VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Judančių biologinių objektų aptikimą ir sekimą atliekančio roboto prototipo kūrimas

Creating a Prototype Robot that Detects and Tracks Moving Biological Objects

Magistro baigiamasis darbas

Atliko: Raminta Girdzijauskaitė.....
(parašas)

Darbo vadovas:
lekt. Irus Grinis .....
(parašas)

Recenzentas:
lekt. dr. Tomas Plankis .....
(parašas)

Vilnius, 2015 m.

Turinys

Įvadas

Tai yra trečioji mokslinio tiriamojo magistrinio darbo dalis. Pirmose dalyse aptarta, jog yra gausu naujų įrenginių ar technologijų skirtų atlikti įvairius darbus, priklausomai nuo juos supančios aplinkos. Šios technologijos geba prisitaikyti ir reaguoti į aplinkoje vykstančius įvykius, judesius, apeiti susidariusias kliūtis ir padėti žmogui atlikti darbus, kurių dažnai neįmanoma atlikti ar jų atlikimas reikalauja labai daug resursų.

2013 metais sukurta pirma pilnai autonominė mašina, taip pat įvairių kitų autonominių įrenginių. Ši sritis ir toliau plačiai nagrinėjama ir vystoma. Tai rodo poreikį autonominių technologijų, kurios galėtų ne tik atlikti konkrečius darbus, juos vykdyti nutolus, bet juos atlikti ir realiu laiku. Todėl vis didėja optimizuotų skaičiavimų reikšmė, greitas duomenų perdavimas belaidžiu ryšiu.

Šiame darbe aptariama galimybė sekti biologinius objektus esančius po vandeniu. Šioje srityje atlikta žymiai mažiau tyrimų ar sukurta technologijų padedančių apdoroti vaizdo medžiagą. Teigiama, jog sistemos ne tik sekančios, bet ir analizuojančios organizmus gyvenančius po vandeniu, yra labai reikalingos, tačiau sudėtinga jas realizuoti. Povandeninių stebėjimų metu kyla problema, jog reikia papildomų apsaugos priemonių stebinčiajam įrenginiui, kurios apsaugotų jį nuo vandens poveikio. Stebint objektus esančius po vandeniu žymiai sudėtingiau gauti ryškų vaizdą, dėl aplinkoje esančių šiukšlių ar fizikinių dėsnių, tokių kaip šviesos lūžiai. Taip pat tokias sistemas sudėtinga realizuoti, nes dažnai neraiškiame vaizde gyvūnija susilieja su augmenija ir kyla problemų jas atskiriant. Atsižvelgiant į šias problemas iškeltas mokslinio tiriamojo magistro darbo tikslas sukurti techninės robotizuotos sistemos prototipą, leidžiantį stebėti akvariumo gyvūnus bei atlikti tyrimą, kurioje padėtyje, vandenyje ar išorėje, esantis įrenginys geba tai padaryti geriau.

Pirmiausia reikia sukurti sistemą gebančią sekti ir išskirti žuveles. Šioje magistrinio darbo dalyje priimamas sprendimas, kaip turi būti įgyvendintas biologinių objektų aptikimas. Pasirinkti sprendimai realizuojami kompiuterinėje programoje. Taip pat pateikiamos rekomendacijos ir sprendimai, kokius darbus ir tyrimus reikia atlikti ateityje. Priimant sprendimus vadovautasi anksčiau atliktoje literatūros analizėje gauta informacija. Šią informaciją taikant realizuojamoje programoje naudojamos kompiuterinės regos (angl. *Computer vision*) sukurtos priemonės, leidžian-

čios apdoroti filmuotos medžiagos kadrų paveikslėlius. Anksčiau atliktoje literatūros analizėje buvo nagrinėjami grafų pjūvio (angl. *Graph cut*) ir aprėpties rėmelių (angl. *appearance model*), pagal objektų kraštus, metodai, skirti paveikslėlyje išskirti pasirinktą objektą. Šiame darbe pasirenkamas ir realizuojamas grafų pjūvio metodas, dėl jo savybės išsaugoti objekto tekstūrą. Literatūros analizės metu išsiaiškinta, jog grafų pjūvio metodas vis dar nagrinėjamas ir optimizuojamas. 2013 metais autoriai M. Tang, L. Gorelick, O. Veksler, Y. Boykov publikavo naują darbą [TGV13], kuriame patobulino minimalaus grafo pjūvio metodą taikant interaktyvų segmentavimą. Tai ypač aktualu, kai paveikslėlyje esantis objektas yra labai panašus į foną, kuriame jis randasi ir jų spalvų gamos stipriai persidengia. Šiuo atveju tiesinis grafų pjūvio metodas nebetinka, nes daugeliu atvejų objektas būtų palaikomas fonu. Tuo tarpu interaktyviame metode pradinis spalvų rinkinys apskaičiuojamas pagal objekto aprėpties rėmelį (angl. *bouding box*) sumažindamas paieškos lauką, atmesdamas visus kadro pikselius esančius už aprėpties rėmelio kaip priklausančius fonui [TGV13].

Biologinių objektų stebėjimas video medžiagoje yra kompiuterinės regos dalis. Pagrindinis darbas vykdomas analizuojant paveikslėlius, todėl svarbu pasirinkti tinkamas priemones. Grafų pjūvio metodas remiasi pikselių spalvų panašumo savybėmis, todėl objekto paieškai naudojamos spalvų histogramos. Pagal jas galima atlikti objekto paiešką ir taikyti iškirpimo funkcijas, apskaičiuoti objekto buvimo vietą bei sekti nueitą kelią. Darbui su spalvomis ir paveikslėliais yra sukurtos specialios bibliotekos. Viena tokių – OpenCv biblioteka, kurios 2014 metų išleistoje 2.4.9 versijoje realizuotas ir grafų pjūvio metodas [ODT14a]. Pasirinkus darbui su paveikslėliais tinkamą biblioteką ir objekto išskyrimo metodą, toliau darbe nagrinėjamas algoritmas, kuriuo remiantis ieškoma tinkamiausia histograma, jos pritaikymas žuvies paieškai. Šiuo atveju nagrinėjama efektyvaus ir tikslaus algoritmo, veikiančio realiu laiku, problema. Šiame darbe siekiama apjungti aptartus metodus, įrankius, savybes ir realizuoti žuvelės aptikimą filmuotoje medžiagoje, kuomet žmogus įsiterpia nurodant tik pradinius duomenis. Remiantis gautais duomenimis priimami sprendimai, kurie yra būtini norint toliau plėtoti ir kurti sistemą.

1. Biologinio objekto aptikimas ir išskyrimas

Objekto stebėjimas ir išskyrimas iš aplinkos yra paveikslėlių apdorojimo (angl. *Image processing*) problemos dalis. Šiame skyriuje analizuojamos priemonės ir algoritmai, kurie taikomi realizuoti kompiuterinę programą. Darbe siekiama, jog ši programa gebėtų aptikti ir išskirti žuvelę filmuotoje medžiagoje. Nors siekiama sekti biologinį objektą, esantį po vandeniu, tačiau realizacija gali būti taikoma ir sekimui atvirame ore ar ant žemės.

Skyriuje aptariama:

- Darbui su paveikslėlių apdorojimu skirta biblioteka, kuri buvo pasirinkta esamai programos dalies realizacijai;
- Histogramų reikšmė ir GrabCut metodo pritaikymas objektui išskirti;
- Realiu laiku adaptyvaus slankaus langelio metodas objektui aptikti;
- Pateikiami bandymai ir rezultatai, analizuojamos iškilusios problemos, priimami sprendimai reikalingi toliau realizuoti sistemai.

1.1. Paveikslėlių apdorojimui skirta OpenCV biblioteka

Tobulinant technologijas, atrandant naujus metodus skirtus atlikti stebėjimams, nebėra naujiena sukurti ir tam skirtą robotą. Šiuo metu viena iš esminių roboto kūrimo savybių – siekiamybė sukurti jam aplikaciją išleidžiant kaip galima mažiau pinigų. Todėl šiame darbe pasirinkta naudoti atviro kodo nemokamas programas, technologijas ar bibliotekas.

Analizuojant vaizdo medžiagą dirbama su kompiuterinė regos metodais. Pagrindinė biblioteka darbui su paveikslėliais buvo pasirinkta OpenCV (angl. *Open Source Computer Vision Library*). Ši biblioteka skirta pateikti bendrą infrastruktūrą kompiuterinės regos programoms. Pagrindinės jos pasirinkimo priežastys [Its15]:

- Tai yra atviro kodo nemokama biblioteka skirta naudoti tiek moksliniais tiek komerciniais tikslais. Ją taikant sutaupomi tiek piniginiai, tiek laiko resursai, nes nereikia patiems realizuoti daugelio naudojamų funkcijų, kurias parašyti gali užtrukti kelis mėnesius. OpenCV turi daugiau nei 2500 optimizuotų algoritmų, kurie apima tiek klasikinius, tiek modernius (angl. *state-of-the-art*)

kompiuterinės regos (angl. *computer vision*) ir sistemų mokymosi (angl. *machine learning*) algoritmus. Jie pritaikyti sekimui, veidų atpažinimo funkcijoms atlikti, atpažinti daiktams ar kitiems objektams, bei sekti kameros judesį, sekti judančius objektus, leidžia klasifikuoti žmonių judesius analizuojant vaizdo medžiagą. Taip pat ji leidžia išskirti objektų 3D modelius ir formuoti trimates formas naudojantis duomenis gautus filmuojant stereo kameromis. OpenCV suteikia galimybę rasti panašius kadrus paveikslėlių duomenų bazėje. Ši funkcija ypatingai svarbi realizuojant 3D objekto struktūros atkūrimą;

- Ji turi C, C++, Python, Java ir MATLAB interfeisus bei yra palaikoma visų pagrindinių operacinių sistemų: Windows, Linux, Mac OS, iOS ir Android. Ši savybė yra būtina siekiant įgyvendinti aplikaciją, kurią galima būtų perkelti į telefono ar roboto aplinkas;

- Pagrindinis OpenCV privalumas, jog ji buvo specialiai sukurta skaičiavimo efektyvumui padidinti ir ypač dirbant su realaus laiko programomis (angl. *real-time applications*);

- Dar vienas šios bibliotekos privalumas tai, jog ji pateikia ne tik realizuotas funkcijas, bet ir algoritmų ir tų funkcijų junginių pavyzdžius įvairiomis kalbomis, kuriuos galima naudoti, kaip pradinį išeities tašką.

Nors OpenCV teikia tūkstančius įvairių algoritmų ir funkcijų, šiame darbe naudojamos tik kelios jų, kurios aptariamos kituose skyriuose:

- GrabCut metodas;

- Histogramų apskaičiavimo ir palyginimo funkcijos;

- Paveikslėlio spalvų suvienodinimo (normalizavimo) funkcija;

- Funkcijų rinkinius skirtus objekto masės centrui rasti;

- Funkcija skirta objekto kraštams, kontūrui rasti;

- Pelės įvykių apdorojimų metodai (angl. *mouse events*), figūrų braižymo metodai, tokie kaip tiesės, kvadrato ar apskritimo piešimas,paveikslėlių rodymo ir saugojimo funkcijos.

1.2. Objekto išskyrimas GrabCut metodu

Vienas iš šio magistro tiriamojo darbo uždavinių išskirti sekamą objektą. Ankstesniame „Literatūros analizė“ darbe buvo nagrinėjami du metodai žuvelei išskirti:

- Aprėpties rėmelio metodas, kuris aptinka sekamą objektą pagal paveikslėlyje randamus kraštus;
- Grafų pjūvio metodas, išpjaunantis objektą pagal nurodytas spalvų aibes priklausančias fonui ar priekiniam planui.

Abu šie metodai yra plačiai analizuojami, taikomi, bandoma juos optimizuoti. Jie bus taikomi šiame darbe realizuojamoje programoje, tačiau aprėpties rėmelio metodas yra mažiau tinkamas dėl savo savybių. Jis yra lėtesnis, nes pirmiausiai turi apskaičiuoti kadre esančius visus kraštus, juos palyginti ir atskirti, kurie priklauso norimai stebėti žuvelei. Taip pat šis metodas nėra tinkamas, nes aptikti kraštai dažnai nėra apjungti į vieną nepertraukiamą formą, todėl forma yra numanoma, o sritis pažymima rėmeliu. Remiantis šiais duomenimis negalima išskirti žuvelės formos, taip pat neišsaugoma informacija apie stebimosios tekstūrą. Tuo tarpu grafų pjūvio metodas išsaugo šią informaciją, kuri yra būtina tolesniems darbams.

Filmuojant žuvelę prastesnia kamera, paveikslėliai gali gautis gana neryškūs (žiūr. pav. 1). Šiuo atveju paieška remiantis kraštai gali būti netiksli ir sudėtinga, nes objektų kontūrai yra išplaukę ir kartais persidengę tarpusavyje, todėl būtų apskaičiuota daug netinkamų pikselių, taip vadinamų šiukšlių, pažymėtų kaip priklausančių ieškomai žuvelei. Grafų pjūvio metodas, šiuo atveju yra tinkamiausiais. Jis remiasi kaimyninių pikselių spalvų skirtumu, todėl nesaugo informacijos apie pavienius nutolusius pikselius.

Šiame darbe naudojamas patobulintas grafų pjūvio metodas, pavadintas GrabCut. Pagrindinis jo skirtumas nuo klasikinio grafų pjūvio yra tai, jog šiame metode vykdomas iteratyvus segmentavimas. Vykdant klasikinį grafų pjūvį parenkamos foninių (angl. *background*) ir priekinių spalvų (angl. *foreground*) aibės ir taikant specialius skaičiavimus iškarto iškerpamas aptiktas objektas.

GrabCut metodą sukūrė ir pristatė Carsten Rother, Vladimir Kolmogorov bei Andrew Blake darbe [RKB04]. Jis sukurtas siekiant atskirti pagrindinį vaizdą nuo fono, reikalaujant kuo mažiau žmogaus įsikišimo. Plačiau apie GrabCut metodą

1 pav.: Žuvelė filmuota 3 megapikselių kamera

galima paskaityti darbuose [RKB04] ir [TGV13]. Šiame darbe naudojama OpenCV bibliotekoje realizuota GrabCut funkcija:

$$cv2.grabCut(img, mask, rect, bgdmodel, fgdmodel, iterCount, mode), \qquad (1)$$

kur *img* – paveikslėlis, kuriame norima iškirpti formą;

mask – tai kaukės paveikslėlis (angl. *mask image*), kuriame pažymimos galimo ir tiksliai žinomo fono bei objekto vietos;

rect – stačiakampio, kuriame aptikta žuvelė, koordinatės. *Rect* nurodomas aprėpties rėmelio pradžios taškas, jo aukštis bei plotis $rect = (x, y, plotis, aukstis)$;

bdgModel, fgdModel – spalvų masyvai naudojami algoritmo viduje, todėl yra vienodi ir standartiniai: 64 elementų (1,64) nuliniai masyvai;

iterCount – iteracijų, kurias turi atlikti algoritmas, skaičius;

mode – tai yra konstanta nurodanti būseną, kuri pažymi pagal ką atliekamas pjūvis: aprėpties rėmelį ar kaukės paveikslėlį [ODT14a];

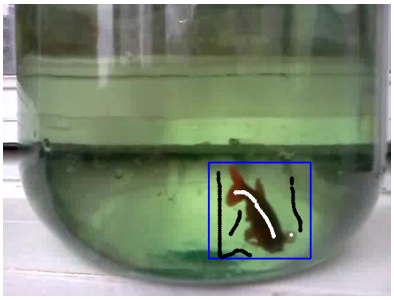
Realizuojant programą objekto iškirpimas vykdomas dviem etapais:

- Pirmiausiai naudotojas sukuria du paveikslėlius. Pirmąjį naudoja kaip duomenų šaltinį. Antrajame paveikslėlyje pažymi aprėpties rėmelį (angl. *Bounding box*) – stačiakampį (žiūr. pav. 2a, pažymėta mėlynai) taip, jog stebimo objekto kraštai neišlįstų už jo ribų. Antrasis paveikslėlis yra pirmojo kadro kaukė (angl. *mask image*). Visi pikseliai ir jų spalvų reikšmės esančios už stačiakampio ribų iškarto priskiriamos foniniam vaizdui. Pagal pateiktą rėmelį algoritmas apskaičiuoja paveikslėlio taškų spalvų reikšmių pasiskirstymą ir priskiria

stebimam objektui ir fonui jiems priklausančius spalvų rinkinius. Naudojant Gauso mišinio modelį (GMM) (angl. *Gaussian mixture model*) sudaromi priekinio ir galinio vaizdo modeliai. GMM perskirsto paveikslėlio taškus, taip jog nežinomi pikseliai (taškai, kurie randasi viduje rėmelio) priskiriami stebimam objektui arba galimo fono aibėms. Tai atliekama lyginant spalvų panašumą ir ryšį tarp kaimyninių taškų [ODT14a]. Pagal naująjį paveikslėlio taškų reikšmių pasiskirstymą sudaromas grafas, kurio viršūnės atitinka paveikslėlio pikselius. Viršūnės būna dviejų tipų: šaltinio (angl. *source node*) ir fono (angl. *sink node*). Šaltinio viršūnė jungiama su objekto pikseliais, o fono viršūnė jungiama su fono pikseliais remiantis apskaičiuota priklausymo objektui tikimybe. Sudarant grafą pikseliai jungiami briaunomis su svoriais. Svoriai paskirstomi taip, jog tos briaunos, kurios jungia taškus su didžiausiu spalvų reikšmių skirtumu, gauna mažiausią svorį. Sudarius tokį grafą, taikomas literatūros analizėje aptartas minimalaus pjūvio algoritmas, kuris grafą padaliną į du regionus, pjaudamas mažiausio svorio briaunas ir atskirdamas stebimo objekto taškus nuo foninių. Šis procesas taikomas tol kol randamas pilnas baigtinis kontūras, t.y. kol randamas nenutrūkstamas objekto kraštas [ODT14a];

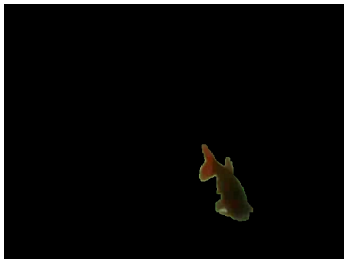
- Nors rėmelis padeda atmesti didžiąją dalį nereikalingos informacijos, tačiau paklaida gali likti (žiūr pav. 2b), nes rėmelis nurodo tik tikrus fono taškus, o objektui priskiriami tik numanomi taškai. Paklaida susidaro tuomet, kai rėmelis, šiuo atveju, apriboja ne tik žuvelę, bet į jo ribas pilnai pakliūnantį, kokį nors kitą, pašalinį daiktą, pavyzdžiui oro burbulą. Taip pat paklaida gali susidaryti jei paveikslėlio raiška yra prasta ir susidaro šešėlių sluoksniai. Atsiradusiai paklaidai ištaisyti ranka parenkamos fonui ir priekiniam planui priklausančių spalvų aibės bei pikseliai. Abi šios aibės vėl taikomos GrabCut algoritme, aprašytame pirmame etape, taip iškerpant tikslų stebimą objektą. Jei paklaidų vis tiek lieka – kartojamas antras žingsnis. GrabCut metodo taikymo pavyzdį galima pamatyti šioje nuorodoje: [Rah13].

Ši funkcija puikiai iškerpa pirmame ir antrame kadre esančius objektus (žiūr. pav 3), nes žuvelės padėtis juose nelabai skiriasi. Tačiau tolimesniuose kadruose arba tuomet, kai žuvelė juda greičiau, atsiranda paklaida, kuri visą laiką didėja. Taip atsitinka dėl nepaslankių pažymėtųjų spalvų aibių.



(a) Žuvelės kadras, kuriame pažymėtos aprėpties rėmelio, fono ir objekto sritys: atitinkamai mėlynai, juodai, baltai

(b) GrabCut metodu išpjauta žuvelė, taikant tik aprėpties rėmelį. Paklaida susidarė rėmelyje esančių šešėlių



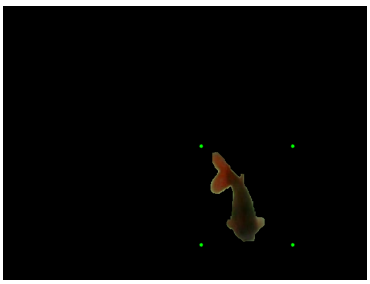
(c) GrabCut metodu išpjauta žuvelė, kuri taikoma objekto nueitam „keliui“ apskaičiuoti, bei atkuriant jo formą

2 pav.: GrabCut metodo taikymas

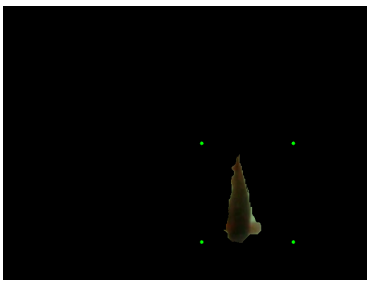
Grafų pjūvio metode spalvų aibės sudaromos pagal pažymėtus pikselius. Tai yra vienas pagrindinių GrabCut metodo trūkumų. Siekiant realizuoti autonominę sistemą ir minimizuoti žmogaus kišimąsi į jos veiklą, kiekvienam kadrui taikomos tos pačios pikselių aibės. Kadangi sekamas objektas yra nekintančios tekstūros ir dydžio, pažymėtos fonui ar žuvelei priklausančių spalvų kreivės (pav. 2a pažymėta juodai ir baltai) turėtų tikti visiems kadrams. Tačiau GrabCut metodui perduoti tik spalvų negalima. Jis pats parenka reikšme pagal nurodytas koordinates. Šiuo atveju kreivės koordinatės neatitinka objekto taškų. Todėl taikant statinę kreivę GrabCut parenka naują spalvų aibę, kuri išpjauna dalį foninių šešėlių ar daiktų, kurių spalvų reikšmės yra artimiausios naujajai kreivės reikšmei. Dar viena problema, jog naudojant statinę kreivę, žuvelę bandoma iškirpti toje vietoje kurioje ji buvo pirmame kadre. Šiuo atveju neatsižvelgiama į tai kur, taikant adaptyvaus langelio algoritmą (aprašytame tolimesniuose skyreliuose), einamajame kadre buvo aptikta žuvelė (žiūr. pav 3 objekto plotas pažymėtas keturiais žaliais taškais).

Kreivės koordinačių keitimo problema bus sprendžiama kitoje magistro tiriamojo darbo dalyje. Kadangi tos pačios pažymėtos vietos kadruose nėra tinkamos, jas perbraižyti reikia iš naujo. Tai galima padaryti naudojant Bezjė kreives (angl. *Bézier curve*) – parametrizuotas glodžias kreives (angl. *smooth curves*), kurios dažnai naudojamos kompiuterinėje grafikoje. Jos taikomos siekiant sukurti „kelius”, kurie naudojami paveikslėlių apdorojime [GHT14].

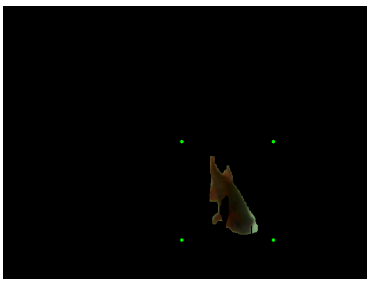
Bezjė lankai sudaromi pagal duotus taškus, parinktus taip, kad pirmasis ir paskutinis priklausytų kreivei – pradžios ir pabaigos taškai, o likę kontroliniai taškai – jai nepriklausytų. Duotieji taškai yra jungiami tiesėmis ir sudaro kontūrą, kuriame



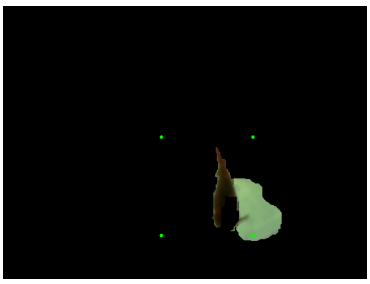
(a) 2 kadre iškirpta žuvelė



(b) 3 kadre iškirpta žuvelė



(c) 4 kadre iškirpta žuvelė

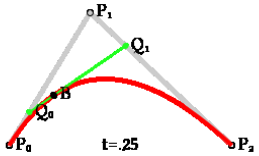


(d) 5 kadre iškirpta žuvelė

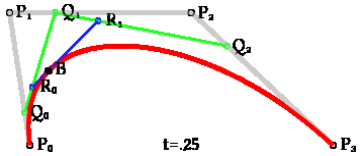
3 pav.: Automatinis GrabCut metodo taikymas kadrų sekoje. Taikant šį metodą žuvelė aptinkama tiksliai, tačiau atsiranda pjovimo paklaida, nes naudotojui nesuteikiama galimybė pakoreguoti kaukės pikselių. Žuvelių kadrus, iš kurių jos buvo išpjautos, galima pamatyti pav. 5

brėžiama kreivė taip, jog ji neišeitų už numatytų ribų. Ši savybė puikiai tinka žuvelės pažymėjimui, nes kadre radus kelis žuvelei priklausančius taškus, algoritmas užtikrintai ras kreivę neišeinančią iš stebimo objekto užimamo ploto. Bezjė kreivių trūkumas, jog jos negali suformuoti apskritimo formos lanko. Darant prielaidą, jog žuvelė stebėjimų metu nesusisuks į apskritimą, į šį trūkumą galima neatsižvelgti.

Pirmojo laipsnio kreivė yra tiesė, o antrojo ir trečiojo laipsnio kreivės – parabolės formos. Antrojo laipsnio Bezjė lanko pavyzdys parodytas pav. 4a. Ji turi tik vieną P_1 kontrolinį tašką. Kontūro taškai Q_0 ir Q_1 t laiko momentu kinta nuo 0 iki 1. Taškas Q_0 bėga tiese nuo P_0 iki P_1 ir sudaro Bezjė pirmojo laipsnio tiesę, o Q_1 – tiese nuo P_1 iki P_2. Taškas $B(t)$ kinta nuo Q_0 iki Q_1 ir nubraižo kvadratinę Bezjė kreivę [GHT14].



(a) Antrojo laipsnio Bezjė kreivė



(b) Trečiojo laipsnio Bezjė kreivė

4 pav.: Antrojo ir trečiojo laipsnio Bezjė kreivės. Šaltini paimtas iš viešai prieinamos bibliotekos.

Bezjė kreivių yra skirtingų: nuo linijinės pirmojo laipsnio, kuri neturi kontrolinių taškų, iki n-tojo laipsnio polinomų, kurie turi n kontrolinių taškų. N-tojo laipsnio kreivės formulė:

$$f(t) = \sum_{i=0}^{n} P_i B_i^n(t), \tag{2}$$

kur P_i – vektoriaus koeficientai, kontroliniai taškai, o

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \tag{3}$$

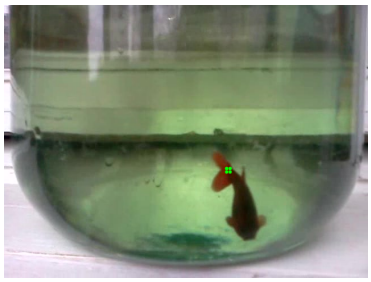
kur $\binom{n}{i}$– binominis koeficientass, o $B_i^n(t)$ – Bernsteino funkcija [GHT14].

Atsižvelgiant į skaičiavimų kiekį, dažniausiai kompiuterinėje grafikoje taikomos antros arba trečios eilės kreivės, turinčios, vieną arba du kontrolinius taškus [Tur11]. Tuo atveju jei reikia apskaičiuoti „kelius“ sudėtingesnėms figūroms, joms skaičiuojamos kelios žemesniojo laipsnio kreivės. Vėliau lankai yra sujungiami ir sudaro sudėtinę Bezjė kreivę (angl. *composite Bézier curve*), kuri vadinama Bezjė splainu. Kelias antrojo laipsnio kreives sujungus, gaunamas norimas vingiuotas kelias mažiausiomis skaičiavimo sąnaudomis.

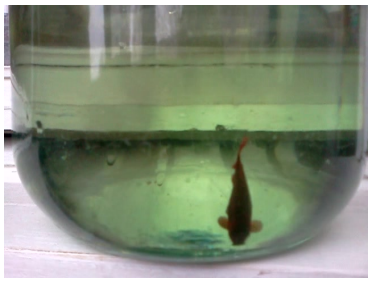
Dar vienas Bezjė kreivių trūkumas, jog pasikeitus bent vienam taškui, visas kelias turi būti perskaičiuojamas iš naujo. Ši savybė yra labai svarbi nagrinėjamu žuvelės stebėjimo atveju, nes siekiama kaip galima labiau sumažinti skaičiavimų kiekius. Tai gali būti aktualu tuo atveju jei žuvelė mažai juda ar išviso stovi. Tokiu atveju kai kurie jos taškai, pavyzdžiui akys, gali likti ilgesnį laiką vietoje. Šiuo atveju galima būtų neperskaičiuoti prieš tai buvusiame kadre pažymėtų teritorijų. Perskaičiavmų problemą spręsti gali padėti sudėtinės Bezjė kreivės. Jas naudojant, ne tik vykdomi efektyvesni skaičiavimai, bet esant reikalui galima perskaičiuoti tik tą kreivės dalį, kurioje pasikeitė kontrolinio taško pozicija.

Šioje magistrinio darbo dalyje realizuotas algoritmas randa tik vieną žuvelei priklausantį tašką (žiūr. pav. 5), tačiau dėl prastos raiškos jis randamas ne visai tiksliai. Todėl atsiranda dar viena problema, kaip efektyviau ir tiksliau nustatyti tuos taškus. Šiai problemai spręsti galima taikyti slankaus langelio metodą taškų masyvui. Taip pat galima apmokyti programą dirbti su kreivėmis ir taikyti jas pagal šabloną.

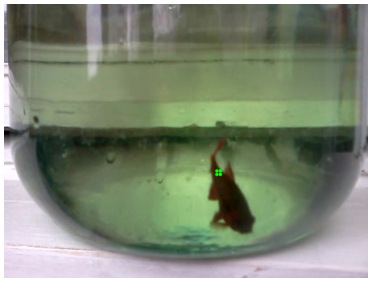
Siekiant sukurti realiu metu skaičiavimus vykdančią sistemą, kurioje naudojamos Bezjė kreivės, atsižvelgiant į skaičiavimų sudėtingumą, reikia atlikti bandymus ir nustatyti optimalų taškų skaičių skirtą suformuoti kreivei, kuri padėtų tiksliai atpažinti stebimą objektą. Šiuo atveju reikia atsižvelgti į tai, jog rasti taškai priklausys žuvelei ir visa kreivė statistiškai dažnai paklius į jos užimamą plotą.



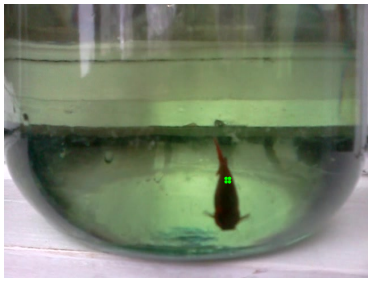
(a) Aptiktas 1 uodegos taškas



(b) Aptiktas 2 uodegos taškas



(c) Aptiktas 3 uodegos taškas



(d) Aptiktas 4 uodegos taškas

5 pav.: Žuvelės uodegos paieškos rezultatai. Ieškomas taškas, karuose a), C) ir d) pažymėtas žaliu kvadratu. b) kadre ieškoma pažymėtoji dalis nerasta

1.3. Histogramų taikymas žuvelės paieškoje

Realizuojant paiešką svarbu tiksliai atpažinti sekamą objektą. Kaip minėta ankstesniame skyrelyje, objekto formos išpjovimas vykdomas atsižvelgiant į spalvų aibes. Dirbant su spalvomis, jas taikyti galima taip pat ieškant objekto ar objekto taškų.

Šiame darbe realizuojant programą, gebančią rasti žuvelę filmuotoje medžiagoje, paieškai ir kadrų palyginimui naudojamas histogramų metodas. Histogramų metodas pasirinktas todėl, jog jam nėra svarbūs paveikslėlyje esančių figūrų formos ar kraštai, kuriuos sudėtinga tiksliai atrasti. Histogramos parodo spalvų pasiskirstymą paveikslėlyje ar pasirinktoje jo dalyje. Kaip ir grafų pjūvio atveju, galima teigti, jog objekto kaimyninių pikselių atspalviai yra labai panašūs, tačiau skiriasi

nuo foninių objektų. Todėl ieškant žuvelės galima kurti histogramas ir ieškoti ploto, kuriame yra didžiausias jai priklausančių spalvų pasiskirstymas.

Kiekvienas paveikslėlio pikselis turi spalvą, kuri susideda iš pirminių spalvų kombinacijų: raudonos, žalios ir mėlynos (angl. *red, green, blue (RGB)*). Kiekviena šių spalvų turi šviesumo (angl. *brightness*) reikšmę tarp 0 ir 255 [Sea14]. Kiekvienas kadro pikselis yra palyginamas su šiomis reikšmėmis ir sumuojama kiek taškų priklauso kiekvienam šviesumo lygiui:

$$n = \sum_{i=1}^{k} m_i, \tag{4}$$

kur n – tikrinamų pikselių skaičius, k – šviesumo reikšmių skaičius, m_i – i-tosios šviesumo reikšmės pikselių skaičius. Suvestinė histograma randama formule:

$$M_i = \sum_{j=1}^{i} m_j, \tag{5}$$

kur m_i – histograma iš prieš tai buvusios formulės. Pagal šį pasiskirstymą sudaromas spalvų dažnių grafikas.

Dirbant su paveikslėliais reikia atsižvelgiant į tai, jog dažniausiai stebimo objekto tekstūra nėra sudaryta iš tiksliai vienos spalvos. Taip pat jo spalvų pasiskirstymas kinta priklausomai nuo apšvietimo. Viena stebimo objekto pusė gali būti tamsesnė už kitą. Tai gali iškraipyti duomenis ir sudaryti nepageidaujamas histogramas. Šiai problemai spręsti galima taikyti šviesumo intervalo sąvoką. Kadangi žinoma, jog šviesumo intervalo dydis: 256, jį galima padalinti į rėžius (angl. *bins*), pavyzdžiui į 16 dalių:

$$\begin{aligned} &[0, 255] = [0, 15] \cup [16, 31] \cup .... \cup [240{,}255] \\ &intervalas = bin_1 \cup bin_2 \cup .... \cup bin_{n=15} \end{aligned} \tag{6}$$

Tokiu būdu galima sumuoti pikselių reikšmes priklausančias kiekvienam rėžiui bin_i, taip išvengiant mažų spalvos pokyčių paklaidos [ODT14b]. Kiekvienam paveikslėliui ar paveikslėlių grupei intervalo rėžių dydis gali skirtis. Mokslinė literatūra pateikė įvairių skaičiavimo metodų, skirtų sužinoti optimaliam rėžių skaičiui. Plačiau apie juos pasiskaityti galima [VR02].

Histogramos yra plačiai naudojamos paveikslėlių analizavimo ir tvarkymo srityje. Stebint histogramas gaunama informacija apie paveikslėlių kontrastų, šviesų, bei atspalvių intensyvumo pasiskirstymą. Todėl beveik visi paveikslėlių apdorojimo įrankiai leidžia dirbti su histogramomis. OpenCV biblioteka, kaip ir daugelis kitų, taip pat turi įgyvendintus histogramos kūrimo ir palyginimo funkcijas. Šiame darbe realizuotoje programoje naudojamos dvi funkcijos:

- *cv2.calcHist(images, channels, mask, histSize, ranges[, hist[, accumulate]])*;
- *cv2.compareHist(prevHist, histMask, 3).*

Pirmoji funkcija apskaičiuoja paveikslėlio ar jo dalies histogramą.

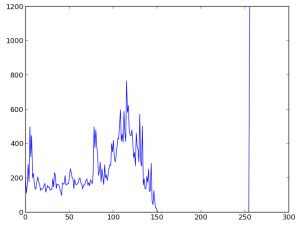
Parametras *images* – paveikslėlis, kurio duomenys naudojami histogramai formuoti.

channels – kanalų skaičius. Ši reikšmė dažniausiai būna [0], tačiau spalvotiems paveikslėliams gali būti taikomos [0], [2] arba [3], priklausomai nuo to, kokios spalvos: raudonos, žalios, mėlynos, histogramą norima matyti.

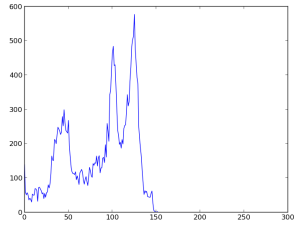
histSize – rėžių skaičius. Kaip minėta anksčiau, jis gali būti intervale nuo 0 iki 255, priklausomai nuo to kaip triukšmo paveiktas analizuojamas paveikslėlis [ODT14b]. Šiame darbe naudojama 16 rėžių histograma. Pav. 6 galima pažiūrėti kaip atrodo dviejų gretimų kadrų histogramos su 255 ir 16 rėžių parametru. Iš paveikslėlio akivaizdu, jog 16 rėžių histogramos yra žymiai panašesnės, todėl jas lengviau palyginti ir gaunami tikslesni duomenys, tiksliau randama žuvelės pozicija.

Paveikslėlio apšvietimo sudaromą triukšmą galima šalint ne tik mažinant rėžius, bet rekomenduojama normalizuoti duomenis. Normalizacijos proceso metu yra keičiamos pikselių intensyvumo reikšmės, uždedami kontrastai taip, jog paveikslėlio spalvų diapazonas būtų normalizacijos metu nustatyto dydžio. Normalizacijos metu siekiama supaprastinti ryšius tarp duomenų, taip, kad juos būtų lengviau modifikuoti. Todėl normalizuoti galima tiek paveikslėlį, tiek pačią histogramą. Šiame darbe pateiktos histogramos apskaičiuotos normalizuotiems paveikslėliams. Normalizacija vykdoma OpenCV bibliotekos funkcija: *normalize(img,img,0,255,cv2.NORM_MINMAX)*. Normalizavus paveikslėlį gaunamos žymiai tikslesnės histogramos ir palyginimo duomenys.

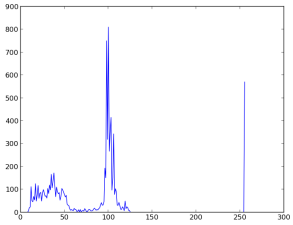
Stebimo objekto paieškoje naudojamos histogramos ir jų palyginimo funkcija. Palyginimui naudojama OpenCV realizacija: *compareHist(prevHist, histMask,*



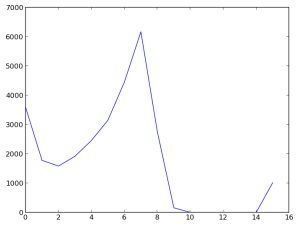
(a) Histograma teritorijos, kurioje rasta žuvelė pirmame kadre. Rėžis 255



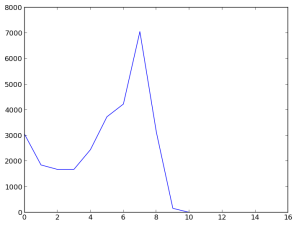
(b) Histograma teritorijos, kurioje rasta žuvelė antrame kadre. Rėžis 255



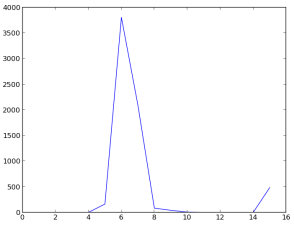
(c) Histograma tokios pat teritorijos, kurioje žuvelė nebuvo rasta. Rėžis 255



(d) Histograma teritorijos, kurioje rasta žuvelė pirmame kadre. Rėžis 16



(e) Histograma teritorijos, kurioje rasta žuvelė antrame kadre. Rėžis 16



(f) Histograma tokios pat teritorijos, kurioje žuvelė nebuvo rasta. Rėžis 16

6 pav.: Žuvelė histogramos skirtinguose kadruose

method). Ši funkcija palygina grafikus ir įvertina jų panašumą atstumo koeficientu. Kuo atstumas tarp histogramų mažesnis tuo jos yra panašesnės. *compareHist* turi trys parametrus: pirmieji du – lyginamos histogramos, trečiasis – metodas, kuriuo tapatinami duomenys. OpenCV turi realizuotus keturis palyginimo metodus [ODT14c]:

- Koreliacijos metodas (CV_COMP_CORREL) – ieškomas statistinis ryšis tarp histogramos reikšmių. Koreliacijų koeficiento metodu atstumas skaičiuojamas vadovaujantis formule:

$$d(H_1,H_2) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H_2})^2}}, \tag{7}$$

  kur $d(H_1,H_2)$ – atstumas tarp histogramų, o H_1 ir H_2 – lyginamos histogramos,

$$\bar{H_k} = \frac{1}{N} \sum_J H_k(J), \tag{8}$$

  kur N – rėžių skaičius;

- Chi-Kvadrato (angl. *Chi-Square*) metodas (CV_COMP_CHISQR). Šis metodas taiko Chi kvadrato suderinamumo kriterijų histogramoms. Chi-kvadratas naudojamas hipotezėms apie kintamojo skirstinį populiacijoje tikrinti. Kriterijus parodo, ar empirinio ir teorinio skirstynių skirtumas yra reikšmingas. Jis taikomas grupuotiems duomenims, todėl puikiai tinka histogramoms palyginti [JMR08]. Chi-kvadrato kriterijus randamas:

$$d(H_1,H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)}; \tag{9}$$

- Histogramų sankirtos metodas ($CV\_COMP\_INTERSECT$) (angl. *intersection*). Sankirta tarp dviejų histogramų skaičiuojama vadovaujantis formule:

$$d(H_1,H_2) = \sum_I \min(H_1(I), H_2(I)); \tag{10}$$

- Bhattacharyya atstumo metodas ($CV\_COMP\_BHATTACHARYY$) (angl. *Bhattacharyya distance*). OpenCV šis metodas dar vadinamas Helling atstumu ($CV\_COMP\_HELLINGER$). Šis metodas matuoja histogramų persidengimą. Bhattacharyya koeficientas apskaičiuojamas formule:

$$d(H_1,H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}}, \tag{11}$$

kur $\bar{H}_1$ apskaičiuojama pagal (7) formulę.

A. Rosebrock savo darbe [Ros14] pateikė trys metodus kaip galima palyginti histogramas. Jame atliktas tyrimas kuris iš OpenCV bibliotekos metodų geriausiai palygina histogramas ir gauna tiksliausius duomenis. Pirmiausiai buvo palygintas paveikslėlis su savim pačiu. Visi metodai atpažino, jog tai tas pats paveikslėlis. Taip pat pradinis paveikslėlis buvo palygintas su paveikslėliu, kuris buvo paveiktas Gauso (angl. *Gaussian*) triukšmo. Pastebėta, jog Chi-kvadrato metodas jautriausias triukšmui. Realizuojant paiešką prastos raiškos paveikslėliuose ši savybė yra netinkama, todėl šis metodas atmetamas. Šiame darbe buvo lyginami koreliacijos ir Bhattacharyya metodai. Geriausią rezultatą parodė Bhattacharyya metodas, todėl jis pasirinktas ir naudojamas realizuojant paiešką. 1 lentelėje parodyti histogramų palyginimo duomenys gauti ieškant objekto pirmuose 5 kadruose.

1 lentelė.: Histogramų atstumų palyginamas priklausomai nuo rėžių skaičiaus. Atstumai varijuoja nuo 0 iki 1. Jei atstumas – 0, tuomet laikoma, jog histogramos yra vienodos. Neigiamas atstumas rodo, jog žuvelė nerasta. Taikomas Bhattacharyya metodas. 16 rėžių intervale sudaromi tikslesni grafikai ir tiksliau randama žuvelė.

Kadro numeris	Koreliacijos metodo mažiausias atstumas su 255 rėžiu	Bhattacharyya metodo mažiausias atstumas su 255 rėžiu	Koreliacijos metodo mažiausias atstumas su rėžiu 16	Bhattacharyya metodo mažiausias atstumas su rėžiu 16
2	0.28961757	0.29843765	0.267178918	0.147200831
3	-0.3196158	0.29652120	-0.42624807	0.039290917
4	0.23321171	0.29033770	-0.10215228	0.078393893
5	-0.0144843	0.32606463	-0.07254427	0.059574665

1.4. Adaptyvus slankaus langelio algoritmas

Histogramos gana tiksliai palygina paveikslėlius ir pateikia duomenis tolesniems tyrimams. Tačiau lyginti kadrą su kitu kadru neturi prasmės, nebent norima patikrinti ar stebimas objektas vis dar yra kadre. Norint aptikti objekto tikslią vietą ir analizuoti jo judėjimo kryptį bei greitį, reikia taikyti histogramas tik specifiniams nustatytiems kadrų plotams.

Siekiant sukurti autonominę sistemą su minimaliu žmogaus kišimusi į jos veiklą, paieška turi vykti automatiškai. Šiuo atveju reikia perrinkti visus galimus nustatyto dydžio paveikslėlio plotus, jiems paskaičiuoti histogramas ir išrinkti grafiką, mažiausiai nutolusį nuo stebimo objekto. Tokiai paieškai galima taikyt slankaus langelio (angl. *Sliding window*) algoritmą. Šis algoritmas perbėga visus nurodyto dydžio langus paveikslėlyje. Šio metodo trūkumas, jog jis atlieka labai daug skaičiavimų ir yra lėtas, todėl jį reikia optimizuoti.

Realizuojant žuvelės paiešką pasirinkta naudoti realiu laiku prisitaikančio slankaus langelio metodą (angl. *Run-time Adaptive Sliding Window*). Šį metodą detaliai aptaria Francesco Comaschi, Sander Stuijk, Twan Basten ir Henk Corporaal darbe [CSB13]. Patį algoritmą objektų paieškai pritaikė Viola ir Jones. Šis metodas remiasi ribinėmis reikšmėmis, kurios nurodo per kelias vietas pastumti ieškomą langelį, taip mažinant skaičiavimų resursus. Jei tikimybė, jog einamajame langelyje objektas yra, yra aukštesnė nei nustatyta riba, tuomet langelis pastumiamas mažesniu žingsneliu, pavyzdžiui per 1 pikselį (žiur. pav. 8). Jei einamajame langelyje –

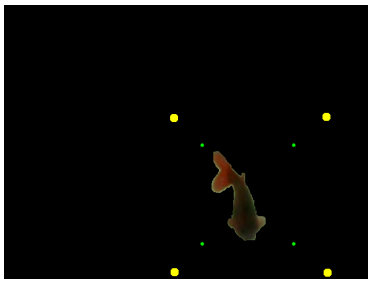
maža tikimybė, jog yra objektas, tai daroma prielaida, kad jo nebus ir gretimuose langeliuose, todėl žingsnis daromas didesnis, pavyzdžiui 5 pikselių.

Kadangi algoritme taikoma informacija gauta nuosekliai filmuojant žuvelę, tai daroma prielaida, jog objekto pozicija mažai skirsis nuo prieš tai buvusiame kadre rastos vietos. Šia prielaida vadovaujantis teigiama, jog nereikia peržiūrėti viso paveikslėlio teritorijos. Adaptyvus slankaus langelio algoritmas taikomas tik toms vietoms, kurios yra nutolusios per nustatytą nuotolį nuo prieš tai rasto objekto vietos. Šiuo atveju objektas gali pajudėti į bet kurią pusę, todėl prie kiekvienos langelio kraštinės pridedamas n pikselių paieškos laukas.

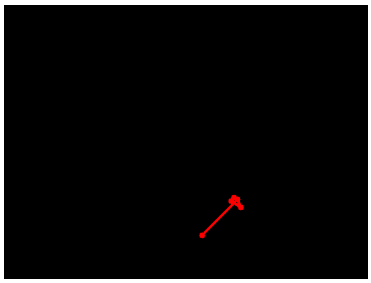
Šiame darbe realizuotoje programoje pirmasis langelis nurodomas naudotojo. Naudotojas pažymi stačiakampiu vietą, kurioje randasi žuvelė. Pagal šį keturkampį apskaičiuojama pirmoji histograma, kuri bus lyginama su histogramomis kitame kadre. Toliau paieška vykdoma ne be visame paveikslėlyje, tačiau paieškos teritorija sumažinamas, taip jog prie kiekvienos pirmajame kadre pažymėtos kraštinės pridedama po n – pikselių paieškos laukelį. Taikytame algoritme n reikšmė pažymėta, kaip $searchFieldWidth$. Tai yra jei pirmame kadre pažymėto stačiakampio pradžios taškas yra (x,y), o plotis – p ir aukštis – a, tai antrame kadre paieškos laukas sudaromas taip:

$$P_1(x,y) = (x - n, y - n); P_2(x,y) = (x + p + n, y + a + n), \tag{12}$$

kur P_1 ir P_2 - pirmas ir paskutinis paieškos lauko taškai (žiūr. pav 8). Paieškos lauko pavyzdys parodytas 7 paveikslėlyje.



(a) Aptiktos žuvelės paieškos laukas – pažymėtas geltonais taškais. Žali taškai rodo kurioje vietoje ji buvo rasta ir išpjauta.



(b) Aptiktos žuvelės nueitas kelias pirmuose 5 kadruose. Nubrėžtas kelias rodo, jog ji pajudėjo tik paskutiniame kadre.

7 pav.: Adaptyvaus slankaus langelio algoritmo paieškos rezultatai

Vykdant adaptyvaus slankaus langelio algoritmą, pirma problema, kuri iškyla – koks optimalus n dydis. Kuo didesnė n reikšmė, tuo daugiau skaičiavimų reikia atlikti ir tai kainuoja laiko. Priskyrus mažą reikšme, kyla tikimybė nerasti arba rasti tik dalį ieškomo objekto. Atlikus bandymus paaiškėjo, jog kuo didesnis pirminis laukas apibrėžiamas apie žuvelę, tuo mažesnė n reikšmė gali būti, tačiau tuomet bus sudėtingiau ją išpjauti. Ir priešingai, kuo mažesnis laukas pažymimas pirmajame kadre, tuo didesnė n reikšmė turi būti, nes algoritmas tampa jautresnis objekto postūmiui. Pavyzdžiui ieškant vieno žuvelės taško, kuris būtų taikomas Beze kreivei sudaryti, paieškos laukas turi būti gana didelis, nes jei ieškoma uodegos dalis, ji gali stipriai pasislinkti, nors pats objektas ir stovės vietoje.

```
searchFielfWidth = 50
//jei P_1 ar P_2 kuri nors reikšmė yra neigiama jai priskiriamas kraštinio pikselio
 koordinatė
P_1(x,y) = (corners[0]-searchFielfWidth,corners[1]-searchFielfWidth)
P_2(x,y) = (corners[2]+searchFielfWidth,corners[3]+searchFielfWidth)

for y in range(P_1(y), P_2(y)):
    stepX = 1
    exitStageY = 1
    stepY = stepY - 1
    for x in range(P_1(x), P_2(x)):
         stepX = stepX - 1
         if stepX == 0 and stepY == 0:
               findHistograma(n,img2)
               compDistance = compareHist()
                   if compDistance < 0.5 and compDistance < exitStage:
                          exitStage = compDistance
                          exitStageY = compDistance
                          objectPoint = (x,y) //pirmas aptikto žuvelės loto taškas
                          prev_hist2 = hist_mask
                          if compDistance >= 0.5:
                            stepX = 5  # max step 5 pixels
                          elif compDistance < 0.5:
                            stepX = 1  # min step 1 pixel
                    if exitStageY >= 0.5 and stepY == 0:
                         stepY = 5
                    elif exitStageY < 0.5 and stepY == 0:
                        stepY = 1
if exitStage != 1:
     prev_hist = prev_hist2
     cv2.grabCut()
     findCentroid()
else: "zuvyte pamesta"
```

8 pav.: Adaptyvaus slankaus langelio algoritmo, taikyto šio darbo realizacijoje, pagrindiniai žingsniai

Antroji problema kaip parinkti tinkamą žingsnį. 8 paveikslėlyje parodyta, jog žingsnis turi būti parenkamas tiek x, tiek y ašims. Atlikus bandymus su filmuota žuvele, išsiaiškinta, jog priešingai nei paieškos papildomas plotis n, žingsnio dydžiai priklauso nuo ieškomo objekto ploto. Jei ieškoma histograma žuvelės daliai, pavyzdžiui uodegos pelekui, tai ir žingsnelių dydis turi būti mažesnis, jei ieškoma visa žuvelė, tuomet ir žingsniai gali būti didesni. Atliekant bandymus su žuvele, žings-

niai, kur mažiausiai tikėtina, jog yra žuvelė galėjo būti 5 pikseliai. Ieškant žuvelės dalies, žingsnis varijavo tarp 2 ir 1 pikselio.

Taigi modifikuotas adaptyvaus slankaus langelio algoritmas žuvelę gali aptikti pakankamai greitai – tarp 0,5 ir 1 sekundės, priklausomai nuo žingsnių ir papildomo paieškos ploto dydžio.

Įvykdžius paiešką ir radus žuvelę, ji yra iškerpama (žiūr. pav. 8 30 eilutė). aptikus stebimą objektą galima apskaičiuoti ir jo masės centrą. Pagal šį rodiklį toliau skaičiuojamas ir braižomas žuvelės nueitas kelias (žiūr. pav. 7 b)). Masės centrui apskaičiuoti naudojamas OpenCV realizuotas centroido paieškos metodas. Realizuota centroido funkcija remiasi objekto kraštais, todėl pirmiausiai randamos iškirptos žuvelės kontūrai, taikant OpenCV funkciją – $findContours()$. Pagal juos skaičiuojamas objekto momentas (angl. *moments*) – paveikslėlio pikselių intensyvumo svertinis vidurkis. Kompiuterinėje regoje momentai dažniausiai naudojami po segmentavimo. Iš apskaičiuoto momento matricos galima sužinoti informaciją apie objekto užimamą plotą, perimetrą, posūkio kampą ar šiuo atveju – masės centrą. Masės centras pasiskaičiuoja formule:

$$C_x = \frac{M_{10}}{M_{00}} \tag{13}$$

$$C_y = \frac{M_{01}}{M_{00}}, \tag{14}$$

kur C_x ir C_y - centroido koordinatės.

Šiame darbe realizuotoje programoje stebimo objekto masės centras randamas pagal grafų pjūviu iškirptą žuvelės formą. pav. 7 parodyta pirmų 5 kadrų centroidai ir nueitas kelias. Nors duotuose kadruose žuvelė beveik nejudėjo, vienas masės centras apskaičiuotas gana stipriai nutolęs. Taip yra todėl, kad grafų pjūvio metu buvo gauta paklaida ir žuvelė buvo blogai iškirpta (žiūr. pav. 3, d) kadras). Masės centro paieška gali būti taikoma ir pradiniam paveikslėliui, tačiau šis metodas veikia, tik tuomet jei yra aptinkami aiškūs objekto kontūrai. Pradiniame paveikslėlyje arba būtų sudėtinga rasti objektui priklausančius kraštus arba jų išvis neįmanoma būtų rasti. Tokiu atveju cendroido apskaičiuoti neįmanoma, arba jis apskaičiuojamas labai netiksliai. Nepaisant to, naudojant grafų pjūvio metodą, stebimo objekto masės centras yra gana tiksliai apskaičiuojamas ir toliau darbe galima bus apskaičiuoti

nueito kelio ilgį, greitį bei posūkio kampą atsižvelgiant į kameros judėjimo greitį, jos kryptį bei stebimo objekto dydį.

Išvados

Šio darbo metu buvo realizuota programa kuri geba rasti ieškoma objektą arba objekto dalį. Nors realizuota programa kol kas nelabai tiksliai iškerpa rastą objektą, tačiau gana tiksliai geba paskaičiuoti nueitą kelią.

Atsižvelgiant į literatūros analizėje nagrinėtus metodus skirtus žuvelės paieškai, priimtas sprendimas realizacijoje naudoti Grafų pjūvio metodą. Šio metodo privalumas, jog jis padeda tiksliau nustatyti objekto vietą, formą ir svarbiausiai leidžia apskaičiuoti tikslią jo formą bei išsaugo tekstūrą. Deja Grafų pjūvio metode objekto forma randama atsižvelgiant į naudotojo pažymėtus fonui ir manomai objektui priklausančius taškus, jų koordinates. Pirmame kadre naudotojas šiuos pikselius nurodo ranka, tačiau toliau analizuojant kadrus, siekiama, jog naudotojas nebesikištų, todėl reikia tuos taškus nurodyti automatiškai. Tų pačių taškų kiekviename kadre naudoti negalima, nes objektas juda ir pikseliai priklausantys objektui pirmame kadre, gali nebe sutapti antrame kadre. Filmuojant gyvą žuvelę, jos forma dvimačiame paveikslėlyje gali keistis, todėl jei pažymėtos sritys būtų tiesiog pastumiamos į kitą vietą, pažymėtosios kreivės vis tiek gali nebe atitikti stebimo objekto. Šiai problemai spręsti, toliau darbe rekomenduojama taikyti Bezjė kreives. Šios kreivės pagal duotus taškus nubraižo lankus esančius tarp jų, todėl galima teigti, jos suformuos taškų aibes priklausančias norimai aplinkai, tai yra jei norima nubrėžti kreivę fonui, tai ji nepaklius į žuvelės užimamą plotą ir atvirkščiai, jei norima nubrėžti lanką žuvelėje, jis neišeis už jos ribų. Bezjė kreivės skaičiavimai priklauso nuo kontrolinių taškų skaičiaus ir kainuoja daug laiko bei skaičiavimų resursų, todėl svarbu optimaliai parinkti taškų skaičių. Darbe realizuotas algoritmas šiuo metu gali rasti tik vieną Bezjė kreivės kontrolinį tašką, todėl toliau reikia pritaikyti slankaus langelio algoritmą dviejų ar daugiau taškų paieškai.

Pats slankaus langelio algoritmas taip pat buvo optimizuojamas šiame darbe. Realizacijoje naudotas realaus laiko adaptyvus slankaus langelio metodas, tačiau taikomas ne visam paveikslėlio plotui, kaip tai siūloma daryti kituose darbuose. Šiame darbe padaryta prielaida, jog stebima žuvelė gretimuose kadruose pajudės tik nedidelį atstumą, paieška vykdoma tik nustatytame plote. Šis plotas parenkamas pagal prieš tai buvusiame kadre nurodytą žuvelės buvimo vietą. Atlikus bandymus paaiškėjo, jog paieškos ploto dydis atvirkščiai proporcingas ieškomo objekto dydžiui.

Optimizuotas adaptyvus slankaus langelio algoritmas gana tiksliai ir greitai, paieška trunka mažiau nei sekundę, randa žuvelę. Tačiau paieškos greitis priklauso ir nuo adaptyvumo žingsnių. Svarbu tinkamai parinkti slankaus langelio postūmio žingsnius. Šiuo atveju žingsnis per kiek pikselių turi būti pastumtas slankusis langelis priklauso nuo to kokio dydžio ieškomas daiktas yra. Jei tai taškas, tai ir žingsnis turi būti mažas, nes jei ieškomas žuvelės plotas, žingsnis gali būti didesnis.

Pats objekto srities palyginimas paieškoje vykdomas histogramų pagalba. Dirbant su paveikslėliais ir spalvomis grafų pjūvio metu, histogramos yra tinkamiausias metodas palyginti ieškomas teritorijas. Histogramos tiksliai parodo, kurioje paveikslėlio vietoje yra artimiausia spalvų aibė ieškomam objektui. Šie grafikai naudoja mažiau skaičiavimo resursų, nes ieškant objektų naudojant histogramas nereikia tikrinti objekto kraštų, nebūtina žinoti tikslios jo formos ar dydžio. Atlikus bandymus paaiškėjo, jog tiksliausiai žuvelė aptinkama taikant histogramas su 16 rėžių intervalu. Standartinės histogramos turi 256 šviesumo rėžius kiekvienai iš trijų spalvų: raudonai, žaliai bei mėlynai. Rezultatai parodė, kad standartinės histogramos duomenys dvigubai prastesni už 16 rėžių grafikus. Histogramų metodo trūkumas, kad norint jas lyginti reikia pakankamai daug duomenų pateikti norint gauti tikslesnius rezultatus. Ieškant pasirinktos žuvelės dalies, kartais histogramos negali aptikti jos kituose kadruose dėl nepakankamo pikselių skaičiaus. Šiuo atveju rekomenduojama pasirinktos objekto dalies ieškoti, ne pagal vieną tašką, bet bent pagal 9 pikselių langelį.

Šiame darbe taip pat realizuotas ir aptikto objekto masės centro skaičiavimas. Šis duomuo naudojamas norint nubrėžti žuvelės nuplauktą kelią. Šiuo metu masės centras randamas netiksliai. Taip yra todėl, kad jis ieškomas pagal grafų pjūvio metodu išpjautą žuvelės formą. Kai objektas išpjaunamas netiksliai, jos masės centras taip pat pasislenka. Vis dėl to patys masės centro skaičiavimai atliekami teisingai ir šį metodą galima taikant analizuojant objekto nueitą kelią. Apskaičiavus teisingą masės centro padėtį galima apskaičiuoti nueito kelio greitį, atstumą bei posūkio kampą. Todėl toliau reikia realizuoti papildomus skaičiavimus, kurie atsižvelgtų į kameros greiti ir judėjimo kryptį bei stebimo objekto dydį bei jais remiantis pateiktų duomenis apie žuvelės judesius.

Šiuo metu daugelis sistemos dalių veikia atskirai. Todėl toliau jas reikia apjungti, patobulinti pagal aprašytas savybes ir pritaikyti darbui su robotu.

Literatūros sąrašas

[CSB13] Francesco Comaschi; Sander Stuijk; Twan Basten; Henk Corporaal. *RASW: a Run-time Adaptive Sliding Window to Improve Viola-Jones Object Detection* . Electronic Systems Group, Eindhoven University of Technology, The Netherlands , 2013.

[GHT14] Teofilo Gonzalez; Jorge Diaz-Herrera; Allen Tucker. *Computing Handbook, Third Edition: Computer Science and Software Engineering*. CRC Press, 2014.

[Its15] Itseez. *OpenCV*.
[žiūrėta 2015-01-03]. Prieiga per internetą:
<http://opencv.org/>.

[JMR08] Vytautas Janilionis; Vaidas Morkevičius; Rimantas Rauleckas. *STATISTINĖ KIEKYBINIŲ DUOMENŲ ANALIZĖ SU SPSS IR STATA*. II dalis. Įvadinio kurso į statistinę analizę mokomoji medžiaga. Kauno technologijų universitetas, 2008.

[ODT14a] OpenCV dev team. *Interactive Foreground Extraction using GrabCut Algorithm*.
[žiūrėta 2015-01-03]. Prieiga per internetą:
<http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_grabcut/py

[ODT14b] OpenCV dev team. *Histograms*.
[žiūrėta 2015-01-03]. Prieiga per internetą:
<http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_calc
http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_histograms/

[ODT14c] OpenCV dev team. *Histograms comparitin*.
[žiūrėta 2015-01-03]. Prieiga per internetą:
<http://docs.opencv.org/doc/tutorials/imgproc/histograms/histogram_comp

[Rah13] Abid Rahman K. *Grabcut Sample*.
[žiūrėta 2015-01-03]. Prieiga per internetą:
<https://www.youtube.com/watch?v=kAwxLTDDAwU>.

[RKB04] Carsten Rother; Vladimir Kolmogorov; Andrew Blake. *GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts*. Microsoft Research Cambridge, UK, 2004.

[Ros14] Adrian Rosebrock. *How-To: 3 Ways to Compare Histograms using OpenCV and Python*. Pyimagesearch, 2014.
[žiūrėta 2015-01-05]. Prieiga per internetą:
<http://www.pyimagesearch.com/2014/07/14/3-ways-compare-histograms-usi

[Sea14] Sean. *CAMERA HISTOGRAMS: TONES AND CONTRAST*.
[žiūrėta 2015-01-03]. Prieiga per internetą:
<http://www.cambridgeincolour.com/tutorials/histograms1.htm>

[TGV13] M. Tang; L. Gorelick; O. Veksler; Y. Boykov. *GrabCut in One Cut. International Conference on Compiuter Vision*. University of Western Ontario. Canada, 2013.

[Tur11] David Turner; Freetype Development Team. *FreeType Glyph Conventions*. Freetype.org, 2011.
[žiūrėta 2015-01-03]. Prieiga per internetą:
<http://www.freetype.org/freetype2/docs/glyphs/glyphs-6.html>

[VR02] W. N. Venables; B. D. Ripley. *Density Estimation*. Modern Applied Statistics with S, 4 leidimas, 2002.