VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

Vizualaus objekto sekimas realiu laiku pasinaudojant grafiniu procesoriumi

(Real Time Visual Object Tracking using a Graphical Processor)

Atliko: 4 kurso, 1 grupės studentas
Neimantas Kosovskis (parašas)

Darbo vadovas:
Lekt. Mindaugas Eglinskas (parašas)

Recenzentas:
Doc. dr. Antanas Lenkevičius (parašas)

Vilnius
2016

Turinys

Įvadas 4
1. Optinis srautas 5
1.1. Optinio srauto apibrėžimas 5
1.2. Bazinė optinio srauto lygtis 6
1.2.1. Bazinės optinio srauto lygties apibrėžimas 6
1.2.2. Pritaikymas matricoje 7
1.3. Klasikinis Lucas – Kanade metodas 9
1.3.1. Lucas – Kanade metodo apibrėžimas 9
1.3.2. Pritaikymas matricoje 11
1.4. Piramidinis Lucas – Kanade metodas 12
2. Objekto sekimas pasinaudojant optiniu srautu 14
2.1. Objekto sekimo apibrėžimas 14
2.2. Elementarus sekamo objekto judėjimas 15
2.2.1. Judėjimas viena kryptimi 15
2.2.2. Judėjimas artyn ir tolyn nuo žiūrovo 16
2.2.3. Objekto uždengimas 17
2.3. Sekamo objekto tranzitas skirtingo atspalvio fonais 18
2.3.1. Aplinkos įtaką sekamiems taškams 18
2.3.2. Nagrinėjamo ploto įtaka sekamiems taškams 19
2.4. Objekto sekimas atnaujinant sekamo objekto taškus 21
2.4.1. Nuolatinis naujų sekamų taškų didinimas 21
2.4.2. Naujų taškų priskyrimas pašalinus senus 22
2.5. Objekto sekimas judant stebėtojui 23
3. Lucas – Kanade metodo paralelizacija 24
3.1. Metodo realizacija ant grafinio procesoriaus 24
3.2. Algoritmo skaidymas į dalis 25
3.3. Algoritmo greičių palyginimas 26
4. Metodo palyginimas su OpenCV objektų sekimui skirta realizacija 29

4.1. OpenCV realizacija objektų sekimui 29
4.2. Pirmasis palyginimas 29
4.2.1. Objektų sekimas realizuotu metodu 29
4.2.2. Objektų sekimas su OpenCV 31
4.3. Antrasis palyginimas 32
4.3.1. Objektų sekimas realizuotu metodu 32
4.3.2. Objektų sekimas su OpenCV 33
Išvados ir rezultatai 35
Summary 36
Literatūros sąrašas 37

Įvadas

Kompiuterinė rega yra viena iš technologijų, kuri metams bėgant tampa vis populiaresnė. Technologijos populiarumą nulėmė plačios jos panaudojimo galimybės, o išpopuliarinti leido kompiuterinės įrangos tobulėjimas. Kompiuterinė rega yra naudojama pramonėje, robotikoje, viešosios tvarkos užtikrinimui, pavyzdžiui, eismo registratoriuose ar apsaugos sistemose, kariniais tikslais. Pastaruoju metu, ypač atsiradus galingiems mobiliems įrenginiams, kompiuterinė rega pradėta naudoti ir kasdieniniame gyvenime. Praturtintosios realybės (angl. *augmented reality*) sistemos tampa viena iš pagrindinių pramogų neišeinant iš namų. Tokiu atveju kompiuterinė rega turi fiksuoti ir apdoroti stebimą vaizdą realiu laiku. Nors kompiuterinė įranga yra pakankamai galinga, visgi vaizdo apdorojimas realiu laiku išlieka sudėtingas uždavinys, reikalaujantis didelių resursų ir skatinantis optimalių algoritmų ir greitų metodų ieškojimą.

Vienas iš kompiuterinėje regoje naudojamų metodų yra optinio srauto skaičiavimas. Optinis srautas leidžia aptikti objektų judėjimą erdvėje ir toliau iš gautų duomenų įgyvendinti judesio detektorius, objektų atpažinimą ir sekimą, navigaciją ar panaudoti odometrijai. Vizualių objektų sekimas yra vienas iš svarbiausių kompiuterinės regos uždavinių, kuris ir bus aptartas šiame darbe.

Darbo tikslas - realizuoti objekto sekimą, pasinaudojant optiniu srautu.

Darbo uždaviniai:

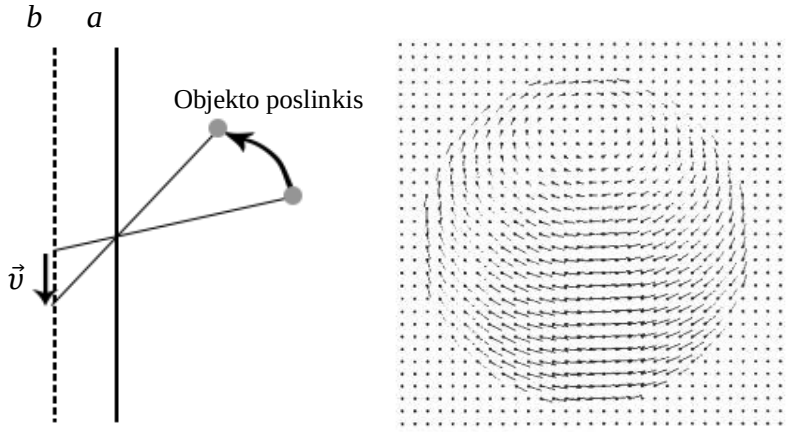
1. Realizuoti objektų sekimo programą pasinaudojant optinio srauto algoritmu ir išanalizuoti sekimo savybes;
2. Rasti ir realizuoti metodus pagerinančius turimo algoritmo veikimą;
3. Palyginti gautus rezultatus su OpenCV siūloma, objektų sekimui skirta, realizacija.

Optiniam srautui apskaičiuoti pasitelktas Lucas – Kanade metodas piramidėms, skaičiavimams pagreitinti algoritmas perkeltas ant grafinio procesoriaus.

1. Optinis srautas

1.1. Optinio srauto apibrėžimas

Optinis srautas, tai akivaizdžių vaizdo ryškumo pokyčių, bei tų pokyčių intensyvumo pasiskirstymas ir nustatymas vaizdo kadre (žr. 1 pav.). Optinis srautas atsiranda esant judėjimui tarp stebimo vaizdo ir žiūrovo. Kompiuterinėje regoje optinis srautas gali suteikti daug naudingos informacijos apie stebimų objektų išsidėstymą erdvėje, bei esamo išsidėstymo pokyčius bėgant laikui. Dažniausiai šie pokyčiai yra randami skaičiuojant skirtumą tarp dviejų vaizdo kadrų. Įprastu atveju, optinis srautas yra atvaizduojamas greičio vektoriumi, nusakančiu, kuria kryptimi objektas pajūdėjo ir kokiu intensyvumu. [HS81]



1 pav. Optinio srauto vektorius ir pasiskirstymo žemėlapis

Šaltinis: „Determining optical flow“, 1981, p. 200

Realiame pasaulyje judėjimas vyksta trimatėje erdvėje, stebimas objektas gali pasislinkti, atitolti, priartėti, pasisukti, deformuotis, gauti skirtingą apšvietimą ir t.t. Visi šie pokyčiai erdvėje yra sudėtingi ir kartais optinio srauto apskaičiuoti nepavyksta, ypač stebint vaizdą pro stacionarų ekraną. Kartais stebimo objekto judėjimas nebūna toks akivaizdus kaip galima tikėtis. Pavyzdžiui, erdvėje matoma vienspalvė sfera, į ją nukreiptas vienas šviesos šaltinis, kuris iš vienos sferos pusės atspindi, o iš kitos sukuria šešėlį. Pasukus sferą aplink savo ašį, o aplinką palikus statinę, optinis srautas bus lygus nuliui, kadangi lygus vienspalvis sferos paviršius nekeičia atspindžio kampo. Tokiu atveju, nors sfera iš tikrųjų juda, jos judėjimo greičio nustatyti beveik neįmanoma. Visai kitas atvejis būtų jei besisukanti sfera kartu su savimi suktų ją apšviečiantį šviesos šaltinį. Stebint sferą aiškiai matytųsi šviesos intensyvumo pokyčiai ant jos paviršiaus. Svarbiausia, kad jos judėjimui

nustatyti net nėra būtina trimatė erdvė, todėl dažniausiai skaičiuojant optinį srautą, erdvė yra supaprastinama iki dvimatės ir visas stebimas vaizdas yra laikomas plokščiu, taip pat yra įvedamos papildomos sąlygos (apribojimai), kurios būtinos optiniam srautui apskaičiuoti. [PU13]

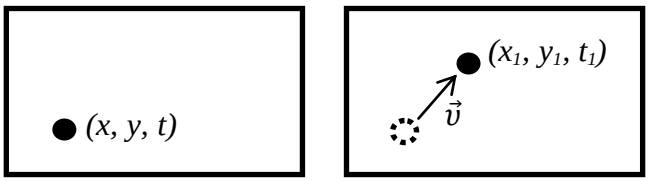
Kompiuterijoje apskaičiuoti optinio srauto vektoriai yra saugomi į masyvą ir demonstravimo tikslais gali būti atvaizduojami kaip papildomas sluoksnis vaizdo sraute. Paveikslėlyje (žr. 1 pav.) linija *a* nusako kadrą iš vaizdo srauto, o linija *b* yra ant pradinio vaizdo kadro uždėtas papildomas sluoksnis, kuriame yra piešiami optinio srauto vektoriai. Papildomas sluoksnis yra plokščias ir nusako vaizdo ryškumo pokyčių pasiskirstymą kadre (žr. 1 pav.). Toliau su gautais duomenimis yra atliekami papildomi skaičiavimai ir optinio srauto pagalba yra realizuojamos kompiuterinės regos funkcijos, tokios kaip objektų sekimas, atpažinimas ir pan.

1.2. Bazinė optinio srauto lygtis

1.2.1. Bazinės optinio srauto lygties apibrėžimas

Optiniam srautui rasti yra naudojami diferencialiniai metodai pagrįsti Teiloro eilute. Šie metodai skaičiuoja kiekvieno pikselio judėjimą tarp dviejų kadrų paimtų iš vaizdo srauto laiko momentu t ir t_1. Vaizdo srautas kinta laike, todėl skaičiavimuose bus naudojami laiko rodikliai. Tiksliau, dirbant su visu vaizdo srautu, o ne pavieniais kadrais, vien dvimatės erdvės nepakanka ir nagrinėjama sistema turi būti papildyta. Dabar nagrinėjama sistema yra *2D + t*, kitaip sakant skaičiavimai naudojant dalines išvestines bus atliekami atsižvelgiant į laiko ir dvimatės erdvės koordinates. [HS81]

Pikselis, kurio koordinatės yra *(x, y, t)*, o jo ryškumą nusako funkcija *I(x, y, t)*, pajuda iš vieno kadro į kitą per $\Delta x, \Delta y, \Delta t$. Čia $\Delta x = x_1 - x$, $\Delta y = y_1 - y$, $\Delta t = t_1 - t$.



2 pav. Optinio srauto vektorius

Pajudėjęs pikselis yra vis dar tas pats pikselis, tik kitoje kadro vietoje, todėl galima daryti prielaidą, kad bent jau trumpą laiką judančio pikselio ryškumas nekinta. Realioje situacijoje tai

nebūtinai yra tiesa, tačiau norint apskaičiuoti optinį srautą, reikia supaprastinti esamą situaciją įsivedant šį apribojimą. Esant tokiai sąlygai galima sudaryti tokią lygybę:

$$I(x, y, t) = I(x+\Delta x, y+\Delta y, t+\Delta t);$$

Įvedus dar vieną apribojimą, jog pikselio poslinkis yra mažas, lygybę galima praplėsti pasinaudojus Teiloro eilute:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \cdots ;$$

Iš turimų dviejų lygybių galima teigti, kad:

$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \cdots = 0$; Daugtaškiu yra žymimos aukštesnio laipsnio išraiškos, kurios yra pakankamai mažos, tad galima jas ignoruoti. Gauname:

$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0$; Lygtis yra suprastinama padalinus abi puses iš laiko pokyčio.

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0;$$

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0;$$

Gautoje lygtyje $V_x = \frac{\Delta x}{\Delta t}$ ir $V_y = \frac{\Delta y}{\Delta t}$ nusako pikselio judėjimo greitį *x* ir *y* ašių atžvilgiu. $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ ir $\frac{\partial I}{\partial t}$ lygčių išvestinės pikseliui esant koordinatėse *(x, y, t)*, jas patogu užrašyti atitinkamai I_x, I_y, I_t.

$$I_x V_x + I_y V_y = -I_t;$$

Galiausiai atliekami paskutiniai pakeitimai, jie nėra būtini, bet reiškinys atrodys aiškiau:

$$(I_x, I_y) \cdot (V_x, V_y) = -I_t;$$

$$\nabla I \cdot \vec{v} = -I_t;$$

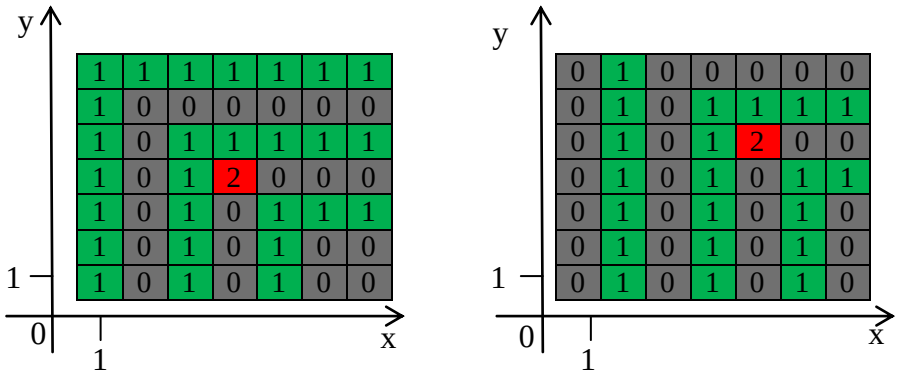
Galutinėje lygtyje turima $\nabla I = (I_x, I_y)$, kas yra erdvinis ryškumo gradientas, ir $\vec{v} = (V_x, V_y)$, kas yra pikselio greitis arba optinio srauto vektorius pikselyje *(x, y)* laiko momentu *t* (žr. 2 pav.). Erdvinis ryškumo gradientas yra gaunamas pritaikant vaizdui tam tikrus filtrus. Vienas iš galimų filtrų yra Sobelio filtras. I_x – gradientas x kryptimi, I_y – gradientas y kryptimi.

Sudarius lygtį optinio srauto paskaičiuoti vis dar negalima, nes lygtyje yra du nežinomieji V_x ir V_y.

1.2.2. Pritaikymas matricoje

Sudarytą optinio srauto lygtį dabar galima plačiau išnagrinėti pritaikius ją konkrečiam pavyzdžiui. Darbas su gradientais padės labiau įsigilinti į tai kas iš tikrųjų yra daroma su vaizdo kadrais. Paimamas pavyzdys su matricomis.

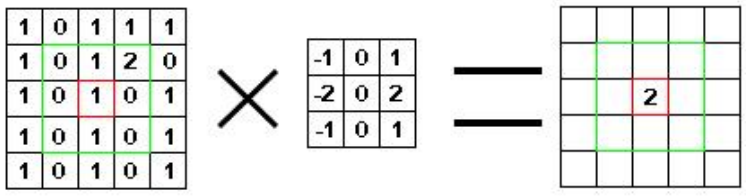
Paveikslėlyje žemiau (žr. 3 pav.) yra pavaizduotos dvi matricos, matrica iš kairės atvaizduoja pradinį vaizdo kadrą, o matrica dešinėje atvaizduoja sekantį kadrą su įvykusiu poslinkiu. Matricos elementai laikomi pikseliais, o jų reikšmės nusako pikselio ryškumą, taigi šiuo atveju funkcija $I(x, y, t)$ gražins matricos elemento reikšmę pagal *x* ir *y* koordinates. Kintamasis *t* leis identifikuoti su kurio kadru yra dirbama, kad būtų aiškiau, nusakoma, jog kairysis kadras buvo paimtas pradiniu laiko momentu t_0, o dešinysis t_1.



3 pav. Vaizdo kadrai laiko momentu t_0 ir t_1

Esamame pavyzdyje nagrinėjamas pikselis yra pažymėtas raudona spalva ir jo ryškumas yra 2 $(I(3, 3, t_0) = 2)$. Pradžioje nagrinėjamam pikseliui pritaikoma bazine optinio srauto lygtis $I_x V_x + I_y V_y = -I_t$.

Pirmame žingsnyje yra paskaičiuojamos I_x ir I_y reikšmės. I_x yra gradientas x kryptimi, o I_y – y kryptimi, gradientai gaunami vaizdo kadrui laiko momentu t_1 atitinkamai pritaikius $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ ir $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$ filtrus. Filtrų taikymo pavyzdys parodytas paveikslėlyje (žr. 4 pav.).



4 pav. Filtro pritaikymas matricai

Pritaikius filtrą *x* kryptimi nagrinėjamam pikseliui, gauta, jog pikselio I_x gradiente reikšmė yra lygi 2. Tą patį atlikus su filtru *y* kryptimi $I_y = 0$ (žr. 4 pav.).

Sekančiame žingsnyje apskaičiuojama I_t iš dešiniosios matricos atėmę kairiąją. I_t yra skirtumas tarp kadrų kiekvienu laiko momentu. Skirtumas nagrinėjamo pikselio koordinatėse yra -1.

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

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

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

5 pav. Apskaičiuoti gradientai I_x, I_y ir I_t

Apskaičiavus visus kintamuosius (žr. 5 pav.), belieka įstatyti į lygtį ir rasti nežinomuosius:

$$2 * V_x + 0 * V_y = -(-1);$$

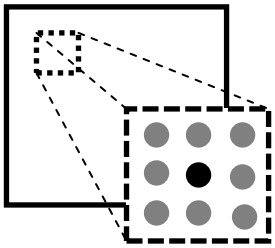
$$2V_y = 1;$$

Iš gauto rezultato matyti, jog nežinomųjų rasti neįmanoma, lygtis yra viena, o nežinomųjų du. Norint lygtį išspręsti, reikia įsivesti dar keletą papildomų apribojimų ir sudaryti papildomas lygtis, tai bus atlikta pasinaudojus Lucas – Kanade metodu.

1.3. Klasikinis Lucas – Kanade metodas

1.3.1. Lucas – Kanade metodo apibrėžimas

Lucas – Kanade algoritmas yra plačiai naudojamas diferencialinis metodas optiniam srautui paskaičiuoti, sukurtas Bruce D. Lucas ir Takeo Kanade 1980 metais. Šis algoritmas, ankščiau aprašyto bazinio optinio srauto lygties neapibrėžtumą, išsprendžia apjungiant informaciją iš keleto gretimų pikselių, esančių šalia nagrinėjamo pikselio, darant prielaidą, jog srautas aplink jį yra pastovus. Šiuo metodu yra neblogai išsprendžiama vaizdo srauto triukšmo problema, tačiau metodas yra grynai lokalus, apimantis tik nedidelį plotą aplink nagrinėjamą pikselį, todėl algoritmas negali įvertinti optinio srauto jei tame plote esantys pikseliai yra vienodi, jų ryškis vienodas. [PU13]



6 pav. Nagrinėjamas plotas

Naudojant Lucas – Kanade algoritmą, daroma prielaida, jog pokytis, nagrinėjamame plote, tarp artimų vaizdo srauto kadrų yra mažas ir daugmaž pastovus, todėl optinio srauto lygtis gali būti pritaikyta kiekvienam pikseliui esančiam tame plote. Iš to gaunama, kad vektorius $\vec{v} = (V_x, V_y)$ turi tenkinti šią lygtį:

$$\begin{aligned} I_x(p_1)V_x + I_y(p_1)V_y &= -I_t(p_1) \\ I_x(p_2)V_x + I_y(p_2)V_y &= -I_t(p_2) \\ &\vdots \\ I_x(p_n)V_x + I_y(p_n)V_y &= -I_t(p_n) \end{aligned}$$

Lygtyje p_1, p_2, ..., p_n yra pikseliai nagrinėjamame plote, atitinkamai dalinės išvestinės funkcijos I skaičiuojamos kiekvienam pikseliui p_i laiko momentu t. Visą reiškinį galima perrašyti matricine forma $Av = b$:

$$A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, \; v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \; b = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{bmatrix};$$

Dabar yra pakankamai lygčių, kad rasti nežinomuosius. Lucas – Kanade metodas sprendinį gauna naudojantis mažiausių kvadratų principu, todėl reiškinys gali būti pertvarkomas iki tinkamos išraiškos:

$$\begin{aligned} A^T A v &= A^T b; \\ v &= (A^T A)^{-1} A^T b; \end{aligned}$$

Pilna išraiška atrodo taip:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} I_x(p_i)^2 & \sum_{i=1}^{n} I_x(p_i) I_y(p_i) \\ \sum_{i=1}^{n} I_y(p_i) I_x(p_i) & \sum_{i=1}^{n} I_y(p_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{i=1}^{n} I_x(p_i) I_t(p_i) \\ -\sum_{i=1}^{n} I_y(p_i) I_t(p_i) \end{bmatrix};$$

Gautas reiškinys yra paprasta klasikinė Lucas – Kanade metodo forma. Ši forma dar nėra plačiai praktikoje naudojama, nes nėra pakankamai tiksli ir kadrų pokyčiai turi būti labai maži, maždaug iki kelių pikselių.

1.3.2. Pritaikymas matricoje

Šiame skyrelyje yra pratęsiami 1.2.2 skyriuje atliekami skaičiavimai ir pasinaudojus išnagrinėtu algoritmu, bus bandoma rasti optinio srauto vektorių.

Pasak algoritmo, reikia išsiskirti nagrinėjamą plotą, šiuo atveju tai bus *3x3* dydžio plotelis, kurio centras yra nagrinėjamas pikselis koordinatėse *(3,3)*. Gaunama 9 lygčių sistema:

$$I_x(p_1)V_x + I_y(p_1)V_y = -I_t(p_1)$$
$$I_x(p_2)V_x + I_y(p_2)V_y = -I_t(p_2)$$
$$\vdots$$
$$I_x(p_9)V_x + I_y(p_9)V_y = -I_t(p_9)$$

p_3	p_4	p_5
p_2	p_9	p_6
p_1	p_8	p_7

Kiekvienam pikseliui yra paskaičiuojamos I_x, I_y, I_t reikšmės ir įstatomos į lygčių sistemą:

$$2V_y = 1$$
$$2V_y = 1$$
$$2V_y = 1$$
$$-V_x - 1V_y = 0$$
$$-2V_y = -1$$
$$V_x - V_y = 0$$
$$2V_y = 1$$
$$-2V_x = -1$$
$$2V_x = 1$$

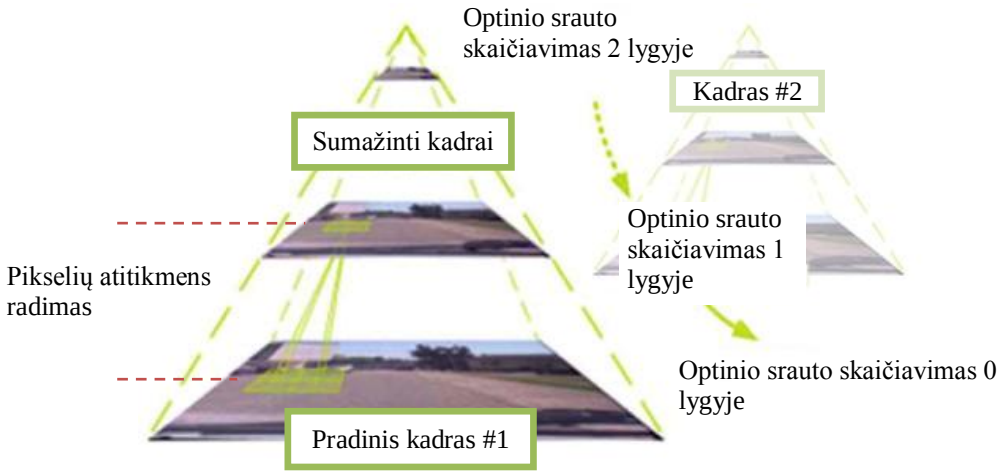
Sistemos sprendinys yra $V_x = \frac{1}{2}$ ir $V_y = \frac{1}{2}$. Iš anksčiau žinoma, kad $V_x = \frac{\Delta x}{\Delta t}$ ir $V_y = \frac{\Delta y}{\Delta t}$. Daroma prielaida, jog vaizdo kadrai yra filmuojami 30 kadrų per sekundę, tai $\Delta t = 60 / 30 = 2$, taigi galime rasti tikslų pikselio poslinkį:

$$\frac{\Delta x}{\Delta t} = \frac{1}{2}\ ;\ \frac{\Delta y}{\Delta t} = \frac{1}{2};$$
$$\Delta x = 1;\ \Delta y = 1;$$

Pasinaudojus Lucas – Kanade algoritmu, rastas optinio srauto vektorius *(1,1)*. Įvykus poslinkiui, nagrinėjamas pikselis pasislinko į naują poziciją, kurios koordinatės yra *(4,4)*.

1.4. Piramidinis Lucas – Kanade metodas

Klasikinis Lucas – Kanade metodas optinį srautą gali apskaičiuoti tik esant mažam stebimo objekto poslinkiui erdvėje, tik keliems pikseliams. Praktikoje, esant realioms situacijoms, metodas gali būti pritaikytas labai ribotai, todėl kompiuterinės regos poreikiams didėjant Jean-Yves Bouguet 2001 metais pristatė patobulintą Lucas – Kanade algoritmo versiją. Nauja algoritmo modifikacija optinį srautą skaičiuoja kelis kartus tam pačiam, tik sumažintam, vaizdo kadrui. Originalus vaizdo kadras yra sumažinamas keletą kartų ir suskaidomas į sluoksnius nuo mažiausio iki didžiausio, sluoksniai sudaro piramidę, iš čia ir kilo pavadinimas piramidinis Lucas – Kanade algorimtas. [JYA81]



7 pav. Gauso piramidės dviem vaizdo kadrams

Šaltinis: „Real-Time Dense and Accurate Parallel Optical Flow using CUDA“, 2009, p. 107

Piramidinis algoritmo įgyvendinimas nuo šiol leidžia atsekti didesnius objekto poslinkius erdvėje. Didžiausi objektų poslinkiai yra nustatomi mažiausioje kadro rezoliucijoje, o originalaus dydžio vaizdo kadre yra nustatomi smulkiausi ir tiksliausi. Algoritmo darbo pradžioje yra sukuriamos Gauso piramidės kiekvienam nagrinėjamam kadrui. (žr. 7 pav.) Piramidės susideda iš N lygių, nuliniame lygyje yra originalaus dydžio vaizdo kadras, sekantis lygis yra sukuriamas prieš tai buvusį kadrą sumažinus dvigubai ir taip, kol pasiekiamas aukščiausias lygis (N-1 lygis sukuriamas N-2 lygį sumažinus koeficientu 2). Lygių skaičius priklauso nuo pradinio vaizdo kadro dydžio, kuo didesnis kadras, tuo daugiau lygių galima panaudoti. Pavyzdžiui, 640 x 480 pikselių kadro dydžiui užtenka 3 arba 4 lygių, daugiau lygių dažniausiai naudoti neverta, nes paskutiniame lygyje kadro dydis bus pernelyg mažas. Toliau algoritmas skaičiavimus pradeda nuo mažiausios rezoliucijos kadro, tai yra nuo aukščiausio lygio. Algoritmas taiko klasikinį Lucas – Kanade metodą optiniam

srautui rasti. Gauti duomenys yra laikomi kaip pradiniai ir toliau yra tikslinami leidžiantis vienu lygiu žemiau, taip kiekvienam lygiui yra skaičiuojamas optinis srautas, pasinaudojant prieš tai buvusio lygio gautais duomenimis, kol yra pasiekiamas paskutinis, nulinis lygis. [JYA81]

Skaičiuojant optinį srautą dviem vaizdo kadrams, skaičiavimų reikia atlikti tiek kartų daugiau, kiek piramidėje yra lygių, kad nesusipainioti klasikinis Lucas – Kanade metodas yra papildomas papildomais žymėjimais. Sudarant piramides yra svarbu apsibrėžti kadro dydį, kuris bus žymimas *w x h*, *w* - plotis, *h* - aukštis. *L* nurodys einama piramidės lygį (*L* = 0,1,2...,*N*-1). Pirmajame lygmenyje, kai *L* = 0 optinio srauto lygtis atrodys taip:

$$I_x^0(p_n)V_x + I_y^0(p_n)V_y = -I_t^0(p_n);$$

Dabar pažiūrėjus į lygtį yra aišku su kokiu lygiu yra dirbama ir kokios rezoliucijos kadrui yra skaičiuojami ryškumo gradientai I_x ir I_y . Pradinis kadro dydis *w x h* yra tas dydis, kuris yra paimamas tiesiai iš vaizdo srauto dar neatlikus jokių transformacijų, todėl pirmajame lygmenyje jis bus žymimas taip $w = w^0$ ir $h = h^0$. Toliau yra transformuojamas kadras ir yra pereinama į sekantį piramidės lygį. Iš I_x^0, I_y^0 ir I_t^0 yra padaroma I_x^1, I_y^1 ir I_t^1 ir taip toliau iki galutinio lygio. Bendra išraiška atrodo taip I_x^N, I_y^N ir I_t^N. Optinio srauto lygtis:

$$I_x^L(p_n)V_x + I_y^L(p_n)V_y = -I_t^L(p_n);$$

Atitinkamai žymimi ir kadro dydžiai w^L *x* h^L. Kartais mažinant kadrą, jo dydis ne visada pasidalina be liekanos, todėl, kad išvengti nepilnų pikselių piramidinis Lucas – Kanade metodas numato sąlygą, jog kadro dydis w^L *x* h^L turi būti didžiausi sveikieji skaičiai tenkinantys šias nelygybes:

$$w^L \leq \frac{w^{L-1}+1}{2};$$

$$h^L \leq \frac{h^{L-1}+1}{2};$$

Pasiekus galutinį lygį, piramidės statyba yra baigta ir dabar, atvirkštine eiga, galima pradėti skaičiuoti optinį srautą. Radus optinio srauto vektorių aukščiausiame piramidės lygmenyje, tai yra mažiausią rezoliuciją turinčiame paveikslėlyje, reikia pereiti prie žemesnio lygio. Pereinant į kitą lygį yra svarbu nepamesti apskaičiuoto vektoriaus koordinačių ir jau padidintame paveikslėlyje rasti jo atitikmenį. Tam yra skirta formulė, kuri leidžia gauti to paties pikselio ryškumą padidintame kadre. [Bou11]

$$I^L(x,y) = \frac{1}{4}I^{L-1}(2x,2y) + \frac{1}{8}\big(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)\big) + \frac{1}{16}\big(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)\big);$$

2. Objekto sekimas pasinaudojant optiniu srautu

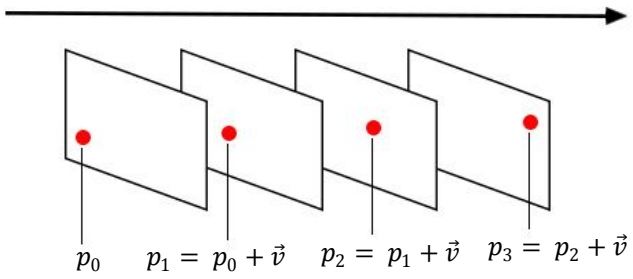
2.1. Objekto sekimo apibrėžimas

Optinis srautas leidžia aptikti objekto poslinkį ir optinio srauto vektoriumi apibrėžti poslinkio kryptį ir atstumą. Turint šiuos duomenis galima rasti naują objekto poziciją, o taikant optinio srauto skaičiavimus kiekvienam vaizdo kadrui, sekti objekto judėjimą aplinkoje.

Objekto sekimas pasinaudojant optiniu srautu gali būti apibrėžtas tokia lygtimi:

$$p_1 = p_0 + \overrightarrow{v};$$

Lygtyje žymuo p_1 reiškia naujas sekamo pikselio koordinates (x_1, y_1, t_1) sekančiame vaizdo kadre laiko momentu t_1. Tuo tarpu p_0 nurodo pradines pikselio koordinates pradiniu laiko momentu t_0. Taigi norint gauti naujas pikselio koordinates, prie pradinių koordinačių reikia pridėti rastą optinio srauto vektorių, kuris lygtyje pažymėtas $\vec{v}$. Šis procesas yra kartojamas viso objekto sekimo metu, tik pradines objekto koordinates keičiant naujomis, veiksmų seką galima pamatyti paveikslėlyje žemiau (žr. 8 pav.). [Bou11]



8 pav. Taško sekimas

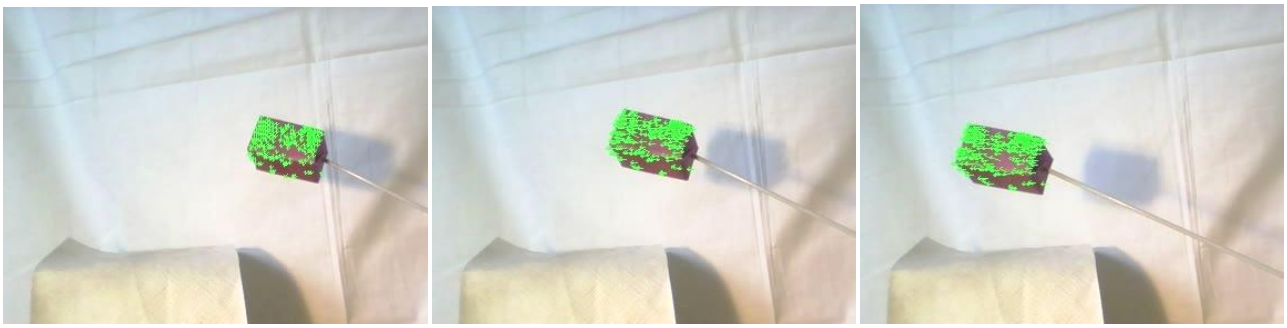
Objekto sekimui išanalizuoti buvo atlikta keletą eksperimentų, kurie leis nustatyti su kokiomis realiomis problemomis yra susiduriama. Eksperimentai buvo atlikti kompiuterine įranga turinčia Intel Pentium vieno branduolio 1,8 Ghz procesorių ir NVIDIA Geforce 720M vaizdo plokštę. Grafinio procesoriaus sparta yra 625 Mhz, procesoriuje yra integruoti 96 CUDA branduoliai. Objektų sekimui įgyvendinti buvo realizuota programa C++ kalba, pasinaudojus OpenCV biblioteka. Programa vaizdo kamera filmuoja judančius objektus ir tuo pačiu metu, fone, realiu laiku veikia objekto sekimo algoritmas.

2.2. Elementarus sekamo objekto judėjimas

2.2.1. Judėjimas viena kryptimi

Sekant objektus yra svarbu sekamą objektą apsibrėžti iš anksto. Eksperimento metu objekto aptikimui ir sekamų taškų nustatymui buvo realizuota funkcija, kuri sekamą objektą atpažįsta pagal jo spalvą. Funkcija yra iškviečiama pirmajame registruojamame vaizdo kadre, laiko momentu t_0. Kadras yra perkonvertuojamas iš RGB į HSV (*Hue, Saturation, Value*) spalvų erdvę. Pasirinkus tam tikrus HSV spalvų paletės reikšmių rėžius objektas yra išskiriamas iš aplinkos. Tuomet išskirtas objektas yra traktuojamas kaip sekamasis ir dalis jį apibrėžiančių pikselių yra pažymimi kaip sekamieji taškai. Šiuo atveju rėžiai yra nuo [130, 60, 60] iki [175, 255, 255], kas apibrėžia violetinę spalvą.

Pirmajame eksperimente buvo išanalizuotas objekto sekimas, kuomet objektas juda viena kryptimi. Paveikslėlyje (žr. 9 pav.) matoma popierinė dėžutė judanti į kairę, žalia spalva yra pažymėti sekamieji taškai. Iš viso objektui buvo nurodyti 300 sekamų taškų.



9 pav. Objekto judėjimas į kairę

Įvykus judesiui objektas buvo susektas sėkmingai. Iš visų 300 sekamų taškų, nei vienas nebuvo pamestas. Pats objekto poslinkis nebuvo didelis, tik keletas pikselių, nes sekamas objektas judėjo nedideliu greičiu. Yra žinoma, jog klasikinė Lucas – Kanade metodo versija sugeba rasti optinį srautą, tik kai objekto poslinkis yra mažas, o piramidinis Lucas – Kanade metodas geba apskaičiuoti žymiai didesnius objektų poslinkius. Nustatyti, koks yra piramidinio metodo pranašumas, objekto judėjimo greitis buvo po truputį didinamas, kol algoritmas pametė sekamą objektą. Didžiausias nustatytas objekto poslinkis, kurį dar sugebėjo apskaičiuoti algoritmas, buvo apytiksliai 30 pikselių. Paveikslėlyje (žr. 10 pav.) pavaizduotas apskaičiuotas optinis srautas, kai objektas greitai juda iš kairės į dešinę. Gautas rezultatas taip pat priklauso nuo įrašančios vaizdo kameros kokybės, kuo kokybiškesnis yra vaizdo įrašymas, tuo didesnį objekto judėjimą galima apskaičiuoti.



10 pav. Didžiausias apskaičiuotas objekto šuolis

2.2.2. Judėjimas artyn ir tolyn nuo žiūrovo

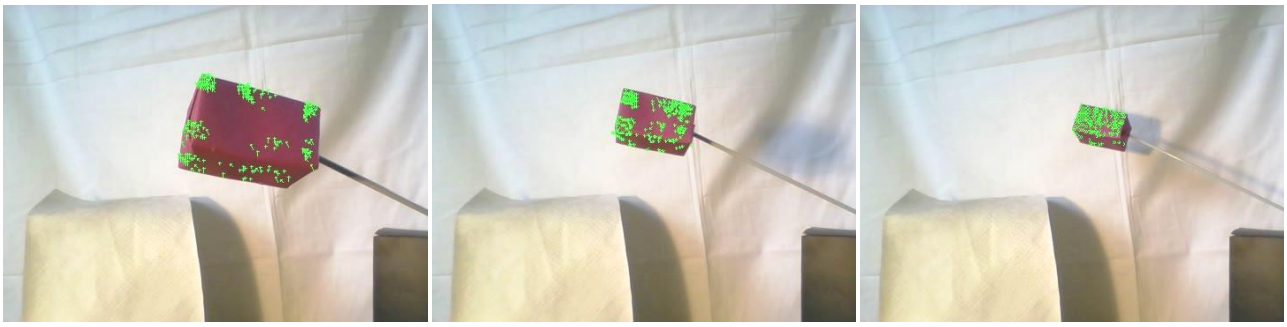
Optinis srautas yra skaičiuojamas objektų judėjimui dvimatėje erdvėje, kaip buvo aptarta 1.2.1 skyriuje, tačiau realiomis situacijomis judėjimas vyksta trimatėje erdvėje. Šiame eksperimente sekamas objektas buvo artinamas ir vėliau tolinamas nuo vaizdo kameros. Toks judėjimas negali būti užfiksuotas atsižvelgiant tik į dvi koordinates ir dvimatėje erdvėje artinimas ir tolinimas gali būti traktuojamas, kaip objekto didinimas ir mažinimas.

Iš pradžių sekamas objektas buvo priartintas prie vaizdo kameros. Paveikslėlyje (žr. 11 pav.) matomi trys kadrai, kairysis kadras paimtas laiko momentu t_0, kuomet yra iškviečiama objekto atpažinimo funkcija. Objektui vėl priskirta 300 sekamų taškų. Pradėjus objektą artinti, viduriniame kadre galima pastebėti, jog sekamieji taškai pradeda atsiskirti ir grupuotis ties dėžutės kraštais. Paskutiniame kadre dauguma taškų lieka išsidėstę ties dėžutės kampais. Kol kas nei vienas sekamas taškas nebuvo pamestas ir visi 300 taškų vis dar nusako sekamo objekto poziciją. Kitą vertus, išnagrinėjus kiekvieno sekamo taško judėjimą rezultate gauname, jog atskiros taškų grupės juda skirtingomis kryptimis. Panaikinus kadrų foną ir palikus tik sekamus taškus atrodytų, jog eksperimento metu buvo fiksuojamas sprogimas, kadangi pikselių grupės tolo viena nuo kitos visomis kryptimis.



11 pav. Objekto artinimas prie stebėtojo

Pradėjus objektą tolinti nuo vaizdo kameros situacija kartojasi, tik priešinga kryptimi. Paveikslėlio (žr. 12 pav.) viduriniajame kadre sekamų taškų išdėstymas yra beveik toks pats kaip ir prieš tai matytame paveikslėlyje. Sekamieji pikseliai grįžta į savo, prieš tai buvusią poziciją. Paskutiniame dešiniajame kadre situacija yra panaši kaip kadre paimtame laiko momentu t_0, tik taškai dėl įvykusiu skaičiavimo paklaidų nėra tvarkingai išdėstyti, kaip pirmajame kadre, tačiau taškų sankaupos dingo ir sekamas objektas vėl proporcingai padengtas sekamais taškais. Vėl panaikinus kadrų foną, atrodytų, jog taškai buvo lyg sutraukti į vieną bendrą debesį.



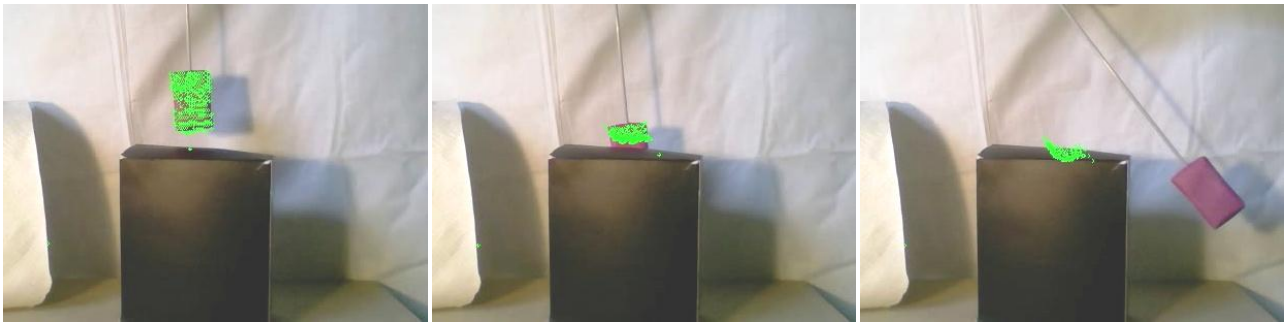
12 pav. Objekto tolinimas nuo stebėtojo

Iš gautų rezultatų galima daryti išvadą, jog tikrasis objekto judėjimas nėra užfiksuotas. Trimatėje erdvėje objekto judėjimą galima būtų apibrėžti *z* ašimi, o eksperimento atveju sekamieji taškai judėjo 360 laipsniu kampu nuo koordinačių pradžios, *x* ir *y* ašimis. Tokiu atveju sekamą objektą didinant arba mažinant būtų gauti panašūs rezultatai ir kas iš tikrųjų vyksta su sekamu objektu, pasakyti būtų sunku.

Svarbus yra ir taškų išdėstymas ties objekto kraštais ir kampais. Šį reiškinį galima paaiškinti tuo, jog skaičiuojant kadrui ryškumo gradientą yra išryškinami stebimo vaizdo kampai ir briaunos, ir būtent šios savybės labiausiai įtakoja optinio srauto rezultatą. Stebima dėžutė yra vienspalvė ir priartinus ją prie vaizdo kameros ties dėžutės centru, atsiranda didelis vienspalvis plotas. Tame plote nepavyksta rasti jokių vaizdo pokyčių, todėl optinio srauto paskaičiuoti nepavyksta. Tuo tarpu ties kraštais pokyčius fiksuoti lengviau ir visi optinio srauto vektoriai susikoncentruoja ties objekto kraštais.

2.2.3. Objekto uždengimas

Nagrinėjamo algoritmo reakcijai į sekamo objekto uždengimą patikrinti atliktas eksperimentas, kuriame objektas iš lėto pasislepia už didesnio objekto. Eksperimento metu sekamas objektas buvo vėl pažymėtas su 300 sekamų taškų.



13 pav. Objekto uždengimas

Gavus rezultatus (žr. 13 pav.) matyti, jog buvo pamesti visi sekamieji taškai. Įdomiau yra tai, jog visi sekamieji taškai liko ten kur paskutinį kartą buvo matytas objektas. Viduriniajame kadre galima matyti, kaip dėžutės apačioje esantys sekamieji taškai sustoja, kai ji pradeda slėptis už kadre matomos juodos dėžės. Taip visi taškai susikoncentruoja ties juodos dėžės kraštine. Juodos dėžės kraštinę galima išsivaizduoti kaip sieną, kuri nepraleidžia sekamųjų taškų, o pastarieji taškai atsimušę į sieną sustoja. Dabar objektas grįžęs į regos lauką yra nebesekamas ir norint vėl sekti objektą, reikia pakartotinai paleisti objekto atpažinimo funkciją.

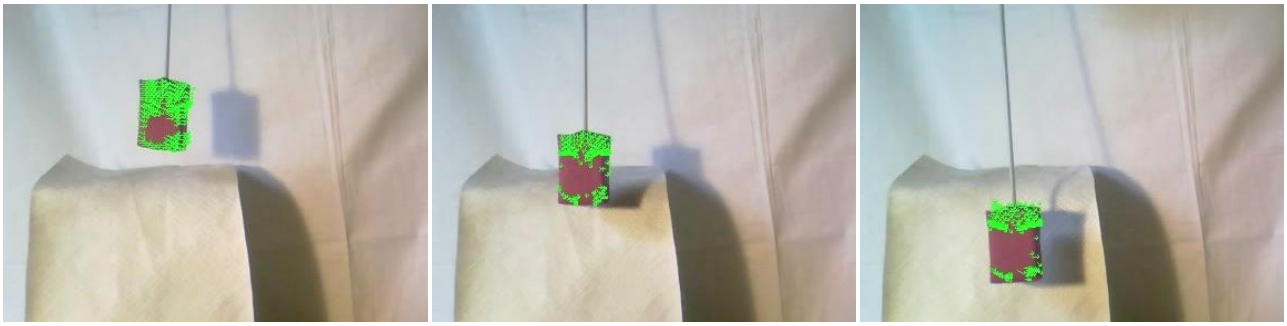
Galima daryti išvadą, jog algoritmas nėra atsparus objekto uždengimui. Visi sekamieji taškai lieka toje pozicijoje, kurioje sekamas objektas buvo paskutinį kartą aptiktas prieš jį uždengiant.

2.3. Sekamo objekto tranzitas skirtingo atspalvio fonais

2.3.1. Aplinkos įtaką sekamiems taškams

Ankščiau išnagrinėti atvejai yra labiau sutelkti į patį objektą, todėl šį kartą atkreiptas didesnis dėmesys į aplinką, kurioje objektas juda. Realiomis situacijomis aplinka, kurioje juda objektas, yra labai įvairi, sekamas objektas gali judėti danguje, vėliau kirsti horizonto liniją ir nusileisti ant žemės. Tokiu atveju prieš tai buvęs šviesus ir daugmaž vienspalvis fonas staiga tampa tamsesnis ir labiau triukšmingas. Šiame eksperimente algoritmas buvo išmėgintas, kuomet sekamas objektas pereina iš vieno fono į kitą.

Pirmoje situacijoje pradinio ir pereinamo fono kontrastas nėra didelis. Paveikslėlyje (žr. 14 pav.) objektas pajuda žemyn ir viduriniame kadre galima stebėti sekamų taškų reakciją į fonus skiriančią liniją. Perėjus objektui per fonų sankirtą dalis kraštinių taškų buvo nubraukti. Visgi iš visų 300 sekamų taškų nei vienas nebuvo pamestas, tai yra visi taškai vis dar apibrėžia sekamą objektą, tačiau atsižvelgiant į pradžioje nurodytus sekamus pikselius, galima teigti, kad jų sekimas nutrūko, o sekamieji taškai prisitvirtino prie naujų objekto pikselių.



14 pav. Objekto perėjimas tarp mažo kontrasto fono

Dabar padidinus fonų skirtumą, objektas pereis nuo šviesaus fono ant tamsaus, juodo. Paveiksle matomam žemiau (žr. 15 pav.) galima stebėti kaip algoritmas sunkiai susitvarko su užduotimi. Iš 300 sekamų taškų buvo pamesti 175 taškai, kas sudaro daugiau nei pusę. Didelį dėmesį reikia atkreipti į dėžutės kraštus. Viduriniajame kadre matyti kaip sekamieji taškai yra nubraukiami nuo dėžutės kraštų. Pirma, taškai dėžutės apačioje yra pastumiami aukštyn ties viduriu, tada pradedami stumti ir dėžutės kairėje esantys taškai. Galiausiai paskutiniame kadre nutraukiami ir dėžutės viršuje buvę taškai. Nutraukti taškai lieka kaboti ant tamsaus objekto fono ir nebeseka dėžutės. Rezultate turime daugiau nei puse pamestų taškų, o likusieji išsidėstę sekamo objekto centre.

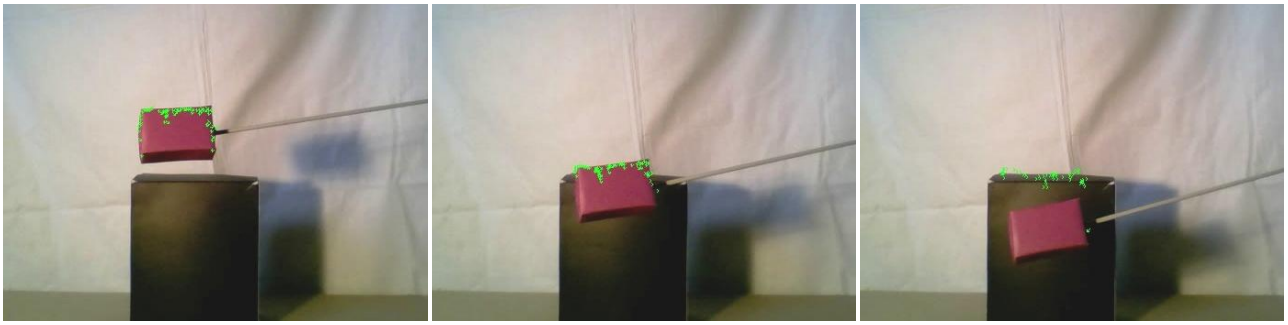


15 pav. Objekto perėjimas tarp didelio kontrasto fono

2.3.2. Nagrinėjamo ploto įtaka sekamiems taškams

Atlikus eksperimentus pastebėta, jog dažnu atveju, sekamieji taškai išsidėstę sekamo objekto kraštuose gali būti nubraukti ir nebesekti objekto. Šiai situacijai plačiau ištirti buvo atliktas dar vienas eksperimentas. Šį kartą buvo realizuota nauja objekto atpažinimo funkcija, kuri objektą atpažįstą pagal jo formą ir sekamus taškus pažymi ties jo kontūrais. Funkcija pradinį kadrą paverčia į dvejetainį paveikslėlį, kuriame pikselių reikšmė gali būti arba juoda arba balta. Tuomet yra

randamos visos baltų ir juodų pikselių sankirtos, kurios traktuojamos kaip kažkokio objekto briaunos, belieka tik pasirinkti kurį būtent kontūrą sekti.

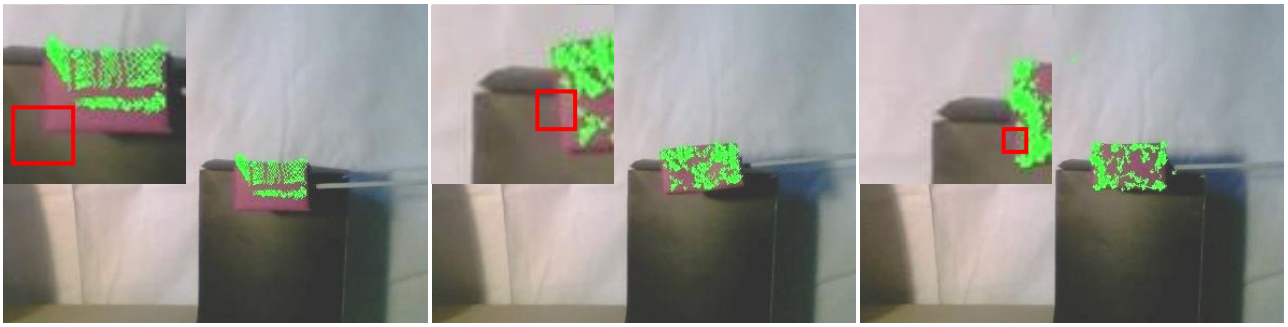


16 pav. Sekamų kraštinių taškų praradimas

Šiame eksperimente atkartota ta pati situacija, kaip ir prieš tai buvusiame, sekamas objektas pereina nuo šviesaus fono ant tamsaus, tačiau sekamieji taškai yra pažymėti sekamos dėžutės kraštuose. Sekamųjų taškų buvo nurodyta 100. Įvykus perėjimui, rezultate (žr. 16 pav.) gavome, jog buvo pamesti visi sekamieji taškai išskyrus vieną. Sulyginus du paskutinius eksperimentus, galima teigti, jog objekto sekimas atsižvelgiant tik į jo kraštus nėra pakankamai patikimas, būtina turėti sekamų taškų ir objekto centre.

Iš prieš tai gautų rezultatų matyti, jog taškai yra nubraukiami tam tikru atstumu nuo sekamo objekto krašto. Eksperimentas buvo atliktas Lucas – Kanade metodui nurodant 33x33 pikselių nagrinėjamą plotą, bei 3 piramidės lygius. Taškų nubraukimą galimai gali įtakoti būtent nagrinėjamo ploto dydis, todėl ta pati eksperimento situacija buvo atkartota du kartus, nagrinėjamą plotą sumažinus iki 15x15 ir 5x5 pikselių. Gauti rezultatai (žr. 17 pav.) iš tikrųjų patvirtino, jog sumažinus nagrinėjamą plotą sumažėja ir nubraukiamų taškų kiekis. Kairiojoje paveikslėlio dalyje nagrinėjamas plotas yra 33x33 pikseliai, viduriniojoje 15x15 ir dešiniojoje 5x5. Nagrinėjamam plotui esant 15x15 pikselių, prarastų taškų kiekis nukrito iki 63, o esant 5x5 iki 37. Lucas – Kanade metodas, optinį srautą apskaičiuoja pasinaudodamas kaimyninių pikselių reikšmėmis, kuo didesnis nagrinėjamas plotas pasirinktas, tuo daugiau pikselių įtakoja optinio srauto vektoriaus radimą. Geresniam supratimui yra paimamas žemiau esančio paveikslėlio (žr. 17 pav.) kairysis kadras. Raudonas kvadratas nurodo nagrinėjamo plotą, jo centre yra sekamas pikselis, šiuo atveju sekamas pikselis yra ant paties sekamo objekto (dėžutės) kampo. Tokiu atveju nagrinėjamas plotas tęsiasi už objekto ribų. Galima įsivaizduoti kaip sekamas objektas artėja prie paveikslėlyje matomos juodos dėžės. Savaime suprantama, jog dėžės kraštą pirma pasieks nagrinėjamo ploto kraštas (raudono kvadrato kraštinė) ir tik paskui, kai objektas pilnai priartės prie juodos dėžės, pasieks sekamas pikselis. Iki to laiko, juodos dėžės kraštinė įtakos optinio srauto skaičiavimą nagrinėjamam

pikseliui, nes ji patenka į nagrinėjamą plotą. Tokios įtakos pasekmės ir matomos eksperimento rezultatuose. Kuo nagrinėjamas plotas mažesnis, tuo vėliau atsiranda juodos dėžės kraštinės įtaka.



17 pav. Nagrinėjamo ploto įtaka sekamiems taškams

Nubraukiamų taškų problemai išspręsti dalinai padeda nagrinėjamo ploto mažinimas, tačiau tuo pačiu, tai mažina algoritmo tikslumą. Paveikslėlyje (žr. 17 pav.) galima pastebėti, jog kairiojoje pusėje sekamų taškų išdėstymas yra žymiai tvarkingesnis, nei palyginus su sekančiomis paveikslėlio dalimis. Objekto atpažinimo funkcija sekamus taškus sudeda tinkleliu, vėliau taškų išdėstymas priklauso nuo sekamo objekto transformacijų ir optinio srauto skaičiavimo tikslumo. Kairiojoje pusėje vis dar galima iššiūrėti taškų tinklelį, nors objektas nukeliavo nemažą atstumą nuo pradinės padėties (žr. 15 pav.) iki dabartinės. Tuo tarpu, mažesnio nagrinėjamo ploto atveju, taškai jau išsibarsto padrikai po sekamo objekto paviršių. Sumažinus nagrinėjamo ploto dydį, sumažėja ir algoritmo tikslumas. Rezultate gauname, jog taškų nubraukimo problemos negalima visiškai išspręsti sumažinus nagrinėjamą plotą, todėl reikia parinkti tokius algoritmo parametrus, kad nubraukiamų taškų kiekis būtų pakankamai mažas kartu išlaikant ir patenkinamą algoritmo tikslumą. Šiuo konkrečiu atveju nagrinėjamas plotas gali būti nustatytas 21x21 pikselių. Piramidės lygių kiekis šiam taškų praradimui įtakos neturi.

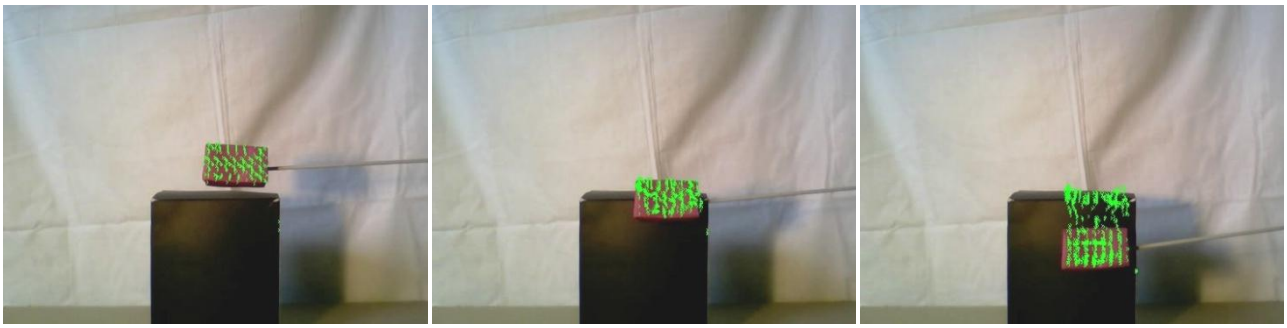
2.4. Objekto sekimas atnaujinant sekamo objekto taškus

2.4.1. Nuolatinis naujų sekamų taškų didinimas

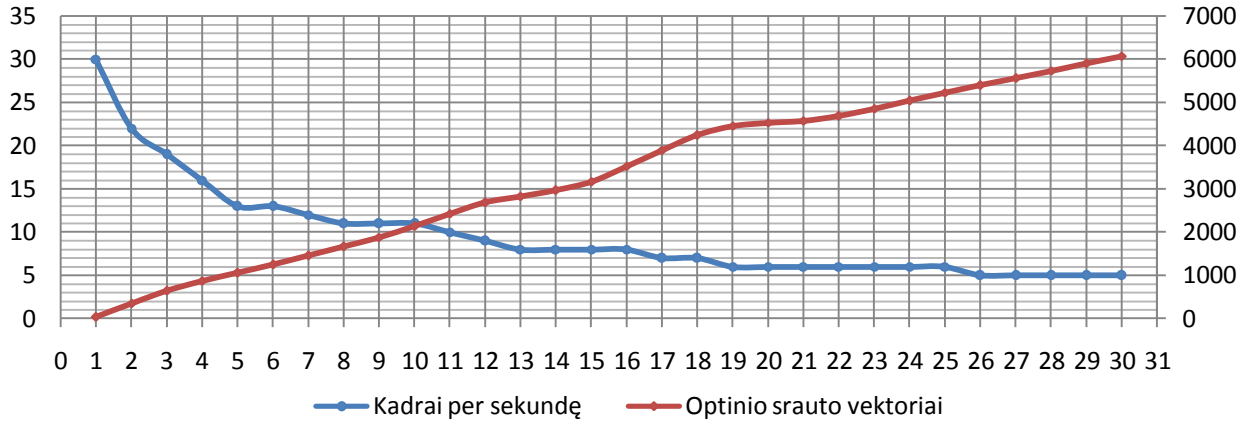
Sekamų taškų praradimui sumažinti naudojama programa buvo modifikuota taip, jog objekto atpažinimo funkcija būtų iškviečiama kiekviename kadre, taip užtikrinant, kad visos galimos naujos objekto sekimo savybės būtų išnaudotos ir pamesti pikseliai būtų vėl pažymėti sekamais taškais.

Atlikus eksperimentą paveikslėlyje (žr. 18 pav.), dešiniajame kadre matyti, jog objektui perėjus iš vieno fono ant kito, jis liko visas padengtas sekamaisiais taškais. Kitą vertus, situacija pasikartojo kaip ir 3.3.1 stebėtame eksperimente ir buvo pamesta daug sekamųjų taškų. Maždaug 200 taškų

nustojo sekti objektą ir liko kabėti virš tamsaus fono. Nors objektas vis dar pilnai sekamas, tokie algoritmo grąžinami duomenys yra netenkinantys ir jų panaudojimas gali būti komplikuotas. Pamesti taškai tampa šiukšlėmis ir apie sekamą objektą nieko nepasako.



18 pav. Naujų sekamų taškų priskyrimas



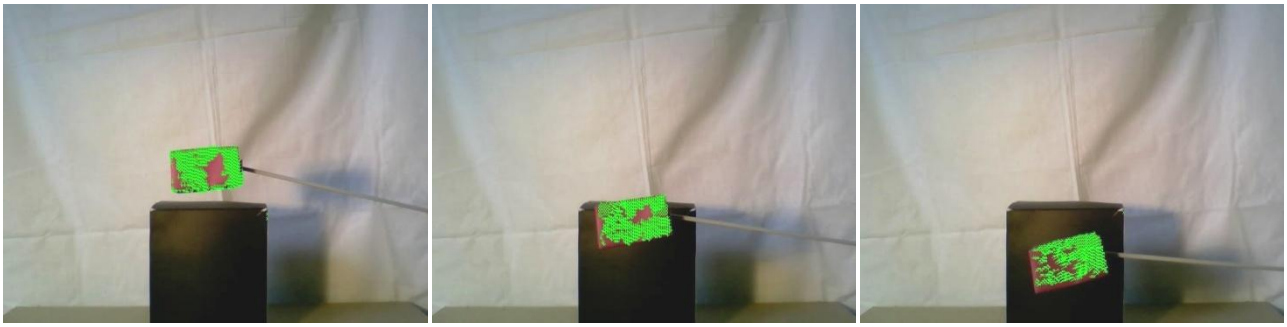
19 pav. Sekamų taškų skaičiaus įtaka greičiui

Nuolatinis sekamų taškų didinimas taip pat apkrauna kompiuterinę įrangą. Grafike (žr. 19 pav.) matyti, kaip krenta skaičiavimų sparta priklausomai nuo sekamųjų taškų kiekio. Taškų skaičiui pasiekus 1000, algoritmo sparta nukrenta dvigubai ir programa sugeba apdoroti tik 15 kadrų per sekundę. Toks apdorojamų kadrų kiekis netenkina realaus laiko sąlygos. Maža to, sekamų taškų kiekis toliau augą, kol galiausiai programos greitis nusistovi ties 5 kadrais per sekundę.

2.4.2. Naujų taškų priskyrimas pašalinus senus

Algoritmo greičio problemai išspręsti buvo atliktos dar kelios programos modifikacijos. Objekto atpažinimo funkcija nuo šiol prieš tai buvusius sekamus taškus panaikina ir priskiria naujus, bei uždėtas apribojimas, leidžiantis pažymėti maksimaliai 300 sekamų taškų.

Eksperimentas su perėjimu iš šviesaus fono į tamsų vėl buvo pakartotas, o Lucas – Kanade nagrinėjamas plotas nustatytas 33x33 pikselių. Iš rezultatų (žr. 20 pav.) matyti, jog po objekto perėjimo, sekamas objektas išliko padengtas sekamais taškais, o visi pamesti taškai buvo pašalinti objekto atpažinimo funkcijos. Algoritmo greitis išlieka pastovus, 30 kadrų per sekunde, viso programos darbo metu, nes sekamų taškų kiekis niekada neviršija 300 taškų.



20 pav. Sekamų tašku atnaujinimas

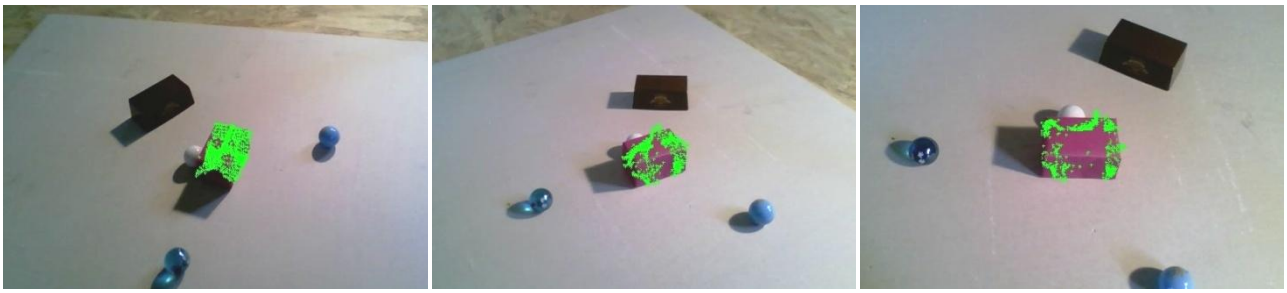
Svarbu yra paminėti vieną algoritmo grąžinamų duomenų pasikeitimą. Objekto atpažinimo funkciją kviečiant kiekviename kadre ir naikinant prieš tai buvusius sekamus taškus, nebelieka vientisos grandinės nurodančios, kaip konkretus pikselis judėjo viso objekto sekimo metu. Norint apskaičiuoti objekto judėjimą, nebegalima pasikliauti vieno pikselio optinio srauto vektoriumi. Nuo šiol objekto judėjimo kryptis nustatoma pagal visą sekamų taškų debesį, atsižvelgiant tik į du gretimus kadrus. Optinis srautas skaičiuojamas dviem gretimiems kadrams, todėl ir objekto judėjimas turėtų būti nustatomas pagal du kadrus. Vėliau rezultatai turi būti sujungiami su kiekviena kadrų pora ir taip randama bendra objekto judėjimo trajektorija.

2.5. Objekto sekimas judant stebėtojui

Paskutinis eksperimentas buvo atliktas su tikslu ištirti, kaip algoritmas elgiasi judant ne stebimam objektui, o pačiam stebėtojui. Eksperimento metu sekmas objektas buvo pastatytas šalia kitų nejudančių objektų ir vaizdo kamera pradėta sukti ratu iš kairės į dešinę, vėliau priartinama prie stebimo objekto. Objekto atpažinimo funkcija buvo paleista tik kartą, vaizdo fiksavimo pradžioje. Objektui sekti buvo nurodyti 300 taškų.

Išanalizavus gautus rezultatus, sekimo pabaigoje buvo pamesti 42 taškai. Sukant kamerą viena sekamo objekto kraštinė nusisuka ir dingsta iš regos lauko, tai gali būti traktuojama kaip kraštinės uždengimu, kas ir įtakojo taškų praradimą. Taip pat šalia sekamo objekto yra balta sfera (žr. 21 pav.), kuri įtakoja optinio srauto skaičiavimą, kaip aprašyta 3.3.3 skyriuje. Nors sfera nejuda, jos

pozicija vaizdo kameros atžvilgiu keičiasi. Paveikslėlyje (žr. 21 pav.), dešiniajame kadre taip pat galima pastebėti, jog sekamieji taškai susikoncentravo ties objekto kampais, kas taip buvo aprašyta 3.2.2 skyriuje.



21 pav. Objekto sekimas judant stebėtojui

Rezultate galima teigti, jog algoritmas gali būti naudojamas ir kai vyksta abipusis judėjimas, tai yra kai juda tiek objektas, tiek stebėtojas. Pritaikius programos modifikacijas aptartas 2.4.2 skyriuje ir esamą eksperimentą pakartojus, sekamų taškų išdėstymą po visą sekamą objektą, pavyko išlaikyti viso objekto sekimo metu.

3. Lucas – Kanade metodo paralelizacija

3.1. Metodo realizacija ant grafinio procesoriaus

Optiniam srautui apskaičiuoti dažnu atveju prireikia didelių kompiuterinės įrangos resursų, kas apriboja optinio srauto algoritmų panaudojimą. Pagrindinis resursas yra kompiuterio procesoriaus greitis. Vieno branduolio kompiuterio procesorius turi būti ypač spartus, jog sugebėtų apskaičiuoti optinį srautą didesniems vaizdo kadrams. Esamoje situacijoje gali būti naudojama procesų paralelizacija, pasinaudojant kelių branduolių arba grafiniais procesoriais.

Pagerinti turimos programos veikimą buvo nuspręsta pasinaudoti grafiniu procesoriumi ir CUDA (*Compute Unified Device Architecture*) architektūra. CUDA leidžia rašyti programinę įrangą, suderinamą su ateinančios kartos grafiniais procesoriais (GPU). Dėl techninių aplinkybių, tokios programos turi būti suskirstytos į tris lygius. Pirmajame lygyje yra gijos, kurios vykdo elementarius nuoseklius nurodymus. Gijos yra sugrupuotos į skirtingus blokus, kurie patys yra suskirstyti tinkleliu. Visos gijos vykdo tas pačias veiksmų sekas, tik su skirtingais duomenimis. Kiekvienas blokas yra vykdomas vieno procesoriaus branduolio, kuris turi ryšį su keletą kitų blokų ir siekiant sumažinti duomenų perdavimo vėlavimus, esant spartinančiosios atminties trūkumui,

procesorius gali persijungti ir vykdyti kitą bloką. Kuomet blokų yra daugiau nei laisvų procesoriaus branduolių, neaptarnaujami blokai laukia savo vykdymo eilės. Tokio veikimo sparta priklauso nuo įrangos resursų, kuo daugiau branduolių procesorius turės, tuo daugiau blokų bus aptarnaujama lygiagrečiai, bei kuo didesnė bus spartinančioji atmintis, tuo daugiau blokas galės savyje turėti gijų, tokiu atveju bus atlikta daugiau operacijų. Efektyviai parašytos programos gali veikti nepriklausomai nuo grafinio procesoriaus modelio, kol nebus nurodyti fiksuoti blokų kiekiai ir dydžiai. Programa vykdoma ant skirtingų grafinių procesorių, skirsis tik blokų dydžiu ir jų pasiskirstymų per visus procesoriaus branduolius. [JYA81]

OpenCV biblioteka turi realizuotus Lucas – Kanade metodus piramidėms skirtus tiek centriniam procesoriui, tiek grafiniam procesoriui. Pagal nutylėjimą bibliotekoje metodai skirti grafiniam procesoriui yra išjungti, todėl norint įgalinti CUDA palaikymą, biblioteką reikia perkompiliuoti nurodant papildomus parametrus. Kompiuterio programinė įranga taip pat buvo praturtinta CUDA Toolkit 6.5 programavimo aplinka.

3.2. Algoritmo skaidymas į dalis

Lucas – Kanade algoritmas gali būti lygiagretinamas, nes optinio srauto skaičiavimai, vienam pikseliui, yra nepriklausomi vienas nuo kito, tai yra skaičiavimo rezultatas nepriklauso nuo jau prieš tai apskaičiuotos reikšmės, todėl visiems pikseliams optinis srautas gali būti skaičiuojamas lygiagrečiai tuo pačiu metu. Išnagrinėjus plačiau galima pastebėti, kad algoritmas turi keturias lygiagretinamas dalis:

- Piramidės sudarymas;
- Gradientų skaičiavimas;
- Optinio srauto atitikmens radimas pereinant į kitą piramidės lygį;
- Optinio srauto skaičiavimas kiekvienam kadro pikseliui.

Piramidžių konstravimas iš tikrųjų vyksta tiek nuosekliai, tiek lygiagrečiai, sekantis piramidės lygis gali būti sudarytas tik esant jau sudarytam prieš tai buvusiam lygiui, tačiau pats kadro mažinimas, tai yra pikselio perstumdymas, gali vykti lygiagrečiai, nes nauja pikselio pozicija sumažintame kadre priklauso tik nuo jau sudaryto žemesnio lygio, o ne nuo kaimyninio pikselio. Lygiai tas pats galioja ir gradientų skaičiavimui bei optinio srauto atitikmens radimui, procesai vyksta paeiliui kiekvienam piramidės lygiui. Galiausiai optinis srautas yra apskaičiuojamas naudojantis tik duomenimis, gautais apskaičiavus ryškumo gradientus, todėl vieno pikselio poslinkiui rasti gali būti priskirta viena CUDA gija.

3.3. Algoritmo greičių palyginimas

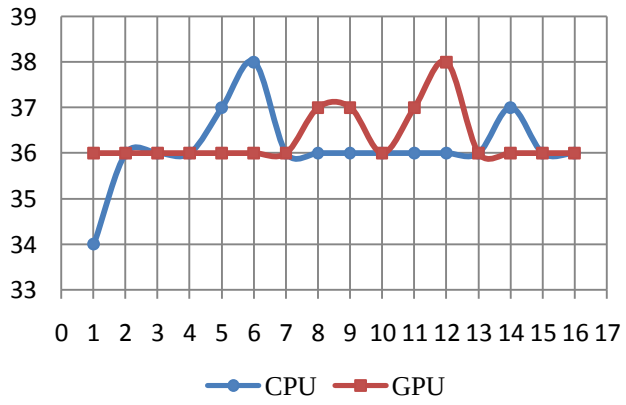
Pritaikius programą vykdymui ant grafinio procesoriaus buvo atliktas eksperimentas, kuris leido nustatyti kokį pranašumą programa įgauna Lucas – Kanade algoritmą vykdant lygiagrečiai. Eksperimento metu buvo palygintas prieš tai buvęs algoritmas, skirtas tik centriniam procesoriui su algoritmu grafiniam procesoriui, pritaikius juos tam pačiam vaizdo įrašui (žr. 1 lentelę) ir nustačius tuos pačius parametrus.

1 Lentelė

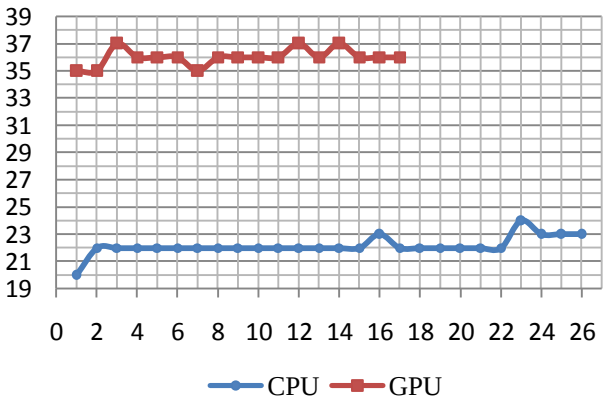
Kadro dydis (pikseliai)	640x320
Kadrų kiekis (vnt.)	586
Kadrų dažnis (kadrai/sekundė)	30
Trukmė (sekundės)	20

Nagrinėjamo vaizdo įrašo kadrų dydis yra 640x320 pikselių, todėl algoritmams buvo nustatytas optimalus piramidės dydis su trim lygiais. Pirmajame palyginime nagrinėjamo ploto dydis buvo nustatytas 5x5 pikselių. Gavus rezultatus matyti (žr. 22 pav.), jog abi algoritmo versijos sugebėjo apdoroti apytiksliai tą patį kadrų kiekį per sekundę, vidutinis greitis yra 36 kadrai per sekundę. Vaizdo įrašo kadrų atvaizdavimo greitis yra 30 kadrų per sekundę, o viso įrašo trukmė yra 20 sekundžių. Algoritmai vaizdo įrašą sugebėjo apdoroti per 16 sekundžių, kas yra net greičiau nei natūrali įrašo peržiūros trukmė, todėl galima teigti, jog rezultatai tenkina realaus laiko sąlygą.

Sekančiame palyginime nagrinėjamo ploto dydis buvo padidintas iki 15x15 pikselių. Šį kartą greitis ant centrinio procesoriaus nukrito iki 22 kadrų per sekundę, o ant grafinio liko nepakitęs. Skirtumas tarp greičių yra 14 kadrų per sekundę. Grafike (žr. 23 pav.) matyti, jog vaizdo įrašo apdorojimas užtruko 26 sekundes ir algoritmas ant centrinio procesoriaus nesugebėjo veikti realiu laiku.



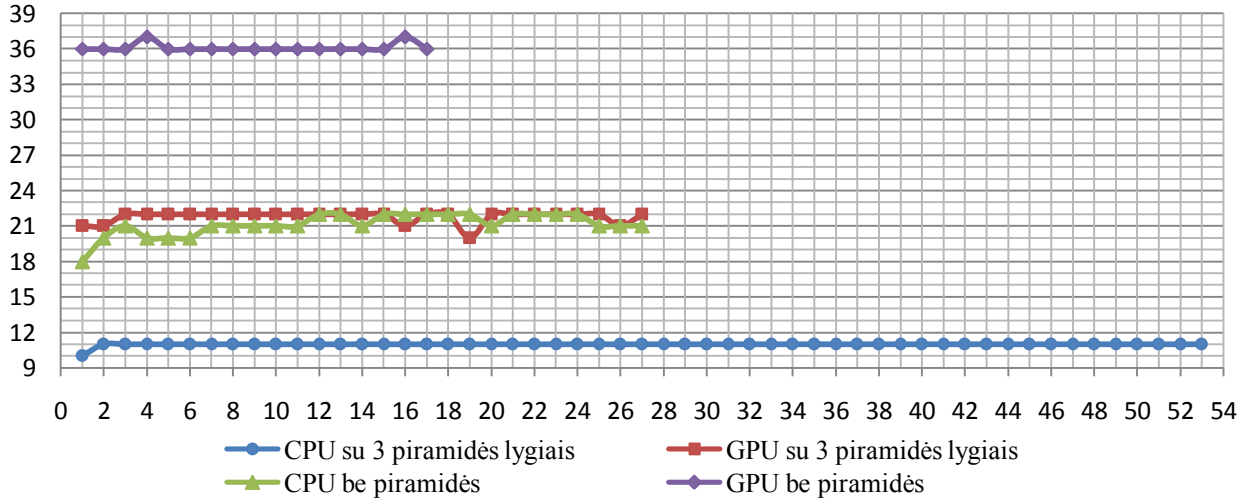
22 pav. Greičio palyginimas kai nagrinėjamo ploto dydis 5x5 pikselių



23 pav. Greičio palyginimas kai nagrinėjamo ploto dydis 15x15 pikselių

Nagrinėjamas plotas buvo dar kartą padidintas iki 33x33 pikselių. Šioje situacijoje algoritmas tiek vienu, tiek kitu atveju, nesugebėjo vaizdo įrašo apdoroti realiu laiku. Pasinaudojant grafiniu procesoriumi greitis buvo pasiektas tik 22 kadrai per sekundę, tačiau algoritmas ant centrinio procesoriaus, vaizdo įrašą apdorojo 11 kadrų per sekundę greičiu ir užtruko 53 sekundes, todėl rezultatas vis tiek yra priimtinas. Patikrinti piramidės dydžio įtaką skaičiavimo greičiui, bandymas buvo pakartotas atsisakius piramidės, tai yra piramidės lygių kiekį nustačius lygų vienam. Šiuo atveju yra atsisakoma viena algoritmo lygiagretinama dalis ir optinis srautas bus skaičiuojamas tik originalaus dydžio vaizdo kadrams.

Gavus rezultatus matyti, jog algoritmas ant grafinio procesoriaus vėl pasiekė 36 kadrų per sekundę greitį, o ant centrinio procesoriaus 21 kadrą per sekundę. Grafinio procesoriaus atveju, greitis tenkina veikimo realiu laiku sąlygą, tačiau svarbu priminti, jog esant tik vienam piramidės lygiui, algoritmas nesugebės apskaičiuoti optinio srauto esant dideliems objektų poslinkiams erdvėje. Greičių palyginimus su ir be piramidės, galima pamatyti grafike žemiau (žr. 24 pav.). Šis bandymas taip pat parodė, kad atsisakius vienos algoritmo lygiagretinamos dalies, grafinio procesoriaus pranašumas sumažėjo. Antroje lentelėje matyti, jog esant 33x33 nagrinėjamo ploto dydžiui ir trijų lygių piramidei, algoritmas ant grafinio procesoriaus veikė 2 kartus greičiau, nei ant centrinio procesoriaus. Tuo tarpu atsisakius piramidžių, pranašumas sumažėjo iki 1,7 karto.

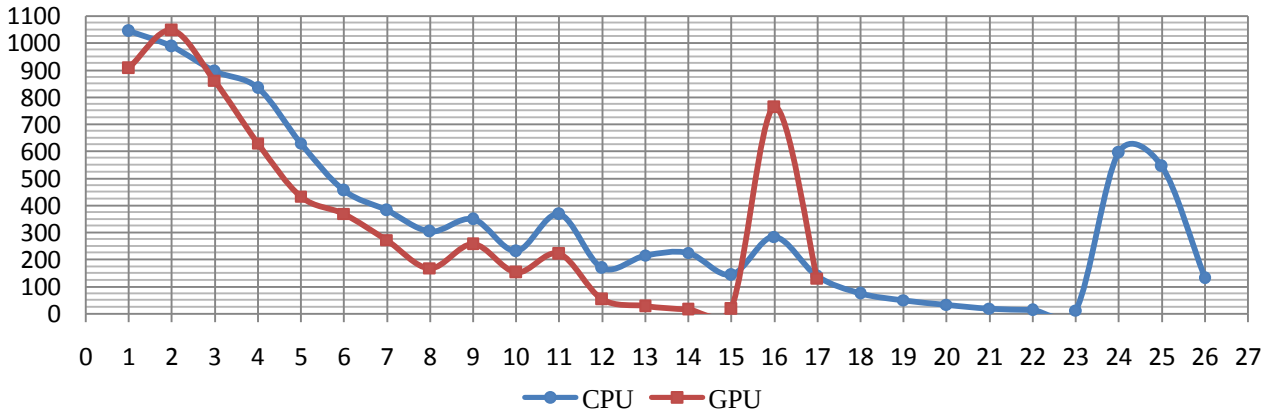


24 pav. Greičio priklausomybė nuo piramidės lygių esant 33x33 nagrinėjamo ploto dydžiui

2 lentelė

Nagrinėjamo ploto dydis	Piramidės lygiai	Greičio vidurkis ant centrinio procesoriaus	Greičio vidurkis ant grafinio procesoriaus	Grafinio procesoriaus pranašumas (kartais)
5x5	3	36	36	1
15x15	3	22	36	1,6
33x33	3	11	22	2
33x33	1	21	36	1,7

Palyginus algoritmų rastą optinio srauto vektorių kiekį (žr. 25 pav.), galima matyti, jog tiek vienu, tiek kitu atveju, algoritmai į judesių intensyvumą reaguoja panašiai. Abi algoritmų versijos atlieka tas pačias veiksmų sekas ir tuos pačius skaičiavimus, skiriasi tik laikas kuomet veiksmai yra atliekami.



25 pav. Optinio srauto vektoriai

Iš gautų rezultatų galima daryti išvadą, kad algoritmas tikrai gali būti paralelizuojamas ir paspartinamas pasinaudojus grafiniu procesoriumi. Eksperimento metu naudojama aparatūra, sugebėjo vaizdo įrašą apdoroti realiu laiku.

4. Metodo palyginimas su OpenCV objektų sekimui skirta realizacija

4.1. OpenCV realizacija objektų sekimui

Išanalizavus objekto sekimą pasinaudojant piramidiniu Lucas – Kanade metodu ir pritaikius jį veikimui ant grafinio procesoriaus, buvo atlikti keli bandymai, skirti ištirti objektų sekimą realiomis situacijomis. Prieš tai atlikti eksperimentai buvo atliekami laboratorinėmis sąlygomis, dirbtinai sukuriant situaciją konkrečiai objekto sekimo savybei ištirti. Šį kartą buvo fiksuojamas vaizdas orlaiviu iš viršaus. Sekamieji objektai buvo gatvėmis judantys automobiliai. Gauti rezultatai buvo sulyginti su jau standartine tapusia OpenCV realizacija, skirta objektams sekti.

Realizacija susideda iš *cv::Tracker* klasės, kuri siūlo objektų sekimą pagrįstą penkiais skirtingais algoritmais: MIL, Boosting, MedianFlow, TLD ir KCF, bei papildomų įrankių, kurie leidžia naudotojui rankiniu būdu, patogiai pasirinkti sekamus objektus. Palyginimui atlikti buvo pasirinktas MedianFlow algoritmas, kadangi šis metodas kaip ir nagrinėjamas yra pagrįstas Lucas – Kanade metodu optiniam srautui rasti. Algoritmui yra nurodomi du vaizdo įrašo kadrai I_t ir I_{t+1}, bei aprėpties plotas β_t. Plotas β_t yra užpildomas taškais išdėstant juos stačiakampiu tinkleliu. Taškai vėliau yra susekami Lucas – Kanade algoritmu, kuris apskaičiuoja optinį srautą tarp kadrų I_t ir I_{t+1}. Algoritmo išvestis yra aprėpties plotas β_{t+1} su apskaičiuotomis naujomis sekamųjų taškų pozicijomis. [KMM10]

4.2. Pirmasis palyginimas

4.2.1. Objektų sekimas realizuotu metodu

Sekimui buvo pasirinkti penki objektai, paveikslėlyje žemiau (žr. 26 pav.) pažymėti skaičiais, jų judėjimo trajektorijos pažymėtos skirtingomis spalvos linijomis. Objektai yra nurodomi pirmajame vaizdo įrašo kadre, nuspalvinus norimus sekti pikselius tam tikra spalva. Sekimo pradžioje yra iškviečiama objektų atpažinimo funkcija, kuri nuspalvintus pikselius pakeičia į sekamuosius taškus (žr. 2.2.1 skyrių). Nagrinėjamas vaizdo įrašas buvo sumažintas du kartus, taip objektų sekimas buvo išbandytas ant trijų skirtingų rezoliucijų. Kiekvienam sekamam objektui buvo priskirta 30 sekamųjų taškų.



26 pav. Sekamų objektų judėjimo trajektorijos

Trečioje lentelėje matomi rezultatai. Objektas pametęs puse savo sekamųjų taškų yra laikomas pamestu. Pavyzdžiui, toks objektas yra pažymėtas antru numeriu, 960x540 rezoliucijoje po dalinio objekto uždengimo, jis prarado 25 sekamuosius taškus.

3 lentelė

Kadro dydis	**Objektas 1**	**Objektas 2**	**Objektas 3**	**Objektas 4**	**Objektas 5**	**Greitis (k/s)**
960x540	Dalinai uždengtas	Dalinai uždengtas	Susektas	Susektas (uždengtas)	Susektas	37
Pamesti taškai	13/30	25/30	0/30	0/30	3/30	
1280x720	Dalinai uždengtas	Dalinai uždengtas	Susektas	Susektas (uždengtas)	Susektas	33
Pamesti taškai	30/30	27/30	0/30	0/30	0/30	
2704x1520	Dalinai uždengtas	Dalinai uždengtas	Susektas	Susektas (uždengtas)	Susektas	10
Pamesti taškai	30/30	30/30	0/30	0/30	0/30	

4.2.2. Objektų sekimas su OpenCV

Palyginimui buvo atkartota ta pati situacija ir pasirinkti tie patys objektai. Sekamieji objektai yra pasirenkami iškviečiant funkciją cv::selectROI pirmajame vaizdo įrašo kadre. Funkcija leidžia, pasinaudojant kompiuterio pele, apibrėžti stačiakampiu rėmeliu norimus objektus (žr. 27 pav.). Apibrėžtas plotas ir yra algoritmo naudojamas aprėpties plotas β_t. Kuo daugiau sekamųjų objektų yra, tuo daugiau atskirų aprėpties plotų bus naudojama (β_t^1, β_t^2, β_t^3, ...).



27 pav. Sekamų objektų judėjimo trajektorijos (OpenCV)

Iš rezultatų ketvirtoje lentelėje matyti, jog OpenCV programa veikė lėčiau, nei programa realizuota šiame darbe, nes nebuvo naudojamas grafinis procesorius. Įvertinus susektų objektų kiekį matyti, jog šiuo atveju buvo geriau susektas 1 objektas. Šis objektas yra dalinai uždengiamas kelio ženklo, tai galima pamatyti ties nutrūkstančia objekto trajektoriją nurodančią liniją. Nepaisant to, objekto sekimas nenutrūko visose vaizdo įrašo rezoliucijose. Tuo tarpu darbe realizuotoje programoje, objektą pavyko susekti tik žemiausioje rezoliucijoje, nes vaizdo įrašo detalumas yra mažiausias. Didinant vaizdo įrašo rašką didėjo ir pamestų taškų skaičius.

4 lentelė

Kadro dydis	Objektas 1	Objektas 2	Objektas 3	Objektas 4	Objektas 5	Greitis (k/s)
960x540	Dalinai uždengtas (susektas)	Pamestas	Susektas	Susektas (uždengtas)	Susektas	11
1280x720	Dalinai uždengtas (susektas)	Pamestas	Susektas	Susektas (uždengtas)	Pamestas	7
2704x1520	Dalinai uždengtas (susektas)	Pamestas	Susektas	Pamestas	Susektas	2

4.3. Antrasis palyginimas

4.3.1. Objektų sekimas realizuotu metodu



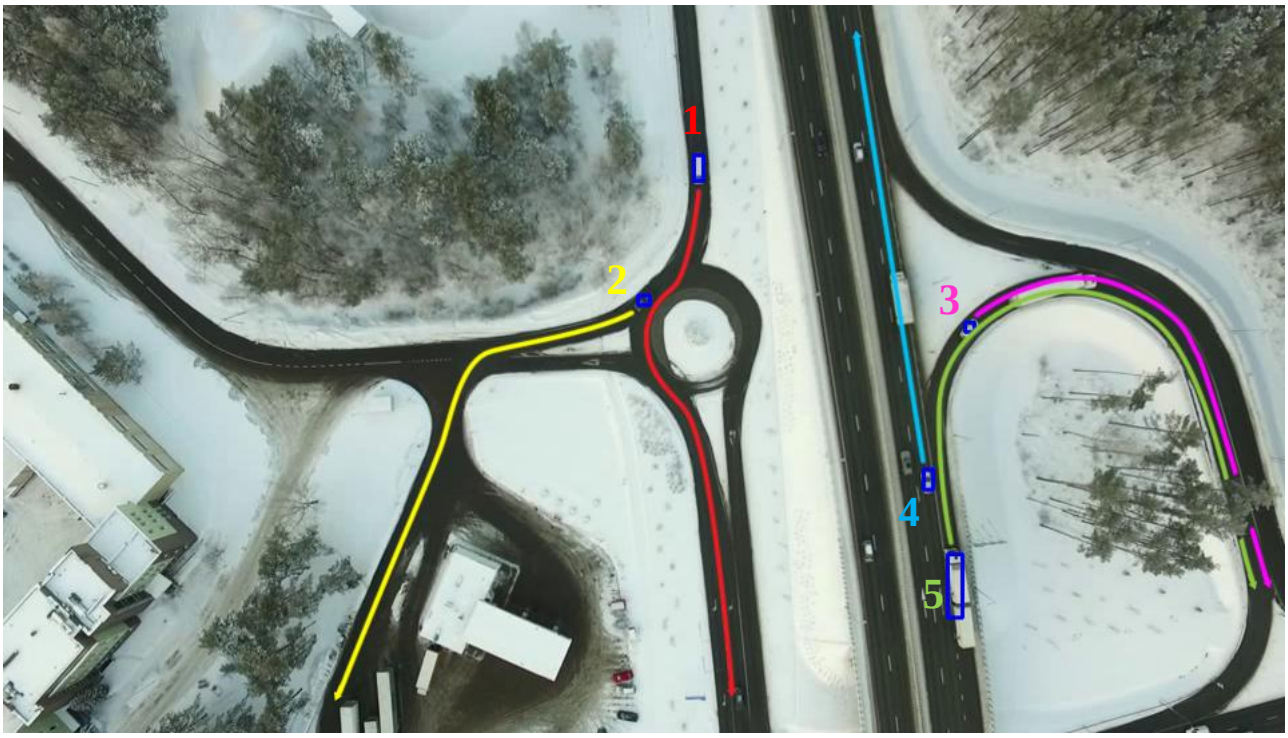
28 pav. Sekamų objektų judėjimo trajektorijos

Antrame palyginime buvo pasirinkti vėl penki objektai, tik esant kitai situacijai, objektai dabar juda kitomis kryptimis (žr. 28 pav.). Gauti rezultatai matomi penktoje lentelėje. Objektui pažymėtam 5 numeriu, buvo priskirta 100 sekamųjų taškų dėl jo didesnės apimties, kitiems sekamiesiems objektams vėl buvo priskirta 30 taškų.

5 lentelė

Kadro dydis	Objektas 1	Objektas 2	Objektas 3	Objektas 4	Objektas 5	Greitis (k/s)
960x540	Susektas	Susektas	Susektas (uždengtas)	Susektas	Dalinai uždengtas	36
Pamesti taškai	0/30	9/30	0/30	0/30	79/100	
1280x720	Susektas	Susektas	Susektas (uždengtas)	Susektas	Dalinai uždengtas	30
Pamesti taškai	0/30	0/30	0/30	0/30	94/100	
2704x1520	Susektas	Susektas	Susektas (uždengtas)	Susektas	Dalinai uždengtas	10
Pamesti taškai	0/30	0/30	0/30	0/30	100/100	

4.3.2. Objektų sekimas su OpenCV



29 pav. Sekamų objektų judėjimo trajektorijos (OpenCV)

6 lentelė

Kadro dydis	Objektas 1	Objektas 2	Objektas 3	Objektas 4	Objektas 5	Greitis (k/s)
960x540	Pamestas	Pamestas	Susektas (uždengtas)	Susektas	Dalinai uždengtas (pamestas)	10
1280x720	Pamestas	Pamestas	Susektas (uždengtas)	Susektas	Dalinai uždengtas (pamestas)	6
2704x1520	Susektas	Susektas	Susektas (uždengtas)	Susektas	Dalinai uždengtas (pamestas)	2

Atlikus visus bandymus, gauti rezultatai buvo susumuoti ir išvesti į vieną bendrą lentelę (žr. 7 lentelę). Iš rezultatų matyti, jog šiame darbe realizuotas metodas gavo geresnius rezultatus, nei OpenCV pasirinktas metodas. Pasinaudojus grafiniu procesoriumi, objekto sekimas vyko realiu laiku, tiek 960x540, tiek 1280x720 rezoliucijas turinčiame vaizdo įraše, kai tuo tarpu palygintame metode, pasiektas greitis mažiausioje rezoliucijoje tik 11 kadrų per sekundę. Susektų objektų kiekis taip pat yra didesnis darbe realizuoto metodo atveju.

7 lentelė

	Nagrinėjamas metodas		**OpenCV**	
	Susekti objektai	**Greičio vidurkis**	**Susekti objektai**	**Greičio vidurkis**
960x540	8/10	37	6/10	11
1280x720	7/10	32	5/10	7
2704x1520	7/10	10	7/10	2

Išvados ir rezultatai

Darbo metu pasiekti rezultatai:

1. Realizuotas objekto sekimas pasinaudojant Lucas – Kanade metodu optiniam srautui rasti.
2. Pasiūlytas ir realizuotas sprendimas pagerinti algoritmo tikslumą, sekamo objekto sekamuosius taškus atnaujinant viso sekimo metu.
3. Pasiūlytas ir realizuotas sprendimas pagreitinti objekto sekimo algoritmo darbą pasinaudojant grafiniu procesoriumi.
4. Palyginta darbe realizuota objektų sekimo programa su OpenCV siūloma objektų sekimo realizacija.

Išanalizavus ir įvertinus rezultatus buvo gautos išvados:

1. Nagrinėjamą algoritmą pritaikius grafiniam procesoriui buvo pasiektas dvigubai didesnis algoritmo veikimo greitis, nei palyginant su ant centrinio procesoriaus veikiančia algoritmo versija. Tyrimo metu pastebėta, jog didinant Lucas – Kanade metodo nagrinėjamo ploto dydį ir didinant piramidės lygių kiekį, atitinkamai didėja grafinio procesoriaus panaudojimo pranašumas.
2. Su darbe realizuota programa pavyko pasiekti geresnių rezultatų nei pasinaudojant OpenCV realizacija. Vaizdo įrašų apdorojimas vyko nuo trijų iki penkių kartų greičiau, esant mažai vaizdo įrašo raiškai, nagrinėjama programa parodė geresnius rezultatus, susekė dviem objektais daugiau, tačiau raišką padidinus iki 2704x1520 pikselių abu metodai susekė vienodą kiekį objektų.

Summary

Visual object tracking is one of the main tasks of computer vision and there are developed many various solutions to solve this task. This thesis investigates one of the possible solutions to implement object tracking using optical flow estimation. The aim of this thesis is to implement the object tracker, find solutions that improve the tracking performance and compare obtained results with one of existing globally approved method.

The object tracker implementation is based on Lucas – Kanade method for optical flow estimation. To determine the problems that can be faced while tracking objects, the implementation was analyzed by imitating different situations such as moving tracking object closer and further from the observer, moving object in different background environment and moving observer instead of object. The analysis showed that algorithm has intention to lose tracing points of the object, so there was suggested and implemented method which searches and updates object trace points on each video frame. To increase the speed of the optical flow estimation the algorithm was parallelized using graphical processor. After all improvements object tracker was used to track vehicles moving on the streets in video taken from aircraft. Finally, all results were compared with OpenCV object tracker.

The results in this thesis show the exact advantage in speed of parallelized Lucas – Kanade algorithm compared with classic version and approve that implemented object tracker successfully surpass OpenCV method.

Literatūros sąrašas

[PU13] D. Patel, S. Upadhyay „Optical Flow Measurement using Lucas – Kanade Method“, International Journal of Computer Applications (0975 – 8887) Volume 61– No.10, 2013.

[JYA81] J. Marzat, Y. Dumortier, A. Ducrot „Real-Time Dense and Accurate Parallel Optical Flow using CUDA“ WSCG'2009 - The 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp 105–110, 2009.

[Bou11] Jean-Yves Bouguet, „Pyramidal Implementation of the Affine Lucas – Kanade Feature Tracker Description of the algorithm“ Intel Corporation Microprocessor Research Labs, 2001.

[SA11] N. Sethi, A. Aggarwal „Robust Face Detection and Tracking Using Pyramidal Lucas – Kanade Tracker Algorithm“ Department of Computer Science and Department of Computer Science and Engineering, Vol 2 (5), 1432-1438, 2011.

[MKM+14] S.A. Mahmoudi, M. Kierzynka, P. Manneback, K. Kurowski „Real-time motion tracking using optical flow on multiple GPUs“ Bulletin of The Polish Academy of Sciences, Vol. 62, No. 1, 2014

[HS81] B.K.P. Horn and B.G. Schunck, „Determining optical flow“ Artificial Intelligence, vol 17, pp 185–203, 1981.

[KMM10] Z. Kalal, K. Mikolajczyk, J. Matas „Forward-Backward Error: Automatic Detection of Tracking Failures“ International Conference on Pattern Recognition, 2010

OpenCV (Open Source Computer Vision) Documentation 2016, prieiga per internetą: http://docs.opencv.org/

CUDA Toolkit Documentation 2016, prieiga per internetą: http://docs.nvidia.com/cuda/