matricos determinantas, dvimatė vilnelių transformacija (angl. 2*D Wavelet Transform*). Judančių taškų grupavimo į objektus metodas taikomas įgyvendinant fono vaizde pašalimo ir skirtumų tarp kadrų skaičiavimo algoritmuose. Bresenham algoritmas taikomas linijoms monitoriuje braižyti.

Darbo naujumas ir praktinė nauda

Darbe įgyvendinti algoritmai veikia realiuoju laiku, vaizdas apdorojamas 60 kadrų per sekundę dažniu (išskyrus sekimą pagal spalvą), kai taktinis vaizdo signalo dažnis yra lygus 25 MHz. Įgyvendinant vaizdo apdorojimo ir analizės algoritmus nebuvo prisirišta prie konkrečios vaizdo kameros raiškos ar kadrų dažnio, todėl sukurti algoritmai yra universalūs ir gali būti taikomi su didesnės arba mažesnės raiškos bei kadrų dažnio kameroms. Reikiama greitaveika pasiekiama išlygiagretinus, vienu metu kreipiantis į kelis tūkstančius filtro matricos elementų.

Požymių vaizde išskyrimo bei jų kodavimo etapai procesoriniuose įrenginiuose sunaudoja daug aparatūros resursų, dėl to turi būti mažinamas apdorojamų kadrų skaičius. Esant mažesniam kadrų dažniui, prarandamos galimybės be iškraipymų įrašyti greitai judančius objektus ir vaizdo objektų sekimo ir atpažinimo algoritmai nebegali būti taikomi. Objektų sekimo algoritmų (arba atskirų, daug skaičiavimų reikalaujančių etapų) įgyvendinimas LPLM įrenginiuose žymiai paspartina skaičiavimus.

Darbo struktūra

Magistro baigiamojo darbo aiškinamąjį raštą sudaro 5 skyriai. 1 aiškinamojo rašto skyrius – įvadas.

2 skyriuje pateikiama analitinė literatūros apžvalga. Nagrinėjami objektų sekimo algoritmai, jų taikymo sritys, privalumai, trūkumai. Apžvelgiamos algoritmų įgyvendinimo galimybės LPLM įrenginiuose. Pasirenkami keli algoritmai tolesniam įgyvendinimui.

3 skyrius skirtas algoritmų įgyvendinimui LPLM įrenginyje. Pateikiamos algoritmų schemos su detaliu veikimo paaiškinimu, blokinės sukurtų modulių jungimo schemos ir nuoseklaus veikimo algoritmų dalių būvių diagramos.

4 skyriuje pateikiamas objektų sekimo vaizde algoritmų tyrimas. Tiriamas objektų sekimo vaizde stabilumas keičiant apšviestumo lygį, fono sudėtingumą, objekto spalvą, judesio greitį, posūkio kampą ir atstumą iki kameros.

5 skyriuje darbas apibendrinamas ir pateikiamos išvados apie objektų sekimo vaizde algoritmų įgyvendinimo pagrįstumą LPLM įrenginiuose.

2. ANALITINĖ LITERATŪROS APŽVALGA

Šio skyriaus tikslas yra apžvelgti objektų sekimo vaizde algoritmus, kurie yra įgyvendinti tiek bendros paskirties skaitmeninio signalų apdorojimo procesoriniuose, tiek LPLM įrenginiuose. Algoritmai sisteminami pagal panašumą ir įgyvendinimo sudėtingumo laipsnį. Analizės metu išskiriamos kelios algoritmų grupės: objektų sekimo pagal panašumų tarp dviejų vaizdų atitikimą; objektų sekimo pagal aptiktą judesį; objekto sekimo pagal tekstūrą; objektų sekimo pagal vaizde išskirtus būdinguosius taškus.

Kiekvienai darbe analizuojamai algoritmų grupei yra skiriamas atskiras skyriaus poskyris. Išvardinami visų aptartų algoritmų privalumai, trūkumai bei galimos jų taikymo sritys, nagrinėjami algoritmų įgyvendinimo LPLM įrenginiuose ypatumai ir galimybės. Darbe nustatoma, ar efektyvu perkelti (pritaikyti vykdymui LPLM) algoritmus iš skaitmeninio signalų apdorojimo procesoriais grįstų sistemų į LPLM įrenginius atsižvelgiant į greitaveiką, sunaudojamą galią ir pasirinktos programavimo kalbos ypatybes.

2.1. Viena kitą atitinkančių vaizdo sričių paieškos algoritmai

Prie viena kitą atitinkančių vaizdo sričių paieškos ir sekimo grupės priskiriami šie metodai [2]:

- Sekimas pagal spalvą – pagal spalvą segmentuotų vaizdo fragmentų sekimo metodas;
- Sekimas pagal formą – metodas, grįstas iš anksto žinomos formos objektų paieška vaizde;
- Koreliacijos arba minimalios skirtumų kvadratų sumos (toliau – SSD) (angl. *Sum of Squared Differences*) skaičiavimo tarp kadro ir ieškomo objekto pavyzdžio taškų metodas.

Tyrimais nustatyta, kad sekimas pagal spalvą yra vienas iš lengviausiai įgyvendinamų algoritmų objektams sekti [3–10]. Apžvelgus pastaruosius straipsnius nustatyta, kad sekimas pagal spalvą yra vienas iš lengviausiai įgyvendinamų algoritmų objektams sekti. Algoritmo įgyvendinimas tiek procesoriniuose, tiek LPLM įrenginiuose nereikalauja didelių aparatūros resursų, todėl gali būti įgyvendintas senos kartos *Virtex II* arba *Cyclone II* LPLM moduliuose [3, 4]. Pagrindinis reikalavimas sekimo pagal spalvą algoritmui – pakankamai didelis (2^{16}) spalvų lygių skaičius. Be to, vaizde nepageidautina turėti panašios arba identiškos spalvos objektų, nes sklandžiam algoritmo veikimui reikės įvesti papildomų požymių (vaizdo fragmento tikėtina vieta, dydis, forma), padedančių išskirti sekamą objektą nuo kitų vaizde esančių objektų [9].

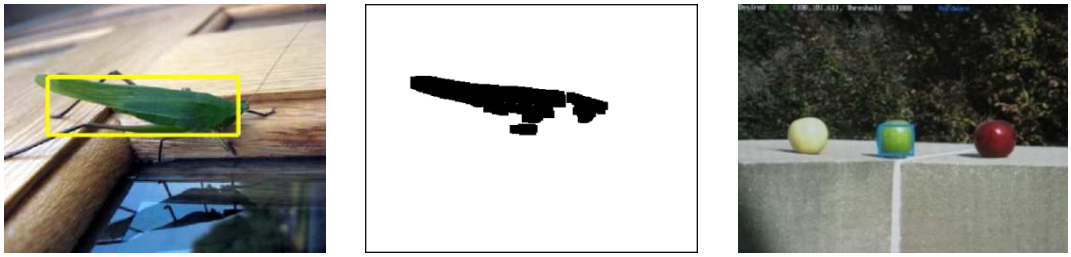
Pirmoji funkcija, taikoma pasirinktos spalvos vaizdo sričiai (fragmentui) išskirti, yra vaizdo segmentavimas. 1 paveiksle pateikiamas spalvoto vaizdo segmentavimo pavyzdys [3]. Pavyzdyje bandyta išskirti raudonos spalvos segmentus (2.1 pav., *a*). Iš segmentavimo rezultatų (2.1 pav., *b*)

matyti, kad kadre yra aptiktos kelios raudonos spalvos sritys (2.1 pav., *b*, pažymėtos apskritimais). Tai patvirtina faktą, kad realiomis sąlygomis nepatartina taikyti vien tik sekimą pagal išskirtą sritį. Siekiant pagerinti sutapdinimą, naudojamos morfologinės operacijos [5] arba kiekybiškai vertinamas išskirtos srities taškų skaičius. Palyginimui (2.1 pav., *c*) pateiktas į tris lygius segmentuotas pustonis vaizdas. Matoma, kad objekto sekimas pagal pustonius praktiškai naudingas tik jei sąlygos yra artimos idealioms, kai vaizde yra ryškiausių arba tamsiausių (pagal skaisčio lygį) sričių, kurias norima sekti. Šis algoritmas gali būti taikomas žvaigždėms vaizde pažymėti bei sekti [10].

a) *b*) *c*)

2.1 pav. Vaizdo kadras (*a*), segmentuotas pagal spalvą (*b*), segmentuotas pagal pustonius (*c*) [3]

Pritaikyti LPLM įrenginiams du to paties objektų sekimo pagal spalvą algoritmų įgyvendinimai lyginami kuriant kelis programinius procesorius [4]. 2.2 paveiksle pateikti žalios spalvos objekto išskyrimo rezultatai. 2.2 paveiksle *b* pateikiamas išskirtai žaliai sričiai pritaikytos morfologinės operacijos triukšmui vaizde nufiltruoti rezultatas. 2.2 paveiksle *c* pateikiamas sėkmingai sekamo žalios spalvos obuolio pavyzdys. Nustatyta, kad algoritmas, įgyvendintas įterptinėje sistemoje nenaudojant programinio procesoriaus, sunaudoja mažiau galios ir resursų. Taip yra todėl, kad, prieš kuriant programinį procesorių LPLM, visada rezervuojamas duomenų ir programos atminties kiekis, kuris algoritmo veikimo metu dažnai neišnaudojamas. Programinio procesoriaus privalumas yra tas, kad jis pakankamai lengvai programuojamas, nėra būtinybės kontroliuoti procesų sinchronizaciją, tačiau neracionaliai išnaudojama atmintis ženkliai sumažina tokių programų vykdymo našumą. LPLM architektūra iš prigimties skirta lygiagrečiam procesų įgyvendinimui, todėl programiniai procesoriai negali visiškai išnaudoti LPLM įrenginio ir pasiekti didelę veikimo spartą [4].



a) *b*) *c*)

2.2 pav. Spalvotas kadras ir jame pažymėtas objektas (*a*), išskirta žalios spalvos sritis (*b*), aptiktas žalios spalvos objektas (*c*) [4]

Objektams vaizde sekti gali būti taikomi sekimo pagal formą algoritmai [11–16]. Darbe [11] aprašomas automobilio sekimo algoritmas, įgyvendintas LPLM įrenginyje. Vaizdui taikomas gradiento operatorius, kuriuo išskiriami kontūrai vaizde. Žinomas automobilio profilis sutapdinamas su išskirtaisiais kontūrais. Iš gauto rezultato (2.3 pav., *a*) matyti, kad objektas vaizde aptinkamas tinkamai. Tačiau tinkamas objekto vaizde aptinkamumas ir uždarų kontūrų paieškos rezultatai priklauso nuo oro sąlygų ir apšvietimo kaitos (2.3 pav., *b*) [12].

a) *b*) *c*)

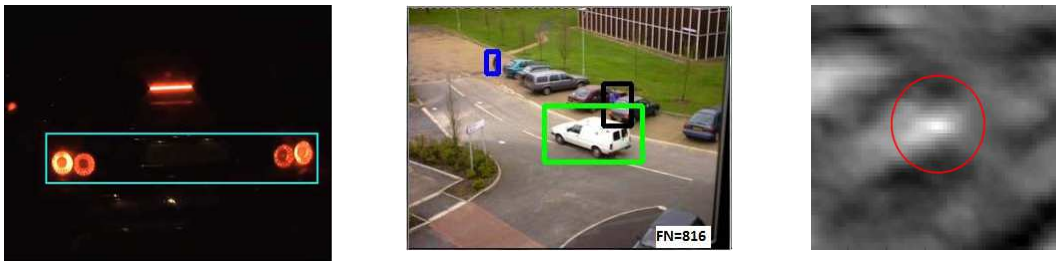
2.3 pav. Automobilio sekimo pagal formą rezultatai (*a*, *b*) [11], bakterijų sekimo rezultatas (*c*) [13]

LPLM įrenginyje įgyvendintas bakterijų sekimo algoritmas skirtas baltiems taškams tamsiame vaizdo fone rasti [13]. Sekimo rezultatai pateikti 2.3 paveiksle *c*. Algoritmas stabiliai seka vieną pažymėtą objektą, skaičiuoja jo judėjimo greitį ir kryptį. Algoritmas iš dalies įgyvendintas VHDL kalba, kai kurios programos dalys sukurtos taikant *Simulink* programų paketą ir įterptos į programinį procesorių. Literatūros šaltinyje taip pat pateiktos įgyvendinto algoritmo blokinės schemos, signalų laiko diagramos.

Dviejų vaizdų sutapdinimui gali būti naudojami ir dvimačiai koreliaciniai filtrai [17–21]. Taikant koreliacinį filtrą skaičiuojama ieškomo pavyzdžio su kadro taškais sandaugų suma. Tose vaizdo vietose, kur koreliacijos vertė didžiausia, yra didelė tikimybė, kad aptiktas būtent ieškomas pavyzdys. Koreliacinio filtro taikymo rezultatai pateikti 2.4 paveiksle *a*. Literatūroje [19] pateiktas įgyvendintas algoritmas, kuris naktį filmuotame vaizde ieško galinių automobilio žibintų šviesų. Algoritmo pirmajame etape išskiriama raudona spalva, antrajame – skaičiuojama nufiltruoto vaizdo ir žinomo žibintų pavyzdžio tarpusavio kryžminė koreliacija. Žibintų išsidėstymo vienas kito atžvilgiu simetriškumas panaudotas atsparumui triukšmams ir sekimo stabilumui didinti.

2.4 paveiksle *b* pateiktas *Kalman* filtro taikymo rezultatas sekant ir vaizde pažymint kelis objektus [18]. *Kalman* filtro taikymo privalumas tas, kad galima numatyti sekamo taikinio padėtį, kai sutapdinamų vaizdų koreliacinio filtro atsakas yra mažesnis už nustatytą slenkstį [17]. Dviejų vaizdų kryžminės koreliacijos rezultatas pateiktas 4 paveiksle *c*. Raudonos spalvos apskritimas pažymi vietą, kur koreliacijos vertė yra didžiausia. Žinoma, kad koreliacijos taikymo trūkumas tas, kad ieškomo pavyzdžio koreliacijos rezultatas yra didžiausias (globalus maksimumas kadre) balto fono vaizde, kuriame yra maksimali skaisčio vertė [20]. Todėl dažnai, prieš apskaičiuojant

koreliaciją, nuo vaizdo atimama vidutinė jo skaisčio vertė. Tokiu būdu tik persidengiančios šviesios arba tamsios ieškomo pavyzdžio ir kadro dalys sąlygos koreliacijos įvertį. Taip pat patartina vaizde ieškoti ne globalių, o lokalių koreliacijos ekstremumų, atsižvelgiant į buvusiame kadre rasto pavyzdžio vietą [20].



a) *b*) *c*)

2.4 pav. Automobilio galinių šviesų sekimas (*a*) [19], kelių objektų sekimas (*b*) [18], koreliacijos atsakas (*c*) [20]

Koreliacijos skaičiavimo įgyvendinimas reikalauja daug sandaugos operacijų, proporcingų ieškomo vaizdo pavyzdžio plotui. Kadangi LPLM turi ribotą daugintuvų kiekį, realiuoju laiku įmanoma su vaizdu sutapdinti tik mažo ploto pavyzdį. Todėl pavyzdžio ploto dydžio didinimas galimas pakėlus signalų apdorojimo taktinį dažnį arba apskaičiavus koreliaciją ne kiekvienam vaizdo taškui, o taškų grupės vidurkiui, gautam tinkleliu padalijus vaizdą į segmentus. Tinklelio žingsnį reikėtų suderinti su ieškomo pavyzdžio dydžiu ir kurti tuo tankesnį, kuo mažesnis objektas yra sekamas. Koreliacijos operaciją taip pat galima pakeisti kadro ir pavyzdžio skirtumų kvadratų sumos skaičiavimu [20, 21]. SSD rezultato vaizdas yra atvirkščias pateiktam 2.4 paveiksle *c*, nes skirtumų kvadratų suma mažiausia tose vietose, kur aptiktas ieškomas pavyzdys. Dėl šios priežasties, skaičiuojant SSD, vaizde ieškoma minimalaus atsako. Skaičiuojant SSD taikoma tiek pat sandaugos operacijų, kiek apskaičiuojant koreliaciją. Lyginant koreliacijos ir SSD rezultatus nustatyta, kad skaičiuojant SSD globalus minimumas bus tik toje vietoje, kur pavyzdys labiausiai sutampa su vaizdu. Pagal globalaus ekstremumo radimo patikimumą, tik normuotos kryžminės koreliacijos atsakas prilygsta SSD rezultatui [20]. Toliau pateikiamos koreliacijos ir skirtumų kvadratų sumos skaičiavimo formulės:

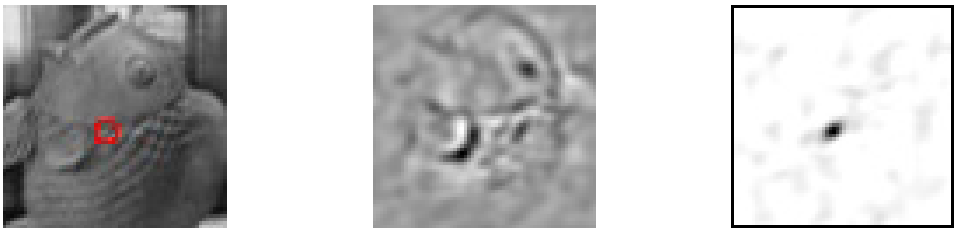
$$C = \sum_{[i,j]\in R}^{I,J} f(i,j)\cdot g(i,j); \qquad SSD = \sum_{[i,j]\in R}^{I,J} \left(f(i,j) - g(i,j)\right)^2, \tag{2.1}$$

čia C – koreliacijos vertė, SSD – skirtumų kvadratų sumos vertė, $f(i, j)$ – vaizdo kadro taško vertė, $g(i, j)$ – ieškomo pavyzdžio taško vertė koordinatėse i ir j.

Objekto sekimą vaizde pagal vieną neatsinaujinantį pavyzdį patogu taikyti, kai nekinta apšvietimas, objekto dydis ir pasukimo kampas kameros atžvilgiu. Tačiau, jeigu norima nepriklausomai nuo anksčiau paminėtų faktorių pastoviai sekti tą patį objektą, būtina periodiškai

atnaujinti sekamo vaizdo pavyzdį. *Kalman* filtras yra dažnai taikomas įgyvendinant sekamų pavyzdžių atnaujinimo algoritmus [17–19].

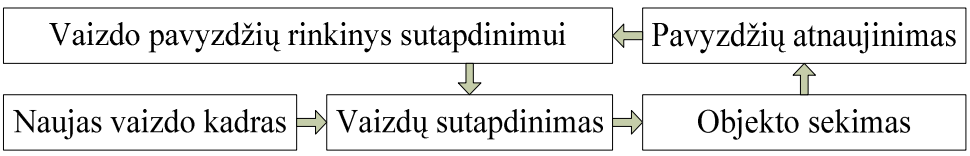
Kitame šaltinyje siūloma kaupti ir saugoti sekamo objekto pavyzdžius [21]. Kai kurie tyrėjai siūlo įgyvendinti tokį veidų sekimo algoritmą, kuris sekdamas žmogaus veidą saugo jo nuotraukas. Naujos nuotraukos įrašomos tik tuo atveju, kai aptinkamas veidas ir sutapdinimo rezultatas nepatenka į iš anksto nustatytą pasikliovimo intervalą. Straipsniuose [22–24] pateikiami objektų sekimo rezultatai taikant autorių siūlomus adaptyviuosius koreliacinius filtrus ir periodiškai atnaujinant sekamą vaizdo pavyzdį atitinkantį filtrą. Koreliacinis filtras, kurio pavyzdys pateiktas 5 paveiksle *b*, atnaujinamas pagal siūlomus ASEF (angl. *Average of Synthetic Exact Filter*) ir MOSSE (angl. *Minimum Output Sum of Squared Error*) filtrus. Algoritmas pastaruosius abu filtrus suderina taip, kad vaizdo ir filtro tarpusavio kryžminės koreliacijos atsakas būtų vienas centruotas ekstremumas (2.5 pav., *c*). Koreliacinio filtro taškai yra sveriami atstumo nuo centro atžvilgiu, todėl didžiausią įtaką koreliacijos rezultatui turi taškai, išsidėstę arčiau filtro centro, o tolstant nuo jo į kraštus mažėja adaptyvaus koreliacinio filtro taškų svarba (2.5 pav., *b*). Per vidurį pastebimi didžiausi skaisčio pokyčiai, o prie kraštų skaisčio vertė artėja prie vidutinės (pilka spalva). Kadangi sekamas objektas gali judėti, o filtras yra inertiškas (inertiškumas pasireiškia todėl, kad vaizdas neatnaujinamas kas kadrą), tai koreliacijos atsakas pasislenka nuo centro ta kryptimi, į kurią pusę slenka vaizdas ir greičiu, proporcingu sekamo objekto judėjimo vaizde greičiui.



a) *b*) *c*)

2.5 pav. Vaizdas įėjime (*a*), dvimačio adaptyvaus koreliacinio filtro vaizdas (*b*), koreliacijos rezultatas (*c*) [24]

Adaptyvaus sekimo algoritmams [17–24] galima nubraižyti apibendrintą blokinę veikimo schemą, kuri pateikta 2.6 pav.



2.6 pav. Adaptyvių sekimo algoritmų blokinė schema

Vaizdų sutapdinimo bloke gali būti skaičiuojama kryžminė koreliacija arba skirtumų kvadratų suma. Objektų sekimo bloke ieškoma ekstremumų. Pagal rasto ekstremumo koordinates

atnaujinamas seklys, kuris monitoriuje pažymi sekamo objekto vietą. Algoritmo grįžtamojo ryšio grandinėje svarbiausiu laikomas pavyzdžių atnaujinimo blokas. Čia priimamas sprendimas, kada pridėti arba atnaujinti sekamą pavyzdį. Priklausomai nuo algoritmo vaizdų pavyzdžių rinkinyje gali būti vienas arba šimtai pavyzdžių. Kiekvieno naujai iš kameros atsiųsto kadro taškai yra sutapdinami su atnaujintais pavyzdžiais. Sekimo stabilumą sąlygoja vaizdų sutapdinimo metodas ir pavyzdžių atnaujinimo algoritmas, nes būtina tinkamai atrinkti tik tuos pavyzdžius, kurie atspindi sekamo objekto centruotus vaizdus filmuojant iš įvairių kampų.

Adaptyvaus sekimo algoritmus patariama taikyti aplinkoje, kurioje didesnė dalis sekamo objekto nėra užstojama. Geriausi sekimo rezultatai pasiekiami, kai objektas juda tolydžiai ir yra pilnai matomas arba nežymiai užstojamas fono detalių tam, kad sekamas pavyzdys būtų teisingai atnaujinamas. Adaptyvusis algoritmas yra atsparus objekto pasisukimui, objekto dydžio pasikeitimui ir apšvietimo pokyčiui tik tuo atveju, kai vaizdo kameros kadrų dažnis yra pakankamas, kad pavyzdys būtų atnaujinamas tinkamai [17–24]. Vaizdo kameros kadrų dažnis yra tiesiogiai susietas su sekamo objekto judėjimo greičiu. Kuo greičiau objektas vaizde keičia savo koordinates, tuo spartesnę vaizdo kamerą patariama taikyti. Tam, kad būtų išlaikytas pastovus kadrų greitis, vaizdo apdorojimas turi veikti realiu laiku, todėl sekimo algoritmą patartina įgyvendinti LPLM. Kadangi yra galimybė sukurti ir lygiagrečiai adresuoti tūkstančius slenkančio koreliacinio filtro elementų, algoritmas veiks realiu laiku net tada, kai taktinis dažnis bus tik dešimčių megahercų eilės.

2.2. Sekimas pagal judesio aptikimą

Kelių eismo stebėjimo, apsaugos sistemose kartais nebūtina sekti konkretaus objekto, pakanka registruoti ir pažymėti visus objektus, kurie juda arba kerta apibrėžtą draudžiamąją sritį. Judesio vaizde aptikimo ir sekimo grupei priskiriami šie metodai [31–32]:

- Skirtumų tarp dviejų gretimų kadrų skaičiavimas;
- Fono vaizde pašalinimas (angl. *Background Subtraction*);
- Optinio srauto analizė.

LPLM įrenginiuose įgyvendinti skirtumų tarp kadrų skaičiavimo algoritmai taikomi judantiems objektams vaizde pažymėti [25–27]. Algoritmai veikia realiu laiku ir yra pakankamai nesunkiai įgyvendinami, nes skaičiuojami tik skirtumai tarp taškų gretimuose kadruose. Jautris judesiui reguliuojamas priderinant slenkstį. Jeigu skirtumo tarp taškų vertė viršija nustatytąjį slenkstį, tai laikoma, kad taškas pajudėjo. 2.7 paveiksle *a*, *c* pateikti kadrai iš gatvės stebėjimo kamerų, 2.7 paveiksle *b* – aptiktas judesys vaizde. Pastebėtina, kad sunku atskirti kelis važiuojančius automobilius, jeigu jie yra arti vienas kito. Šio metodo trūkumas – tai, kad ne visada galima

teisingai įvertinti judančio objekto matmenis, nes jie priklauso nuo šalia kuriamo šešėlio, fono skaisčio bei objekto judėjimo greičio.

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

2.7 pav. Vaizdai iš gatvės stebėjimo kamerų (*a*, *c*), aptikti skirtumai tarp kadrų (*b*, *d*) [27, 28]

Objektams vaizde išskirti taikomi fono vaizde pašalinimo algoritmai [28–32]. Atmintyje saugomas vaizdo fono kadras yra lyginamas su naujai gautu iš kameros, o apskaičiuoti skirtumai gali būti išvedami į vaizduoklį. Dažnai skirtumai tarp taškų lyginami su slenkstine verte, kurią viršijus registruojamas objekto taškas [30, 31]. 2.7 paveiksle *d* pateiktas fono vaizde pašalinimo rezultatas [28]. Matoma, kad teisingai išskiriamos tos objektų sritys, kurios pagal spalvą skiriasi nuo fono. Fono pašalinimo ir skirtumų tarp dviejų gretimų kadrų skaičiavimo algoritmai skiriasi tik atimties operacijos įgyvendinimu.

Optinio srauto analizė gali būti taikoma objektams vaizde sekti [8, 33–38]. Taškams vaizde skaičiuojami jų judėjimo greitis ir kryptis (2.8 pav., *b*). Priklausomai nuo krypties ir greičio 2.8 paveiksle *d* sugrupuoti ir skirtinga spalva pavaizduoti taškai. Šio metodo privalumas tas, kad yra nustatoma objekto judėjimo kryptis, kuri vertinama kaip papildomas požymis vaizde atskiriant ir pažymint taškų grupes.

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

8 pav. Vaizdai iš kamerų (*a*, *c*), taškų judėjimo vektoriai (*b*), aptikti judantys objektai (*d*) [8, 37]

Sekimo pagal judesio vaizde aptikimą algoritmus galima apibendrinti pagal 2.9 pav. pateiktą blokinę schemą. Algoritmams būdingas skirtumų tarp gretimų arba esamo ir fono kadrų aptikimas. Skirtumai tarp taškų vaizde vertinami kaip skaisčio pokyčiai arba taškų slinkties vektoriai. Pavieniai objektai yra aptinkami ir išskiriami grupuojant judančius taškus pagal greitį, kryptį ar atstumą iki artimiausio judančio taško.

2.9 pav. Apibendrintas sekimo pagal judesio aptikimą algoritmas

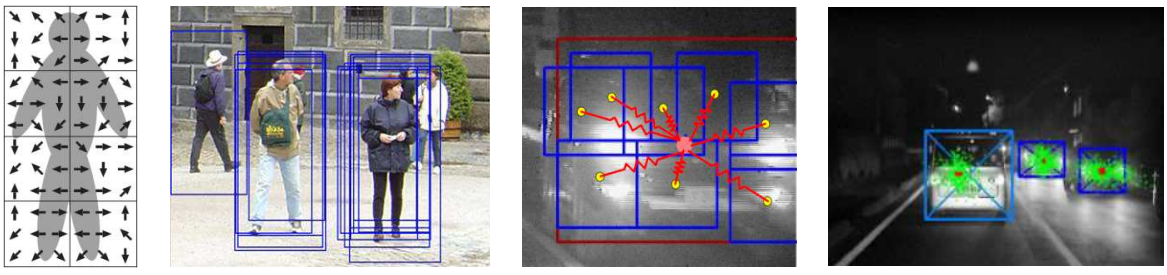
Objektų sekimo pagal judesio aptikimą algoritmus patartina taikyti, kai vaizdo kamera nejuda. Norint vienodai sėkmingai sekti lėtai ir greitai judančius objektus svarbu naudoti adaptyvų jautrio judesiui slenkstį, kuris didinamas, kai judėjimas vaizde yra mažas arba mažinamas, kai judėjimas intensyvus. Norint aptikti greitai judančius objektus būtina didinti kadrų greitį, todėl duomenų apdorojimo sparta turi didėti iki dešimčių ar šimtų megabaitų per sekundę greičio. Todėl racionalu sekimo algoritmą įgyvendinti LPLM dėl vykdomų procesų išlygiagretinimo ir veikimo realiu laiku.

2.3. Sekimas pagal tekstūrą

Sekimas pagal tekstūrą grindžiamas vaizdo skaisčio verčių, esančių tam tikrose apibrėžtose srityse, kodavimu. Tikslas yra efektyviai sumažinti apdorojamų duomenų kiekį kiekybiškai vertinant pasikartojančius tekstūros elementus. Užkoduotai tekstūrai sudaromos histogramos, kurias lyginant sprendžiama apie sekamo objekto buvimą vaizde. Objektui pagal jo tekstūrą aptikti dažniausiai taikomi šie metodai [39, 45]:

- Orientuotų gradientų histograma HOG;
- Lokalusis dvejetainis modelis LBP.

HOG metodas dažniausiai taikomas žmonėms vaizde aptikti ir sekti [39–43]. 2.10 paveiksle *a* pateikiamas segmentuoto žmogaus silueto pavyzdys. Ieškomas pavyzdys suskirstomas į sritis, kurioms skaičiuojamos orientuotų gradientų histogramos. Kuo pavyzdžio suskirstymo tinklelio tankis didesnis, tuo tiksliau objektas aptinkamas vaizde, tačiau kartu didėja atliekamų skaičiavimų kiekis. Algoritmas dar labiau sudėtingėja, kai reikia vertinti objekto mastelio pasikeitimus.



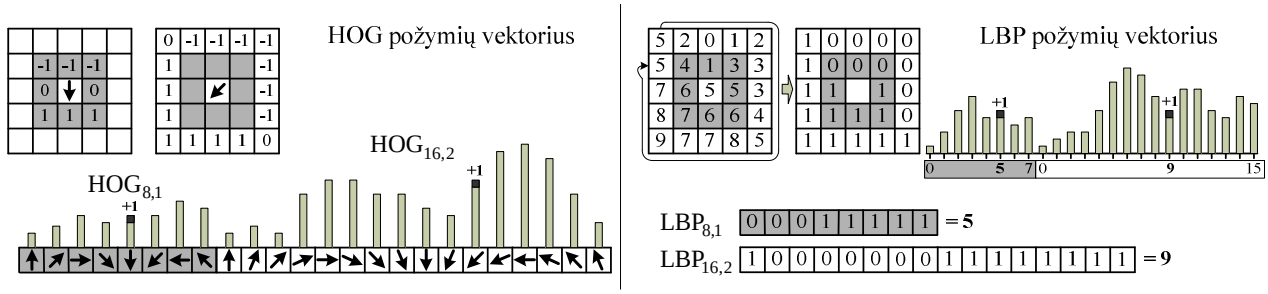
a) *b*) *c*) *d*)

2.10 pav. Vaizdo segmentavimas taikant HOG (*a*, *c*), vaizde aptikti objektai (*b*, *d*) [43, 39, 44]

Taikant HOG metodą tas pats objektas gali būti aptiktas kelis kartus arba neaptiktas, kai mastelis sumažėja (2.10 pav., *b*). HOG metodas taip pat taikomas automobiliams sekti [44]. Automobilio vaizde suskirtos būdingosios sritys, kurioms skaičiuojamos histogramos (2.10 pav., *c*).

Histogramos sutapdinamos su iš kameros gaunamam vaizdui paskaičiuotomis būdingosiomis sritimis. Tose vietose, kur skirtumas mažesnis už nustatytą slenkstį, laikoma, kad aptiktas automobilis (2.10 pav., *d*).

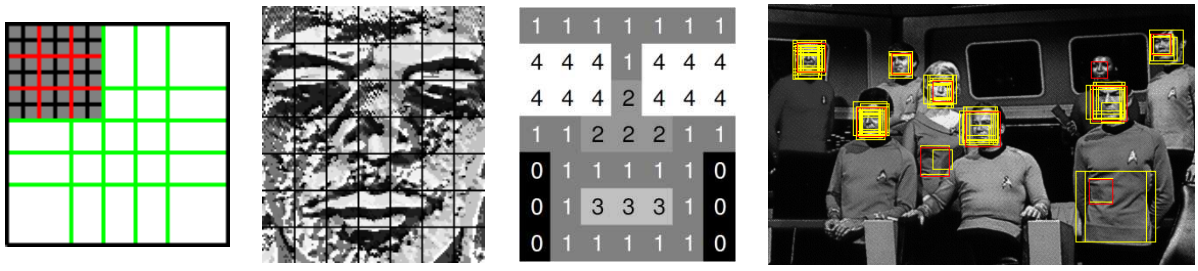
HOG ir LBP histogramų sudarymo principai pateikti 2.11 paveiksle. Lokaliam skaisčio vaizde pokyčiui nustatyti taikomi gradiento operatoriai, kurie žymimi $HOG_{8,1}$ – pirmosios skalės gradiento operatorius, $HOG_{16,2}$ – antrosios skalės gradiento operatorius. Pirmasis indeksas žymi gradiento krypčių skaičių, antrasis – skalės eilę. Histograma sudaroma pridedant po vienetą prie stulpelio, atitinkančio gradiento kryptį nagrinėjamame taške (2.11 pav., *a*). 3×3 ir 5×5 dydžio gradiento operatoriuose esantys skaičiai 1 ir –1 nurodo, kad skaisčio vertės tuose taškuose turi būti atitinkamai didesnės ir mažesnės už vertes skaičiumi 0 pažymėtuose laukuose. $LBP_{8,1}$, $LBP_{16,2}$ histogramos sudaromos lyginant centrinio taško ir jo 8-ių ar 16-os kaimynų skaisčio vertes. Jei centrinio taško vertė mažesnė už kaimyno, tai požymių vektoriaus elementas, atitinkantis tą kaimyną, koduojamas nuliu, kitais atvejais – vienetu. Prie histogramos stulpelio pridedamas vienetas, kai požymių vektoriuje yra tik po vieną perėjimą 0→1 ir 1→0.



a) *b*)

2.11 pav. HOG (*a*) ir LBP (*b*) histogramų sudarymas

LBP metodas pirmą kartą pasiūlytas 2002 m. ir iki šiol yra laikomas vienu iš geriausių tekstūros deskriptorių [45, 46]. LBP šiuo metu ypač dažnai taikomas veidų nuotraukose ir vaizdo medžiagoje aptikimui [47–51]. 2.12 paveiksle *a* pateikiamas vienas iš galimų dvimačio filtro pavyzdžių veidams vaizde rasti. Jis susideda iš 9-ių dalinai persidengiančių mažesnių filtrų (pilka spalva). Pastarieji padalyti dar į 9 dalis (raudona spalva). Kuo smulkesniais kvadratėliais suskirstytas

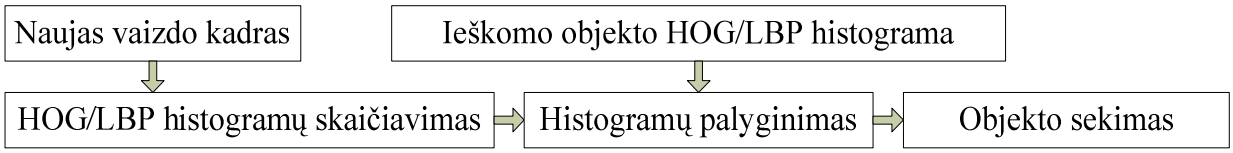


a) *b*) *c*) *d*)

2.12 pav. Filtro tinklelis (*a*), veido LBP (*b*), svorių koeficientai (*c*), vaizde aptikti veidai (*d*) [51]

filtras, tuo ilgesnis gaunamas veido požymių vektorius, tuo tiksliau galima išskirti vieną veidą nuo kito. Tačiau tankėjant filtro tinkleliui proporcingai didėja tikimybė neaptikti veido dėl jo nežymaus pasisukimo. Taikant slankiojantį dvimatį filtrą vaizdui skaičiuojamos LBP histogramos, kurių vertės pasveriamos pagal 2.12 paveiksle *c* pateiktus svorių koeficientus. LBP metodas aptinka veidų ir ne veidų sritis (2.12 pav. *d*), todėl būtina taikyti papildomą filtravimą.

Apibendrintas sekimo pagal objekto tekstūrą algoritmas pateiktas 2.13 pav. Svarbiausiais algoritmo etapais ir daugiausia aparatinių resursų sunaudojančiais laikomi histogramų skaičiavimas ir palyginimas. Sekimo rezultatams pagerinti dažnai taikomi atraminių vektorių mašinos [40–43, 51], vieno sluoksnio DNT [39], artimiausių kaimynų klasifikatoriai [48].



2.13 pav. Apibendrintas sekimo pagal tekstūrą algoritmas

HOG ir LBP tekstūros deskriptorių privalumas tas, kad abu yra atsparūs apšvietimo pokyčiams vaizde, nes histogramos sudarymas grindžiamas taško ir kaimyninių taškų skaisčio verčių santykiniu palyginimu. Kuo didesnis artimiausių kaimynų skaičius naudojamas histogramai kurti, tuo stabilesnis yra atpažinimas. LBP deskriptorius atsparus tekstūros pasukimui. Tyrimo rezultatais pagrįsta, kad galima pasiekti 0,6 % pasuktos ir nepasuktos tekstūrų histogramų sutapimo santykinę paklaidą taikant kartu trijų skalių LPB [47]. Objektų sekimas pagal tekstūrą dažnai įgyvendinamas LPLM įrenginiuose [39, 41–43, 49]. Greitaveika pasiekiama dėl lygiagrečiai įgyvendintų histogramų sudarymo ir palyginimo procesų, o tai leidžia VGA raiškos 60 kadrų per sekundę vaizdą apdoroti realiu laiku taktiniam dažniui esant 25 MHz [49].

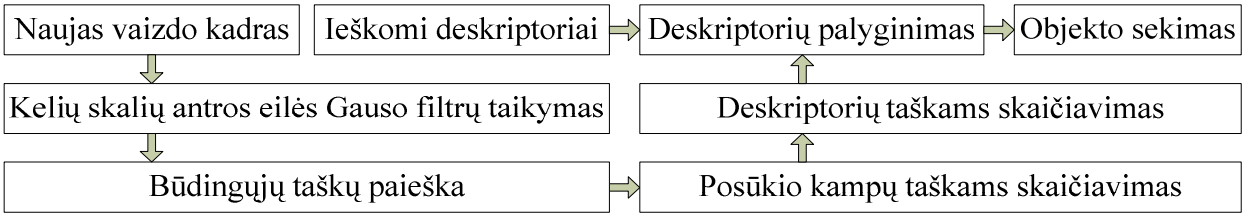
2.4. Sekimas pagal būdinguosius taškus

Sekimas pagal būdinguosius taškus grindžiamas požymių (taškų, kampų, linijų, dėmių) vaizde radimu ir kodavimu deskriptoriais. Saugomi ieškomo objekto deskriptoriai lyginami su suskaičiuotais ir pagal jų atitikimo laipsnį sprendžiama apie objekto buvimą kadre. Dažniausiai taikomi būdingųjų taškų vaizde išskyrimo algoritmai: SURF ir SIFT (angl. *Scale-Invariant Feature Transform*) [52–58]. Abiems algoritmams būdingi panašūs taikymo etapai, todėl 2.14 paveiksle pateikiama apibendrinta blokinė algoritmų schema. SURF algoritmo atveju gautam kadrui skaičiuojamas sudėtinis vaizdas tam, kad antros eilės dvimačių Gauso filtrų taikymas nepriklausytų nuo filtrų skalės [52]. Būdingųjų taškų paieška atliekama skaičiuojant *Hessian* determinantą $H(x, y)$ ir ieškant lokaliai didžiausio determinanto atsako tarp gretimų skalių.

$$H(x,y)=\det\begin{bmatrix}A & C\\ C & B\end{bmatrix}=\det\begin{bmatrix}\partial^2 f/\partial x^2 & \partial^2 f/\partial x\partial y\\ \partial^2 f/\partial x\partial y & \partial^2 f/\partial y^2\end{bmatrix}, \quad (2.2)$$

čia *A*, *B*, *C* – antros eilės Gauso išvestinės atitinkamai *x*, *y* ir *xy* kryptimis.

Požymių posūkių kampams ir deskriptoriams skaičiuoti taikomos dvimatės vilnelių transformacijos. SIFT nuo SURF skiriasi tuo, kad neskaičiuojamas integrinis vaizdas, vietoje Hessian determinanto skaičiavimo taikomas DoG (angl. *Difference of Gaussian*) metodas, kurio metu skaičiuojami skirtumai tarp Gauso filtrų atsakų gretimose skalėse [58]. Deskriptorių ir kampų būdingiesiems taškams nustatymo metodai skiriasi naudojamų aplinkinių taškų skaičiumi.



2.14 pav. Apibendrintas SURF ir SIFT požymių vaizde paieškos algoritmas

Taikant SURF ar SIFT metodus būdingieji taškai vaizde dažniausiai pažymimi apskritimais arba stačiakampiais (2.15 pav., *a*). Požymių palyginimo rezultatas pateiktas 2.15 paveiksle *b*. Matoma, kad algoritmai atsparūs objekto skalės pakeitimui ir posūkiui.

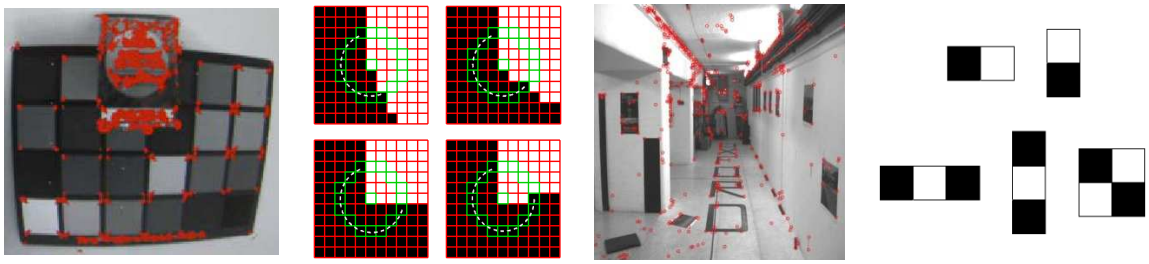
a) *b*)

2.15 pav. Vaizde pažymėti požymiai (*a*), požymių sutapdinimo rezultatas (*b*) [56]

LPLM įrenginiuose dažnai įgyvendinamas Harriso būdingųjų taškų vaizde išskyrimo algoritmas (angl. *Harris Corner Detector*) [8, 59]. Vaizdui taikomi gradiento pagal *x*, *y* koordinates operatoriai bei antros eilės išvestinės *x*, *y* ir *xy* kryptimis. Harriso algoritmas skirtas kampų ir taškų vaizde aptikimui, jo taikymo rezultatas pateiktas 2.16 paveiksle *a*. Kitų kampų vaizde išskyrimo algoritmų SUSAN (angl. *Smallest Univalue Segment Assimilating Nucleus*) ir FAST (angl. *Features from Accelerated Segment Test*) tyrimai pateikiami šaltiniuose [60–62]. Kampams aptikti taikomi 7×7 taškų dydžio filtrai (2.15 pav., *b*). Rezultatai pateikti 2.16 paveiksle *c*. Kampams aptikti taip pat galima taikyti morfologinius filtrus [63].

Veidams [64] arba kitiems objektams [65] vaizde sekti taikomi dvimačiai vilnelių modeliai (stačiakampiai *Haar* požymių filtrai) (2.16 pav. *d*). Požymio skaitine verte laikomas į baltą ir juodą

stačiakampių plotą patenkančių taškų skaisčio verčių skirtumas. Norint apskaičiuoti vaizdą, požymiai yra pasveriami erdvėje ir, ieškant atitikimų tarp jų, lyginami su esančiais duomenų bazėje.



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

2.16 pav. Vaizde rasti kampai (*a*, *c*) [8, 62], SUSAN kampų filtrai (*b*) [62], stačiakampiai *Haar* požymių filtrai (*d*) [64]

Lyginant su aptartais algoritmais, sekimo pagal būdinguosius taškus algoritmai yra įgyvendinami sudėtingiau, nes požymiams išskirti kuriami šimtų ar tūkstančių elementų filtrai, gauti požymiai koduojami ir sutapdinami su žinomais [55]. Geriausi sekimo rezultatai pasiekiami lygiagrečiai taikant kelis, vienas kitą papildančius sekimo algoritmus [21]. Nustatyta, kad didžiausia skaičiavimo resursų dalis (iki 90 %) sunaudojama būdingiesiems taškams išskirti taikant dvimačius kelių skalių filtrus [57]. Per pastaruosius penkerius metus objektų sekimo [4–10, 12–16, 25–30, 32–36] ir ypač požymių išskyrimu grįsti algoritmai vis dažniau perkeliami į LPLM įrenginius [39, 41–43, 49, 54, 55, 57, 60]. Didėjant vaizdo raiškai ir kadrų dažniui vis dažniau kils poreikis išlygiagretinti vaizdo apdorojimo procesus. Todėl kasmet kuriamos vis didesnio integracinio laipsnio loginės matricos tam, kad būtų pakankamai resursų algoritmams įgyvendinti.

2.5. Trumpas skyriaus apibendrinimas

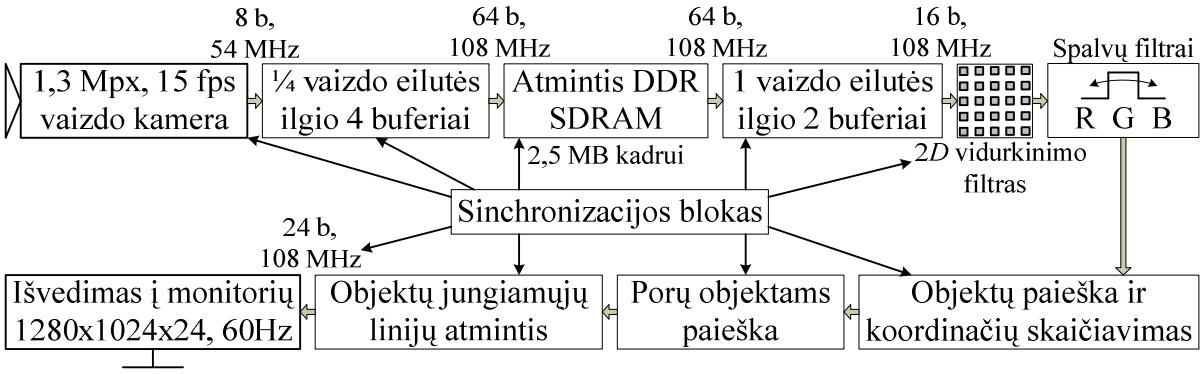
Iš apžvelgtos literatūros matyti, kad viena kitą atitinkančių vaizdo sričių paieškos ir sekimo pagal judesio aptikimą algoritmai įgyvendinami nesudėtingai. Sekimo pagal tekstūrą ir būdinguosius taškus algoritmai reikalauja atlikti daugiau skaičiavimų, nes požymius būtina aptikti, filtruoti, koduoti ir lyginti su žinomais. Aptarti algoritmai ar jų dalys yra tobulinami ir modifikuojami tam, kad būtų lengviau įgyvendinti ir racionaliau panaudoti loginių matricų resursai. Todėl šiame magistro darbe įgyvendinimui LPLM įrenginyje iš kiekvienos grupės pasirenkama bent po vieną objekto sekimo algoritmą, kurie naudojami sekti vieną ar kelis objektus ir veikti realiuoju laiku. Taip pat darbe siekiama ištirti objektų sekimo stabilumą keičiant apšviestumo lygį, fono sudėtingumą, objekto spalvą, atstumą iki kameros ir posūkio kampą.

3. OBJEKTŲ SEKIMO VAIZDE ALGORITMŲ ĮGYVENDINIMAS LPLM ĮRENGINYJE

Šiame skyriuje pateikiamas objektų sekimo vaizde algoritmų įgyvendinimas LPLM įrenginyje. Įgyvendinami viena kitą atitinkančių vaizdo sričių paieškos, judesio aptikimo, tekstūros kodavimo, būdingųjų taškų išskyrimo algoritmai. Visi algoritmai įgyvendinti aparatūroje programuojant VHDL kalba nekuriant įterptinių programinių procesorių, kurie neatskleidžia LPLM greitaveikos, procesų lygiagretumo bei aparatinių išteklių racionalaus panaudojimo privalumų [68].

3.1. Sekimo pagal spalvą algoritmų įgyvendinimas

Toliau pateikiamas sekimo pagal spalvą algoritmo įgyvendinimas LPLM *Virtex*-4 įrenginyje. Algoritmas skirtas pasirinktos (raudonos, žalios, mėlynos) spalvos objektams vaizde sekti bei atstumams nuo kameros iki objektų porų nustatyti. Vaizdui gauti taikoma spalvota 1,3 Mpx SXGA (angl. *Super eXtended Graphics Array*) raiškos 15 kadrų per sekundę dažnio vaizdo kamera. Kamera sukonfigūruota duomenis išvesti RGB565 formatu (5 bitai raudonai ir mėlynai spalvoms, 6 bitai – žaliai) per 8 bitų duomenų išvesties šyną. Vieno taško spalvos vertė LPLM siunčiama per du taktinius signalus. Rezultate turima 65536 (16 bitų) spalvos lygių taškui ir 37,5 MB per sekundę duomenų srautą. Vienam vaizdo kadrui saugoti reikalinga 2,5 MB dydžio atmintis (3.1 pav.) ir bent dvigubai didesnis už iš kameros gaunamų duomenų srautą *RAM* atminties duomenų sąsajos pralaidumas, nes vaizdo įrašymui ir skaitymui iš atminties taikoma viena 32 bitų šyna.

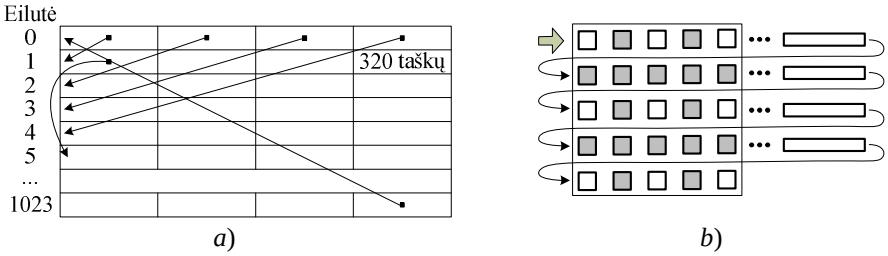


3.1 pav. Objektų sekimo pagal spalvą blokinė diagrama

DDR SDRAM (angl. *Double Data Rate Synchronous Dynamic Random Access Memory*) (toliau *RAM*) atminties taikymo privalumas tas, kad duomenys rašomi arba skaitomi abejais taktinio signalo frontais. Trūkimas tas, kad *RAM* atmintis turi vieną duomenų sąsają, todėl rašymas ir skaitymas vienu metu negalimas. Šiai problemai išspręsti operacijų susijusių su atmintimi vykdymas yra sinchronizuotas su duomenų išvedimu į monitorių. Monitoriaus ir kameros raiškos yra vienodos, bet kadrų dažnis skiriasi keturis kartus. Todėl per vienos eilutės išvedimo į monitorių

laiką iš kameros gaunama ¼ vaizdo eilutės dalis (320 taškų), kuri įrašoma į *RAM* per 80 taktinių impulsų po 64 bitus 108 MHz dažniu sekančios eilutės išvedimo pradžioje (3.2 pav., *a*). Po įrašymo per 320 taktinių impulsų iš *RAM* nuskaitoma viena vaizdo eilutė. 4 vaizdo eilučių spartinančios atminties elementai *RAM* atminties įėjime taikomi duomenims iš kameros kaupti ir sinchroniškai rašyti. Spartinančios atminties elementai sudaryti iš vidinių *BRAM* (angl. *Block RAM*) atminties blokų, kurių privalumas tas, kad jie turi dvi viena kitos atžvilgiu asinchroniškai veikiančias duomenų sąsajas. Ši savybė panaudota 16 bitų duomenis į spartinančios atminties elementus rašant 27 MHz dažniu ir iš spartinančios atminties elementų skaitant 64 bitų duomenis 108 MHz dažniu. Dar 2*BRAM* spartinančios atminties elementai pakaitomis taikomi *RAM* atminties išėjime tam, kad kol į pirmą įrašoma iš antro būtų skaitomi duomenis. Sinchronizacijos blokas pagal horizontalų ir vertikalų taškų skaitiklius laiduoja visų kitų blokų sinchronišką veikimą.

Iš *BRAM* spartinančios atminties elementų duomenis lygiagrečiai siunčiami į monitorių ir į dvimatį vidurkinimo filtro bloką. Slankiojančio vidurkinimo filtro sudarymas pateiktas 3.2 paveiksle *b*. Iš 5×5 dydžio filtro naudojama tik 16 pilka spalva pažymėtų elementų. Tai daroma dėl to, kad dalybos iš 16 operacijai įgyvendinti pakanka į dešinę per 4 bitus pastumti sumos rezultatą. Iš vidinių *BRAM* atminties blokų sudarytos keturios 1280 elementų ilgio ir 16 bitų gylio eilutės, iš kurių duomenys kiekvieno taktinio impulso metu perduodami į filtrą. Vidurkinimo (arba žemų dažnių) filtro taikymas yra būtinas, nes dauguma vaizdo kameros matricų turi sugadintų taškų, kurių vertės nekinta ir dažniausiai būna nulinės arba maksimalios vertės.



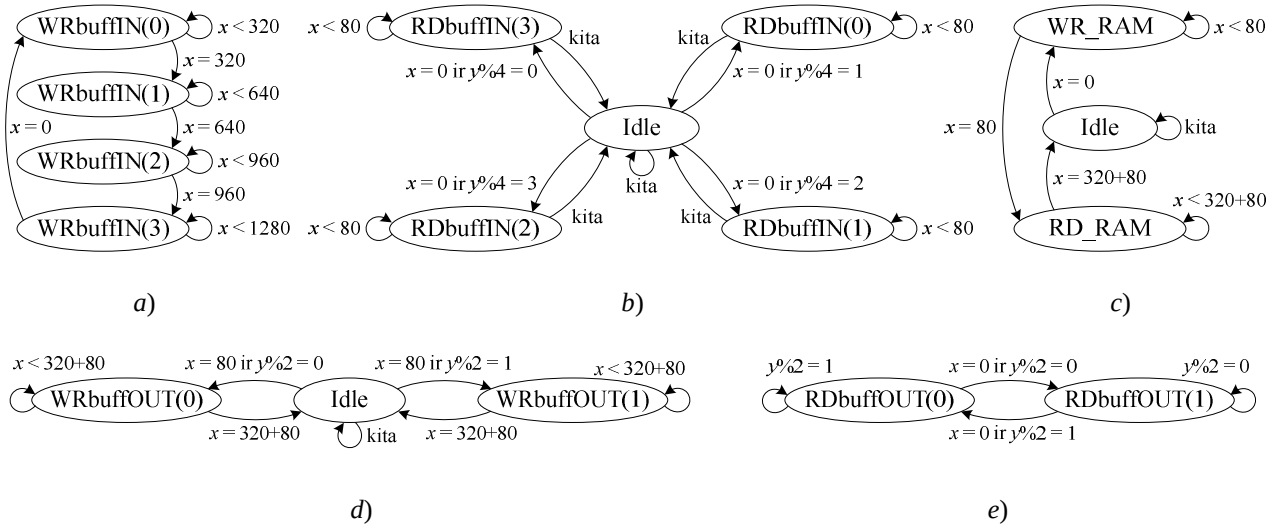
3.2 pav. Sinchronizuotas rašymas į *RAM* atmintį (*a*), slenkantis vidurkinimo filtras (*b*)

Nuoseklaus veikimo programos dalis, pavyzdžiui, duomenų rašymo į spartinančios atminties elementus ir atmintį diagramas patogu atvaizduoti baigtinių būvių mašinomis (angl. *Finite State Mashine*) (3.3 pav.). Penki būvių automatai veikia lygiagrečiai ir yra sudaryti iš sekančių būsenų:

- *Idle* – pradinės būsenos, kai nerašoma ir neskaitoma iš atminčių;
- *WRbuffIN*(0, ..., 3)– iš kameros gaunamų duomenų saugojimo į įėjimo spartinančios atminties elementus būsenos. Vienos vaizdo eilutės išvedimo į monitorių metu praeinama pervisas būsenas;
- *RDbuffIN*(0, ..., 3)– duomenų skaitymo iš įėjimo spartinančios atminties elemento būsenos. *y*/4 žymi vertikalaus skaitiklio dalybos iš keturių liekaną. Automatas būna

vienoje iš keturių būsenų periodiškai kas ketvirtąją eilutę 80 pirmų taškų išvedimo į monitorių metu;

- *WR_RAM* – ¼ vaizdo eilutės rašymo į RAM atmintį būsena, kurioje automatas būna 80 impulsų 108 MHz dažnio taktinio signalo;
- *RD_RAM*– vienos vaizdo eilutės skaitymo iš RAM atmintį būsena, kurioje esama 320 taktinių impulsų;
- *WRbuffOUT*(0, 1) – iš RAM gaunamų duomenų saugojimo į išėjimo spartinančios atminties elementus būsenos. Lyginių vaizdo eilučių išvedimo į monitorių metu duomenys rašomi į nulinį buferį, nelyginių – į pirmąjį;
- *RDbuffOUT*(0, 1) – duomenų skaitymo iš išėjimo spartinančios atminties elementų būsenos. Automatas keičia būseną kiekvienos naujos vaizdo eilutės pradžioje, kai $x = 0$.



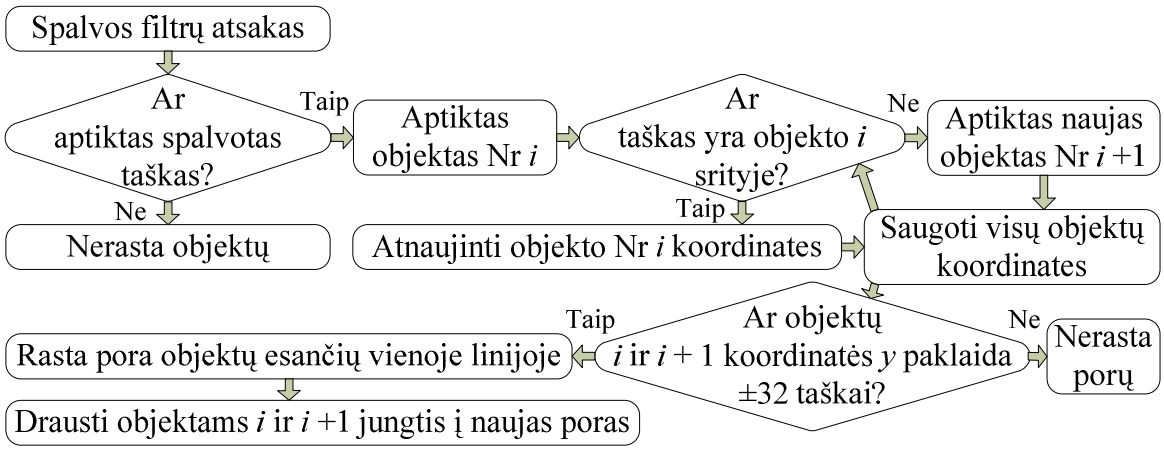
3.3 pav. Rašymo (*a*), skaitymo (*b*) iš įėjimo spartinančios atminties elemento, RAM valdymo (*c*), rašymo (*d*), skaitymo (*e*) iš išėjimo spartinančios atminties elemento baigtinių būvių mašinos

Objektų sekimo pagal spalvą algoritmas pateiktas 3.4 paveiksle. Iš vaizdo kameros gaunamas kadras išsaugomas atmintyje. Vaizdas apdorojamas lygiagrečiai su jo išvedimu į monitorių. Kitaskadras įrašomas į senojo vietą, todėl pakanka turėti 2,5 MB *RAM* atminties. Modifikuotas vidurkinimo 5×5 dydžio filtras taikomas triukšmams pašalinti kiekvienoje iš trijų spalvų dedamųjų. Slankiojančio filtro taikymas realiu laiku pavėlina filtro atsaką per dvi eilutes ir tris taškus filtruojamo vaizdo atžvilgiu, todėl sekamų objektų žymekliai, skirti atvaizduoti sekamo objekto padėtį monitoriuje, paderinami, juos pastumiant per atitinkamai du ir tris taškus į viršų bei į kairę. Yra galimybė pasirinkti vieną iš trijų sekamų spalvų taikant spalvos filtrą. Taško vertė priskiriama, pvz., žaliai spalvai, jeigu žalios spalvos kiekis taške yra trečdaliu didesnis už raudonos ir mėlynos. Papildomai įvesta apatinė žalios spalvos riba (01000), žemiau kurios taškai atmetami. Nufiltruoti pasirinktos spalvos taškai grupuojami su artimiausiais ir žymimi kaip objektai. Taškai, esantys

vienoje horizontalioje linijoje, jungiami į poras. Skaičiuojami atstumai tarp porų ir perskaičiuojamas atstumas iki kameros. Rezultatai realiu laiku išvedami į monitorių.

3.4 pav. Objektų sekimo pagal spalvą algoritmas

Spalvotų objektų paieškos vaizde ir jų jungimo į poras algoritmas detaliau paaiškintas 3.5 paveiksle. Algoritmas taikomas kiekvienam kadrui ir yra grįstas spalvotų taškų grupavimu ieškant artimiausių tos pačios spalvos kaimynų. Tikrinamas kiekvienas pasirinktos spalvos filtro atsakas. Jeigu aptiktas spalvotas taškas, tai reiškia, kad rastas objektas. Tikrinama ar taškas nepriklauso anksčiau aptiktiems objektams. Jeigu taškas patenka į jau aptikto objekto sritį, tuomet atnaujinamos to objekto koordinatės. Jeigu – ne, tai laikoma, kad rastas naujas objektas ir vienetu padidinamas objektų skaitiklis. Išsaugomos visų objektų koordinatės. Ieškant objektų porų tikrinama ar i ir $i + 1$ objektų y koordinačių paklaida mažesnė už 32 taškus. Jeigu – ne, tai porų nerasta. Jeigu – taip, tai rasta pora objektų esančių vienoje linijoje ir draudžiama jiems jungtis į naujas poras.

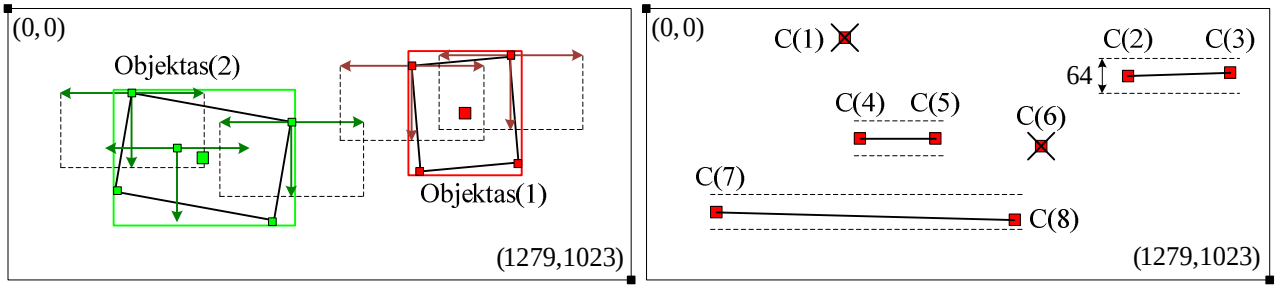


3.5 pav. Objektų aptikimo ir jungimo į poras algoritmas

Spalvotų taškų grupavimo į objektus principas pateiktas 3.6 paveiksle *a*. Spalvotų kaimyninių taškų paieška vykdoma iš viršaus į apačią. Kaimynams priskiriami taškai, kurie patenka į brūkšninę liniją pažymėtą stačiakampį. Kaimynų paieškos atstumas reguliuojamas rankiniu būdu. Yra galimybė pasirinkti kartotinį aštuoniems atstumą. Kuo mažesnis objektas ir kuo daugiau spalvotų taškų jame išskirta, tuo mažesnį kaimynų paieškos žingsnį galima taikyti. Mažas žingsnis sąlygoja, kad arti vienas kito esantys objektai nebus pažymėti bendru žymekliu. Didesnį (virš 64 taškų)

paieškos žingsnį patartina taikyti, kai blogėja apšvietimo sąlygos, didėja atstumai tarp aptinkamų spalvotų taškų arba, kai sekami objektai keičia mastelį arba, kai to reikalauja objekto forma.

Sekami objektai monitoriuje pažymimi stačiakampiais, kurių centrų koordinatės taikomos objektų porų paieškai (3.6 pav., *b*). Poros paieška taikoma kiekvienam objektui, jeigu jis dar nebuvo priskirtas porai. Paieška vykdoma horizontaliai (su ±32 taškų paklaida) nuo kiekvieno laisvo objekto centro. Esant trims centrams vienoje horizontalioje paieškos srityje (C(4), C(5), C(6)) pirmenybė jungtis į porą suteikiama kairiau išsidėsčiusiems. Aptiktos objektų poros monitoriuje atvaizduojamos jų centrus jungiančia linija.



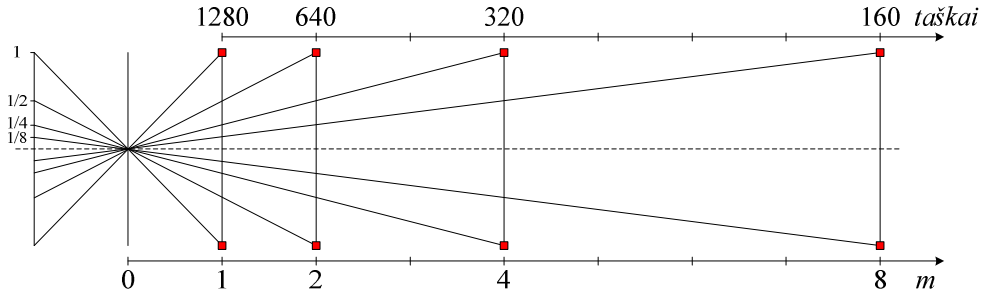
a) *b*)

3.6 pav. Spalvotų taškų grupavimas į objektus (*a*), objektų centrų jungimas į poras (*b*)

3.7 paveiksle pateikiama atstumo tarp taškų monitoriuje priklausomybė nuo realaus objekto atstumo iki kameros. Matoma, kad atstumas iki kameros yra atvirkščiai proporcingas objekto ilgiui. Objektų porų atstumui iki kameros nustatyti taikoma formulė:

$$d_{kam} = \frac{H_{len}}{x} d_0, \tag{3.1}$$

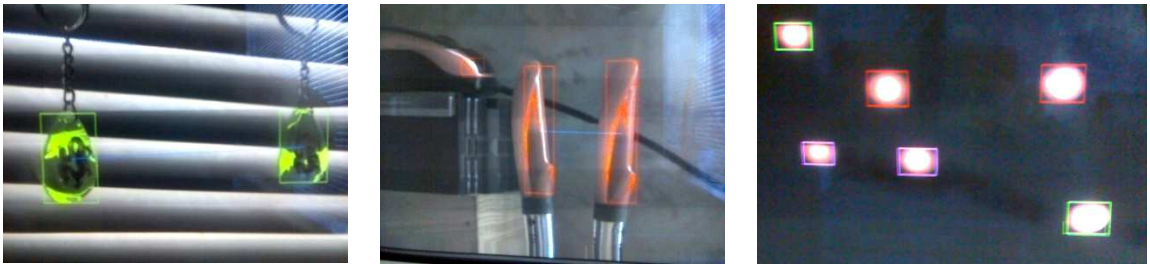
čia d_{kam} – perskaičiuotas atstumas iki kameros (metrais), x – atstumas tarp objektų centrų (taškais), d_0 – objekto atstumas iki kameros (metrais), kai centrai išsidėstę per 1280 taškus vienas nuo kito. Kintamasis H_{len} = 1280 priklauso nuo kameros taškų skaičių pagal horizontalę, kintamasis d_0 – nuo raiškos, lęšio ir objekto dydžio.



3.7 pav. Atstumo iki kameros perskaičiavimas per atstumus tarp objektų centrų

Programos testavimo rezultatai pateikti 3.8 paveiksle *a*, *b* vaizde sekant žalios ir raudonos spalvos objektus bei rastas jų poras jungiant mėlynos spalvos linija. Žali ir raudoni taškai vaizde

pažymi nufiltruotas sekamas spalvas. Nepaisant silpno apšvietimo ir trūkių tarp spalvotų sričių aptikti objektai teisingai pažymimi, sekami ir jungiami į poras.



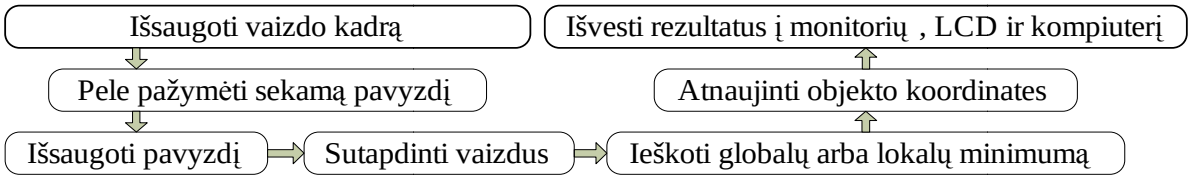
a) *b*) *c*)

3.8 pav. Žalios spalvos objektų sekimas (*a*), raudonos spalvos objektų sekimas (*b*), raudonų spalvų sekimas

3.8 paveiksle *c* pateikti raudonos spalvos šviesų vaizde sekimo rezultatai. Šviesos, esančios vienodame lygyje, teisingai pažymėtos raudonos ir violetinės spalvos stačiakampiais. Šviesos šaltiniai neturintys porų žymimos žalios spalvos stačiakampiais.

3.2. Viena kitą atitinkančių vaizdo sričių paieškos algoritmų įgyvendinimas

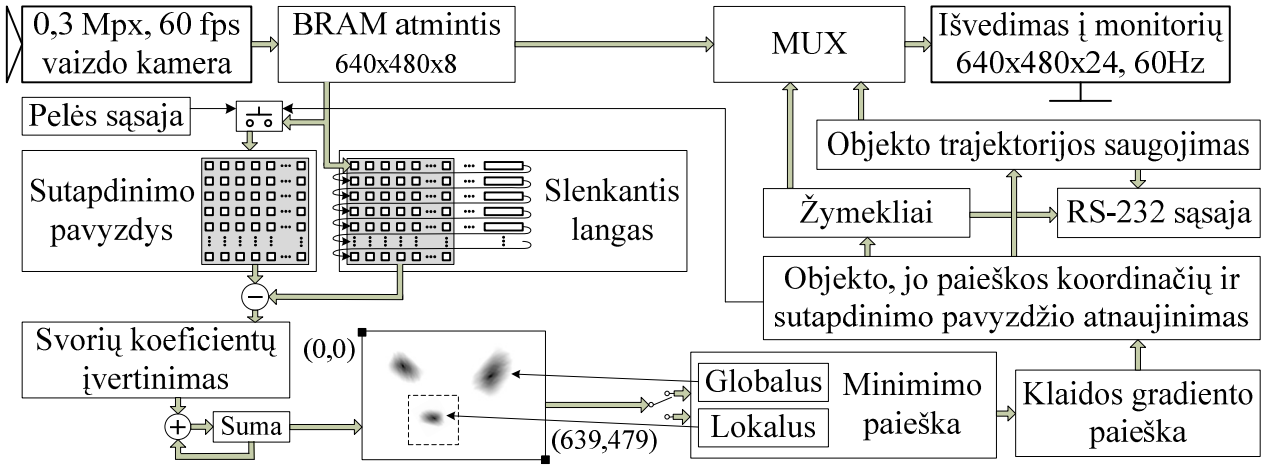
Toliau pateikiamas sekimo pagal vaizdo pavyzdžio sutapdinimą su kadru algoritmo įgyvendinimas LPLM *Virtex*-4 įrenginyje. Algoritmas skirtas pele monitoriuje pažymėtam objektui sekti. Objekto dydis ribojamas iki slenkančio sutapdinimo filtro dydžio. Apibendrintas algoritmas pateiktas 3.9 paveiksle. Iš kameros gaunamas vaizdo kadras ir pele pažymėto objekto pavyzdys išsaugomi vidinėje LPLM atmintyje. Sutapdinant šiuos vaizdus ieškoma globalaus arba lokalaus minimumo. Išsaugomos ekstremumų vietas aprašančios koordinatės. Kito kadro metu objekto žymeklis perkeliamas į naują vietą ir atvaizduojamas monitoriuje.



3.9 pav. Objekto sekimo pagal pavyzdžio sutapdinimą su vaizdu algoritmas

Objekto sekimo pagal pavyzdį blokinė schema pateikta 3.10 paveiksle. Vaizdui gauti naudojama 640×480 taškų raiškos 60 kadrų per sekundę dažnio vaizdo kamera. Kamera veikia maksimaliu 27 MHz dažniu, duomenų šyna 10 bitų, duomenų perdavimo sparta apie 22 MB per sekundę. Duomenys saugomi vidinėje LPLM atmintyje sudarytoje iš BRAM blokų. Atmintis skirta tik vieno vaizdo kadro taškų aštuoniems aukščiausiems skaisčio bitams saugoti. Pelės mygtuku pažymėjus sekamą objektą jo pavyzdys išsaugomas atmintyje. Slenkantis langas yra tokio pat dydžio kaip sutapdinimo pavyzdys. Kiekvieną taktinio signalo impulsą slenkantis langas

pastumiamas per vieną stulpelį. Sinchroniškas kiekvienos lango eilutės elementų pastūmimas per vieną tašką įgyvendintas pasitelkiant vaizdo eilučių spartinančios atminties elementus sudarytus iš vidinių BRAM atminčių. Spartinančios atminties elementų skaičius yra pasirenkamas vienetu mažesnis už slenkančio lango aukštį. Tai reikalinga tam, kad iš paskutinės lango eilutės išeinančios vaizdo taškų skaisčio vertės nėra naudojamos skaičiavimuose, todėl jų saugoti nebūtina.

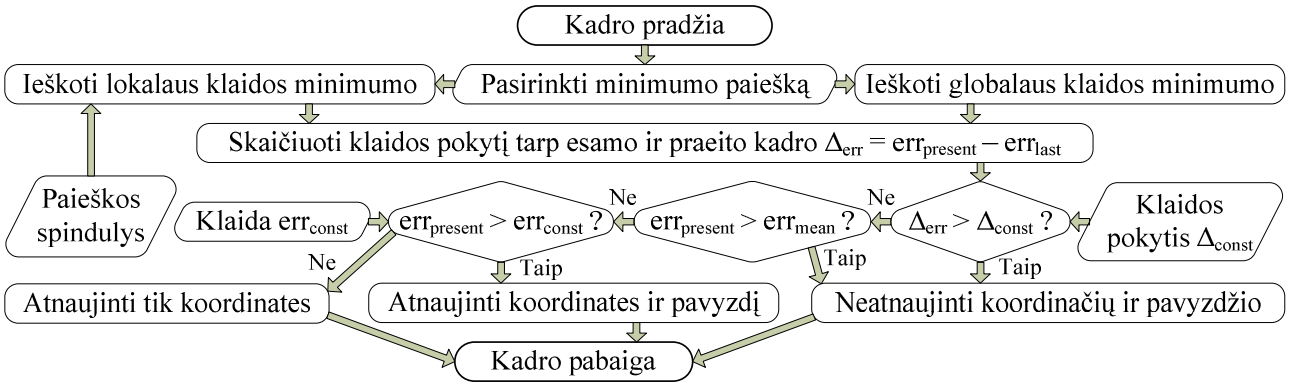


3.10 pav. Objekto sekimo pagal pavyzdį blokinė schema

Vaizdams sutapdinti skaičiuojama skirtumų tarp slenkančio lango ir pavyzdžio skaisčio taškų pasvertoji suma. Skirtumų svorių koeficientus k_i galima keisti ir jie gali įgyti reikšmę iš aibės $[1, 2, 4, 8]$ (3.12 pav., *a*). Kuo mažesnė skirtumų sumos vertė, tuo labiau pavyzdys sutampa su vaizdu. Skirtumų suma atvaizduota plokštumoje (3.10 pav.). Tamsūs taškai joje parodo skirtumų sumų minimumų vietas. Minimumo paieškos algoritme yra galimybė pasirinkti globalaus arba lokalaus minimumo sekimą. Globalaus minimumo paieškos privalumas tas, kad paieška yra nesudėtingai įgyvendinama. Trūkumas – globalus minimumas ne visada nurodo sutapdinamo pavyzdžio vietą, kai vaizde yra keli panašūs objektai į sekamą. Lokalaus minimumo paieškos privalumas tas, kad nepaisant didėjančios skirtumų sumos (objektui nežymiai pasisukus, nutolus, pasikeitus apšvietimui) pavyzdys yra teisingai aptinkamas. Trūkumas – sekimo metu objektas turi būti visą laiką matomas, nes lokalios paieškos srities koordinatės atnaujinamos kartu su sekamo objekto koordinatėmis. Objektui dingus paieškos srities koordinatės išlieka senos. Objektui pasirodžius už paieškos srities ribų jis nebus aptiktas. Sekamo pavyzdžio koordinatės yra atnaujinamos atsižvelgiant į klaidos gradiento pokytį. Pagal koordinates į monitorių išvedamas objekto ir pavyzdžio paieškos sekančiame kadre stačiakampiai žymekliai, o į LCD (angl. *Liquid Crystal Display*) vaizduoklį – objekto koordinatės ir pasvertoji skirtumų suma bei kiti parametrai. Monitoriaus ir vaizdo kameros raiška bei kadrų atnaujinimo dažnis yra identiški.

Sekamo pavyzdžio ir jo paieškos koordinačių atnaujinimo algoritmas pateiktas 3.11 paveiksle. Pagal pasirinktą minimumo paieškos būdą sprendžiama kokio (lokalaus ar globalaus) minimumo

ieškoti. Lokaliam minimumui rasti papildomai įvedamas paieškos spindulio parametras. Kiekvieną kadrą skaičiuojamas vaizdų sutapdinimo klaidos pokytis Δ_{err} tarp esamame kadre rastos klaidos $err_{present}$ ir praeito kadro klaidos err_{last}. Vertinama tik teigiama Δ_{err} dalis, t.y., ieškoma staigaus klaidos padidėjimo. Jeigu klaidos pokytis Δ_{err} didesnis už pasirinktą Δ_{const}, tai objekto koordinatės ir pats pavyzdys neatnaujinami, nes teigiama, kad staigus klaidos pokytis atsiranda dėl sekamo objekto užstojimo arba dalinio persidengimo su kitu. Jeigu klaidos pokytis yra leistinose ribose, tai tikrinama ar esama klaida $err_{present}$ vis dar yra didesnė už vidutinę gradiento padidėjimo metu išsaugotą klaidą err_{mean}. Jeigu taip – pavyzdys ir koordinatės neatnaujinami, priešingu atveju teigiama, kad objektas grįžo į paieškos sritį. Trečioje sąlygoje tikrinama, ar esamo kadro klaida didesnė už pasirinktą err_{const}. Kadangi klaida nepadidėjo staigiai, bet tolygiai priartėjo ir viršijo err_{const}, tai reiškia, kad pats objektas pasisuko arba pakito apšvietimas, arba galimai dalinai liko užstotas. Todėl atnaujinamos objekto, jo paieškos koordinatės bei pats pavyzdys. Jeigu $err_{present} < err_{const}$, tai atnaujinamos tik objekto koordinatės, o sutapdinimo pavyzdys lieka senas. Koordinačių atnaujinimas atliekamas prieš pat esamo kadro pabaigą, kad kito kadro metu būtų taikomi nauji duomenys.

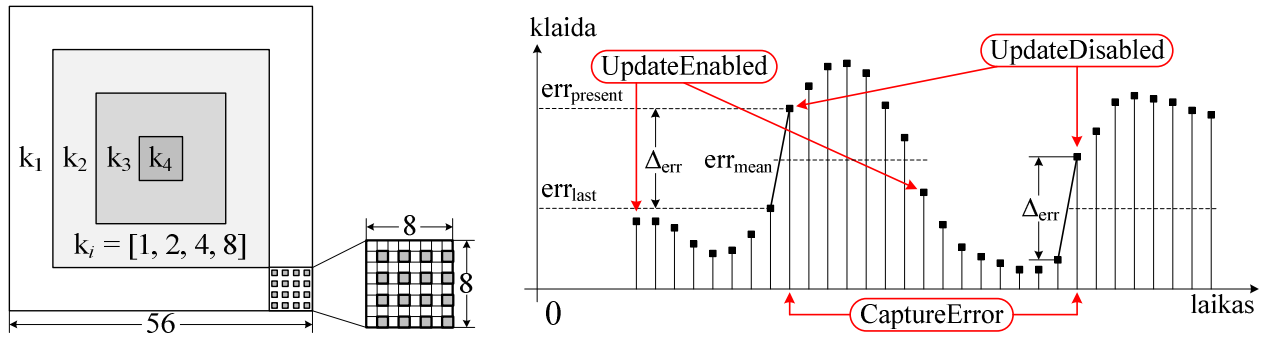


3.11 pav. Sekamo pavyzdžio bei jo koordinačių atnaujinimo algoritmas

Sutapdinimo pavyzdžio filtras pateiktas 3.12 paveiksle *a*. Jis sudarytas iš keturių sričių, kurių įtaką bendrai klaidai paderinama keičiant koeficientų k_i vertes. Dėl ribotų loginės matricos xc4vsx35 resursų, veikimui realiu laiku įgyvendintas tik 56×56 taškų raiškos pavyzdžio sutapdinimas. Filtras suskirstytas į 49 sritis po 8×8 taškų. Kiekvienoje srityje sutapdinimui taikoma 16 taškų (kas ketvirtas taškas iš 64 taškų srities), tokiu būdu 4 kartus praplečiamas pavyzdžio dydis dėl į kas antrą eilutę ir stulpelį patenkančių taškų adresavimo. Nuo krašto link centro didėjančius filtro koeficientus patogu taikyti norint padidinti sekimo stabilumą, kai į kraštinius filtro elementus patenka besikeičiantis fono vaizdas, taip didindamas sutapdinimo klaidą.

Klaidos kitimo laike grafikas pateiktas 3.12 paveiksle *b*. Jame raudona spalva pažymėtos būvių pradžios, į kurias pereina koordinačių atnaujinimo algoritmas. Žingsnis tarp stulpelių atitinka vieno kadro trukmę. Kai teigiamas Δ_{err} gradientas viršija Δ_{const} vertę, tai išsaugoma klaidos err_{mean} vertė ir

pavyzdys bei jo koordinatės neatnaujinamos. Objekto koordinatės pradedamos atnaujinti, kai klaida sumažėja iki err_{mean}.

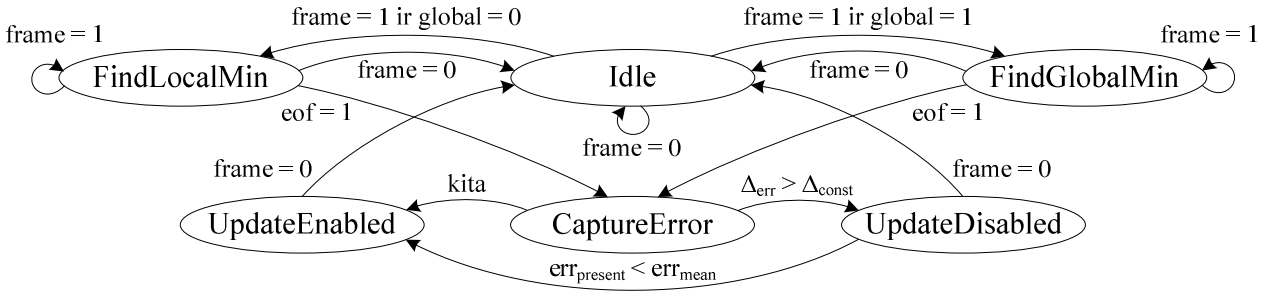


a) *b*)

3.12 pav. Sutapdinimo pavyzdžio filtras ir koeficientai k_i (*a*), klaidos kitimo laike pavyzdys (*b*)

Vaizde sekamo pavyzdžio atnaujinimo algoritmo baigtinių būvių diagrama pateikta 3.13 paveiksle. Būvių automatą sudaro 6 būsenos:

- *Idle* – pradinė būsena, kurioje automatas būna tą laiką, kai į monitorių vaizdas neišvedamas. Į šią būseną grįžtama, kai parametras *frame* = 0;
- *FindLocalMin* – lokalaus klaidos minimumo paieškos būsena. Joje automatas būna taškų į monitorių išvedimo metu, kai parametras *global* = 0;
- *FindLocalMax* – globalaus klaidos minimumo paieškos būsena. Joje automatas būna taškų į monitorių išvedimo metu, kai parametras *global* = 1;
- *CaptureError* – klaidos pokyčio skaičiavimo būsena. Į šią būseną pereinama, kai aptinkama kadro pabaiga (*eof* = 1);
- *UpdateDisabled* – sekamo pavyzdžio atnaujinimo išjungimo būsena, į kurią automatas patenka, kai klaidos pokytis viršija pasirinktąjį;
- *UpdateEnabled* – sekamo pavyzdžio atnaujinimo įjungimo būsena, į kurią automatas patenka, kai $\Delta_{err} < \Delta_{const}$ arba $err_{present} < err_{mean}$. Kiekvienoje iš trijų paskutinių būsenų automatas būna tik vieno taktinio signalo metu;



3.13 pav. Sekamo pavyzdžio atnaujinimo baigtinių būvių mašina

3.14 paveiksle pateikiami objektų sekimo taikant vaizdo pavyzdžio sutapdinimą su kadru algoritmo testavimo rezultatai. Pele pažymėtas USB atmintinės sutapdinimo pavyzdys sėkmingai sekamas vaizde netaikant filtro svorių koeficientų (3.14 pav., *a*).

a) *b*)

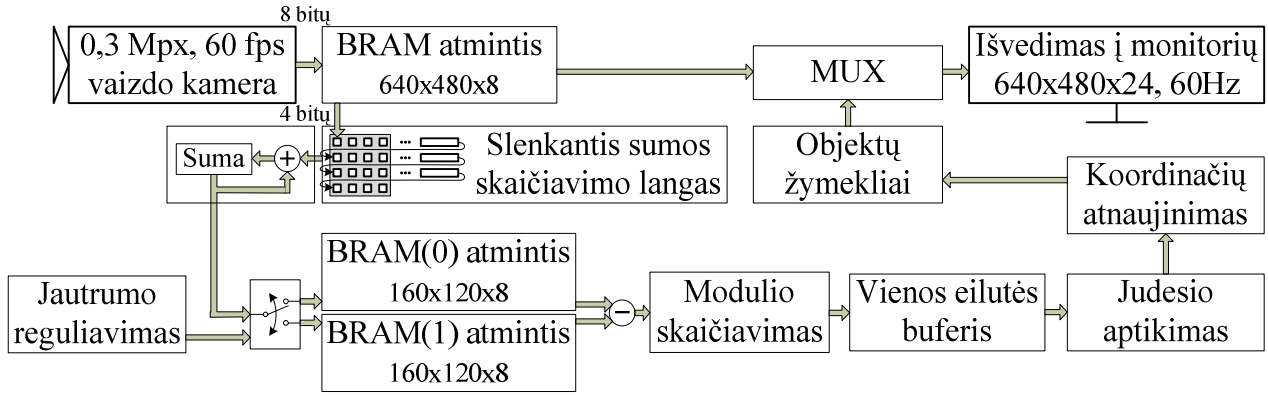
3.14 pav. USB atmintinės (*a*), markerio (*b*) sekimas taikant vaizdo pavyzdžio sutapdinimą su kadru

Markerio sekimo vaizde pavyzdys taikant sutapdinimo filtro svorių koeficientus pateiktas 3.14 paveiksle *b*. Pastebėta, kad objektas sėkmingai sekamas keičiant fono vaizdą, nes fono įtaka pašalinama sumažinus sekamo pavyzdžio kraštinių elementų skirtumų poveikį sumos vertei.

3.3. Sekimo pagal judesio aptikimą algoritmų įgyvendinimas

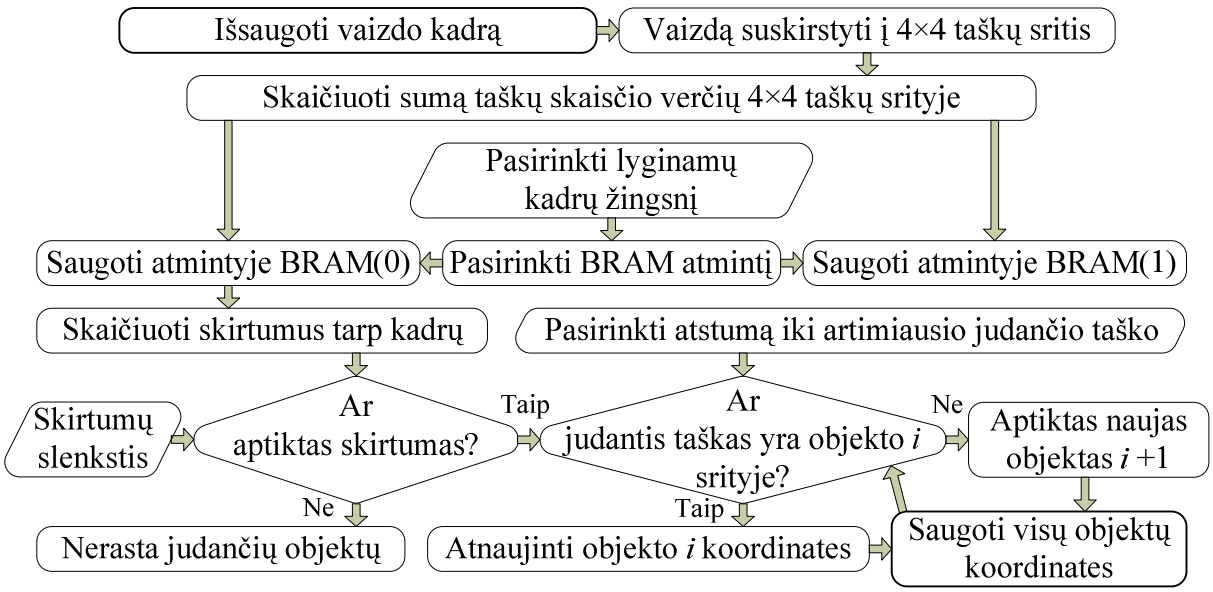
Šiame poskyryje planuojama įgyvendinti objektų sekimo pagal judesio vaizde aptikimą ir fono vaizde pašalinimą algoritmai.

Toliau pateikiamas objektų sekimo pagal judesio vaizde aptikimą algoritmo įgyvendinimas LPLM *Virtex*-4 įrenginyje. Algoritmas turi sekti kelis judančius objektus ir monitoriuje juos pažymėti stačiakampiais žymekliais. Algoritmo blokinė schema pateikta 3.15 paveiksle. Vaizdui gauti taikoma 640×480 taškų raiškos 60 kadrų per sekundę dažnio nespalvota vaizdo kamera. Vidinėje LPLM įrenginio BRAM atmintyje saugomas 8 bitų skaisčio gylio vienas vaizdo kadras. Į atmintį vaizdas įrašomas 27 MHz, o iš jos skaitomas ir į monitorių išvedamas 25 MHz dažniu. Judesiui vaizde aptikti būtina skaičiuoti skirtumus tarp skaisčio taškų kadruose. Nustatyta, kad skaičiuojant skirtumus tarp kiekvieno taško kadruose klaidingai aptinkama daug judančių taškų. Todėl judesio aptikimui taikomas slenkantis 16 gretimų skaisčio taškų sumos skaičiavimo langas. Slenkančio lango taikymas 16 kartų sumažina judančių taškų vaizde paieškos plotą ir padidina atsparumą atsitiktiniams staigiems pavienių taškų skaisčio verčių pokyčiams. Du 160×120 taškų raiškos vaizdo kadrai saugomi vidinės BRAM(0) ir BRAM(1) atminties blokuose. Jautrumas judesiui reguliuojamas pasirenkant kas kelintą kadrą skaičiuoti skirtumus tarp skaisčio verčių vaizde. Vienos eilutės spartinančios atminties elementas skirtas saugoti paskutinės vaizdo eilutės taškų skirtumo modulių vertes. Judesio aptikimo bloke įgyvendintas judančių taškų grupavimas į objektus bei jų koordinačių nustatymas. Pagal apskaičiuotas koordinates monitoriuje nubraižomi judančių objektų stačiakampiai žymekliai.



3.15 pav. Objektų sekimo pagal judesio aptikimą algoritmo blokinė schema

Judančių objektų koordinačių nustatymo algoritmas detaliau pateiktas 3.16 paveiksle. Algoritmas veikia realiu laiku ir sekančio kadro atvaizdavimo monitoriuje metu išvedami objektų žymekliai į ankstesniame kadre apskaičiuotą vietą. Išsaugotas vaizdo kadras suskirstomas į 4×4 taškų dydžio sritis (3.17 pav. *a*). Nustatyta, kad žemiausi 4 skaisčio bitai neįtakoja objekto aptikimo tikslumo, todėl 16 taškų skaisčio verčių sumai skaičiuoti taikomi 4 aukščiausi vaizdo skaisčio bitai. Dėl to trečdaliu sutaupoma vidinės BRAM atminties blokų 160×120 raiškos vaizdui saugoti. Pasirenkamu lyginamų kadrų žingsniu nustatomas jautris judesiui t.y. kokiu dažniu skaičiuoti skirtumus tarp kadru. Vaizdas į BRAM(0) ir BRAM(1) atminties blokus saugomas pakaitomis, kai algoritmas veikia judesio aptikimo režimu. Taip pat palikta galimybė pasirinkti objektų sekimą taikant fono pašalinimą. Tokiu atveju į BRAM(0) atmintį kartą išsaugotas vaizdas yra lyginamas su kiekvienu naujai saugomu kadru BRAM(1) atmintyje. Dėl to vaizde lengvai aptinkami ir sekami objektai, kurių taškų skaistis skiriasi nuo išsaugoto fono.

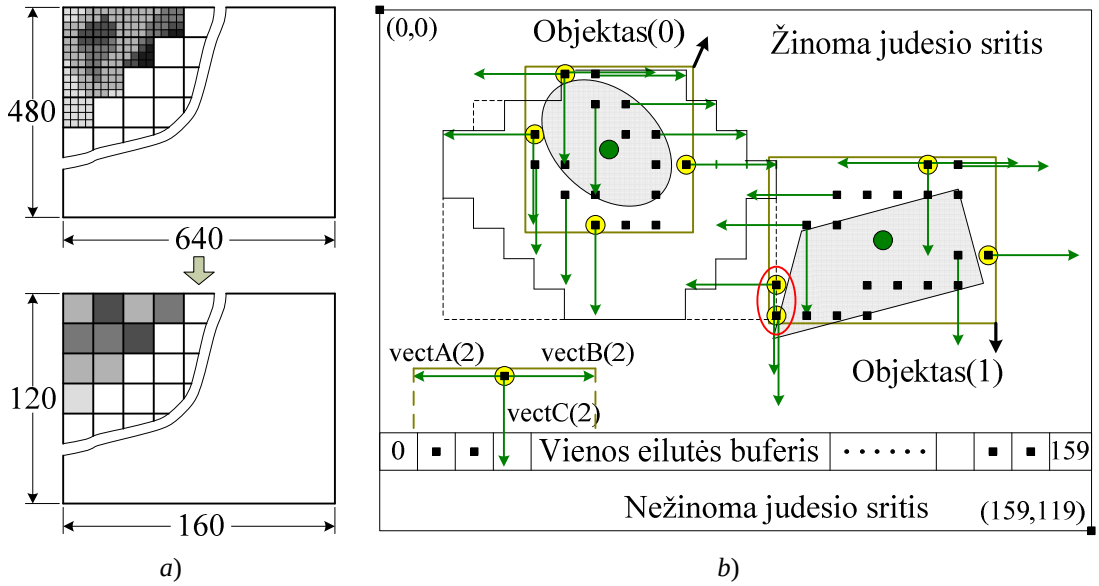


3.16 pav. Judančių objektų vaizde koordinačių nustatymo algoritmas

Aptiktas skirtumas tarp kadro taškų yra lyginamas su pasirinkta slenkstine vertę, kuri apibrėžia fonui ir objektui priklausančių taškų skaisčio verčių panašumo lygį. Kuo šis slenkstis aukštesnis,

tuo labiau sekamas objektas turi išsiskirti nuo fono, kuo žemesnis, tuo jautresnis nežymiems skaisčio pokyčiams. Aukštą slenkstį patartina taikyti, kai yra žinoma, kad sekamo objekto ir fono taškų skaisčio skirtumai didesni už slenkstį. Žemas slenkstis taikomas, kai svarbu aptikti ir sekti visus judančius objektus. Aptikus judantį tašką tikrinama ar jis priklauso jau aptikto objekto sričiai. Jeigu – taip, tai atnaujinamos to objekto koordinatės. Jeigu – ne, tai reiškia, kad aptiktas naujas objektas ir išsaugomos jo koordinatės bei vienetu padidinamas apriktų objektų skaitiklis $i = i + 1$. Tikrinama ar sekančio judančio taško koordinatės nepatenka į i objektų sritis.

Judančių, juoda spalva pažymėtų taškų grupavimo į objektus principas pateiktas 3.17 paveiksle *b*. Vienos vaizdo eilutės spartinančios atminties elementas taikomas saugoti paskutinės vaizdo eilutės taškų skirtumo modulių vertes tam, kad pagal poreikį galima būtų monitoriuje atvaizduoti judančius taškus. Žalios spalvos rodyklės nurodo reguliuojamą atstumą iki kaimyninių judančių taškų, kurie priskiriami vienam objektui. Brūkšninės linijos kontūras pažymi objekto(0) sritį. Ištisinės linijos kontūras pažymi objekto(0) sritį, kai taikomas srities mažinimas sekančiose eilutėse neaptikus taškų praplečiančių paieškos sritį. Paieškos srities mažinimą patartina taikyti, kai žinoma, kad sekami objektai yra arti vietas kito ir norima, kad būtų aptikti atskirai. Raudonos spalvos elipse pažymėti du klaidingai objektui(0) priskiriami taškai, jeigu netaikomas srities mažinimas. Geltona spalva pažymėti pirmi labiausiai į viršų, apačią, kairę ir dešinę aptikti objekto taškai. Šių ir perskaičiuotų žalios spalvos centro taškų koordinatės saugomos ir naudojamos kito kadro metu judančių objektų žymeklių išvedimui į monitorių.



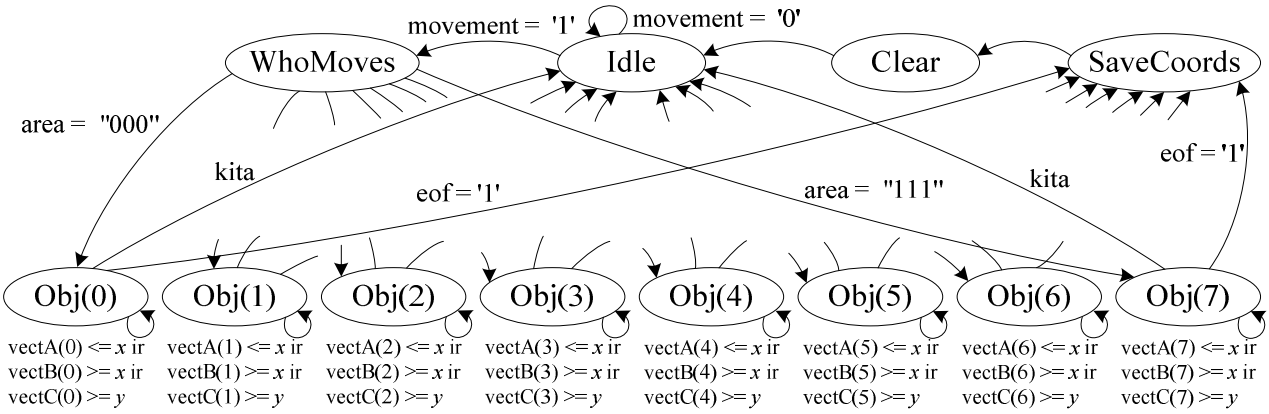
a) *b*)

3.17 pav. Vaizdo mastelio pakeitimo iliustracija (*a*). Judančių taškų apjungimo iliustracija (*b*)

Judančių taškų vaizde jungimo algoritmo baigtinių būvių diagrama pateikta 3.18 paveiksle. Būvių automatą sudaro 12 būsenų:

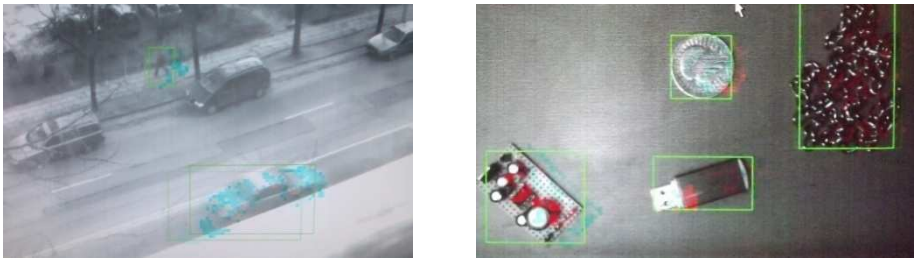
- *Idle* – pradinė būsena, kurioje automatas būna tą laiką, kai neaptikta judesio;

- *WhoMoves* – judančio taško priskyrimo vienai iš objektų sričiai būsena. Generuojamas vienas iš aštuonių *area* skaičius nurodantis, kuris objektas aptiktas arba, kurio objekto koordinatės bus atnaujinamos;
- *Obj*(0, ..., 7) – objektų koordinačių skaičiavimo būsenos. Objekto *i* koordinatės atnaujinamos tol, kol jusiantis taškas patenka į *i*-tojo objekto paieškos sritį tenkinant sąlygas *vectA*(*i*) <= *x* ir *vectB*(*i*) >= *x* ir *vectC*(*i*) >= *y*. Jeigu neaptikta kadro pabaiga, tai grįžtama į *Idle* būsena, kurioje tikrinama ar juda sekantis taškas;
- *SaveCoords* – sekamų objektų žymeklių ir centro taškų koordinačių saugojimo būsena, į kurią automatas patenka kiekvieno vaizdo kadro pabaigoje;
- *Clear* – *vectA*(*i*), *vectB*(*i*), *vectC*(*i*) bei kitų kintamųjų pradinių verčių prieš kiekvieną naują kadrą nustatymo būsena.



3.18 pav. Judančių taškų vaizde jungimo baigtinių būvių mašinos veikimo iliustracija

Objektų sekimo pagal judesio vaizde aptikimą algoritmo testavimo rezultatai pateikti 3.19 paveiksle *a*. Matoma, kad vaizde teisingai aptikti du judantys objektai, kurie pažymėti žaliais stačiakampio formos žymekliais. Mėlyni taškai pažymi skirtumus tarp kadrų. Nustatyta, kad kuo objektas greičiau juda, tuo mažesnį lyginamų kadrų žingsnį reikia taikyti, kad žalios spalvos stačiakampio formos objekto žymeklis nebūtų praplečiamas dėl įnešamo vėlinimo. Tačiau mažinant lyginamų kadrų žingsnį lėtai judantys objektai yra dalinai aptinkami.



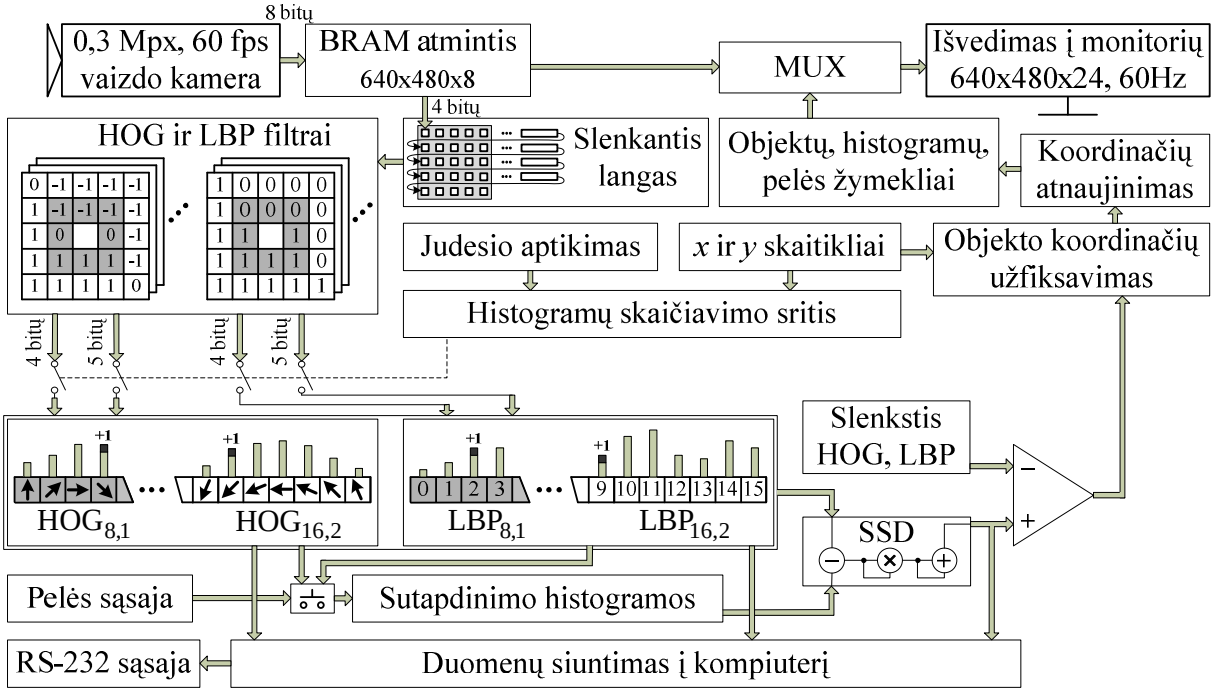
a) *b*)

3.19 pav. Objektų sekimas taikant judesio vaizde aptikimą (*a*) ir fono pašalinimą (*b*)

Kelių objektų sekimo taikant fono pašalinimą testavimo rezultatas pateiktas 3.19 paveiksle *b*. Išsaugotas fono vaizdas lyginamas su kiekvienu nauju kadru. Aptikti skirtumų taškai grupuojami ir pažymimi kaip objektai. Nustatyta, kad taikant judesio vaizde aptikimo ir fono pašalinimo algoritmus vaizdo kamera nuturi judėti, nes aptinkami atsitiktiniai skirtumai tarp kadrų, kurie irgi algoritmo pažymimi kaip objektai.

3.4. Sekimo pagal tekstūrą algoritmų įgyvendinimas

Toliau pateikiamas objektų sekimo vaizde algoritmų įgyvendinimas taikant tekstūros deskriptorius LPLM *Virtex*-4 įrenginyje. Vaizdo kodavimo metodai HOG ir LBP yra panašūs pagal taikomų filtrų dydį, histogramų sudarymo principą, todėl bus įgyvendinami lygiagrečiai. Algoritmas turi sekti vieną pasirenkamą objektą ir monitoriuje jį pažymėti stačiakampiu žymekliu. Algoritmo veikiančio 25 MHz dažniu blokinė schema pateikta 3.20 paveiksle. Vaizdui gauti taikoma VGA raiškos 60 kadrų per sekundę dažnio vaizdo kamera. 8 bitų gylio vaizdo taškų skaisčio vertės saugomos vidinėje BRAM atmintyje. Slenkančiam langui sukurti taikomos keturios BRAM eilutės. Vaizdo analizei naudojami tik keturi aukščiausi taškų skaisčio bitai. Visi 5×5 dydžio slenkančiame lange saugomi duomenys yra lygiagrečiai pasiekiami HOG ir LBP filtrams, kurių sudėtis bus aptarta vėliau. Pirmosios skalės HOG ir LBP filtrų atsakai gali būti užkoduoti keturiais bitais, antrosios – penkiais. Kai filtrų bloke sugeneruojamas stulpelio numerį atitinkantis kodas, histogramos stulpelio vertė padidinama vienetu.



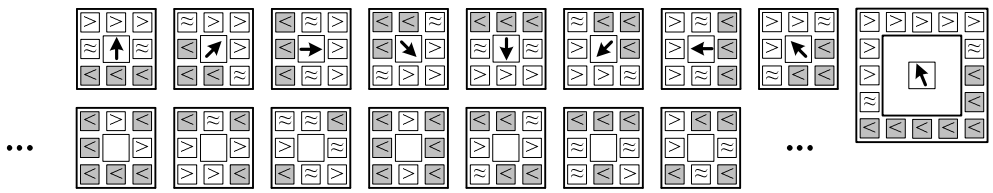
3.20 pav. Objektų sekimo taikant tekstūros deskriptorius HOG ir LBP algoritmo blokinė schema

Ribota loginės matricos atmintis neleidžia įgyvendinti realiu laiku veikiančio slenkančio lango, kuris būtų didesnis už 24×24 taškų sekamą objektą. Taktinio dažnio didinimas iki maksimaliai

leistino 200 MHz tik iki 8 kartų padidintų sekamo objekto išmatavimus, tačiau reikėtų taikyti išorinę atmintį tarpiniams rezultatams saugoti, kurios valdiklis bei papildomas histogramų sudarymo algoritmo įvedimas papildomai sunaudotų LPLM resursus. Taktinio dažnio kėlimas veda prie didesnio srovės kiekio sunaudojimo ir loginės matricos aušinimo. Sekamo objekto išmatavimai gali būti padidinti mažinant filtruojamų kadrų dažnį ir apdorojant ne kiekvieną kadrą, bet, pvz., kas dešimtą. Tačiau greitai judančių objektų žymekliai nebūtų tolygiai atnaujinami monitoriuje ir atnaujinimo dažnis mažėtų didėjant pažymėtam objektui. Papildomas koordinačių prognozavimo algoritmų taikymas kiek pagerintų tolygų žymeklių atvaizdavimą.

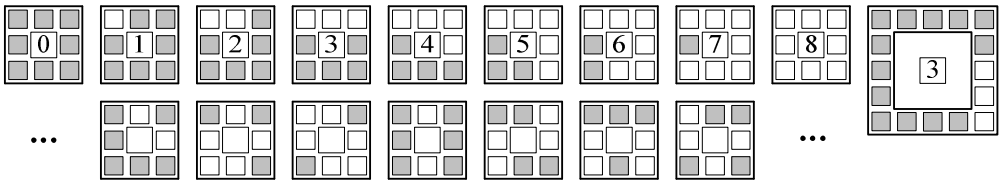
Norint objektą sekti realiu laiku, apdoroti kiekvieną kadrą ir nedidinti taktinio dažnio siūloma HOG ir LBP histogramas skaičiuoti tik judantiems objektams. Judesio aptikimo bloke skaičiuojamos visų judančių objektų centrų koordinatės, pagal kurias nustatomos sritys vaizde, kuriose skaičiuojamos histogramos. Pele pažymėto objekto HOG ir LBP histogramos išsaugomos atmintyje ir kiekvieną kadrą sutapdinamos su paskaičiuotomis judantiems objektams. Deskriptorių sutapdinimas įvykdomas skaičiuojant skirtumų kvadratų sumas SSD_{HOG} ir SSD_{LBP}. Paskaičiuotos sumos palyginamos su slenkstinėmis vertėmis, kurias viršijus laikoma, kad juda sekamas objektas, todėl saugomos jo koordinatės ir atnaujinama žymeklio padėtis monitoriuje. Į monitorių realiu laiku išvedamos sekamo objekto, pradinės sutapdinimo histogramos bei jų skirtumų kvadratų sumos. Šie duomenys tolimesnei analizei gali būti nusiųsti kompiuteriui į MATLAB aplinką per RS-232 sąsają. Judesio aptikimo parametrai ir histogramų slenksčiai pagal poreikį paderinami MATLAB aplinkoje ir siunčiami į LPLM.

HOG histogramai kurti taikomi 3.21 paveiksle pateikti filtrai. Vaizdo taškų skaisčio gradientams pirmoje skalėje skaičiuoti taikomi pirmoje eilutėje pateikti aštuoni 3×3 dydžio filtrai. Gradiento kryptis centriniam taškui nustatoma atsižvelgiant į kaimyninių taškų skaisčio vertes. Žymenys „> = <“ nurodo, kokio dydžio turi būti kaimyninių taškų vertės centrinio atžvilgiu. Antroje skalėje taikoma 16 HOG filtrų, iš kurių vienas pateiktas dešinėje 3.21 paveikslo pusėje. Pilkai pažymėti filtrų elementai vaizdžiau pateikia neigiamą gradiento kryptį. Antroje eilutėje pateikti keli histogramai sudaryti netaikytini HOG filtrai. Nustatyta, kad šiuos filtrus atitinkančių taškų vaizde yra daugiausia. Tačiau jų vertinti ir koduoti tolimesniam histogramų sudarymui nepatartina, nes jie atspindi nestabilias gradientų kryptis.



3.21 pav. Vaizdo kodavimui taikomi HOG filtrai

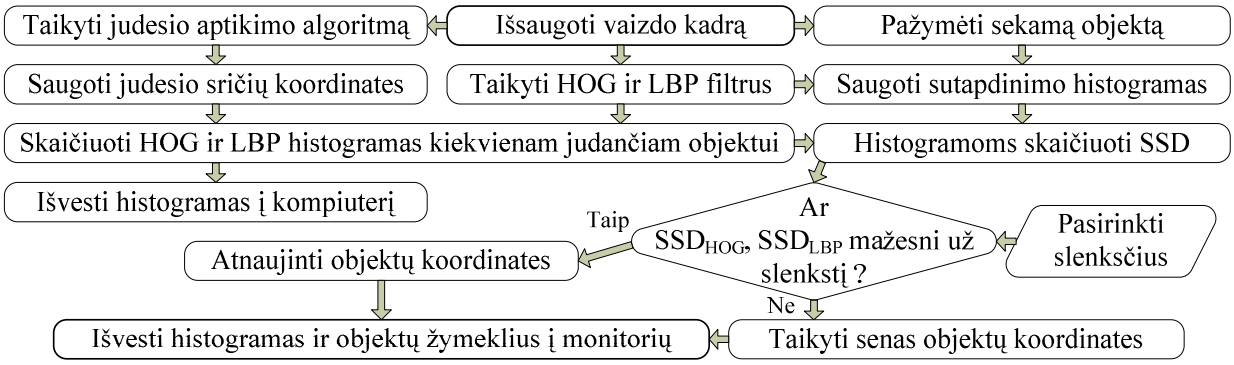
LBP histogramai kurti taikomi 3.22 paveiksle pateikti LBP filtrai. Pirmoje skalėje galima kurti 9 filtrus (vadinamus tolyginiais lokaliaisiais dvejetainiais modeliais) su ne daugiau dviejų perėjimų tarp baltos ir pilkos spalvos kaimyninių taškų. Antroje skalėje galima taikyti 5×5 dydžio 17 filtrų, iš kurių vienas su trimis kaimynų skaisčio vertėtis didesnėmis už centrinį tašką pateiktas dešinėje 3.17 paveikslo pusėje. Antroje eilutėje (3.22 pav.) pateikti netolyginiai modeliai su daugiau nei dvejais lokaliais gradiento perėjimais tarp kaimynų. Dėl histogramos sudėtingėjimo netolyginiai modeliai netaikomi deskriptorių sudaryme, nes pagal juos sunku vertinti tekstūros atsparumą pasukimui [45].



3.22 pav. Vaizdo kodavimui taikomi LBP filtrai

Bandymų metu nustatyta, kad 3.22 paveiksle pateikto aštunto pirmoje bei šešiolikto antroje skalėje LBP filtrų atsakai yra labai jautrūs nežymiems objekto tekstūros posūkiams. Tuos filtrus atitinkantys histogramos stulpeliai kiekybiškai įvertina lokalius minimumus arba vienodo skaisčio 3×3 arba 5×5 dydžio taškų sritis, kurių tekstūrose yra daugiausia. Todėl 8 bei 16 stulpelių LBP histogramoje siūloma neįtrauki į SSD skaičiavimus dėl didelių amplitudžių pokyčių, kurie didina deskriptorių sutapdinimo klaidą, todėl sekamo objekto koordinatės gali būti neatnaujinamos.

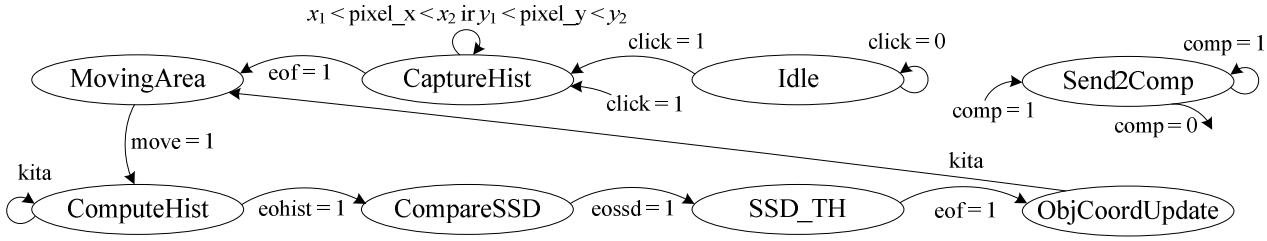
Objektų sekimo taikant tekstūros deskriptorius HOG ir LBP algoritmas pateiktas 3.23 paveiksle. Išsaugotam vaizdui taikomas judesio aptikimo algoritmas, kuris kiekviename kadre aptinka visus judančius objektus ir saugo kiekvieno koordinates. HOG ir LBP filtrai vaizdui taikomi tose srityse, kur aptikti judantys objektai. Paskaičiuoti deskriptoriai sutapdinami su pele pažymėtos srities monitoriuje histograma. HOG ir LBP histogramoms ieškoma mažiausia stulpelių skirtumų kvadratų suma. Jeigu ji mažesnė už slenkstinę vertę, tai atnaujinamos sekamo objekto koordinatės. Priešingu atveju taikomos senos koordinatės ir laikoma, kad objektas nejuda. Iš algoritmo seka, kad svarbu tiksliai aptikti judančių objektų ribas ir perskaičiuoti centrų koordinates, nes nuo jų priklauso deskriptorių skaičiavimo srities tikslumas.



3.23 pav. Objektų sekimo taikant tekstūros deskriptorius HOG ir LBP algoritmas

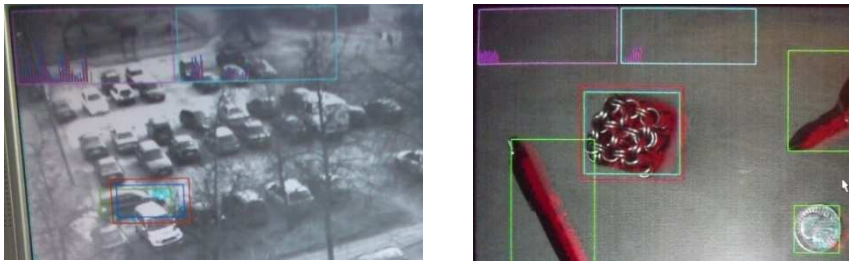
Objekto sekimo pagal tekstūrą algoritmo baigtinių būvių diagrama pateikta 3.24 paveiksle. Būvių automatą sudaro 8 būsenos:

- *Idle* – pradinė būsena, kurioje automatas būna tą laiką, kai pele nepažymėtas joks objektas. Tuo metu lygiagrečiai veikia judesio aptikimo algoritmas;
- *CaptureHist* – stačiakampio formos srities ribojamos taškais (x_1, y_1) ir (x_2, y_2) tekstūros kodavimo būsena. Į šią būsena pereinama iš bet kurios kitos, kai aptinkamas kairiojo pelės mygtuko paspaudimas;
- *MovingArea* – judančių objektų koordinačių išsaugojimo būsena. Automatas šioje būsenoje būna tik vieno taktinio signalo metu tam, kad iš lygiagrečiai veikiančio judesio aptikimo algoritmo paimtų sričių koordinates histogramoms skaičiuoti;
- *ComputeHist* – HOG ir LBP histogramų sudarymo būsena, kurioje automatas būna kadro į monitorių išvedimo metu;
- *CompareSSD* – skirtumų kvadratų sumų tarp pažymėto ir aprinkamų sričių histogramų palyginimo būsena;
- *SSD_TH* – mažiausios skirtumų kvadratų sumų palyginimo su slenkstinę vertę būsena;
- *ObjCoordUpdate* – sekamo objekto koordinačių atnaujinimo kadro pabaigoje būsena;
- *Send2Comp* – išsaugotų histogramų siuntimo į kompiuterį būsena. Į šią būseną peršokama ir joje būnama kol parametras *comp* = 1. Šioje būsenoje esančios operacijos vykdomos lygiagrečiai su kitų būsenų operacijomis.



3.24 pav. Objekto sekimo pagal tekstūrą algoritmo baigtinių būvių mašinos veikimo iliustracija

Objektų sekimo taikant tekstūros deskriptorius algoritmo testavimo rezultatai pateikti 3.25 paveiksle. Viršutiniame kairiajame monitoriaus kampe nubrėžtuose stačiakampiuose atvaizduojamos mėlyna spalva sutapdinimo ir raudona – sekamo objekto HOG ir LBP histogramos. Papildomas žalios spalvos stulpelis parodo skirtumų kvadratų sumos vertę. Objekto sekimo taikant tekstūros deskriptorius ir judesio aptikimą pavyzdys pareiktas 3.25 paveiksle *a*. Kartą pažymėto nejudančio automobilio histogramos sutapdinamos su vaizde judančių objektų histogramomis ir mažiausią SSD vertę turintis objektas apibrėžiamas mėlynu HOG arba raudonu LBP stačiakampiu žymekliu. Pastebėta, kad stabiliausiai sekami tie automobiliai, kurie pagal mastelį geriausiai atitinka pažymėtą pavyzdį.



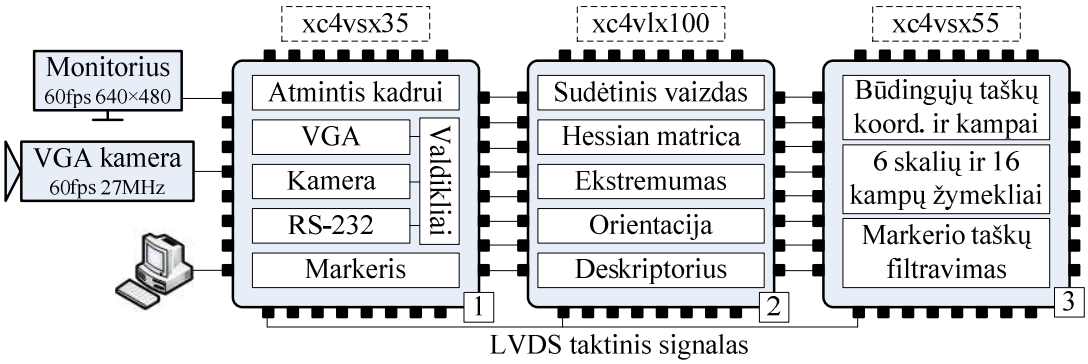
a) *b*)

3.25 pav. Objekto sekimas taikant tekstūros deskriptorius ir judesio aptikimą (*a*), objektų sekimas taikant tekstūros deskriptorius ir fono vaizde pašalinimą (*b*)

Objekto sekimo taikant tekstūros deskriptorius ir fono vaizde pašalinimą testavimo rezultatai pateikti 3.25 paveiksle *b*. Histogramoms kurti taikomi tik pirmos skalės tekstūrų filtrai, todėl sutapdinimas vykdomas tik per aštuonis stulpelius. Matoma, kad fone pasirodę visi nauji objektai pažymimi žalios spalvos stačiakampiais. Pažymėtasis metalinis kubas sekamas stabiliai nepaisant objekto pasisukimų ir apšvietimo pasikeitimų.

3.5. Sekimo pagal būdinguosius taškus algoritmų įgyvendinimas

Toliau pateikiamas objektų sekimo vaizde algoritmo įgyvendinimas taikant būdingųjų taškų išskyrimo metodą SURF. Algoritmo įgyvendinimui nepakako vienos loginės matricos atminties resursų, todėl programa suskirstyta į tris dalis ir įgyvendinta trijose skirtingos atminties LPLM *Virtex*-4 įrenginiuose. Trijų loginių matricų sujungimo schema pateikta 3.26 paveiksle. Valdančiuoju įrenginiu laikoma ML402 plokštėje esanti loginė matrica xc4vsx35. Ši plokštė taikant Mictor jungtį sujungta su kita DN8000K10PSX, kurioje yra du pavaldūs įrenginiai: xc4vlx100 ir xc4vsx55 matricos. Pirmojoje LPLM įgyvendinti monitoriaus, kameros ir nuosekliosios RS-232 sąsajos valdikliai. Joje taip pat įgyvendinti linijų braižymo Bresenham ir erdvinio kubo kūrimo algoritmai. Pirmosios LPLM blokinėje RAM atmintyje saugomas vienas vaizdo kadras, kurio keturi

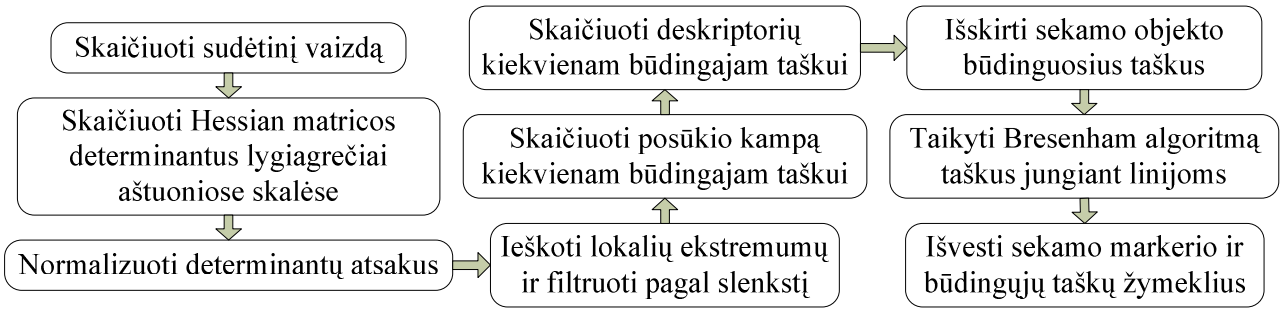


3.26 pav. Algoritmo funkcijų paskirstymo tarp trijų loginių matricų iliustracija

aukščiausi taškų skaisčio bitai sinchroniškai perduodami antrajai matricai. Antrojoje LPLM įgyvendintas požymių vaizde išskyrimo ir kodavimo SURF algoritmas. Trečiojoje LPLM saugomi

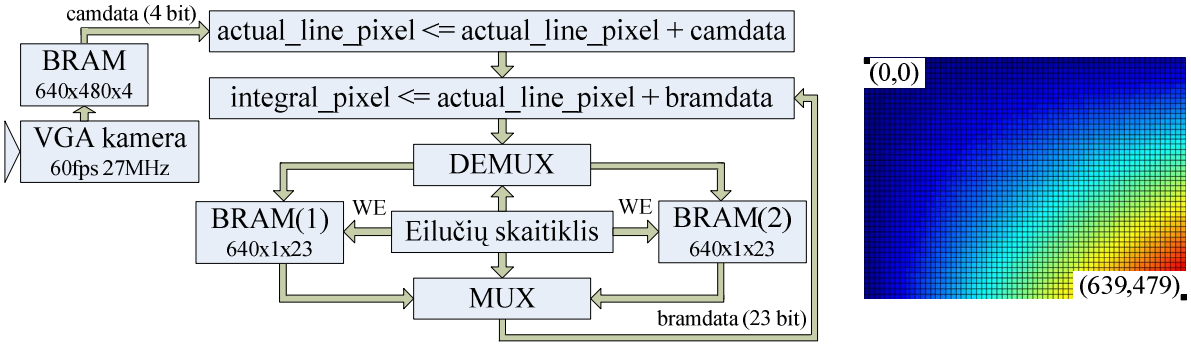
būdingųjų taškų koordinatės ir kampai, kurie susieti su šešių skalių ir šešiolikos kampų žymekliais. Trečiojoje LPLM taip pat įgyvendintas sekamo markerio taškų išskyrimas. Būdingųjų taškų žymekliai ir markerio taškai siunčiami atgal valdančiajam įrenginiui į monitorių išvesti. Antroji ir trečioji LPLM taktuojamos žemos įtampos skirtuminiu triukšmui atspariu signalu LVDS (angl. *Low Voltage Differential Signal*) taikant koaksialinius laidus. Paskirstyto algoritmo veikimas sinchronizuojamas vienalaikiu skaitikliu, esančiu trijose LPLM, numetimu. Vaizdo kadras su pažymėtais požymiais ir sekamu objektu pagal poreikį siunčiamas kompiuteriui.

Apibendrintas objekto sekimo pagal būdinguosius taškus algoritmas pateiktas 3.27 paveiksle. Jame trumpai pateikiami pagrindiniai skaičiavimo žingsniai, kurie vėliau detaliai išaiškinami. Pirmu žingsniu skaičiuojamas sudėtinis vaizdas. Taikant sudėtinį vaizdą skaičiuojami Hessian matricos determinantai lygiagrečiai aštuoniose skalėse. Kuo didesnė skalė, tuo didesnė determinanto vertė, todėl determinantų atsakus būtina normalizuojami pagal skalę. Tarp trijų gretimų skalių ieškoma lokalių ekstremumų – tai determinanto vertės, kurios didesnės už kaimyninius taškus. Aptikti ekstremumai papildomai filtruojami pagal slenkstį tam, kad būtų išskirti stipriausi požymiai. Kiekvienam būdingajam taškui skaičiuojami posūkio kampai ir deskriptoriai. Taikant Bresenham linijų braižymo algoritmą pagal išskirtus sekamo objekto būdinguosius taškus braižomas erdvinis kubas. Į monitorių išvedami sekamo markerio ir būdingųjų taškų žymekliai.



3.27 pav. Apibendrintas objekto sekimo pagal būdinguosius taškus algoritmas

Sudėtinio vaizdo (angl. *Integral Image*) skaičiavimo algoritmas pateiktas 3.28 paveiksle *a*. Vaizdui gauti taikoma VGA raiškos 60 kadrų per sekundę dažnio vaizdo kamera. Sudėtiniam 23 bitų gylio vaizdui sudaryti kiekvieno 25 MHz taktinio signalo metu taikomos dvi sumavimo operacijos. Pirmosios sumos operacijos metu sudedami vienoje eilutėje esančių taškų vertės, antrosios metu – viename stulpelyje esančių taškų vertės. Pagal vaizdo eilučių skaitiklio vertę valdomas sudėtinio vaizdo taško rašymo ir skaitymo iš BRAM(1) ir BRAM(2) atminčių eiliškumas. Kol pirmoje atmintyje saugoma viena paskutinė sudėtinio vaizdo eilutė, į antrąją įrašoma nauja. Vienodos spalvos kadrams būdingas sudėtinio vaizdo pavyzdys pateiktas 3.28 paveiksle *b*. Mažiausia sudėtinio vaizdo taško vertė visada būna viršutiniame kairiajame, o didžiausia – apatiniame dešiniajame kampuose.



a) *b*)

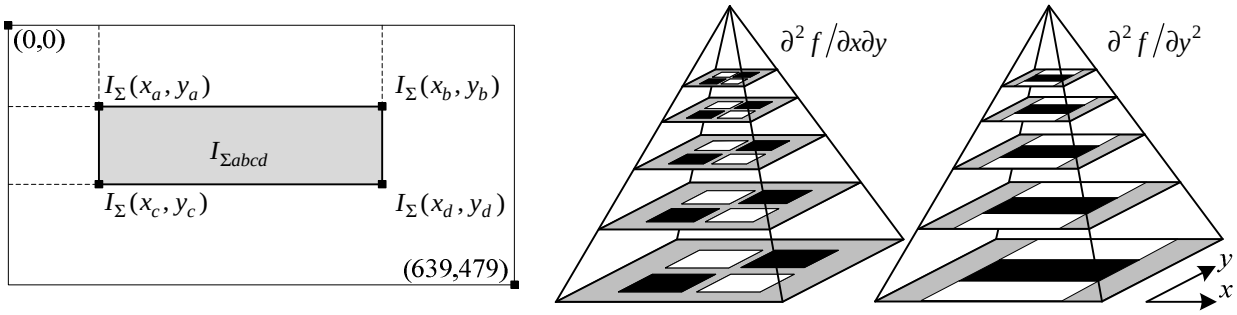
3.28 pav. Sudėtinio vaizdo skaičiavimo algoritmas (*a*), sudėtinio vaizdo pavyzdys (*b*)

Žemiau pateikiamos sudėtinio vaizdo sudarymo ir taikymo lygtis:

$$I_{\Sigma}(x,y)=\sum_{x=1}^{X}\sum_{y=1}^{Y} I(x,y); \qquad I_{\Sigma abcd}=I_{\Sigma}(x_a,y_a)+I_{\Sigma}(x_d,y_d)-I_{\Sigma}(x_b,y_b)-I_{\Sigma}(x_c,y_c), \tag{3.2}$$

čia $I(x, y)$ – skaisčio vertė taške (x, y), $I_{\Sigma}(x, y)$ – sudėtinio vaizdo taško vertė, $I_{\Sigma abcd}$ – sudėtinio vaizdo taškų verčių suma ribojamame stačiakampio *abcd* plote.

Sudėtinio vaizdo taikymo pavyzdys pateiktas 3.29 paveiksle *a*. Viena sumos, dvi skirtumo ir keturios duomenų skaitymo operacijos atliekamos kiekvieno taktinio signalo metu. Sudėtinio vaizdo taikymo privalumas tas, kad vienodai greitai įvertinama taškų skaisčio suma bet kokio ploto stačiakampyje. Ši savybė spartina aštuonių skalių ir trijų krypčių dvimačių antros eilės Gauso filtrų taikymą, iš kurių keli pateikti 3.29 paveiksle *b*.

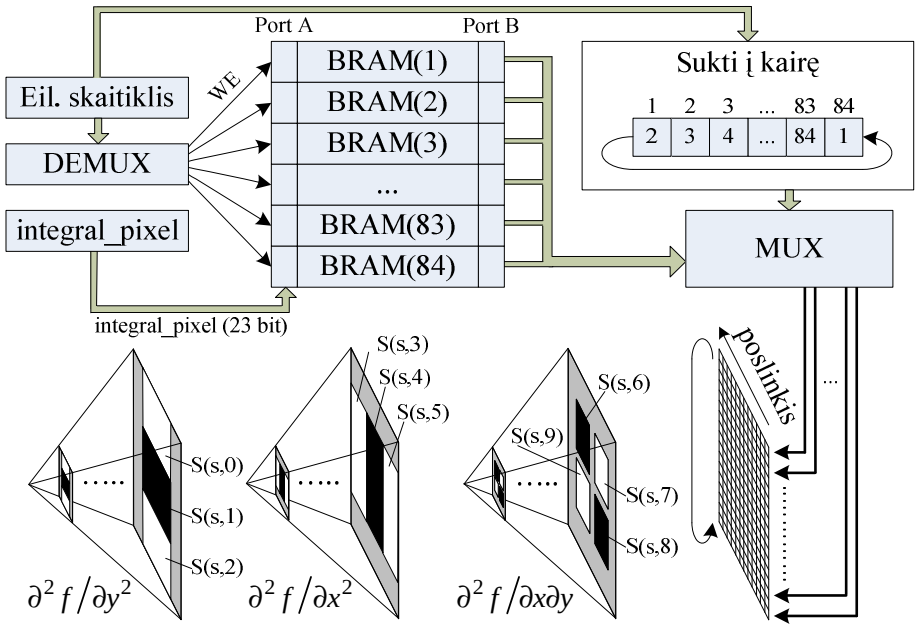


a) *b*)

3.29 pav. Sudėtinio vaizdo taikymas (*a*), kelių skalių antros eilės dvimačiai Gauso filtrai (*b*)

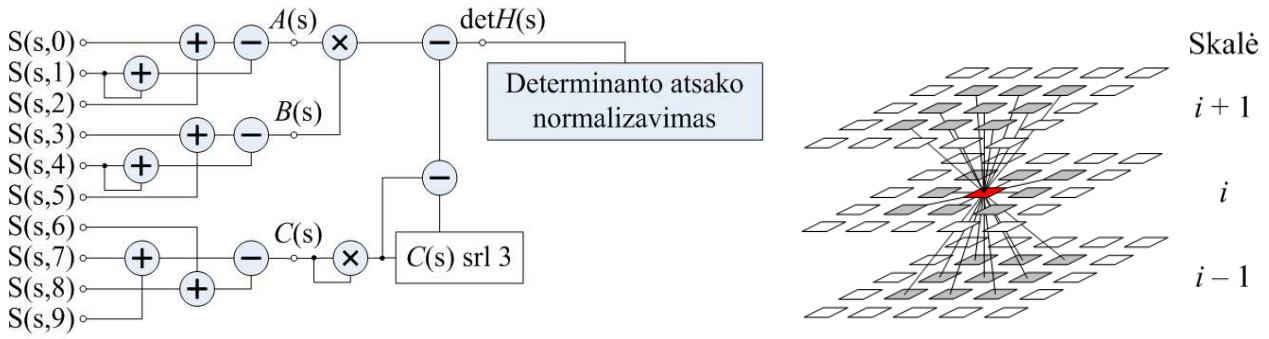
Slenkančio lango įgyvendinimo LPLM įrenginyje schema pateikta 3.30 paveiksle. Slenkančiam 84×84 dydžio langui sukurti taikomos 84 BRAM eilutės sudėtiniam vaizdui saugoti. Šis atminties eilučių skaičius pasirinktas dėl to, kad supaprastintam deskriptoriui skaičiuoti septintoje skalėje būtina turėti 10σ sudėtinio vaizdo eilučių į deskriptorių sudėtį įeinantiems taškams įvertinti skaičiuojant dvimates Haar vilneles. Haar vilnelių dydis lygus 4σ ir pusė šio filtro pločio turi dengti visą slenkančio lango perimetrą. Skaičius σ žymi diskretų skalės žingsnį, kuris septintai skalei lygus σ = 6, todėl 14σ = 84 taškai.

Paskaičiuotasis sudėtinio vaizdo taškas iš karto įrašomas į vieną iš BRAM(1, ..., 84) atminčių. Įrašymo eiliškumas sinchronizuotas su vaizdo eilučių skaitiklių. Skaitikliui nuosekliai didėjant nuosekliai keičiamas BRAM atminties indeksas. Indeksui pasiekus 84 vertę, sekančios sudėtinio vaizdo eilutės taškai įrašomi į BRAM(1) atmintį. BRAM atmintys kas 84 eilutės periodiškai perrašomos, todėl saugomi tik realiu laiku apdoroti būtini duomenys taikant ribotą pastovų atminties kiekį. Per antrąjį BRAM atminties prievadą B sinchroniškai kiekvieno taktinio impulso metu 23 bitų gylio duomenys nuskaitomi iš visų 84 atminties eilučių ir siunčiami į multiplekserių bloką. Slenkančio lango veikimui realiu laiku užtikrinti būtina ne tik duomenis jame stumti į kairę, bet ir eilučių skaitikliui padidėjus vienetui visuose stulpeliuose esančius duomenis per vieną elementą kelti į viršų. Šiam uždaviniui išspręsti taikomas besisukantis į kairę eilučių indeksų vektorius. Sunumeruoti indeksai vektoriuje nurodo, iš kurios BRAM atminties į kurią slenkančio lango eilutę reikia įrašyti duomenis.



3.30 pav. Slenkančio lango kūrimas Hessian determinantų, būdingųjų taškų posūkio kampų ir deskriptorių lygiagrečiam skaičiavimui

Slenkančio lango duomenimis dalijamasi su Hessian matricos determinanto skaičiavimo, būdingųjų taškų posūkio kampų ir deskriptorių skaičiavimo blokais. Dėl lygiagretaus slenkančio lango elementų adresavimo galimybės užtikrinamas vienalaikis kelių SURF algoritmo dalių (determinantų, posūkio kampų, deskriptorių skaičiavimas) veikimas. Žymenis S(s, x) (čia x = 0, …, 9) nurodo sumą vaizdo taškų parenkančių į stačiakampių ar kvadratų ribojamą plotą (3.30 pav.). Kiekvieno taktinio signalo metu 80 sumos verčių (10 kiekvienai skalei) perduodama į Hessian matricos determinanto skaičiavimo bloką, kuris pateiktas 3.31 paveiksle *a*. Bloke įgyvendintos 6×8 sumos, 5×8 skirtumo ir 2×8 daugybos operacijų.



a) *b*)

3.31 pav. Hessian matricos determinanto skaičiavimo įgyvendinimas LPLM (*a*), lokalaus ekstremumo paieška (*b*)

Hessian determinantas skaičiuojamas pagal lygtis:

$$H(x,y)=\det\begin{bmatrix}A & C\\ C & B\end{bmatrix}=\det\begin{bmatrix}\partial^2 f/\partial x^2 & \partial^2 f/\partial x\partial y\\ \partial^2 f/\partial x\partial y & \partial^2 f/\partial y^2\end{bmatrix};\qquad H(x,y)=AB-w^2C^2, \tag{3.3}$$

čia w^2 – svorio koeficientas, kuris lygus 0,83 [52]. Jis taikomas kompensuoti tolydžios funkcijos Gauso filtrų aproksimavimą stačiakampių formos filtrais. Aparatūroje žymiai greičiau vykdoma ne sandaugų iš slankaus kablelio koeficientų, bet aritmetinės ir postūmio operacijos. Todėl algoritme įvestas pakeitimas $w^2 = 1 – 1/8 = 0{,}875$. Tai reiškia, kad nuo skaičiaus C^2 atimama 1/8 jo dalis [54].

Hessian matricos determinantų antroje bei trečioje skalėse vaizdai pateikti 3.32 paveiksle. Žalia spalva pažymėti būdingieji taškai. Matyti, kad skalei didėjant vaizde aptinkama mažiau smulkių požymių (kampų ir taškų), tačiau randami nauji. Kuo didesnis žalios spalvos skaistis, tuo požymis yra stipresnis ir tuo labiau atsparus apšvietimo pasikeitimui.

a) *b*) *c*)

3.32 pav. Vaizdo kadras (*a*), Hessian determinantų antros (*b*) ir trečios (*c*) skalių vaizdai

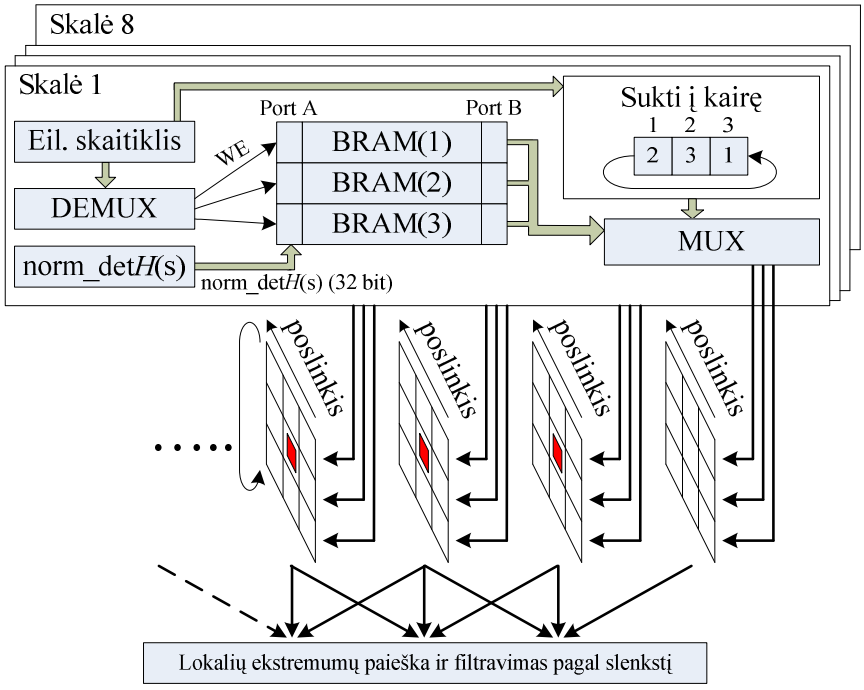
Aštuoni kiekvienos skalės determinanto rezultatai siunčiami į determinanto atsako normalizavimo bloką, kuriame kiekvienas aukštesnę už pirmąją skalę atitinkantis rezultatas dauginamas iš skalės koeficiento. Normavime taikomi koeficientai pateikti 3.1 lentelėje. Vietoje slankaus kablelio daugybos operacijų, kurių rezultatas vėlinamas per *N* taktinių impulsų (*N* priklauso nuo norimo tikslumo), taikomos sumos į dešinę pastumtų per suderintą bitų skaičių

operacijos. Lentelėje pateikiamos 1 – 8 skalių determinantų atsakų sumuojamosios dalys ir, palyginimui, santykinė skalės koeficiento paklaida. Matoma, kad slankaus kablelio skaičiaus daugybos operacijos pakeitimas duomenų postūmiu ir sumavimu įgyvendintas pakankamai tiksliai, nes santykinė skalės koeficiento paklaida nesiekia nei pusės procento.

3.1 lentelė. Originalūs ir taikomi skalės koeficientai bei jų santykinė paklaida

Skalės nr.	Skalės koeficientas	Sumuojamos dalys	Taikomas skalės koeficientas	Skalės koeficiento santykinė paklaida
1	1	2^{0}	1	0,0
2	0,145815	$2^{-3} + 2^{-6} + 2^{-8} + 2^{-10} + 2^{-12} + 2^{-14}$	0,145781	0,0002
3	0,037481	$2^{-5} + 2^{-8} + 2^{-9} + 2^{-13}$	0,037475	0,0001
4	0,011917	$2^{-7} + 2^{-8} + 2^{-13} + 2^{-14} + 2^{-16}$	0,011917	0,0
5	0,004231	$2^{-8} + 2^{-12} + 2^{-14} + 2^{-16}$	0,004234	0,0007
6	0,001607	$2^{-10} + 2^{-11} + 2^{-13} + 2^{-16} + 2^{-17}$	0,001609	0,0012
7	0,000638	$2^{-11} + 2^{-13} + 2^{-16} + 2^{-17} + 2^{-18}$	0,000637	0,0016
8	0,000262	$2^{-12} + 2^{-16} + 2^{-18}$	0,000263	0,0038

Normalizuotieji Hessian determinanto rezultatai siunčiami į būdingųjų taškų išskyrimo bloką (3.33 pav.). Kiekvienai iš aštuonių skalių kuriamas 3×3 dydžio slenkantis langas, kuris veikia identiškai 3.30 paveiksle patektai schemai. Slenkančių langų duomenys perduodami į lokalių ekstremumų paieškos ir filtravimo pagal slenkstį bloką. Lokalusis ekstremumas nustatomas lyginant centrinio taško vertę (raudona spalva) su aštuoniais toje pačioje skalėje ir aukštesnėje bei žemesnėje skalėse esančiais 2×9 kaimyniniais taškais (3.31 pav., *b*). Jeigu taško vertė didesnė už jo kaimynų vertes, laikoma, kad rastas būdingasis taškas.



3.33 pav. Būdingųjų taškų išskyrimas taikant 3×3×3 dydžio slenkančius langus

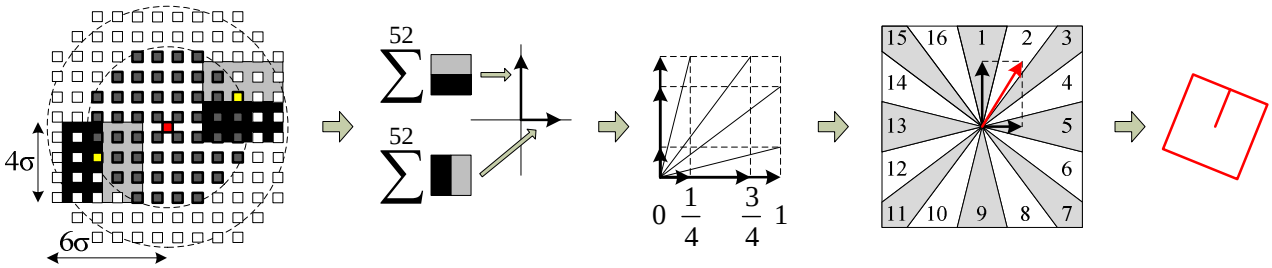
Norint vaizde išrinkti tik stipriausius požymius būtina būdinguosius taškus papildomai filtruoti pagal slenkstį. Statinį slenkstį patogu taikyti, kai žinoma, kad apšvietimas vaizde nekinta. Apšvietimui mažėjant kartu mažėja Hessian determinanto atsako skaitinės vertės, todėl dauguma jų neviršys slenksčio ir nebus aptikti požymiai. Slenksčio vertė sekančiam kadrui gali būti nustatoma pagal taškų skaisčio verčių vidurkį [70]. Tačiau požymiai vis tiek gali būti neaptikti dėl didelio kiekio baltos spalvos fone. Šiam uždaviniui išspręsti siūloma taikyti adaptyvų slenkstį, kuris priklausytų nuo vaizde aptiktų požymių ir slenksčio vertės praeitame kadre. Slenksčio vertė sekančiam kardui išreiškiama lygtimi:

$$TH_{\text{next}} = TH_{\text{prev}} + (N_{\text{prev}} - N_{\text{d}}) \cdot v_{\text{th}}, \tag{3.4}$$

čia TH_{next} – slenksčio vertė sekančiame kadre, TH_{prev} – esamo kadro slenksčio vertė, N_{prev} – būdingųjų taškų skaičius esamame kadre, N_{d} – ribinis būdingųjų taškų skaičius, v_{th} – adaptyvaus slenksčio prisitaikymo greitis.

Adaptyvusis slenkstis turi nustatytą apatinę ribą tam, kad požymiai, kurių determinanto vertės mažos (dažniausiai taškai išsidėstę išilgai įstrižos linijos ir požymį dubliuojantys atsakai), būtų atmetami. Dėl ribotos LPLM atminties algoritme nustatyta, kad kiekvienoje skalėje gali būti aptikta iki 100 būdingųjų taškų. Jeigu požymių vaizde rasta mažiau negu 80 % nuo maksimalios vertės, tai slenksčio vertė mažinama iki minimalios. Priešingu atveju, slenkstis pakeliamas. Maksimalaus būdingųjų taškų kadre rezervas paliekamas 20 % lygyje dėl galimo požymių skaičiaus šuolio per vieno kadro trukmę. Iš algoritmo veikimo pastebima, kad esant dideliam požymių vaizde skaičiui bus stengiamasi būdingųjų taškų skaičių išlaikyti kuo artimesnį 80-čiai, silpnesnių požymių eliminavimo dėka.

Aptiktų požymių koordinatės yra saugomos ir siunčiamos būdingųjų taškų orientacijos skaičiavimo blokui. Orientacijos nustatymo principas pateiktas 3.34 paveiksle. Aplink būdingąjį tašką atstumu 4σ apskaičiuotos 52 Haar dvimatės vilnelės *x* ir *y* kryptimis sumuojamos sudarant du

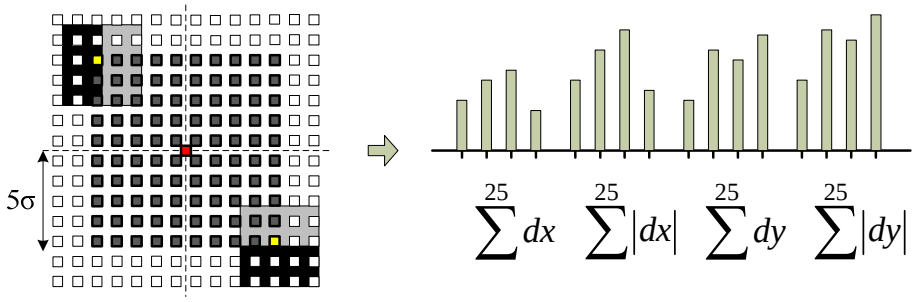


3.34 pav. Būdingųjų taškų orientacijos skaičiavimo principas

vienas kitam stačius vektorius. Taikant bitų postūmio ir sumos operacijas vertinami šių vektorių tarpusavio ilgiai. Vektorių palyginimas įgyvendintas tikrinant keliasdešimt sąlygų ir nustatant, į kurią iš sričių tinklelyje patenka suminis vektorius. Požymiui gali būti priskirta viena iš šešiolikos

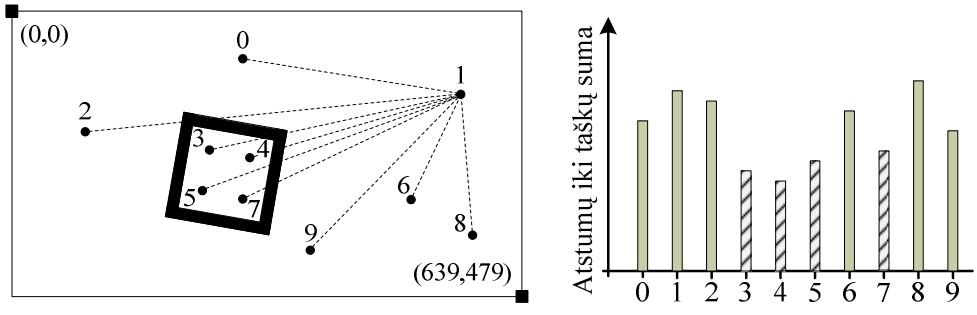
kampų orientacija. Būdingasis taškas pažymimas kvadrato formos žymekliu, kurių kiekvienai iš šešių skalių yra po šešiolika.

Būdingiesiems taškams koduoti yra taikomi deskriptoriai, kurių skaičiavimo principas pateiktas 3.35 paveiksle. Originalus SURF metodo deskriptorius skaičiuojamas išilgai orientacinio kampo 20σ×20σ pločio srityje centruotoje pagal požymio vietą. Deskriptorių sudaro 64 stulpelių histograma [52], kuriai realiu laiku kurti reikėtų taikyti 144×144 dydžio slenkantį sudėtinio vaizdo langą. Šio lango dydį lemia septintoje skalėje taikomųjų sudėtinio vaizdo eilučių kiekis deskriptoriui kurti skaičiuojant dvimates 4σ pločio Haar vilneles. Įgyvendinant deskriptoriaus skaičiavimo algoritmą nustatyta, kad viršijama loginės matricos atmintis, todėl požymiams koduoti taikomas supaprastintas 16 stulpelių deskriptorius, kuriam kurti pakanka 10σ ir papildomai 4σ pločio filtro perimetrui padengti. Dėl šios priežasties taikomas 14σ = 84 taškų pločio slenkantis sudėtinio vaizdo langas. Supaprastintam deskriptoriui kurti 4×25 taškuose skaičiuojamos Haar vilnelės (lokalūs gradientai *x* ir *y* kryptimis). Deskriptoriaus histograma sudaroma kiekvienai iš keturių 5σ×5σ sričių skaičiuojant gradientų pagal *x*, *y* bei jų modulių sumas.



3.35 pav. Būdingųjų taškų deskriptoriaus skaičiavimo principas

SURF algoritmas taikomas būdingųjų taškų dviejuose vaizduose paieškai ir sutapdinimui. Sutampantys požymiai vertinami kiekybiškai ir objekto žymeklis braižomas toje vietoje, kur aptinkama daugiausia sutampančių deskriptorių. LPLM įrenginyje įgyvendintas SURF algoritmas taikomas markeriui sekti pagal jame esančius keturių taškų požymius (3.36 pav., *a*). Pastebėta, kad taškai yra stipresni požymiai už kampus, todėl jie taikomi sekamam markeriui kurti. Vaizdo fone gali būti aptikti identiški markeriui būdingieji taškai. Todėl jų atmetimui kiekvienam požymiui kadre

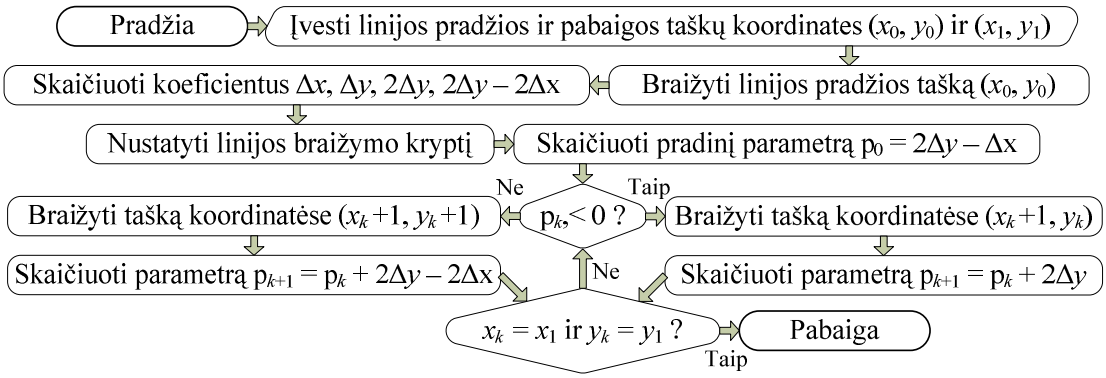


a) *b*)

3.36 pav. Vaizde aptikti būdingieji taškai (*a*), jų atstumų iki kitų požymių sumos (*b*)

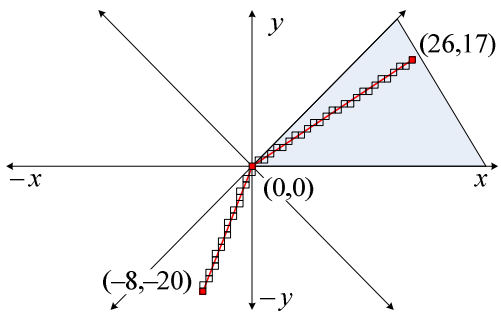
skaičiuojamos atstumų iki kitų požymių sumos, kurios 3.36 paveiksle *b* pateikiamos histogramos pavidalu. Požymiai, kuriuos atitinka keturios mažiausios vertės sumos, priskiriami markeriui ir pagal jų koordinates braižoma plokštuma arba erdvinis kubas.

Tiesių linijų braižymui taikomas Bresenham algoritmas, kuriam įgyvendinti naudojamos tik sumos, skirtumo skaičiavimo bei palyginimo operacijos [69] pateiktas 3.37 paveiksle. Algoritmo veikimas yra nuoseklus, nes sekančio linijos taško koordinatėms apskaičiuoti būtina žinoti esamo taško parametrą p_k. Į algoritmo bloką įvedamos linijos pradžios ir pabaigos taškų koordinatės. Braižomas linijos pradžios taškas x_0, y_0. Skaičiuojami keturi koeficientai Δx, Δy, $2\Delta y$, $2\Delta y - 2\Delta x$, kurių vertės taikomos sekančiuose algoritmo žingsniuose. Pagal pirmųjų dviejų koeficientų vertes nustatoma linijos braižymo kryptis. Bresenham algoritmas linijas gali braižyti tik pirmame koordinačių sistemos oktante. Todėl atsižvelgiant į linijos braižymo kryptį būtina sukeisti koordinačių ašis. Skaičiuojamas pradinis parametras p_0, pagal kurio vertę sprendžiama, kur braižyti sekantį tašką $(x_k + 1, y_k + 1)$ ar $(x_k + 1, y_k)$ koordinatėse ir koks bus sekantis p_{k+1} parametras. Linija braižoma tol, kol taško (x_k, y_k) koordinatės nesutampa su pabaigos tašku (x_1, y_1).



3.37 pav. Linijų braižymo Bresenham algoritmas

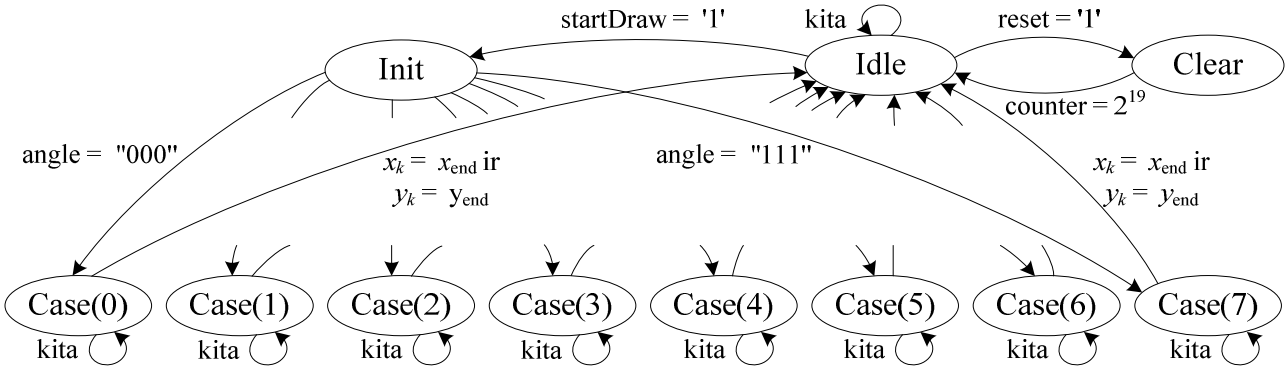
Linijos iš pavienių taškų braižymo pavyzdys pateiktas 3.38 paveiksle. Bresenham algoritmas kiekvienos iteracijos metu didina *x* koordinatės vertę vienetu ir, jeigu parametras p_k mažesnis už nulį, tai taškas braižomas iš dešinės pusės, priešingu atveju – papildomai pakeliamas per vieną tašką į viršų. Kai linijos braižymo kryptis patenka, pavyzdžiui, į šeštąjį oktantą, tai neigiamo *y* kryptis tampa teigiamą *x* kryptimi, o neigiama *x* – teigiama *y* kryptimi.



3.38 pav. Linijų iš pavienių taškų braižymas

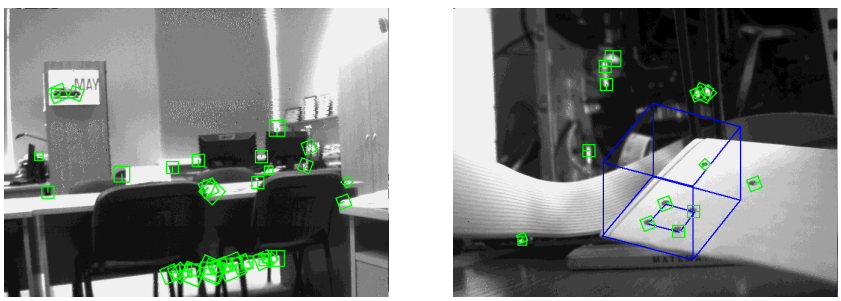
Bresenham linijų braižymo algoritmo baigtinių būvių diagrama pateikta 3.39 paveiksle. Būvių automatą sudaro 11 būsenų:

- *Idle* – pradinė būsena, kurioje automatas būna tą laiką, kai nereikia braižyti linijų;
- *Init* – linijos braižymo krypties nustatymo, koordinačių sistemos pakeitimo ir pradinių koeficientų skaičiavimo būsena, į kurią patenkama, kai gaunama komanda *startDraw* = '1' linijai braižyti;
- *Case*(0, ..., 7) – aštuonios linijų braižymo būsenos, iš kurių išeinama, kai nubraižomas paskutinis linijos taškas (x_{end}, y_{end});
- *Clear* – linijų iš atminties ištrynimo būsena, iš kurios išeinama, kai skaitiklis pasiekia reikšmę 2^{19} (10 bitų horizontaliai ir 9 vertikaliai monitoriaus skleistinei).



3.39 pav. Linijų braižymo baigtinių būvių mašinos veikimo iliustracija

Požymių vaizde išskyrimo rezultatas taikant SURF algoritmą pateiktas 3.40 paveiksle *a*. Paveiksle matyti, kad išskirti 2 – 6 skalių požymiai, kurie pažymėti stačiakampiais žymekliais. Žymekliai teisingai centruoti šviesių taškų tamsiame fone arba kampų atžvilgiu. Testavimui taikoma šešių žemiausių skalių SURF algoritmo versija be adaptyvaus slenksčio stipriems požymiams filtruoti.



a) *b*)

3.40 pav. Požymių išskyrimo rezultatas taikant SURF algoritmą (*a*), objekto sekimo pagal būdinguosius taškus rezultatas (*b*)

LPLM įgyvendintas SURF algoritmas taip pat testuotas vaizde išskiriant ir sekant keturių taškų markerį (3.40 pav., *b*). Juodi taškai baltame fone teisingai aptikti, pažymėti ir sujungti linija taikant Bresenham linijų braižymo algoritmą. Pagal aptiktus būdinguosius taškus papildoma plokštuma ir erdvinis kubas nubraižomi teisingai.

LPLM įgyvendintų objektų sekimo algoritmų resursų išnaudojimas pateiktas 3.2 lentelėje.

3.2 lentelė. LPLM resursų išnaudojimas objektų sekimo algoritmams įgyvendinti

Sekimo algoritmas	LPLM tipas	Loginės celės	LUT	BRAM	DSP	Taktinis dažnis (MHz)
Pagal spalvą	xc4vsx35	4807/15360	8518/30720	186/192	1/192	108
Pagal pavyzdį	xc4vsx35	14417/15360	23765/30720	188/192	6/192	25
Pagal judesio aptikimą ir fono pašalinimą	xc4vsx35	4834/15360	8619/30720	170/192	0/192	25
Pagal tekstūrą	xc4vsx35	15358/15360	27588/30720	170/192	192/192	25
Pagal požymių išskyrimą taikant SURF	xc4vsx35 xc4vlx100 xc4vsx55	9651/15360 48694/49152 24574/24576	18463/30720 93320/98304 36058/49152	132/192 110/240 91/320	2/192 18/96 0/512	25

Iš lentelės matyti, kad daugiausia resursų išnaudota įgyvendinant objektų sekimą taikant SURF algoritmą. Algoritmų veikimo taktiniai dažniai parinkti pagal į monitorių išvedamų taškų dažnį.

3.6. Trumpas skyriaus apibendrinimas

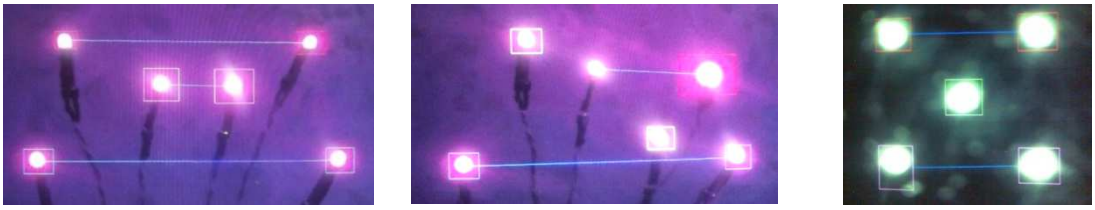
LPLM įrenginyje įgyvendinti ir eksperimentiškai ištirti objektų sekimo pagal spalvą, pavyzdį, judesį, fono pašalinimą, tekstūros deskriptorius (HOG ir LBP), požymių išskyrimą taikant SURF algoritmai. Algoritmų blokų taktiniai signalai sinchronizuoti su vaizdo kameros arba monitoriaus taktiniu dažniu, todėl algoritmai sėkmingai veiks taikant didesnės raiškos arba greitaveikos kameras. Visi algoritmai įgyvendinti VHDL aparatūros aprašymo kalba, todėl pasiektas jų veikimas realiu laiku. Greitaveika pasiekta dėl lygiagrečiai įgyvendintų procesų ir dėl vienalaikio dvimačių filtrų elementų adresavimo galimybės, bet ne dėl taktinio dažnio didinimo. Požymių išskyrimų vaizde grįstas objektų sekimo algoritmas SURF paskirstytas tarp trijų LPLM dėl atminties trūkumo.

4. OBJEKTŲ SEKIMO VAIZDE ALGORITMŲ TYRIMAS

Šiame skyriuje pateikiamas realiu laiku veikiančių objektų sekimo vaizde algoritmų, įgyvendintų LPLM, tyrimas. Sekimo pagal spalvą algoritmas tiriamas vertinant objekto atstumą iki vaizdo kameros pagal dviejų horizontaliai išsidėsčiusių vienodos spalvos šviesų tarpusavio padėtį. Tiriamas žmogaus veido ir kitų objektų sekimo stabilumas taikant pavyzdžio sutapdinimą su vaizdu. Atliekamas judesio aptikimo ir fono vaizde pašalinimo algoritmų tyrimai sekant automobilius ir kitus objektus. Tiriami objekto sekimo algoritmai taikant tekstūros deskriptorius siekiant nustatyti atsparumą posūkio kampo ir apšvietimo pokyčiams. Tiriamas būdingųjų taškų sekamame objekte išskyrimo algoritmas. Nustatomas objekto mastelio, posūkio ir transformacijos įtaka būdingųjų taškų aptikimo tikslumui.

4.1. Viena kitą atitinkančių vaizdo sričių paieškos algoritmų tyrimas

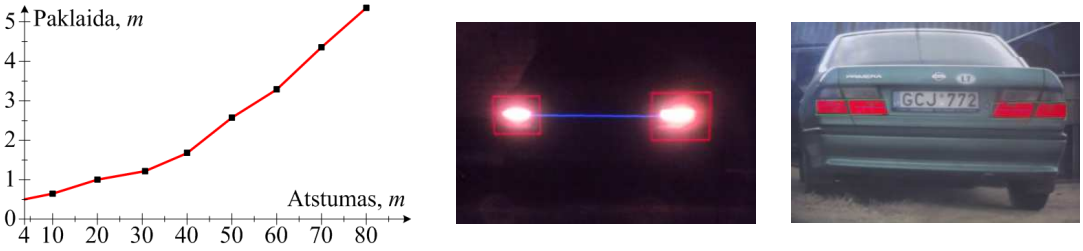
Objektų sekimo pagal spalvą vaizde rezultatai pateikti 4.1 paveiksle. Raudonos ir žalios spalvos šviesos teisingai aptinkamos ir pažymimos stačiakampiais žymekliais. Vienodame aukštyje esančios šviesų poros jungiamos mėlynos spalvos linijomis. Atstumai tarp suporuotų objektų ir jų atstumai iki kameros išvedami į LCD vaizduoklį.



a) *b*) *c*)

4.1 pav. Raudonos spalvos šviesų sekimas (*a*, *b*), žalios spalvos šviesų sekimas (*c*)

Objektų sekimo pagal spalvą algoritmas taikomas galiniams automobilio žibintams sekti (4.2 pav.). Atstumo iki automobilio matavimai atlikti tamsiu ir šviesiu paros metu. Nustatyta, kad paklaida didėja neproporcingai automobilio atstumui iki vaizdo kameros. Kuo didesnis atstumas tuo didesnė paklaida. Kadangi sekami šviesos šaltiniai aptinkami ne kaip pavieniai taškai, bet kaip taškų

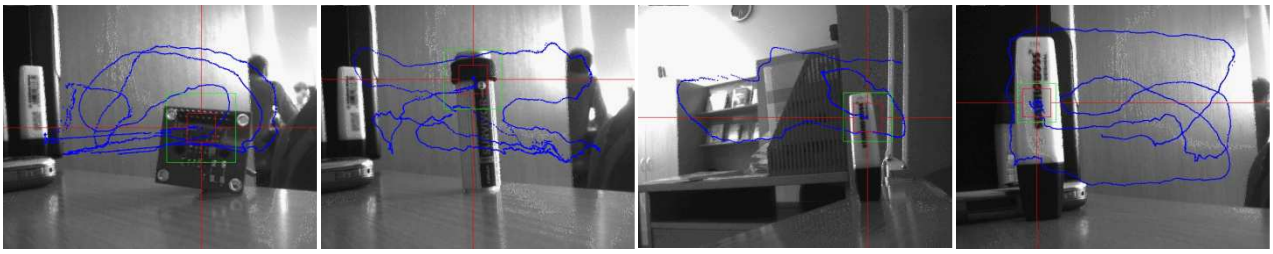


a) *b*) *c*)

4.2 pav. Absoliučios atstumo paklaidos priklausomybė nuo automobilio atstumo iki kameros (*a*), sekami automobilio galiniai žibintai (*b*, *c*)

grupės, pagal kurių koordinates perskaičiuojami šviesų centrų koordinatės, tai negarantuojama, kad nustatytos koordinatės tiksliai atitiks žibintų vietas. Paklaidą taip pat didina automobilio pasisukimas kameros atžvilgiu ir tuo labiau, kuo toliau jis yra.

Objektų sekimo taikant vaizdo pavyzdžio sutapdinimą su kadru rezultatai pateikti 4.3 paveiksle. Vaizde sekami objektai pažymėti raudonos spalvos žymekliais. Jų judėjimo trajektorija pažymėta mėlynos spalvos linija. Pavyzdžio (lokalaus sutapdinimo klaidos minimumo) paieškos sritis sekančiame kadre pažymėta žalios spalvos kvadratėliu. Algoritmas tiriamas sekant įvairius objektus, tolinant bei artinant juos nuo kameros, pasukant ir keičiant vaizdo foną bei paieškos sritį. Nustatyta, kad globalaus minimumo paiešką geriausia taikyti, kai vaizde nėra į sekamą objektą panašių sričių. Priešingu atveju žymeklis klaidingai peršoka į nesekamas vaizdo fono sritis, pagal kurių pavyzdį gali išsaugoti, jeigu tuo metu sekamo objekto sutapdinimo klaida padidėjo dėl jo pasisukimo ar persidengimo. Naujo pavyzdžio saugojimas reikalingas stabiliam objekto sekimui užtikrinti, kai keičiasi apšvietimas arba objektas pasisuka, artėja (tolsta) nuo vaizdo kameros.



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

4.3 pav. Objektų sekimo rezultatai taikant vaizdo pavyzdžio sutapdinimą su kadru, kai paieškos sritis ±40 taškų (*a*), ±30 taškų (*b*), ±20 taškų (*c*), ±10 taškų (*d*)

Nustatyta, kad kuo didesnė paieškos sritis taikoma, tuo greičiau sekamas objektas gali judėti ir tuo didesnė tikimybė, kad lokalus minimumas bus klaidingai rastas. Kuo mažesnė paieškos sritis, tuo stabiliau sekamas ir atnaujinamas pavyzdys, tačiau sekimo greitis mažėja proporcingai paieškos spinduliui. Tuo atveju, kai judantis sekamas objektas trumpam užstojamas ir pavyzdys bei jo koordinatės neatnaujinamos, didesnės paieškos srities taikymas padidina tikimybę, kad vaizde pasirodęs objektas bus sėkmingai aptiktas. Nustatyta, kad ±20 taškų paieškos sritis užtikrina pakankamą paieškos plotą bei didelį objekto judėjimo greitį (20 taškų × 60 s^{-1} = 1200 taškų per sekundę). Tai reiškia, kad objektą galima stabiliai sekti jam kiek daugiau nei pusę sekundės įveikiant išilgai vaizdą.

Tyrimo metu norima nustatyti, kokią slenkstinę sutapdinimo klaidos ribą err_{const} bei didžiausią leistiną klaidos pokytį Δ_{const} pasirinkti, kad sekamo objekto pavyzdys būtų atnaujinamas teisingai. Buvo sekami 4.3 paveiksle pateikti objektai papildomai taikant filtrus (4.1 lentelė) kraštinių pavyzdžio skaisčio taškų įtakai bendrai klaidai sumažinti. Nustatyta, kad sutapdinimo klaida didėja proporcingai skaičiui Σk_i dėl centrinių pavyzdžio taškų dauginimo iš filtro koeficientų. Todėl

kiekvienam filtrui eksperimentų metų turi būti parinkta slenkstinė klaidos riba err_{const}, kurią viršijus saugomas naujas pavyzdys ir teigiamas klaidos pokytis Δ_{const}, kurį viršijus objekto koordinatės neatnaujinamos. 4.1 lentelėje pateiktos err_{const} slenkstinės vertės prie kurių gaunami stabiliausi sekimo rezultatai taikant skirtingus filtrus. Santykis err_{const} / Σk_i parodo, kad nepriklausomai nuo filtro tipo sekamo objekto pavyzdį geriausia atnaujinti, kai sutapdinimo klaida yra lygi pasvertai sekamo pavyzdžio elementų sumai. Tyrimo metu nustatyta, kad didžiausias leistinas klaidos pokytis Δ_{const} turi būti apie 2,6 karto mažesnis už klaidos slenkstinę vertę err_{const}.

4.1 lentelė. Koeficientų k_i įtaka slenkstinei klaidos vertei err_{const} ir jos pokyčiui Δ_{const}

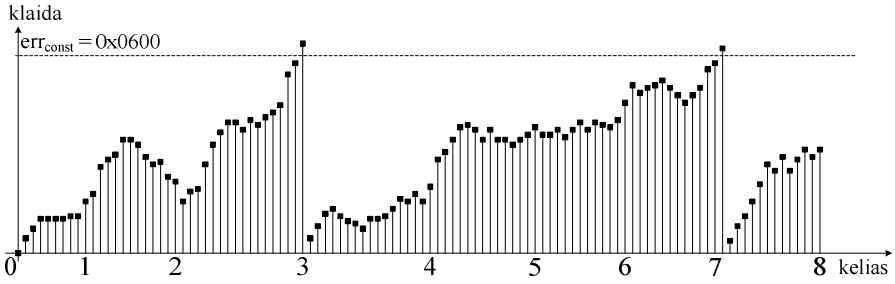
Filtras	1 1 1 1	1 1 4 4	1 4 8 8	1 2 4 8
Σk_i	49×16	80×16	160×16	96×16
err_{const}	0x0300	0x0500	0x0A00	0x0600
err_{const} / Σk_i	0,98	1	1	1
Δ_{const}	0x0120	0x0200	0x0400	0x0240
err_{const} / Δ_{const}	2,67	2,5	2,5	2,67

Objektų sekimui taikant įvairius filtrus nustatyta, kad esant vienodiems koeficientams (pirmasis filtras) objektas sekamas stabiliai, kai pavyzdys sudarytas tik iš objekto elementų arba dalinai iš fono, jeigu skaistis jame keičiasi tolygiai, kad vaizdų sutapdinimo klaida neviršytų Δ_{const}. Taikant 2–4 filtrus sumažinama kraštinių pavyzdžio taškų įtaka bendrai klaidai bei didinama centrinių taškų vertė. Šiuos filtrus patartina taikyti greitai judantiems objektams sudėtingame vaizdo fone sekti.

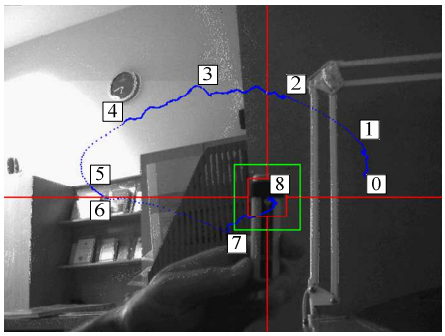
Nustatyta, kad kuo algoritme taikoma slenkstinė klaida err_{const} yra didesnė už 4.1 lentelėje nurodytą, tuo rečiau atnaujinamas pavyzdys. Kadangi atnaujinimas vyksta prie didelės klaidos, tai negarantuojama, kad saugomas naujas pavyzdys bus teisingas. Kuo mažesnis err_{const} pasirenkamas, tuo dažniau atnaujinamas pavyzdys. Dažnas pavyzdžio atnaujinimas veda prie žymeklio atsilikimo ir klaidingo pavyzdžio išsaugojimo, kai objektas juda greitai. Mažą err_{const} slenkstį patartina taikyti, kai objektas juda ar sukasi lėtai. Didžiausiu teigiamos klaidos pokyčiu Δ_{const} reguliuojamas sekamo pavyzdžio bei jo koordinačių atnaujinimo išjungimas, kai objektas persidengia su kitu arba dingsta iš kadro. Kuo Δ_{const} didesnis už 4.1 lentelėje nurodytą, tuo labiau klaidai leidžiama kisti, tuo greičiau sekamas objektas gali judėti. Per didelis klaidos gradientas lemia klaidingą pavyzdžio atnaujinimą, kai objektas staigiai užstojamas arba dingsta iš kadro. Kuo mažesnis Δ_{const}, tuo dažniau pametamas greitai judantis objektas, tuo dažniau sustabdomas sekimas ir pradedamas, kai objektas patenka į paieškos sritį.

Vaizdo pavyzdžio sutapdinimo su kadru klaidos kitimo grafikas pateiktas 4.4 paveiksle *a*. Pavyzdžio judėjimui sekti taikomas ketvirtas 4.1 lentelėje pateiktas filtras bei 40×40 taškų paieškos sritis centruota minimalios klaidos taško atžvilgiu (toliau žymima ±20 taškų paieškos sritis).

Sekamas objektas, jo žymeklis ir sunumeruotos objekto judėjimo kelio atkarpos patektos 4.4 paveiksle *b*. Tankiai išsidėsčiusių mėlynų taškų linija nurodo atkarpas, kuriuose objektas judėjo lėtai. Kuo trajektorijos taškai rečiau išsidėstę, tuo objektas toje atkarpoje judėjo greičiau. Ties numeriais 3 ir 7 objektas buvo pasukamas, dėl ko klaida viršijo slenkstinę vertę err_{const} = 0x0600 ir algoritmas atnaujino pavyzdį, todėl grafike pastebimi staigūs klaidos sumažėjimai.



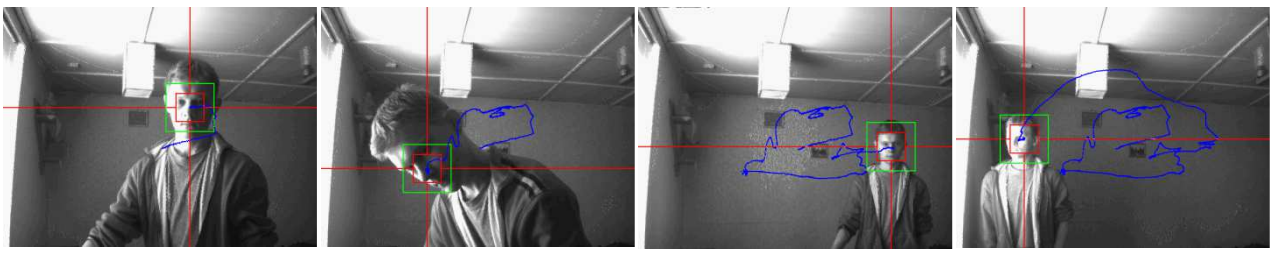
a)



b)

4.4 pav. Vaizdo pavyzdžio sutapdinimo su kadru klaidos kitimo grafikas (*a*), sekamo objekto trajektorija (*b*)

Pavyzdžio sutapdinimo su vaizdu algoritmas esant ±20 taškų paieškos sričiai ir filtro koeficientams k_{1234} = [1 2 4 8] taikomas žmogaus veidui vaizde sekti realiu laiku. Pele pažymėtas veido pavyzdys sėkmingai sekamas bei atnaujinamas nepriklausomai nuo apšvietimo pasikeitimų, veido pasisukimo, staigaus priartėjimo ar nutolimo. Veido judėjimo trajektorija pažymėta mėlyna kreive. Dingus veidui iš kadro arba jį staigiai užstojus viršijama leistina klaidos gradiento riba Δ_{const} ir algoritmas laukia, kol veidas vėl pasirodys paskutinėje žinomoje paieškos srityje. Nestabilumai sekime gali kilti dėl paieškos srityje atsirandančių panašių į pavyzdį objektų. Todėl pavyzdys gali būti

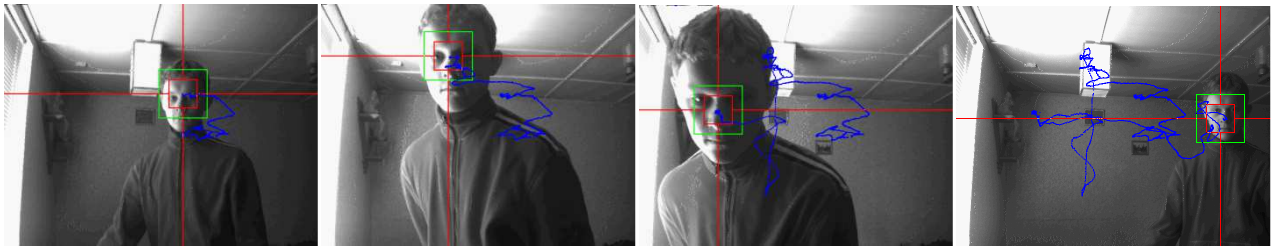


a) *b*) *c*) *d*)

4.5 pav. Veido sekimo rezultatai, kai pavyzdys ieškomas ±20 taškų nuo mažiausios klaidos taško praeitame kadre

per kelis taškus pastumiamas dėl klaidingo atnaujinimo. Todėl siūloma mažinti paieškos sritį, kai objektas nejuda ir didinti tik jam pajudėjus.

4.6 paveiksle pateikti veido sekimo rezultatai taikant nuo objekto greičio padidėjimo prasiplečiančią paieškos sritį. Algoritme nustatyta minimali ±5 taškų paieškos sritis, kai greitis lygus nuliui. Algoritmas tapo atsparesnis vaizdo fono pokyčiams, nes greičiui mažėjant į paieškos sritį patenka vis mažiau fono taškų. Vaizdo kadrai ir žymekliai saugoti, kai veidas nejudėjo, todėl paieškos sritis yra minimali.



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

4.6 pav. Veido sekimo rezultatai, kai pavyzdžio paieškos sritis priklauso nuo judėjimo greičio

Tyrimo metu nustatyta, kad tolstantis nuo kameros objektas turi užimti bent pusę sutapdinimo pavyzdžio srities, kad galėtų būti teisingai sekamas bei atnaujintas, kai artės prie kameros. Objektas artėti prie kameros gali tol, kol sutapdinimo pavyzdį įmanoma išskirti iš fono pagal taškų skaistį. Kuo pavyzdys labiau skiriasi nuo fono vaizdo, tuo stabiliau sekamas, tuo mažesnė tikimybė, kad bus klaidingai atnaujintas padidėjus minimalios lokalios klaidos paieškos plotui.

4.2. Sekimo pagal judesio aptikimą ir tekstūrą algoritmų tyrimas

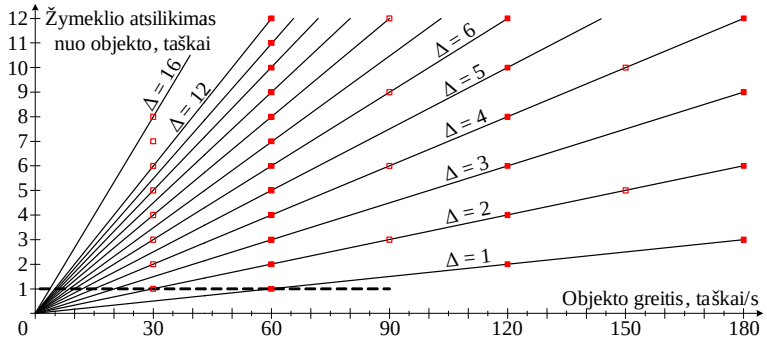
Objektų sekimo taikant judesio aptikimą, fono vaizde pašalinimą ir tekstūros kodavimo algoritmai tiriami šiame poskyryje. Norima nustatyti jautrio judesiui įtaką teisingam judančių objektų vaizde pažymėjimui bei sekimo pagal tekstūrą stabilumą ir atsparumą apšvietimo pokyčiams, objektų pasisukimui sekant įvairių tekstūrų objektus. Jautrį judesiui galima derinti dvejais būdais, keičiant skaisčio verčių skirtumo tarp kadrų slenkstį arba skaičiavimams taikyti kas *n*-tąjį kadrą.

4.7 paveiksle pateikti judančių automobilių ir žmonių gatvėje sekimo rezultatai. Tyrimo metu nustatyta, kad skaisčio verčių skirtumo slenkstis turi būti apie 24. Kadangi judesio aptikimui taikomi 4×4 = 16 taškų 4 aukščiausi skaisčio bitai (16 skaisčio lygių), tai reiškia, kad taškas laikomas judančiu, jeigu jo skaistis pasikeitė per 24/16 = 3/2 skaisčio lygio dalį. Parenkant mažesnį už 24 jautrio slenkstį didėja jautrumas nežymiems pokyčiams tarp kadrų (aptinkami medžių šakų svyravimai, apšvietimo pokyčiai). Parenkant didesnį jautrio slenkstį mažėja jautrumas pokyčiams kadre ir geriausiai sekami tik tie objektai, kurių taškų ir fono skaisčio skirtumas didžiausias (tamsūs objektai baltame fone arba šviesūs – juodame). Nustatytasis skaisčio slenkstis suteikia galimybę neregistruoti triukšmų sukeltų klaidingų judesių ir teisingai vaizde aptikti judančius kelių eismo dalyvius.

a) *b*) *c*)

4.7 pav. Judančių objektų vaizde sekimo rezultatai

Lėtai judantiems objektams aptikti ir sekti skaičiuojami skirtumai tarp kas dešimto kadro, todėl lėtai judantis automobilis ir einantis žmogus pažymimi teisingai (4.7 pav., *a*). Tačiau, kai automobilis važiuoja greitai, objekto žymeklis atsilieka per atstumą nuvažiuotą dešimties kadrų metu (šiuo atveju per 1/6 sekundės). Todėl norint sekti lėtai judančius objektus, patartina didinti kadrų apdorojimo žingsnį Δ ir mažinti, jeigu objektas juda greitai. Žymeklio atsilikimo nuo objekto priklausomybė nuo greičio Δ-ai esant nuo 1 iki 16 pateikta 4.8 paveiksle. Iš grafiko matyti, kad žymeklis nuo objekto atsiliks per vieną tašką, kai Δ = 1 ir objekto greitis bus 60 taškų per sekundę arba Δ = 2 ir objekto greitis – 30 taškų per sekundę ir t.t.

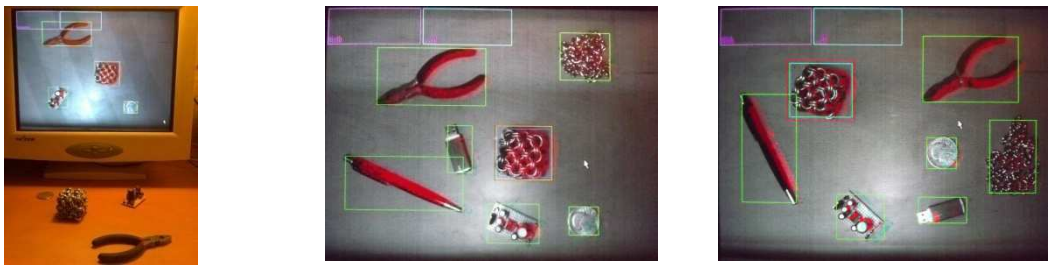


4.8 pav. Objekto judėjimo greičio įtaka žymeklio atsilikimui nuo objekto vaizde taikant 60 kadrų per sekundę dažnio vaizdo kamerą

Lygiagrečiai su judančių objektų vaizde sekimo algoritmu veikia objekto sekimo pagal tekstūrą algoritmas, kuris tiriamas sekant vaizdo sraute pele pažymėtą automobilį. Automobilis vaizde atpažįstamas ir pažymimas raudonos bei mėlynos spalvos žymekliais esant dalinai užstotam medžių ar kito automobilio (4.7 pav. *a*, *b*), kai histogramų sutapdinimo leidžiamos klaidos slenkstis SSD_{TH} yra aukštas (kiekvieno iš deskriptoriaus stulpelio nesutapimas siekia vidutiniškai 20 %). Tokiam klaidos slenksčiui esant žymekliai peršoka ant judančio automobilio, kurio sutapdinimo klaida mažiausia (4.7 pav. *c*). Mažinant SSD_{TH} didinama tikimybė, kad algoritmas seks tik pažymėtą objektą, tačiau kartu mažinamas atsparumas apšvietimo, fono pokyčiams bei automobilio pasisukimui. Kai deskriptorių stulpelių nesutapimas siekia vidutiniškai 5 %, automobilis sekamas tik tame kelio ruože, kuriame jo dydis sutampa su kartą pažymėto automobilio dydžiu. Sekimo

pagal tekstūrą algoritmo taikymas automobiliams sekti yra abejotinas dėl nestabilaus objekto sekimo visame kelio ruože, jeigu taikoma maža deskriptorių sutapdinimo klaida.

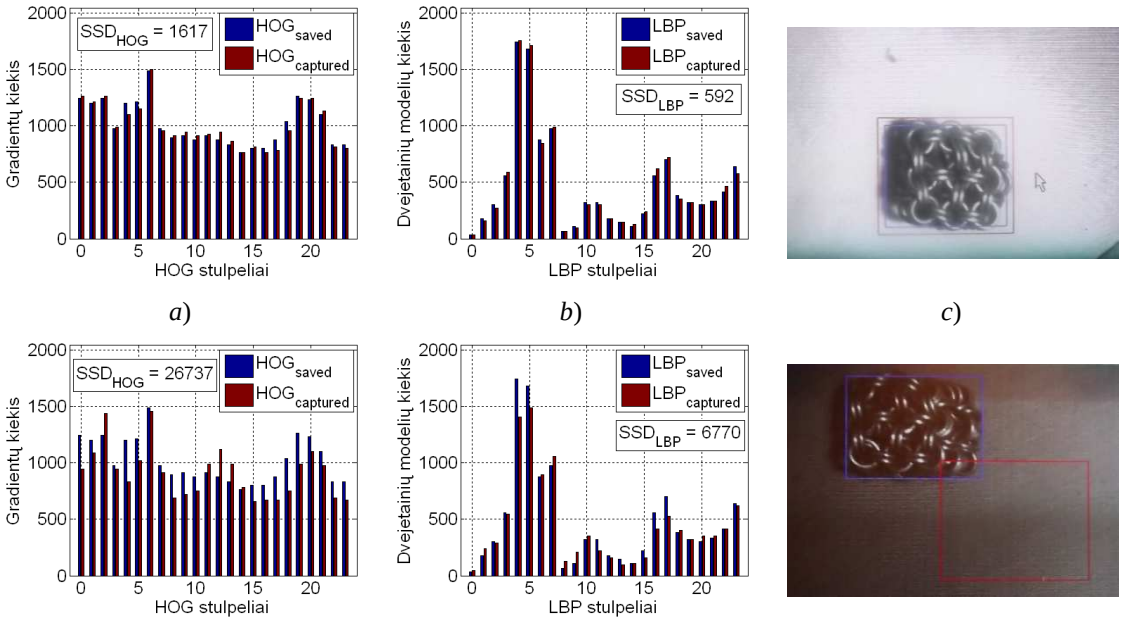
Objektų sekimo algoritmo taikant fono vaizde pašalinimą ir tekstūros deskriptorius tyrimo rezultatai pateikti 4.9 paveiksle. Skirtumai tarp fono vaizdo ir aptiktų objektų pažymimi raudonai, kai objekto skaisčio vertė mažesnė už fono, priešingu atveju – mėlynai. Nustatyta, kad vaizde esantys ir naujai įkeliami objektai aptinkami ir sekami teisingai. Fono vaizde pašalinimo algoritmo privalumas tas, kad skirtumų tarp išsaugoto ir naujų kadrų skaičiavimas nesudėtingai įgyvendinamas, tačiau taikomas tik kai vaizdo kamera yra fiksuota arba judanti naujai saugant foną. Įvairių objektų sekimui taikant fono vaizde pašalinimo algoritmą ir $HOG_{8,1}$ bei $LBP_{8,1}$ tekstūros deskriptorius nustatyta, kad stabiliausiai sekami objektai, kurių tekstūra sudaryta iš pasikartojančių elementų.



a) *b*) *c*)

4.9 pav. Objektų sekimas taikant fono vaizde pašalinimą ir tekstūros deskriptorius

Sekimo pagal tekstūrą algoritmo atsparumas apšvietimo pokyčiams tiriamas sekant 4.10 paveiksle *c* pateiktą objektą. Baltame fone esantis objektas pažymėtas pele, išsaugotos jo koduotos tekstūros $HOG_{saved} = HOG_{8,1} + HOG_{16,2}$ (4.10 pav., *a*) ir $LBP_{saved} = LBP_{8,1} + LBP_{16,2}$ (4.10 pav., *b*)



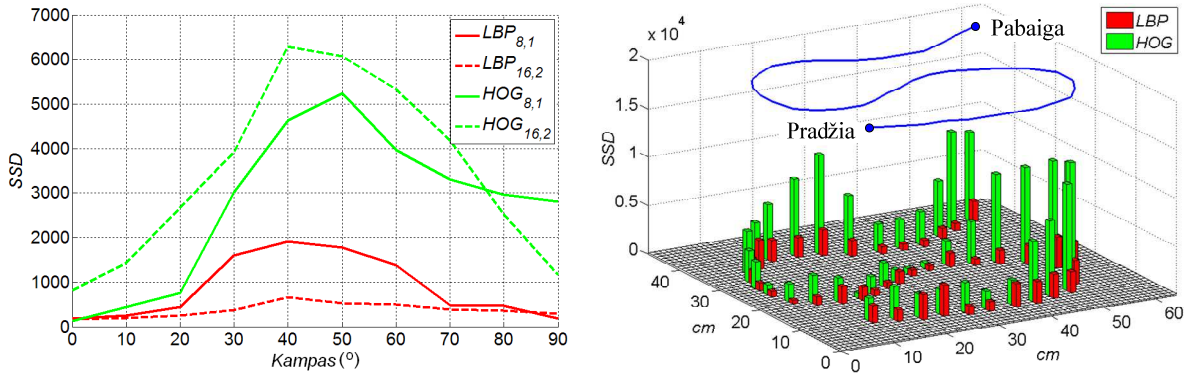
a) *b*) *c*)

d) *e*) *f*)

4.10 pav. HOG (*a*, *d*) ir LBP (*b*, *e*) deskriptoriai skaičiuoti objektui esant šviesiame (*c*) ir tamsiame (*f*) fone

ir sutapdinamos su kiekviename sekančiame kadre judesio sritims paskaičiuotomis $HOG_{captured}$ ir $LBP_{captured}$. Leidžiamas klaidos slenkstis lygus $SSD_{TH} = 2^{13} = 8192$, jis atitinka 5 % vidutinišką deskriptorių stulpelių nesutapimo lygį sekamo objekto dydžiui. Objektas stumiamas link neapšviesto fono srities ir realiu laiku matuojamos histogramos bei didėjanti klaida. Iš 4.10 paveikslo *f* matoma, kad taikant HOG deskriptorių sekamas objektas pamestas anksčiau negu taikant LBP, nes sutapdinimo klaida slenkstinę vertę pasiekia greičiau ir objekto buvimo vietoje viršijama tris kartus. Kadangi HOG metodas dėl skaisčio gradientų priskirimo griežtai apibrėžtiems stulpeliams nėra atsparus objekto pokyčiams, tai nežymus objekto pasisukimas ar pakreipimas mažina arba didina gradientų kiekybinį įvertį tam tikruose stulpeliuose.

Sekimo pagal tekstūrą algoritmo atsparumas objekto pasisukimui tiriamas sekant 4.10 paveiksle *c* pateiktą objektą. Deskriptorių $LBP_{8,1}$, $LBP_{16,2}$, $HOG_{8,1}$, $HOG_{16,2}$ sutapdinimo klaidos priklausomybė nuo objekto posūkio kampo pateikta 4.11 paveiksle *a*. Ties 40° – 50° ruože gaunamas didžiausias skirtumas tarp saugomo ir objektui skaičiuojamo deskriptorių, kas atitinka kitų autorių gautus rezultatus, kur ties 45° gaunama didžiausia klaida [45]. $LBP_{16,2}$ turi mažesnį už $LBP_{8,1}$ sutapdinimo skirtumą, nes lokalieji dvejetainiai modeliai sudaromi iš dvigubai didesnio kiekio kaimyninių taškų išsidėsčiusių $R = 2$ spindulio atstumu nuo nagrinėjamo taško. Objektui besisukant $LBP_{16,2}$ turi dvigubai daugiau diskrečių kampų taško kodavimui negu $LBP_{8,1}$, todėl dvejetainis kodas gali labiau tolygiai suktis. $HOG_{16,2}$ turi didesnį už $HOG_{8,1}$ sutapdinimo skirtumą, nes gradiento kryptis taške nustatoma dvigubai didesniu tikslumu, todėl nežymus objekto posūkis sukelia didelį skirtumo šuolį.



a) *b*)

4.11 pav. Deskriptorių sutapdinimo skirtumo priklausomybė nuo objekto posūkio kampo (*a*), skirtumo vertės kitimas kelyje (*b*)

Norint kelio ruože teisingai sekti objektus pagal jų tekstūrą, reikia nustatyti klaidos slenksčius SSD_{TH} deskriptoriams $HOG_{8,1} + HOG_{16,2}$ ir $LBP_{8,1} + LBP_{16,2}$. Deskriptorių sutapdinimo skirtumo grafikas pateiktas 4.11 paveiksle *b* sekant 4.10 paveiksle *c* pateiktą objektą. Matyti, kad sutapdinimo klaida yra mažiausia centre, nes šioje pozicijoje išsaugoti pradiniai deskriptoriai. Didžiausi skirtumai pastebimi trajektorijos posūkiuose. Nustatyta, kad teisingam objekto sekimui

taikant HOG metodą reikia parinkti klaidos slenkstį $SSD_{HOG} = 1{,}2\times10^4$ ir $SSD_{LBP} = 0{,}3\times10^4$ taikant LPB tekstūros kodavimo metodą.

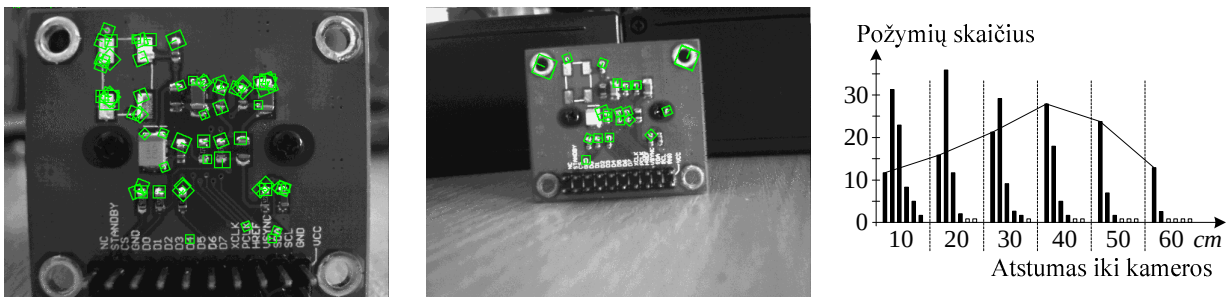
4.3. Sekimo pagal būdinguosius taškus algoritmų tyrimas

Trijuose LPLM įrenginiuose įgyvendintas būdingųjų taškų vaizde išskyrimo algoritmas tiriamas ieškant požymių įvairiuose vaizduose (4.12 pav.). Nustatyta, kad algoritmas teisingai aptinka ir pažymi juodus taškus bei kampus baltame fone ir atvirkščiai. Tyrimo metu nustatyta, kad adaptyviojo slenksčio apatinė riba turi būti lygi 500, žemiau kurios normuotieji Hessian determinanto atsakai nelaikomi požymiais. Aparinės ribos mažinimas veda prie didesnio požymių aptikimo skaičiaus esant blogam apšvietimui, tačiau aptinkama vis daugiau nepageidaujamų besidubliuojančių požymių esant geram apšvietimui. Kuo didesnė apatinė riba, tuo determinanto vertė turi būti didesnė, kad būdingasis taškas būtų išskirtas.

a) *b*) *c*)

4.12 pav. Būdingųjų taškų vaizde išskyrimo ir sekimo rezultatai

Objekto atstumo iki kameros įtakos išskirtų požymių vaizde kiekiui tyrimo rezultatai pateikti 4.13 paveiksle. Nustatyta, kad tolinant objektą nuo kameros būdingieji taškai iš aukštesnės skalės pereina į žemesnę. Iš vaizdo dingsta žemiausios antros skalės požymiai, kai Hessian determinanto vertė mažesnė už slenkstinę ribą ir aptinkami nauji (dažniausiai aukštesnėje skalėje), kai vaizde išskiriamas taško arba kampo elementas. Aptiktų požymių kiekis kadre gali staigiai pakisti pasukus objektą link šviesos šaltinio, nes determinanto vertė priklauso nuo vaizdo kontrasto ir šviesio.

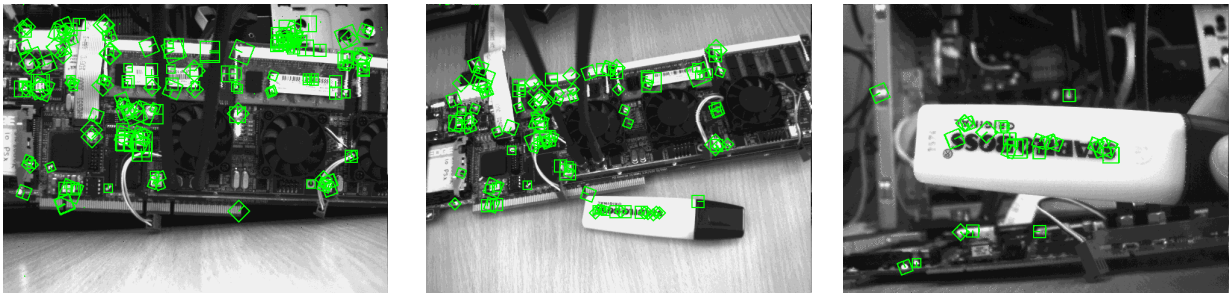


a) *b*) *c*)

4.13 pav. Išskirtų požymių skaičiaus vaizde priklausomybės nuo objekto atstumo iki kameros tyrimo rezultatai

Tiriant būdingųjų taškų išskyrimą įvairiuose objektuose nustatyta, kad požymių skaičiaus kitimo pobūdis didinant atstumą iki objekto priklauso nuo objektą sudarančių smulkių dalių kiekio, jų skaisčio lygio ir tarpusavio atstumo.

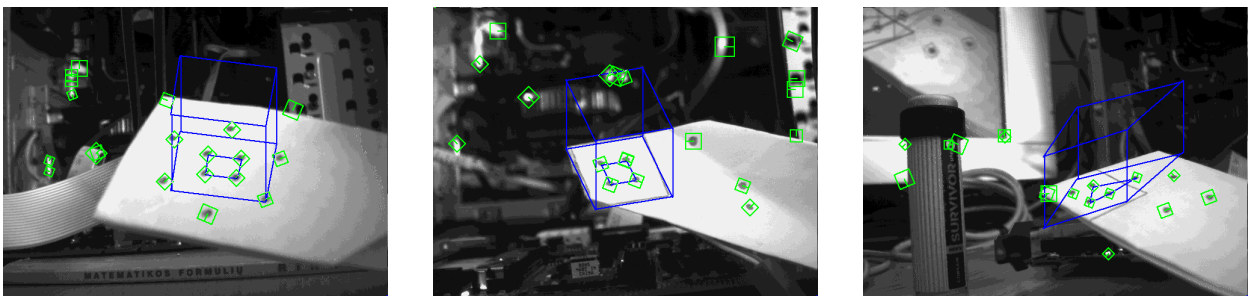
Objekto pasisukimo ir mastelio pasikeitimo tyrimo rezultatai išskiriant būdinguosius taškus pateikti 4.14 paveiksle. Matyti, kad dalis požymių perėjo į aukštesnę ar žemesnę skalę, ką pažymi pasikeitę būdingųjų taškų žalios spalvos žymeklių dydžiai. Dalis požymių dingsta dėl jų užstojimo arba pernelyg didelio posūkio kampo, kad determinanto vertė yra mažesnė už apatinę adaptyviojo slenksčio ribą. Todėl, norint stabiliai sekti objektą, reikia kiekybiškai vertinti būdingųjų taškų deskriptorių sutapdinimą.



a) *b*) *c*)

4.14 pav. Objekto pasisukimo ir mastelio pasikeitimo įtakos teisingam būdingųjų taškų aptikimui vaizde tyrimo rezultatai

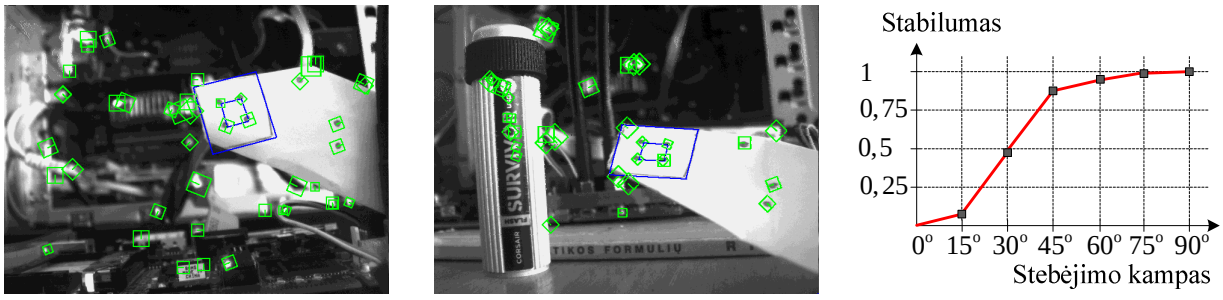
Erdvinio kubo sekimo ir išvedimo į monitorių rezultatai taikant SURF – požymių vaizde kodavimo, keturių taškų išskyrimo ir linijų braižymo Bresenham algoritmus pateikti 4.15 paveiksle. Keturių markerio taškų vaizde išskyrimo atsparumui identiškiems taškams vaizde patikrinti aplink keturių taškų markerį nubraižyti šeši taškai (4.15 pav., *a*). Nustatyta, kad keturių mažiausių atstumų iki kitų būdingųjų taškų paieškos algoritmas veikia teisingai, bet tik su sąlyga, kad yra aptikti visi keturi markeriui priklausantys taškai. Iš 4.15 paveikslo *b* matoma, kad trys šalia markerio esantys taškai teisingai atmetami. Jeigu nepageidaujamų juodų taškų baltame fone yra keturi ir daugiau, ir jie koncentruojasi vienoje iš markerio pusių, tai markeris išskiriamas klaidingai, kadangi mažiausios atstumų sumos yra tuose taškuose, kurie kitų atžvilgiu išsidėstę arčiau vidurio (4.15 pav., *c*).



a) *b*) *c*)

4.15 pav. Keturių markerio taškų vaizde išskyrimo erdviniam kubui braižyti tyrimo rezultatai

Markerio plokštumos pasisukimo kampo kameros atžvilgiu tyrimo rezultatai pateikti 4.16 paveiksle. Iš 4.16 paveikslo *a* matoma, kad skirtingose skalėse esantys keturi taškai teisingai priskiriami sekamam markeriui. Mažinant markerio plokštumos ir vaizdo kameros ašies sudaromą kampą mažėja keturių taškų išskyrimo stabilumas (4.16 pav., *c*). Kadangi požymių deskriptorius koduoja juodą būdingąjį tašką ir aplink jį esantį baltąjį foną, tai kampui mažėjant mažėja juodojo taško dydis ir į aplinkinio balto fono sritį patenka gretimi požymiai.



a) *b*) *c*)

4.16 pav. Markerio pasisukimo kampo įtakos taškų išskyrimo stabilumui tyrimo rezultatai

Markerio plokštuma arba erdvinis kubas teisingai braižomi tik tuo atveju, kai visi keturi taškai yra išskirti. Kai vaizde aptinkami mažiau negu keturi markeriui būdingi taškai, tai plokštuma nebraižoma. Nors vieno iš markerio taškų neaptikimo atveju plokštuma braižoma klaidingai, jeigu vaizde egzistuoja požymiai atitinkantys markerio taškus. Kubo ir plokštumos braižymo tyrimai atlikti esant geram apšvietimui. Apšvietimui blogėjant didėja dalies požymių neišskyrimo tikimybė, todėl patikimam objektų sekimui patartina taikyti kuo daugiau požymių (nebūtinai taškų) kiekybiškai vertinant deskriptorių sutapdinimo rezultatus.

4.4. Trumpas skyriaus apibendrinimas

Visi LPLM įrenginyje įgyvendinti objektų sekimo vaizde algoritmai veikia realiu laiku 25 MHz taktiniu dažniu (išskyrus sekimą pagal spalvą). Ištirtas spalvotų sričių vaizde sekimo algoritmas. Išskirtos spalvotos sritis poruojamos ir taikomos nustatant atstumus tarp jų ir objekto iki vaizdo kameros. Pavyzdžio sutapdinimo su vaizdu algoritmas taikomas įvairiems objektams ir veidui sekti. Įmanoma stabiliai sekti objektą, kurio judesio pagreitis kinta laike taikant pavyzdžio persimokinimo ir adaptyvų minimalios sutapdinimo klaidos paieškos srities algoritmus. Taikant judesio vaizde aptikimo algoritmą greitai arba lėtai judantiems objektams tiksliai sekti, būtina parinkti atitinkamai mažą arba didelį žingsnį tarp apdorojimui taikomų kadrų. Kuo sudėtingesnė objekto tekstūra, tuo stabiliau jis sekamas taikant HOG ir LBP deskriptorius kartu su fono pašalinimo arba judesio aptikimo algoritmais. Iš požymių vaizde išskyrimo algoritmo SURF tyrimo seka, kad įmanomas mastelio ir posūkio kampo pasikeitimams atsparus objekto sekimas, tačiau būtina kiekybiškai vertinti sutapdinamus objekto būdinguosius taškus dėl dalies požymių užstojimo ir apšvietimo pasikeitimų.

5. APIBENDRINIMAS. IŠVADOS

Baigiamajame magistro darbe iškeltas tikslas pasiektas ištyrus realiu laiku veikiančių objektų sekimo vaizde algoritmų įgyvendinimo LPLM įrenginiuose galimybes. Analitinės literatūros apžvalgos metu, įgyvendinus algoritmus ir ištyrus jų veikimą, suformuluotos darbo išvados:

1. Atlikus analitinę literatūros apžvalgą galima teigti, kad:
   a) Elektros ir elektronikos inžinerijoje šiuo metu yra jaučiamas poreikis įgyvendinti LPLM objektų sekimo vaizde algoritmus arba jų dalis, kurios reikalauja dažno kreipimosi į atmintį, dėl vienalaikio šimtų filtro elementų adresavimo galimybės.
   b) sekimo pagal tekstūrą ir būdinguosius taškus algoritmai reikalauja atlikti daug loginių ir aritmetinių operacijų, nes būtina apskaičiuoti būdingųjų taškų požymius vaizde, filtruoti, koduoti ir lyginti su žinomais, todėl šių procesų išlygiagretinimas paspartina vaizdo apdorojimą.
   c) LPLM įrenginiuose įgyvendinami algoritmai dažnai modifikuojami ir supaprastinami tam, kad būtų racionaliai panaudoti loginių matricų resursai ir išlaikyta greitaveika taktiniam dažniui esant dešimties ar šimtų megahercų eilės.
   d) Siekiant greitaveikos, procesų lygiagretumo ir aparatinių išteklių panaudojimo privalumų programinius procesorius loginėse matricose patartina kurti toms algoritmų dalims, kurioms būdingas nuoseklus veikimas (dažniausiai sąsajų protokolai) ir tik tuo atveju, kai laiko sąnaudos būtų didesnės juos įgyvendinant aparatūroje.
2. Įgyvendinus LPLM įrenginyje objektų sekimo vaizde algoritmus ir atlikus eksperimentinę jų veikimo patikrą galima teigti, kad:
   a) VHDL aparatūros aprašymo kalba įgyvendinti ir modifikuoti objektų sekimo vaizde algoritmai gali veikti realiu laiku ir kitame kadre atnaujinti objekto žymeklio vietą monitoriuje.
   b) pavyzdžio sutapdinimui su vaizdu taikomo slenkančio lango dydis turi būti apribotas atsižvelgiant į turimus atminties resursus, todėl jo padidinimui siūloma skaičiavimams taikyti kas ketvirto vaizdo taško skaisčio vertę.
   c) dėl ribotų atminties resursų tekstūros deskriptorius siūloma skaičiuoti srityse, kuriose aptiktas judėjimas.
   d) tarp trijų LPLM paskirstytos SURF požymių vaizde išskyrimo algoritmo dalys gali veikti sinchroniškai ir be klaidų pažymėti būdingųjų taškų vietas.
3. Atlikus objektų sekimo vaizde algoritmų tyrimą galima teigti, kad:

a) spalvotos sritys vaizde pažymimos ir atitikmenys randami teisingai, tačiau tikslaus atstumo iki sekamų automobilio šviesų nustatymas yra abejotinas dėl didėjančios paklaidos objektui tolstant.
b) pavyzdžio sutapdinimo su vaizdu algoritmas objektą seka stabiliai, kai taikomas adaptyvios paieškos srities algoritmas, filtro koeficientai lygūs $k_{1234} = [1\ 2\ 4\ 8]$, o persimokinimo klaida – $err_{const} = 0x0600$.
c) judesio aptikimo ir fono vaizde pašalinimo algoritmus patartina taikyti, kai vaizdo kamera nėra sukama.
d) 5 % vidutinis tekstūros deskriptorių stulpelių nesutapimas leidžia atskirti sekamos tekstūros objektą nuo kitų. Kuo sudėtingesnė objekto tekstūra, tuo stabiliau jis sekamas.
e) dėl dalies būdingųjų taškų užstojimo arba apšvietimo pasikeitimų patartina naudoti kuo daugiau išskirtų požymių taikant SURF algoritmą.

4. Atlikus eksperimentinius tyrimus, suformuluotos pirminio vaizdo apdorojimo ir dvimačių filtrų įgyvendinimo aparatūroje signalams realiu laiku apdoroti rekomendacijos:
a) vienalaikiam dvimačio filtro elementams adresuoti taikyti slenkantį langą, kurio eilutes saugoti BRAM atmintyse. Vieną iš asinchroniškai veikiančių BRAM atminties blokų duomenų sąsajų taikyti duomenų rašymui, kitą – skaitymui.
b) procesų jautrumo sąrašuose taikyti kuo mažesnį taktavimo signalą mažinant galios suvartojimą ir išlaikant pakankamą atstumą tarp taktinio signalo frontų, nes algoritmui sudėtingėjant panaudojama vis daugiau loginių celių, kurioms yra ribojamas aukšto dažnio signalo sinchroniškas generavimas.
c) taikant DCM ar PLL parinkti tokį vaizdo signalų apdorojimo dažnį, kad jis būtų kartotinis taškų į monitorių išvedimo dažniui.
d) sveiko skaičiaus dalybai arba sandaugai iš slenkančio kablelio konstantos taikyti logines, postūmio ir sumos arba skirtumo operacijas, resursų kurioms loginėse matricose yra daugiau ir jos vykdomos sparčiau (per vieną taktinį signalą) negu įterpiant slenkančio kablelio modulius, kurie rezultatą vėlina per 8 ir daugiau (priklauso nuo norimo tikslumo) taktinių impulsų.
e) patartina algoritmo dalis įgyvendinti atskiruose moduliuose, saugoti skirtinguose bylose juos sujungiant pagrindiniame modulyje. Nuoseklaus ir lygiagretaus veikimo kodo dalis jungti į funkcijas arba procedūras, kurias saugoti bibliotekose, taip darant programos kodą labiau įskaitoma ir taikytina kituose projektuose.

Literatūra

1. NIXON, M. S.; AGUARDO, A. S. 2002. Feature Extraction and Image Processing, *British Library*, 360 p.
2. YILMAZ, A.; JAVED, O.; SHAH, M. 2006. Object Tracking: A Survey, *ACM Computing Surveys* 38(4), 45 p.
3. GORRY, B.; CHEN, Z. 2007. Using Mean-Shift Tracking Algorithms for Real-Time Tracking of Moving Images on an Autonomous Vehicle Testbed Platform. *World Academy of Science, Engineering and Technology*, 209-214 p.
4. MARSHNER, A. R. 2007. An FPGA-based Target Acquisition System, *Thesis of Master of Science in Computer Engineering*, Virginia Polytechnic Institute, 108 p.
5. PRICE, A.; PYKE, J.; ASHIRI, D.; CORNALL, T. 2006. Real-Time Object Detection for an Unmanned Aerial Vehicle using an FPGA-based Vision System, *Proceedings of the IEEE International Conference on Robotics and Automation*, 2854-2859 p.
6. YU, Y. H.; KWOK, N. M.; Ha, Q. P. 2009. FPGA-Based Real-Time Color Tracking for Robotic Formation Control, *26th Internetional Symposium on Automation and Robotics in Construction*, 252-258 p.
7. WALCZYK, R.; ARMITAGE, A.; BINNIE, T. D. 2010. FPGA Implementation of Hot Spot Detection in Infrared Video, *ISSC 2010*, 6 p.
8. ISHII, I.; TANIGUCHI, T.; SUKENOBE, R.; YAMAMOTO, K. 2009. Development of High-Speed and Real-Time Vision Platform, *IEEE International Conference on Intelligent Robots and Systems*, 3671-3678 p.
9. YAMAOKA, K.; MORIMOTO, T.; ADACHI, H. 2006. Image Segmentation and Pattern Matching-Based FPGA/ASIC Implementation Architecture of Real-Time Object Tracking, *Research Center for Nanodevices and Systems in Hiroshima University*, 176-181 p.
10. JIANG, J.; ZHANG, G.; WEI, X.; LI, X. 2009. Rapid Star Tracking Algorithm for Star Sensor, *IEEE A&E Systems Magazine*, 23-33 p.
11. SCHOENEMANN, T.; CREMENS, D. 2008. Globally Optimal Shape-based Tracking in Real-Time. *IEEE Computer Society Conference on Computer Vision and Patter Recognition*, 6 p.
12. HUSIN, M. H.; OSMAN, H. 2010. Development of Shape Pattern Recognition for FPGA-based Object Tracking System, *Internation Conference on Computer Applications, Industrial Electronics*, 80-84 p.
13. TALEI, V. 2010. Implementation of a Bacterium Tracking System on FPGA, *Thesis of Master of Science in Informatic Engineering*, Montreal Polytechnic Institute, 76 p.
14. BAUM, M.; HANEBECK, U. D. 2011. Shape Tracking of Extended Object and Group Targets with Star-Convex RHMs, *Karlsruhe Institute of Technology*, 8 p.
15. BOCHEM, A.; HERPERS, R.; KENT, K. B. 2010. Hardware Acceleration of Blob Detection for Image Processing, *Third International Conference on Advances in Circiuts, Elevtronics and Micro-electronics*, 28-33 p.
16. YAN, X.; MEILIAN, S.; ZUOLIANG, C. 2011. Multi-Object Tracking for Mobile Navigation in Outdoor with Embedded Tracker, *7-th International Conference on Natural Computation*, 1739-1743 p.
17. ALI, A.; MIRZA, S. M. 2006. Object Tracking using Correlation, Kalman Filter and Fast Mean Shift Algorithms, *2nd International Conference on Emerging Technologies*, 174-178 p.
18. AZARI, M.; SEYFI, A.; REZAIE, A. H. 2011. Real-Time Multiple Object Tracking and Occlusion Reasoning Using Adaptive Kalman Filter, *AIMS Research Lab. Amirkabir Uni. of Technology*, 5 p.
19. O'MALLEY, R.; JONES, E.; GLAVIN, M. 2010. Rear-Lamp Vehicle Detection and Tracking in Low-Exposure Color Video for Night Condition, *IEEE Transactions on Intelligent Transportation Systems* 11(2), 453-462 p.
20. *Introduction to Computer Vision* [interaktyvus]. 2007. CSE Department, Penn State University [žiūrėta 2012 m. kovo 19 d.]. Prieiga per internetą: < http://www.cse.psu.edu/~rcollins/CSE486>

21. KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. 2011. *Centre for Vision, Speach and Signal Processing, University of Surrey*, 4 p.
22. BOLME, D. S.; DRAPER, B. A.; BEVERIDGE, J. R. 2009. Average of Synthetic Exact Filters, *Computer Vision and Pattern Recornition*, 2105-2112 p.
23. BOLME, D. S.; LUI, Y. M.; DRAPER, B. A.; BEVERIDGE, J. R. 2009. Simple Real-Time Human Detection Using a Single Correlation Filter, *International Workshop on Performance Evaluation of Tracking and Surveilance*, 1-8 p.
24. BOLME, D. S.; BEVERIDGE, J. R.; DRAPER, B. A.; LUI, Y. M. 2010. Visual Object Tracking using Adaptive Correlation Filters, *Computer Vision and Pattern Recornition*, 2544-2550 p.
25. MISHRA, P.; BHAVANI, B.; NAIR, L. 2011. Architectures for FPGA-Based Implementation of Motion Estimation of Dynamic Obstacles for Autonomous Robot Navigation, *Third International Conference on Computational Intelligence, Communication Systems and Networks*, 292-297 p.
26. MENEZES, G.; SILVA-FILHO, A. 2010. Motion Detection of Vehicles Based on FPGA, *VI Southern Programmable Logic Conference*, 151-154 p.
27. CUCCHIARA, R.; PICCARDI, M.; Prati, A.; Scarabottolo, N. 1999. Real-Time Detection of Moving Vehicles, *International Conference on Image Analysis and Processing*, 618-623 p.
28. KRYJAK, T.; KOMORKIEWICZ, M.; GORGON, M. 2011. Real-Time Moving Object Detection for Video Surveillance System in FPGA, *Conference on Design and Architectures for Signal and Image Processing*, 1-8 p.
29. MATRELLA, G.; MARANI, D. 2011. An Embedded Video Sensor for a Smart Traffic Light, *14th Euromicro Conference on Digital System Design*, 769-776 p.
30. ABUTALEB, M. M.; HAMDY, A.; ABUELWAFA, M. E.; SAAD, E. M. 2009. FPGA-Based Object-Extraction Based on Multimodal Σ-Δ Background Estimation, *2009 2nd International Conference on Computer, Control and Communication*, 1-7 p.
31. DONGXIANG, Z.; HONG, Z. 2005. Modified GMM Background Modeling and Optical Flow for Detection of Moving Objects," *IEEE International Conference on Systems, Man and Cybernetics* 3(1), 2224- 2229 p.
32. LI, X.; JING, X. 2011. FPGA-Based Mixture Gaussian Background Modeling and Motion Detection, *Seventh International Conference on Natural Computation* 4(1), 2078-2081 p.
33. TOMASI, M.; VANEGAS, M.; BARRANCO, F. 2012. Massive Parallel-Hardware Architecture for Multiscale Stereo, Optical Flow and Image-Structure Computation, *IEEE Transactions on Circuits and Systems for Video Technology* 22(2), 282-294 p.
34. CLAUS, C.; LAIKA, A. 2009. High Performance FPGA-Based Optical Flow Calculation Using the Census Transformation, *IEEE Intelligent Vehicles Symposium*, 1185-1190 p.
35. GONZALEZ, D.; BOTELLA, G.; MOKHEERJE, S. 2011. FPGA-Based Acceleration of Block Matching Motion Estimation Techniques, *International Conference on Field Programmable Logic and Applications*, 389-392 p.
36. KANGLIN, C.; LORENZ, D. A. 2012. Image Sequence Interpolation Based on Optical Flow, Segmentation, and Optimal Control, *IEEE Transactions on Image Processing* 21(3), 1020-1030 p.
37. JAY, H. C.; Lee, D.; Bang, H. 2011. Tracking an Unknown Moving Target from UAV: Extracting and Localizing an Moving Target with Vision Sensor Based on Optical Flow, *5th International Conference on Automation, Robotics and Applications*, 384-389 p.
38. ATAER, E.; GHADARGHADAR, N.; ZAREIAN, R. 2011. Motion Flow Analysis in Cell Videos Using a Multi-Level Clustering Method, *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 7767-7770 p.
39. BROOKSHIRE, J.; STEFFENSEN, J.; XIAO, J. 2010. FPGA-Based Pedestrian Detection, *Massachusetts Institute of Technology*, 24 p.
40. ZHU, Q.; AVIDAN, S.; YEH, M. 2006. Fast Human Detection Using a Cascade of Histograms of Oriented Gradients, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2(1), 1491- 1498 p.

41. BAUER, S.; KOHLER, S.; DOLL, K.; BRUNSMANN, U. 2010. FPGA-GPU architecture for kernel SVM pedestrian detection, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 61-68 p.
42. KADOTA, R.; SUGANO, H.; HIROMOTO, M.; Ochi, H. 2009. Hardware Architecture for HOG Feature Extraction, *Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 1330-1333 p.
43. HIROMOTO, M.; MIYAMOTO, R. 2009. Hardware Architecture for High-Accuracy Real-Time Pedestrian Detection with CoHOG Features, *IEEE 12th International Conference on Computer Vision Workshops*, 894-899 p.
44. TEHRANI, N. H.; TAKAHASHI, K.; MITA, S.; McALLESTER, D. 2011. Vehicle Detection and Tracking at Nighttime for Urban Autonomous Driving, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4442-4447 p.
45. OJALA, T.; PIETIKÄINEN, M. 2002. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(7), 971-987 p.
46. LI, L.; FIEGUTH, P. 2012. Texture Classification from Random Features, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(3), 574-586 p.
47. ZHAO, G.; AHONEN, T.; MATAS, J.; PIETIKAINEN, M. 2012. Rotation-Invariant Image and Video Description With Local Binary Pattern Features, *IEEE Transactions on Image Processing* 21(4), 1465-1477 p.
48. JAE, Y. C.; YONG, M. R.; PLATANIOTIS, K.N. 2012. Color Local Texture Features for Color Face Recognition, *IEEE Transactions on Image Processing* 21(3), 1366-1380 p.
49. JIN. S.; KIM, D. 2009. An FPGA-Based Parallel Hardware Architecture for Real-Time Face Detection Using a Face Certainty Map, *20th IEEE International Conference on Application-Specific Systems, Architectures and Processors*, 61-66 p.
50. MATURANA, D.; MERY, D.; SOTO, A. 2011. Learning discriminative local binary patterns for face recognition," *IEEE International Conference on Automatic Face & Gesture Recognition and Workshops*, 470-475 p.
51. LOPEZ, L. S. 2010. *Local Binary Patterns applied to Face Detection and Recognition*: Tiriamasis projektas. Barselona: Universitat Politècnica de Catalunya. 146 p.
52. BAY, H.; ESS, A.; TUYTELAARS, T.; VAN GOOL, L. 2007. Speeded-Up Robust Features (SURF), *Computer Vision and Image Understanding* 1(10), 346-359 p.
53. MURILLO, A.C.; GUERRERO, J.J.; SAGUES, C. 2007. SURF Features for Efficient Robot Localization with Omnidirectional Images, *IEEE International Conference on Robotics and Automation*, 3901-3907 p.
54. FISCHER, J.; RUPPEL, A.; WEISSHARDT, F.; VERL, A. 2011, A Rotation Invariant Feature Descriptor O-DAISY and its FPGA Implementation, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2365-2370 p.
55. SCHVAB, J. 2011. FPGA-Based Computer Vision Embedded Module, *Thesis of Master of Science in Electrical Engineering*, Czech Technical University in Prague, 80 p.
56. GUICHARD, F.; MOREL, J. M.; RYAN, R. 2008. *Contrast Invariant Image Analysis and PDE's* [interaktyvus]. Cachan : Ecole Normale Supérieure [žiūrėta 2012 m. kovo 24 d.]. Prieiga per internetą: <http://dev.ipol.im/~morel/JMMBookOct04_complete.pdf>
57. MIZUNO, K.; NOGUCHI, H.; GUANGJI, H.; TERACHI, Y. 2010. Fast and Low-Memory-Bandwidth Architecture of SIFT Descriptor Generation with Scalability on Speed and Accuracy for VGA Video, *International Conference on Field Programmable Logic and Applications*, 608-611 p.
58. LOWE, D.G. 1999. Object Recognition from Local Scale-Invariant Features, *The Proceedings of the Seventh IEEE International Conference on Computer Vision* 2(1), 1150-1157 p.
59. BEI, N. W.; YU, S.; GETIAN, Y.; JIE, X. 2008. Developing a Smart Camera for Road Traffic Surveillance, *IEEE 10th Workshop on Multimedia Signal Processing*, 826-831 p.

60. CLAUS, C.; HUITL, R.; RAUSCH, J.; STECHELE, W. 2009. Optimizing the SUSAN Corner Detection Algorithm for a High Speed FPGA Implementation, *International Conference on Field Programmable Logic and Applications*, 138-145 p.
61. BYUNG, J.; DONG, W. 2009. An Improved Corner Point Detection using Extreme Value of SUSAN Method for Measuring a Displacement, *ICCAS-SICE*, 5392-5396 p.
62. ROSTEN, E. 2006. High Performance Rigid Body Tracking, *Thesis of Doctor of Philosophy*, University of Cambridge, 153 p.
63. LAGANIERE, R. 1998. Morphological Corner Detection, *Sixth International Conference on Computer Vision*, 280-285 p.
64. MORIMOTO, T.; ADACHI, H.; KIRIYAMA, O. 2010. Multi-View Face Detection and Recognition using Haar-Like Features, *Research Center for Nano-Devices and Systems in Hiroshima University*, 3 p.
65. LEI, J.; FENG, Y.; LIXIN, F. 2011. Real-Time Object Tracking on Mobile Phones, *First Asian Conference on Pattern Recognition*, 560-564 p.
66. *XILINX FPGA Devices* [interaktyvus]. 2012. XILINX [žiūrėta 2012 m. kovo 26 d.]. Prieiga per internetą: <http://www.xilinx.com/products/silicon-devices/fpga/index.htm>
67. *ALTERA FPGA Devices* [interaktyvus]. 2012. ALTERA [žiūrėta 2012 m. kovo 26 d.]. Prieiga per internetą: <http://www.altera.com/devices/fpga/fpga-index.html>
68. ARMINAS, V. 2010. Microblaze programinio procesoriaus įgyvendinimo tyrimas, *Magistro baigiamasis darbas*, VGTU, 115 p.
69. *Line drawing by using Bresenham's algorithm, Wu's algorithm – Computer graphics* [interaktyvus]. 2011. Indiana State University [žiūrėta 2012 m. balandžio 17 d.]. Prieiga per internetą: <http://cs.indstate.edu/~psunkara/p.pdf>
70. PIRZADA, S. J. H., BAIG, M. W., HAQ, E. U., HYUNCHUL S. 2011. A New Adaptive Threshold Technique for Improved Matching in SIFT, *IEEE 54th International Midwest Symposium on Circuits and Systems* (*MWSCAS*), pp.1-4.

PRIEDAI

A priedas. Pranešimo 15-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga

B priedas. Straipsnio IEEE 8 regiono studentų straipsnių konkurse medžiaga

C priedas. Magistro baigiamojo darbo aiškinamasis raštas, sukurtų programų kodai ir objektų sekimo vaizdinė medžiaga (DVD)

A priedas. Pranešimo 15-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

Požymių vaizde išskyrimo algoritmo įgyvendinimas LPLM įrenginyje

Tomyslav Sledevič, EKSfm-10 gr.

Vadovas – dr. Artūras Serackis

Vilnius
2012 03 16

ELEKTRONINIŲ SISTEMŲ KATEDRA
ELEKTRONIKOS FAKULTETAS
VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Turinys

- Tikslas/problema/uždaviniai
- Algoritmo paskirstymas tarp 3-jų LPLM
- Integrinio vaizdo sudarymas
- Slenkančio lango sudarymas
- Determinanto skaičiavimas
- Ekstremumų paieška
- Orientacijos ir deskriptoriaus skaičiavimas
- Rezultatai
- Išvados

T. Sledevič tomyslav.sledevic@el.vgtu.lt

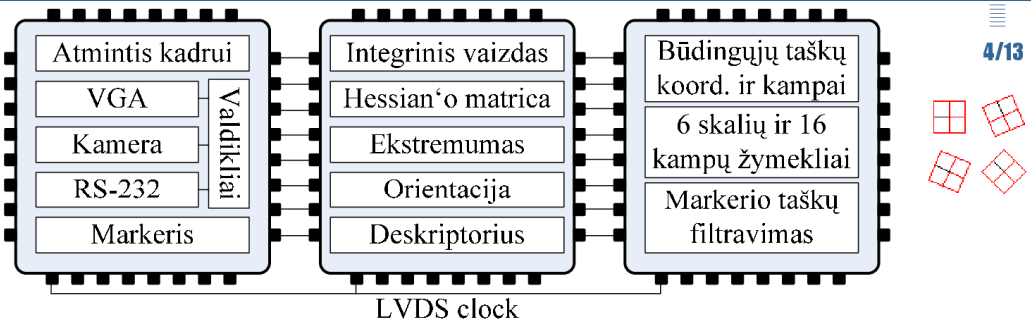
Vilnius
2012 03 16

ELEKTRONINIŲ SISTEMŲ KATEDRA
ELEKTRONIKOS FAKULTETAS
VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

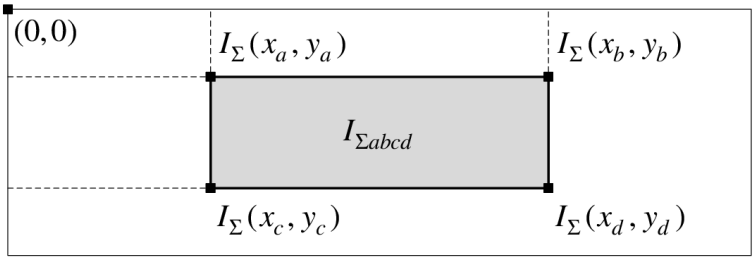
Tikslas/problema/uždaviniai

- **Tikslas** – įgyvendinti ir patikrinti požymių vaizde išskyrimo algoritmą *SURF*, kuris veiktų realiuoju laiku LPLM įrenginyje.
- **Problema** – *2D* filtrų vaizdui apdoroti įgyvendinimo sudėtingumas.
- **Uždaviniai:**
  - Įgyvendintą algoritmą paskirstyti tarp kelių LPLM.
  - Įvertinti požymių vaizde aptikimo teisingumą.

Algoritmo paskirstymas tarp 3-jų LPLM

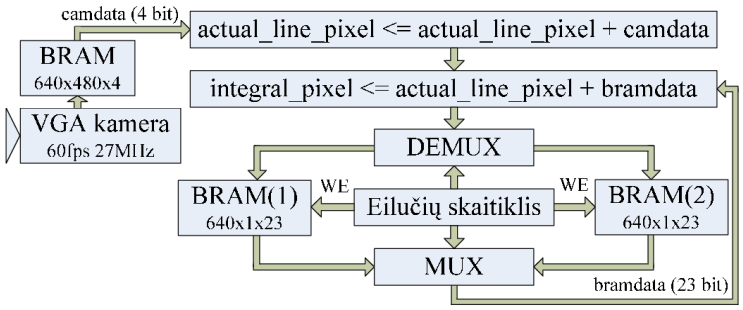


Integrinio vaizdo sudarymas

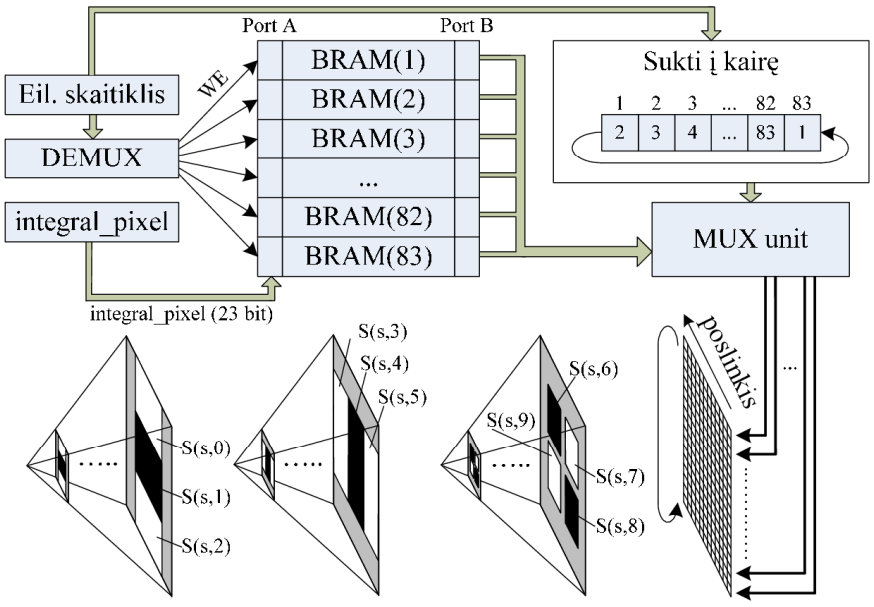


$$I_\Sigma(x,y) = \sum_{x=1}^{X}\sum_{y=1}^{Y} I(x,y)$$

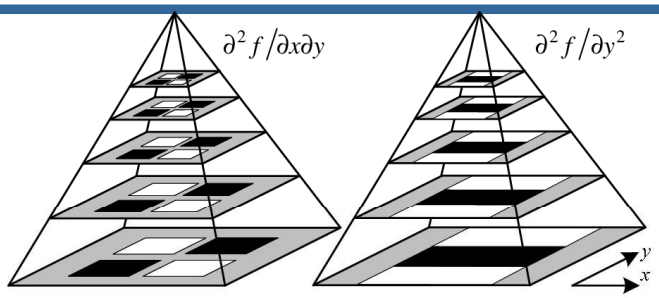
$$I_{\Sigma abcd} = I_\Sigma(x_a, y_a) + I_\Sigma(x_d, y_d) - I_\Sigma(x_b, y_b) - I_\Sigma(x_c, y_c)$$



Slenkančio lango sudarymas



Determinanto skaičiavimas

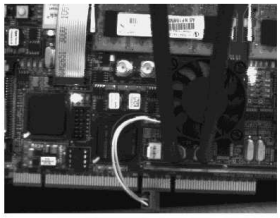


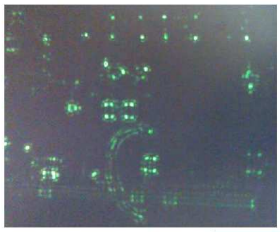
$$w^2 = 0{,}832$$

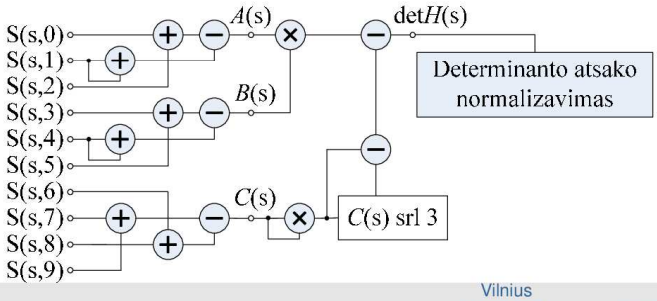
$$w^2 = 0{,}875$$

$$H(x,y) = AB - w^2C^2$$

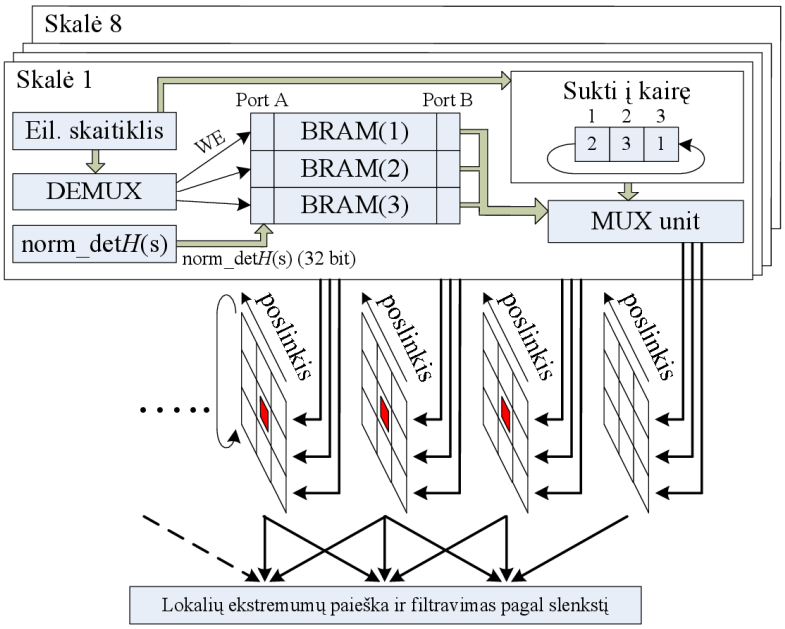
$$H(x,y) = \det\begin{bmatrix} A & C \\ C & B \end{bmatrix} = \det\begin{bmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x\partial y \\ \partial^2 f/\partial x\partial y & \partial^2 f/\partial y^2 \end{bmatrix}$$





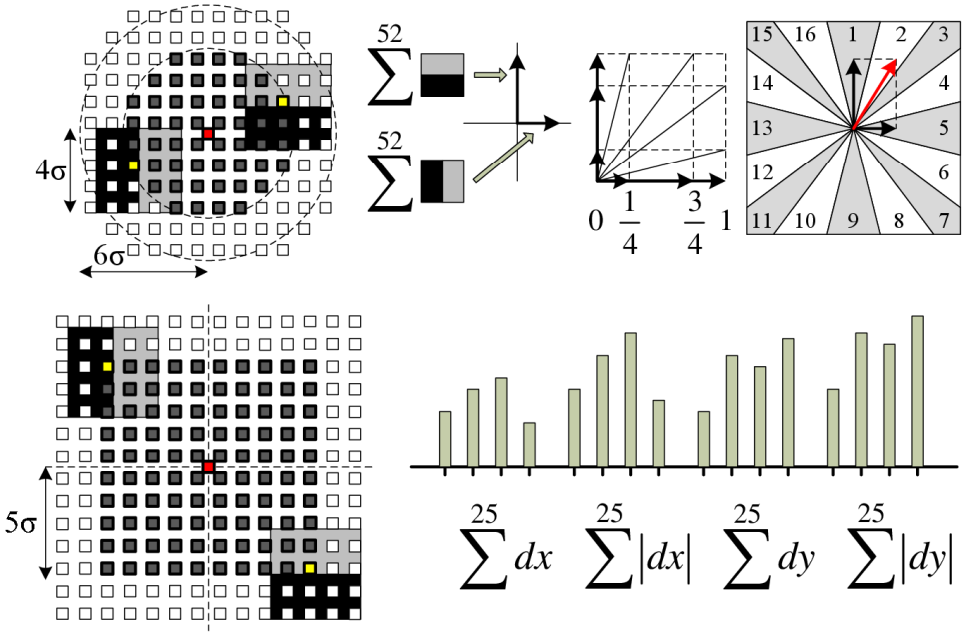


Ekstremumų paieška

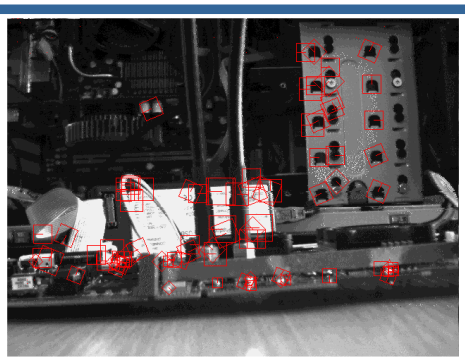


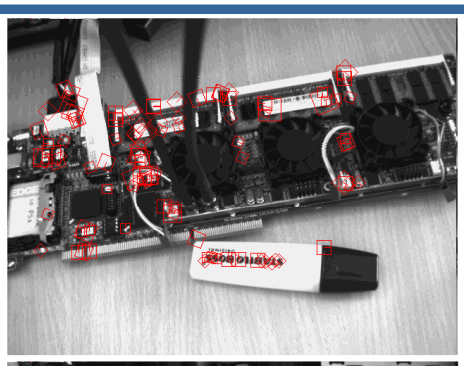
$$TH_{next} = TH_{last} + (N_{last} - N_{const}) \cdot v_{th},$$

Orientacijos ir deskriptoriaus skaičiavimas

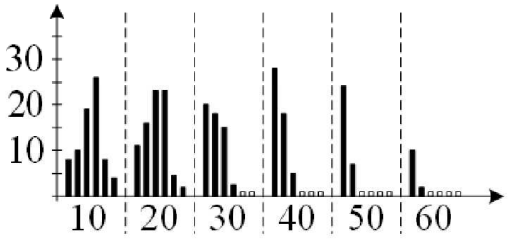


Rezultatai

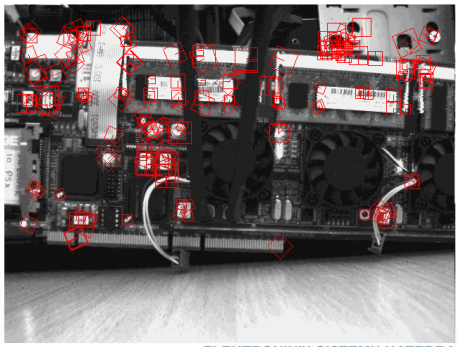




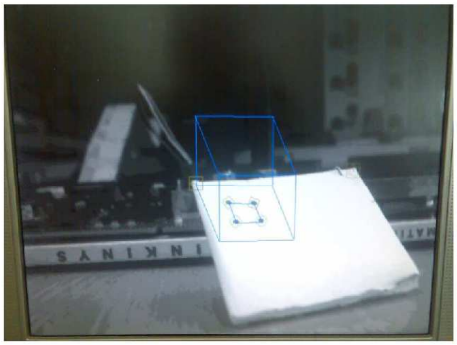
Požymių skaičius

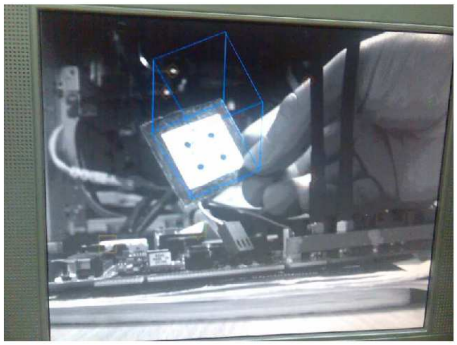


Objekto atstumas iki kameros (*cm*)

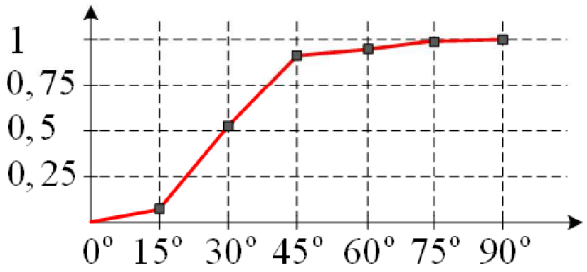


Rezultatai





Stabilumo tikimybė



Markerio plokštumos kampas

Išvados

- Įgyvendintas algoritmas paskirstytas tarp trijų LPLM.
- Vaizdo filtravimas veikia realiu laiku (60 kadrų/s, VGA raiškos vaizdas), loginės matricos taktiniam dažniui esant tik 25 MHz.
- Spartus algoritmo veikimas pasiektas dėl galimybės lygiagrečiai adresuoti kelių tūkstančių elementų slenkantį langą kiekvieno taktinio impulso metu.
- Požymių vaizde aptikimą ir sekamo markerio aptikimo stabilumą įtakoja apšvietimas, todėl patartina naudoti slankiojantį lokalų slenkstį.

Dėkoju kantriems klausytojams.

T. Sledevič tomyslav.sledevic@el.vgtu.lt

Vilnius
2012 03 16

ELEKTRONINIŲ SISTEMŲ KATEDRA
ELEKTRONIKOS FAKULTETAS
VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS

XV-toji Lietuvos jaunųjų mokslininkų konferencija
„MOKSLAS – LIETUVOS ATEITIS“

ELEKTRONIKA IR ELEKTROTECHNIKA

PAŽYMA

Tomyslav Sledevič

Signalų technologijų ir aukštų dažnių technologijų jungtinėje sekcijoje perskaitė pranešimą

Požymių vaizde išskyrimo algoritmo įgyvendinimas LPLM įrenginyje

Mokslo komiteto pirmininkas prof. habil. dr. Romanas Martavičius

Sekcijos pirmininkas dr. Raimondas Pomarnacki

2012 m. kovo mėn. 16 d.
Vilnius

MOKSLAS – LIETUVOS ATEITIS
SCIENCE – FUTURE OF LITHUANIA

2012, Y tomas, Nr. X
2012, Vol. Y, No X

Elektronika ir elektrotechnika
Electronics and electrical engineering

Signalų technologija T 121

MODIFIKUOTO SURF POŽYMIŲ VAIZDE IŠSKYRIMO ALGORITMO ĮGYVENDINIMAS LPLM OBJEKTUI SEKTI REALIU LAIKU

Tomyslav Sledevič
Vilniaus Gedimino technikos universitetas
El. paštas: tomyslav.sledevic@vgtu.lt

Santrauka. Straipsnyje pateikiamas modifikuoto požymių vaizde išskyrimo algoritmo SURF įgyvendinimas lauku programuojamų loginių matricų (LPLM) įrenginiuose. LPLM įrenginiai pasirinkti dėl lygiagrečiai veikiančių procesų įgyvendinimo galimybės taikant VHDL kalbą, kas garantuoja požymių vaizde išskyrimą realiu laiku. Skaičiavimams paspartinti taikomas slenkantis 84×84 taškų dydžio langas, kuriame saugomas sudėtinis vaizdas. Šio slenkančio lango duomenys taikomi Hessiano determinantui, būdingųjų taškų orientacijai ir deskriptoriams skaičiuoti. Požymiai ieškomi aštuoniose skalėse taikant lokalių ekstremumų paiešką. Požymių orientacijos vektorius ir supaprastintas deskriptorius skaičiuojami šešiose skalėse lygiagrečiai. Algoritmo veikimas tiriamas sekant keturių taškų žymeklį ir pagal jį braižant plokštumą arba erdvinį kubą. Skaičiavimai atliekami 25 MHz taktiniu dažniu. Vaizdui gauti taikoma 60 kadrų per sekundę dažnio 640×480 taškų raiškos vaizdo kamera.

Reikšminiai žodžiai: lauku programuojama loginė matrica, požymių išskyrimas, vaizdo filtravimas, objekto sekimas.

Įvadas

SURF (angl. *Speeded up Robust Features*) algoritmas skirtas būdingiesiems taškams vaizde išskirti, koduoti ir sutapdinti nepriklausomai nuo požymių skalės ir posūkio kampo pasikeitimų (Bay *et al.* 2008). Algoritmas plačiai taikomas objektams vaizde sekti, papildytos realybės programose, trimačiams vaizdams rekonstruoti. Procesoriniuose įrenginiuose įgyvendinto SURF algoritmo greitaveika priklauso nuo dvimačių filtrų kiekio, aptiktų požymių ir jiems taikomų posūkio kampų skaičiaus bei deskriptorių dydžio. Dėl algoritmo sudėtingumo net šiuolaikiniai procesoriai negarantuoja stabilaus didelės raiškos ir kadrų per sekundę dažnio vaizdo apdorojimo.

Šiai problemai išspręsti siūloma SURF algoritmą įgyvendinti LPLM. Juose yra galimybė kurti lygiagrečiai veikiančius filtrus. Filtrai gali turėti tūkstančius elementų, kuriuos įmanoma adresuoti sinchroniškai su kiekvienu taktiniu signalu. Lygiagretus procesų paskirstymas LPLM garantuoja vaizdo apdorojimą realiu laiku taktavimo signalui esant kelių dešimčių megahercų eilės ir sunaudojant mažiau galios lyginant su procesoriais.

Vis dažniau daug aritmetinių ir kreipimosi į atmintį operacijų reikalaujantys SURF algoritmo etapai įgyvendinami LPLM. LPLM įgyvendintas Hessian matricos skaičiavimas taikant kelis algoritmo supaprastinimus dėl ribotų aparatinių resursų (Svab *et al.* 2009). Požymių paieškos etapas perkeltas į kompiuterį. Nustatyta, kad algoritmas teisingai aptinka požymius 1024×768 raiškos vaizde 10 kadrų per sekundę dažniu.

LPLM įgyvendintas SURF algoritmas iki posūkio kampų požymiams skaičiavimo etapo imtinai (Bouris *et al.* 2010). Autorių siūlomas algoritmas sunaudoja didesnę dalį BRAM blokų visiems vieno kadro sudėtiniams taškams saugoti. Pasiektas 56 kadrų per sekundę dažnio būdingųjų taškų išskyrimas 640×480 raiškos vaizde taikant 200 MHz taktinį signalą.

Pasiektas 406 kadrų per sekundę dažnio deskriptorių požymiams skaičiavimas 640×480 raiškos vaizde, tačiau tik vienoje skalėje (Fisher *et al.* 2011). Vaizdai siunčiami iš kompiuterio LPLM įrenginiui ir kiekvieno taktinio signalo metu atgal grąžinamos deskriptorių vertės. Požymiams koduoti autoriai taiko posūkiui atsparų O-DAISY deskriptorių. Siūlomas algoritmas veikia 125 MHz dažniu, sunaudojama 353 BRAM ir 403 DSP celių.

SURF algoritmas įgyvendintas dviejose LPLM sunaudojant 90 % atminties resursų (Battezzati *et al.* 2012). Nustatyta, kad požymių kadre sutapdinimo etapas reikalauja daugiausia skaičiavimų, sunaudoja 49 % LUT (angl. *Look-up Table*) bei užtrunka 355 ms. Algoritmas gali aptikti iki 4096 požymių 1024×1024 raiškos vaizde.

Požymių išskyrimo ir linijų braižymo algoritmai

Apibendrintas objekto sekimo pagal būdinguosius taškus algoritmas SURF pateiktas 1 paveiksle. Skaičiuojamas sudėtinis vaizdas, Hessian matricos determinantai lygiagrečiai aštuoniose skalėse. Kuo didesnė skalė, tuo didesnė determinanto vertė, todėl determinantų atsakus būtina normalizuoti. Tarp trijų gretimų skalių ieškoma lokalių ekstremumų – tai determinanto vertės, kurios

ISSN 2029-2341 print / ISSN 2029-2252 online

didesnės už kaimyninius taškus. Stipriausiems požymiams išskirti ekstremumai filtruojami pagal slenkstį. Kiekvienam požymiui skaičiuojami posūkio kampai ir deskriptoriai. Taikant Bresenham linijų braižymo algoritmą pagal išskirtus sekamo objekto požymius braižoma plokštuma arba erdvinis kubas. Į monitorių išvedami sekamo žymeklio ir būdingųjų taškų žymekliai.

1 pav. Būdingųjų taškų išskyrimo ir objekto sekimo algoritmas

Fig. 1. Features extraction and object tracking algorithm

Sudėtinis vaizdas sudaromas ir taikomas pagal lygtis:

$$I_{\Sigma}(x,y)=\sum_{x=1}^{X}\sum_{y=1}^{Y}I(x,y)\,, \tag{1}$$

$$I_{\Sigma}=I_{\Sigma}(x_a,y_a)+I_{\Sigma}(x_d,y_d)-I_{\Sigma}(x_b,y_b)-I_{\Sigma}(x_c,y_c)\,, \tag{2}$$

čia $I(x, y)$ – taško skaisčio vertė, $I_{\Sigma}(x, y)$– sudėtinio vaizdo taško vertė, I_{Σ} – sudėtinio vaizdo taškų verčių suma ribojamame stačiakampio *abcd* plote. Sudėtinio vaizdo taikymo privalumas tas, kad vienodai greitai įvertinama taškų skaisčio suma bet kokio ploto stačiakampyje. Ši savybė spartina kelių skalių ir krypčių 2*D* antros eilės Gauso filtrų ir Haar vilnelių taikymą.

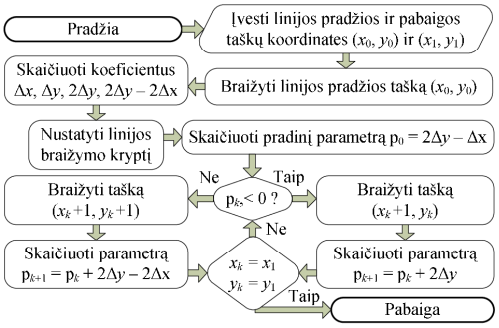
Hessiano determinantas skaičiuojamas pagal lygtis:

$$H(x,y)=\det\begin{bmatrix}A & C\\ C & B\end{bmatrix}=\det\begin{bmatrix}\partial^2 f/\partial x^2 & \partial^2 f/\partial x\partial y\\ \partial^2 f/\partial x\partial y & \partial^2 f/\partial y^2\end{bmatrix}, \tag{3}$$

$$H(x,y)=AB-w^2C^2\,, \tag{4}$$

čia w^2 – svorio koeficientas, kuris lygus 0,83 [1]. Jis taikomas kompensuoti tolydžios funkcijos Gauso filtrų aproksimavimą stačiakampių formos filtrais. Aparatūroje žymiai greičiau vykdoma ne sandaugų iš slankaus kablelio koeficientų, bet sumavimo ir postūmio operacijos. Todėl algoritme įvestas pakeitimas $w^2 = 1 – 1/8 = 0{,}875$.

Linijų braižymui taikomas Bresenham algoritmas (3 pav.), kuriam įgyvendinti naudojamos tik sumos, skirtumo ir palyginimo operacijos (Sunkara 2011). Algoritmo veikimas nuoseklus, nes sekančio linijos taško koordinatėms apskaičiuoti būtina žinoti esamo taško parametrą p_k. Pagal Δx, Δy vertes nustatoma linijos braižymo kryptis.



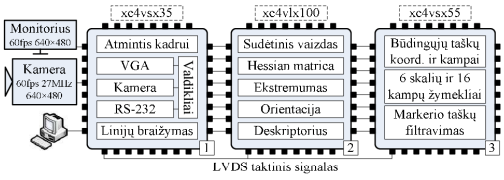
2 pav. Linijų braižymo Bresenham algoritmas

Fig. 2. Bresenham's line drawing algorithm

Bresenham algoritmas linijas braižo tik pirmame koordinačių sistemos oktante, todėl būtina sukeisti koordinačių ašis, kai linijos vektorius nepatenka į 0° – 45°. Parametras p_k nurodo, kur braižyti sekantį tašką $(x_k + 1, y_k)$ ar $(x_k + 1, y_k + 1)$ koordinatėse ir koks bus sekantis p_{k+1} parametras. Linija braižoma tol, kol taško (x_k, y_k) koordinatės nesutampa su pabaigos tašku (x_1, y_1).

SURF algoritmo įgyvendinimas LPLM

SURF algoritmas įgyvendintas trijose LPLM (3 pav.). Valdančiuoju įrenginiu laikoma xc4vsx35, pavaldžiais: xc4vlx100 ir xc4vsx55 matricos. Pirmojoje LPLM yra kadro atminties ir periferinių sąsajų valdikliai, plokštumos ir erdvinio kubo kūrimo algoritmai. Antrojoje LPLM įgyvendintas SURF algoritmas. Trečiojoje LPLM išskiriami žymeklio taškai, saugomi būdingųjų taškų koordinatės ir kampai, kurie susieti su šešių skalių ir šešiolikos kampų žymekliais. Antroji ir trečioji LPLM taktuojamos LVDS (angl. *Low Voltage Differential Signal*) signalu. Algoritmų veikimas sinchronizuojamas vienalaikiu skaitiklių, esančių trijose LPLM, numetimu.

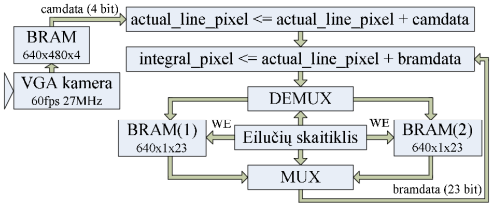


3 pav. Algoritmo paskirstymas tarp trijų LPLM

Fig. 3. Distribution of algorithm in three FPGA

Sudėtiniam vaizdui (angl. *Integral Image*) sudaryti kiekvieno 25 MHz taktinio signalo metu taikomos dvi sumavimo operacijos (4 pav.). Pirmosios sumos operacijos metu sudedamos vienoje eilutėje esančių taškų vertės,

antrosios metu – viename stulpelyje esančių taškų vertės. Kol BRAM(1) atmintyje saugoma viena paskutinė sudėtinio vaizdo eilutė, į BRAM(2) įrašoma nauja.



4 pav. Sudėtinio vaizdo sudarymo struktūrinė diagrama

Fig. 4. Structure diagram of integral image calculation

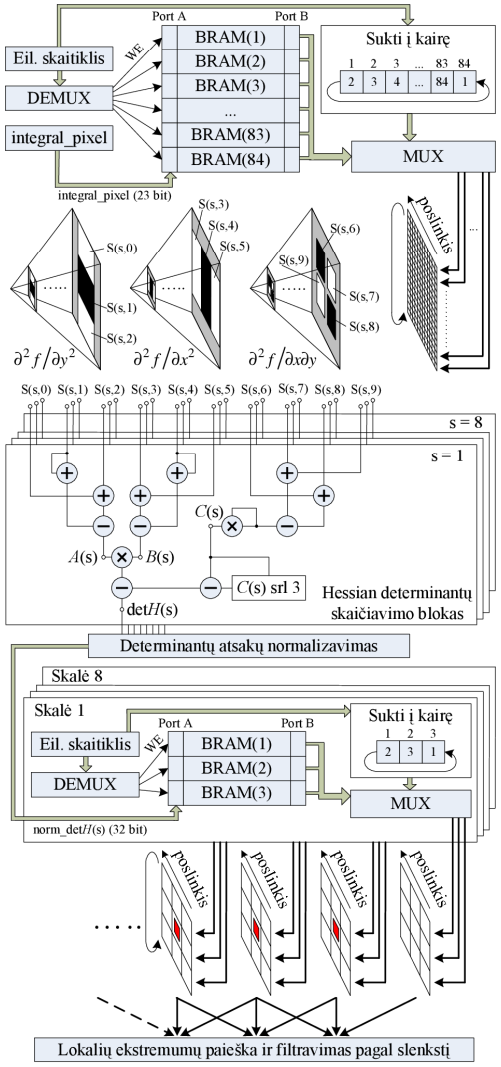
Slenkančiam 84×84 dydžio langui sukurti (5 pav.) taikomos 84 BRAM eilutės sudėtiniam vaizdui saugoti (šis eilučių skaičius būtinas septintosios skalės supaprastintam deskriptoriui skaičiuoti). BRAM atmintys kas 84 eilutes periodiškai perrašomos, todėl saugomi tik realiu laiku apdoroti būtini duomenys taikant ribotą pastovų atminties kiekį. Duomenis slenkančiame lange būtina ne tik stumti į kairę, bet ir per vieną elementą kelti į viršų eilutei pasibaigus. Šiam uždaviniui išspręsti taikomas besisukantis į kairę eilučių indeksų vektorius. Slenkančio lango duomenimis dalijamasi su Hessian determinanto, požymių posūkio kampų ir deskriptorių blokais.

Žymenis S(s, 0..9) nurodo sumą vaizdo taškų patenkančių į stačiakampių ar kvadratų ribojamą plotą. Kiekvieno taktinio signalo metu 80 sumos verčių (10 kiekvienai skalei) perduodama į Hessian matricos determinanto skaičiavimo bloką. Aštuoni kiekvienos skalės determinanto rezultatai siunčiami į determinanto atsako normalizavimo bloką, kuriame kiekvienas aukštesnę už pirmąją skalę atitinkantis rezultatas dauginamas iš skalės koeficiento. Normalizuotieji atsakai taikomi lokalių ekstremumų paieškai ir filtravimui pagal slenkstį.

Statinį slenkstį patogu taikyti, kai žinoma, kad apšvietimas vaizde nekinta. Kitu atveju siūloma taikyti adaptyvų slenkstį, kuris priklausytų nuo vaizde aptiktų požymių ir slenksčio vertės praeitame kadre. Slenksčio vertė sekančiam kardui gali būti išreiškiama lygtimi:

$$TH_{\text{next}} = TH_{\text{prev}} + (N_{\text{prev}} - N_{\text{d}}) \cdot v_{\text{th}}, \qquad (5)$$

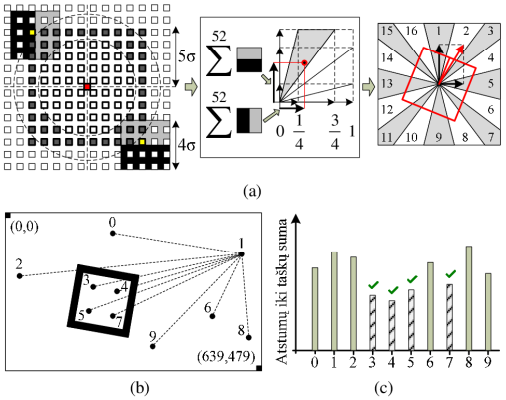
čia TH_{next} – slenksčio vertė sekančiame kadre, TH_{prev} – esamame kadre, N_{prev} – būdingųjų taškų skaičius esamame kadre, N_{d} – ribinis būdingųjų taškų skaičius, v_{th} – adaptyvaus slenksčio prisitaikymo greitis. Tam, kad požymiai, kurių determinanto vertės mažos (taškai išsidėstę išilgai įstrižos linijos ir požymius dubliuojantys atsakai), būtų atmetami, adaptyvusis slenkstis turi apatinę ribą.



5 pav. Slenkančio lango kūrimas 2*D* Gauso filtrams taikyti, determinanto skaičiavimas ir lokalių ekstremumų paieška

Fig. 5. Creation a sliding window for 2*D* Gaussian filters, determinant calculation and local extreme point search

Požymių orientacijai nustatyti aplink būdingąjį tašką atstumu 4σ apskaičiuotos 52 Haar dvimatės vilnelės *x* ir *y* kryptimis sumuojamos sudarant du vienas kitam stačius vektorius (6 pav. a). Taikant bitų postūmio ir sumos operacijas vertinami šių vektorių tarpusavio ilgiai. Požymiui gali būti priskirta viena iš šešiolikos kampų orientacija. Būdingasis taškas gali būti pažymimas kvadrato formos žymekliu, kurių yra po 16 kiekvienai iš 6 skalių. Deskriptorius sudaromas keturioms 5σ×5σ sritims skaičiuojant gradientų bei jų modulių sumas pagal *x* ir *y* koordinates.



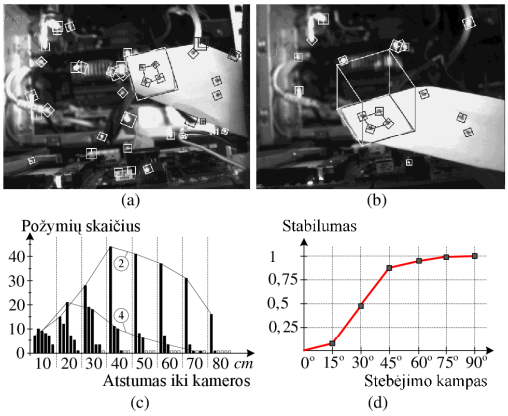
6 pav. Būdingųjų taškų orientacijos skaičiavimo principas (a); vaizde aptikti žymekliui būdingi taškai (b); būdingųjų taškų atstumų iki kitų požymių sumų histograma (c)

Fig. 6. Orientation vector estimation technique (a); marker's points detected in video (b); sum of distance from features to each detected point of interest (c)

Įgyvendintas SURF algoritmas taikomas žymekliui sekti pagal keturių taškų požymius (6 pav., b). Vaizde gali būti aptinkami identiški žymekliui būdingieji taškai. Todėl jų atmetimui skaičiuojamos atstumų iki kitų požymių sumos (6 pav. c). Požymiai, kuriuos atitinka keturios mažiausios vertės sumos, priskiriami žymekliui.

Rezultatai

Požymiai (taškai arba kampai) vaizde išskirti teisingai ir pažymėti kvadratėliais (7 pav., a, b). Iš aptiktų būdingųjų



7 pav. Vaizdo kadruose aptikti būdingieji taškai (a, b); požymių skaičiaus priklausomybė nuo atstumo iki kameros (c); žymeklio sekimo stabilumo priklausomybė nuo stebėjimo kampo (d)

Fig. 7. Detected features in video frames (a, b); features dependance on distance to camera (c); Marker tracking stability dependance on view angle (d)

taškų skaičiaus vaizde priklausomybės nuo atstumo iki kameros (7 pav., c) matoma, kad tolinant kamerą nuo objekto požymiai iš aukštesnių skalių pereina į žemesnes. Iš keturių žymeklio taškų sekimo stabilumo grafiko (7 pav., d) matyti, kad plokštumos stebėjimo kampui esant iki 45° objektas sekamas 80 % stabilumu.

Išvados

1. Būdingieji taškai vaizde aptinkami ir sekami teisingai keičiant posūkio kampą ir atstumą iki jų. Taikant keturių mažiausių atstumų sumų tarp požymių paiešką sekamo žymeklio taškai išskiriami teisingai, kai kampas tarp plokštumos ir kameros ašies kuo artimesnis 90°.

2. Modifikuotas SURF algoritmas sėkmingai įgyvendintas trijose LPLM ir veikia realiu laiku taikant tik 25 MHz taktinį signalą. Algoritmas gali būti greitai pritaikytas didesnės raiškos ir dažnio vaizdui apdoroti.

Literatūra

Bay, H. *et al*. 2008. SURF: Speeded up Robust Features, *Computer Vision and Image Understanding*, 110(3): 346–359.

Svab, J. *et al*. 2009. FPGA-based Speeded up Robust Features, *IEEE International Conference on TePRA*: 35–41.

Bouris, D. *Et al*. 2010. Fast and Efficient FPGA-based Feature Detection Employing the SURF Algorithm, *18th IEEE Annual International Symposium on FCCM*: 3–10.

Fischer, J. *et al*. 2011. A Rotation Invariant Feature Descriptor O-DAISY and its FPGA Implementation, *IEEE International Conference on IROS*: 2365–2370.

Battezzati, N. *et al*. 2012. SURF Algorithm in FPGA: A Novel Architecture for High Demanding Industrial Applications, *DATE Conference & Exhibition*: 161–162.

Sunkara, P. 2011. Line Drawing by using Bresenham's Algorithm, Wu's Algorithm, *Indiana State University*: 1–10.

MODIFIED SURF ALGORITHM IMPLEMENTATION ON FPGA FOR REAL-TIME OBJECT TRACKING

T. Sledevič

Abstract

FPGA-based implementation of the modified speeded-up robust features (SURF) algorithm is described in this paper. FPGA is selected for parallel process implementation using VHDL to ensure features extraction in real-time. A sliding 84×84 size window is used to store integral pixels, accelerate Hessian determinant calculation, orientation assignment and descriptor estimation. The local extreme searching is used to find point of interest in 8 scales. The simplified descriptor and orientation vector are calculated in parallel in 6 scales. The algorithm is investigated by tracking marker and drawing plane or cube. All parts of algorithm works on 25 MHz clock. Video stream is generated by 60 fps and 640×480 pixel camera.

Keywords: field programmable gate array, features extraction, video filtering, object tracking.

B priedas. Straipsnio IEEE 8 regiono studentų straipsnių konkurse medžiaga

FPGA-Based Selected Object Tracking Using LBP, HOG and Motion Detection

Tomyslav Sledevič[#1]

[#]*Department of Electronic Systems, Vilnius Gediminas Technical University, Naugarduko str. 41, LT-03227 Vilnius, Lithuania*

[1]tomyslav.sledevic@el.vgtu.lt

***Abstract*—This paper describes the hardware architecture for selected object tracking on embedded system. The LBP and HOG feature extraction algorithm is combined with motion detection to compute and compare the features vectors with captured once only when the target moves. $LBP_{8,1}$, $LBP_{16,2}$ and $HOG_{8,1}$, $HOG_{16,2}$ are used to create the feature vector. The unit that makes finally decision to update tracker is based on searching of the least *SSD* of features histogram. Motion detection algorithm can find and mark 8 moving object simultaneously. All trackers locations in every further frame are updated by computed in previous. The implementation results show that HOG features based tracker is robust to luminescence variation and partial occlusion. LBP additionally is more robust to rotation. The proposed architecture is implemented on Xilinx Virtex 4 FPGA using VHDL and it achieves real-time desired and other moving object tracking on 60 fps 640×480 video.**

I. Introduction

Real-time FPGA-based object tracking is widely used in traffic monitoring, vehicle navigation, video surveillance, human-computer interaction and etc. There are applied many different algorithm based on feature descriptors, optical flow, template matching or texture operators. In most cases the algorithms implemented on FPGA track all moving objects or only pre-defined object. Accordingly, the observer hasn't opportunity to change the target in real-time and track him.

To implement the "click and track" principle based tracking algorithm we apply histogram of oriented gradient (HOG) and local binary pattern (LBP) as local shape and texture descriptors accordingly. Therefore, let's make a little review of HOG and LBP applications.

Many engineers are concentrated on pedestrian detection on embedded vision system [1-7]. Because of large distance from human to camera it is assumed that human silhouette fit to well known window area [1,3,5,7]. The window is divided in blocks and for each block HOG is computed. Each frame is scanned to looking for pedestrian appearance. The main task is to separate detected objects into human and non human targets. Therefore, a support vector machine is widely used. Because of parallel process implementations there are obtained real-time pedestrian tracking.

Better results can be achieved by combining two features extracting methods. In the literature [4,7] is proposed to use HOG with LBP descriptor simultaneously. The HOG and LBP algorithms are realized on PC to detect humans in image. It is possible to detect the human with partial occlusion using trilinear interpolated HOG with LBP feature set [4]. But the proposed detector can not properly recognize the articulated deformation of people. In work [7] is presented a modified features extractor based on dense center-symmetric LBP and pyramid HOG with the histogram intersection. The result shows that modified combination of HOG and LBP improves detection performance of well known pedestrian dataset.

HOG feature set is also used to detect signs on road [8]. The result shows that FPGA-based stop sign recognition system has high detection rate and low false positive.

Nowadays, the main purpose of LBP operator is the face detection in images. In the paper [9] is presented real-time face detection based on FPGA. There are used a 12 level image pyramid. The LBP is computed in parallel for each level image. The experimental result shows 93% accuracy to detect faces in color image.

II. LBP and HOG Features

LBP features are widely used in rotation and illumination invariant texture classification [10]. The pattern creation process is simple. We look by clockwise to the neighbours of centre pixel and write 1 when neighbour value is more or equal to centre pixel, otherwise write 0. When the pattern have two or zero 0/1 transitions it is called uniform. Only the uniform patterns are collected in the feature vector, as shown in Fig. 1. Because of noise the non uniform pattern are rejected. The $LBP_{8,1}$ and $LBP_{16,2}$ are connected in one vector to increase amount of features and make the classifier stronger. Pattern of all ones are not applied to features vector because it signify the flat area or a darker pixel. Experiments shows that this column in histogram alternates hard during object moves.

HOG features are also collected in 24 bit width, as shown in Fig. 1. This features vector is illumination invariant but not invariant to rotation. HOG comparing to LBP is rated as stronger because it store the amount of gradients in 8 or 16 direction. LBP evaluate the amount of similar neighbours and don't care about its direction. Therefore, when tracked object don't rotate it is better to use HOG. LBP is useful when object texture differs from background.

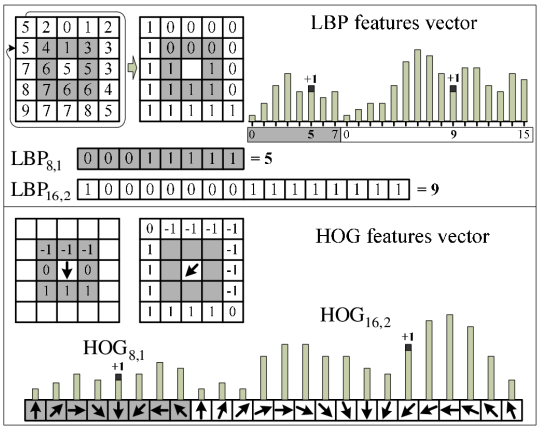


Fig. 1 Construction of 24 bit width LBP and HOG feature vector.

III. Implementation

The proposed algorithm to track moving and selected objects is shown in Fig. 2. From the beginning algorithm automatically detect the movement, compute coordinates of moving objects and compute LBP and HOG features vector for each moving object. The observer select desired target on monitor using mouse and the algorithm immediately computes LBP and HOG vectors to selected area. In the further frames the captured LBP and HOG is compared with each moving. The best matching is finding by computing sum of squared differences (*SSD*) of columns in feature vector according to below equation.

$$SSD_{\text{LBP, HOG}} = \sum_{i=1}^{N} \left(H_d(i) - H_m(i)\right)^2,$$

where H_d and H_m represent the desired and moving object histogram. Number of columns N is equal to 8, 16 or 24.

When the *SSD* is less than threshold then coordinates of rectangular tracker are updated. If no, then old coordinates are used. The experiments show that threshold SSD_{TH} is proportionnal to size S of selected target. The threshold is manually or automatically settable by expression: $SSD_{TH} \approx 25 \times S$.

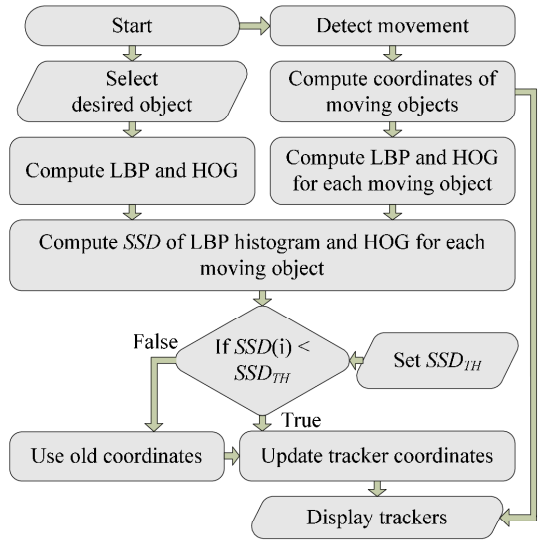


Fig. 2 Algorithm to track moving and selected objects.

A. Estimation of Moving Objects Coordinates

Coordinates of all moving objects are computed in accordance with algorithm, as shown in Fig. 3. To detect movement in video the whole frame is segmented in windows which contains sum of intensities over 4×4 pixel, as shown in Fig. 4. The sum of intensities is stored in two BRAM memories. The step between captured frames is in range from 1 to 16. For fast moved objects the step can be equal to 1. It means that every frame is compared with previous. For object with slow velocity the step equal to 16 is chosen to properly estimate the movement.

At this stage there is additional option to choose step equal to infinity. It means that frame is captured once, stored in BRAM (0) and is not updated until observer will change this option. The next frames are stored in BRAM(1) and constantly

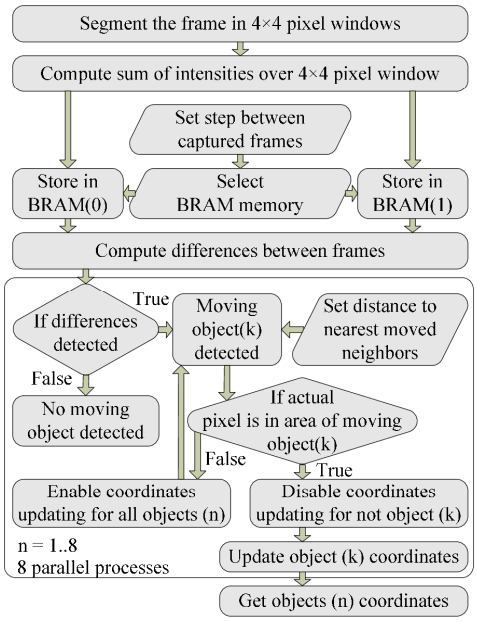


Fig. 3 Algorithm to estimate moving objects coordinates.

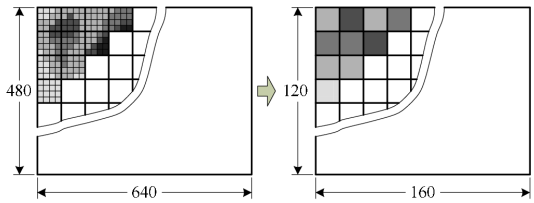


Fig. 4 Frame segmentation principle.

compared with BRAM (0). It work like background extraction and only appeared objects are displayed. This option is useful when objects are tracked in well known and not changeable background, otherwise not useful when camera moves a bit.

If differences between frames are detected the algorithm start to search moving neighbours, as shown in Fig. 5. If second moving pixel is in area of moving object (*k*) then this pixel belongs to object (*k*), else pixel belongs to other already detected object and else new object is detected. To avoid connection of distinct objects the searching distance for nearest moving neighbour must be less or equal to constant. This distance is constantly checked by first and last marginal moving pixel and adjusted when in second line no pixel detected below. This function avoids the undesired linking of pixels which are marked by red ellipse in Fig. 5. To track large object it is suggested to use higher distance to nearest moving neighbour.

Eight parallel processes are implemented on FPGA. It means that eight moving objects can be track simultaneously. Because of parallelism two eight bit flag vectors are created. First, to control coordinates updating for each object. Second, to indicate appearance of object. The border and centre coordinates of all moving objects are known at end of frame. LBP and HOG features in second frame will be calculated at estimated area. Objects moving directions are easy estimated

by computing differences between centre pixels in actual and previous frame.

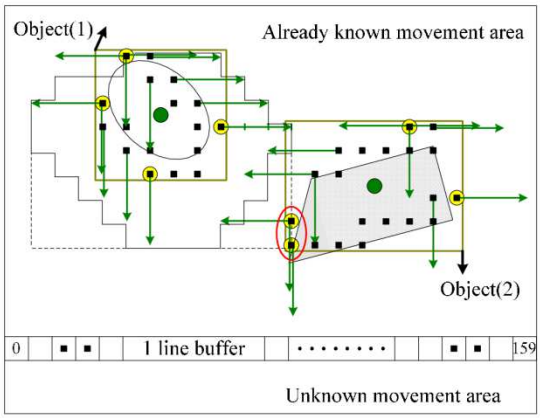


Fig. 5 Moving objects separation principle.

B. Hardware Architecture

The proposed hardware architecture is shown in Fig. 6. Aptina VGA 60 fps image sensor is connected to Virtex-4 FPGA via 10 bit parallel data bus. Incoming frame is written to dual port BRAM which store exactly one frame and rewrite it when new appears. There is created four lines plus five pixels FIFO buffer to realize 5×5 size 4 bit depth intensity matrix. LBP, HOG and movement detection units share this matrix simultaneously. Movement detection unit control the sensibility to movement which depends on BRAM (0/1) switching frequency. The absolute differences between pixels are stored in one line buffer. Using above described algorithm all founded coordinates are passed to trackers coordinate unit and LBP, HOG unit. Last-mentioned collect the captured by mouse click and current frame histograms of all detected moving objects. *SSD* unit calculate sums of squared differences and send result to decision unit that decide on LBP and HOG tracker updating.

The video from BRAM, differences between frames, both histograms, *SSD* values, all moving objects trackers and desired object trackers are displayed on VGA monitor with identical to camera 60 fps refresh rate. All system units are running at 25 MHz clock except camera and port A in BRAM. Synchronization and clock lines are connected to all units to ensure real-time object tracking.

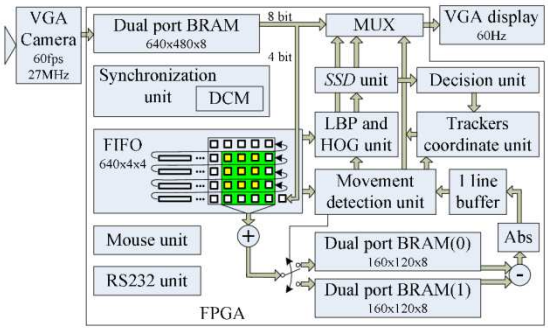


Fig. 6 Proposed hardware architecture.

IV. Experiments and Results

The proposed algorithm with $LBP_{8,1}$, $HOG_{8,1}$, $LBP_{16,2}$, $HOG_{16,2}$ features is applied to traffic surveillance. One car is selected once and tracked until *SSD* is less than threshold or car disappears. The result of partially occluded and forward moving car is shown in Fig. 7 a. The algorithm also detects and tracks another driving car on road and one pedestrian on footpath. These objects are marked in green rectangles. The light blue pixels mark the differences between frames. The red and blue rectangles mark the HOG and LBP features based trackers respectively. In upper left corner the captured and tracked object histograms are constantly displayed.

The result of another selected, partially occluded and backward moving car is shown in Fig. 7 b. In this example the difference between captured frames is intentionally increased to detect slow moving objects. But it makes influence to properly fast car tracking. Delayed tracker effect is observed on right side in Fig. 7 b.

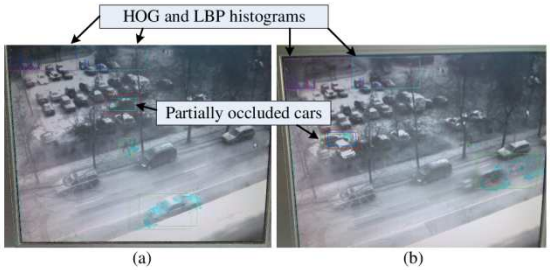


Fig. 7 (a, b) Result of partially occluded cars and others moving object tracking using $LBP_{8,1}$, $HOG_{8,1}$, $LBP_{16,2}$, $HOG_{16,2}$ features.

The algorithm with $LBP_{8,1}$, $HOG_{8,1}$ is also tested on various objects, as shown in Fig. 8. In this case, because of well known searching area the background extraction is applied to track the target. The knitted wired cube is selected once and tracked during frames. Other six objects are marked in green rectangles. Red and light blue pixels mark the differences between frames. The higher luminance of these pixels the higher intensity changes between frames.

The same moved and rotated objects are shown in Fig. 8 b. The selected target is properly tracked by LBP and HOG features.

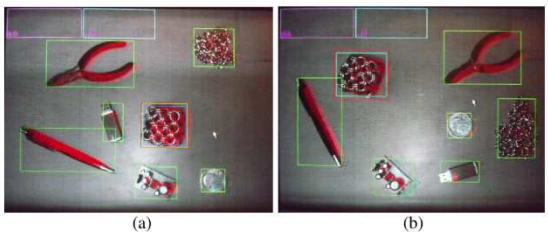


Fig. 8 (a, b) Result of object tracking using $LBP_{8,1}$ and $HOG_{8,1}$ features only. The wired cube is properly tracked by red and blue rectangles.

The successful target tracking depends on correct coordinate's estimation by moving detection algorithm. When target rotates the detected moving area changes a bit. This makes influence to space where LBP and HOG features are

calculated. Additional classification accuracies are influenced by texture [10] and gradient operators' dependence on rotation. As a result this little area and rotation changes gives high *SSD* variation.

The *SSD* dependence on rotation angle of cube is shown in Fig. 9. The points of these performances are collected and averaged over 10 measurements for angle. Because of horizontally and vertically cube symmetry the angles are in range from 0° to 90°. The dashed red line shows that $LBP_{16,2}$ is more rotation invariant than $LBP_{8,1}$ because the first has less average *SSD* value. To ensure continuously target tracking the SSD_{LBP} value must be more than maximum of SSD_{LBP}.

The HOG is non-rotation invariant operator, as shown green lines in Fig. 9. The dashed green line shows that $HOG_{16,2}$ is less rotation invariant than $HOG_{8,1}$. The first can be used as strong feature for non-rotating targets. The $LBP_{8,1}$ and $HOG_{8,1}$ operators collect more false samples than $LBP_{16,2}$ and $HOG_{16,2}$ because the SSD_{LBP} and SSD_{HOG} should be as less and as high as possible respectively.

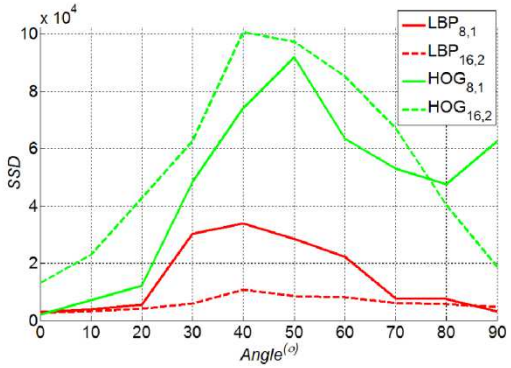


Fig. 9 Performance of cube rotation angle.

The result of target tracking by using $LBP_{8,1} + LBP_{16,2}$, $HOG_{8,1} + HOG_{16,2}$ is shown in Fig. 10. The *SSD* values are collected through the path of moving object. The highest *SSD* values are sampled at corners of target path. It means that threshold for SSD_{LBP} must be $5 \cdot 10^4$ and $2 \cdot 10^5$ for SSD_{HOG}. The best matching appears when object rotates ± 20° from initial angle. When it is known that our target rotates in this range it is useful to decrease the threshold which makes tracker robust.

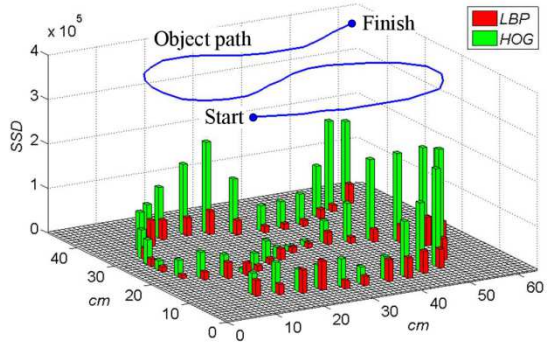


Fig. 10 Result of cube tracking using $LBP_{8,1} + LBP_{16,2}$, $HOG_{8,1} + HOG_{16,2}$ features. Initial LBP and HOG are captured when target is in frame centre.

TABLE I
FPGA OCCUPATION

Resource Type	Utilization Rate
Number of Slices	15358 of 15360 (99%)
Number of 4 input LUTs	27588 of 30720 (89%)
Number of BRAMs	170 of 192 (88%)
Number of DSP48s	192 of 192 (100%)

The FPGA occupation summary is shown in Table 1. All DSP units are used to real time *SSD* computation. All calculations are running on 25 MHz clock that decrease power consumption.

V. CONCLUSIONS

In this paper we propose hardware architecture for LBP and HOG features extraction aiming real-time selected object tracking on embedded system. The LBP and HOG features are computed in area where the motion is detected. The experimental results of selected cars tracking show that algorithm correctly track partially occluded cars and additionally track other cars and pedestrians. The results of selected cube tracking show that target is properly detected and follow despite the rotation. The algorithm is implemented on a Xilinx Virtex 4 FPGA using VHDL to ensure real-time object tracking at 60 fps and 25 MHz clock.

ACKNOWLEDGMENT

The author wishes to acknowledge dr. Artūras Serackis and Darius Plonis for supervising the work presented in this paper.

REFERENCES

[1] M. Hiromoto and R. Miyamoto, "Hardware architecture for high-accuracy real-time pedestrian detection with CoHOG features," *IEEE 12th International Conference on Computer Vision Workshops*, pp.894-899, Sept. 2009.

[2] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, Y. Nakamura, "Hardware Architecture for HOG Feature Extraction," *Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp.1330-1333, Sept. 2009.

[3] S. Bauer, S. Kohler, K. Doll and U. Brunsmann, "FPGA-GPU architecture for kernel SVM pedestrian detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 61-68, June 2010.

[4] X. Wang,; T. X. Han and S. Yan, "An HOG-LBP human detector with partial occlusion handling," *IEEE 12th International Conference on Computer Vision*, pp.32-39, Sept. 2009.

[5] T. Wilson, M. Glatz and M. Hodlmoser, "Pedestrian detection implemented on a fixed-point parallel architecture," *IEEE 13th International Symposium on Consumer Electronics*, pp.47-51, May 2009.

[6] S. Martelli, D. Tosato, M. Cristani and V. Murino, "FPGA-based pedestrian detection using array of covariance features," *Fifth ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 1-6, Aug. 2011.

[7] Y. Zheng, C. Shen, and X. Huang, "Pedestrian detection using centersymmetric local binary patterns," *Proceedings of the International Conference on Image Processing*, pp. 3497-3500, Sept. 2010.

[8] P. C. Tam and D. Guang, "Real-Time Vision-Based Stop Sign Detection System on FPGA," *Digital Image Computing Techniques and Applications*, pp.465-471, Dec. 2008.

[9] J. Seunghun, K. Dongkyun, T.T. Nguyen, J. Bongjin, K. Daijin, W. J. Jae, "An FPGA-based Parallel Hardware Architecture for Real-Time Face Detection Using a Face Certainty Map," *20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp.61-66, July 2009.

[10] Ojala, T.; Pietikainen, M.; Maenpaa, T., "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.24, no.7, pp.971-987, Jul 2002.