VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
PROGRAMŲ SISTEMŲ BAKALAURO STUDIJŲ PROGRAMA

Krepšinio taisyklių pažeidimo automatinis nustatymas

Recognizing violations of basketball rules using computer vision

Bakalauro baigiamasis darbas

Atliko:	Lukas Cedronas	(parašas)
Darbo vadovas:	partn. prof., dr. Vytautas Ašeris	(parašas)
Darbo recenzentas:	lekt. Donatas Kimutis	(parašas)

Vilnius – 2021

Santrauka

Glaustai aprašomas darbo turinys: pristatoma nagrinėta problema ir padarytos išvados. Santraukos apimtis ne didesnė nei 0,5 puslapio. Santraukų gale nurodomi darbo raktiniai žodžiai.

Raktiniai žodžiai: raktinis žodis 1, raktinis žodis 2, raktinis žodis 3, raktinis žodis 4, raktinis žodis 5

Summary

Santrauka anglų kalba. Santraukos apimtis ne didesnė nei 0,5 puslapio.

Keywords: keyword 1, keyword 2, keyword 3, keyword 4, keyword 5

TURINYS

ĮVADAS 4

1. NAUDOTI ĮRANKIAI 5

2. VAIZDO MEDŽIAGOS PARUOŠIMAS 6

3. VAIZDO ATPAŽINIMO METODAI 7
   3.1. Spalvinis atpažinimas 7
      3.1.1. Segmentavimas 8
         3.1.1.1. Dominančių objektų spalvų rėžiai 8
      3.1.2. Morfologinės transformacijos 9
      3.1.3. Kamuolio kontūrų radimas 10
   3.2. Atpažinimas remiantis skirtumais 10
      3.2.1. Fono pašalinimas 10
      3.2.2. Judesio atpažinimas 11
   3.3. Žmogaus kūno dalių atpažinimas neuroniniais tinklais 12
      3.3.1. Atpažinimo neuroniniais tinklais optimizavimas 14
   3.4. Kamuolio atpažinimas Hough transformacija 14
   3.5. Nagrinėtų atpažinimo metodų palyginimas 15

4. KREPŠINIO TAISYKLIŲ PAŽEIDIMO ATPAŽINIMAS 18
   4.1. Žingsnių taisyklė 18
   4.2. Žingsnių taisyklės pažeidimo atpažinimo algoritmas 18
      4.2.1. Būsenos saugojimas 20
   4.3. Dvigubo varymo taisyklė 21
   4.4. Dvigubo varymo taisyklės atpažinimo algoritmas 21

REZULTATAI IR IŠVADOS 23

LITERATŪRA 24

SANTRUMPOS 26

PRIEDAI 26
   1 priedas. Sukurtos programinės įrangos kodo nuoroda 27
   2 priedas. Neuroninio tinklo struktūra 28
   3 priedas. Eksperimentinio palyginimo rezultatai 29

Įvadas

Įvade nurodomas darbo tikslas ir uždaviniai, kuriais bus įgyvendinamas tikslas, aprašomas temos aktualumas, apibrėžiamas tiriamasis objektas akcentuojant neapibrėžtumą, kuris bus išspręstas darbe, aptariamos teorinės darbo prielaidos bei metodika, apibūdinami su tema susiję literatūros ar kitokie šaltiniai, temos analizės tvarka, darbo atlikimo aplinkybės, pateikiama žinių apie naudojamus instrumentus (programas ir kt., jei darbe yra eksperimentinė dalis). Darbo įvadas neturi būti dėstymo santrauka. Įvado apimtis 2 — 4 puslapiai.

Krepšinis yra itin intensyvus ir dinamiškas žaidimas, iš žaidėjų reikalaujantis greičio, spartaus sprendimų darymo ir reakcijos laiko. Modernizuojantis krepšinis vystosi - žaidėjai tampa didesni, stipresni bei greitesni. Žaidimo spartumas daro įtaką ir teisėjams, kurie nuolatos turi observuoti bei realiu laiku nuspręsti, kada buvo pažeistos tam tikros taisyklės. Nenuostabu, jog šis darbas yra ne tik varginantis, bet ir neatsparus žmogiškosioms klaidoms. 2015 m. atliktame tyrime nustatyta, jog tų metų sezono paskutinėmis įtempto žaidimo minutėmis teisėjų tikslumas skiriant baudas tesiekė 80.1% [**Deutscher**], tuo tarpu 2018 m. analizuojant pačios NBA išsaugotas statistikas apie kiekvieno žaidimo nuo 2015 m. paskutines 2 minutes nustatyta, jog teisėjai atlieka 1.49 klaidingus sprendimus kiekvieną žaidimą, o, pavyzdžiui, žingsnių taisyklę atpažįsta tik 20.6% tikslumu [Sig19]. Negana to, teisingo ir tikslaus teisėjavimui trukdyti gali ir tokie faktoriai kaip šališkumas: NCAA (*angl.* National Collegiate Athletic Association, viena populiariausių studentų krepšinio organizacijų JAV) lygoje nustatyta, jog teisėjai labiau linkę švilpti komandos, turinčioms mažiau baudų, žaidėjams, taip pat tos, kuri žaidžia svečiuose ar pirmauja *OfficiatingBias*.

Visa tai rodo, jog teisėjavimas yra sritis, kuri gali tobulėti.

Vaizdo atpažinimas kompiuteriu - tai sparčiai besivystanti sritis. Egzistuoja daug algoritmų, kurių pagalba galima atpažinti objektus. Sportas. //@todo fix Sudetingesniems objektams dažnai yra naudojami giliuoju mokymu paremti metodai, šiame darbe išnagrinėti keli iš jų. //@todo Krepšinyje atpažinimas state of the art

1. Naudoti įrankiai

Kompiuterinės regos ir taisyklių pažeidimo atpažinimo algoritmams įgyvendinti buvo pasirinkta Python programavimo kalba. Python – interpretuojama, lengvai skaitoma kalba, puikiai tinkama įvairioms problemoms spręsti [Kuh12]. Dėl kalbos paprastumo ji dažnai naudojama kompiuterinės regos ir giliojo mokymo srityse, kadangi kalba leidžia susifokusuoti į abstrakcijas. Kadangi kalba yra plačiai naudojama, nesunku rasti daug pavyzdžių bei šaltinių.

Objektų atpažinimui buvo pasirinkta OpenCV biblioteka [Ope21a]. Tai – nemokama biblioteka, plačiai naudojama spręsti kompiuterinės regos uždavinius, kurios fokusas – įgalinti kurti aplikacijas leidžiančias efektyviai spręsti kompiuterinės regos problemas realiu laiku [BK08].

Python paketų, įgyvendinančių įvairius kompiuterinės regos algoritmus ir pagalbines funkcijas, siuntimui ir jų valdymui naudota Miniconda [Ana21]. Ši biblioteka leidžia susikurti skirtingas aplinkas su tam tikrais reikalingais Python moduliais ir jas itin lengvai keisti priklausomai nuo norimo vykdyti kodo.

Vaizdo apdorojimui neuroniniais tinklais naudojama PyTorch biblioteka [Tor21], skirta lengvai valdyti neuroninių tinklų modelius bei juos integruoti į Python kodą. Taip pat PyTorch įgalina vaizdo apdorojimą naudojantis GPU.

Vaizdo medžiaga rinkta filmuojant Xiaomi Redmi 8 Pro kamera.

2. Vaizdo medžiagos paruošimas

Šio darbo metu sukurtai programinei įrangai paruošta vaizdo medžiaga, kurioje žaidėjas atlieka įvairius judesius su krepšinio kamuoliu. Siekiant supaprastinti vaizdo atpažinimo algoritmą, medžiagai keliami reikalavimai yra: aiškiai matomas žaidėjas, telpantis į kadrą ir užimantis nemažą dalį vaizdo. Vaizdo atpažinimui pagal spalvą algoritmui žaidėjas taip pat privalo dėvėti skirtingų spalvų pirštines, batus bei mušinėti atskiros spalvos kamuolį. Žaidėjas turi būti kiek galima labiau skirtis nuo fono, kad algoritmas veiktų kuo efektyviau, kadangi įvairių formų ir spalvų objektai fone pasunkina tikslų atpažinimą. Vaizdo medžiagoje žaidėjas mušinėja kamuolį ir atlieka įvairius judesius, dalis kurių pažeidžia taisykles, pavyzdžiui – pamušinėjęs kamuolį jį pasiima į rankas ir padaro keletą žingsnių. Programinė įranga turi atpažinti, jog tai – taisyklės pažeidimas. Vaizdo kokybė yra itin svarbi siekiant tikslių rezultatų, todėl buvo filmuojama kuo didesne raiška – Xiaomi Redmi Note 8 Pro kameros maksimali raiška yra 4K, 30 kadrų per sekundę.

3. Vaizdo atpažinimo metodai

Šiame darbe apibrėžtiems algoritmams, atpažįstantiems krepšinio taisyklių pažeidimus, reikalinga tam tikra pradinė informacija apie žaidėjo kūno dalis ar objektus (rankas, pėdas, kamuolį ir pan.). Ši informacija yra gaunama vaizdo įrašo kadrus apdorojant kompiuterinės regos metodais. Nagrinėjamus metodus galima sugrupuoti į tris grupes:

- Metodai, kurių pagalba remiantis kadre iš anksto žinomi spalvomis galima išskirti dominančius regionus (pvz. segmentavimas) bei metodai, papildomai apdorojantys gautus rezultatus (pvz. atliekant morfologines operacijas ar randant regionus apimančius kontūrus).
- Metodai, kuriais naudojantis atpažinimą vykdyti galima nepriklausomai nuo spalvų, pavyzdžiui, kamuolio atpažinimo algoritmas paremtas Hough transformacijomis, taip pat - judesio atpažinimas remiantis fono pašalinimu ir skirtumų tarp kadrų atradimu.
- Metodai paremti neuroniniais tinklais - žmogaus skeleto atpažinimas naudojantis OpenPose bei Lightweight modeliais.

Skyriaus pabaigoje visi šie metodai palyginami praktiškai, sukuriant minimalų žingsnių skaičiavimo algoritmą, įgyvendintą minėtais metodais, išryškinant jų privalumus bei trūkumus.

3.1. Spalvinis atpažinimas

Vienas iš būdų išskirti ir atpažinti objektus yra tada, kai žinome jų spalvą iš anksto. Šiam procesui egzistuoja daug metodologijų, pavyzdžiui, grupavimas, slenksčio nustatymas bei erdvės domenu paremti metodai, tokie kaip regionų auginimas, skaidymas ir sujungimas, grafiniai pjaustymai ar panašiai [VS12]. Vienas iš slenksčio nustatymo ar grupavimo privalumų yra tas, kad be iš anksto žinamų ir suklasifikuotų spalvų rėžių atpažinimas nereikalauja daugiau a priori žinių apie paveikslėlį ar jame egzistuojančius objektus ir yra paprastai įgyvendinami [VS12]. Spalvinio atpažinimo metodai, tarp jų ir slenksčio nustatymas, gali būti panaudojami tokioms problemoms spręsti, kaip žmogaus veido atpažinimas [WY01]. Pagal spalvų reikšmes galime išskirti regionus, juos reprezentuojant vienetukų ir nuliukų matricomis, ir su jomis atlikti įvairias logines operacijas tam, kad būtų galima palengvinti tolimesnį atpažinimą, pvz., pašalinant triukšmą.

Tačiau tai, jog siekiant atpažinti ranką ar kamuolį būtina iš anksto žinoti, kokios spalvos bus tie objektai, yra vienas iš praktinių trūkumų: darosi sudėtinga išskirti atskiras kūno dalis naudojantis informacija apie odos spalvą, kadangi tikėtina, kad ne tik žaidėjo rankos, kojos bei veidas bus tos pačios spalvos, bet galimai sutaps ir su kitų žaidėjų. Taip pat dažnai sunku iš anksto žinoti, kokios spalvos apskritai reikia ieškoti, kadangi skirtingi žmonės gali turėti skirtingą odos spalvą ar ji gali keistis priklausomai nuo apšvietimo [KMB07].

Dėl šių limitacijų siekiant ištirti spalvinių metodų efektyvumą ir pritaikomumą krepšinio žaidėjas turėtų iš anksto dėvėti tam tikros spalvos pirštines, batus bei valdyti ryškios spalvos kamuolį, tačiau realiame pasaulyje žaidėjus priversti nešioti skirtingų spalvų batus, pirštines būtų itin nepraktiška. Todėl šiame darbe taip pat bus nagrinėjami dar keli kompiuterinės regos atpažinimo būdai, nepriklausantys nuo spalvų.

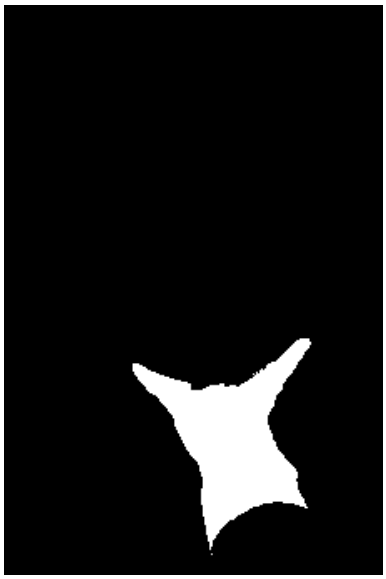
3.1.1. Segmentavimas

Segmentavimą kompiuterinėje regoje galima suprasti kaip radimą grupės panašių pikselių. Kai paveikslėliai segmentuojami pagal spalvą, panašumą galima išmatuoti pagal reikšmių skirtumą remiantis RGB arba HSV spalvų erdvėmis. Naudojantis vienu iš segmentavimo būdų - spalvos slenksčio (arba rėžių) nustatymu – OpenCV įgalina ganėtinai paprastai išskirti dominantį regioną, kadangi užtenka apsibrėžti tam tikrą intervalą spalvų paveikslėlyje, reprezentuotame trimate matrica, ir nufiltruoti matricos reikšmes nepatenkančias į duotąjį intervalą.

RGB modelyje spalvą koduoja trimis reikšmėmis, nusakančiomis raudonumą, žalsvumą ir mėlynumą. Tačiau naudoti jį spalvinio atpažinimo metoduose yra sudėtinga - keičiantis apšvietimui nenuspėjamai gali pasikeisti spalvos RGB reikšmė. Egzistuoja algoritmai, gebantys atskirti apšvietimo pokyčius RGB modelyje [KPS03], tačiau HSV modelis leidžia informacija apie atspalvį, ryškumą ir šviesumą saugoti kaip atskiras reikšmes. Šis modelis puikiai tinka spalvų atpažinimui, kadangi H (atspalvio) reikšmė mažai kinta paveikslėliuose atsiradus šešėliams [LBC02] ar miglotumui [WC15]. Atspalvis dažniausiai išlieka tas pats nepriklausomai nuo pašalinių efektų, todėl lengva nusistatyti reikšmių intervalą. Konvertavimas iš RGB ir HSV yra pakankamai trivialus, OpenCV jis yra įgyvendintas cvtColor() funkcija.

Turint intervalą užtenka nufiltruoti visus pikselius, kurių reikšmė nepatenka į duotą intervalą. Tiems, kurie patenka, loginėje matricoje priskiriamas 1, o tiems, kurie ne - 0. Rezultate gauname išskirtą regioną kur žinome, kad tai - tam tikras dominantis objektas, su sąlyga, kad jis vienintelis kadre buvo tos spalvos, pagal kurią vyko filtravimas.

(a) Vaizdas prieš segmentavimo pagal pirštinės spalvą



(b) Vaizdas po segmentavimo pagal pirštinės spalvą

1 pav. Segmentavimo pavyzdys

3.1.1.1. Dominančių objektų spalvų rėžiai

Atliekant bandymus pastebėta, jog net ir naudojantis HSV spalvų erdve, dominančių objektų spalvų rėžiai itin skiriasi priklausomai nuo apšvietimo. Šiame darbe naudoti spalvų rėžiai: //@todo

correct the values

- Žalsvo kamuolio - nuo [61, 20, 32] iki [84, 255, 200]
- Geltonų pirštinių - nuo [„] iki [„]
- Raudonų batų - nuo [„] iki [„]

Tinkamų HSV rėžių radimo procedūra buvo pasirinkus tam tikrą pradinį intervalą patikrinti, ar nufiltravus netinkamus pikselius, išskirtas regiono plotas atitinka dominančio objekto plotą. Jei išskirto regiono plotas yra mažesnis - vadinasi, rėžiai yra per siauri, tad jie buvo didinami, tuo tarpu jei plotas yra didesnis - rėžiai mažinami.

Šitaip bandymų būdu buvo sukalibruoti paruoštai vaizdo medžiagai labiausiai tinkami spalvų rėžiai. Dėl galimų artifaktų atsiradimo (pvz. triuškmo ar nesusijusių objektų) nufiltravus pikselius pagal spalvos rėžius, buvo bandoma juos turėti kuo siauresnius, tačiau kaip vėliau pasirodė iš rezultatų, net ir menkiausias apšvietimo pokytis turėjo itin didelę įtaką spalvoms. Kiekvienai vaizdo medžiagai, nufilmuotai skirtingose aplinkose, ieškomų spalvų reikšmes gali tekti kalibruoti iš naujo.

Taip pat buvo pastebėta, jog lauke filmuotuose vaizdo įrašuose spalvos kito mažiau, nei kambaryje - tai, galimai, yra geresnio apšvietimo pasėkmė.

3.1.2. Morfologinės transformacijos

Atlikus segmentavimą pagal spalvą, rezultate dažnai lieka triukšmo ar bereikalingų artifaktų, kurie gali affectinti tolimesnį apdorojimą. @todo fix

Tokiais atvejais apsivalyti nuo triukšmo gan dažnai naudojamos slenkstinės (*angl.* threshold) operacijos [Sze21, 112]. Apsibrėžus tam tikrą filtrą, su juo galima atlikti operacijas ant matricų. Šiame darbe naudotos operacijos yra erozija ir plėstis. Erozija naudojama tada, norima sumažinti turimų atskirų regionų paveikslėlyje plotus, nuardant kraštus. Jei objektai yra pakankamai maži, o erozijos operacija - stipri (?), tuomet taip pašalinamas triukšmas. Formaliai erozija buvo aprašyta 1980 metais K. Y. Yoo [HSZ87] kaip morfologinė operacija, sukombinuojanti dvi aibes naudojant vektorių atimtį aibių elementams. Matematiškai tai galima aprašyti tokia formule:

$$A \ominus B = \bigcap_{b \in B} (A)_{-b}. \tag{1}$$

1 lygtyje A žymi dominantį paveikslėlį, išreikštą matrica, o B - filtrą arba kitaip - struktūrinį elementą, su kuriuo A matricoje pakartotinai atliekamos sankirtos.

Erozijai priešinga operacija yra plėstis [HSZ87]. Ši operacija leidžia padidinti išskirtų regionų plotą. Plėstis naudinga tuomet, kai po erozijos operacijos objektai praranda per daug ploto. Plėstį galima užrašyti šitaip:

$$A \oplus B = \bigcup_{b \in B} (A)_{b}. \tag{2}$$

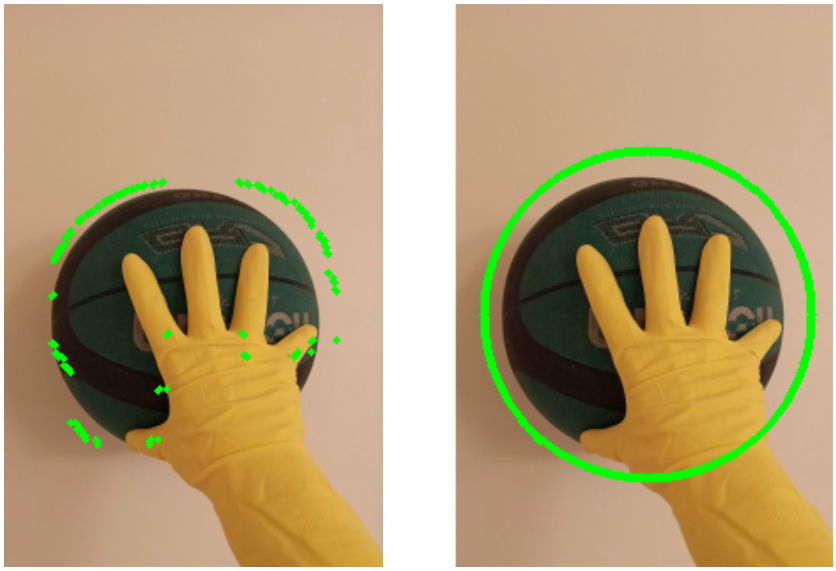
Atliekant bandymus pastebėta, jog siekiant geriausių rezultatų šalinant triukšmą, plėstį reikėtų vykdyti po erozijos, priešingu atveju triukšmas gali nebūti panaikintas.

OpenCV įgyvendina funkcijas erode ir dilate, kurios priema paveikslėlio matricą ir struktūrinį elementą. Taip pat egzistuoja funkcija morphologyEx(), kuri, priklausomai nuo paduodamų parametrų keičia savo veikimą: su MORPH_OPEN atliekama plėstis po erozijos, su MORPH_CLOSE - atvirkščiai.

3.1.3. Kamuolio kontūrų radimas

Siekiant iš paveikslikų išgauti daugiau kontekstualios informacijos, pavyzdžiui, ar kamuolys yra rankose, išsiskirti tarpusavy nesusikertančius regionus neužtenka. Tam, kad gauti šią informacijos dalį, reikia rasti, kada rankos ir kamuolio regionai susikerta, t.y. jų matricų sankirta yra netuščia. Tam reikia žinoti, kokią erdvę apima kamuolys - šios informacijos trūksta po segmentavimo pagal spalvą, jei kamuolys yra uždengtas, pavyzdžiui, rankos.

Vienas iš būdų šią problemą išspręsti - pabandyti atspėti, kokią sritį užima kamuolys randant jo kontūrus. Skritulių kontūrams rasti 1991 metais E. Welzl pasiūlė algoritmą, rekursiškai apskaičiuojantį tam tikrai taškų aibei mažiausią ją apimantį apskritimą [Wel91]. Šis algoritmas - tiesinio kompleksiškumo, jį įgyvendina OpenCV funkcija minEnclosingCircle().



(a) Kamuolio kontūrai prieš mažiausio aibę apimančio apskritimo radimo algoritmą

(b) Kamuolio kontūrai po mažiausio aibę apimančio apskritimo radimo algoritmo

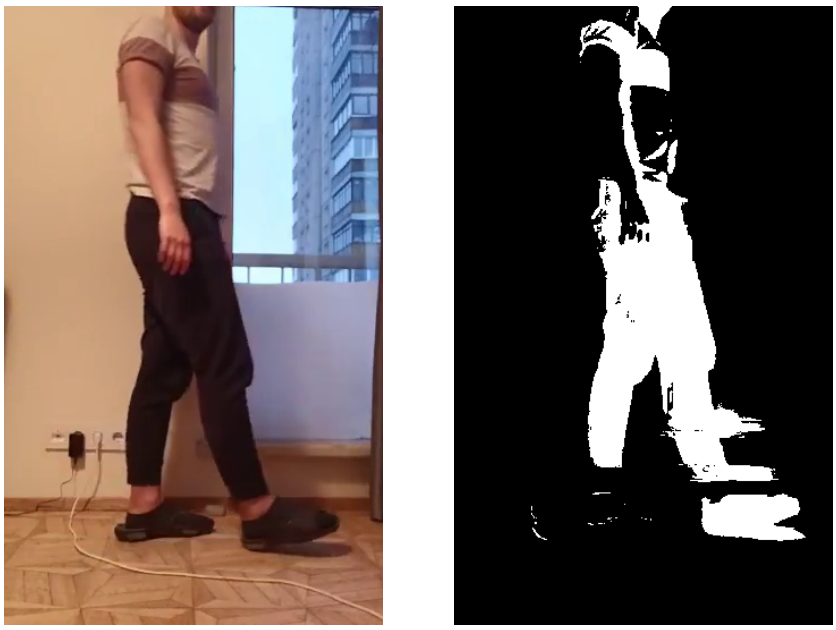
2 pav. Kontūrų radimo algoritmo pavyzdys

3.2. Atpažinimas remiantis skirtumais

3.2.1. Fono pašalinimas

Fono pašalinimas yra vienas iš esminių metodų kompiuterinėje regoje, naudojamas išskirti dominančius vaizdus ir pašalinti nereikalingus statiškus objektus iš fono. Pavyzdžiui, jei yra fil-

muojama lauke, fone besimatantys medžiai gali trukdyti tolimesniam atpažinimui, tad žinant, jog mus domina tik žaidėjas, pašalinius objektus galima pabandyti pašalinti. Vienas iš būdų yra iš anksto turėti fono paveiksliuką ir apdorojant kitus kadrus jį išimti, bet tai dažnai nepasiteisina dėl to, kad fonas gali kisti – gali atsirasti šešėlių, pasikeisti apšvietimas, gali būti įvairaus pašalinio judėjimo, pavyzdžiui – linguojantys medžiai, vaikštantys sirgaliai ir pan. 2002 m. P. KaewTraKulPong pasiūlė algoritmą, išsprendžiantį šią problemą [KB02]. Algoritmas kiekvieną pikselį sumodeliuoja į Gauso skirstinį pagal tai, kiek pikselio spalva keičiasi bėgant laikui. Kuo mažiau pikseliai keičiasi, tuo didesnė tikimybė, kad jie priklauso fonui. Naudojantis šiuo atradimu galima sėkmingai atsikratyti fono, taip išskiriant mus dominantį vaizdą. Šis metodas ypač tinkamas atpažinti judėjimui – vietoje stovintis žaidėjas bus priskirtas fonui, tačiau jam judant, lengvai galima išskirti jo siluetą.



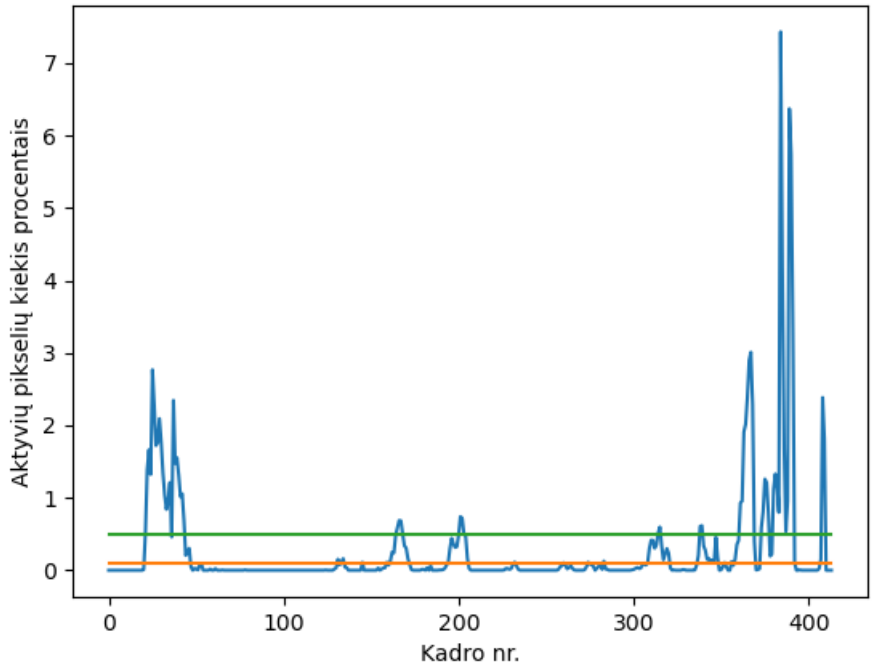
(a) Vaizdas prieš fono pašalinimą. (b) Vaizdas po fono pašalinimo.

3 pav. Fono pašalinimas naudojantis Gauso skirstiniu paremtu fono segmentavimo algoritmu.

Paveikslėlyje matome, jog algoritmas atpažino žmogaus siluetą. Šiame pavyzdyje ranka priskirta fonui dėl to, jog spalva sutampa su sienos spalva, tačiau kitos kūno dalys išskiriamos iš fono.

3.2.2. Judesio atpažinimas

Žaidėjui vaikštant, kai kurios kūno dalys juda greičiau, nei kitos. Kūnas išlieka santykinai statiškas palyginus su kojomis. Žingsniavimo mechanika yra tokia, kad atliekant žingsnį, viena koja lieka vienoje vietoje, kol kita yra perstatoma iš vienos pozicijos į kitą. Tam, kad pagauti tą kojos judesį, galima paprasčiausiai iš antro kadro atimti pirmą. Padalijus kadrą į dvi dalis – viršutinę ir apatinę – ir darant prielaidą, jog apatinėje dalyje matysis tik kojų judesiai, galima teigti, kad jei atėmus antrą kadrą iš pirmo yra skirtumas, buvo atliktas žingsnis. Koją pastačius ant žemės, tam tikrą laiką skirtumai sumažėja iki nustatytos ribos – pasinaudojus visa šia informacija galima apskaičiuoti, kiek žingsnių buvo atlikta.



4 pav. Aktyvių pikselių kiekis kiekviename kadre, indikuojantis žmogaus judėjimą.

4 paveikslėlyje matyti, kiek buvo aktyvių pikselių kiekvieną kadrą panaudojus fono pašalinimą bei kadrų skirtumo algoritmus. Ši informacija rodo judėjimo kiekį tam tikrame kadro regione. Žalia linija žymi minimalų judėjimo kiekį, kuris gali atsirasti dėl triukšmo ir pašalinio judėjimo. Viršijus šią liniją galima teigti, jog žaidėjas šiuo metu juda. Oranžinė linija žymi ribą, iki kurios galima sakyti, jog jokio judesio nėra, t.y. žaidėjas pastatė koją ir ruošiasi atlikti kitą žingsnį.

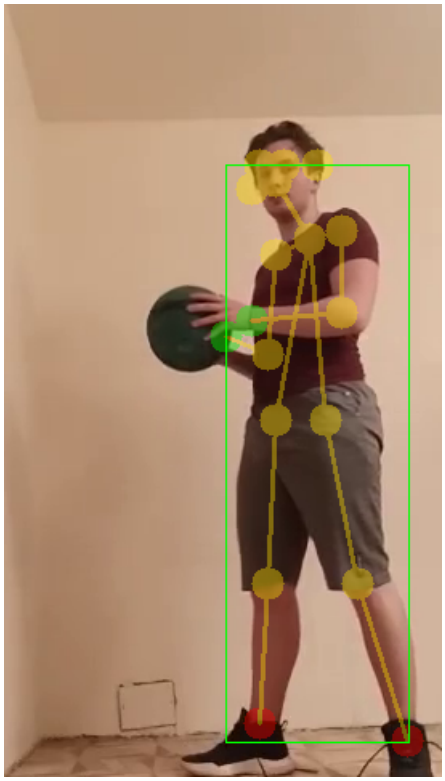
3.3. Žmogaus kūno dalių atpažinimas neuroniniais tinklais

Šiuolaikinėje kompiuterinės regos srityje daug naudos galima gauti pasinaudojus neuroninių tinklų pagalba. Neuroniniai tinklai naudojami sprendžiant įvairias problemas, šiam darbui aktualiausia yra objektų atpažinimo problema. Turint tam tikrą kadrą, taisyklių pažeidimo algoritmui būtina, kad būtų atskirtos rankos, kojos ir kamuolys. Žmogaus kūno dalių klasifikavimas yra sudėtinga problema, kadangi dauguma algoritmų yra priklausomi nuo surinktų duomenų. Kompleksija tampa akivaizdi sprendžiant sporto problemas, kadangi labai dažnai žaidėjai daro įvairiausius judesius, apsirengę įvairiausiais rūbais ir pan., kas apsunkina žaidėjo kūno dalių atpažinimą [APG+14]. Tam reikalinga turtinga ir didelė duomenų aibė, dėl ko neuroninio tinklo apmokymo laikas išauga.

Bene visi šiuolaikiniai metodai remiasi tuo pačiu principu: apmokinamas neuroninių tinklų modelis, jiems paruošiant teigiamus ir neigiamus pavyzdžius (pvz., siekiant sukurti kojų atpažinimo algoritmą, paveiksliukai, kuriuose yra koja pažymimi kaip teigiami, tie, kuriuose kojos nėra tampa neigiamais). Mokinimo procesas gali trukti daug laiko – kartais net savaites, todėl daug paprasčiau yra pasinaudoti jau sukurtu modeliu. Vienas iš modelių, pavadinimu OpenPose, buvo pasiūlytas 2018 metais [CHS+19]. Modelio architektūra paremta konvoliuciniais neuroniniais tinklais. Modelis suskirstytas į dvi šakas. Viena jų priskiria tikimybes, kad tam tikras regionas yra tam

tikra kūno dalis, kita – asociacijas tarp skirtingų kūno dalių. Atpažinimas vyksta keliais etapais, siekiant gauti kuo tikslesnius rezultatus.

Šiame darbe kuriamas vaizdo atpažinimo algoritmas kiekvieną kadrą pateikia neuroniniui tinklui, kuris apskaičiuoja ir grąžina šiluminį žemėlapį (*angl.* heatmap) su sąnarių koordinatėmis ir tikimybėmis, kad tose koordinatėse jas galima rasti. Vėlesnis apdorojimas pagal grąžintas koordinates kadre skirtingomis spalvomis sužymi dominančias kūno dalis, aplink gautas koordinates nupiešiant fiksuoto spindulio ir spalvos skritulį. Taisyklių pažeidimo atpažinimo algoritmui aktualios yra rankų ir pėdų sritis. Verta pabrėžti, jog randamos yra sąnarių pozicijos, ir nuo algoritmo priklauso, ar tokia informacija yra pakankamai tiksli.



5 pav. OpenPose algoritmo pagalba atpažintos kūno dalys.

Vienas iš algoritmo trūkūmų yra tai, kad be optimizavimo su kiekvienu kadru gauti rezultatą užtrunka iki 0.5 sekundės. Galimas optimizavimo būdas yra sumažinti kadro dydį prieš pateikiant jį į neuroninį tinklą, tačiau tokiu atveju rezultatų tikslumas yra atvirkščiai proporcingas kadro dimensijom.

2018 m. buvo pasiūlytas OpenPose modelio patobulinimas, įgalinantis žmogaus kūno atpažinimą beveik realiu laiku [Oso18]. Lightweight optimizuoja OpenPose modelio architektūrą sumažinant sluoksnių skaičių, ryškiai pagerinant atpažinimo greitį nežymiai prarandant tikslumą.

Atliekant bandymus su vaizdo medžiaga pastebėta, jog Lightweight modelis žaidėjo pozą atpažįsta ne tik greičiau, bet ir tiksliau. Eksperimentui buvo sukurti penki vaizdo įrašai, kuriuose žmogus atlieka tam tikrus veiksmus. Tikslumas buvo apskaičiuotas už teisingą kūno dalies atpažinimą kiekviename kadre pridedant 1 tašką, už neteisingą atpažinimą atimant 1, o gautą rezultatą padalinant iš kūno dalių skaičiaus, kurį atpažįsta modelis.

1 lentelė. OpenPose ir Lightweight modelių palyginimimas apdorojant vienodus vaizdo įrašus

Modelis	Vidutinė trukmė apdoroti vieną kadrą (ms)	Vidutinis rezultatų tikslumas (%)
OpenPose	581	62
Lightweight	163	91

Dalį Lightweight modelio tikslumo pranašumo galėjo lemti OpenPose parametrų ir paduodamo kadro dimensijų neatitikimas, o spartumo - tai, jog duomenų apskaičiavimas OpenPose tinklu buvo įgyvendintas su OpenCV dnn moduliu, kuris bandymų metu nebuvo sukonfigūruotas naudoti GPU akceleravimą. Kadangi po optimizacijos Lightweight modelis tenkino tikslumo ir greičio lūkesčius, toliau naudoti jį.

3.3.1. Atpažinimo neuroniniais tinklais optimizavimas

Atliekant bandymus pastebėta, jog atpažinimą Lightweight modeliu galima optimizuoti. Viena iš optimizacijų, kuri buvo atlikta, tai skaičiavimo naštos perkėlimas ant GPU. Tam panaudota CUDA - NVIDIA sukurtas API paraleliems skaičiavimams [NVI21]. Integravus OpenCV su CUDA, paveikslėlių apdorojimas gali pagreitėti iki 30 kartų [Ope21b]. Be GPU akceleracijos Lightweight tinklas vieną kadrą užtrukdavo apdoroti iki 680 ms.

Pastebėta, jog apdorojimo greičiui įtakos turi ir kadro dimensijos. Jei į neuroninį tinklą paduodamo kadro plotis yra didesnis, nei jo aukštis, apdorojimo greitis pakyla iki 5 kartų. Tačiau tai buvo mitiguota atnaujinus OpenCV versiją iš 4.0.1 į 4.5.1 - po šio atnaujinimo, apdorojimo greičiai tapo nebepriklausomi nuo santykio tarp paveikslėlio aukščio ir pločio.

Dar viena išbandyta optimizacija - neuroninio tinklo pradinio sluoksnio įvesties dydžio sumažinimas. Dvigubai sumažinus įvesties kanalų skaičių iš 256 į 128 rezultatų tikslumas pasikeitė sąlyginai nedaug. Žiūrint į vaizdo įrašą, plika akimi galima pamatyti, jog atpažintas skeletas yra labiau ”klibantis”, t.y. kiekvienos kūno dalies koordinatės nežymiai skiriasi, kai įvesties dydis yra mažesnis, tačiau pačios kūno dalys dažnai lieka atpažintos teisingai. Siekiant tikslesnių rezultatų, siūloma pasilikti prie 256 kanalų skaičio, tačiau norint vaizdą apdoroti realiu laiku, galima rasti kompromisą tarp tikslumo ir greičio.

Atliekant žmogaus skeleto atpažinimo bandymus su mažesnės rezoliucijos kadrais spartumo bei tikslumo pagerėjimų nepastebėta.

2 lentelė. Lightweight optimizavimas sumažinus įvesties kanalų skaičių apdorojant vienodus vaizdo įrašus

Kanalų skaičius	Vidutinė trukmė apdoroti vieną kadrą (ms)	Vidutinis rezultatų tikslumas (%)
256	28	91
128	16	86

3.4. Kamuolio atpažinimas Hough transformacija

Kompiuterinės regos srityje vieni iš pirmųjų algoritmų, skirtų atpažinti objektus pagal jų formas, yra Hough transformacijos. Hough transformacija paremtas algoritmas, skirtas skritulių bei

apskritimų radimui, buvo aprašytas H. K. Yuen, J. Princen, J. Illingworth ir kitų [YPI+90]. Būtent šį algoritmą OpenCV įgyvendina kaip numatytąjį, bandant rasti apskritimų kontūrus paveikslėliuose.

Patikrinus algoritmą su vaizdo įrašu tapo akivaizdu, jog šis jis tinka tik idealios formoms sferoms ir apskritimams, tačiau judėdamas kamuolys praranda apvalią formą. Darant prielaidą, jog tokiu atveju kamuolio forma tampa panaši į elipsės, kamuolio radimui išbandytas Hough transformacija paremtas algoritmas elipsėms rasti [XJ02], tačiau šio algoritmo sparta pasirodė itin prasta - atlikus bandymus su keliais vaizdo įrašais, vienam kadrui apdoroti užtrukdavo iki 10 minučių.

(a) Atrastas kamuolys

(b) Kamuolys nerastas, kadangi jis deformuotas dėl judėjimo

6 pav. Kamuolio kontūrų radimas

Dėl šių trūkumų tolimesniam vaizdo atpažinimui nuspręsta taikyti kamuolio atpažimo pagal spalvą metodus, kadangi preliminarūs rezultatai parodė, jog kadrus apdoroti spalviniu atpažinimu paremtais algoritmais užtrunka šimtus kartų mažiau laiko, nei elipsių paieškos algoritmas, taip pat - jie yra tikslesni, pasirinkus teisingus spalvos rėžius.

3.5. Nagrinėtų atpažinimo metodų palyginimas

Tam, kad būtų palygintas šių metodų spartumas bei tikslumas tarpusavy, nuspręsta juos panaudoti sprendžiant praktinę problemą: vaizdo įraše suskaičiuoti, kiek žmogus atliko žingsnių. Žingsnių skaičiavimo algoritmas - gana paprastas, nereikalaujantis daug papildomų sudėtingų skaičiavimų, tad jis puikiai tinka įvertinti pačio vaizdo atpažinimo metodo tinkamumą.

Žingsnių skaičiavimo algoritmas yra žingsnių taisyklės pažeidimo atpažinimo dalis, plačiau apibūdintas 4 skyriuje. Žmogaus skeleto (pozos) bei spalvinio atpažinimo metodai įgyvendinti tuo pačiu algoritmu, kurio metu pėdų (batų) regionai yra išskiriami, ir laikoma, jog žingsnis yra atliktas tada, kai išskirti regionai persikloja - tai reiškia, jog žiūrint iš šono, įprasto vaikščiojimo metu viena koja trumpai pridengė kitą. Neuroninių tinklų atpažinimo metu prie atpažintų kojų priskiriamos spalvos, tolimesnis atpažinimas panašus į spalvinį.

Metodo, besiremiančio skirtumais tarp kadrų, žingsnių atpažinimo procesas aprašytas 3.2.2 poskyryje.

Visuose vaizdo įrašuose žmogus dėvi raudonus batus, tam, kad veiktų spalvomis paremti atpažinimo metodai ir juos būtų galima lyginti su kitais. Vaizdo įrašų sukurta septyni, juose - skirtingas apšvietimas, nuotolis iki žmogaus, žingsniai atliekami skirtingu tempu ir kamera filmuoja skirtingais kampais.

Algoritmai buvo patikrinti kelis kartus su tais pačiais įrašais, siekiant sumažinti įvairių kitų kompiuteryje vykstančių procesų įtaką vaizdo apdorojimo greičiui.

3 lentelė. Vaizdo įrašai ir jų aprašymai

Vaizdo įrašo nr.	**Aprašymas**	**Trukmė (s)**
1	Besikeičiantis apšvietimas, žaidėjas eina įstrižai nuo kameros, vaizdo įrašas filmuotas vertikaliai	12
2	Žaidėjas eina atgal, kamera filmuoja iš priekio, filmuota horizontaliai	2
3	Žaidėjas vaikšto gan tolokai nuo kameros, filmuota horizontaliai	5
4	Žaidėjas labai arti kameros, filmuota horizontaliai	7
5	Žaidėjas vaikšto gan tolokai nuo kameros, filmuota vertikaliai, žaidėjas matomas pilnai	8
6	Prastas apšvietimas, žaidėjas gan arti kameros, filmuota horizontaliai	8
7	Žaidėjas toli nuo kameros, atsiranda antras žaidėjas, filmuota horizontaliai. Tikimasi, kad bus suskaičiuota abiejų žaidėjų žingsnių suma	10

4 lentelė. Spalvinio vaizdo atpažinimo žingsnių skaičiavimo eksperimento rezultatai

Vaizdo įrašo nr.	**Atlikta žingsnių**	**Algoritmo suskaičiuota žingsnių**	**Vidutinė vaizdo apdorojimo trukmė (ms)**
1	10	13	6035
2	3	0	4681
3	4	2	12600
4	7	7	4126
5	5	5	4029
6	7	6	3949
7	10	6	4235

5 lentelė. Vaizdo atpažinimo pagal skirtumus žingsnių skaičiavimo eksperimento rezultatai

Vaizdo įrašo nr.	**Atlikta žingsnių**	**Algoritmo suskaičiuota žingsnių**	**Vidutinė vaizdo apdorojimo trukmė (ms)**
1	10	9	4807
2	3	4	5488
3	4	4	11529
4	7	10	2164
5	5	2	3147
6	7	8	3012
7	6	4	3983

6 lentelė. Vaizdo atpažinimo su neuroniniais tinklais žingsnių skaičiavimo eksperimento rezultatai

Vaizdo įrašo nr.	**Atlikta žingsnių**	**Algoritmo suskaičiuota žingsnių**	**Vidutinė vaizdo apdorojimo trukmė (ms)**
1	10	8	12859
2	3	0	7196
3	4	4	14565
4	7	0	1257
5	5	5	11638
6	7	8	12438
7	6	6	10045

Palyginę rezultatus matome, jog neuroniniais tinklais paremtas atpažinimas yra tiksliausias, nors ir apdoroti vaizdo įrašą užtrunka daugiausiai laiko. Atpažinimas remiantis skirtumais - greičiausias, tačiau mažiausias yra jo tikslumas, o atpažinimas paremtas spalvomis yra per vidurį.

Spalvomis paremtas atpažinimas yra mažiausiai tikslus, kai varijuoja apšvietimas arba jis yra prastas, taip pat - jei kadre yra kitų objektų ar žaidėjų, kurių spalvos sutampa su dominančiomis spalvomis. Taip pat šiuo atpažinimu paremtas algoritmas vienintelis, galintis apskaičiuoti žaidėjo atliktus žingsnius iš priekio.

Iš rezultatų apdorojant vaizdo įrašą nr. 4 galime pastebėti, jog neuroniniais tinklais paremtas atpažinimo metodas neatpažįsta žingsnių, kai žaidėjas yra per arti kameros ir nebesimato jo pilnos figūros, tuo tarpu vaizdo įrašas nr. 7 parodo, jog šis atpažinimo metodas puikiai susidoroja su atvejais, kai yra daugiau žmonių, nei vienas.

Apibendrinus - spalviniai vaizdo atpažinimo metodai tinka tuomet, kai prie dominančių objektų galime priskirti unikalias spalvas, jos nekinta, neatsiranda naujų, o apšvietimas išlieka puikus - visa tai indikuoja, jog vaizdo įraše fonas ir aplinka turi būti griežtai kontroliuojami ir iš anksto paruošti. Skirtumais paremti atpažinimo metodai labiausiai tinka tais atvejais, kai juos naudojantys algoritmai yra paprasti bei nereikalaujantys didelio tikslumo, siekiant išgauti didžiausią spartumą. Neuroniniais tinklais paremti atpažinimo metodai yra itin tikslūs, tačiau apdorojimo trukmė yra šiek tiek ilgesnė.

4. Krepšinio taisyklių pažeidimo atpažinimas

Šiame darbe bus nagrinėjamos kelios taisyklės.

4.1. Žingsnių taisyklė

Žingsnių (*angl.* traveling) taisyklė yra aprašyta oficialioje NBA taisyklių knygelės 10 taisyklės 13 skyriuje. Jos formuluotė, kuria remiasi analizė, atlikta šiame darbe, yra tokia:

- Žaidėjas, varydamas kamuolį, gali padaryti du žingsnius prieš sustojant arba kamuolį metant ar atiduodant [NBA21].

Būdų pažeisti šią taisyklę yra gausu, tačiau šiame darbe fokusuojamasi į atvejį, kai žaidėjas suėmęs kamuolį atlieka daugiau nei du žingsnius. Tam, kad atlikti žingsniai būtų laikomi kaip pažeidžiantys taisyklę, kamuolys turi būti nemušamas į žemę, o laikomas rankose - krepšinyje tai dažniausiai nutinka prieš metant kamuolį į krepšį, kai yra atliekamas dvižingsnis (NBA tai vadinama kamuolio surinkimu (*angl.* gathering)). Taigi, algoritmas, siekiantis atpažinti šios taisyklės pažeidimą, privalo taip pat gebėti atpažinti, kada kamuolys yra rankose, ir tuomet skaičiuoti atliktų žingsnių skaičių.

4.2. Žingsnių taisyklės pažeidimo atpažinimo algoritmas

Algoritmo reikalavimai: nustatytos rankų, kojų bei kamuolio vietos paveikslėlyje. Apibendrintas algoritmas užsirašytų šitaip:

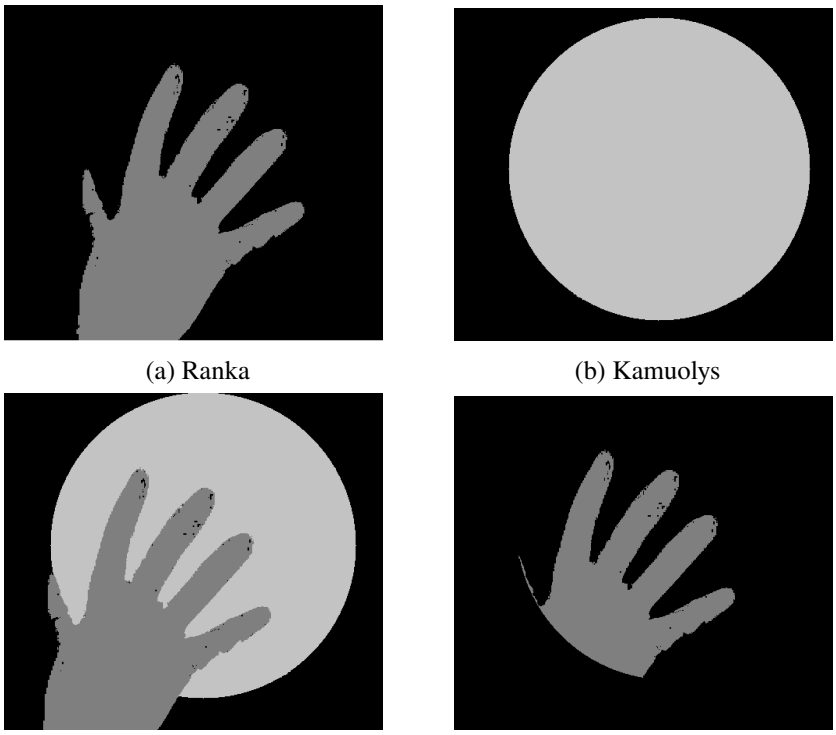
1. Nuskaitomas kadras.
2. Nustatoma, ar kamuolys yra laikomas rankose.
3. Tol, kol kamuolys yra rankose, vykdomas žingsnių skaičiavimo algoritmas.
4. Jei per intervalą, kai kamuolys yra laikomas rankose, buvo atlikti daugiau nei 2 žingsniai - grąžinamas rezultatas, jog taisyklė pažeista.

Algoritmui vienas po kito pateikiami vaizdo įrašo kadrai, kurie yra apdorojami po vieną. Po anksčiau aprašytų atpažinimo metodų turima pakankamai informacijos nustatyti, kur kiekviename kadre yra ranka, kojos ir kamuolys. OpenCV įgalina visų minėtųjų metodų metu išgautą informaciją reprezentuoti loginėse vienetų bei nulių matricose, kur vienetai žymi objekto buvimą tam tikrame plote.

Tam, kad būtų nustatyta, ar kamuolys yra laikomas rankose, su išskirtomis kamuolio ir rankų užimamą erdvę kadre reprezentuojančiomis loginėmis matricomis atliekama Hadamardo daugyba, t.y. kiekvienam matricos elementui atliekama disjunkcija su toje pačioje vietoje esančiu elementu iš kitos matricos.

Spalvinio atpažinimo metodų sukurtos matricos yra pačių kūno dalių projekcija, tačiau naudotas neuroninių tinklų modelis grąžina apytikslias sąnarių koordinates (riešo, kulkšnies), aplink kuriuos nupiešiamas skritulys - iš to galima spręsti, kur maždaug yra ranka ar pėda.

Rezultate gaunama matrica, žyminti sritį, kur rankos bei kamuolys persikloja. Tai reiškia, jog kamuolio ir rankų pozicija kadre sutampa, tačiau tam tikra rankos dalis yra arčiau kameros ir šiek tiek užstoja kamuolį.



(a) Ranka

(b) Kamuolys

(c) Ranka arba kamuolys

(d) Ranka ir kamuolys (Hadamardo daugyba)

7 pav. Kamuolio ir rankų santykio nustatymas

Žingsnių skaičiavimas yra gan kompleksiška problema: ganėtinai sunku apibrėžti universalų būdą atpažinti atliktus žingsnius iš bet kokio kampo. Paprastumo dėlei nuspręsta žingsnių skaičiuoti tik iš žiūrint šono. Žingsnių skaičiavimo algoritmas remiasi išskirtų kojų regionų kontūrų skaičiavimu. Jei kontūrų suskaičiuota vienas, o prieš tai jų buvo du, galima teigti kad 1) Nerasta informacijos apie vieną iš pėdų (išėjo iš kadro arba nepavyko atpažinimas) arba 2) Žingsnis, žiūrint iš šono, yra toje fazėje, kai viena pėda pridengia (arba yra netoli) kitą. Tam, kad būtų galima atmesti pirmąjį atvejį, laikoma informacija apie atpažintus objektus - jei jų yra vienas, tuomet tai, kad rastas tik vienas kontūras, nekeičia turimų suskaičiuotų žingsnių kiekio.

Žingsnių skaičiavimo algoritmą, kuris taikytinas reikiamą informaciją išgavus spalviniais atpažinimo metodais arba neuroninių tinklų pagalba, galima apibrėžti taip:

1. Išskiriami kojų (batų) regionai.
2. Suskaičiuojami kontūrai.
3. Jei kontūrų yra daugiau nei vienas - pasižymima, jog sankirta neįvyko.

4. Jei kontūras yra vienas, sankirta dar nebuvo įvykus ir išskirtų regionų skaičius yra daugiau, nei vienas- pasižymima, jog sankirta įvyko ir padidinamas žingsnių skaičius.

(a) Žingsnio pradžia, randami du kontūrai

(b) Žingsnis tęsiasi, kontūrai vis dar du

(c) Įpusėjęs žingsnis, kontūrai susilieja į vieną

8 pav. Žingsnio atpažinimo procesas skaičiuojant kontūrus

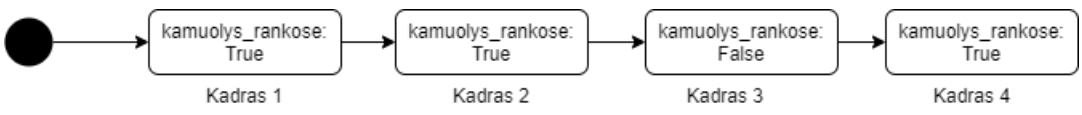
Šis žingsniavimo atpažinimo metodas taikytinas tik tada, kai žingsnis yra kiek galima labiau natūralus, t.y. kojos nėra keliamos per aukštai ir pėdos įpusėjus atliekamam žingsniui yra netoli viena kitos. Lankstesnis yra skirtumais tarp kadrų paremtas algoritmas, atpažįstantis, kada atliktas bet koks judėjimas, tačiau jo tikslumas yra prastas.

Turint informaciją apie kamuolio laikymą tam tikrame laiko intervale ir per jį atliktų žingsnių kiekį, galime daryti išvadą, kad žingsnių taisyklė yra pažeista, kai per minėtą intervalą yra atlikti daugiau nei 2 žingsniai.

4.2.1. Būsenos saugojimas

Atliekant pradinius bandymus su algoritmu pastebėta, jog viena iš problemų, kylančių iš metodų, nustatančių, ar kamuolys yra rankose, tai išskirtų rankų bei kamuolio regionų plotų nepastovumas priklausomai nuo apšvietimo ir kitų faktorių. Tai reiškia, jog laiko intervalas, kada kamuolys yra laikomas rankose, gali būti su trūkiais, indikuojančiais, kad tą laiko momentą nepavyko nustatyti, ar kamuolys yra rankose. Žingsnių skaičiavimo algoritmas skaičiuoja žingsnius tik tada, kai kamuolys yra rankose, tad intervalui nutrūkus dėl netikslaus atpažinimo, skaičiavimas prasidės iš naujo.

Informacija apie kamuolio buvimą rankose yra renkama ne iš vieno kadro, o iš kelių, ir darant prielaidą, jog rankose esančio kamuolio judėjimo greitis yra arti nulio, galima teigti, jog tokia būsena turėtų nepasikeisti tam tikrą skaičių kadrų. Kitaip tariant, su kiekvienu kadru išsaugoma būsena, ar kamuolys yra rankose ar ne:



9 pav. Būsenos saugojimo pavyzdys kiekviename kadre

Išsaugojus būseną suskaičiuojama, kokia buvo jos vidutinė reikšmė per pastaruosius n kadrų:

$$M_{vid.\ būsena} = \sum_{n=i-x}^{i} \frac{S(n)}{n}, \tag{3}$$

kur *i* žymi einamo kadro numerį, *x* - konstantą, nurodančią dominančių kadrų skaičių, *S* - *n*-tojo kadro būseną, kur 1 indikuoja kamuolio buvimą rankose, o 0 - atvirkščiai.

Jei vidutinė būsena yra didesnė už tam tikrą iš anksto nustatytą konstantą, pvz., 0.3, galima teigti, jog kamuolys vis dar yra rankose - tam, kad ši reikšmė pasikeistų, turi praeiti bent keli kadrai, pakeičiantys būsenos vidurkį.

Šis būsenos skaičiavimo būdas atliekant pradinius bandymus su programinės įrangos kodu pasirodė gana naudingas siekiant sumažinti įvarius būsenų svyravymus, kylančius iš atpažinimo netikslumo, tad jis taip pat yra įgyvendintas ir kitame taisyklės pažeidimo algoritme.

4.3. Dvigubo varymo taisyklė

Pagal NBA taisyklių knygelės 10 taisyklės 2 skyriaus c dalį, žaidėjas negali tęsti kamuolio varymo jei jį pats sustabdė. Scenarijai, kada varymas laikomas sustabdytu, aprašyti 4 taisyklės 2 skyriuje. Varymas sustoja, kai kamuolys yra suimamas abejomis rankomis, atiduodamas kitam žaidėjui, metamas į krepšį ar prarandamas kuriuo nors kitu būdu.

Žaidėjas varymą gali tęsti jei kamuolį atidavus komandos draugui jis buvo grąžintas atgal, taip pat jei po metimo į krepšį žaidėjas pats kamuolį atsikovoja. Visus taisyklei aktualius varymo scenarijus galima supaprastinti į vieną: kamuolys yra suimamas abejomis rankomis. Tam įvykus, žaidėjas gali tik atiduoti kamuolį, mesti į krepšį arba laukti, kol pasibaigs atakai skirtas laikas.

Taisyklės pažeidimo atpažinimo algoritmas turi atpažinti, jog varymas yra sustojęs, taip pat nustatyti, kada kamuolys buvo išmestas iš rankų ir vėl pagautas (pvz. atliekant varymą antrą kartą). Paprastumo dėlei kuriant algoritmą priimta, jog situacija, kai sustabdžius varymą kamuolys yra išmetamas į orą ir vėl pagaunamas yra taisyklės pažeidimas - tokį veiksmą iš algoritminio atpažinimo perspektyvos sunku atskirti nuo pasavimo ar metimo į krepšį dėl tikslesnių apibrėžimų trūkumo bei subjektyvumo vertinant veiksmo intenciją. Panašus scenarijus apibūdinamas NBA taisyklių 13 skyriuje - žaidėjas metęs ar atidavęs kamuolį negali jo paliesti nepažeidęs taisyklės, jei kamuolio prieš tai nepalietė komandos draugas, priešininkas ar kamuolys neatsimušė nuo lentos ar lanko. Taigi, sustabdžius varymą, kamuolio pagavimas jį išmetus aukštyn ar žemyn yra laikytinas kaip pažeidžiantis taisykles.

4.4. Dvigubo varymo taisyklės atpažinimo algoritmas

Šiam algoritmui, kaip ir žingsnių taisyklės atpažinimo algoritmui, būtina sąlyga - informacija apie kojų (pėdų), kamuolio ir rankų padėtį kadre. Supaprastinta algoritmo versija atrodo taip:

1. Nuskaitomas kadras.

2. Nustatoma, ar varymas yra sustabdytas.

3. Nustatoma, ar kamuolys yra išmetamas iš rankų.

4. Jei kamuolys yra žaidėjo vėl pagaunamas, grąžinama, kad taisyklė yra pažeista.

Varymo sustabdymą indikuoja tai, jog po varymo kamuolys buvo suimtas abejomis rankomis. Tai atpažinti, jei žaidėjas yra filmuojamas iš priekio, galima atlikus Hadamardo daugybą su išskirtomis rankų ir kamuolio sritį reprezentuojančiomis matricomis (darant prielaidą, jog jų dimensijos sutampa):

$$M_{kamuolys\ rankose} = M_{kamuolys} \odot (M_{kairė\ ranka} + M_{dešinė\ ranka}). \tag{4}$$

4 lygtyje M reiškia loginę matricą, reprezentuojančią tam tikros kūno dalies sritį kadre.

Šis būdas netinkamas tada, kai žaidėjas yra filmuojamas iš šono - tuomet spalviniais bei žmogaus pozos atpažinimo metodais neįmanoma nustatyti, kur yra žaidėjo užstojama ranka, nes jos tiesiog nesimato. Tam, kad dalinai išspręsti šią problemą, galima pasinaudoti įžvalga, jog sustabdžius varymą ir kamuolį suėmus abejomis rankomis, kamuolys bent tam tikrą laiko tarpą lieka rankose. Šis laiko tarpas yra pakankamai reikšmingas, kad būtų galima jį atpažinti apdorojant kadrus vienas po kito.

Laiko tarpui, kai kamuolys laikomas rankose, nustatymui galima pasinaudoti 4.2.1. poskyriuje aprašytu būsenos saugojimo algoritmu.

Rezultatai ir išvados

Rezultatų ir išvadų dalyje išdėstomi pagrindiniai darbo rezultatai (kažkas išanalizuota, kažkas sukurta, kažkas įdiegta), toliau pateikiamos išvados (daromi nagrinėtų problemų sprendimo metodų palyginimai, siūlomos rekomendacijos, akcentuojamos naujovės). Rezultatai ir išvados pateikiami sunumeruotų (gali būti hierarchiniai) sąrašų pavidalu. Darbo rezultatai turi atitikti darbo tikslą.

Literatūra

[Ana21] Anaconda. Miniconda. https://docs.conda.io/en/latest/miniconda.html/, 2021. tikrinta 2021-04-25.

[APG+14] M. Andriluka, L. Pishchulin, P. Gehler ir B. Schiele. 2d human pose estimation: new benchmark and state of the art analysis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014-06.

[BK08] G. Bradski ir A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., USA, 2008.

[CHS+19] Z. Cao, G. Hidalgo, T. Simon, S. Wei ir Y. Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields, 2019. arXiv: 1812.08008.

[HSZ87] R. M. Haralick, S. R. Sternberg ir X. Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, 1987.

[YPI+90] H. K. Yuen, J. Princen, J. Illingworth ir J. Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77, 1990. URL: https://www.sciencedirect.com/science/article/pii/026288569090059E.

[KB02] P. KaewTraKulPong ir R. Bowden. *An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection*. Springer US, Boston, MA, 2002, p. 135–144.

[KMB07] P. Kakumanu, S. Makrogiannis ir N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40(3):1106–1122, 2007. ISSN: 0031-3203. URL: https://www.sciencedirect.com/science/article/pii/S0031320306002767.

[KPS03] J. Kovac, P. Peer ir F. Solina. 2d versus 3d colour space face detection. *Proceedings EC-VIP-MC 2003. 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications (IEEE Cat. No.03EX667)*, tom. 2, 449–454 vol.2, 2003.

[Kuh12] D. Kuhlman. *A Python Book: Beginning Python, Advanced Python, and Python Exercises*. 2012.

[LBC02] N. Li, J. Bu ir C. Chen. Real-time video object segmentation using hsv space. *Proceedings. International Conference on Image Processing*, tom. 2, p. II–II, 2002.

[NBA21] NBA. Rule no. 10: violations and penalties. https://official.nba.com/rule-no-10-violations-and-penalties/, 2021. tikrinta 2021-05-04.

[NVI21] NVIDIA. Cuda home page. https://developer.nvidia.com/cuda-zone, 2021. tikrinta 2021-04-25.

[Ope21a] OpenCV. OpenCV Modules. https://docs.opencv.org/master/, 2021. tikrinta 2021-04-25.

[Ope21b] OpenCV. OpenCV with CUDA. https://opencv.org/platforms/cuda/, 2021. tikrinta 2021-04-25.

[Oso18] D. Osokin. Real-time 2d multi-person pose estimation on cpu: lightweight openpose, 2018. arXiv: 1811.12004 [cs.CV].

[Sig19] K. Sigler. Nba referee missed calls: reasons and solutions. *The Sport Journal*, 22, 2019.

[Sze21] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer International Publishing, Base, Switzerland, 2021.

[Tor21] Torch Contributors. PyTorch Documentation. https://docs.conda.io/en/latest/miniconda.html/, 2021. tikrinta 2021-04-25.

[VS12] S. R. Vantaram ir E. Saber. Survey of contemporary trends in color image segmentation. *Journal of Electronic Imaging*, 21(4):1–28, 2012. URL: https://doi.org/10.1117/1.JEI.21.4.040901.

[WC15] Y. Wan ir Q. Chen. Joint image dehazing and contrast enhancement using the hsv color space. *2015 Visual Communications and Image Processing (VCIP)*, p. 1–4, 2015.

[Wel91] E. Welzl. Smallest enclosing disks (balls and ellipsoids). *Results and New Trends in Computer Science*, p. 359–370. Springer-Verlag, 1991.

[WY01] Y. Wang ir B. Yuan. A novel approach for human face detection from color images under complex background. *Pattern Recognition*, 34(10):1983–1992, 2001. ISSN: 0031-3203. URL: https://www.sciencedirect.com/science/article/pii/S0031320300001199.

[XJ02] Y. Xie ir Q. Ji. A new efficient ellipse detection method. *Object recognition supported by user interaction for service robots*, tom. 2, 957–960 vol.2, 2002.

Santrumpos

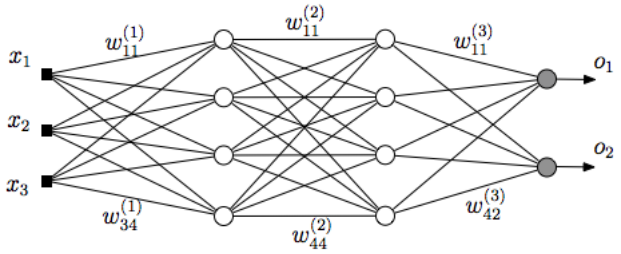
Sąvokų apibrėžimai ir santrumpų sąrašas sudaromas tada, kai darbo tekste vartojami specialūs paaiškinimo reikalaujantys terminai ir rečiau sutinkamos santrumpos.

Priedas nr. 1
Sukurtos programinės įrangos kodo nuoroda

https://github.com/LukasCed/basketball-rules-violation-thesis

Priedas nr. 2

Neuroninio tinklo struktūra



10 pav. Paveikslėlio pavyzdys

Priedas nr. 3
Eksperimentinio palyginimo rezultatai

7 lentelė. Lentelės pavyzdys

Algoritmas	$\bar{x}$	σ^2
Algoritmas A	1.6335	0.5584
Algoritmas B	1.7395	0.5647