KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Vaclovas Stončius

PET PLASTIKO DRIBSNIŲ RŪŠIAVIMO SISTEMOS SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Arūnas Lipnickas

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

PET PLASTIKO DRIBSNIŲ RŪŠIAVIMO SISTEMOS SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas

Studijų programos pavadinimas (kodas 621H66001)

Vadovas
Doc. dr. Arūnas Lipnickas

Recenzentas
Doc. dr. Gintaras Dervinis

Projektą atliko
Vaclovas Stončius

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos fakultetas

(Fakultetas)

Vaclovas Stončius

(Studento vardas, pavardė)

Valdymo technologijos 621H66001

(Studijų programos pavadinimas, kodas)

„PET plastiko dribsnių rūšiavimo sistemos sukūrimas ir tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ____ m. _____ _____ d.

Kaunas

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

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

_____ _____

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

Stončius, Vaclovas. PET plastiko dribsnių rūšiavimo sistemos sukūrimas ir tyrimas. Valdymo sistemų magistro baigiamasis projektas / vadovas doc. dr. Arūnas Lipnickas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Mokslo kryptis ir sritis: Inžinerijos mokslų magistras, Valdymo technologijos.

Reikšminiai žodžiai: *PET dribsnis, spalvų erdvės, optinis rūšiavimas, MATLAB.*

Kaunas, 2017. 57 p.

SANTRAUKA

Šio darbo tikslas yra sukurti sistemą, kuri gebėtų automatiškai ir su pakankama greitaveika atskirti vienodo atspalvio skaidrias plastiko daleles nuo kitų atspalvių dalelių. Neskaidrūs objektai yra išskiriami naudojant skirtingas spalvas aprašančias koduotes bei taikant pozicijos paieškos metodus nustatomos objekto koordinatės. Svarbiausi rūšiavimo sistemos parametrai yra greitaveika ir atpažinimo tikslumas.

Pirmojoje darbo dalyje išanalizuojamas PET dribsnių gavimas, jų rūšiavimo tikslingumas, aprašytos keturios dažniausiai naudojamos spalvas aprašančios spalvų požymių erdvės: RGB, HSV, L*a*b bei YCbCr. Taip pat aprašomi objekto padėties nustatymo metodai dvimačiam objektui: nuoseklios paieškos, fono užpildymo bei dvimačių didelių objektų (BLOB) paieškos metodai. Antrojoje darbo dalyje aprašomas sudarytas algoritmas bei patikrinamas jo veikimas esant skirtingoms spalvų erdvėms, išbandomi pozicijos paieškos metodai bei patikrinamas sistemos veikimas, kai pradinis vaizdas yra veikiamas vaizdo triukšmų. Atrinkta geriausiai šiai užduočiai tinkanti spalvinė erdvė, pozicijos paieškos metodas.

Stončius, Vaclovas. Development and Research of a PET Flake Sorting System: Master‘s thesis in Control technologies / supervisor doc. dr. Arūnas Lipnickas. The Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Research area and field: Master Of Engineering Sciences, Control Technologies.

Key words: *PET flakes, color spaces, optical sorting, MATLAB.*

Kaunas, 2017. 57 p.

SUMMARY

The main focus of this thesis is to develop a system that is able to distinguish clear white PET plastic flakes from the other colors automatically and with sufficient throughput. Colored flakes are distinguished by using different color spaces and different position search methods are applied to determine each flake's position. The most important parameters for the optical sorting system are the speed and recognition accuracy.

The first part of this paper represents the analysis of the literature, how PET flakes are made and why it is important to sort flakes by color, also four most commonly used color spaces are presented: RGB, HSV, L*a*b and YCbCr. Different algorithms for finding an object location are also presented: union-find, flood fill and BLOB analysis. The second part of this paper represents the algorithm which was developed to sort white clear flakes. The algorithm was tested with different color spaces and algorithms for finding an object location. The effectiveness of each color space and algorithms for finding object location were found and the best ones were chosen.

TURINYS

ĮVADAS

Pastaraisiais dešimtmečiais labai padidėjus plastiko gamybai (nuo 225 mln. iki 311 mln. tonų per metus) [1], kartu išaugo ir susidarančių atliekų skaičius. Siekiant sumažinti išmetamų atliekų kiekį, resursus, energiją skirtą pagaminti plastikui vis dažniau kalbama apie panaudoto plastiko perdirbimą. Nuo 2006 m. iki 2014 m. plastiko perdirbimas augo nuo 4,7 iki 7,7 mln. tonų per metus [1]. Tokį didelį kiekį norint perdirbti ir tikslingiau panaudoti, reikia jį išrūšiuoti į vertingesnį skaidrų ir į įvairiaspalvį [2]. Šiam tikslui įgyvendinti yra naudojamos įvairios atpažinimo sistemos, kurios geba išrūšiuoti greitai lekiančias susmulkinto plastiko daleles.

Magistrinio **darbo tikslas** yra sukurti PET plastiko dribsnių rūšiavimo sistemą, kuri gebėtų automatiškai ir su pakankama greitaveika atskirti vienodo atspalvio skaidrias plastiko daleles nuo kitų atspalvių dalelių, naudojantis Matlab programiniu paketu. Ištirti Matlab programiniame pakete siūlomus vaizdo apdorojimo įrankius dribsnių spalvos ir pozicijos erdvėje nustatymui

Darbo uždaviniai:

- aprašyti PET plastiko dribsnių gamybos procesą ir spalvinio rūšiavimo naudingumą;
- apžvelgti, egzistuojančias PET plastiko rūšiavimo technologijas;
- pasiūlyti metodus leidžiančius išskirti skaidrius baltus PET dribsnius nuo spalvotų;
- surinkti eksperimentinius duomenis ir išbandyti programos efektyvumą, siekiant išrūšiuoti skaidrius baltus PET plastiko dribsnius iš įvairiaspalvių dribsnių srauto, Matlab aplinkoje;
- Ištirti vaizdo triukšmų įtaką, rūšiavimo sistemos veikimui.

1. METODINĖ DALIS

1.1 Plastiko perdirbimo procesas

Iki 1980 metų polietileno tereftalatas (angl. PET, *Polyethylene terephthalate)* dažniausiai buvo naudojamas gaminant tekstilės audinius [3]. Laikui bėgant PET plastikas vis labiau įsitvirtino kaip pagrindinė žaliava plastikinių butelių gamyboje. Lengva pigi plastikinė pakuotė greit pakeitė didžiąją dalį iki tol plačiai naudotų stiklinių butelių, tačiau šių pakuočių naudojimas sukėlė ekologinių problemų. PET butelių naudojimo trukmė yra santykinai nedidelė, lyginant su laiku per kurį PET plastikas suirtų natūraliai ir dėl to didelė dalis plastikų yra išmetami ir yra kaupiami sąvartynuose. Taip pat kai stikliniai buteliai gali būti panaudojami keliskart, plastikinės pakuotės dažniausiai yra vienkartinės ir po pirmo naudojimo yra išmetamos. Dėl to augant plastiko atliekų skaičiui ir esant ribotam gamtinių išteklių kiekiui buvo būtina surasti metodus, kaip perdirbti plastiką.

Plastiko atliekų perdirbimas turi nemažai privalumų:

- leidžia sumažinti energijos kiekį reikalingą naujų plastikų gamybai;
- sumažina sąvartynų skaičių;
- sukuria žmonėms naujų darbo vietų.

Perdirbtas plastikas gali būti panaudotas gaminant aibę produktų, pavyzdžiui, įvairias produktų pakuotes, baldus bei kitas kompozicines medžiagas.

Plastiko perdirbimo metu kylą problemų dėl to, kad yra ne viena plastiko rūšis, kuri viena nuo kitos skiriasi savo fizikinėmis bei cheminėmis savybėmis. Yra trys populiariausios plastikų rūšys [4]:

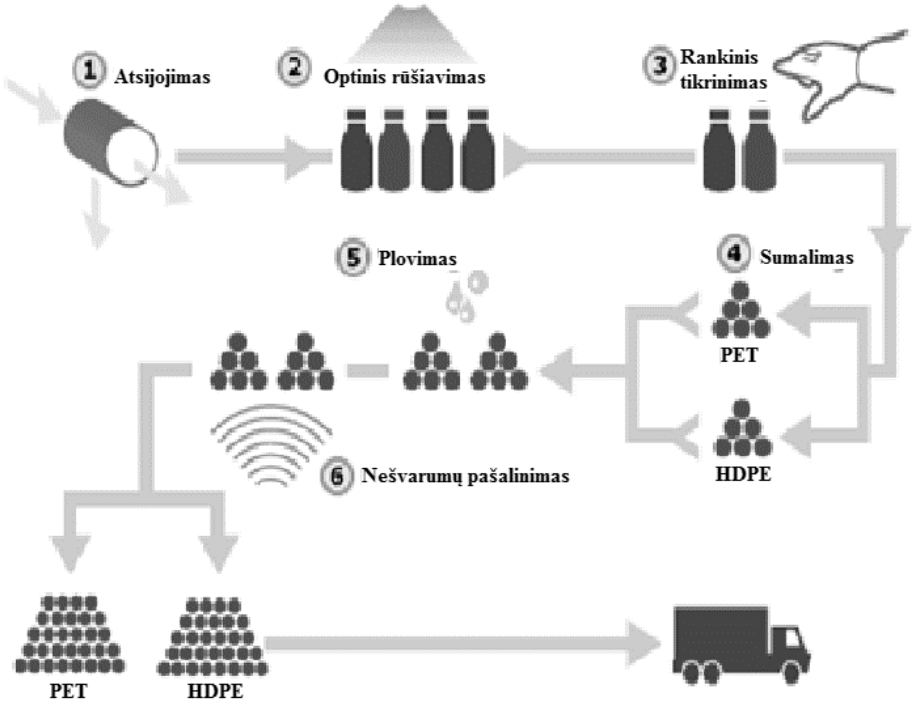
- polietileno tereftalatas;
- didelio tankio polietilenas (angl. HDPE, *High-density polyethylene*);
- polivinilchloridas (angl. PVC *Polyvinyl chloride*).

Kiekviena plastiko rūšis turi būti atskirta viena nuo kitos, nes kiekviena iš jų pasižymi skirtingomis lydymosi temperatūromis bei pritaikymo sritimis tam, kad jas perdirbus būtų galima panaudoti vėliau.

Yra keletas metodų perdirbti panaudotą plastiką į naudingą žaliavą. Skirtingos plastiko savybės, pavyzdžiui, spalva ar skaidrumas įtakoja PET plastiko rūšiavimo metodo parinkimą. Nuo tinkamo metodo parinkimo priklauso koks bus rūšiavimo sistemos atsiperkamumas bei veikimo tikslumas.

Bendras perdirbimo procesas atvaizduotas 1.1 paveiksle ir susideda iš šešių dalių:

- 1 – išsijojimas;
- 2 - optinis rūšiavimas;
- 3 - rankinis rūšiavimas;
- 4 – smulkinimas;
- 5 – valymas;
- 6 – nešvarumų pašalinimas;



1.1 pav. Plastiko išrūšiavimas ir perdirbimas [5]

1.1.1 Rūšiavimas ir smulkinimas

Plastikas į perdirbimo centrus yra atvežamas supresuotas į ryšulius. Čia jis yra išpakuojamas panaudojant besisukantį būgną 1 (*žr.* 1.1 pav.), kuriuo kartu yra išpurtomi nešvarumai ir kai kurie butelių kamšteliai. Panaudojant elektromagnetą yra pašalinami magnetiniai metalai, o sūkurinis magnetinis laukas leidžia pašalinti nemagnetinius metalus, tokius kaip aliuminis, varis, bronza. Perdirbimo gamykloje plastikas yra išrūšiuojamas pagal skirtingus plastiko tipus. Šiame etape naudojamas tiek optinis 2, tiek rankinis 3 rūšiavimo metodai. Po to, kai plastiko produktai yra išrūšiuoti, jie yra susmulkinami į mažas daleles 4, kurių dydis priklauso nuo konkretaus perdirbimo metodo, kuris vėliau bus naudojamas. Šios plastiko dalelės yra vadinamos dribsniais [6].

Siekiant išauginti perdirbamo plastiko vertę reikalinga išlaikyti dribsnių grynumą, nes neperdirbami elementai sumažina plastiko panaudojimo sritis bei kainą. Po susmulkinimo įvairūs rūšiavimo metodai naudojantys oro nupūtimą gali būti taikomi siekiant nufiltruoti lengvesnes medžiagas, pavyzdžiui, etiketes arba dangtelius iš perdirbamo plastiko srauto. Smulkinimas padeda atskirti plastikus nuo ne plastikinių medžiagų, tačiau pašalinės priemaišos: butelių etiketės bei kitos šiukšlės sumažina perdirbimo efektyvumą, o tai padidina veiklos sąnaudas dėl sumažėjusio produktyvumo [6]. Siekiant padidinti produktyvumą reikalinga išrūšiuoti plastikus.

1.1.2 Plastiko plovimas

Sekantis žingsnis po plastiko susmulkinimo yra dribsnių plovimas 5 (*žr.* 1.1 pav.). Plovimas vykdomas kambario temperatūros arba šiltesniame vandenyje. Siekiant pagerinti plovimą yra naudojami specialios dezinfekcinės medžiagos ir įvairūs plovikliai, kurie leidžia pašalinti biologines atliekas. Tačiau nebiologinės atliekos, pavyzdžiui, rašalas, klijai ar etiketės gali trikdyti plovimą ir tuo pačiu užteršti visą plastiką [6]. Išsimaišęs etikečių rašalas plovimo vandenyje gali nudažyti visą plastiką arba prie esamų klijų likučių gali prikibti įvairūs nešvarumai. Siekiant sumažinti riziką sugadinti visą partiją, perdirbtas plastikas dažnai perrūšiuojamas antrą kartą po skalbimo ciklo, taip pašalinant nepageidaujamas medžiagas.

1.1.3 Plastikų atskyrimas pagal medžiagos tankį

Plastikai ir kitos neperdirbamos medžiagos dažniausiai turi skirtingą tankį, kuris leidžia juos atskirti naudojant hidrocikloninius įrenginius arba vandens vonias [6]. Kai naudojamas vanduo, sunkesnės plastiko dalelės nusėda apačioje, o lengvesni teršalai pakyla į paviršių. Tačiau atskirti skirtingus plastikus vieną nuo kito gali būti ypač sudėtinga, jei jie turi panašų tankį. Pavyzdžiui, išskirti dribsnius, kuriuose yra PET ir PVC plastiko naudojat vandens vonias yra beveik neįmanoma, nes šie plastikai turi beveik identiškas fizikines charakteristikas: PET tankis 1,38g/cm³, kai PVC tankis 1,3–1,45g/cm³. Dėl to reikalinga naudoti kitus rūšiavimo metodus [7]: rūšiavimą rentgeno, artimaisiais infraraudonaisiais spinduliais. Skirtingas medžiagas yra daug lengviau atskirti kai jos pagamintos iš skirtingo tankio medžiagų negu iš tos pačios rūšies plastiko [6].

1.1.4 Džiovinimas ir filtravimas

Pabaigus smulkinimo, plovimo bei atskyrimo procesus, perdirbamas plastikas yra dar kartą praskalaujamas vandenyje siekiant pašalinti likusius nešvarumus ar valymo produktus nuo

ankstesnių etapų 6 (*žr. 1.1 pav.*). Po praskalavimo vykdomas džiovinamas ir plastikas išlydomas. Plastiko lydimas taip pat padeda išrūšiuoti nesilydančias medžiagas, kurios praėjo visus ankstesnius rūšiavimo, valymo etapus. Naudojant ekstruderį plastikiniai dribsniai yra išlydomi ir yra suformuojamos plastikinės granulės. Granuliavimas išaugina plastiko vertę [6], nes yra daug paprastesnė medžiaga naudoti gamyboje, taip pat yra patogesnė transportavimui.

1.2 Tiriamasis objektas – PET dribsnis

Polietileno tereftalatas (PET) yra viena iš dažniausiai perdirbamų polimerinių dervų, kuri naudojama gaminant sintetinius pluoštus bei plastikinius konteinerius, dažniausiai butelius. Milijonai tonų PET plastiko yra perdirbama kiekvienais metais ir parduodama gamintojams visame pasaulyje [6]. Vien Lietuvoje per metus pagaminama apie 500 000 tonų PET granulių, skirtų maisto pakuočių gamybai [8].

PET yra 100 proc. perdirbama pakuotė. Jos poveikis aplinkai lyginant su kitomis perdirbamų pakuočių rūšimis yra daug palankesnis. PET pakuotėje supakuoti produktai lyginant su kitomis tarai gaminti naudojamomis medžiagomis yra lengvesni ir atsparesni mechaniniam poveikiui, todėl transportavimui reikia mažiau kuro, daugiau produktų pervežami nesugadinti. Antrinė PET žaliava naudojama įvairių kilimėlių, elektrinių prietaisų, automobilių įrangos detalių, rūbų, statybinių medžiagų gamyboje [9].

Plastiko perdirbimo proceso metu, plastikai yra susmulkinami į dribsnius (*žr*. 1.2 pav.), kurių dydis gali skirtis nuo 3 mm iki 25 mm diametro. Susmulkinimas leidžia paversti trimačius butelius į vienos plokštumo daleles, taip palengvindamas atpažinimo procesą.

Priklausomai nuo užsakovo poreikio gali būti reikalaujamos skirtingos PET dribsnių savybės, kuriose nurodoma kiek daugiausiai gali būti pašalinių dalelių viename milijone visų dalelių (angl. ppm, *Pieces per million*). Bendri maksimalūs užterštumo lygiai PET plastike [10]:

- 8-10ppm aliuminio;
- 10ppm etikečių;
- 10ppm nesilydančių medžiagų;
- 10-25ppm PVC (angl. *Polyvinyl chloride***)**;
- 10ppm EVOH (angl. *Ethylene vinyl alcohol***)**;
- 10ppm HDPE (angl *High-density polyethylene***)**;
- 10ppm klijų.

1.2 pav. PET dribsniai [11]

1.3 Plastikos rūšiavimas pagal spalvas

Siekiant panaudoti PET plastiko dribsnius gaminant dirbtinius pluoštus ar plastikus maisto pramonei, jie turi būti išvalyti nuo įvairių priemaišų [12]: kitų plastiko rūšių (PVC, PP, PS), medžio, metalo bei kitos spalvos PET.

Yra trys pagrindinės plastiko atpažinimo sistemų rūšys [17]. Pirmiausia yra optinės rūšiavimo sistemos, kurios naudoja regimąją šviesą plastikinių butelių atskyrimui pagal spalvą. Antra yra sistemos, kurių veikimas paremtas tuo, kad bangos praeina tiesiai per plastiką ir jutiklis jas užfiksuoja iš kitos plastiko pusės. Kadangi kiekviena plastiko rūšis turi skirtingą reakciją į šviesos bangas priklausomai nuo jos cheminės sudėties, tai leidžia išrūšiuoti plastiką pagal tipą, neatsižvelgiant į objekto spalvą. Trečioji yra paviršiaus skenavimo įrenginiai, kur bangos atsispindi nuo plastiko paviršiaus ir jas fiksuoja jutiklis esantis toje pačioje pusėje kaip ir bangų generatorius. Analogiškai sistemoms, kurios peršviečia plastiką kiaurai, skirtingos plastikų rūšys skirtingai atspindi bangas.

Siekiant pašalinti priemaišas yra naudojami įvairūs plastiko rūšies atpažinimo metodai [13], kurių veikimas pagrįstas:

- spektroskopija;
- artimaisiais infraraudonaisiais spinduliais (angl. *NIR, Near-infrared*);
- lazeriais;
- rentgenu;
- rentgeno spindulių fluorescencija (angl. *XRF, X-ray fluorescence*).

Spalvos nustatymui naudojami optiniai metodai [13]:

- vaizdo kameros;
- krūvio sąsajos kameromis (angl. *CCD, charge-coupled device*);
- CCPD (angl. *Charge Coupled Photodiode*) kameros;
- Spektroskopija.

Dabartiniu metu populiariausios ir geriausiai veikiančios rūšiavimo sistemos naudoja rentgeno, rentgeno spindulių fluorescencijos (XRF) ar artimųjų infraraudonųjų spindulių (NIR) spindulius arba veikia regimojoje šviesoje [14]. Kadangi šio darbo problema yra skaidrių PET plastiko dribsnių atskyrimas nuo įvairiaspalvių, bus nagrinėjamas optinis rūšiavimas naudojant skaitmenines vaizdo kameras.

Optinis rūšiavimas – tai vizualus butelių ar dribsnių rūšiavimas naudojant fotodetektorius (šviesos daviklius), fotoaparatą ar žmogaus akis [15]. Vizualus rūšiavimas yra technologija, kuri gali daryti tai, ką ir žmogaus akis – gali vizualiai rūšiuoti medžiagas. Ji yra naudojama atskirti spalvotus polimerus arba etiketes.

Norint atlikti kokybišką vizualinę analizę reikia įvertinti du pagrindinius veiksnius: darbinio lauko apšvietimą ir tinkamos vaizdo kameros parinkimą. Tai yra du faktoriai, kurie tiesiogiai įtakoja gaunamo vaizdo kokybę. Šie pagrindiniai įrankiai turi būti parenkami atsižvelgiant į tokius veiksnius: atpažinimo tikslumas, vaizdo dydis (laukas), greitis (vaizdo fiksavimo ir apdorojimo) ir spalvinės informacijos poreikis.

Tačiau ši technologija negali būti naudojama atskirti plastiko tipą nebent medžiaga yra vizualiai pakeista, pavyzdžiui buvo panaudotas papildomas dažiklis. Tačiau gali būti naudojama:

- rūšiuojant tiek butelius, tiek dribsnius;
- atskirti skaidrų PET nuo žalio, mėlyno, geltono, ir t. t.;
- atskirti skaidrų HDPE nuo spalvoto HDPE ir t. t.

Tinkamai sumontuotos ir prižiūrimos optinio rūšiavimo sistemos paprastai gali veikti daugelį metų su gana mažomis eksploatacijos ir priežiūros sąnaudomis [11].

1.4 Spalvos komponentai ir matematinis atvaizdavimas

Skaitmeninis vaizdas pirmą kartą buvo atvaizduotas 1957 metais, kai Russelas A. Kirschas išrado įrenginį leidžiantį išsaugoti nuskenuotos nuotraukos vaizdą skaitmeniniu būdu [16]. Pats pirmas užfiksuotas vaizdas buvo nespalvotas – monochromatinis. Vėliau atsiradus poreikiui atvaizduoti spalvotus vaizdus, iškilo papildomų problemų su realios spalvos atvaizdavimu. Problema kilo todėl, kad vaizdą užfiksuojantys ir atvaizduojantys prietaisai turi

ribotas spalvų atvaizdavimo galimybes, ribas. Siekiant spręsti šią problemą buvo sukurti skirtingi metodai leidžiantys atvaizduoti vaizdus skaitmeninėje erdvėje. Izaokas Niutonas tirdamas saulės šviesą pastebėjo, kad šviesos spalva priklauso nuo jos spektrinių komponentų. Vėliau Maksvelas ir Grasmannas pateikė teoremą, jog spalvą sudaro trys viena nuo kitos nepriklausančios dedamosios [17] bei pademonstravo, jog bet kuri spalva gali būti gauta iš trijų pagrindinių spalvų komponenčių. Pagrindinės spalvų komponentės:

a) intensyvumas – energijos srauto matas, galia tenkanti ploto vienetui (W/m²), kuo krentančios šviesos energijos srautas yra didesnis, tuo vaizdas yra ryškesnis;
b) spalvos tonas, kuris priklauso nuo šviesos bangos ilgio;
c) spalvos sotumas, kuris priklauso nuo baltos spalvos kiekio spalvoje. Kuo daugiau baltos spalvos, tuo spalva yra mažiau soti.

Šios trys kombinacijos leidžia sudaryti tūkstančius chromatinių spalvų atspalvių.

Vaizdo kamerose spalvotas vaizdas yra fiksuojamas 3 jutikliais. Kiekvienas jutiklis yra skirtas dirbti skirtingose šviesos bangos ilgio diapazonuose:

- su ilgosiomis bangomis (raudona spalva);
- vidutinio ilgumo (žalios spalvos);
- trumposiomis bangomis (mėlyna spalva).

Norint aprašyti spalvas skaitmeniniu būdu yra naudojamos matematinės spalvų erdvės, kurios leidžia supaprastinti spalvos aprašymą ir atvaizdavimą skirtingose situacijose. Plačiausiai žinomos ir naudojamos yra RGB, HSV, L*a*b bei YCbCr spalvų erdvės [18]. Kiekviena iš jų turi savo privalumų ir trūkumų bei skirtingų pritaikymo galimybių.

1.4.1 RGB spalvinė erdvė

Plačiausiai naudojama ir žinoma yra RGB spalvinė erdvė [19]. RGB (angl. *Red, Green, Blue*) spalvinę erdvę sudaro trys pagrindinės spalvos: raudona, žalia bei mėlyna. Kiekvienas aktyvusis taškas (angl. *pixel)* vaizde yra aprašomas vektoriumi (R, G, B), kurio dedamosios yra atitinkamos raudonos, žalios ir mėlynos spalvos vertės. Suliejus šias spalvas gaunama spalva, kuri atitinka žmogaus matomą spalvą. Maišant šias tris spalvas galima išgauti bet kokią kitą spalvą [20]. Kai visos vektoriaus dedamosios yra lygios nuliui, tuomet gaunama spalva yra juoda bei atvirkščiai jei atitinka maksimalias reikšmes, gaunama balta spalva [21]. Visos kitos spalvos, kurias sudaro vektoriaus dedamosios kintant nuo 0 iki maksimalios vertės gali būti aprašomi įvairiais būdais:

- aritmetiškai – skaičiais nuo 0.0 iki 1.0. Naudojami ir trupmeniniai skaičiai;
- procentais – nuo 0 iki 100%;
- 8 bitais – sveikaisiais skaičiais nuo 0 iki 255. Naudojami kompiuterinėse sistemose;

- 16 bitų – sveikaisiais skaičiais nuo 0 iki 65535.

Kaip pavyzdys pateikta 1.1 lentelė, kurioje aprašoma pilnai apšviesta žalia spalva skirtingais būdais.

1.1. lentelė. Žalia spalva RGB spalvinėje erdvėje

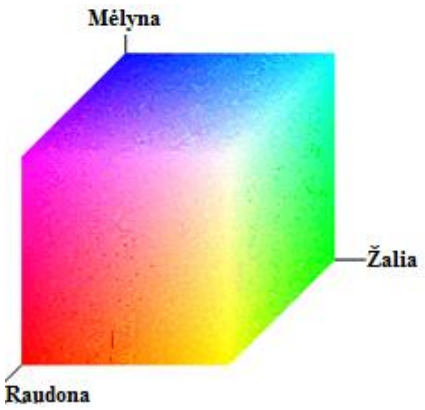
Būdas	Reikšmės
Aritmetiškai	(0.0, 1.0, 0.0)
Procentais	(0, 100, 0)
8 bitais	(0, 255, 0)
16 bitų	(0, 65535, 0)

Keičiant spalvos dedamųjų vertes gauname bet kokią spalvą iš spalvų spektro. Pagrindinių spalvų: baltos, geltonos, žydros, žalios, violetinės, raudonos, mėlynos bei juodos dedamųjų reikšmės RGB spalvinėje erdvėje yra pateikiamos 1.2 lentelėje, kai RGB spalvų erdvės dedamosios aritmetiškai kinta nuo 0,0 iki 1,0.

1.2 lentelė. Skirtingų spalvų dedamųjų vertės RGB erdvėje

	Dedamųjų intervalas	Balta	Geltona	Žydra	Žalia	Violetinė	Raudona	Mėlyna	Juoda
R	0,0..1.0	1,0	1,0	0,0	0,0	1,0	1,0	0,0	0,0
G	0,0..1,0	1,0	1,0	1,0	1,0	0,0	0,0	0,0	0,0
B	0,0..1,0	1,0	0,0	1,0	0,0	1,0	0,0	1,0	0,0

RGB spalvų erdvę galima atvaizduoti kaip trimatį kubą Dekarto koordinačių sistemoje (*žr.* 1.3 pav.). Spalvų vertės kinta nuo 0 iki maksimalios vertės pasirinktame intervale. Koordinačių pradžios taškas (0,0,0) aprašo juodą spalvą. Slenkant išilgai kiekviena ašimi šviesos intensyvumas didėja iki taško, kuris yra lygus maksimaliai vertei. Pavyzdžiui, jei intervalas yra nuo 0 iki 255, tai maksimali kubo kraštinė bus taške (255,255,255), kuris atitiks baltą spalvą.



1.3 pav. RGB spalvų erdvė [22]

RGB spalvinė erdvė yra plačiausiai naudojama, nes kompiuterių ekranuose, televizoriuose kiekvienas vaizdo taškas yra sudarytas iš trijų skirtingų spalvų: raudonos, žalios bei mėlynos. Tai leidžia supaprastini spalvų atvaizdavimą, nes fiksuojant vaizdą vaizdo kamera, fotoaparatu gautos RGB reikšmės kiekviename vaizdo taške yra tiesiogiai išvedamos ir nereikalingi jokie papildomi vaizdo apdorojimai, transformacijos. Problema iškyla, kai reikalinga atvaizduoti kuo realesnį vaizdą, nes RGB spalvinė erdvė aprašo tik kiekvienos komponentės spalvos kiekį, tačiau nesuteikia informacijos apie šviesos intensyvumą, spalvos toną, sotumą. Tai apsunkina vaizdo apdorojimą, nes, pavyzdžiui, siekiat pakeisti vaizdo spalvą reikalinga atlikti tris žingsnius:

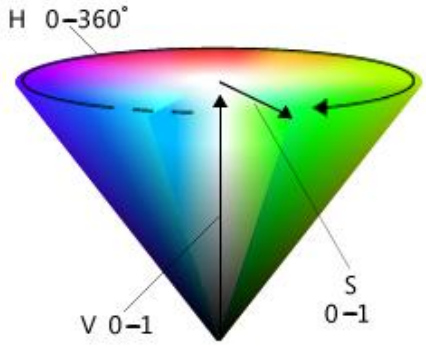
1. atpažinti vaizdo taško spalvos dedamųjų vertes;
2. pagal naujai parinktą spalvą, perskaičiuoti reikšmes visoms dedamosioms;
3. išsaugoti pakeistą vaizdą.

Vaizdo apdorojimas taptų daug lengvesnis, jei būtų galima tiesiogiai keisti spalvą ar spalvos intensyvumą nekeičiant visų spalvos parametrų. Dėl to vaizdo apdorojimui dažniau yra naudojamos kitos spalvų erdvės, kurios leidžia paprasčiau pakeisti spalvas, kitus parametrus.

1.4.2 HSV spalvinė erdvė

Siekiant palengvinti vaizdų apdorojimą 1970 metais buvo sukurta nauja spalvų sistema HSV, kuri leido supaprastinti spalvų aprašymą. Skirtingai nuo RGB spalvų erdvei atvaizduoti naudojamo kubo Dekarto koordinačių sistemoje, HSV erdvėje spalvos yra išdėstomos kūgio pavidalu.

HSV spalvų erdvė kaip ir RGB spalva yra apibūdinama 3 reikšmėmis [20] [23]. Šią sistemą galime nubrėžti kaip apverstą kūgį (*žr.* 1.4 pav.). Vertikali ašis atitinka spalvos apšviestumą (angl. V, *value*), kuo ji didesnė tuo spalva yra šviesesnė. Ši reikšmė kinta nuo 0 iki 100, kur 0 yra juoda spalva, o 100 – balta [21]. Pasirenkant kampą, nurodomas atspalvis (angl. H, *hue*), kuo siauresnis ruožas, tuo konkretesnė spalva yra pasirenkama. Spalvos gali kisti nuo 0 iki 360°. Apskritimo skersmuo apibūdina spalvos įsisotinimą (angl. S, *saturation*), kuo ji artimesnė vienetui, tuo spalva yra ryškesnė ir atvirkščiai, kuo ji artimesnė 0, tuo ji labiau išblukusi.



1.4 pav. HSV spalvų erdvė [24]

Renkantis spalvos atspalvį nereikia maišyti komponenčių verčių, kaip daroma RGB spalvinėje erdvėje, o pakanka pasirinkti norimą atspalvį, spalvos įsisotinimą ir apšviestumą. Keičiant spalvos dedamųjų vertes gauname bet kokią spalvą iš spalvų spektro. Pagrindinių spalvų: baltos, geltonos, žydros, žalios, violetinės, raudonos, mėlynos bei juodos dedamųjų reikšmės HSV spalvinėje erdvėje yra pateikiamos 1.3 lentelėje.

1.3 lentelė. Skirtingų spalvų dedamųjų vertės HSV erdvėje

	Spalvos intervalas	Balta	Geltona	Žydra	Žalia	Violetinė	Raudona	Mėlyna	Juoda
H	0°..360°	0	60	180	120	300	0	240	0
S	0..100	0	100	100	100	100	100	100	0
V	0..100	100	100	100	100	100	100	100	0

Iš vaizdo kameros gautas atvaizdas yra RGB erdvėje. Tačiau norint atlikti skaičiavimus HSV erdvėje reikalinga perskaičiuoti spalvas iš RGB į HSV erdvę. Verčiant iš RGB spalvos į HSV atliekami tokie skaičiavimai, kurie pateikti 1.0-1.8 formulėse.

$$R' = R/255\,, \tag{1.0}$$

$$G' = G/255, \tag{1.1}$$

$$B' = B/255, \tag{1.2}$$

$$C_{max} = \max(R', G', B'), \tag{1.3}$$

$$C_{min} = \min(R', G', B'), \tag{1.4}$$

$$\Delta = C_{max} - C_{min}\,, \tag{1.5}$$

$$H = \begin{cases} 60° * \left(\frac{G'-B'}{\Delta} mod 6\right), C_{max} = R' \\ 60° * \left(\frac{B'-R'}{\Delta} + 2\right), C_{max} = G' \\ 60° * \left(\frac{R'-G'}{\Delta} + 4\right), C_{max} = B' \\ 0° \qquad\qquad\qquad \Delta = 0 \end{cases}, \tag{1.6}$$

$$S = \begin{cases} 0, C_{max} = 0 \\ \frac{\Delta}{C_{max}}, C_{max} \neq 0 \end{cases}, \quad (1.7)$$

$$V = C_{max}. \quad (1.8)$$

čia *R* – originalaus vaizdo RGB spalvų erdvėje raudonos spalvos įvertis, kuris kinta nuo 0 iki 255; *G* – originalaus vaizdo RGB spalvų erdvėje žalios spalvos įvertis, kuris kinta nuo 0 iki 255; *B* – originalaus vaizdo RGB spalvų erdvėje mėlynos spalvos įvertis, kuris kinta nuo 0 iki 255; *H* – perversto vaizdo į HSV spalvų erdvę atspalvio dydis, kuris kinta nuo 0° iki 360°; *S* – perversto vaizdo į HSV spalvų erdvę įsisotinimo dydis, kuris kinta nuo 0 iki 1; *V* – perversto vaizdo į HSV spalvų erdvę spalvos apšviestumo dydis, kuris kinta nuo 0 iki 1.

*1.4.3 L*a*b spalvinė erdvė*

L*a*b spalvinę erdvę sukūrė Richard S. Hunter 1948m. Ją sudaro taip pat trys kintamieji, kurie aprašo:

- spalvos šviesumą (angl. L, *lightness*);
- „a“ parametras apibūdina spalvas: raudoną ir žalią;
- „b“ apibūdina spalvas: geltoną ir mėlyną.

Kai spalvos šviesumas L lygus nuliui gaunama balta spalva, o kai L=128 – juoda. Kai parametras „a“ lygus 100 gaunama raudona spalva, o kai a= -100 – žalia. Kai „b“ parametras lygus 100 gaunama mėlyna spalva, o kai b= -100 – geltona (*žr*. 1.5 pav.).

Norint parinkti tam tikrą atspalvį reikalinga parinkti a ir b komponenčių dydžius taip, kad gautume norimą spalvą bei parinkti L dydį, kuris nurodytų atspalvio šviesumą [21]. Siekiant apdoroti skaitmeninį vaizdą RGB formatu reikalinga atlikti spalvinės erdvės pakeitimą į L*a*b, tam naudojamos formulės:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}412453 & 0{,}357580 & 0{,}180423 \\ 0{,}212671 & 0{,}715160 & 0{,}072169 \\ 0{,}019334 & 0{,}119193 & 0{,}950227 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \quad (1.9)$$

$$L = \begin{cases} 116 * \sqrt[3]{Y} - 16, kai\ Y > 0{,}008856 \\ 903{,}3 * Y, \quad kai\ Y \leq 0{,}008856 \end{cases}, \quad (1.10)$$

$$X_t = \frac{X}{0{,}950456}, \quad (1.11)$$

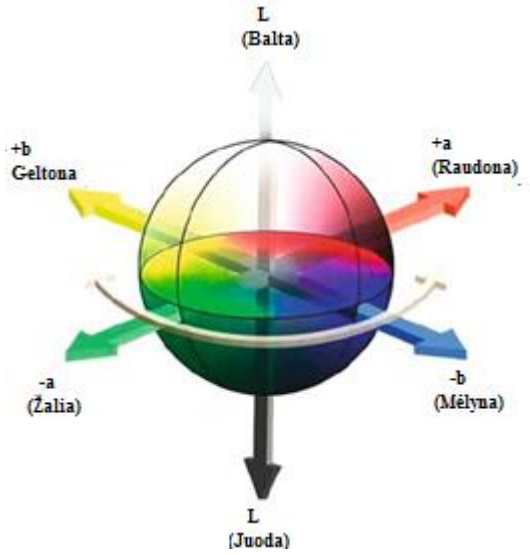
$$Z_t = \frac{Z}{1{,}088754}, \quad (1.12)$$

$$a = 500(f(X) - f(Y)), \quad (1.13)$$

$$b = 200(f(X) - f(Z)), \quad (1.14)$$

$$f(t) = \begin{cases} \sqrt[3]{t}, & kai\ t > 0{,}008856 \\ 7{,}787t + 0{,}137931, & kai\ t \leq 0{,}008856' \end{cases} \quad (1.15)$$

čia *R* – originalaus vaizdo RGB spalvų erdvėje raudonos spalvos įvertis, kuris kinta nuo 0 iki 1; *G* – originalaus vaizdo RGB spalvų erdvėje žalios spalvos įvertis, kuris kinta nuo 0 iki 1; *B* – originalaus vaizdo RGB spalvų erdvėje mėlynos spalvos įvertis, kuris kinta nuo 0 iki 1; *L* – perversto vaizdo į L*a*b spalvų erdvę šviesumo dydis, kuris kinta nuo 0 iki 100; *a* – perversto vaizdo į L*a*b spalvų erdvę raudoną ir žalią spalvos apibūdinantis dydis, kuris kinta nuo -127 iki 127; *b* – perversto vaizdo į L*a*b spalvų erdvę geltoną ir mėlyną spalvas apibūdinantis dydis, kuris kinta nuo -127 iki 127.



1.5 pav. L*a*b spalvų erdvė [17]

Pagrindinių spalvų: baltos, geltonos, žydros, žalios, violetinės, raudonos, mėlynos bei juodos dedamųjų reikšmės L*a*b spalvinėje erdvėje yra pateikiamos lentelėje 1.4.

1.4 lentelė. Skirtingų spalvų dedamųjų vertės L*a*b erdvėje

	Spalvos intervalas	Balta	Geltona	Žydra	Žalia	Violetinė	Raudona	Mėlyna	Juoda
L	0,0..1,0	100	97,14	91,12	87,74	60,32	53,23	32,3	0
*a	-127..127	0	-21,56	-48,08	-86,18	98,25	80,11	79,2	0
*b	-127..127	0	94,48	-14,14	83,18	-60,84	67,22	-107,86	0

1.4.4 YCbCr spalvinė erdvė

YCbCr spalvų modelis – tai monochromatinis spalvų modelis, kuriame yra apšviestumo informacija bei informacija apie spalvas [20]. *Y* dydis koduoja apšviestumą, *Cb* – mėlynos spalvos

kiekį, *Cr* – raudonos spalvos kiekį. Šis spalvų modelis naudojamas vaizdų apdorojimo algoritmuose, nes galima nesunkiai pašalinti apšviestumo įtaką. Tai labai padeda, kai norima apdoroti vaizdus, kuriuose skiriasi apšvietimo lygis.

Keitimas iš RGB spalvų modelio į YCbCr atliekamas formulėmis:

$$Y = 16 + \frac{65{,}738*R}{256} + \frac{129{,}057*G}{256} + \frac{25{,}064*B}{256}, \quad (1.16)$$

$$Cb = 128 - \frac{37{,}945*R}{256} - \frac{74{,}494*G}{256} + \frac{112{,}439*B}{256}, \quad (1.17)$$

$$Cr = 128 + \frac{112{,}439*R}{256} - \frac{94{,}154*G}{256} - \frac{18{,}285*B}{256}, \quad (1.18)$$

čia *R* – originalaus vaizdo RGB spalvų erdvėje raudonos spalvos įvertis, kuris kinta nuo 0 iki 255; *G* – originalaus vaizdo RGB spalvų erdvėje žalios spalvos įvertis, kuris kinta nuo 0 iki 255; *B* – originalaus vaizdo RGB spalvų erdvėje mėlynos spalvos įvertis, kuris kinta nuo 0 iki 255; *Y* – perversto vaizdo į YCbCr spalvų erdvę apšviestumo dydis, kuris kinta nuo 0 iki 1; *Cb* – perversto vaizdo į YCbCr spalvų erdvę mėlynos spalvos kiekis, kuris kinta nuo -0.5 iki 0.5; *Cr* – perversto vaizdo į YCbCr spalvų erdvę raudonos spalvos kiekis, kuris kinta nuo -0.5 iki 0.5.

Pagrindinių spalvų: baltos, geltonos, žydros, žalios, violetinės, raudonos, mėlynos bei juodos dedamųjų reikšmės YCbCr spalvinėje erdvėje yra pateikiamos lentelėje 1.5.

1.5 lentelė. Skirtingų spalvų dedamųjų vertės YCbCr erdvėje

	Spalvos intervalas	Balta	Geltona	Žydra	Žalia	Violetinė	Raudona	Mėlyna	Juoda
Y	0,0..1,0	1	0,89	0,7	0,59	0,41	0,3	0,1	0
Cb	-0,5..0,5	0	0,08	-0,5	-0,42	0,42	0,5	-0,08	0
Cr	-0,5..0,5	0	-0,5	0,17	-0,33	0,33	-0,17	0,5	0

1.5 Srities išskyrimas

Norimo objekto išskyrimas (angl. *keying*) iš duoto vaizdo yra vienas plačiausiai naudojamų metodų vaizdų apdorojime. Šio metodo esmė, jog reikiamas objektas yra išskiriamas iš fono, o fonas tuo metu yra užmaskuojamas. Tai leidžia greičiau apdoroti vaizdą, nes nereikia apdoroti objektų fone. Gautas rezultatas yra dvimatės formos – atpažinti, išrinkti vaizdo taškai atvaizduojami kaip vienetas arba balta spalva, o fonas – juodas ir turi reikšmę 0.

1.5.1 Intensyvumo išskyrimo metodas

Pats paprasčiausias fono išskyrimo metodas yra išskiriant vaizdą pagal spalvų intensyvumą. Vaizdas yra padalijamas į dvi dalis: šviesius ir tamsius atspalvius. Atskyrimo riba yra nustatoma pagal tai kokio šviesumo/tamsumo objektas yra.

Jei norima išskirti objektą iš spalvoto vaizdo, kurį sudaro RGB spalvos, tai jis yra perverčiamas į monochromatinį vaizdą, kurį sudaro tik juoda bei balta spalvos bei pilki atspalviai. Tada parinkus ribas, kuriose yra norimas objektas, visi taškai kurie tenkina šias ribas yra išryškinami ir jiems priskiriama balta spalva, o visiems likusiems juoda. Taip gaunamos išskirtos sritys, kuriose yra norimas objektas, o neaktualios vietos užmaskuojamos.

Kadangi vaizdo jutiklis fiksuoja vaizdą RGB spalvų erdvėje, pirmiausia reikia persiversi vaizdą iš spalvoto į monochromatinį. Tai atliekama naudojant formulę, kurioje skaičiuojamas dydis I:

$$I = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B, \quad (1.19)$$

čia I – monochromatinio vaizdo taško reikšmė, perskaičiavus iš RGB spalvų erdvės; R, G, B – RGB spalvų modelio parametrų vertės nuo 0 iki 255. Gautas skaičius I yra įrašomas vietoj esamų R, G ir B dydžių, taip gaunant monochromatinį vaizdą.

1.5.2 Spalvos išskyrimo metodas

Spalvotame vaizde pagrindiniai spalvų parametrai yra intensyvumas, spalvos tonas bei sotumas. Vaizdo signalas gali būti išskaidytas į atskirus intensyvumo ir spalvos tono, sotumo kanalus. Pavyzdžiui RGB spalvinėje erdvėje spalvos gali būti išskaidytos į 3 kanalus. Kiekvienas kanalas parodo kiek atitinkamame vaizde yra raudonos, žalios ar mėlynos spalvos. Šie kanalai leidžia išskirti norimą spalvą iš kitų. Pasirinkus raudonos spalvos kanalą, gaunamas monochromatinis vaizdas parodantis kiek raudonos spalvos yra konkrečioje vietoje.

Analogiškai spalvų išskyrimas galimas ir kitose spalvinėse erdvėse, panaudojant jų privalumus. RGB spalvų erdvės pagrindinis trūkumas, tai kad sudėtinga aprašyti skirtingus vienos spalvos atspalvius, nes keičiantis atspalviui keičiasi visi trys spalvų dydžiai, o naudojant HSV spalvinę erdvę keičiasi tik vienas – atspalvis H. Dėl to yra daug paprasčiau išskirti spalvą ir jai artimus atspalvius, nes dažnai objektas apšviestas skirtingu kampu gali matytis kaip skirtingo atspalvio. Dėl to įvedus norimą toleranciją galima tiksliau išskirti spalvas.

Spalvos išskyrimui galime naudoti formulę:

$$I = \begin{cases} 0, & H < (H_i - H_t), \\ 1, & (H_i - H_t) \leq H \leq (H_i + H_t), \\ 0, & H < (H_i + H_t), \end{cases} \quad (1.20)$$

čia I – vaizdo taško vertė, atlikus spalvos išskyrimą. Vertė gali būti 1 arba 0; H – originalaus vaizdo HSV spalvinėje erdvėje atspalvio dydis; H_i – išskiriamo vaizdo HSV spalvinėje erdvėje atspalvio dydis; H_t – išskiriamo vaizdo HSV spalvinėje erdvėje atspalvio dydžio tolerancija.

Šis metodas nėra tinkamas surasti objekto kraštines, dėl to reikalingi kiti ar papildomi metodai siekiant gauti kokybiškesnį rezultatą. Nes taikant šį metodą, gaunamos susiliejusios kraštinės, dėl skirtingų spalvos atspalvių, bet tinkamas gauti bendram rezultatui.

1.5.3 Vaizdo skirtumo metodas

Skirtuminio vaizdo gavimo metodas veikia kitokiu principu nei ankščiau paminėti srities išskyrimo metodai. Šio metodo veikimo principas paremtas tuo, kad objektas yra išskiriamas naudojant dviejų paeiliui einančių arba originalo ir šabloninio vaizdo skirtumą [20]. Šis metodas atskiria foną, o ne vaizdą kaip spalvos išskyrimo atveju, tai leidžia atskirti naujai atsiradusi objektą fone, kurio prieš tai nebuvo. Fonas atskiriamas, kai šabloninio ar prieš tai buvusio vaizdo taškų spalvų vertės sutampa su esamuoju vaizdu ir sutapusios vertės yra pašalinamos, o skirtingos paliekamos, taip išskiriant naujai atsiradusį objektą [25]. Fono pašalinimas atliekamas naudojant formulę kiekvienam vaizdo taškui:

$$O = n - |I_1 - I_2| , \quad (1.21)$$

čia I_1 ir I_2 – paveikslėlių vaizdo taškų reikšmės, O – vaizdo taškų skirtumas, n – maksimali galima vaizdo taško vertė.

Šis metodas tinkamas taip pat kai norima pašalinti iš vaizdo taškus, kurie buvo atrinkti įvairiais paieškos metodais ir palikti tik tuos objektus, kurie yra naudingi.

1.6 Objekto padėties nustatymas

PET plastiko dribsnio padėties erdvėje nustatymui reikalinga surasti tik netinkamų dribsnių apytikslę padėtį, kad juos vėliau būtų galima pašalinti iš bendro srauto. Šiai užduočiai atlikti nereikalingi metodai, kurie ieškotų objekto posūkio kampo, mastelio, išskirtinių savybių, o užtenka surasti objekto centrą. Visi ankščiau aptarti srities išskyrimo metodai gražina vaizdą dvimatėje plokštumoje, dėl to galima naudoti pozicijos paieškos metodus, tinkamus veikti dvimatėje erdvėje:

- nuoseklios paieškos,
- fono užpildymo,
- dvimačių didelių objektų paieškos metodas (angl. BLOB, *binary large object*).

1.6.1 Nuoseklios paieškos metodas

Naudojant nuoseklios paieškos metodą, paveikslėlio kiekvienas vaizdo taškas yra patikrinamas atskirai. Tikrinimas pradedamas iš viršaus į apačią vieno vaizdo taško pločio stulpeliu. Kai vaizdo taško vertė yra lygi 1 (balta spalva), jam yra priskiriama etiketė, jog tai yra naujas objektas. Naujam objektui yra priskiriami visi kiti sekantys taškai, kurie taip pat yra lygūs vienetui. Pasiekus apačią yra tikrinamas sekantis stulpelis, vėliau kitas ir taip kol patikrinamas visas paveikslėlis. Jei buvo surastas vaizdo taškas, kuriam iš kairio krašto nebuvo jokių taškų, kurie lygus 1, tai yra priskiriamas nauja etiketė.

Šio metodo veikimo algoritmas [26]:

1. tikrinami visi paveikslėlio vaizdo taškai;
2. priskiriamos lokalios laikinos etiketės taškams ir įrašomos lokalios pozicijos į lentelę;
3. surandamos ekvivalentinės klasės;
4. pervadinamos objektų etiketės, pagal surastas ekvivalentines klases.

MATLAB programinėje įrangoje yra modifikuota šio algoritmo versija aprašoma *bwlabel* [27] funkcija, kurioje paieška vykdoma taip pat, tačiau sprendimas apie skirtingas etiketes yra priimama iš karto, naudojant sąjungos paieškos (angl. *Union-find*) algoritmą. Norint nuskaityti objekto centro koordinates reikalinga naudoti funkciją *regionprops,* kuri apskaičiuoja ir gražina atskirų objektų pasirinktas savybes: plotą, orientaciją erdvėje, perimetrą, centro koordinates.

1.6.2 Užpildymo algoritmas

Užpildymo algoritmas (angl. *Flood fill*) yra naudojamas kai norima surasti objekto taškus, kuris jungiasi vienas su kitu.

Veikimo algoritmas susideda iš 3 dalių [20]:

1. ieškomas dar nepažymėtas vaizdo taškas p;
2. naudojant užpildymo algoritmą sužymimi visi vaizdo taškai kurie priklauso taškui p.
3. kartojami 1 ir 2 žingsniai kol bus sužymėti visi taškai.

MATLAB šis algoritmas įgyvendintas naudojant funkciją *bwconncomp*. Ši funkcija gražina duomenis apie surastų objektų skaičių ir kokie vaizdo taškai sudaro kiekvieną objektą. Norint sužinoti objekto centrą reikia naudoti MATLAB funkciją *regionprops*.

1.6.3 Dvimačių didelių objektų atpažinimas

Paskutinis nagrinėjamas metodas yra BLOB (angl. *Binary Large Object*) paieška. Šis metodas skirtas išskirti aiškiai matomus objektus iš fono ir pritaikomas įvairioms užduotims [28].

BLOB veikimas susideda iš trijų etapų:

- išskyrimas – vaizdas išskiriamas, naudojant slenkstines vertes ieškomam objektui;
- vaizdo patobulinimas – išskirtas vaizdas turi įvairių pašalinių triukšmų, dėl to naudojami vaizdo apdorojimo filtrai jiems pašalinti;
- analizavimas – nufiltruotas vaizdas yra išmatuojamas, nustatomos koordinatės ir išsaugojamos atmintyje.

Vaizdo išskyrimas gali būti dviejų rūšių: paremtas šviesumo arba spalvos išskyrimu taip pat jis gali būti tiek statinis tiek dinaminis. Statiniame vaizde išskiriama viena konkreti spalva, o dinaminėje kelios, kintančių atspalvių.

Koreguojant vaizdą naudojami įvairūs metodai:

- vaizdo išplėtimas (angl. *dilation*), kai pridedami papildomi vaizdo taškai kraštinėse, siekiat užpildyti tuštumas;
- vaizdo sutraukimas (angl. *erosion*), atvirkštinis veiksmas vaizdo išplėtimui, kai yra ne pridedami vaizdo taškai, bet atimami;
- vaizdo uždarymas (angl. *closing filter*) – užpildo vidines tuštumas, esančias šalia kraštų. Veikimo rezultatas panašus į tą kurį gautume pirma vaizdui pritaikius vaizdo išplėtimą ir paskui pritaikius vaizdo sutraukimą, tai leidžia išsaugoti kraštinę ir užpildyti tuštumas;
- vaizdo atidarymas (angl. *opening filter*) – atvirkštinis filtras vaizdo uždarymui, kai pirma pritaikomas vaizdo sutraukimas, o paskui vaizdo išplėtimas.

Analizuojant vaizdą yra surandamas plotas, kurį užima išskirti vaizdo taškai, perimetro ilgis, formos statmenumas, apvalumas, tankumas, tuštumų skaičius bei kiti parametrai apibūdinantis objekto formą, dydį.

2. APARATŪROS APRAŠYMAS

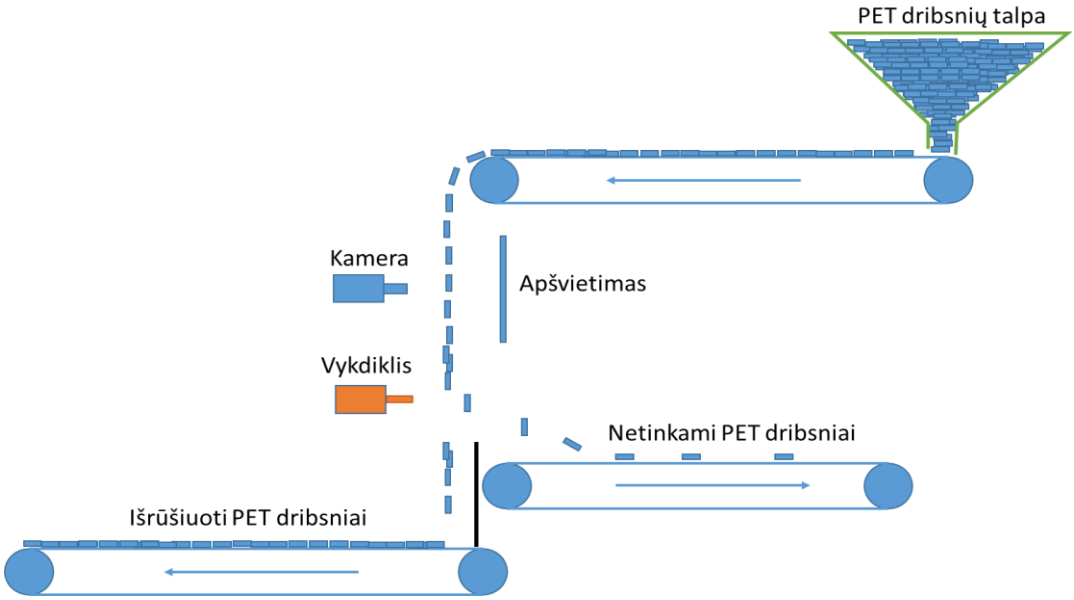
2.1 Naudojama eksperimentinė įranga

Išsiaiškinti, kuri spalvinė erdvė geriausiai tinka naudoti PET dribsnių rūšiavimui buvo atliktas eksperimentas. Eksperimentas buvo atliekamas laboratorijoje, kurioje naudojant vaizdo kamerą buvo fiksuojami laisvai krentantys PET plastiko dribsniai nuo konvejerio.

Tyrime buvo naudojama kompanijos „Pointgrey“ kamera „Blackfly BFLY-U3-03S2C-CS“ (*žr.* 2.1 pav.) kartu su objektyvu Fujinon 1:1.2/2.8-8mm. Eksperimentinių duomenų fiksavimui ir duomenų išsaugojimui buvo naudojama „Pointgrey“ programinė įranga „FlyCapture2“. Duomenų apdorojimui buvo naudojama programinė įranga „MATLAB R2016a“ bei kompiuteris su Intel Core i3-2310M 2.1Ghz procesoriumi ir su 6 Gb operatyviosios atminties.

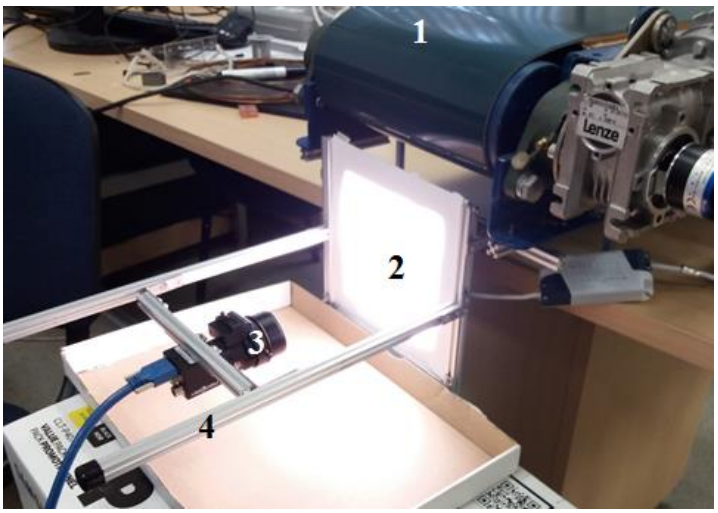
2.1 pav. „Pointgrey“ kompanijos vaizdo kamera „Blackfly BFLY-U3-03S2C-CS“

Eksperimentas buvo atliekamas imituojant 2.2 paveiksle pateiktą PET plastiko spalvinio rūšiavimo sistemą. Sistemą sudaro PET dribsnių talpa iš kurios dribsniai plonu sluoksniu yra paduodami ant konvejerio. Konvejerio pabaigoje dribsniai krenta žemyn veikiami žemės traukos. Kritimo metu su vaizdo kamera yra fiksuojami PET dribsniai. Kameros darbui ir vaizdo kokybei pagerinti naudojamas LED apšvietimas. Apšvietimas įdiegtas už krentančių dribsnių, tai leidžia peršviesti skaidrius dribsnius ir išryškinti visus neskaidrius objektus kameros matomame vaizde. Aptikti netinkami PET dribsniai yra išrūšiuojami naudojant eile išdėstytus oro nupūtėjus, kurie suveikia gavę signalą apie blogą PET dribsnį ir juos nupučia. Atrinkti geri dribsniai nukrenta žemyn ir nuvažiuoja vienu konvejeriu, o blogi nupučiami ir nuvažiuoja kitu.



2.2 pav. PET plastiko spalvinio rūšiavimo sistema

Šio darbo tikslas – surasti metodus plastiko dalelių spalvos atpažinimui ir jų padėties erdvėje nustatymui, tai atliekant eksperimentą laboratorijoje buvo surinktas paprastesnis rūšiavimo sistemos modelis (*žr.* 2.3 pav.) be vykdymo įrenginių ir kuriame yra naudojama: 1 – įvežimo konvejeris, 2 – šviesos šaltinis, 3 – kamera bei 4 – kameros laikiklis. Juostinio konvejerio greitis gali būti reguliuojamas naudojant dažnio keitiklį. Prie konvejerio buvo sumontuotas aliuminio profilio rėmas, kurio paskirtis – išlaikyti šviesos šaltinį, kamerą bei užtikrinti, kad kameros atstumas nuo konvejerio visą laiką bus pastovus. Apšvietimui buvo panaudotas 12V šviesos diodų (angl. LED; *Light - emitting diode*) natūralios, dienos šviesos šviestuvas, kurio spalvinė temperatūra yra 4000-4500 K. LED šviestuvas buvo pasirinktas būtent tokios spalvos, nes esant baltai šviesai yra mažiau spalvų iškraipymų vaizdo apdorojimo sistemose. Siekiant tolygaus, ne taškinio apšvietimo, šviestuvo stiklas parinktas difuzinis.



2.3 pav. Eksperimentinė įranga

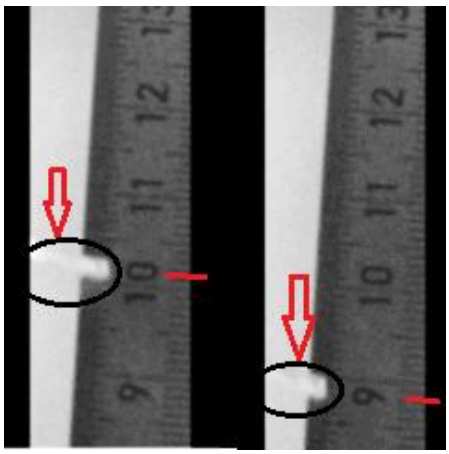
Eksperimentui atlikti buvo naudojami įvairių spalvų PET dribsniai (*žr.* 2.4 pav.), kurie buvo gauti iš plastiko perdirbimo įmonės. Išrūšiavus dribsnius buvo atrinkti aštuonių spalvų dribsniai, kurie sudarė didžiausią dalį visų dribsnių. Didžiausią dribsnių dalį sudarė balti skaidrūs plastikai, kiek mažiau pasitaikė rudos spalvos. Taip pat buvo raudonos, mėlynos, žalios, melsvos, rožinės bei geltonos spalvos dribsnių.

2.4 pav. Eksperimentui panaudotos PET dribsnių spalvos

2.2 Eksperimentinių duomenų gavimas

Esminis dalykas norint atlikti vaizdo analizę yra laiku pamatyti ir užfiksuoti objektą, kurį norima išanalizuoti. Dėl to reikalinga parinkti tokį filmavimo dažnį, kuris leistų užfiksuoti objektą bent 3 kartus, vaizdo kameros matomame plote. Tai reikalinga dėl to, kad dribsnis būtų užfiksuotas bent viename vaizdo kadre. Jeigu dažnis būtų parinktas toks, kad dribsnis būtų matomas tik 2 kartus, tai yra tikimybė, kad abiem fiksavimo momentais dribsnis bus ant matomo vaizdo krašto ir bus sudėtinga, gauti pilną dribsnio vaizdą.

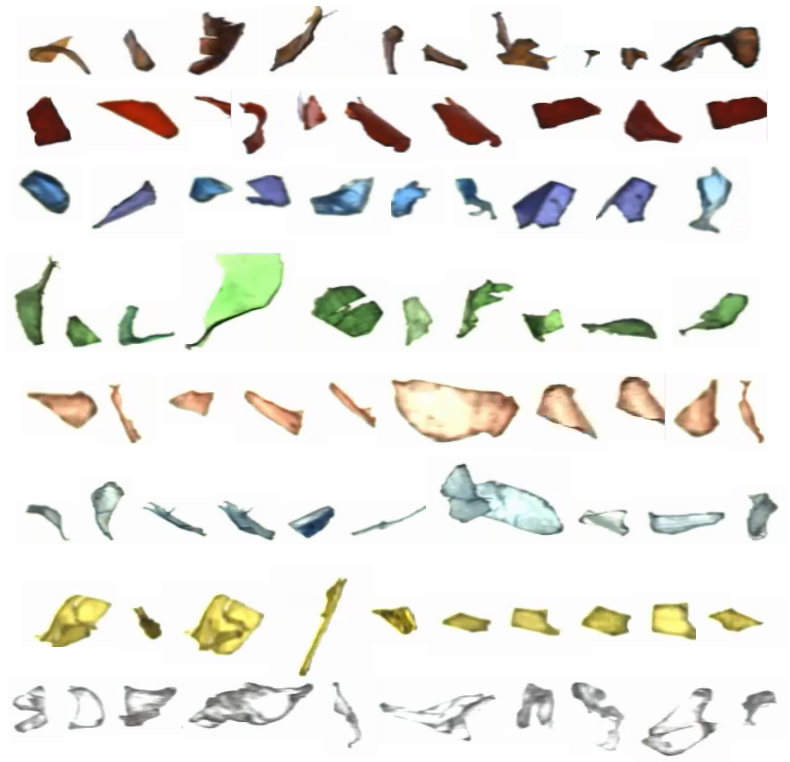
Eksperimento metu su vaizdo kamera filmuojant 128 kadrų per sekundę greičiu buvo išmatuota, kad objektas per 1 vaizdo kadrą nukrenta 12mm (*žr.* 2.5 pav.). Vadinasi, reikalinga, kad vaizdo kameros matymo lauke, būtų matomas ruožas lygus bent 36mm aukščio, kuris leistų užfiksuoti krentantį objektą 3–uose kadruose, kai filmavimo dažnis yra 128 kadrai per sekundę.



2.5 pav. PET dribsnio kritimo atstumas per 1/128 sekundžių

Kadangi kameros vaizdo fiksavimo dažnis priklauso nuo fiksuojamo vaizdo dydžio ir reikalinga stebėti tik riboto aukščio erdvę, pasirinktas maksimalus leidžiamas 648 vaizdo taškų plotis, kuris leistų maksimaliai išnaudoti kameros galimybes. Bandymo metu buvo naudojamas šviestuvas, kurio plotis 12cm ir norint, kad būtų pilnai jis išnaudojamas parinktas toks kameros atstumas nuo konvejerio, kad būtų matomas maksimalus vaizdo plotis. Taip gauta, kad matuojamo vaizdo plotis yra 10,5 cm. Aukštis parinktas įvertinant santykį, jog 10,5cm atitinka 648 vaizdo taškus, tai 3,6cm atitiks 222 vaizdo taškus. Pasirinkus šį dydį buvo gautas 134 kadrų per sekundę greitis, kuris tenkina pradines sąlygas ir viršija reikalingus 128 kadrus per sekundę.

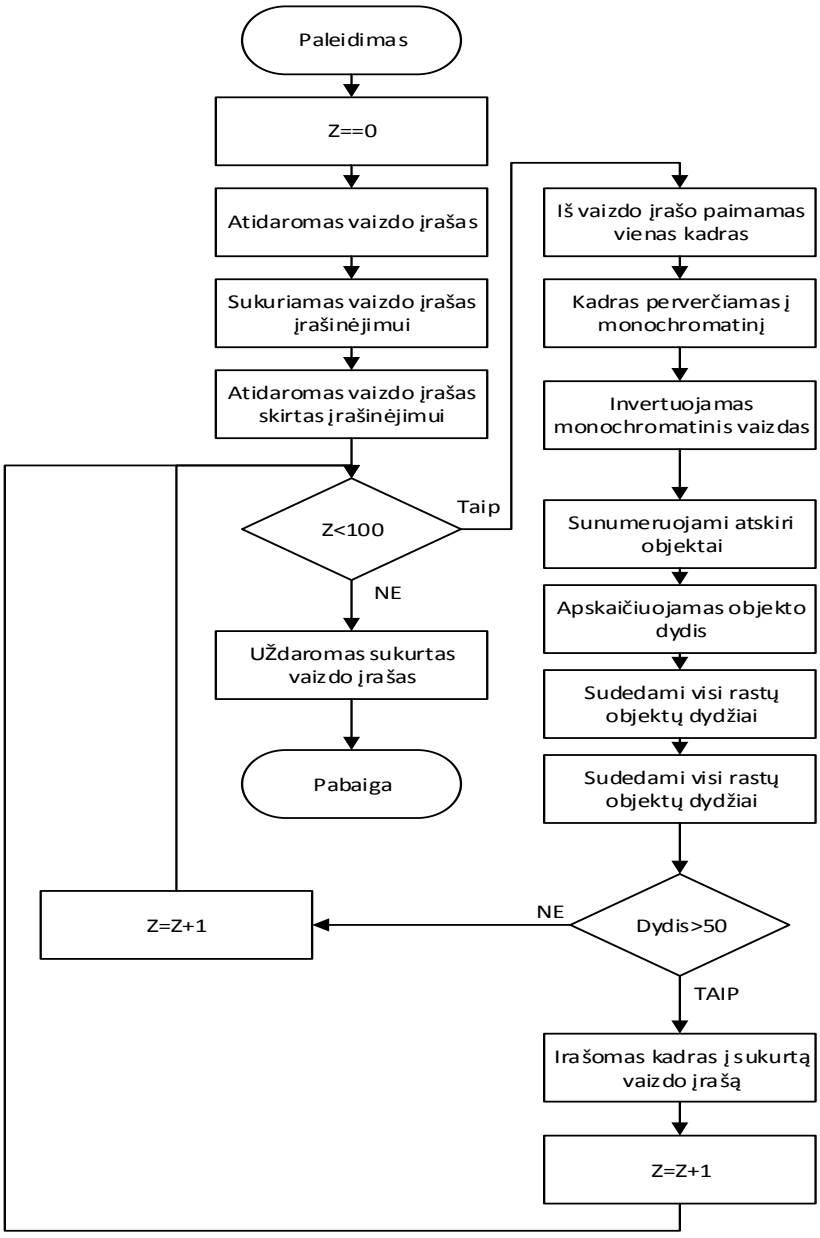
Eksperimento metu buvo pilami PET dribsniai ant besisukančio konvejerio ir įrašinėjamas vaizdo įrašas „mp4“ formatu, 128 kadrų per sekundę greičiu. Bandymo metu buvo fiksuojami visų turimų spalvų PET dribsniai (*žr.* 2.6 pav.): rudi, raudoni, mėlyni, žali, rožiniai, žydri, geltoni bei skaidrūs ir kurie bus naudojami sekančiuose bandymuose.



2.6 pav. Skirtingų spalvų PET dribsniai užfiksuoti su kamera

Gauti eksperimentiniai duomenys papildomai buvo apdoroti su MATLAB programine įranga tam, kad būtų pašalinti tušti vaizdo kadrai ir palikti tik eksperimentui naudingi duomenys. Tam buvo pasitelktas intensyvumo išskyrimo metodas, kuris leido iš originalaus vaizdo įrašo gauti tik tuos vaizdo kadrus, kurie teikia informacijos. Programos veikimo algoritmas pateiktas 2.7 paveiksle, o programas kodas pateiktas priede Nr. 1. Programos veikimas pagristas spalvos paieška pagal šviesos intensyvumą. Visi taškai kurie yra žemiau slenkstinės vertės yra paverčiami 0 (juoda

spalva), o likę taškai – 1 (balta spalva). Jeigu viename vaizdo kadre yra daugiau nei 50 vaizdo taškų, kadras yra išsaugojamas į naują vaizdo įrašą, jeigu mažesnis – atidaromas ir tikrinamas sekantis kadras. Apdorojus duomenis, gautas rezultatas – vaizdo įrašas, kuriame yra 100 vaizdo kadrų, kuriuose nėra kadrų, kuriuose bendras aptiktų objektų plotas mažesnis už 50 vaizdo taškų.



2.7 pav. Gautų duomenų apdorojimas, tolimesniems eksperimentams

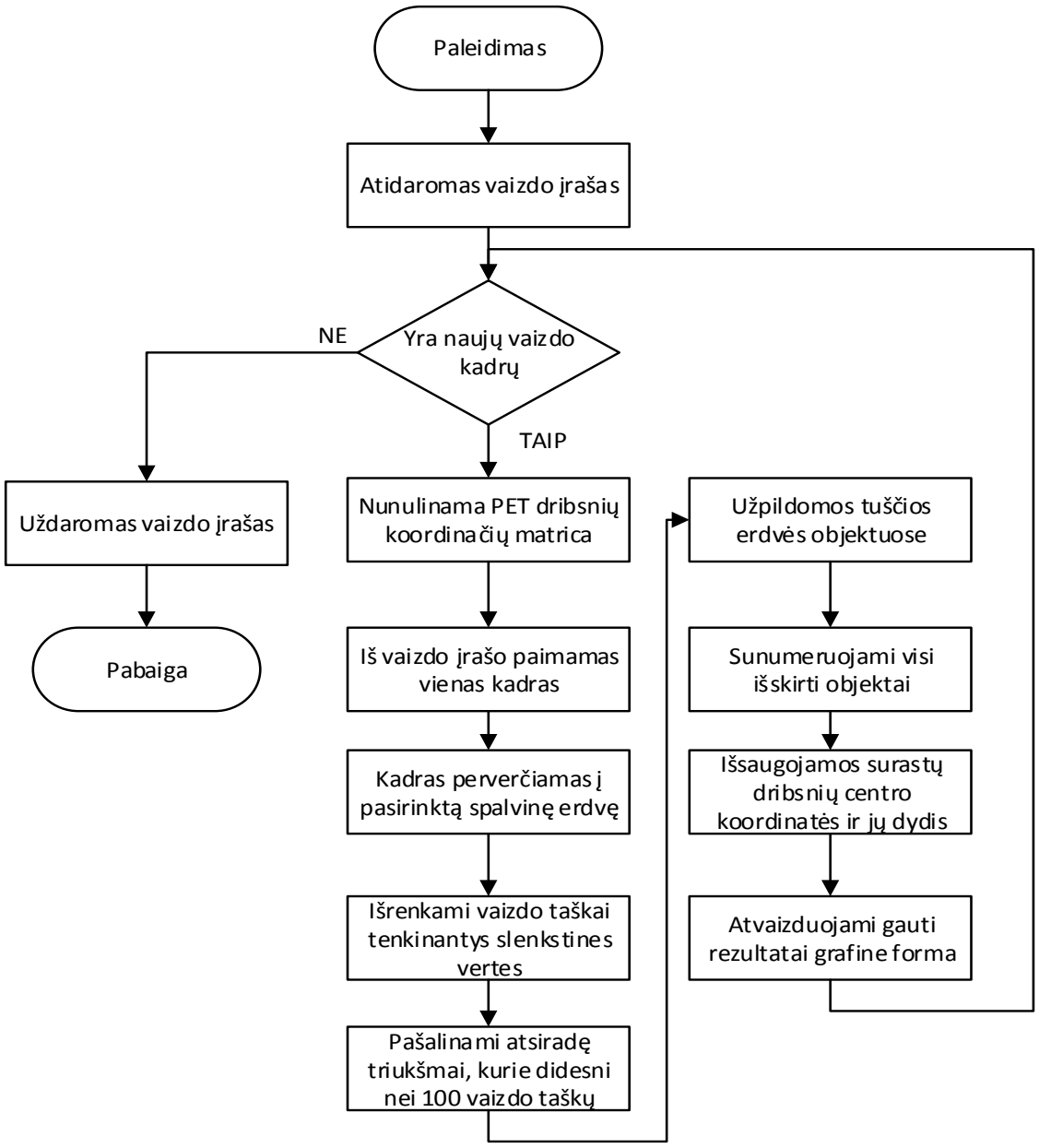
3. SISTEMOS VEIKIMO ALGORITMAS

Skaidrių PET dribsnių išskyrimui iš įvairiaspalvių dribsnių srauto buvo sudarytas algoritmas MATLAB programiniam paketui, kuris pateiktas 3.1 paveiksle bei parašytas programos kodas, kuris pateikiamas priede Nr. 2. Eksperimento metu bandymai atliekami su įrašytu vaizdo įrašu, tačiau algoritmas lengvai pritaikomas ir tiesiogiai transliuojam vaizdui. Vienintelis skirtumas, tai kad vietoj vaizdo įrašo atidarymo reikia nuskaityti tiesioginį vaizdą.

Programos veikimo žingsniai:

- Paleidus programą atidaromas vaizdo įrašas, kuris bus tiriamas. Tai atliekama su MATLAB komanda „vaizdo_irasas = *vision.VideoFileReader('visos_spalvos.mp4')*“, kuri leidžia nuskaityti pasirinktą vaizdo įrašą .avi, .mpeg, .mp4, .wmv bei kitais formatais, kuriuos palaiko Microsoft DirectShow® 9.0. Šiuo atveju testuojamas įrašas yra „visos_spalvos.mp4“ tai jis ir atidaromas.
- Atidarius vaizdo įrašą tikrinama ar yra naujų vaizdo kadrų, tai atliekame su funkcija „*while ~isDone(vaizdo_irasas)*“. Jeigu vaizdo kadrų nebėra ciklas užbaigiamas ir uždaromas atidarytas vaizdo įrašas „visos_spalvos.mp4“ su MATLAB komanda „*release(vaizdo_irasas)*“.
- Jeigu yra naujų vaizdo kadrų, ciklas tęsiamas toliau ir išvaloma matrica „*objektai=[]*;“, kurioje išsaugojamos rastų objektų koordinatės.
- Išvalius matricą, nuskaitomas vienas vaizdo kadras iš vaizdo įrašo su komanda „*kadras = step(vaizdo_irasas)*“. Funkcija „step“ nuskaito sekantį vaizdo kadrą iš įrašo.
- Jei vaizdas apdorojamas RGB spalvinėje erdvėje, tai transformacijos atlikti nereikia, tačiau atliekant bandymus su HSV, L*a*b bei YCbCr spalvinėmis erdvėmis transformaciją atlikti būtina. Iš RGB verčiant į HSV spalvinę erdvę naudojama funkcija „*rgb2hsv*“, verčiant į YCbCr – „*rgb2ycbcr*“, o į L*a*b – funkcija „*rgb2lab*“.
- Išrenkami vaizdo taškai kurie atitinka slenkstines vertes. Slenkstinės vertės randamos eksperimento metu. Išrinkimas vykdomas lyginant matricos spalvų vertes su slenkstinėmis.
- Vykdomas vaizdo taškų pašalinimas, kurių dydis mažesnis nei 100 vaizdo taškų su komanda „*bwareaopen*“, kuri suranda susijungusius tarpusavyje vaizdo taškus ir jei jų plotas mažesnis nei 100 yra pašalinami iš vaizdo.
- Jeigu yra, užpildomos tuščios erdvės surastuose objektuose su funkcija „*imfill(filtravimas_po_skirtumo,'holes')*;“. Užpildomos tiks tos skylės kurias iš visų pusių gaubia juoda spalva.

- Atpažįstami ir išskiriami skirtingi objektai monochromatiniame vaizde. Galimi metodai MATLAB programoje: nuoseklios paieškos, fono užpildymo, dvimačių didelių objektų paieškos metodas.
- Į matricą „*objektai=[]*;“ įrašomi surastų objektų centro koordinatės, dydis.
- Gauti rezultatai atvaizduojami naudojant funkciją „plot“.



3.1 pav. Sistemos veikimo algoritmas

4. EKSPERIMENTINIAI TYRIMAI

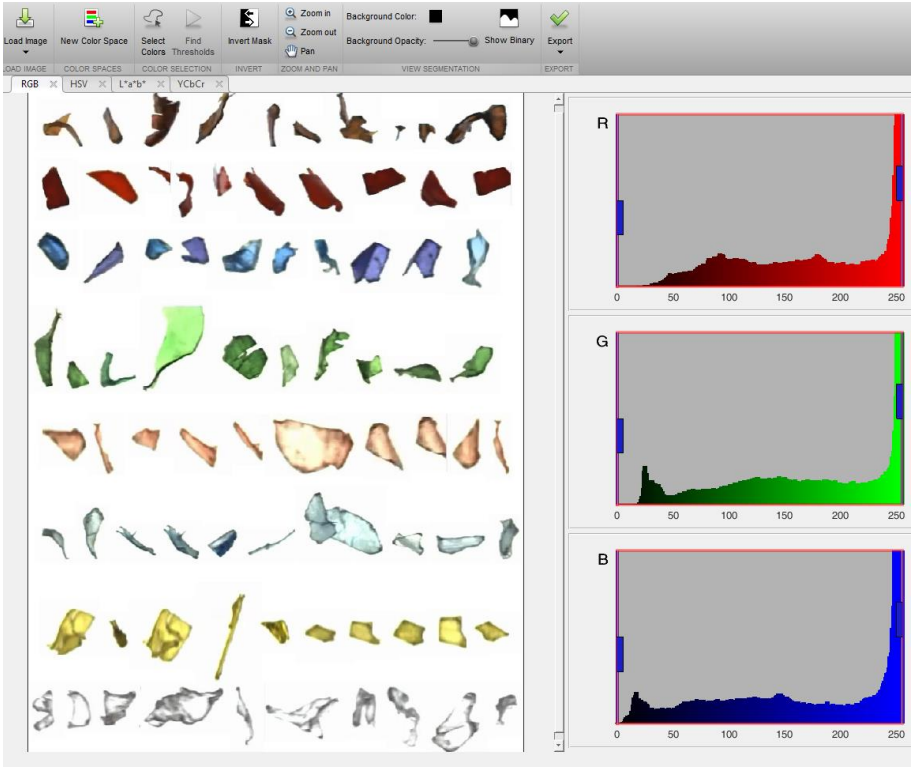
4.1 Segmentacija spalvinėse erdvėse

Siekiant atlinkti spalvų segmentaciją pirmiausia reikia nustatyti spalvas, kurias reikia išskirti. Kadangi užduotis yra išrūšiuoti skaidrius PET dribsnius iš įvairiaspalvių dribsnių reikia parinkti taip spalvas spalvos išskyrimo metodui, kad liktų neatpažinti balti skaidrūs ir atpažintos visos kitos spalvos. Taigi šiai užduočiai įgyvendinti buvo ieškomos spalvų slenkstinės vertės su MATLAB programine įranga.

4.1.1 Segmentacija RGB spalvų erdvėje

RGB spalvinėje erdvėje slenkstinės vertės buvo ieškomos pasinaudojus MATLAB įrankiu „*Color Thresholder*“.

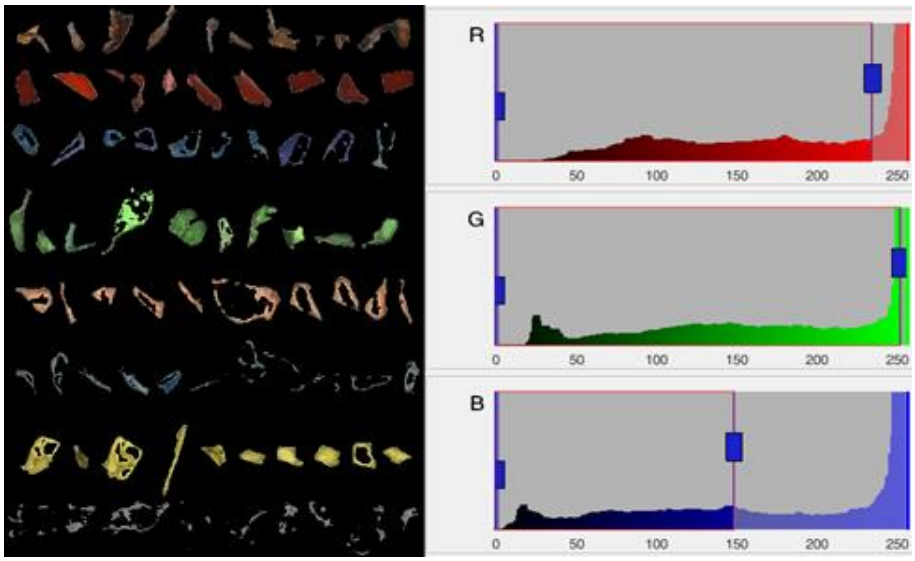
Norint parinkti slenkstinę vertę reikalinga į programą įkelti vieną neapdorotą kadrą, kuriame matytųsi visi reikalingi išskirti PET dribsniai (*žr.* 4.1 pav.).



4.1 pav. Neapdorotas vaizdas

Spalvos išskyrimas atliekamas su dešinėje pusėje esančiais slankikliais prie R, G ir B spalvų dedamųjų. Mažinant maksimalias ar didinant minimalias R, G, B parametrų vertes gaunamas atnaujintas vaizdas (*žr.* 4.2 pav.), kuriame išskirtos spalvos paliekamos originalios, o neatpažintos pažymimos juoda spalva. Tikslas yra parinkti tokias parametrų vertes, kad

maksimaliai būtų užmaskuoti skaidrūs balti PET dribsniai ir kuo mažiau įvairiaspalviai. Bandant skirtingas parametrų reikšmes ir kombinacijas gautas geriausias rezultatas pateiktas 4.2 paveiksle, kuriame matoma, kad geriausiai buvo išskirti tamsios spalvos dribsniai, o šviesesnių spalvų: žydros, rožinės nebuvo atpažinti kokybiškai: liko didelis plotas neišskirtas. Taip pat klaidingai buvo išskirti baltų, skaidrių PET dribsnių tamsesni ruožai. Gautos geriausios reikšmės, kuriose daugiausia išskiriama įvairiaspalvių dribsnių ir mažiausiai skaidrių, kai parametrų reikšmės yra rėžiuose: R(0,..,234), G(0,..,251) ir B(0,..,148).

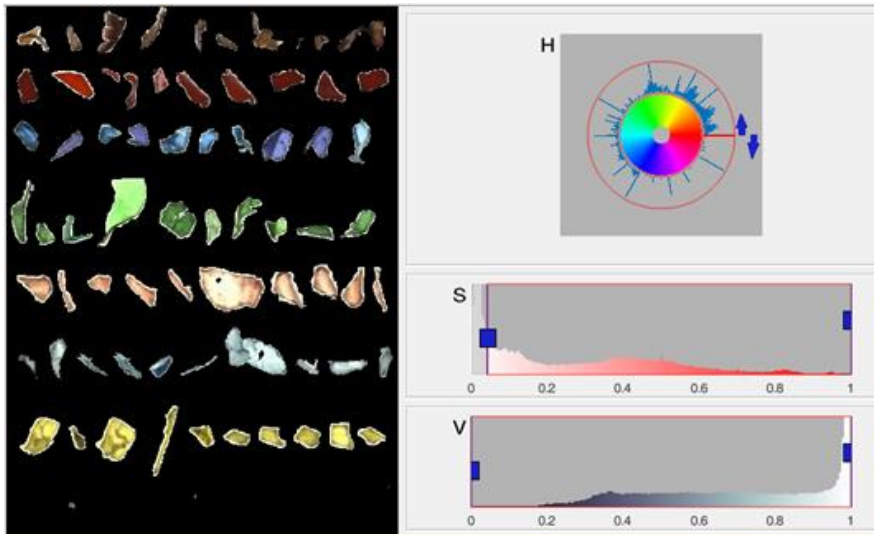


4.2 pav. Išskirti skaidrūs dribsniai RGB erdvėje

4.1.2 Segmentacija HSV spalvų erdvėje

HSV spalvinėje erdvėje slenkstinės spalvų vertės randamos analogiškai kaip ir RGB spalvinėje erdvėje naudojant MATLAB įrankį " *Color Thresholder*".

Skirtingai nuo RGB spalvinės erdvės, kurioje parenkant slenkstines vertes reikėjo koreguoti visus 3 parametrus, HSV spalvinėje erdvėje buvo reikalinga koreguoti tik spalvos įsisotinimo vertę S, paliekant H ir V vertes pilno diapazono. Taip pat HSV spalvinėje erdvėje buvo gautas daug didesnis atpažinimo tikslumas, nes lyginant 4.3 paveikslą su 4.2 matome, kad visi įvairiaspalviai dribsniai yra išskirti, o skaidrūs užmaskuoti. Ir gauname, kad išskiriamos visos spalvos, kai spalvų slenkstinės vertės yra H(0,..,1), S(0,043,..,1), V(0,..,1).

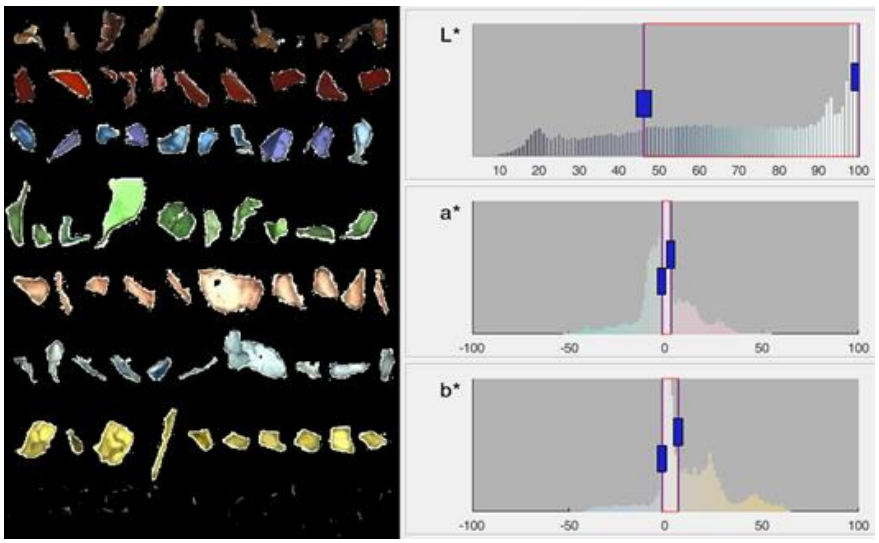


4.3 pav. Išskirti skaidrūs dribsniai HSV erdvėje

*4.1.3 Segmentacija L*a*b spalvų erdvėje*

L*a*b spalvinėje erdvėje slenkstinės spalvų vertės randamos analogiškai kaip ir RGB spalvinėje erdvėje naudojant MATLAB įrankį "*Color Thresholder*".

Atlikus slenkstinės vertės paiešką L*a*b spalvinėje erdvėje (*žr.* 4.4 pav.) surastos geriausiai tinkančios reikšmės, kurios išrinktų maksimalų skaičių įvairiaspalvių PET dribsnių ir kuo mažiau skaidrių ir gauta, kad geriausios vertės yra kai L(47,22,..,100), a(-2,084,..,2,643), b(-1,978,..,6,431). Lyginant su ankstesnėmis slenkstinės vertės paieškomis RGB ir HSV spalvinėse erdvėse, pastebėta, kad L*a*b erdvėje, buvo reikalinga keisti visų parametrų vertes ir ribos yra gan siauros ypač „a" ir „b" parametruose. Tačiau kaip matoma iš paveikslo atpažinimo tikslumas yra gan didelis.

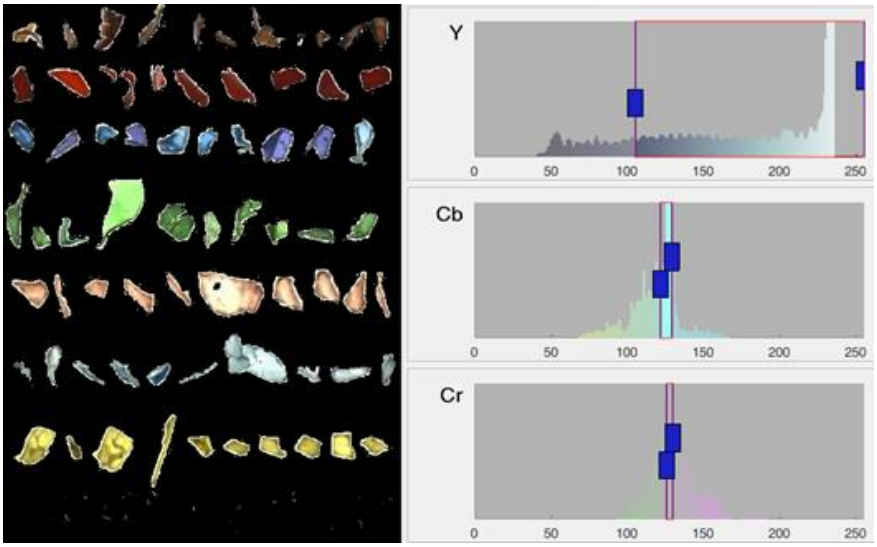


4.4 pav. Išskirti skaidrūs dribsniai L*a*b erdvėje

4.1.4 Segmentacija YCbCr spalvų erdvėje

YCbCr spalvinėje erdvėje slenkstinės spalvų vertės randamos analogiškai kaip ir RGB spalvinėje erdvėje naudojant MATLAB įrankį " *Color Thresholder*".

Atlikus slenkstinių verčių paiešką surasta, kad geriausiai skaidrūs PET dribsniai išrenkami, kai reikšmės yra Y(105,..,255), Cb(122,..,129), Cr(126,..,130). Šioje spalvinėje erdvėje slenkstinių verčių parinkimas yra artimas L*a*b spalvinei erdvei, nes parametrų Cb ir Cr vertės yra siaurame ruože. Atpažinimo kokybė aukšta: lengvai išskiriami neskaidrūs dribsniai.



4.5 pav. Išskirti skaidrūs PET dribsniai YCbCr erdvėje

4.1.5 Segmentacija pagal šviesos intensyvumo lygį

Siekiant išrinkti spalvą pagal šviesos intensyvumo lygį, vaizdas turi būti perverstas į monochromatinį. Matlab aplinkoje vaizdo transformavimas vykdomas funkcija *rgb2gray,* kuri vaizdą RGB formatu perverčia į matricą, kurioje kiekvienai vaizdo taško vertei priskiriama reikšmė nuo 0 iki 255, kur 0 yra juoda spalva, o 255 – balta. Taip gaunama pilkų atspalvių matrica, kuriai pritaikius slenkstines vertes galima išskirti norimą šviesumo lygį.

Atlikus eksperimentą ir parinkus slenkstinę vertę 125 buvo gauta, kad naudojant pilkų atspalvių matricą, pavyksta gerai išskirti tamsesnių spalvų PET dribsnius, tačiau šviesesnių spalvų išskiriami klaidingai (*žr.* 4.6 pav.*)*. Iš gautų rezultatų matoma, kad skirtingų spalvų PET dribsniams, kurie turi identišką šviesumo lygį, priskiriama ta pati pilkų atspalvių vertė, dėl to

neįmanoma išskirti šviesių dribsnių pagal spalvą. Dėl šios priežasties šis metodas, nėra tinkamas šio darbo tikslui pasiekti.

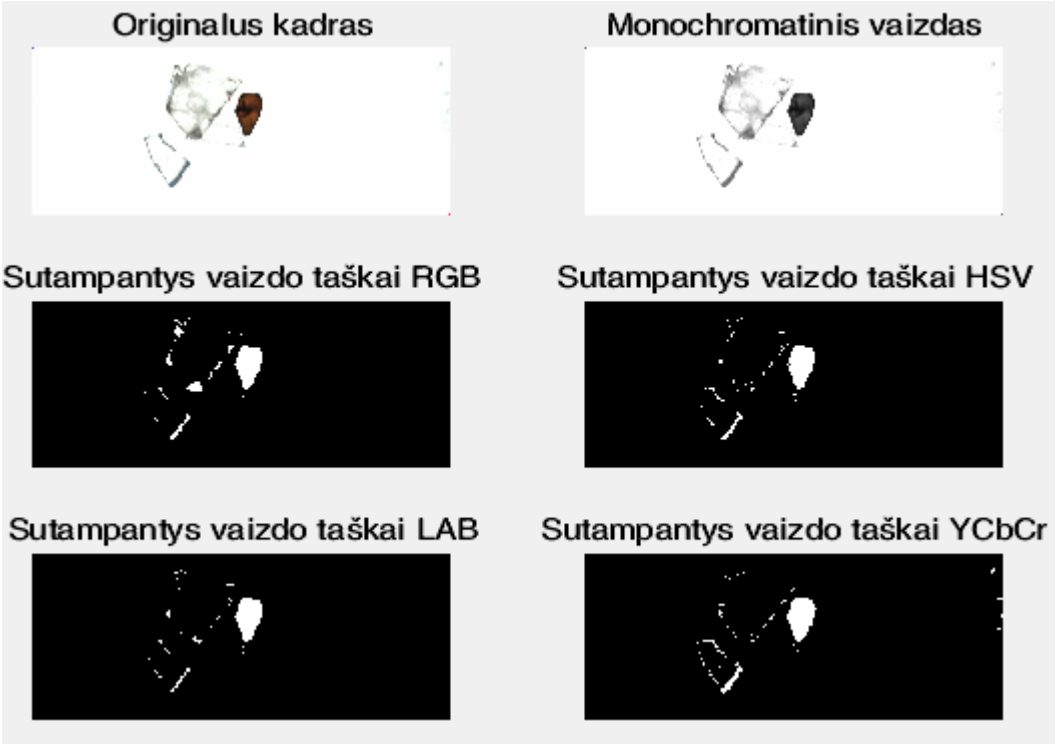
4.6 pav. Segmentacija pagal šviesos intensyvumą

4.2 Spalvos slenkstinių verčių analizė

Bandymai buvo atliekami HSV, RGB, Lab bei YCbCr spalvų erdvėje. Buvo priimta, kad ieškomi įvairiaspalviai PET plastiko dribsnių slenkstinės vertės yra srityje:

- HSV erdvėje: H (0,..,1), S(0,043,..,1), V (0,..,1).
- LAB erdvėje: L(47,22,..,100), a(-2,084,..,2.643), b(-1,978,..,6,431).
- YCbCr erdvėje: Y(105,..,255), Cb(122,..,129), Cr(126,..,130).
- RGB erdvėje: R (0,..,234), G(0,..,251) ir B(0,..,148).

Bandymas buvo atliktas su 100 vaizdo kadrų, kurie buvo parinkti iš visų bandymo duomenų taip, kad kiekviename iš jų būtų bent vienas PET plastiko dribsnis. Bandymo tikslas – išrinkti vaizdo taškus, kurių įverčiai yra tarp nustatytų slenkstinių verčių kiekvienai spalvos erdvei. Rezultatas atvaizduotas 4.7 paveiksle, kuriame matoma, jog neapdorotame vaizde yra vienas rudos spalvos dribsnis bei du skaidrūs. Transformavus RGB spalvinę erdvę į monochromatinę matoma, kad šviesūs vaizdo ruožai, paversti balta spalva, o tamsesni į juodą. Atrinkti vaizdo taškai skirtingose spalvinėse erdvėse atvaizduoti balta spalva, o užmaskuoti – juoda. Matoma, kad atrinktuose vaizdo taškuose pagal slenkstines erdves lieka triukšmų, kur tamsesnės skaidrių PET dribsnių dalys sutapo su slenkstine verte. Dėl to reikalinga naudoti filtrus, kurie pašalintų surastus mažo ploto objektus.



4.7 pav. Išskirti vaizdo taškai tenkinantys slenkstines vertes skirtingose spalvų erdvėse

Siekiant surasti slenkstinių verčių atrinkimo greitaveiką, buvo apskaičiuotas vidurkis, kiek laiko užtruko kiekvienam kadrui apdoroti. Siekiant surasti procesą, kuris ilgiausiai užtrunka analizuojant, laikas buvo išskaidytas į atskirus etapus:

- spalvų erdvės keitimo trukmė – laikas, kurį užtrunka vaizdo failą RGB formatu perversti į norimą spalvų erdvę;
- vaizdo taškų atrinkimas – laikas, nurodantis kiek užtrunka patikrinti visus vaizdo taškus HSV, L*a*b, RGB bei YCbCr erdvėse ir surastus tinkamus vaizdo taškus pažymėti juoda, o likusius balta.

4.1 lentelė. Paieškos trukmė skirtingose spalvų erdvėse

Spalvų erdvė	Vidutinė ciklo trukmė, s	Spalvų erdvės keitimo trukmė, s	Vaizdo taškų atrinkimas, s
HSV	**0,0145± 0,0032**	0,0036 ±0,0007	0,0135±0,0014
L*a*B	0,1278± 0,0275	0,1081± 0,015	0,0135± 0,0014
YCbCr	0,0309± 0,0086	0,0151± 0,0017	0,0135± 0,0011
RGB	**0,0121± 0,0013**	-	0,0121± 0,0013

Gauti rezultatai atvaizduoti 4.1 lentelėje, kurioje matoma, kad greičiausias vidutinis apdorojimo laikas, buvo naudojant RGB spalvų erdvę 0,0121± 0,0013s. Laikas buvo greičiausias

dėl to, kad nebuvo atliekamas kadro spalvų erdvės keitimas ir segmentacija buvo taikoma iš karto nuskaitytam failui. Sekantis metodas, pagal greitaveiką yra HSV spalvinėje erdvėje, kurio vidutinis laikas 0,0145± 0,0032s ir kaip matome verčiant iš RGB į HSV spalvų erdvę vidutinis laikas buvo itin spartus: 0,0036 ±0,0007s. Naudojant L*a*b ir YCbCr spalvines erdves vidutiniai laikai gauti atitinkamai 2 ir 4 kartus didesni nei naudojant RGB spalvinę erdvę. Įtaką greitaveikai lėmė spalvų erdvės keitimo trukmė, nes slenkstinių vaizdo taškų atrinkimas visais atvejais yra beveik identiškas. Taigi, pagal metodų greitaveiką skirtingose spalvų erdvėse galima atrinkti 2 greičiausius: RGB ir HSV.

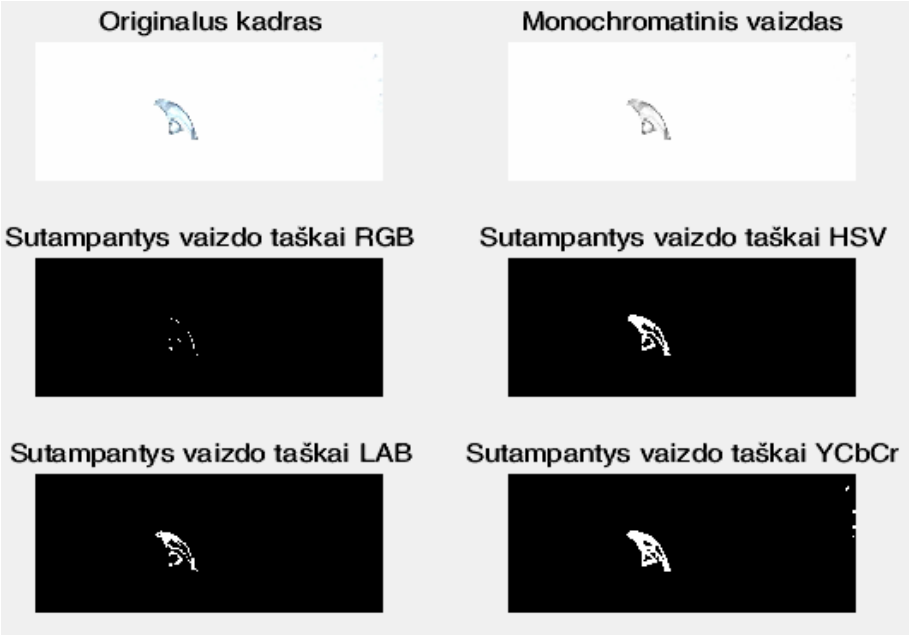
Atliekant segmentacijos tikslumo skaičiavimus skirtingose spalvų erdvėse, buvo paimta 100 vaizdo kadrų, kuriose yra bent vienas bet kokios spalvos PET dribsnis. Taip buvo gautas vaizdo įrašas kuriame iš viso buvo 157 PET dribsniai: iš kurių 99 balti skaidrūs, 35 rudi, 17 mėlynų bei 6 žali. Vadinasi blogų buvo 58 dribsniai, o gerų 99. Rezultato tikslumo įvertinimas buvo atliekamas leidžiant po vieną vaizdo kadrą ir pasižymint ar korektiškai buvo atrinkti vaizdo taškai.

Gauti tikslumo įverčiai atvaizduoti 4.1 lentelėje. Matoma, kad RGB spalvų erdvėje buvo atpažinti tik mažiau nei pusė skaidrių baltų dribsnių: iš 99 dribsnių 56 atpažinti kaip neskaidrūs. Visi rudos spalvos dribsniai buvo atpažinti kaip neskaidrūs, taip pat buvo atpažinti 4 žali dribsniai iš 6, bei 14 mėlyni dribsniai iš 17. HSV spalvų erdvėje buvo atpažinti visi žali, rudi dribsniai, tačiau neatpažintas 1 mėlynas dribsnis iš 16, bei klaidingai atpažinti 3 balti skaidrūs dribsniai. L*a*b spalvų erdvėje teisingai atpažinti visi žali ir rudi dribsniai, o neatpažinti 6 skaidrūs bei 2 mėlyni dribsniai. YCbCr erdvėje teisingai buvo atpažinti visi rudi bei žali dribsniai, tačiau neatskyrė 10 skaidrių ir 2 mėlynų PET dribsnių.

4.1 lentelė. Segmentacijos tikslumo įvertinimas

Spalvų erdvė	Atskirti skaidrūs	Neatskirti skaidrūs	Atskirti rudi	Neatskirti rudi	Atskirti mėlyni	Neatskirti mėlyni	Atskirti žali	Neatskirti žali	**Tikslumas**
RGB	43	56	35	0	14	3	4	2	**61,1%**
HSV	96	3	35	0	16	1	6	0	**97,4%**
LAB	93	6	35	0	15	2	6	0	94,9%
YCbCr	89	10	35	0	17	2	6	0	92,3%

Iš gautų duomenų pastebima, kad geriausiai atpažįstami tamsių spalvų dribsniai, kiek prasčiau šviesesnių spalvų, kurie daug nesiskiria nuo balto skaidraus. Taip pat išryškėjo maskavimo RGB erdvėje pagrindinis trūkumas: esant skaidriems, tačiau ne baltos spalvos PET dribsniams, jie yra atpažįstami kaip balti, nors jie tokie nėra. Tai galime pastebėti 4.8 paveiksle, kur žydros spalvos dribsnis yra pakankamai gerai atskiriamas HSV, L*a*b, YCbCr spalvinėse erdvėse, o RGB spalvinėje erdvėje randama mažai vaizdo taškų, kurie atitiktų slenkstines vertes. Analogiškai atskiriant skaidrius baltus dribsnius, kurie yra šiek tiek tamsesni, jie išrūšiuojami kaip blogi.



4.8 pav. Skaidrus melsvas PET dribsnis skirtingose spalvų erdvėse

Vadinasi RGB spalvų modelis nėra tinkamas ieškoti skirtingų šviesumų ir atspalvių spalvų. RGB dedamosios labai tarpusavyje koreliuoja, dėl to tai sudaro sunkumų atliekant vaizdo apdorojimo analizę, dėl to geriau naudoti HSV skalę, kurioje yra atsižvelgiama į spalvų intensyvumą [19] [29].

Greičiausias metodas nustatyti spalvas yra segmentacija RGB spalvų erdvėje, tačiau dėl itin prasto skaidrių PET dribsnių atpažinimo šis metodas nenaudotinas.

Geriausiai spalvos buvo išskirtos segmentuojant vaizdą L*a*b bei HSV spalvinėje erdvėje, tačiau HSV spalvinė erdvė daug pranašesnė už L*a*b, nes spalvų išskyrimas vykdomas iki 9 kartų greičiau. Dėl to geriausias metodas išskirti spalvas yra segmentacija HSV spalvų erdvėje.

4.3 MATLAB pozicijos paieškos metodai

Bandant skirtingus objekto paieškos metodus MATLAB aplinkoje buvo naudotas 100 atsitiktinių vaizdo kadrų iš bandymo duomenų. Kadangi ankstesni rezultatai parodė, jog geriausias būdas atrinkti slenkstines vertes yra naudoti HSV spalvinę erdvę, tai sekantys eksperimentai buvo atliekami tik joje.

Pasirinktuose paieškos metoduose objektų padėties nustatymas vykdomas tik dvimačiame paveiksle, dėl to reikalinga apdoroti vaizdą. Tam panaudotas anksčiau pateiktas slenkstinių verčių išrinkimo metodas HSV spalvinėje erdvėje.

Originaliame vaizde (*žr.* 4.9 pav.) išrenkami vaizdo taškai, kuriuose spalvas aprašančios komponentės atitiko nustatytas slenkstines vertes, taip išskiriant netinkamus PET dribsnius balta spalva, o foną juoda (*žr.* 4.10 pav.).

4.9 pav. Originalus vaizdas

4.10 pav. Atrinkti taškai

Gautame vaizde matomi vaizdo triukšmai, kuriuos reikia pašalinti. Pirmiausia pašalinami objektai vaizde, kurie yra mažo ploto. Gautas rezultatas atvaizduotas 4.11 paveiksle. Taip pat užpildomos tuščios ertmės atpažintuose objektuose. Taip gaunamas apdorotas vaizdas (*žr.* 4.12 pav.), kuriame taikant skirtingus paieškos metodus reikia surasti baltų objektų koordinatės.

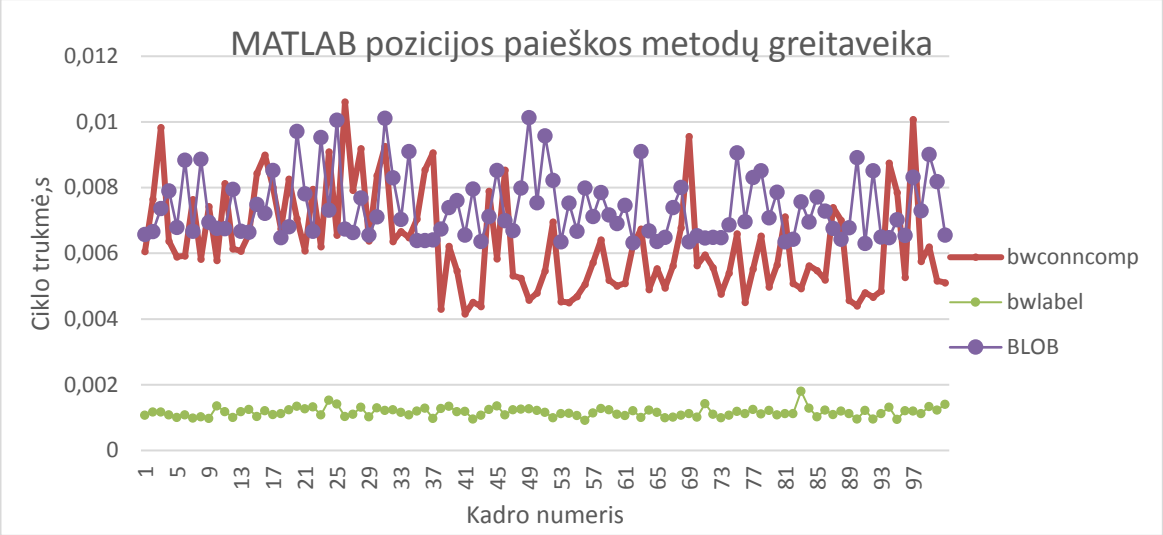
4.11 pav. Triukšmo pašaliniams

4.12 pav. Tuštumų užpildymas

Apdorotame vaizde eksperimentiškai buvo bandomi trys pozicijos paieškos algoritmai: nuoseklios paieškos, fono užpildymo bei dvimačių didelių objektų (BLOB) paieška. Visais metodais gautas beveik identiškos objektų centro koordinatės (*žr.* 4.13 pav.), vienintelis skirtumas tarp metodų – greitaveika.

4.13 pav. Surastos netinkamų PET dribsnių koordinatės

Ištestavus greitaveiką, gauti rezultatai atvaizduoti 4.14 paveikslėlyje, kuriame matoma, kad iš visų pozicijos padėties nustatymo metodų greičiausias buvo nuoseklios paieškos. MATLAB programoje šis metodas aprašomas funkcija *bwlabel*. Šios funkcijos vidutinė vykdymo trukmė 0,00114±0,00014s. Likę du metodai veikė daug lėčiau BLOB paieškos metodo vidutinė trukmė 0,007±0,001s, o fono užpildymo metodas MATLAB funkcija *bwconncomp užtruko* 0,006±0,001s.



4.14 pav. MATLAB pozicijos paieškos metodų greitaveikos rezultatai

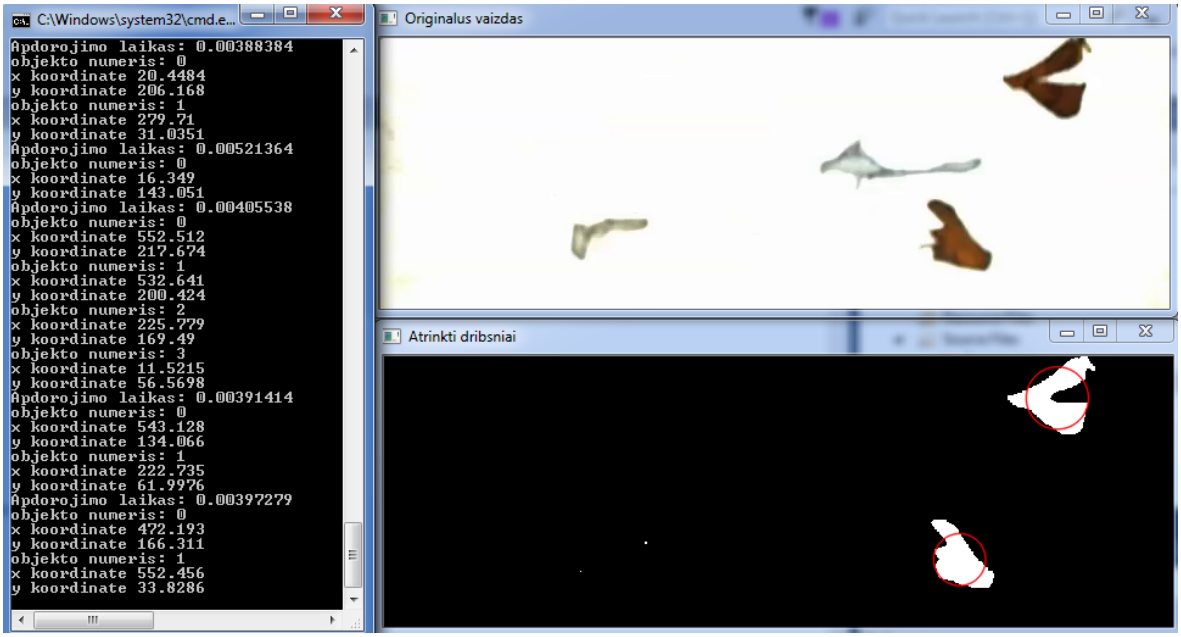
4.4 Pozicijos paieška naudojant programinį paketą Microsoft Visual Studio

Atlikus paieškos bandymus MATLAB aplinkoje surasta, jog greičiausias metodas išrinkti spalvas pagal slenkstines vertes yra segmentacija HSV spalvinėje erdvėje, kurios vidutinė trukmė 0,0145±0,0032s. Taip pat reikalinga surasti poziciją. Greičiausias yra nuoseklios paieškos metodas kurio vidutinė trukmė 0,00114±0,00014s. Kadangi šie procesai vykdomi nuosekliai bendras atpažinimo laikas yra šių dviejų suma. Taigi gauname, kad vidutinis ciklo laikas yra apie 0,0146±0,0033s arba 1/0,0146≈68 kadrai per sekundę, kai numatytas reikalingas vaizdo dažnis 128 kadrai per sekundę. Taigi, reikalinga alternatyva, leidžianti greičiau apdoroti vaizdus. Tam pasirinktas OPENCV 3.2 programinė įranga veikianti su Microsoft Visual Studio 2015 programiniu paketu.

Programos veikimo algoritmas buvo panaudotas identiškas naudotam MATLAB programoje, kuris pavaizduotas 3.1 paveiksle. Parašyto programos kodas pateiktas priede Nr. 3.

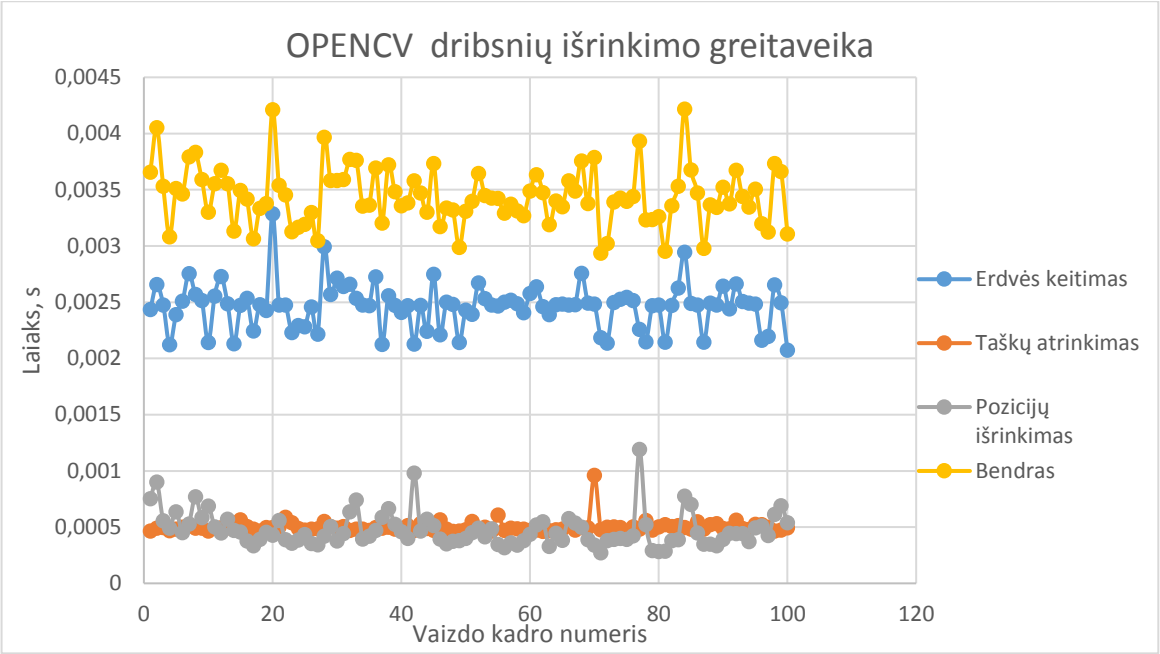
Tikrinant atpažintų objektų kokybę eksperimentiniu būdu pastebėta, kad iš 157 PET dribsnių esančių 100 vaizdo kadrų įraše visi 157 PET dribsniai, iš kurių 99 balti skaidrūs, 35 rudi, 1 mėlynas bei 6 žali buvo atpažinti teisingai. Programos rezultatų išvedimas pavaizduotas 4.15 paveiksle, kuriame kairėje pusėje esančioje konsolėje išvedamas kiekvieno vaizdo kadro

apdorojimo laikas bei kadre surastų įvairiaspalvių objektų koordinatės. Dešiniajame viršutiniame kampe atvaizduojamas originalus vaizdas, kuriam atliekamas atpažinimas. Dešiniajame apatiniame kampe atvaizduojami atrinkti vaizdo taškai, kurie pateko tarp slenkstinių verčių, kartu atvaizduojami raudoni apskritimai žymintis surasto objekto centro koordinates. Koordinatės taip pat atvaizduojamos konsolėje. Paskutinis rezultatas yra apačioje iš kurio matoma, kad surastų objektų koordinatės yra (472,193; 166,311) bei (552,456; 33,8286) ir laikas skirtas išskirti objektus yra 0,00397279s.



4.15 pav. OPENCV rūšiavimo rezultatai

Atlikus greitaveikos bandymus su 100 vaizdo kadrų gauti duomenys pavaizduoti 4.16 paveikslėlyje bei apskaičiuota, jog spalvinės erdvės keitimas iš RGB į HSV vidutiniškai trunka 0,0024±0,0002s, vaizdo taškų atrinkimas pagal slenkstines vertes – 0,00051±0,00005s, išrinktų objektų koordinačių radimas – 0,0004±0,00015s. Gautas bendras vidutinis ciklo trukmės laikas 0,0034±0,00025s laikas yra keturis kartus spartesnis lyginant su rezultatu, gautu naudojant MATLAB programinį paketą. Priimant, kad sistemos veikimo greitis priklauso nuo to, koks buvo lėčiausiai apdorotas vaizdas, surasta, kad ilgiausia vieno ciklo trukmė buvo 0,004219s, galima suskaičiuoti maksimalią sistemos veikimo greitaveiką, kuri lygi 1/0,004219=237 vaizdo kadrai per sekundę. Pasiekta greitaveika tenkina iškeltas sąlygas, jog rūšiavimo trukmė turi būti bent 128 kadrai per sekundę.



4.16 pav. PET dribsnių rūšiavimo greitaveika, naudojant OPENCV

4.5 PET dribsnių paieškos metodų reakcija į triukšmus

Darbo metu aplinkos sąlygos gali keistis. Dėl apšvietimo pasikeitimo gali būti fiksuojami šviesesni ar tamsesni objektai, dėl pašalinių dulkių gali atsirasti įvairių vaizdo triukšmų, taip pat kameros padėtis gali būti pakeista ir dėl to vaizdas bus nesufokusuotas arba perduodant vaizdą atsiras komunikacijos sutrikimų. Kuriant sistemą reikia atsižvelgi į galimus triukšmus, kurie neturėtų daryti didelės įtakos darbo kokybei.

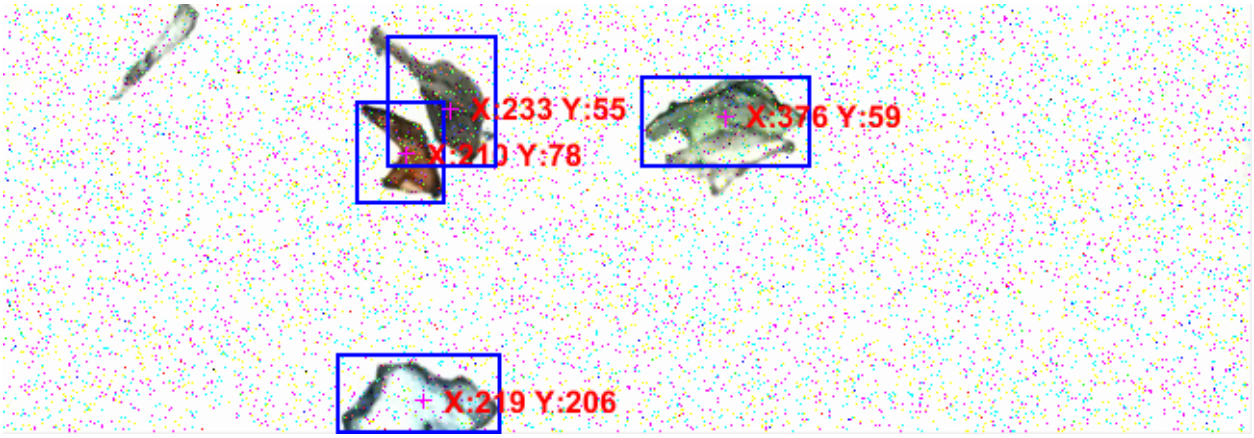
Siekiant išbandyti parinktas slenkstines vertes ir išbandyti sistemos reakciją į triukšmus buvo pritaikyti įvairūs triukšmai originaliam vaizdui:

- druskos ir pipirų (angl. *salt and pepper noise*);
- Gauso baltasis triukšmas (angl. *Gaussian white noise*);
- dėmelės (angl. *speckle*);
- išblukimo filtras;
- šviesumo keitimas.

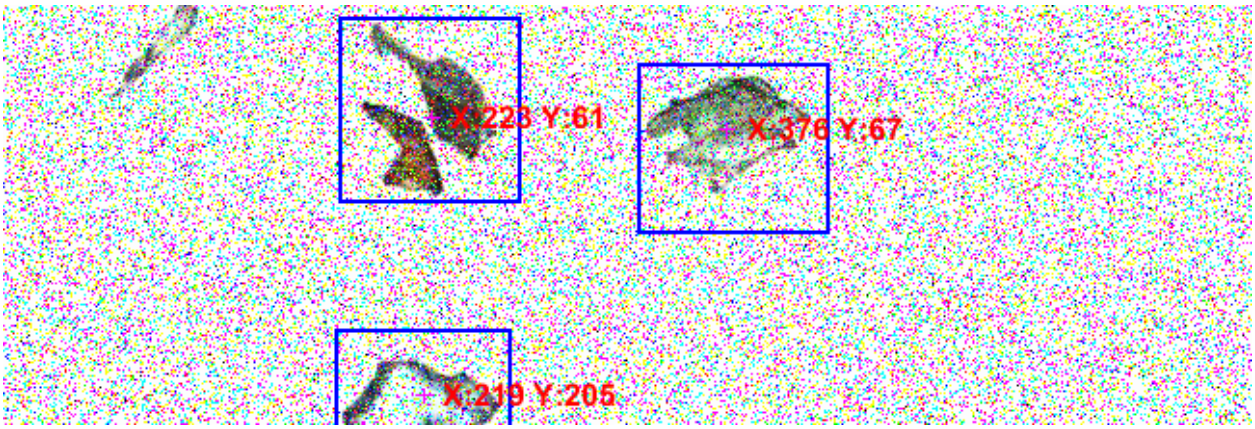
Druskos ir pipirų triukšmas MATLAB aprašomas funkcija „*J = imnoise(I,'salt & pepper',d)*“, kur *J* yra gautas vaizdas su triukšmu, *I* – apdorojamas vaizdas, *d* – nurodo, kuriai daliai visų vaizdo taškų pritaikomas triukšmas. Šio triukšmo rezultatas – atsiradę vaido taškai turintys aukštų ir žemų spalvų dedamųjų. Pirmiausia buvo paimtas vienas vaizdo kadras iš vaizdo įrašo (žr. 4.17 pav.) ir tikrinama, kokia reakcija bus į skirtingus triukšmus.

4.17 pav. Originalus vaizdas be papildomų triukšmų

Druskos ir pipirų triukšme nustačius standartinį 0,05 triukšmo lygį gautas vaizdas parodytas 4.18 paveiksle. Konstanta 0,05 parodo, kad esant vaizdo dydžiui 648x222, kurį sudaro 143856 vaizdo taškai, triukšmas bus pritaikytas 0,05*143856=7192 vaizdo taškams. Iš paveikslėlio matoma, kad surasto objektų koordinatės yra labai artimos originalaus vaizdo be triukšmo ir objektai atpažįstami teisingai. Didinant triukšmo lygį padėties tikslumo nustatymas mažėja, kartu suprastėja ir objekto atpažinimo tikslumas. Tai galima pastebėti 4.19 paveiksle, kuriame pritaikius 20% triukšmą vaizdui, tiksliai neišskiriami dribsniai: du dribsniai esantys vienas šalia kito pripažįstami kaip vienas.



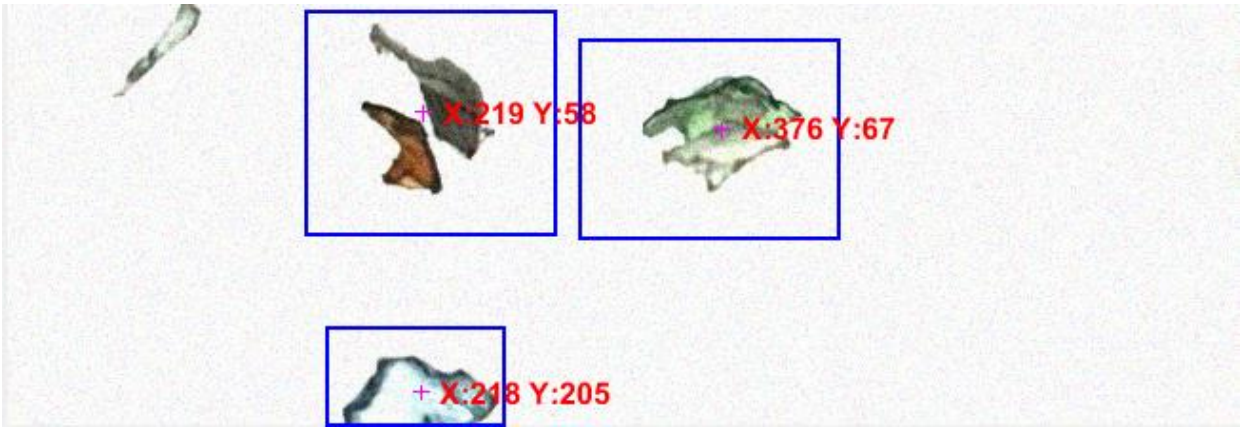
4.18 pav. Druskos ir pipirų triukšmas užimantis 5% visų vaizdo taškų



4.19 pav. Druskos ir pipirų triukšmas užimantis 20% visų vaizdo taškų

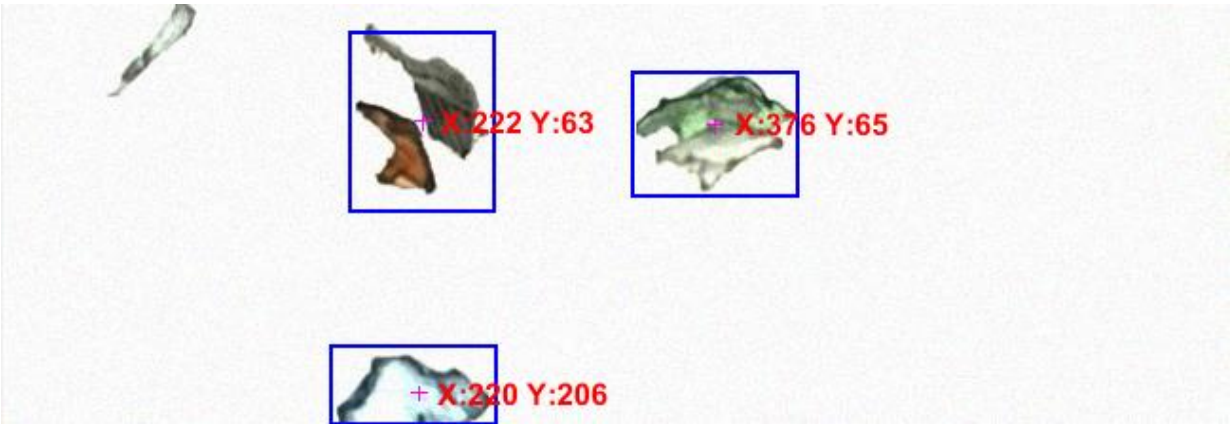
Gauso baltasis triukšmas – yra atsitiktinis signalas, kurio spektras yra pastovus visiems dažniams. MATLAB jis aprašomas funkcija „*J = imnoise(I,'gaussian',M,V)*“, kur *J* – gautas vaizdas su triukšmu, *I* – originalus vaizdas, kuriam norima pritaikyti triukšmą, *M* – triukšmo vidurkis, *V* – dispersija. Taikant triukšmą, kurio vidurkis 0,001 ir dispersija 0,001 gauname, kad dribsniai atpažįstami didesni nei yra iš tikro (žr. 4.20 pav.), nors jų centro koordinatės nedaug skiriasi nuo originalaus vaizdo. Padidinus triukšmo vidurkį ir dispersiją iki 0,003, matoma, kad objekto dydžio paklaida dar labiau padidėjo (žr. 4.21 pav.).

4.20 pav. Baltasis triukšmas, kurio vidurkis 0,001, dispersija 0,001

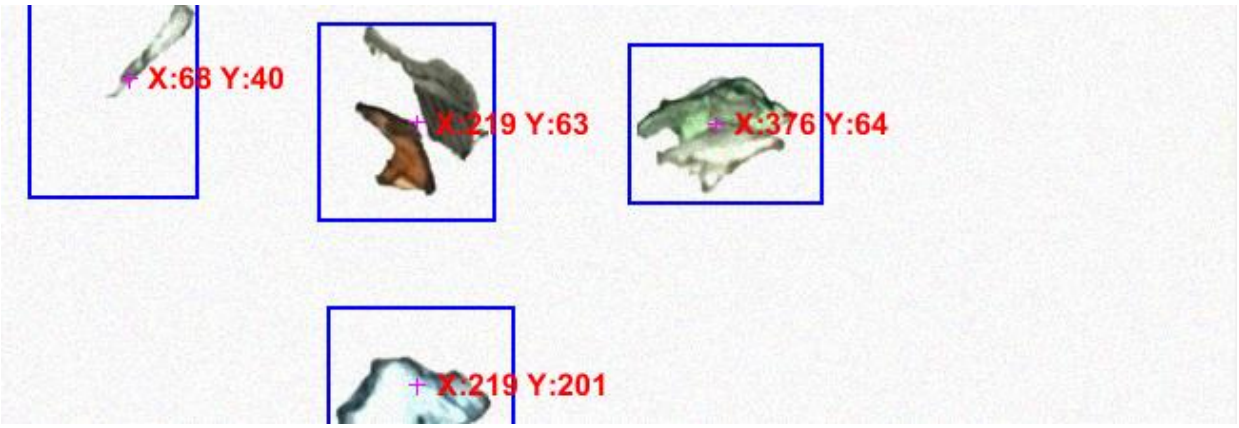


4.21 pav. Baltasis triukšmas, kurio vidurkis 0,003, dispersija 0,003

Dėmelių triukšmas sukuria triukšmą, kuris veikia pagal formulę „*a = b+n*b*“, *a* – vaizdo taškas su triukšmu, *b* – originalus vaizdo taško vertė, *n* – atsitiktinis dydis. MATLAB šis triukšmas aprašomas funkcija „*J = imnoise(I,'speckle',v)*“, kur *J* – vaizdas, *I* – originalus vaizdas, *v* – triukšmo dispersija. Parinkus triukšmo dispersija lygią 0,001, gauta, kad objektų pozicijos aptinkamos pakankamai gerai, bet dribsnių dydis randamas per didelis (*žr.* 4.22 pav.). Dispersiją padidinus iki 0,002 programa nebesugeba gerai išskirti dribsnių: skaidrus baltas dribsnis atpažįstamas kaip blogas (*žr.* 4.23 pav.).



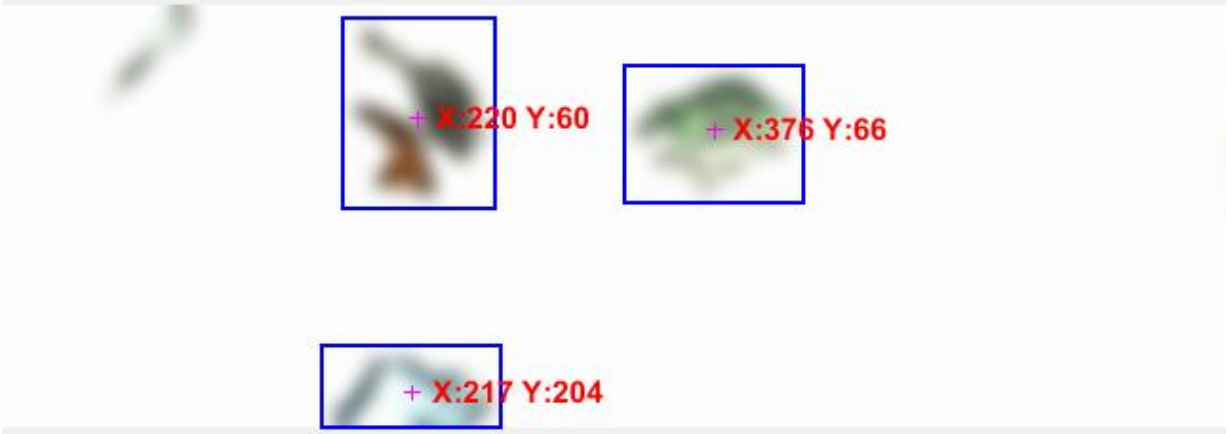
4.22 pav. Dėmelių triukšmas, kurio dispersija 0,001



4.23 pav. Dėmelių triukšmas, kurio dispersija 0,002

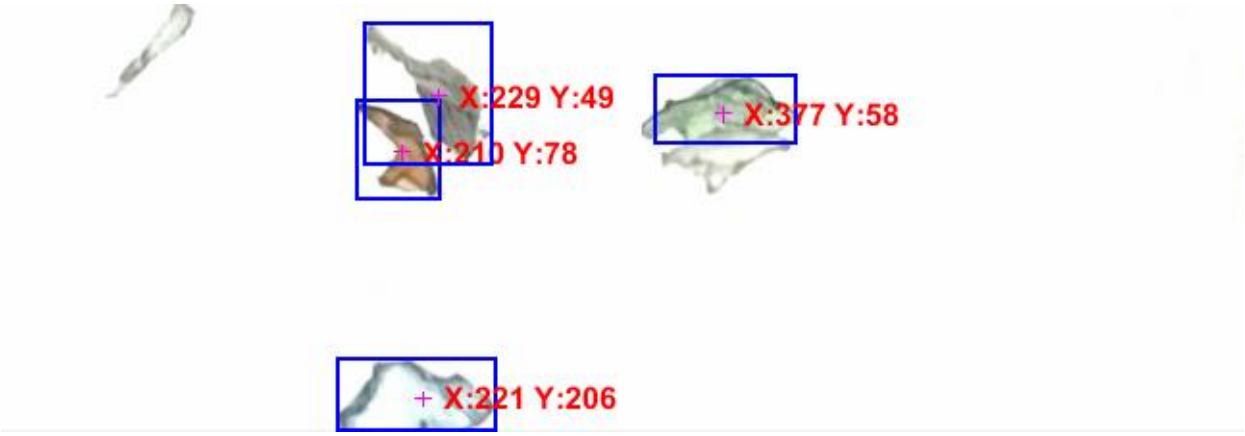
Naudojant išblukimo filtrą, gaunamas išblukęs vaizdas. MATLAB programiniame pakete filtras aprašomas funkcija „*B = imgaussfilt(A,sigma)*“, kur *B* – nufiltruotas vaizdas, *A* – originalus vaizdas, *sigma* – triukšmo standartinis nuokrypis. Didinat filtro standartinį nuokrypį, didinamas išblukimo laipsnis. Pirmiausia buvo bandomas filtras su standartiniu nuokrypiu lygiu 2 ir gauta (*žr.* 4.24 pav.), kad visi objektai aptikti, tačiau šalia esantys dribsniai nustatyti kaip vienas. Padidinus filtro standartinį nuokrypį iki 5, gautas dar labiau išblukęs vaizdas (*žr.* 4.25 pav.), kuriame vienas skaidrus dribsnis esantis šalia žalio pripažintas kaip žalio dribsnio dalis.

4.24 pav. Išblukimo filtras, kurio standartinis nuokrypis 2



4.25 pav. Išblukimo filtras, kurio standartinis nuokrypis 4

Paskutinis bandytas triukšmas – šviesumo keitimas. Šviesinant vaizdą, gauta, kad tiksliau atpažįstami tamsesni objektai, nes skaidrus balti PET dribsniai dar labiau išnyksta iš fono (žr. 4.26 pav.). Tačiau dar labiau padidinus šviesumo lygį, gaunama, kad atpažįstami tik prieš tai buvę tamsiausi dribsniai, o visi šviesesni, tampa nematomi.



4.26 pav. Pašviesintas objektas 2 kartus

REZULTATAI IR IŠVADOS

1. Darbe ištirta PET plastiko optinio rūšiavimo sistemos struktūra ir surinktas eksperimentinis įrenginys, skirtas surinkti tyrimo duomenis.
2. Išanalizavus spalviniu erdvių: RGB, HSV, L*a*b bei YCbCr požymių parametrus, nustatytos spalvų komponenčių vertės, leidžiančios aiškiai išskirti neskaidrius PET plastiko gabalėlius, matomame vaizde.
3. Atlikus eksperimentinius tyrimus, palyginus spalvine koduotes, nustatyta, kad spalvinėje erdvėje HSV požymių radimas ir segmentacija buvo sparčiausia (vidutinis laikas MATLAB programoje 0,0145± 0,0032s arba 68 kadrai per sekundę) lyginant su kitomis spalvų koduotėmis, tačiau sparta du kartus lėtesnė, negu kameros tiekiama sparta (128 kadrai per sekundę).
4. Palyginus pozicijos paieškos metodų: nuoseklios paieškos, fono užpildymo bei BLOB paieškos greitaveikas, nustatytas sparčiausias metodas objekto padėties nustatymui dvimačiame vaizde naudojant nuoseklios paieškos metodą, kuris leidžia surasti objekto koordinates per vidutinį 0,00114±0,00014s laiką.
5. Išbandžius programos veikimą, kai pradinį vaizdą veikia triukšmai, nustatyta, kad nedidelės triukšmų vertės nedaro įtakos sistemos veikimui. Esant dideliems triukšmams sistema dirba nekorektiškai.
6. Realizavus sudarytą algoritmą Microsoft Visual Studio aplinkoje su plėtiniu OPENCV, gautas rezultatas – vaizdo apdorojimo greitis 237 kadrai per sekundę, kuris viršijo kameros tiekiamą spartą 185%.

LITERATŪROS SĄRAŠAS

1. Plasticseurope Plastics – the Facts 2015. 2015. [Žiūrėta 2016 03 17]. Prieiga per internetą: http://www.plasticseurope.org/Document/plastics---the-facts-2015.aspx?Page=DOCUMENT&FolID=2.
2. Li J. Triboelectrostatic separation for granular plastic waste nrecycling: A review. 2012. [Žiūrėta 2016 03 17]. Prieiga per internetą: https://www.researchgate.net/profile/Jia_Li26/publication/233826142_Triboelectrostatic_separation_for_granular_plastic_waste_recycling_A_review/links/543775140cf2643ab9889edc.pdf/download?version=vs.
3. Ruvolo-Filho A., Curti P. S. PET recycled and processed from flakes with different amount of water uptake: characterization by DSC, TG, and FTIR-ATR. 2007. [Žiūrėta 2016 05 17]. Prieiga per internetą: http://link.springer.com/article/10.1007/s10853-007-2282-6.
4. Thomasnet How Plastic Recycling Equipment Works. [Žiūrėta 2016 03 18]. Prieiga per internetą: http://www.thomasnet.com/articles/plastics-rubber/plastic-recycling-equipment.
5. Jeavans C. Plastic recycling comes full circle," 2008. [Žiūrėta 2016 03 17]. Prieiga per internetą: http://news.bbc.co.uk/2/hi/uk_news/magazine/7470662.stm.
6. Thomasnet PET plastic reclamation. [Žiūrėta 2016 03 18]. Prieiga per internetą: http://www.thomasnet.com/articles/plastics-rubber/PET-plastic-reclamation.
7. Moroni M., Mei A., Leonardi A. ir kiti PET and PVC Separation with Hyperspectral Imagery. 2014. [Žiūrėta 18 03 2016]. Prieiga per internetą: http://web.b.ebscohost.com/ehost/pdfviewer/pdfviewer?sid=449397d0-b48c-496f-a449-331906e771df%40sessionmgr198&vid=15&hid=105.
8. Vitkauskienė I. Polietilentereftalato gamybiniu atlieku cheminis perdirbimas: aromatiniu poliesterpoliolių sintezė, savybės ir panaudojimas. 2011. [Žiūrėta 2016 05 17]. Prieiga per internetą:http://vddb.library.lt/fedora/get/LT-eLABa-0001:E.02~2011~D_20110920_152302-41359/DS.005.0.01.ETD.
9. Pakuočių tvarkymo organizacija „Pakuočių atliekų tvarkymas" [Žiūrėta 2016 04 21]. Prieiga per internetą: http://rusiuojigalvoji.lt/wpcontent/wpcontent/uploads/2014/09/Pakuociu-atlieku-tvarkymas.pdf.
10. CWC Best Practices in PET Recycling 1998. [Žiūrėta 2016 03 17]. Prieiga per internetą: http://www.cwc.org/pet_bp/2-01-03.pdf.

11. Pike P. How Optical Sorting Can Boost Your Reclaim Operation. 2015. [Žiūrėta 2016 03 18]. Prieiga per internetą: http://web.b.ebscohost.com/ehost/pdfviewer/pdfviewer?sid=9ed9dc03-b670-4d42-8e0b-d3169081749d%40sessionmgr112&vid=21&hid=128.
12. National Recovery Technologies High-Speed Identification and Sorting of Plastic Resin Flake for Recycling. [Žiūrėta 2016 04 10]. Prieiga per internetą: https://www.epa.gov/sites/production/files/2015-06/documents/highspeed.pdf.
13. Thomas D. D. Sorting It Out. 2013. [Žiūrėta 21 04 2016]. Available: http://www.resource-recycling.com/images/SortingRR0413.pdf.
14. Hurd D. J. Best Practices and Industry Standards in PET Plastic Recycling. 2001. [Žiūrėta 2016 04 21]. Prieiga per internetą: http://www.napcor.com/pdf/Master.pdf.
15. Plastic Forming Enterprises A Brief Overview of Plastic Sorting Technology. 2011. Prieiga per internetą: [Žiūrėta 2016 05 10]. http://www.plasticsrecycling.org/images/pdf/rigid_only/Charlotte-2011/10_11_Rigids_Overview_Plastic_Sort_Tech.pdf.
16. Wright S. Digital Compositing for Film and Video, Third Edition.. Focal Press. ISBN: 978-0-240-81309-7, 2010.
17. Mukhopadhyay J. Image and Video Processing in the Compressed Domain, CRC Press: London ISBN: 978-1-4398-2935-6, 2011.
18. Bruno E. A. Automated Sorting of Plastics for Recycling. 2002. [Žiūrėta 2016 10 15]. Prieiga per internetą: http://infohouse.p2ric.org/ref/09/08620.pdf.
19. Bora D. J., Gupta A. K., Khan F. A. Comparing the Performance of L*A*B* and HSV Color Spaces with Respect to Color Image Segmentation. [Žiūrėta 2016 12 15]. Prieiga per internetą: https://arxiv.org/ftp/arxiv/papers/1506/1506.01472.pdf.
20. Dervinis D. Vaizdų apdorojimas. 2012. [Žiūrėta 2016 12 15]. Prieiga per internetą: https://www.ebooks.ktu.lt/eb/451/vaizdu-apdorojimas/.
21. Luijten H. Basics of color based computer vision implemented in Matlab. [Žiūrėta 2016 10 12]. Prieiga per internetą: https://pure.tue.nl/ws/files/4373031/612499.pdf.
22. Microsoft RGB color space. [Žiūrėta 2016 10 12]. Prieiga per internetą: https://msdn.microsoft.com/ko-kr/dd372185.
23. Sural S., Qian G. ir Pramanik S. Segmentation and histogram generation using the hsv color space for image retrieval. 2002. [Žiūrėta 2016 11 02]. Prieiga per internetą: http://www.cse.msu.edu/~pramanik/research/papers/2002Papers/icip.hsv.pdf.

24. Microsoft Color. [Žiūrėta 2016 10 12]. Prieiga per internetą: https://msdn.microsoft.com/ja-jp/library/dn742482(v=vs.85).aspx.
25. Laganiere R. OpenCV 2 Computer Vision Application Programming Cookbook, Birmingham: Packt Publishing ISBN: 978-1-849513-24-1, 2011.
26. Haralick, Robert M, Linda G. Shapiro Computer and Robot Vision, Volume I, Addison-Wesley, 1992.
27. MATLAB Bwlabel [Žiūrėta 2016 12 20]. Prieiga per internetą: https://se.mathworks.com/help/images/ref/bwlabel.html.
28. Adaptive vision Blob Analysis. [Žiūrėta 10 04 2017]. Prieiga per internetą: http://docs.adaptivevision.com/4.7/studio/machine_vision_guide/BlobAnalysis.html.
29. Bora D. J., Gupta A. K., Khan F. A. Comparison of Different Color Spaces for Image Segmentation using Graph-cut. 2011 m. [Žiūrėta 2016 12 20]. Prieiga per internetą: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7294824.

PRIEDAI

Priedas Nr. 1. Eksperimentinių duomenų pradinis apdorojimas „MATLAB“ programine įranga.

```
%atidarome vaizdo įrašą "visos_spalvos.mp4"
vaizdo_irasas = vision.VideoFileReader('visos_spalvos.mp4');
%sukuriamas naujas video failas "visi", kuriame bus
%išsaugojami apdoroti kadrai
rezultatas = VideoWriter('visi','MPEG-4');
%atidaromas sukurtas failas, į kurį bus įrašinėjama
open(rezultatas)
%Sukuriamas kintamasis vaizdo kadrų skaičiavimui
z=0;

% Vykdomas ciklas, vaizdo įrašo patikrinimui, kol bus rasta 100 kadrų
% tenkinančių sąlygą
while z<100

% paimamas vienas vaizdo kadras iš vaizdo įrašo
   kadras = step(vaizdo_irasas);

% iš RGB splavinės erdvės perverčiamas į monochromatinį
   Nespalvotas = im2bw(kadras);

%invertuojamas gautas monochromatinis vaizdas
% tai leidžia pakeitus foną iš baltos spalvos į juodą, gauti
% objektus baltos spalvos juodame fone
invertuotas = imcomplement(Nespalvotas);

% sunemeruojami išrinkti objektai su fukcija bwlabel
% gaunamas rezultatas L matrica, kurioje vaizdo taškai pakeisti
% skaičiais 0 iki num, kuris atitinka surasta objekta 0- nėra objekto,
% 1- pirmas rastas objektas ir t.t. Kintamasis num parodo, kiek rasta skirtingų objektų
 [L, num] = bwlabel(invertuotas);

%  Surandamas kiekvieno rasto objekto dydis matricoje "vieno_objekto_dydis",
%  naudojant funkciją "bsxfun", kuri išrenka matricoje esančius tuos pačius skaičius
%  ir susumavus juos randame kiek vaizdo taškų sudaro objektą
  vieno_objekto_dydis = sum(bsxfun(@eq,L(:),1:num));

%   Skaiciuojamas bendras objektų dydis vaizdo kadre
dydis=sum(vieno_objekto_dydis )

% Filtruojami tušti vaizdo kadrai, bandymo metu nustatyta, kad triukšmai gali būti iki
% 50 vaizdo taškų, tai toks rėžis ir parinktas
   if dydis>50
%               jei dydis >50, yra įrašomas vaizdo kadras į naują vaizdo
%               įrašą
             writeVideo(rezultatas,kadras)
             z=z+1;
   else
%            Jei netenkina, esamas kadras neįrašomas ir einama prie kito
    z=z+1;
  end

end
% baigus išrinkimą uždaromas vaizdo įrašas, kuriame išsaugoti visi vaizdo
% kadrai
 close(rezultatas)
```

Priedas Nr. 2. Įvairiaspalvių PET dribsnių išskyrimas HSV spalvinėje erdvėje naudojant „MATLAB“ programinę įrangą.

```
clc
clear all
close all
% Nurodomos slenkstinės parametrų vertės
H_min = 0.000;
H_max = 1;
S_min = 0.000;
S_max = 0.043;
V_min = 0;
V_max = 1.000;

 %atidarome vaizdo įrašą "visos_spalvos.mp4"
    vaizdo_irasas = vision.VideoFileReader('visos_spalvos.mp4');

%    Vykdomas ciklas, tol kol yra vaizdo kadrų
while ~isDone(vaizdo_irasas)

%     nunulinama rastų PET dribsnių koordinačių matrica
 objektai=[];

 % paimamas vienas vaizdo kadras iš vaizdo įrašo
 kadras = step(vaizdo_irasas);

%    RGB spalvinė erdvė transformuojama į HSV
IHSV = rgb2hsv(kadras);

% Išrenkami vaizdo taškai, kurie tenkina slenkstines vertes.
% Sudaroma matrica BMHSV, kurioje taškai tenkinantys slenkstines vertes
% yra lygus vienetui, o kas nelygu- 0
BMHSV =(IHSV(:,:,1) >= H_min ) & (IHSV(:,:,1) <= H_max) & ...
    (IHSV(:,:,2) >= S_min ) & (IHSV(:,:,2) <= S_max) & ...
    (IHSV(:,:,3) >= V_min ) & (IHSV(:,:,3) <= V_max);
%  Vaizdo kadras perverčiamas į monochromatinį vaizdą


 %  Pašalinami vaizdo triukšmai, kurie yra didesni nei 100 vaizdo taškų
filtracimas_pos_skirtumo = bwareaopen(BMHSV,100);
% Užpildomo tuščios erdvės aptiktuose objekttuose
uzpildymas = imfill(filtracimas_pos_skirtumo,'holes');

 % sunemeruojami išrinkti objektai su fukcija bwlabel
% gaunamas rezultatas L matrica, kurioje vaizdo taškai pakeisti
% skaičiais 0 iki num, kuris atitinka surasta objekta 0- nėra objekto,
% 1- pirmas rastas objektas ir t.t. Kintamasis num parodo, kiek rasta
% skirtingų objektų

   objektas = bwlabel(uzpildymas, 8);
%    surandamos išrinktų objektų dydis ir centro koordinatės
 objekto_parametrai = regionprops(objektas, 'BoundingBox', 'Centroid');

%   Atvaizduojamas originalus vaizdas
  subplot(1,2,1);
  imshow(kadras)
  title('originalas');

%   Atvaizduojamas vaizdas su aptiktais įvairiaspalviais dribsniais
```

```
  subplot(1,2,2);
   imshow(BMHSV)
 title('Kaukė HSV');

% ant vaizdo uždedamos, dribsnių koordinatės ir jų poziciją žymintis
% kvadratas
 hold on

%  Visiem objektams, kiek surasta atliekami veiksmai
    for objekto_NR = 1:length(objekto_parametrai)

% įrašomos stačiakampio kampų koordinates
       staciakampis = objekto_parametrai(objekto_NR).BoundingBox;
% įrašomos stačiakampio centro koordinatės
       centras = objekto_parametrai(objekto_NR).Centroid;
% įrašomos stačiakampio centro koordinatės  į matricą
      objektai(objekto_NR,1)=centras(1)
      objektai(objekto_NR,2)=centras(2)

%  Nubrėžimamas stačiakampis
       rectangle('Position',staciakampis,'EdgeColor','b','LineWidth',2)

% nubrėžiamas centro taškas
       plot(centras(1),centras(2), '-m+')
% suapvalinamos centro koornatės
       a=text(centras(1),centras(2), strcat('X: ',
num2str(round(centras(1))), ' Y: ', num2str(round(centras(2)))));
% atvaizduojamas tekstas grafike
       set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,
'Color', 'red');
    end
    hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  pause(0.0001)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

% uždaromas vaizdo įrašas
release(vaizdo_irasas);
```

Priedas Nr. 3. Įvairiaspalvių PET dribsnių išskyrimas HSV spalvinėje erdvėje naudojant Visual Studio 2015 programinę įrangą su OPENCV plėtiniu

```
// Opencv biblioteka
#include <opencv2/opencv.hpp>
//Antrassciu biblioteka
#include "stdafx.h"

using namespace std;
using namespace cv;

// Globalūs kintamieji
int H_min = 0;
int H_max = 255;
int S_min = 60;
int S_max = 255;
int V_min = 0;
int V_max = 255;

// Sukuriamos vaizdų matricos
Mat kadras, atrinkti_taškai, HSV, rezultatas;



int main()
{

            //Blob detektoriaus parametrų aprašymas
            SimpleBlobDetector::Params nustatymai;
            // nustatomos slenkstinės vertės BLOB atpažinimui
            // surasto eksperimentiniu būdu
            nustatymai.minThreshold = 20;
            nustatymai.maxThreshold = 40;
            // filtravimas pagal plotą įjungtas
            nustatymai.filterByArea = true;
            nustatymai.minArea = 50;
            nustatymai.maxArea = 150000;
            //filtravimas pagal formos apvalumą- išjungta
            nustatymai.filterByCircularity = false;
            //filtravimas pagal formos iškilumus- išjungta
            nustatymai.filterByConvexity = false;
            //filtravimas pagal formos tankumą- išjungta
            nustatymai.filterByInertia = false;
            //minimalus atstumas tarp dviejų BLOB 0
            nustatymai.minDistBetweenBlobs = 0;
            // filtravimas pagal spalvą-išjungta
            nustatymai.filterByColor = false;
            // Blob parametrai išsaugojami vektoriuje
            vector<KeyPoint> parametrai;

            // atidaromas video iš projekto katalogo visi.mp4
            VideoCapture video("visi.mp4");
            // jei nepavyksta atidaryti failo, išmetamas pranešimas
            if (!video.isOpened()) {
                        cout << "Failas nepasiekiamas\n";
                        exit(-1);
            }

            //vykdomas ciklas, kol nenuspastas mygtukas "q" 1 sekundę
            while ((char)waitKey(1) != 'q') {

                        //nuskaitomas vienas kadras is video
```

```
                        video >> kadras;
                        //jeigu baigėsi kadrai, išeinama iš ciklo
                        if (kadras.empty())
                                    break;

                        //perverciama spalvų erdvė iš BGR į HSV ir išsaugojama
kintamajame "HSV"
                        cvtColor(kadras, HSV, COLOR_BGR2HSV);

                        //išrenkamos reišmės iš HSV kadro tenkinančius nurodytas
slenkstines vertes ir išsaugojama
                        // matricoje "atrinkti_taškai"
                        inRange(HSV, Scalar(H_min, S_min, V_min), Scalar(H_max, S_max,
V_max), atrinkti_taškai);



                        // BLOB detektoriui priskiriami nustatymai
                        Ptr<SimpleBlobDetector> detector =
SimpleBlobDetector::create(nustatymai);

                        /// BLOB detektorius ieško BLOB matricoje "atrinkti_taškai" ir
išsaugo duomenis į vektorių "parametrai"
                        detector->detect(atrinkti_taškai, parametrai);


                        //Nubraižomi raudoni apskritimai virš surastų BLOB ir
išsaugojami matricoje "rezultatas"
                        drawKeypoints(atrinkti_taškai, parametrai, rezultatas, Scalar(0,
0, 255), DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
                        //  Iš BLOB parametrų išrenkammos objektų koordinatės X, Y
                        for (int i = 0; i < parametrai.size(); i++) {
                                    float X = parametrai[i].pt.x;
                                    float Y = parametrai[i].pt.y;
                                    // Koordinatės ir objekto eilės numeris
atvaizduojamas konsolėje
                                    cout << "objekto numeris: " << i << endl;
                                    cout << "x koordinate " << X << endl;
                                    cout << "y koordinate " << Y << endl;
                        }
                        // atspausdinamas skyriklis konsolei, tarp skirtingų vaizdo
kadrų
                        cout << "XXXXXXXXXXXXXXXXXXXXX" << endl;


                        // Atvaizduojami grafikai
                        imshow("Atrinkti dribsniai", rezultatas);
                        imshow("Originalus vaizdas", kadras);
                        //Įvedamas uždelsimas, kad būtų atvaizduojami grafikai
                        waitKey(1);

            }
            return 0;
}
```